И.Г. СЕМАКИН

БАҒДАРЛАМАЛАУ ЖӘНЕ ДЕРЕККҚОР НЕГІЗДЕРІ

ОҚУЛЫҚ

«Білім беруді дамытудың Федералды институты» Федералды мемлекеттік автономды мекемесімен «Компьютерлік желілер» мамандығы бойынша орта кәсіби білім беру бағдарламаларын іске асыратын білім беру мекемелерінің оқу процесінде оқулық ретінде пайдалану үшін ұсынылған

Рецензияның тіркеу нөмірі 307 2014ж. 26 маусым «БДФИ» ФМАМ

ACADEMA

Мәскеу
«Академия» баспа орталығы
2014

ӘОЖ 004.4(075.32)
КБЖ 32.973-018.2-ші 723
С81

Бұл кітап Қазақстан Республикасының Білім және ғылым министрлігі және «Кәсіпқор» холдингі» КЕАҚ арасында жасалған шартқа сәйкес «ТжКБ жүйесі үшін шетел әдебиетін сатып алуды және аударуды ұйымдастыру жөніндегі қызметтер» мемлекеттік тапсырмасын орындау аясында қазақ тіліне аударылды. Аталған кітаптың орыс тіліндегі нұсқасы Ресей Федерациясының білім беру үдерісіне қойылатын талаптардың ескерілуімен жасалды.

Қазақстан Республикасының техникалық және кәсіптік білім беру жүйесіндегі білім беру ұйымдарының осы жағдайды ескеруі және оқу үдерісінде мазмұнды бөлімді (технология, материалдар және қажетті ақпарат) қолдануы қажет.

Аударманы «Delta Consulting Group» ЖШС жүзеге асырды, заңды мекенжайы: Астана қ., Иманов көш., 19, «Алма-Ата» БО, 809С , телефоны: 8 (7172) 78 79 29, эл. поштасы: info@dcg.kz

Пікір берушілер:
Пермь мемлекеттік ұлттық зерттеу университетінің қолданбалы математика және информатика кафедрасының меңгерушісі, фи.-мат. ғылымдарының докторы, профессор *С. В. Русаков;*
ЭЖМ ҒЗУ-Пермь бизнестегі ақпараттық технологиялар кафедрасы-ның доценті, физ.-мат. ғылымдарының кандидаты *Л. В. Шестакова*

Семакин И. Г.

С81 Бағдарламалау және дерекқор негіздері: орта кәсіби білім беру мекемелерінің студенттеріне арналған оқулық / И.Г. Семакин. — М.: «Академия» баспа орталығы, 2014. — 224 б.

ISBN978-601-333-094-5 (каз.)
ISBN978-5-4468-0755-0 (рус.)

Оқулық 230401 «Ақпараттық жүйелер (салалар бойынша)» (ОБ 05) мамандығы бойынша орта кәсіби білім берудің Федералды мемлекеттік білім беру стандартына сәйкес құрылды.

Алгоритмдерді құрудың құрылымдық әдістеме қағидалары, Паскаль бағдарламалау тілінің негіздері, деректердің түрлі типтері мен құрылымда-рын өңдеу бойынша үлгілік бағдарламалар құрудың құралдары мен әді-стері қарастырылды. Object Pascal нұсқасы үшін объектілік-бағдарланған негізгі түсініктер баяндалған, Delphi бағдарламалау жүйесінде графикалық интерфейспен бағдарламалар әзірлеу әдістері сипатталған. Көпкестелік ре-ляциялық дерекқорды жобалау әдістемесіне ерекше көңіл бөлінген. Сұра-ныстарды сипаттау үшін SQL тілі, сондай-ақ СУБД MS Access сұраныста-рының Конструктор нұсқасында QBE тілі қолданылады.

Орта кәсіби білім беру мекемелерінің студенттеріне арналған.

ӘОЖ 004.4(075.32)
КБЖ 32.973-018.2-ші723

ISBN978-601-333-094-5 (каз.)
ISBN 978-5-4468 0755-0 (рус.)

Құрметті оқырман!

Берілген оқулық 230401 «Ақпараттық жүйелер (салалар бойынша)» мамандығы үшін оқу-әдістемелік жиынтықтың бөлігі болып табылады.

Оқулық «Бағдарламалау және дерекқор негіздері» (ОБ 05) жалпы кәсіби пәнді оқуға арналған.

Жаңа ұрпақтың оқу-әдістемелік жиынтықтарына жалпы білім беру және жалпы кәсіби пәндер мен кәсіби модульдерді оқуды қамтамасыз ететін, дәстүрлі және инновациялық оқу материалдары кіреді. Әрбір жиынтыққа жалпы және кәсіби құзыреттерді, соның ішінде жұмыс берушінің талаптарын ескере отырып, меңгеруге қажетті оқулықтар мен оқу құралдары, оқыту және бақылау құрал-жабдықтары кіреді.

Оқу басылымдары электронды білім беру ресурстерімен толықтырылады. Электронды ресурстерде интербелсенді жаттығулар мен жаттығу құрылғылармен берілген теориялық және практикалық модульдер, мультимедиялық объектілер, Интернеттегі қосымша материалдар мен қорларға сілтемелер берілген. Оларға оқу процесінің негізгі параметрлері: жұмыс уақыты, бақылау және практикалық жұмыстарды орындау нәтижелері белгіленетін терминологиялық сөздік пен электронды журнал енгізілген. Электронды ресурстер оқу процесіне оңай кіріктіріледі және түрлі оқу бағдарламаларына бейімделуі мүмкін.

Алғысөз

Бағдарламалау көптеген жағдайларда тек кәсіпқойларға арналған кәсіпке айналуда. 1980 жылдардың ортасында жарияланған «Бағдарламалау - екінші сауаттылық» атты ұран артта қалды. «Компьютерлік сауаттылық» түсінігіне бүгінгі таңда ең алдымен ақпараттық технологиялардың алуан түрлі құралдарын қолдану дағдысы кіреді. Қандай да бір ақпараттық есепті шеше отырып, сәйкес бағдарламалық құралды таңдау қажет. Бұл электронды кестелер, дерекқорды басқару жүйелері, математикалық пакеттер және басқалары болуы мүмкін. Мұндай құралдар есепті шешуге мүмкіндік бере алмаған жағдайда ғана бағдарламалаудың әмбебап тілдеріне жүгінген жөн.

Бағдарламашылардың екі түрлі категориясы бар: жүйелік және қолданбалы. Жүйелік бағдарламашылар ЭЕМ негізгі бағдарламалық құралдарын (операциялық жүйелерді, трансляторларды, сервистік құралдарды және с.с.) әзірлеушілер, бұлар бағдарламалаудағы жоғары дәрежелі мамандар. Қолданбалы бағдарламашылар түрлі саладағы (ғылым, техника, өндіріс, ЭЕМ қолданбалы бағдарламалық қамсыздандыру құралдарын әзірлейді.

Қолданбалы бағдарламалардың да, жүйеліктің де сапасына, қойылатын талаптар қазіргі кезде өте жоғары. Бағдарлама есепті дұрыс шешіп қана қоймай, сонымен қатар заманауи интерфейске ие болуы, өте сенімді болуы, пайдаланушыға жылы шыраймен қарауы және т.с.с. тиіс. Тек осындай ғана бағдарламалар бағдарламалық өнімдердің әлемдік нарығындағы бәсегеге төтеп бере алады.

Компьютерлік техниканың дамуымен бірге бағдарламалаудың

әдістемесі мен технологиясы да дамыды. Алдымен командалық және операторлық бағдарламалау пайда болды, 1960 жылдары құрылымдық бағдарламалау жете дамыды, логикалық және функционалдық бағдарламалау желілері, ал соңғы уақытта - объектілі-бағдарланған және визуалды бағдарламалау пайда болды.

Бағдарламалауды бастапқы үйренуде алдыға қойылатын бірінші міндет – алгоритмдік ойлау қабілетін дамыту, негізгі алгоритмдік құрылымдармен: жолымен, тармақталуымен, кезеңімен, бағдарламалаудың құрылымдық әдістемесін меңгерумен танысу. Бұл мақсатқа осы оқу құралының 1- тарауы арналған.

Берілген курсты оқуға дайындық ретінде оқушылардың алгоритмдеу негіздерін информатиканы мектептің базалық курсы аясында меңгергені абзал. Әдетте мектепте алгоритмдеу құрылымдық әдістеме негіздерін сәтті меңгеруге септігін тигізетін оқу атқарушыларын қолдану арқылы оқытылады:

- алгоритмдерді негізгі құрылымдардан құру;
- жүйелі егжей-тегжейін ашу әдісін қолдану.

Алғашқы оқу үшін ең тиімді сәйкес келетін құрал ретінде Паскаль бағдарламалау тілі болады. Бұл бағдарламалаудың ең танымал тілдерінің бірі; ол бірқатар тілдер үшін негізгі болып саналады. Паскаль тілінің авторы – швейцарлық профессор Никлаус Вирт. Құрылымдық әдістеме бағдарламалық мәдениеттің негізі болып қалады. Бағдарламалауды оқуға бет бұрған адамның оны меңгермей, кәсіпқой маман болуға ешқандай мүмкіндігі жоқ.

2-тараудың мазмұны оқушылардың жоғары деңгейлі бағдарламалау тілдерінің негізгі түсініктерін,оларды Паскаль тілінде жүзеге асыруда меңгеруге бағытталған. Мұндай дайындық болашақта бағдарламалаудың басқа да тілдерін үйренуді жеңілдетеді.

3-тарауда мысалдар келтіре отырып, Паскаль тілінде жүзеге асырып, объектілі-бағдарланған бағдарламалаудың негіздері баяндалады. Осы тарауда визуалды бағдарламалау технологиясы жүзеге асырылған, Паскаль тілін объектілік-бағдарланған кеңейту ретінде Delphi бағдарламалау тілі қарастырылады.

Берілген кітап – бұл ең алдымен Паскаль және Delphi тілдері бойынша емес, бағдарламалау бойынша оқу құралы екенін атап өтеміз, сондықтан берілген тілдерді егжей-тегжейлі сипаттауды бұл кітаптан таппайсыз. Тілдерді сипаттау бағдарламалаудың бастапқы курсына қажетті көлемде ғана берілген. Тілдерді толығырақ сипаттауды әдебиеттер тізімінде берілген кітаптардан табуға болады.

Оқу құралында үйренетін тілдер үшін бағдарламалаудың нақты жүйелерімен жұмыс жасау бойынша берілген нұсқаулар жоқ. Оқушылар олармен басқа дереккөздерді қолдана отырып, ЭЕМ практика жүзінде танысулары тиіс.

4-тарау «Дерекқор теориясының негіздері» деп аталады. Дерекқор - кез келген ақпараттақ жүйенің негізі. Ақпараттық технологиялардың бұл бағытының қаншалықты маңызды екендігін жоғары білім берудің бірсыпыра мамандықтарының автоматтандырылған ақпараттық жүйелер бойынша мамандарды дайындаумен тікелей байланысты болуынан көруге болады.

Дәстүрлерді дәріптей отырып, ақпараттық жүйелер ретінде дерекқорларда ақпаратты сақтауға арналған және оны енгізу мен сұраныстарға жауап дайындауды қоса алғанда, онымен түрлі амалдар жасауды қамтамасыз ететін жүйелерді айтуға болады. Ақпараттық жүйелерге ақпараттық-анықтамалық жүйелерді, құжатайналымды автоматтандыратын жүйелерді, автоматтандырылған жобалау жүйелерін және басқаларын жатқызады.

Информатика курсы шегінде сан түрлі пән салаларында қолданылуы мүмкін болатын жүйелерді құру әдістері, дерекқорда сақталатын ақпаратты нысандандыру және жүйелендіру оқытылады.

1 тарау

АЛГОРИТМДЕРДІ ҚҰРУ ҚАҒИДАЛАРЫ ЖӘНЕ АЛГОРИТМДІК ҚҰРАСТЫРЫЛЫМДАР

1.1. АЛГОРИТМДЕР ЖӘНЕ ШАМАЛАР

ЭЕМ-да есепті шешу кезеңдері. Компьютерді пайдалану арқылы кез келген есепті шығару бойынша жұмысқа алты кезең кіреді:

1) есептің қойылуы;
2) есепті нысандандыру;
3) алгоритмнің қойылуы;
4) бағдарламалау тілінде бағдарлама құру;
5) бағдарламаны ретке келтіру және тестілеу;
6) есептеулер жүргізу және алынған нәтижелерді талдау.

Көбіне бұл бірізділікті *ЭЕМ-да есепті шешудің технологиялық тізбегі* деп атайды (бұл тізімнен бағдарламалауға тікелей қатысы бар 3-5 тармақтар).

Есептің қойылуы кезеңінде *не берілгенін* және *нені табу керектігін* дәл анықтау қажет. Есепті шешуге қажетті алғашқы деректердің толық жиынтығын сипаттау маңызды.

Нысандандыру кезеңінде көбіне есеп математикалық формулалар, теңдеулер және қатынастар тіліне көшіріледі. Егеер есепті шешу қандай да бір нақты объектіні, құбылысты немесе үрдісті сипаттауды қажет ететін болса, оны нысандандыру тиісті математикалық үлгіні алуға тең болады.

Үшінші кезең — алгоритмді құру кезеңі. Тәжірибелі бағдарламашылар көбіне алгоритмдерді сипаттаудың қандай да бір арнайы құралдарына (блок-сызбаларға, жалған кодтарға) жүгінбей,бағдарламаны бірден белгілі бір тілде жазады, алайда оқу мақсатында бұл құралдарды алдымен қолданып, содан соң алынған алгоритмді бағдарламалау тіліне көшіру тиімді.

Алғашқы үш кезең - бұл компьютерсіз жұмыс жасау. Содан кейінгі екі кезең – бұл шын мәнінде, белгілі бір бағдарламалау жүйесінде белгілі бір тілде бағдарламалау. Соңғы – алтыншы кезеңде әзірленген бағдарлама енді практикалық мақсаттарда қолданылады.

Осылайша, бағдарламашы алгоритмдерді құра алуы, бағдарламалау тілдерін білуі, тиісті бағдарламалау жүйесінде жұмыс жасай білуі тиіс. Бағдарламашының кәсіби сауаттылығының негізі ретінде дамыған алгоритмдік ойлау қабілеті болады.

Алгоритм түсінігі. Информатикада іргелі түсініктердің бірі ретінде алгоритм түсінігін атауға болады. Математикадан алынған «алгоритм» түсінігі орта ғасырлық Шығыстың ғұлама математигі Мұхаммед әл-Хорезмидің (787 — 850) атының латынша жазылуы algorithmi сөзінен алынған. XII ғасырда оның математикалық трактатының латындық аудармасы жүзеге асырылды, одан еуропалықтар есептеп шығарудың ондық позициялық жүйесі туралы және көп таңбалы сандардың арифметикасының ережелерімен танысты. Дәл осы ережелерді сол уақытта алгоритмдер деп атаған. Қосу, алу, «бағанмен» көбейту, «бұрыштап» бөлу – бұлар математикадағы алғашқы алгоритмдер. Алгебралық қайта құру ережелері мен теңдеу түбірін есептеуді математикалық алгоритмдерге жатқызуға болады.

Біздің заманымызда алгоритм түсінігі кеңірек ұғымда беріледі. Алгоритм — бұл қандай да бір атқарушыменбасқару командадерінің жүйелігі. Информатиканың мектеп курсында оқушылар алгоритм түсінігімен және алгоритмдерді құру әдістерімен оқу атқарушыларының: Роботтың, Тасбақаның, Сызушының және басқалардың мысалында танысады. Бұл атқарушылар ештеңе де есептемейді. Олар экранда суреттер салады, шытырмандарда жүріп-тұрады, заттарды бір жерден екінші жерге апарып қойды. Мұндай атқарушыларды ахуалда жұмыс істейтін атқарушылар деп атау қалыптасқан.

Информатиканың «Бағдарламалау» тарауында ЭЕМ жұмысын бағдарламалық басқару әдістері оқытылады,яғни атқарушы ретінде компьютер алынады. Компьютер шамалармен жұмыс жасайтын алгоритмдер деп аталатын, компьютерді басқаруға арналған шамалармен – түрлі ақпараттық объектілермен: сандармен, таңбалармен, кодтармен және басқалармен жұмыс жасайды.

Деректер және шамалар. Компьютер жұмыс жасайтын шамалардың жиынтығын деректер деп атау қалыптасқан. Бағдарламаға қатысы бойынша есептеулер процесінде алынатын алғашқы, соңғы (нәтижелер) және аралық деректер бар (1.1-сурет).

1.1. – сурет. Бағдарламаға қатысты деректердің деңгейлері

Мысалы, $ax^2 + bx + c = 0$ квадрат теңдеуін шешу кезінде алғашқы деректер ретіндеа, b, скоэффициенттері,нәтижелер — x_1, x_2, теңдеуінің түбірлері, аралық деректер - $D = b2 - 4ac$ теңдеуінің дискриминанты болады.

Бағдарламалауды ойдағыдай меңгеру үшін келесі ережені білу керек: *кез келген шаманың ЭЕМ жадында өзінің белгілі бір орны болады, кейде оны жады ұяшығы деп те атайды.* «Ұяшық» термині заманауи ЭЕМ сәулеті үшін біршама ескірген, алайда оқу мақсатында оны пайдаланған өте қолайлы.

Кез келген шаманың үш негізгі қасиеті бар: *атауы, мәні* және *түрі*. Процессор командасы деңгейінде шама өзі сақталатын жады ұяшығының мекен-жайымен сәйкестендіріледі. Алгоритмдер және бағдарламалау тілдерінде шамалар *константаларға* және *ауыспалы шамаларға* бөлінеді. Константа - өзгермейтін шама, және алгоритмде ол өздік мәні бойынша беріледі, мысалы: 15, 34.7’, k’, True және т.б. Ауыспалы шамалар өз мәндерін бағдарламаны орындау барысында өзгертулері мүмкін және алгоритмде символдық аттармен – сәйкестендіргіштермен, мысалы: X, S2, cod15 және т.б. беріледі. Кез келген константалар мен айнымалы шамалар жадының ұяшығына ие болады, ал бұл шамалардың мәндері бұл ұяшықта екіліккодпен анықталады.

Енді информатика курсында дерекқорлар мен электронды кестелерді оқығанда кездесетін түсінік - *деректер түрлері* - шама түрлері туралы айтып өтейік. Бұл түсінік бағдарламалауда іргелі болып табылады.

Бағдарламалаудың әрбір тілінде өз тұжырымдамасы және өз деректер түрінің жүйесі болады. Алайда кез келген тілге ең минималды қажетті деректер түрінің: *тұтас, заттық, логикалық* және *таңбалық* жиынтығы кіреді. Шама түрімен оның үш қасиеті байланысты: шектелген мәндердің жиыны, шектелген операциялардың жиыны, ішкі елестету нысаны. Деректер түрінің негізгі қасиеттері 1.1.кестеде берілген.

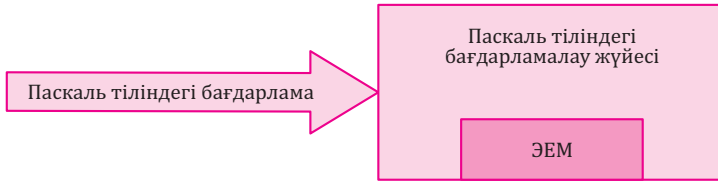
Константа түрлері мәнмәтін бойынша (яғни мәтіндегі жазба нысаны бойынша), ал айнымалы түрлері ауыспалы шаманың сипатталуында белгіленеді.

1.1.-кесте.Деректердің негізгі түрлері

Түрі	Мәні	Операциясы	Ішкі елестету
Тұтас	Кейбір диапазондағы тұтас оң және теріс сандар, мысалы: 23, -12, 387	Тұтас сандармен жүргізілетін арифметикалық операциялар: қосу, азайту, көбейту және қалдықпен бөлу. Қатынастар опера-циялары (<, >, = және т.б.)	Белгіленген нүктемен берілген пішін
Заттық	Кейбір диапазондағы кез келген сандар (тұтас және бөлшек), мысалы: 2.5, - 0.01, 45.0, 3.6×10^9	Арифметикалық операциялар. Қатынастар операциялары	Қалқымалы нүктемен берілген пішін
Логикалық	True (ақиқат) False (жалған)	Логикалық операциялар: ЖӘНЕ(and), НЕМЕСЕ(or), ЖОҚ(not). Қатынастар операциялары	1 — True; 0 — False
Символдық	Компьютерлік алфавиттің кез келген таңбалары, мысалы: a, 5, +, $	Қатынастар операциялары	Таңбалық кодтаудың кодтары, мысалы:ASCII— бір таңба — 1 байт; Unicode — бір таңба — 2 байт

Құрылымы бойынша деректер қарапайым және құрылымдандырылған болып бөлінеді. Скалярлық деп те аталатын қарапайым шамалар үшін бір шама – бір мән, ал құрылымдандырылған шамалар үшін – бір мән – мәндер жиыны деген пайымдау дұрыс. Құрылымдандырылған шамаларға ауқымдар, жолдар, жиындар және т.б. жатады.

ЭЕМ — алгоритмдерді атқарушы. Әрбір алгоритм (бағдарлама) нақты бір атқарушыға арналып, яғни оның командалар жүйесінің шегінде құрылатыны мәлім. «ЭЕМ арналған бағдарламалау» тақырыбымен танысу кезінде қандай атқарушы туралы сөз болады? Жауап белгілі:



1.2. –сурет. Компьютер бағдарламаны Паскаль тілінде атқарушы ретінде

Бұл жерде атқарушы ретінде компьютер болады, дәлірек айтқанда ЭЕМ кешені + бағдарламалау жүйесі. Бағдарламашы бағдарламаны бағдарламалау жүйесі бағдарланған тілде жасайды. Бұл сызба түрінде 1.2-суретінде көрсетілген, онда атқарушының ену тілі ретінде Паскаль бағдарламалау тілі болады.

Бағдарлама бағдарламалаудың қандай тілінде жазылатындығына қарамастан, ЭЕМ-да кез келген есепті шешу алгоритмі келесі командалардан құрылуы мүмкін: атау беру, қосу, шығару, көмекші алгоритмге жүгіну, цикл, тармақталу.

Бұдан әрі алгоритмдерді сипаттау үшін блок-схемалар және информатиканың мектептік курсында енгізілетін алгоритмдік тіл (АТ) қолданылады.

1.2. СЫЗЫҚТЫҚ ЕСЕПТЕУ АЛГОРИТМДЕРІ

Есептеу алгоритмдерінде негізгі қарапайым амал – *ауыспалы шаманың мәнін меншіктену* болып табылады.

Егер константаның мәні оның жазба түрімен анықталатын болса, ауыспалы шама нақты мәнге екі түрлі тәсілмен: меншіктену командасының көмегімен және қосу командасының көмегімен жүзеге асырылатын атау беру нәтижесінде жетеді.

Мысал ретінде қарастырайық. Мектептегі математика пәнінде екі қарапайым бөлшекті бөлу ережесі келесідей берілген:

1) бірінші бөлшектің алымын екінші бөлшектің бөлгішіне көбейту;

2) бірінші бөлшектіңбөлгішін екінші бөлшектің алымына көбейту;

3) бөлшекті жазу, оның алымы 1-т. орындаудың нәтижесі, ал бөлгіші – 2-т. орындау нәтижесі болады.

Осы мысалды жазудың алгебралық формасы мынадай:

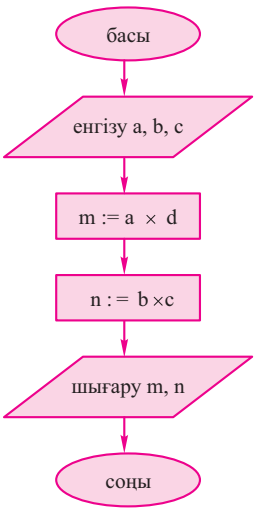
$$\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c} = \frac{m}{n}.$$

Енді алгебралық өрнекте қолданылған айнымалылардың сол белгілерін қалдырып, ЭЕМ арналған бөлшектерді бөлу алгоритмін құрайық.

Бұл жерде алғашқы деректер ретінде *a, b, c, d* тұтас сандықа-уыспалы шамалар,ал нәтижесі — *m* және *n* тұтас шамалар болады. Бөлшектерді бөлу алгоритмінің блок-сызбасы 1.3.-суретте берілген. Берілген алгоритм оқу алгоритмдік тілде келесі түрде беріледі:

алг бөлшекті бөлу
бас **тұт** a, b, c, d, m, n
қосу a, b, c, d
m := a x d
n := b x c
шығ. m, n
соң

Меншіктену командасының пішіні келесідей: ауыспалы := өрнек «:=» таңбасын «меншіктену» деп оқу керек.



Меншіктену командасы компьютер атқаратын келесі әрекеттерді білдіреді:

1) өрнек есептеледі;

2) алынған мән айнымалыға беріледі..

Берілген алгоритмде меншіктенудің екі командасы бар. Блок-схемаларда меншіктену командалары төртбұрыштарда бейнеленеді. Мұндай блок есептеу блогы деп аталады.

1.3.–сурет. Бөлшекті бөлу алгоритмінің блок-схемасы

Алгоритмдерді сипаттауда өрнектерді жазудың қатаң ережелерінің сақталуы міндетті емес, мұны қарапайым математиалық формада жасауға болады, өйткені бұл әлі қатаң синтаксиспен берілген бағдарламалау тілі емес.

Қарастырылып отырған алгоритмде енгізу командасы бар:

енгізу a, b, c, d

Блок-схемаларда енгізу командасы параллелограммда – енгізу-шығару блогында жазылады. Бұл команданы атқару кезінде процессор жұмысты тоқтатады және қолданушының әрекеттерін күтеді. Қолданушы енгізу құрылғысында (пернетақтада) енгізілетін айнымалылардың мәндерін теруі және <Enter> енгізу батырмасын басуы тиіс. Мәндерді осы айнымалылар енгізу тізімінде орналасқан тәртіп бойынша енгізген жөн. Әдетте енгізу командасының көмегімен бастапқы деректердің мәндері беріледі, ал меншіктеу командасы аралық және соңғы шамаларды алу үшін қолданылады.

Компьютермен алынған есепті шешу нәтижелері қолданушыға хабарлануы тиіс, ол үшін шығару командасы қолданылады:

шығару m, n

Осы команданың көмегімен нәтижелер экрнаға немесе басу құрылғысының көмегімен қағазға шығарылады.

Меншіктеу есептеу алгоритмдерінде маңызды операция болғандықтан, ол туралы толығырақ тоқталайық.

1.2. –кесте. Меншіктеу командасын атқару кезінде айнымалының мәнінің өзгеруі

Команда	a	b
a := 1	1	—
b := 2 x a	1	2
a := b	2	2
b := a + b	2	4

Бұл мысал меншіктеудің үш негізгі қасиетін бейнелейді:

1) айнымалыға мән берілмегенше, ол анықталмаған болып қалады;

2) айнымалыға берілген мән осы айнымалыға келесі меншіктеу орындалғанға дейін сақталады;

3) айнымалыға берілген жаңа мән оның алдыңғы мәнін ауыстырады.

Бағдарламалау кезінде жиі қолданылатын алгоритмді қарастырайық. Екі шама: X және Y берілген. Олардың арасындағы мәндерді ауыстыру қажет. Мысалы, егер алдымен $X = 1$, $Y = 2$ болса, ауыстырған соң $X = 2$, $Y = 1$ болуы тиіс.

Бұл есепке келесі жағдай ұқсас. Екі стақан берілген: біреуінде – сүт, екіншісінде – су бар. Олардың арасында ішіндегілерінің орнын ауыстыру керек. Бұл жағдайда үшінші – бос стақан қажет екені мәлім. Ауыстыру кезіндегі әрекет ету тәртібі келесідей болады:

1) сүтті 1-ші стақаннан 3-ге құю;

2) суды 2-ші стақаннан 1-ге құю;

3) сүтті 3-ші стақаннан 2-ге құю.

Осыған ұқсас екі айнымалының мәндерін ауыстыру үшін үшінші қосымша айнымалы қажет. Оны Z деп атайық. Сонда ғана мәндерді ауыстыру есебін меншіктеудің үш командасын жүйелі атқару арқылы шешуге болады (1.3-кесте).

Стакандармен берілген мысал айнымалылар арасындағы ауыстыру жағдайын соншалықты дәл сипаттамайды, өйткені сұйықтықтарды құйғанда стақандардың бірі бол болады. Меншіктеу командасын атқару нәтижесінде ($X := Y$) оң жақта тұрған айнымалы (Y) өз мәнін сақтайды.

Бөлшектерді бөлу алгоритмі сызықтық құрылымға ие болады, яғни онда барлық командалар әрқайсысы бір мәртеден, қатаң белгілі бір жүйелікпен атқарылады. Сызықтық алгоритм меншіктеу,

1.3.-кесте. Айнымалылар арасында мәндермен алмасу

Команда	X	Y	Z
енгізу X, Y	1	2	—
Z:= X	1	2	1
X := Y	2	2	1
Y:= Z	2	1	1

енгізу, шығару командаларынан және көмекші алгоритмдерге жүгінуден тұрады (ол бұдан әрі қарай қарастырылады).

Блок-схемалардың көмегімен алгоритмдерді бейнелеу кезінде деректер түрі, әдетте көрсетілмейді (бірақ тұспалданады). Оқу алгоритмдерінде (АТ-де) барлық айнымалылар үшін деректер түрлері айқын көрсетіледі, және оларды суреттеу алгоритмнің тақырыпатынан кейін бірден жүргізіледі. Бұл ретте келесі белгіленулер қолданылады: **тұт**– тұтас, **зат** – заттық, **лит** — символдық (литерлік), **лог** — логикалық. Бөлшектерді бөлу алгоритмінде барлық айнымалылар тұтас типті болады.

1.3. ЕСЕПТЕУ АЛГОРИТМДЕРІНДЕГІ ТАРМАҚТАЛУ ЖӘНЕ ЦИКЛДЕР

Квадрат теңдеуді шешудің алгаритмін құрайық.

ax2+bx+c=0

Бұл есеп математикадан жақсы таныс. Бұл жерде бастапқы деректер ретінде *a, b, c* , ал жалпы жағдайда шешуі – келесі формула бойынша есептелетін екі түбір (x^1 және x^2) болады:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Бұл бағдарламада қолданылатын барлық шамалар заттық түрде. Квадрат теңдеуді шешу алгоритмі келесідей болады:

алг. квадрат теңдеудің түбірлері
зат. a,b, c, d,x1, x2
бас. енгізу a, b,c
d:= b^2 - 4ac
x1:=(-b + √d) /(2a)
x2:=(-b - √d) /(2a)
шығару x1, x2
соңы

Мұндай алгоритмнің кемшілігі бір көргеннен-ақ байқалады. Ол сапалы алгоритмдерге қойылатын маңызды қасиетке: бастапқы деректерге қатысты жан-жақтылыққа ие болмайды. *Бастапқы деректердің мәндері қандай болғанына қарамастан, алгоритм белгілі бір нәтижеге әкелуі және соңына дейін атқарылуы тиіс.*

Нәтиже ретінде сандық жауап болуы, сонымен бірге мұндай деректері бар есептің шешуі жоқ жайындағы хабарлама да болуы мүмкін. Алгоритмнің орта тұсында қандай да бір операцияны атқару мүмкіндігінің болмауынан тоқтатуға жол берілмейді. Бұл қасиетті бағдарламалау бойынша әдебиетте алгоритмнің нәтижелігі деп атайды (кез келген жағдайда қандай да бір нәтиже алу).

Жан-жақты алгоритмді құру үшін алдымен есептің математикалық мазмұнын мұқият талдап алу қажет.

Квадрат теңдеудің шешуі *a, b, c* коэффициенттерінің мәндеріне байланысты болады.

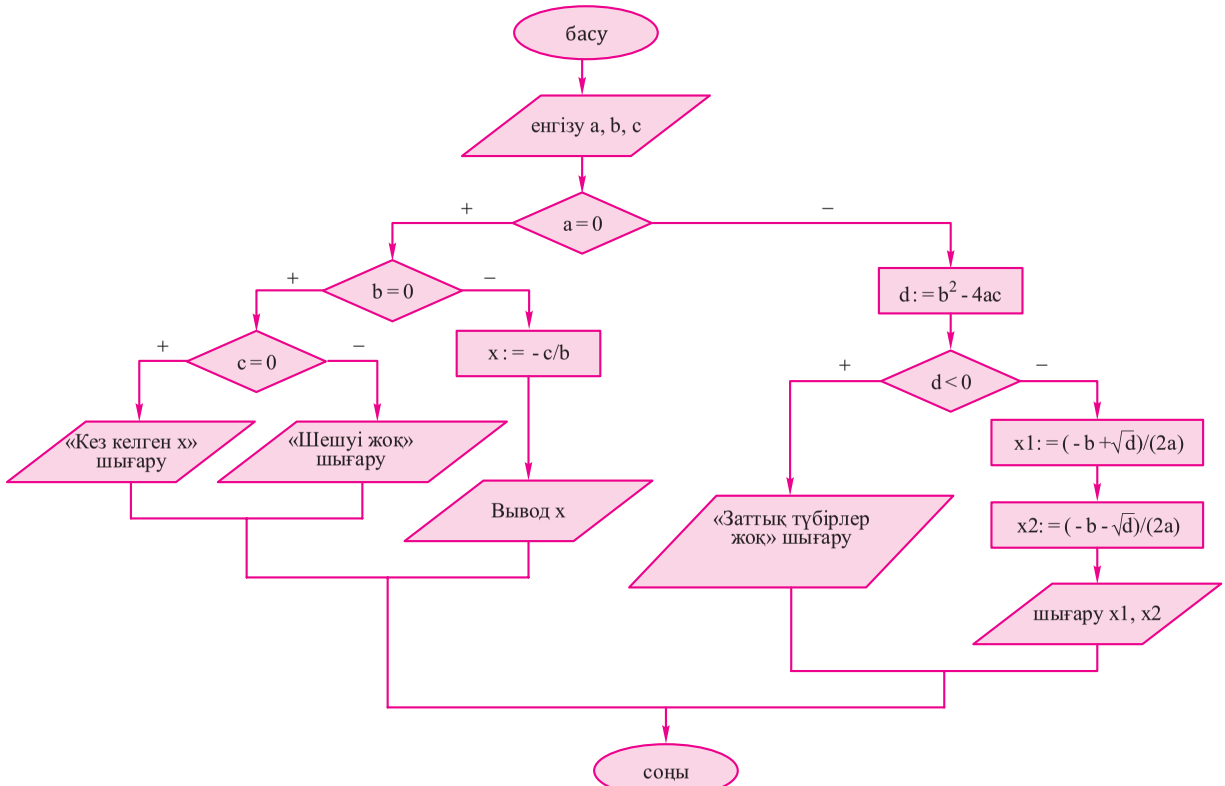
Осы есепті тек заттық түбірлерді іздестірумен шектеліп, талдайық:

- егер $a = 0$, $b = 0$, $c = 0$, x кез келген мәні — теңдеудің шешуі;
- егер $a = 0$, $b = 0$, $c \neq 0$, теңдеудің шешуі жоқ;
- егер $a = 0$, $b \neq 0$, шешуі $x = -c/b$ болатын сызықтық теңдеу;
- егер $a \neq 0$ және $d = b^2 - 4ac \geq 0$, теңдеудің екі заттық түбірі
- x_1, x_2 бар;
- егер $a \neq 0$ және $d < 0$, теңдеудің заттық түбірлері жоқ.

Квадрат теңдеуді шешу алгоритмінің блок-схемасы 1.4.-суретте көрсетілген.

Дәл осы алгоритм АТ-де келесідей беріледі:

```
алг  Квадрат теңдеудің түбірі
зат  a, b, c,  d, x1,   x2
бас  енгізу a, b, c
     егер     a = 0
     онда егер b = 0
          онда   егер c = 0
                 Онда «кез келген x — шешуі» енгізу
                 басқаша «шешуі жоқ» енгізу
                 кв
           басқаша x := - c / b
                 шығару x
                 кв
           басқаша   d := b2  - 4ac
                 егер  d < 0
                 онда «зат. түбірі жоқ шығару
```



1.4.-сурет. Квадрат теңдеуді шешу алгоритмінің блок-схемасы

басқаша x1 := (- b +√d)/(2a); x2 := (- b -√d)/(2a)
шығару «x1 =», x1, «x2 =», x2
кв
кв
соңы

Берілген алгоритмде жалпы көрінісі блок-схема түрінде 1.5.-суретте көрсетілген, тармақталудың құрылымдық командасы бірнеше рет қолданылған.

АТ-де тармақталу командасын келесі түрде жазуға болады:
егер шарты
онда 1 серия
басқ. 2 серия
кв

Берілген тармақталу командалардың блок-схемасына сәйкес алдымен шарты тексеріледі (логикалық өрнек есептеледі). Егер шарты дұрыс болса, онда «Иә» (оң тармақ) – «1 Серия» жазуы бар стрелкамен көрсетілген командалар жүйелігі атқарылады. Кері жағдайда «2 Серия» - командалардың теріс тармағы орындалады.

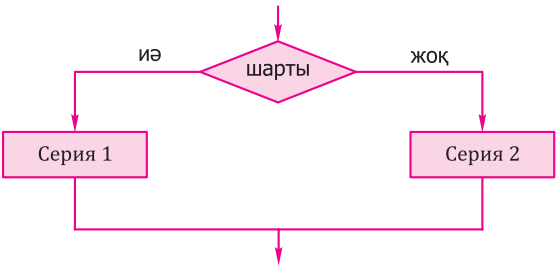
АТ-де шарты «егер» көмекші сөзінен кейін, оң тармақ – «онда» сөзінен кейін, ал теріс – «басқаша» сөзінен кейін жазылады. Мұндай жазбадағы «кв» әріптерімен тармақталудың соңын белгілейді.

Егер бір тармақталудың тармақтарында басқа тармақтар болған жағдайда, онда алгоритм *енгізілген тармақталу* құрылымына ие болады. Квадрат теңдеуді шешудегі алгоритмнің дәл осындай құрылымы бар (1.4.-суретті қараңыз), онда қысқарту үшін «Иә» және «Жоқ» сөздерінің орнына сәйкесінше «+» және «-» таңбалары қолданылған.

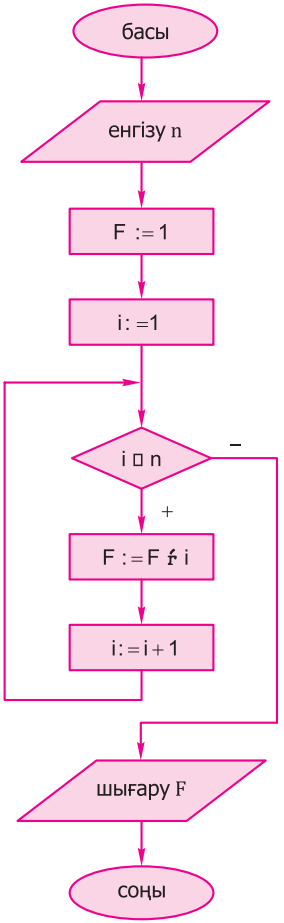
Келесі есепті қарастырайық: n тұтас оң сан берілген. n! (n-факториал) есептеу керек. Факториалдың анықтамасын еске түсірейік:

$$n! = \begin{cases} 1, & \text{егер } n = 0; \\ 1 \times 2 \times \ldots n, & \text{егер } n > 1 \end{cases}$$

1.6-суретте тұтас типтегі үш айнымалысы бар n! есептеу алгоритмінің блок-схемасы көрсетілген: n — аргумент, i— аралық айнымалы, F— нәтиже. Берілген алоритмнің дұрыстығын тексеру үшін



1.5.-сурет. Тармақталу командасының блок-схемасы



1.6.-сурет. *n*! есептеу алгоритмінің блок-схемасы

1.4 жол тарту кестесін құрайық, онда қадамдар бойынша бастапқы деректердің нақты мәндері үшін алгоритмге кіретін айнымалылардың өзгеруі қадағаланады. Берілген кесте *n*= 3 жағдайы үшін құрылған.

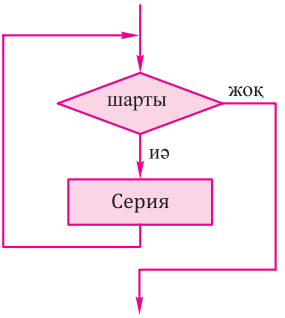
Берілген жол тарту қарастырылып отырған алгоритмнің дұрыстығын дәлелдейді. Енді осы алгоритмді АТ-де жазайық:

```
алг     факториал
тұт     n, i, F
бас     енгізу n
        F := 1;  i := 1
        әзірше i ≤   n, қайталау
        цб F := F x i
        i := i + 1
        цс
шығаруF
соң
```

1.4. –кесте. n! алгоритмін есептеудің жол тартуы

қадам	n	F	i	Шарты
1	3			
2		1		
3			1	
4				1 < 3, иә
5		1		
6			2	
7				2 < 3, иә
8		2		
9			3	
10				3 < 3, иә
11		6		
12			4	
13				4 < 3, жоқ
14		шығару		

1.7-сурет. «Әзірше» циклі командасының блок-схемасы



Берілген алгоритмнің циклдік құрылымы бар. Онда «Әзірше» циклдердің немесе алғышартпен берілген циклдіңқұрылымдық командасы қолданылған. «Әзірше» циклі командасының блок-схемасы 1.7.-суретте көрсетілген. АТ-де ол былай беріледі:

әзірше шарт, **қайталау**
цб
серия
цс

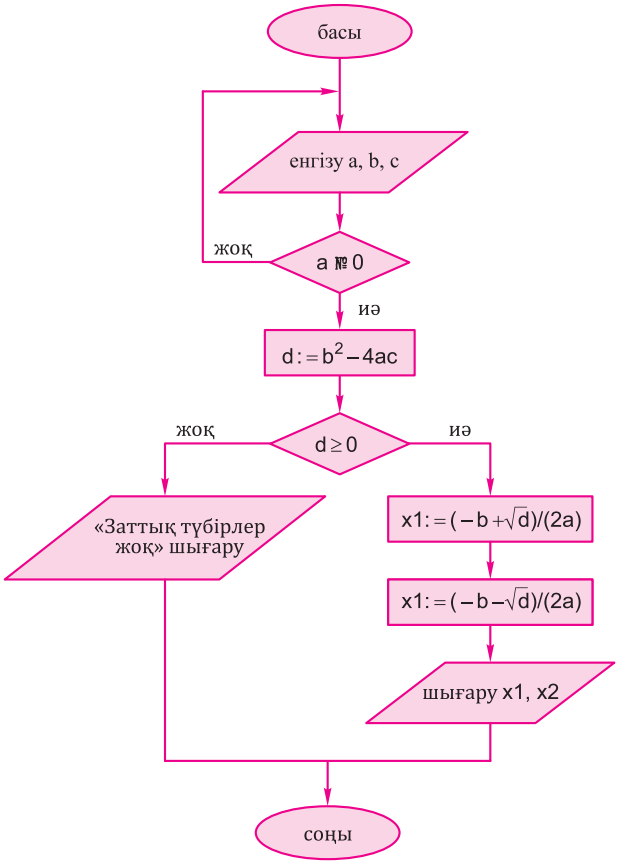
Командалар сериясын атқару (циклдің денесі) циклдің шарты ақиқат болып тұрғанша қайталанады. Шарт жалған болған кезде, циклді орындау аяқталады. «цб» және «цс» көмекші сөздер сәйкесінше циклдің басын және соңын білдіреді.

Алғышартпен берілген цикл – бұл циклдік алгоритмдерді ұйымдастырудың негізгі, бірақжалғыз емес нысаны: шарттан кейінгі циклдер де бар.

Квадрат теңдеуді шешу алгоритміне, оны: егер $a = 0$ - бұл квадрат теңдеу емес және оны шешу мүмкін емес позициясынан қарастыра отырып, оралайық. Бұл жағдайда қолданушы деректерді енгізуде қателесті және енгізуді қайталау керек деп санайық (1.8-сурет). Басқа сөзбен айтқанда, қолданушыға қатені түзету мүмкіндігін бере отырып, алгоритмде бастапқы деректердің ақиқатлығын бақылау қарастырылады. Мұндай бақылаудың болуы – бағдарламаның сапасының жоғарылығының тағы да бір дәлелі болады.

АТ-де деректерді енгізуді бақылаумен берілген квадрат теңдеуді шешу алгоритмі келесі көріністе беріледі:

алг квадрат теңдеу
зат a, b, c, d, x1, x2
бас
қайталау
енгізу a, b, c
a ≠ 0 **дейін**
$d := b^2 - 4ac$
егер $d \geq 0$



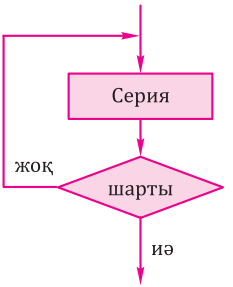
1.8.-сурет. Бақылаумен берілген квадрат теңдеуді шешу алгоритмінің блок-схемасы

онда x1 := (-b + √d)/(2a)
x2 := (-b -√d)/(2a)
шығару x1, x2
басқаша
шығару «Заттық түбірлер жоқ»
кв
соңы

«Дейін» циклінің шарттан кейінгі цикл құрылымдық командасының блок-схемасы 1.9.-суретте көрсетілген. АТ осы команданы былайша жазуға болады:

1.9.- сурет. «Дейін» циклі командасының блок-сызбасы

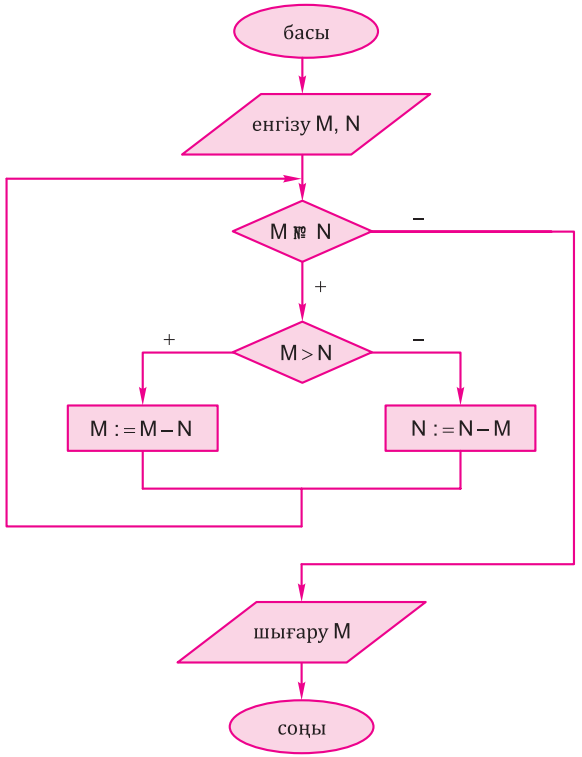
қайталау
серия
дейін шарты



Бұл жерде циклдің соңындағы шарт қолданылады, яғни ол ақиқат болған кезде, цикл жұмысын аяқтайды.

Келесі есептің шешу алгаритмін құрайық: *M* және *N* екі натуралды сан берілген. Олардың ең көп ортақ бөлгішін — НОД (*M, N*) есептеу керек.

Бұл есеп Евклид алгоритмі деген атаумен белгілі әдістің көмегімен шығарылады. Бұл әдістің идеясы, егер *M*>*N*, онда НОД



1.10.-сурет. Евклид алгоритмінің блок-схемасы

(*M, N*) = НОД (*M - N, N*) болады деген пікірге сәйкес құрылған. Осыны өз бетінше дәлелдеп көріңдер. Берілген алгоритмнің негізінде жатқан өзге пайымдау белгілі: НОД (*M, M*) = *M*. «Қолмен» орындау үшін бұл алгоритмді келесі нұсқау нысанында бейнелеуге болады:

1) егер сандар тең болса, жауап ретінде олардың ортақ мәнін алу керек. Кері жағдайда алгоритмді орындауды жалғастыру;

2) сандардың ішінде ең көбін анықтау;

3) ең көп санды ең көп және ең аз мәннің айырмасымен ауыстыру;

4) 1 т. атқаруға қайта келу.

Евклид алгоритмінің блок-схемасы 1.10.суретте берілген. АТ-де бұл алгоритмді келесідей жазуға болады:

```
алг      Евклид
тұт      M, N
бас      енгізу M, N
         әзірше   M ≠ N, қайталау
         цб егер M>N
            онда M := M- N
            басқаша N := N- M
            кв
         цс
соңы
```

Евклидалгоритмінде енгізілген тармақталумен берілген циклдің құрылымы бар. M = 18, N = 12 жағдайы үшін осы алгоритмнің жол тартуын өз бетінше орындаңдаңдар. Нәтижесінде НОД = 6 болуы тиіс.

1.4. АЛГОРИТМДЕУДІҢ ЛОГИКАЛЫҚ НЕГІЗДЕРІ

Бағдарламалауға тікелей қатысы бар пән математикалық логика деп аталады, оның негізін логика алгебрасы немесе пікірді есептеу құрайды. Пікір түсінігі ретінде оған қатысты ақиқат немесе жалған екендігін бірден айтуға болатын кез келген пайымдауды айтуға болады. Мысалы, «Ай – Жердің жерсерігі» деген пікір ақиқат, «5 > 3» — ақиқат, «Мәскеу — Қытай астанасы» — жалған, «1 = 0» — жалған. АҚИҚАТ және ЖАЛҒАН логикалық мәндер болып табылады. Келтірілген пайымдаулардың логикалық мәндері бір мағыналы анықталды. Басқа сөзбен айтқанда, олардың мәндері *логикалық константалар болады.*

x < 0, мұндағы *x*— айнымалы, теңсіздігінің логикалық мәні айнымалы болады, яғни шыққан мәніне байланысты *x* не ақиқат, не жалған болады. Осылайша, *логикалық айнымалының* түсінігі енгізіледі. Алгоритмдерді сипаттауда, сондай-ақ түрлі тілдердегі бағдарламалаудағы бағдарламаларда логикалық айнымалылар деректердің логикалық түрі сәйкес келетін символдық аттармен белігіленуі мүмкін. Басқаша айтқанда, егер *A, B, X, Y* және басқалары – логикалық түрдегі айнымалылар екендігі белгілі болса, бұл олардың тек АҚИҚАТ немесе ЖАЛҒАН мәнін ғана қабылдайтынын білдіреді.

Математикалық логиканың формалды аппаратының негіздерін XIX ғ. ортасында ағылшын математигі Джордж Буль қалады. Оның құрметіне пікірлерді есептеу бульдік алгебра деп, ал логикалық шамалар – бульдік шамалар деп аталады.

Логикалық өрнек (логикалық формула) — бұл қарапайым немесе күрделі пікір.

Күрделі пікір *логикалық амалдардың* (байламдардың) көмегімен құрылады.

Негізгі үш логикалық амал бар: терістеу, конъюнкция (логикалық көбейту) және дизъюнкция (логикалық қосу).

Терістеу математикалық логикада « — » белгіленеді және **ЕМЕС** сөзімен айтылады. Бұл бір орындық амал. Мысалы, (*x* = *y*) жазбасы былай оқылады: ЕМЕС (*x* тең *y*), яғни мәні ақиқат болады, егер *x y*-ке тең болмаса, және жалған болса, онда *x*тең *y*. Терістеу логикалық шаманың мәнін қарама-қарсыға өзгертеді.

Конъюнкция & таңбасымен белгіленеді және ЖӘНЕ сөзімен оқылады.Бұл екі орынды амал. Мысалы, (*x* > 0) & (*x* < 1) жазбасы берілген логикалық формула егер *x* ∈ (0,1) болса АҚИҚАТ мәнін қабылдайды, және кері жағдайда - ЖАЛҒАН мәнін қабылдайтынын білдіреді. Сәйкесінше, екі операндта ақиқат болса, конъюнкция нәтижесі ретінде АҚИҚАТ болады.

Дизъюнкция v таңбасымен белгіленеді, ол НЕМЕСЕ сөзімен оқылады. Мысалы, (*x* = 0) v (*x* = 1) жазбасы, егер *x*— екілік цифр болса (0 немесе 1), формула ақиқат мәнге ие болатындығын білдіреді. Сәйкесінше, егер операндтың біреуі АҚИҚАТ мәніне ие болса, дизъюнкция нәтижесі АҚИҚАТ болады.

Қарастырылған логикалық амалдарды орындаудың ережелері ақиқаттық кестесі деп аталатын 1.5 кестеде берілген (мұндағы *A* және *B* — логикалық шамалар, ал «А» және «Ж» әріптерімен сәйкесінше-АҚИҚАТ және ЖАЛҒАН мәндері белгіленген).

1.5.-кесте, Ақиқаттық кестесі

Рет №	А	В	А ЕМЕС	А ЖӘНЕ В	А НЕМЕСЕ В
1	Ж	А	Ж	А	А
2	А	Ж	Ж	Ж	А
3	Ж	А	А	Ж	А
4	Ж	Ж	А	Ж	Ж

Логикалық өрнектерде амалдарды орындау тәртібі амалдардың үлкендігімен: терістеу, конъюнкция, дизъюнкция анықталады. Бұдан басқа, логикалық формулалардағы амалдарды орындау тәртібін жақшалардың көмегімен өзгертуге болады.

(*А* ЖӘНЕ *В*) НЕМЕСЕ (*А* ЖӘНЕ *В* ЕМЕС) НЕМЕСЕ (*А* ЕМЕС ЖӘНЕ *В* ЕМЕС)

Мысал үшін логикалық формуланың мәнін есептеуді қарастырайық

X ЖӘНЕ *Y* НЕМЕСЕ *X* ЖӘНЕ *Z* ЕМЕС

Логикалық айнымалылардың келесі мәндерінде: *X* = ЖАЛҒАН, *Y*=АҚИҚАТ, *Z*=АҚИҚАТ

Берілген өрнекте амалдарды орындау тәртібін цифрлармен белгілейік:

① ② ④ ③

ЕМЕС *X* ЖӘНЕ *Y* НЕМЕСЕ *X* ЖӘНЕ *Z*.

Ақиқаттылықтың кестесін қолданып, қадамдар бойынша есептеуді орындайық:

1) ЖАЛҒАН ЕМЕС = АҚИҚАТ;

2) АҚИҚАТ ЖӘНЕ АҚИҚАТ = АҚИҚАТ;

3) ЖАЛҒАН ЖӘНЕ АҚИҚАТ = ЖАЛҒАН;

4) АҚИҚАТ НЕМЕСЕ ЖАЛҒАН = АҚИҚАТ.

Қарастырылған логикалық формуланың мәні — АҚИҚАТ.

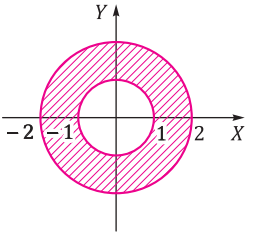
Келесі пайымдауды қарастырайық: «*X* мәні 0 ден 1-дейінгі аралықта болады». Бұл пайымдауды теңсіздіктер жүйесі түрінде жазуға болады

$$0 \le X \le 1.$$

Тиісті логикалық өрнек мынадай түрде беріледі

$$(X \ge 0) \text{ ЖӘНЕ } \le (X\ 1).$$

1.11.-сурет.



Енді есепті күрделілендірейік және келесі пайымдауды логикалық өрнек түрінде жазайық: координаталармен (X,Y) берілген жазықтықтағы нүкте, 1-ге тең ішкі радиуспен, және 2-ге тең сыртқы радиуспен координаталар басында ортасы бар шеңберде болады (1.11-сурет). Бұл өрнек мынадай түрде беріледі:

$(X^2+ Y^2\geq 1)$ ЖӘНЕ $(X^2 + Y^2 \leq 4)$.

Конъюнкция амалы келесідей атқарылады: бірінші теңсіздікке $(X^2+ Y^2> 1)$ қанағаттандыратын нүктелер жиынтығы таңдалады, және алынған жиынтықтан екінші теңсіздікке $(X^2 + Y^2 < 4)$ қанағаттанатын ішкі жиынтық бөлінеді, және ол соңғы нәтиже болып табылады. Сонымен қатар ізделетін жиынтық екі жиынтықтың қиылысуы болады, оның бірі бірінші теңсіздікке, ал екіншісі – екіншісіне қанағаттандырады. Конъюнкция — логикалық көбейту, яғни осындай қиылысуды есептейтін амал.

Логикалық өрнек түрінде келесі пайымдауды жазайық: координаталармен (X,Y) берілген жазықтықтағы нүкте, 1-ге тең ішкі радиуспен, және 2-ге тең сыртқы радиуспен координаталар басында ортасы шеңберден тыс болады (1.11-сурет). Бұл өрнек мынадай түрде беріледі:

$(X^2 + Y^2< 1)$ НЕМЕСЕ $(X^2 + Y^2 > 4)$.

Дизъюнкция амалын орындаукелесі тәртіпте беріледі: бірінші теңсіздікке $(X^2 + Y^2 < 1)$ қанағаттандырушы нүктелер жиынтығы таңдалады, және оған екінші теңсіздікке $(X^2 + Y^2 > 4)$ қанағаттандырушы жиынтық қосылады.Сөйтіп, дизъюнкция амалының екінші атауының мағынасы – логикалық қосу, яғни екі жиынтықты қосу айқындалады.

Соңғы есепті басқаша шешуге болады. Тиісті логикалық өрнекті бірінші логикалық өрнектің терістеуі ретінде жазуға болады,

өйткені шеңберден тыс нүктелер, - бұлар шеңберге тиісті ЕМЕС нүктелер:

$$((X^2 + Y^2 \geq 1) \text{ ЖӘНЕ } (X^2 + Y^2 \leq 4)) \text{ ЕМЕС.}$$

Сәйкесінше, екі логикалық өрнек арасындағы теңдік дұрыс:

НЕМЕСЕ$((X^2 + Y^2 \geq 1)$ ЖӘНЕ $(X^2 + Y^2 \leq 4)) = (X^2 + Y^2 < 1)$ НЕМЕСЕ $(X^2 + Y^2 > 4)$.

Бұл мысалдан түрлендірудің келесі ережесі түсінікті болады: конъюнкциямен байланысқан екі теңсіздікке қолданылған терістеу амалы бұл теңсіздіктердің белгілерін қарама-қарсыларға, ал конъюнкцияны - дизъюнкцияға (немесе дизъюнкцияны конъюнкцияға) өзгертеді.

Күрделі логикалық өрнек қолданылатын алгоритм мысалын қарастырайық. Үшбұрыштың *a, b*, сүшқабырғасының ұзындықтары бойынша оның ауданын табайық.

Берілген есепті шешу үшін Герон формуласы деп аталатын келесі формула қолданылады:

$$\sqrt{p(p-a)(p-b)(p-c)},$$

мұндағыр = (a + b + c) /2 — үшбұрыштың жартылай периметрі.

Бастапқы деректер үшбұрыштың қабырғалары үшін берілген негізгі ара қатынасты қанағаттандыруы тиіс: әрбір қабырғаның ұзындығы оның басқа екі қабырғасының ұзындықтарының қосындысынан кем болуы тиіс.

Үшбұрыш ауданын есептеу алгоритмі тармақталатын құрылымға ие болады.

Тармақталудың құрылымдық командасында шартты қате бастапқы деректердің барлық нұсқаларын «сүзуге» мүмкіндік беретін күрделі логикалық өрнек түрінде жазайық:

```
алг  Герон
зат   A, B, C, P, S
бас  енгізуA, B, C
егер (A> 0) ЖӘНЕ (B> 0) ЖӘНЕ (C> 0) ЖӘНЕ (A+ B>C)
     ЖӘНЕ (B+ C>A) ЖӘНЕ (A+ C>B)
     онда
          P:= (A+ B+ C)/2
          S : = түбір(P * (P - A) * (P - B) * (P - C))
          шығару «Аудан  =», S
     басқаша «Қате бастапқы деректер» шығару
     кв
соң
```

1.5. КӨМЕКШІ АЛГОРИТМДЕР МЕН ПРОЦЕДУРАЛАР

Алгоритмдер теориясында көмекші алгоритм, яғни негізгі шешілетін есептенкейбір ішкі есепті шешу алгоритмі түсінігі бар. Сонымен бірге бастапқы есепті шешу алгоритмі негізгі деп аталады.

Мысал ретінде келесі есепті қарастырайық. Тұтас көрсеткіші бар дәрежелік функцияны есептеу алгоритмін құру керек: y = xk, мұндағық — тұтас сан; x≠ 0.

Алгебрада мұндай функция былай анықталады:

$$x^n = \begin{cases} 1 & \text{Ł } n = 0; \\ \dfrac{1}{x^{-n}} & \text{Ł } n < 0; \\ x^n & \text{Ł } n > 0. \end{cases}$$

Берілген есепте ішкі есеп ретінде санды тұтас оң дәрежеге шығаруды қарастыруға болады.

$1/x^{-n} = (1/x)^{-n}$ ескере отырып, бұл есепті шешудің негізгі алгоритмін жазайық:

```
алг      дәрежелік функция
тұт      n; зат x, y
бас      енгізу x, n
         егер n= 0
         онда y : = 1
         басқаша егер n > 0
         онда ДӘРЕЖЕ (x, n, y)
         басқаша ДӘРЕЖЕ (1/x, -n, y)
         кв
    кв
    шығару y
соң
```

Негізгі алгоритмде оны көп реттік қайта көбейту арқылы тұтас оң дәрежеге шығарудың заттық негіздеуді шығару алгоритмі - ДӘРЕЖЕ атымен берілген көмекші алгоритмге жүгіну командасы екі рет қайталанады. Көмекші алгоритмге жүгіну командасында жақшада тұрған шамалар *нақты параметрлер* деп аталады.

Оқу алгоритм тілінде көмекші алгоритмдер процедуралар түрінде рәсімделеді. АТ-де ДӘРЕЖЕ процедурасын жазып көрсетейік:

```
ДӘРЕЖЕ процедурасы (зат a, тұт k, зат z)
тұт i
бас  z : = 1; i : = 1
     әзірше i ≤k, қайталау
     цс  z : = z x a
         i : = i +1
     цс
соң
```

Көмекші алгоритмнің тақырыпаты «процедура» сөзінен басталады, содан соң осы процедураның аты және жақшада – нысандық параметрлердің тізімі беріледі. Бұл тізімде олардың түрлері көрсетіліп, айнымалы аргументтер және айнымалы нәтижелер аталады. Берілген мысалда *a, k* — нысандық параметрлер-аргументтер, *z* — параметр-нәтиже. Сәйкесінше, ДӘРЕЖЕ процедурасы есептеулерді $z=a^k$ формуласы бойынша шығарады.

Дәрежелік функцияны есептеудің негізгі алгоритмінде процедураға жүгіну оның атын және содан соң жақшада нысандық параметрлер тізімін көрсету арқылы шығарылады.

Процедуралардың нысандық және нақты парметрлері арасында келесі сәйкестік ережелері атқарылуы тиіс:

- саны бойынша (қанша нысандық, сонша нақты параметрлер);
- жүйелігі бойынша (бірінші нысандық параметрге бірінші нақты параметр, екіншісіне – екінші және т.с.с. сәйкес келеді);
- түрлері бойынша (тиісті нысандық және нақты параметрлердің түрлері бір -біріне сәйкес келуі керек).

Нақты параметрлер-аргументтер тиісті түрдің өрнегі бола алады.

Процедураға жүгіну келесі әрекеттерге бастамашылық етеді:

- параметрлер-аргументтердің мәндері тиісті нысандық параметрлерге беріледі;
- процедураның денесі орындалады (процедураның ішіндегі командалар);
- нәтиженің мәні тиісті нақты параметрге беріледі, және негізгі алгоритмнің келесі командасын орындауға көшу болады.

ДӘРЕЖЕ процедурасында бастапқы деректерді енгізу және нәтижені шығару командалары жоқ. Мұнда аргументтерге алғашқы мәндерді беру (*a, n*) параметрлер-аргументтерді жіберу арқылы, ал айнымалының нәтижесін (*y*) беру параметр-нәтижені (*z*) жіберу арқылы шығарылады.

Осылайша, процедура параметрлерінің мәндерін жіберу меншіктеудің үшінші тәсілі болады (меншіктеу және енгізу омандаларымен қатар).

Процедураларды қолдану жүйелі тәптіштеу әдісімен күрделі алгоритмдерді құруға мүмкіндік береді.

1.6. ҚҰРЫЛЫМДЫҚ БАҒДАРЛАМАЛАУ НЕГІЗДЕРІ

Бірінші ЭЕМ пайда болғаннан бері жарты ғасырдан артық уақыт өтті. Осы уақыт ішінде есептеу техникасы қарқынды түрде дамыды. ЭЕМ элементтік қоры өзгерді, олардың тезәрекеттігі және жады көлемі ұлғайды, адамның машинамен интерфейс құралдары өзгерді. Сөзсіз, мұндай өзгерістер бағдарламашының жұмысына тікелей әсерін тигізбей қойған жоқ.

Қандай да бір нәрсені (берілген жағдайда – бағдарламаларды) жалпыға ортақ өндіру тәсілі технология деп аталады, сондықтан әрі қарай *бағдарламалау технологиясы* туралы сөз қозғаймыз.

Жады «тар» және тез әрекеттігі шағын бірінші ЭЕМ үшін бағдарлама сапасының негізгі көрсеткіші ретінде оның жадының алу көлемі және есептеу уақыты бойынша үнемділігін айтуға болады. Бағдарлама неғұрлым қысқа болса, бағдарламашының санаты соғұрлым жоғары саналды. Мұндай бағдарламаларды қысқарту көбіне үлкен күш жұмсауды талап етті. Кейде бағдарламаның «айлалы» болғаны сонша лықты, ол автордан да «айласын асыруы» мүмкін еді: біраз уақыт өткен соң бірдеңе өзгерткісі келіп,өз бағдарламасына қайта оралған бағдарламашы өзінің «даналық пікірін» ұмытып, шатасуы мүмкін еді.

Өте күрделі алгоритм әрдайым бағдарламада қате жіберу ықтималдығын, сондай-ақ қарапайымға қарағанда, күрделі техникалық құрылғының істен шығу ықтималдығын ұлғайтады.

Бағдарламашының бағдарламалық өнімді дайындаудың үш кезеңі:

1) жобалау;
2) кодтау;
3) реттеу.

Жобалау болашақ бағдарлама алгоритмін, мысалы блок-схеманы әзірлеуден тұрады. Кодтау – бұл бағдарламалау тілінде бағдарлама мәтінін құру. Ретке келтіру тестердің көмегімен жүзеге асырылады, яғни нәтижесі белгілі болатын, қандай да бір бұрынырақ ойланған бастапқы деректер жиынтығымен бағдарламаны атқару жүргізіледі. Бұл ретте бағдарлама неғұрлым күрделі болса, оны жан-жақты толық тексеру үшін соғұрлым көбірек тест саны қажет болады. Өте «айлалы» бағдарламаны мұқият тексеру қиындық тудырады, өйткені әрдайым қандай да бір «су астындағы тасты» байқамау мүмкін емес.

ЭЕМ жады мен тезәрекеттілігінің, бағдарламалау тілдері мен осы тілдерден таратушылардың жетілдірілуінің артуымен бағдарламаның үнемділік проблемасы тым өзекті болудан қала бастады. Бағдарламалардың неғұрлым сапалық сипаттары ретінде олардың қарапайымдылығы, көрнекілігі және сенімділігі болды, ал үшінші буын машиналары пайда болуымен бұл қасиеттер негізгі қасиеттерге айнала бастады.

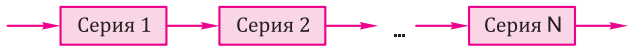
1960 жылдардың соңында 1970 жылдардың басында құрылымдық бағдарламалау атауына ие болған пән пайда болды. Оның пайда болуы және дамуы Э.В.Дейкстрдің, Х.Д.Милстің, Д.Е.Кнуттың және басқаларының есімдерімен байланысты болды. Құрылымдық бағдарламалау осы уақытқа дейін бағдарламалау технологиясының негізі болып келді. Оның қағидаларын сақтау бағдарламашыға анық, кемшіліксіз, сенімді бағдарламаларды құруды тез үйренуге мүмкіндік береді.

Құрылымдық бағдарламалаудың негізінде бағдарламалау теориясында дәлелденген теорема жатыр: *кез келген логикалық есепті шешудің алгоритмін* негізгі алгоритмдік құрылымдар деп аталатын *Ілесу*, *Тармақталу*, *Цикл құрылымдарынан ғана құруға болады*.

Бұл құрылымдар бұдан бұрын келтірілген болатын. Шын мәнісінде, бағдарламалардың барлық қарастырылған мысалдарында құрылымдық бағдарламалау қағидалары қолданылды.

Ілесу — бұл әрекеттердің сызықтық жүйелігі (1.12сурет).

Мұндай жүйеліктегі әрбір серияның ішінде қарапайым команда да, күрделі құрылым да болуы мүмкін, бірақ онда тек бір ғана ену және бір ғана шығу бар.



1.12.-сурет. Ілесу бағдарламалаудың сызықтық құрылымы

Тармақталу — бұл келесідей жазуға болатын алгоритмдік балама:

егер шарты

онда 1 серия

басқаша 2 серия

кв

Бұл құрылымдағы басқару шарттың ақиқаттылығына немесе жалғандығына байланысты екі тармақтың біріне (командалар сериясына) жіберіледі. Содан соң жалпы жалғасымға шығу жүзеге асырылады (1.5-суретті қараңыз).

Тармақталудың толық емес формасы «басқаша» тармағыболмаған кезде орын алады:

егер шарты

онда сериясы

кв

Цикл — бұл шарты бойынша кейбір әрекеттер тобының қайталануы. Циклдердің екі түрі бар. Циклдік құрылымның бірінші түрі – алғы шарт циклі(1.7-суретті қараңыз):

әзірше шарты**, қайталау**

цб

сериясы

цс

Мұнда шарт әзірше ақиқат болса, циклдің денесін құрайтын серия атқарылады.

Циклдік құрылымның екінші түрі – шарттан кейінгі цикл (1.9-суретті қараңыз):

қайталау

сериясы

дейін шарты

Мұнда циклдің денесі цикл шартынан бұрын келеді және шарт жалған болса, өз атқаруын қайталайды. Шарт ақиқат болған кезде қайталау аяқталады.

Теориялық тұрғыдан кез келген циклдік алоритмді құру үшін қажетті және жеткілікті болып алғышарты бар цикл саналады, яғни бұл артқышартпен берілген циклге қарағанда циклдің неғұрлым жалпы нұсқасы. Ақиқатында, «Дейін» циклінің денесі ең құрығанда бір рет болса да міндетті түрде атқарылады, өйткені шартты тексеру оны орындау аяқталған соң жүргізіледі, ал «Әзірше» циклі үшін циклдің денесі бір де бір рет орындалмаған нұсқа мүмкін болады.

Сәйкесінше, бағдарламалаудың кез келген тілінде «Әзірше» циклін қолданумен шектелуге болатын еді, алайда бірқатар жағдайларда «Дейін» циклін енгізу аса қолайлы болады, сондықтан осы цикл қолданылады.

Кей жағдайларда әдебиетте құрылымдық бағдарламалауды GOTO-сыз (сөзсіз ауысу операторысыз) бағдарламалау деп те атайды. Шынында да, бағдарламалауға бұлай қарау кезінде сөзсіз ауысуға орын жоқ. Бағдарламаларда GOTO еш дәлелсіз қолдану олардың құрылымдылығынан айырады, ендеше, мұнымен байланысты алгоритмнің барлық оң қасиеттерінен: айқындылықтан және сенімділіктен айырады. Бағдарламалаудың барлық процедуралық тілдерінде бұл оператор бар болғанымен, бағдарламалауға құрылымдық тәсілдемені қамтамасыз ету үшін оны қолданудан аулақ болған жөн.

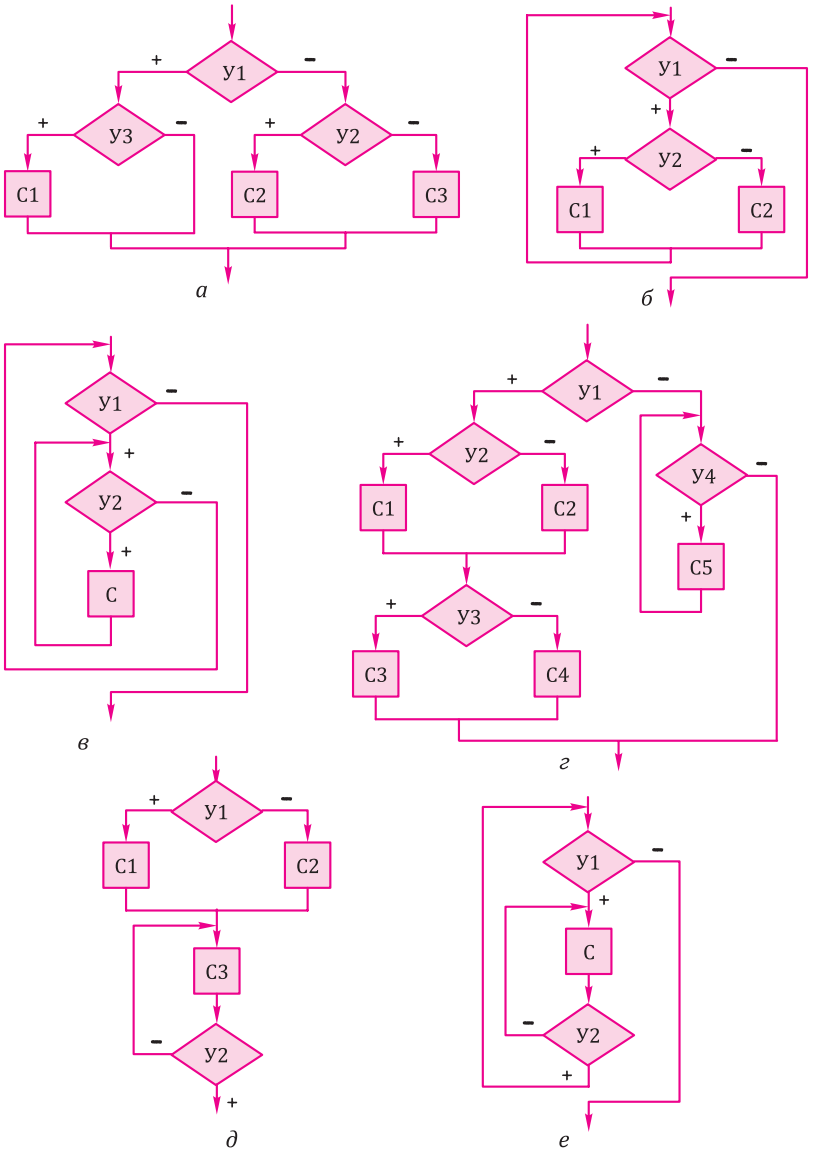
Күрделі алгоритмдер бір-бірімен жалғанған негізгі құрылымдардан тұрады. Бұл құрылымдарды жалғау екі тәсілмен: *жүйелі* және *тіркемеленген* тәсілмен орындалуы мүмкін. Бұл кез келген күрделі электр тізбегін жүйелі және параллельді жалғанған бөліктерге жайғастыруға болатын электротехникадағы жағдайға ұқсас.

Алайда тіркемеленген алгоритмдік құрылымдар параллельді жалғанған электр өткізгіштерінің баламасы бола алмайды. Бұл жерде бірінің ішіне бірі салынған матрешкалармен берілген балама сәйкес келеді. Егер циклдің денесін құрайтын блоктың өзі циклдік құрылым болатын болса, онда тіркемеленген циклдер бар дегенді білдіреді. Өз кезегінде ішкі циклдің ішінде тағы бір цикл және т.с.с. болуы мүмкін. Осыған байланысты *циклдердің тіркемелеуінің тереңдігі түсінігі* қалыптасты. Дәл осылай тармақталулар да бірінің ішіне бірі тіркемелеуі мүмкін.

Құрылымдық тәсілдеме алгоритмдердің блок-схемаларын стандартты түрде бейнелеуді талап етеді. Әрбір негізгі құрылымның бір кіре берісі және бір шығысы болуы керек. Алгоримнің стандартсыз бейнеленген блок-схемасы дұрыс оқылмайды және өз көрнекілігін жоғалтады.

Алгоритмдердің құрылымдық блок-схемаларының мысалдары 1.13 суретте берілген. Мұндай блок-схемалар тез оқылады және көзбен жақсы қабылданады, сонымен бірге әрбір құрылымға ат қоюға болады.

Құрылымдық алгоритмдерді сипаттауға арналған оқу АТ-де сөзсіз ауысу командасы жоқ. Оқу кезінде бұл тілді қолдану бағдарламашының «құрылымдық тәрбиеленуіне» ықпал етеді.



1.13-сурет. Алгоритмдердің құрылымдық блок-схемаларының мысалдары: *а* — тіркемелену тереңдігі бірге тең тіркемеленген тармақталулар; *б* — тіркемеленген тармақталуы бар цикл; *в* — тіркемелену тереңдігі бірге тең «Әзірше» тіркемеленген циклдер; *г* —тармақталудың оң тармағында тіркемеленген жүйелігімен және «Әзірше» тіркемеленген циклімен теріс тармақта тармақталу; *д* — тармақталудың және «Дейін» циклінің жүйелігі; *е* — тіркемеленген «Әзірше» сыртқы циклі және «Дейін» ішкі циклі; У — шарты; С — сериясы

Алгоритм құрылымдарына АТ-де көрнекілікті олардың мәтіндерінің түрін құрылымдау береді. Бұл үшін негізгі қолданылатын тәсіл – келесі ережелерге бағынатын жолдарды ысыру:

- тіркемеленудің бір деңгейлік құрылымдары бір тік деңгейде орналасады (яғни жолақта бір позициядан басталады);
- Тіркемеленген құрылымдар жолақ үстінде құрылымдар үшін сыртқыға қатысты бірнеше позицияға оңға қарай жылжиды.1.13-суретте көрсетілген алгоритм мәтіндерінің құрылымдары АТ-де 1.6-кестеде көрсетілген.

1.6.-кесте. 1.13-суретте көрсетілген схемаларға сәйкес келетін АТ-дегі алгоритмдер

Схема	Алгоритм	Схема	Алгоритм
а	**егер** <У1> **онда егер** <У3> **онда** <С1> **кв** **басқаша егер** <У2> **онда** <С2> **басқаша** <С3> **кв** **кв**	*б*	**әзірше** < У1>, **қайталау** **цб** **егер** <У2> **онда** <С1> **басқаша** <С2> **кв** **цс**
в	**әзірше** <У1>, **қайталау** **цб** **әзірше** <У2>, **қайталау** **цб** <С> **цс** **кц**	*г*	егер<У1> онда егер<У2> онда<С1> басқаша<С2> кв егер<У3> онда<С3> басқаша<С4> кв басқаша әзірше<У4>, қайталау <С5> цс кв
д	**егер** <У1> **онда** <С1> **басқаша** <С2> **кв** **қайталау** С3 **дейін** <У2>	*е*	**әзірше**<У1>, **қайталау** **цб қайталау** <С> **дейін** <У2> **цс**

Алгоритмдеудің құрылымдық әдістемесі – бұл тек алгоритмді сипаттау формасы ғана емес, сонымен қатар *бағдарламашының ойлау қабілеті*. Алгоритмді стандартты құрылымдардан құруға ұмтылу керек. Құрылыс үйлестігін қолдана отырып, алгоритм құрудың құрылымдық әдістемесі стандартты секциялардан ғимарат құрумен бірдей екенін айтуға болады.

Құрылымдық бағдарламалаудың тағы бір маңызды технологиялық тәсілі шешілетін *есепті ішкі есептерге*, яғни бастапқы есептің бөлігін бағдарламалауға арналған қарапайым есептерге *бөлшектеп байланыстыру*. Мұндай ішкі есептерді шешу алгоритмдері *көмекші* алгоримдер деп аталады. Бұл ретте алгоритм құрудың екі түрлі тәсілдемесі мүмкін болады:

- жоғарыдан төмен қарай — алдымен негізгі алгоритм, содан соң көмекші алгоритмдер құрылады;
- төменнен жоғары қарай — алдымен көмекші алгоритмдер, содан соң негізіг алгоритмдер құрылады.

Алгоритм құрудың бірінші тәсілдемесін сондай-ақ жүйелі тәптіштеу әдісі, ал екіншісін- жинақтау әдісі деп атайды.

Жүйелі тәптіштеу кезінде алдымен негізгі алгоритм құрылады және оған бірінші деңгейлегі көмекші алгоритмдерге жүгіну енгізіледі. Содан соң екінші деңгейдегі көмекші алгоритмдерге жүгіну болатын және т.с.с. бірінші деңгейдегі көмекші алгоритмдер құрылады. Ең төменгі деңгейдегі көмекші алгоритмдер тек қарапайым командалардан тұрады.

Жүйелі тәптіштеу әдісі кез келген күрделі объектілерді құрастыру кезінде қолданылады. Бұл конструктордың табиғи логикалық ойлау жүйелігі: біртіндеп егжей-тегжейіне жету. Бағдарламалауда құрылымдау туралы айтылады, бірақ техникалық емес алгоритмдердің құрылымдары туралы сөз болады. Айтарлықтай күрделі алгоритмді басқа тәсілмен құру мүмкін емес.

Жүйелі тәптіштеудің әдістемесі бағдарламашылар ұжымының күрделі жоба бойынша жұмысын ұйымдастыруға мүмкіндік береді. Мысалы, топ жетекшісі негізгі алгоритмді құрады, ал көмекші алгоритмдерді әзірлеуді және тиісті шағын бағдарламаларды жазуды өзінің қызметкерлеріне жүктейді. Сонымен бірге жұмыс тобының қатысушыларына өздерінің бағдарламалалық модульдері арасында интерфейс жайында (яғни өзара байланыс) келіседі, ал бағдарламаны ішкі ұйымдтыру – бағдарламашының жеке ісі.

Құрастырма әдісі бағдарламалау тілдерінде ішкі бағдарламалар,

Жаттығулар

1. (*X* >*Z*) ЕМЕС ЖӘНЕ (*X* = *Y*) ЕМЕС логикалық өрнектің мәнін айнымалылардың келесі мәндерінде анықтау керек:

а) *X* = 3, *Y*= 5, *Z*= 2;

б) *X* = 0, *Y*= 1, *Z*= 19;

в) *X* = 5, *Y*= 0, *Z*= -8;

г) *X* = 9, *Y* = -9, *Z* = 9.

2. Келесі шарттар бойынша ақиқат болатын логикалық өрнектерді (формулаларды) жазу керек:

а) (*X, Y*) координаталары бар нүкте координаталар басында ортасы бар бірлік шеңбердің бірінші төрттен бір бөлігіне жатады;

б) (*X, Y*) координаталары бар нүкте координаталар басында ортасы бар бірлік шеңбердің бірінші төрттен бір бөлігіне жатпайды және радиусы 2-ге тең, координаталар басында ортасы бар дөңгелекке жатады (мұны графикалық түрде бейнелеңіз).

3. Жазықтықтағы үшбұрыштың үш бұрышының декарттық координаттары берілген. Үшбұрыштың ауданын анықтау алгоритмін құру керек.

4. Жердің атмосфера шегінен шығу кезіндегі ракетаның жылдамдығы берілген. Қозғалтқыштарды сөндірген соң ракета қозғалысын анықтау алгортмін құру керек. (Үш космостық жылдамдықтың мәндері: 7,5 км/с; 11,2 км/с; 16,4 км/с.)

5. Үш оң сан берілген. Осы сандар үшбұрыш қабырғаларының ұзындығы бола алатындығын анықтайтын алгоритм құру керек.

6. Компьютер тек екі арифметикалық амалды: қосуды және азайтуды орындай алады делік. Келесі алгоримдер құру керек:

а) екі тұтас санды көбейту;

б) екі санды тұтас сан ретінде бөлу;

в) екі санды тұтас сан ретінде бөлуден қалдық алу.

7. Көмекші алгоритм ретінде квадрат теңдеудің шешуін қолданып, биквадрат теңдеудің шешу алгоритмін құру керек.

8. Екі санның ЕОБ табудың көмекші алгоритмін қолданып, үш натурал санның ЕОБ табу алгоритмін құру керек.

БАҚЫЛАУ СҰРАҚТАР ЖӘНЕ ТАПСЫРМАЛАР

1. Бағдарламалауды қолдана отырып, компьютерде есептер шығару кезеңдері қандай? Оларды сипаттаңдар. Қойылу және нысандандыру кезеңдерін келесі есептің мысалында бейнелеп беріңдер: екі пункт аралығында моторлы қайықтың өзендегі жүру уақытын есептеу қажет.

2. Компьютерде бағдарламалауда деректердің қандай негізіг түрлері қолданылады?
3. Сызықтық алгоритм қандай командалардан тұрады? Сызықтық алгоритмнің көмегімен шығарылатын есепке мысал келтіріңдер.
4. Тармақталудың алгоритмдік құрылымы деген не? Толық және толық емес тармақталудың айырмашылығы неде?
5. 1-т. түсіндірілген тармақталу құрылымы бар алгоритмнің көмегімен шығарылатын есепке мысал келтіріңдер.
6. Циклдің алгоритмдік құрылымы деген не? Алғышарт циклі («Әзірше» циклі) және артқышарт циклінің («Дейін» циклі) арасындағы айырмашылық неде? Циклдік алгоритнің көмегімен шығарылатын есепке мысал келтіріңіз. Осы есепті: «Әзірше» циклімен және «Дейін» циклімен шығарудың екі нұсқасын құрыңыз.
7. Көмекші алгоритм (процедура) деген не? Процедура алгоритм тілінде қалай жазылады? Негізгі алгоритмде процедураға жүгіну қалай жазылады? Келесі есепті шешу үшін ішкі есепті бөлуді және процедураны қолдануды бейнелеңдер: ішкі және сыртқы радиустартың берілген мәндері бойынша шеңбердің ауданын табу керек.
8. Логикалық амалдардан тұратын логикалық өрнекті есептеудің тәртібі қандай? Бірнеше логикалық амалдан тұратын өрнекті есептеуге мысал келтіріңдер.
9. Құрылымдық бағдарламалау деген не? Алгоритмдерді құрудың құрылымдық әдістемесінің негізгі қағидалары қандай?

ҚҰРЫЛЫМДЫҚ БАҒДАРЛАМАЛАУ

2.1. БАҒДАРЛАМАЛАУДЫҢ ТІЛДЕРІН ЖӘНЕ ТЕХНОЛОГИЯСЫН ДАМЫТУ

ЭЕМ үшін бағдарламалау — компьютер жұмысын басқару бағдарламаларын құру үрдісі.

Машиналық-бағытталған бағдарламалау. Бағдарламалық басқарылатын есептеу машиналарын ойлап тапқаннан бері жаңа мамандық – бағдарламашы мамандығы пайда болды. Тарихта бірінші бағдарламашы Чарльз Беббиджбен бірге жұмыс істеген Ада Лавлейс болды, ол аналитикалық машинамен оны басқару бағдарламасын әзірледі (1860—1870 жж.). Алайда бағдарламашы мамандығы ЭЕМ ойлап тапқан соң ғана көпшілікке белгілі бола бастады.

Бағдарламашылар алғашқы буындағы лампалық ЭЕМ-да процессордың командасын тікелей қолдана отырып, өз бағдарламаларын құрды. Сонымен бірге бағдарламашыға жадының ұяшықтарын деректерге және бағдарлама командаларына қарай бөлуге тура келді. Оған процессор командасының жүйесін және барлық командалардың кодтарын білу керек болды. Бастапқы деректер мен командалар екілік код түрінде, яғни тікелей ЭЕМ жадында сақталған түрде берілді. Бағдарламалар жазбаларын қысқарту үшін арнайы блактерде әдетте екілік-сегіздік немесе екілік-он алтылық кодты қолданатын (2.1кесте).

2.1.-кесте. Бірінші буындағы компьютерлердің біріне арналған бағдарлама командасының мысалы

Коды	Команданың мекенжайы	Амалдың коды	1-ші мекенжай	2-ші мекенжай	3-ші мекенжай
Он алтылық	28	02	C0	C4	D8
Екілік	0010 1000	0000 0010	1100 0000	1100 0100	1101 1000

2.1-кестеде берілген команда үшмекенжайлы деп аталады. 02_{16} коды қосу командасына жатады; 1-ші және 2-ші мекенжай- бұлқосылғыштар сақталатын жедел есте сақтайтын құрылғы (ЖЕҚ) ұяшықтарының мекенжайлары, 3-ші мекенжай – қосынды жазылатын ұяшықтың мекенжайы. Команданың өзі 2816мекенжайымен ЖЕҚ ұяшығында сақталады.

Машиналық кодтардағы бағдарламалау өз алдына күрделі үрдіс. Сол себепті бағдарламашылардың жұмыс қабілеттілігі өте төмен болды. 1950 жылдары «бағдарламалауды автоматтандыру» атауына ие болған бағыт пайда болды. Оның негізгі мақсаты – ЭЕМ үшін бағдарлама құру үрдісін жеңілдететін және жылдамдататын құралдар жасау.

Бағдарламалаудың бірінші тілі ретінде машиналық-бағытталған *автокодтар* болды. Кейінірек мұндай деңгейдегі тілдерге «ассемблерлер» деген атау берілді. Алғашқыда *ассемблер* деп ассамблер тілінен машиналық командаларға берілетін бағдарлама-аудармашыны атайтын. Кейін ассемблер тілін де «ассемблер» атымен атайтын болды. Ассемблерде бағдарламалау бағдарламашыдан жадыны деректерге және бағдарлама командаларына бөлу туралы міндетті алып тастайды. Бағдарламашы процессордың барлық командаларының ішкі кодтарын есте сақтауы тиіс емес. Ассемлерде қосудың дәл сол командасының автокодта берілу мысалы:

ADD a, b, c

ADD сөзі «қосу» командасын білдіреді; a, b — айнымалылар-қосылғыштардың аттары; c — нәтиже енгізілетін айнымалы.

Ассемблердің тілі машиналық-бағдарланған деп аталу себебі – процессордың әрбір командасы үшін ассемблерде команданың өз баламасы болады. ЭЕМ түрлі типтері процессор командаларының түрлі жүйелеріне ие болғандықтан, олардың ассемблерлері де бір-бірінен ерекшеленеді. Қазіргі кездегі ассемблерлер процессорлардың белгілі бір түрлеріне дәл осылайша бағдарланған. Кейінірек м*акроассемблерлер* атауы пайда болды, олардың тілінде процессор тіліндегі командалар сериясына (ішкі бағдарламаларға) сәйкес келетін макрокомандалар болады.

Ассемблерде бағдарлама құру процессор командаларының тіліне қарағанда жеңілдеу. Жадыны деректер мен командаларға бөлу бойынша, ассамблер командаларын машиналық командаларға ауыстыруды арнайы жүйелік бағдарлама – транслятор жүзеге асырады.

Ассемблерде бағдарламаларды машиналық бағыттаудан мұндай бағдарламаларды орындау үшін процессор командаларының басқа жүйесімен ЭЕМ басқа түрлеріне ауыстыруға болмайтындығын атап өту керек. Бұл мәселе қолданбалы бағдарламашылар үшін айтарлықтай тосқауыл болды. Бұдан басқа, ассемблердегі бағдарламалаудың өзі ЭЕМ-ды қолданбалы салаларда қолдануды шектеген, жаппай меңгеру үшін айтарлықтай күрделі болды.

Жоғары деңгейлі бағдарламалау тілдері. Бағдарламалауды дамытудың келесі кезеңі жоғары деңгейлі бағдарламалау тілдерін (ЖДБТ) құру болады. ЖДБТ мысалдары: Паскаль, Бейсик, Фортран. Әрбір тіл үшін машиналық-тәуелсіз стандарт болады. Берілген ЖДБТ бағдарламалау мүмкіндігі компьютерде осы тілден алынатын транслятордың болуына байланысты болады. Компьютердің әрбір түріне арналған трансляторларды жүйелік бағдарламашылар құрады.

ЖДБТ-де бағдарлама мәтіні өзінің формасы бойынша табиғи тілдерге (көбіне – ағылшын тіліне), математика тіліне жақын. Дәл сондай екі шаманы қосу командасы ЖДБТ-де математикалық теңдіктің әдеттегі түріне ұқсас:

c:= a+ b(Паскальда);
c= a+ b(Фортранда, Бейсикте, Сиде).

Жоғары деңгейлі тілде бағдарламалауды меңгеру ассемблердегіге қарағанда анағұрлым жеңіл, сондықтан ЖДБТ пайда болуымен қолданбалы бағдарламашылардың саны айтарлықтай артты, ЭЕМ-ды бірсыпыра салаларда қолдану кең етек жайды.

XX ғ. ортасынан бастап және біздің кезімізге дейін жүздеген ЖДБТ құрылды. Алайда олардың ішінде барлығы бірдей кеңінен таратылып, танымал болған жоқ. ЖДБТ ішінен көп жасағандардың бірі – Фортран тілі. Ағылшынша: *Fortran— formula translate*— формулалар транс¬ляторы. Фортранның бірінші нұсқасы 1954 жылы құрылды. Екінші және үшінші буындағы ЭЕМ кезінде Фортран-IV нұсқасы танымал болды. Фортран ғылым мен техникада қолданылатын, математикалық есептеулерге арналған мамандандырылған тіл ретінде құрылды. Біздің кезімізде де бұл тіл Фортран-90 (және оның кейінгі модификацияларында Фортран-95, Фортран-2003) стандартында, физика-техникалық проблемалар саласында есептеулер үшін негізгі бағдарламалау тілі болып қалады.

1950 жылдары құрылған алғашқы ЖДБТ қатарына Кобол (АҚШта құрылды)және Алгол (Еуропада) жатады. Фортран сияқты

Алгол да математикалық сипаттағы ғылыми-техникалық есептеулерге бағдарланды. Кобол — экономикалық есептерді бағдарламалауға арналған тіл. Аталған басқа екі тілмен салыстырғанда,Коболда математикалық құралдар әлсіздеу дамыған, бірақ мәтіндерді өңдеу, деректерді талап етілетін құжат түрінде шығаруды ұйымдастырукұралдары жақсы дамыған. Алғашқы ЖДБТ үшін тілдердің пәндік бағдарлануы өзіндік ерекшелігіне жататын еді.

Бағдарламалау тілдерінің біразы 1960—1970 жылдары пайда болды. 1965 ж. Дартмут университетінде Бейсик тілі әзірленді. Авторлардың ойынша бұл тез үйренетін, күрделі емес, есептелетін есептерді бағдарламалауға арналған қарапайым тіл. Бейсик тілі әсіресе микро-ЭЕМ мен персоналды компьютерлер (ПК) пайда болған кезден бастап танымал бола бастады.

Бағдарламалау тілдерінің тарихында елеулі оқиға ретінде 1969 ж. Паскаль тілін құру болды. Оның авторы - швейцар профессоры Н. Вирт — Паскальді құрылымдық бағдарламалаудың оқу тілі ретінде әзірлеген.

Паскаль тілін таратуды біршама табысқа ие болған ПК қамтамасыз етті. *Borland International*, Inc(АҚШ) фирмасы ПК арналған ТурбоПаскаль бағдарламалау жүйесін дайындады. ТурбоПаскаль — бұл тек тіл және одан алынатын транслятор ғана емес, сонымен бірге пайдаланушыға Паскальда жайлы жұмыс жасауға: бағдарлама мәтінін енгізуге және түзетуге, синтаксистік қателерді іздеуге, ішкі бағдарламалар мен модульдердің кітапханаларын пайдалануға, файлдармен жұмыс жасауға және т.б. мүмкіндік беретін *бағдарламалаудың кіріктірілген ортасы*. ТурбоПаскаль оқу бағдарламасы шегінен шығып, жан-жақты мүмкіндіктерге ие болып, кәсіби бағдарламалау тіліне айналды. Паскаль бағдарламалаудың көптеген негізгі тілдерінің, мысалы Ада, Модула-2 сияқты тілдердің дереккөзіне айналды.

Модула-2 — Н.Вирт ұсынған, Паскаль тілін дамытушы және үлкен бағдарламалар құру үшін құралдарды жинақтаған тағы бір тіл.

Си бағдарламалау тілі (ағылшынша атауы — С) Паскальмен бір уақытта пайда болды. Ол амалдық жүйелерді, трансляторларды, БЗ және басқа да жүйелік және қолданбалы бағдарламаларды даярлау үшін құрылған инструменталды тіл ретінде құрылды. Си жоғары деңгейлі тіл болып саналғанымен, алайда онда бұрын тек ассемблерде ғана мүмкін болған, кейбір машиналық командаларға, компьютер жадының белгілі бір бөліктеріне тікелей жүгіну мүмкіндіктері сақталған.

Си тілі пайда болғаннан кейін көптеген жүйелік бағдарламашылар ассемблерден Сиге көшті. Сиді одан әрі дамыту объектілі-бағдарланған бағдарламалау тілін құруға әкелді (ОББ) Си++.

Бағдарламалаудың парадигмалары. Бағдарламалауға қолданылатын «парадигма» сөзі компьютерде есептеулерді ұйымдастыруға қойылатын белгілі бір жалпыға ортақ тәсілдемені білдіреді. Парадигма соның негізінде бағдарламалау болатын негізгі ұғымдар жүйесін анықтайды.

Осыдан бұрын сөз болған бағдарламалау тілдері *бағдарламалаудың процедуралық парадигмасына* негізделген. Процедуралық тілде іске асырылған алгоритм, компьютердің фон-неймандық сәулеті туралы ұғымдарға негізделеді. Негізгі түсінік ретінде ЭЕМ жадында сақталатын шаманың түсінігі, негізгі амал ретінде – меншіктеу амалы болады. ЭЕМ-да есеп шығару жадының күйін өзгерту жолымен жүзеге асырылады: алдымен оған бастапқы деректер салынады, соңында – нәтижелер алынады. Бағдарлама — бұл командалар жүйелігін орындау процесінде жадының алғашқы күйден соңғы күйге асысу процедурасын сипаттау. Процедуралық парадигма шегінде жұмыс істейтін бағдарламашының ойлау қабілеті ЭЕМ процессоры орындай алатын есепті шешуді командалар жүйелігіне жинақтауға бағытталған.

Басқа тәсілдемеде бағдарламалауға *бағдарламалаудың функционалды парадигмасы* негізделген. Функционалды парадигманы іске асыратын ең танымал тіл ретінде ЛИСП болады.ЛИСП туралы алғашқы сөз 1958 жылы айтылды. Ол рекурсивті функция ұғымы негізінде құрылды. Теориялық бағдарламалауда кез келген алгоритм рекурсивті функциялардың қандай да бір жиынтығының көмегімен суреттелуге болатыны дәлелденді. Сондықтан ЛИСП, шын мәнісінде, әмбебап тіл болып саналады. Оның көмегімен ЭЕМ-да айтарлықтай күрделі үрдістерді, әсіресе адамдардың интеллектуалды қызметін үлгілеуге болады. ЛИСП-тен басқа функционалды бағдарламалаудың басқа да тілдері бар: Haskell, Scheme және т.б.

Бағдарламалаудың логикалық парадигмасы Пролог тілінде іске асырылды. Пролог 1972 жылы Францияда әзірленді. ЛИСП сияқты, оны да жасанды ақыл-ой тілдеріне жатқызады. Прологматематикалық логика аппаратына негізделген, содан – парадигманың атауы туындайды. Прологта фактілер мен ережелердің жиынтығынан тұратын, белгілі бір пән саласында білім негіздері құрылады. Қандай да бір есепті шешу үшін мақсат деп аталатын білім негізіне сұраныс

тұжырымдалады. Мақсатты іске асыру (сұранысқа жауап) шығарудың механизмі деп аталатын, Прологтың түсініктеме берушісіне салынған алгоритм арқылы жүргізіледі. Прологтың көмегімен жасанды ақыл-ой есептерінің тек шектелген ғана ғана санын шешуге болатынын атап өту керек. Логикалық бағдарламалау тілдеріне сондай-ақ мыналар жатады: Planner, Mercury, Fril және басқалары.

Бағдарламалаудың объектілі-бағдарланған парадигмасы объектілер мен класстардың тұжырымдамасына негізделген.Әрбір объект онымен атқарылуы мүмкін болатын қасиеттер мен әрекеттердің жиынтығымен сипатталады. Объект белгілі бір классқа жатады. Объектіге бағдарланған бағдарламалау класстардың сатыларын қатарлауды, объектілерді және олардың өзара әрекеттерін сипаттауды, объектілерге жүргізілетін түрлі әрекеттерді бағдарламалық іске асыруды қиыстырады.ОББ алғашқы тілі 1967 жылы әзірленген Симула тілі болды. Кейін ОББ элементтері процедуралық тілдерге, соның ішінде ТурбоПаскальға енгізіле бастады. ОББ мейлінше танымал құралдары ретінде СИ++, Java тілдері болды. ОББ туралы толығырақ 3-тарауда айтылады.

Бағдарламалаудың әдіснамасы және технологиясы. Жады «тар» және тезәрекетігі шағын бірінші ЭЕМ үшін бағдарлама сапасының негізгі көрсеткіші ретінде оның жадының алу көлемі және есептеу уақыты бойынша үнемділігін айтуға болады. Бағдарлама неғұрлым қысқа болса, бағдарламашының санаты соғұрлым жоғары саналды.

ЭЕМ жады мен тезәрекеттілігінің, бағдарламалау тілдері мен осы тілдерден таратушылардың жетілдірілуінің артуымен бағдарламаның үнемділік проблемасы тым өзекті болудан қала бастады. Бағдарламалардың неғұрлым сапалық сипаттары ретінде олардың қарапайымдылығы, көрнекілігі және сенімділігі болды, ал үшінші буын машиналары пайда болуымен бұл қасиеттер негізгі қасиеттерге айнала бастады.

1960 жылдары бағдарламалау көпшілікке танымал кәсіби қызметке айналды. Бағдарламаларды әзірлеуші компаниялар (фирмалар) пайда бола бастады. Бағдарламашылардың жұмыс қабілетін арттыратын, ең бастысы, бағдарламалық өнімдердің сапасын арттыратын бағдарламалаудың жалпыға ортақ әдіснамасын әзірлеу міндеті өзекті бола бастады. Бағдарламаның негізгі сапалық көрсеткіші – оның жұмысқа қабілеттілігі, қателердің болмауы.

Бағдарламалаудың әдіснамасы – бағдарламаларды жазудың,

реттеудің және сүйемелдеудің белгілі бір тәсілдерінің жиынтығы. Бағдарламалаудың ең алғашқы аса танымал және кең тараған әдіснамасы *құрылымдық бағдарламалау* деген атауға ие болды.

Құрылымдық бағдарламалаудың пайда болуы Э.Дейкстр және Ч.Хоар аттарымен байланысты. 1960 жылдардан бастап құрылымдық бағдарламалаудың тілдері пайда бола бастады. Олардың ішінде бірінші болып Дейкстр әзірлеген Алгол-60 болды, содан соң Паскаль құрылды. Басқа, бастапқыда «құрылымдық емес» тілдер «құрылымдық қасиеттерге» ие бола бастады (ТурбоБейсик, Фортран-77 және т.с.с). Құрылымдық бағдарламалау осы уақытқа дейін бағдарламалаудың маңызды әдіснамасы болып қала береді. Оның қағидаларын сақтау бағдарламашыға анық, қатесіз, сенімді бағдарламаларды құруға мүмкіндік береді.

Технология — бұл тұрақты нәтижені кепілдендіретін, қандай да бір өнімді өндірудің жаппай тәсілі. Технология –б елгілі бір өндіріс құралдарының, аспаптарының болуын жобалайды. 1990 жылдары объектілік-бағдарланған парадигманың, сондай-ақ ПК графикалық интерфейстің құралдарының дамуымен бағдарламалаудың жаңа технологиясы – визуалды бағдарламалау туындайды. Бағдарламалаудың визуалды технологиясы бағдарламашығаөз бағдарламалары үшін объектілерді экранда графикалық түрде бейнелейтін қалыпдардың стандартты жиынтығының негізінде графикалық көрнекі интерфейсті тез әрі жеңіл құруға мүмкіндік береді. Delphi жүйесінде визуалды бағдарламалау технологиясы туралы 3-тарауда айтылады.

2.2. БАҒДАРЛАМАЛАРДЫ ӘЗІРЛЕУ ӘДІСТЕРІ МЕН ҚҰРАЛДАРЫ

Бағдарламалық өнімді әзірлеу – көпкезеңді үрдіс. Бағдарламашы алгоритм құрып, оны бағдарламалаудың белгілі бір тілінде бағдарламалап қоюды компьютерді қолданбай-ақ, бір парақ қағаз бетінде жүзеге асыра алады. Бірақ содан соң бағдарлама мәтіні компьютердің жадына енгізілуі тиіс. Бұл мақсатта бағдарламаны ASCII-коды пішімінде бағдарламаны сақтайтын қарапайым мәтінді редактор қолданылады. Бағдарламаны дискке жазғанда мәтіндік файл құрылады.

ЖДБТ –де бағдарлама әзірлеудің келесі кезеңі трансляция деп аталады. Трансляцияның түбегейлі айырмашылығы бар екі әдісі бар. Олар сәйкесінше *компиляция* және *интерпретация* деп аталады.

Компиляция кезінде ЭЕМ жадына бағдарлама-компилятор жүкте-

леді. Ол ЖДБТ-де мәтіндік файлдан жүктелетін, *бастапқы модуль* деп аталатын бастапқы ақпарат түріндегібағдарлама мәтінін қабылдайды.Бағдарлама компиляторы ЖДБТ-де өңдегеннен кейін объектілік модуль – машиналық-бағдарланған кодқа көшірілген бағдарлама пайда болады.Бағдарламаны өңдеудің келесі кезеңі байланыстарды түзету немесе линктеу деп аталады. Оны арнайы бағдарлама — байланыстарды түзету бағдарламасы атқарады. Ол бастапқы бағдарламаға оның жұмыс жасауына қажетті бағдарламалық модульдерді: процедураларды, функцияларды және т.б. қосады. Нәтижесінде атқаруға дайын машиналық командалар тіліндегі бағдарлама - *жүктейтін модуль* пайда болады. Содан соң жедел жадыда атқарылатын машиналық командалар бағдарламасы ғана қалады және іздестірілетін нәтижелер алынады.

Интерпретатор бағдарламаның жұмыс жасауының барлық кезеңінде ішкі жадыда болады, және ЖДБТ бағдарламасы ЖЕҚ салынады. Интерпретатор алгоритмді атқару жүйелігінде бағдарламаның кезекті операторын «оқиды», оны командаларға ауыстырады және сол мезетте сол командаларды атқарады. Содан соң келесі операторға ауысу және оны атқару орындалады. Сонымен бірге алдыңғы көшіруліердің нәтижелері жадыда сақталмайды. Бір команданы қайталап атқару барысында од қайтадан трансляцияланады.

Компиляция кезінде бағдарламаны атқару үш кезеңге бөлінеді: компиляция, линктеу және атқару. Трансляция мен атқару бірікгірілгендіктен, интерпрета¬ция кезінде ЭЕМ бағдарламаны өңдеу бір кезеңде жүргізіледі. Алайда, компиляцияланған бағдарлама интерпрецияланған бағдарламаға қарағанда тезірек орындалады, сондықтан тез есептеуді қажет ететін үлкен бағдарламаларға компиляторды қолданған қолайлырақ болады. Паскальдағы, Си Фортрандағы бағдарламалар компилятормен өңделеді. Бейсиктің алғашқы нұсқалары түсінік беру арқылы іске асырылды. Бағдарламаның түсінік берілетін тілдері ретінде Perl, Python болады. Түсінік берудің механизмі: Mathematica, Mathlab, Maple, Mathcad сияқты математикалық пакеттерде қолданылады.

Кейбір тілдерде компилятор арқылы да интерпретор арқылы да өткізуге ие болады. Түсінік беру режимінде бағдарламаны бақылау қолайлы, ал жұмыс есептеулерін компиляцияланған режимде жүзеге асыру тиімдірек.

Көбіне құрылған бағдарламада бағдарламалау тілін бұзумен байланысты қателіктер кездеседі. Транслятор бағдарламашыға бағдар-

лама мәтініне синтаксистік (мазмұндық) және семантикалық талдау жүргізуді жүзеге асыра отырып, мұндай қателіктерді табуға көмектеседі. Бағдарламашы қателіктер туралы хабарлама алады, түзетулер енгізеді және трансляцияны қайталайды.

Линктеу кезеңінде сондай-ақ қате табылуы мүмкін. Олар, әдетте негізгі бағдарламадан түрлі дербес бағдарламалық модульдерге: бағдарламашы әзірлеген стандартты процедуралар мен фунцияларға немесе модульдерге жүгіну кезінде табылған қателіктермен байланысты болады.

2.1.-сурет. ТурбоПаскаль в MSDOS ТурбоПаскальдағы жүйенің интерфейсі

Бағдарламадағы алгоритмдік қателерді іздеу және түзету бағдарламаны ретке келтіру деп аталады. Бағдарламалаудың заманауи жүйелерінде ретке келтірушілер деп аталатын сервистік құралдар болады. Ретке келтірушілер бағдарламашыға бағдарламаның орындалуы үшін бағдарлама мәтінінде апатты үзілген жерді табуға мүмкіндік береді, сонымен қатар мұндай үзілудің себебін табуға көмектеседі.

Кіріктірілген әзірлеу ортасы (КӘО) — бұл бағдарламашының жоғары деңгейлі тілде өз бағдарламаларын әзірлеу үшін қолданатын бағдарламалық құралдар жүйесі. Әдетте әзірлеу ортасына бұрынырақ сипаталған бағдарлама құрудың кезеңдерін атқаратын элементер кіреді:

- мәтіндік редактор;
- транслятор (компилятор немесе интерпретатор);
- байланыстар редакторы;
- ретке келтіруші.

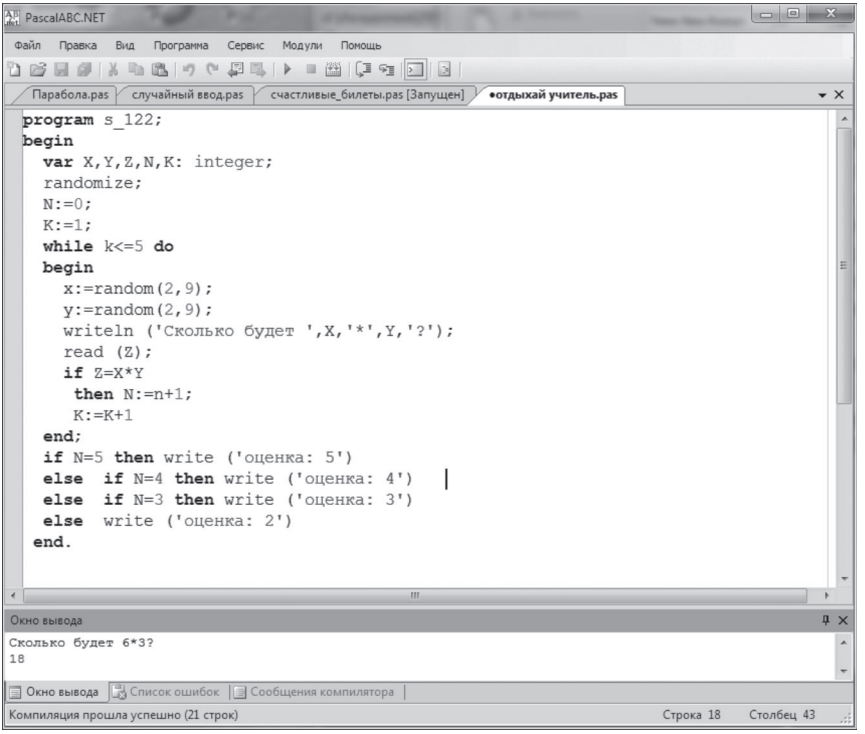
Бастапқыда КӘО бағдарламалаудың нақты бір тілдеріне бағдарланған еді. Мұндай орталардың бірі ретінде Pascal бағдарламалаутілі үшін Borland фирмасымен әзірленген Turbo Pascal жүйесі болды. Түзету тәртіптемесіндегі Turbo Pascal терезесі 2.1.-суретте көрсетілген. Орта MS DOS операциялық жүйесіне бейіпдеп әзірленді, сондықтан оның консолды интерфейсі бар.

Аталған КӘО міндетті элементтерінен басқа, Turbo Pascal-ға тіл бойынша кіріктірілген анықтамалық, стандартты модульдер кітапханасы, файлдармен жұмыс жасау құралдары, көпкезеңді тәртіптемеде жұмыс жасауды қолдау кіреді.

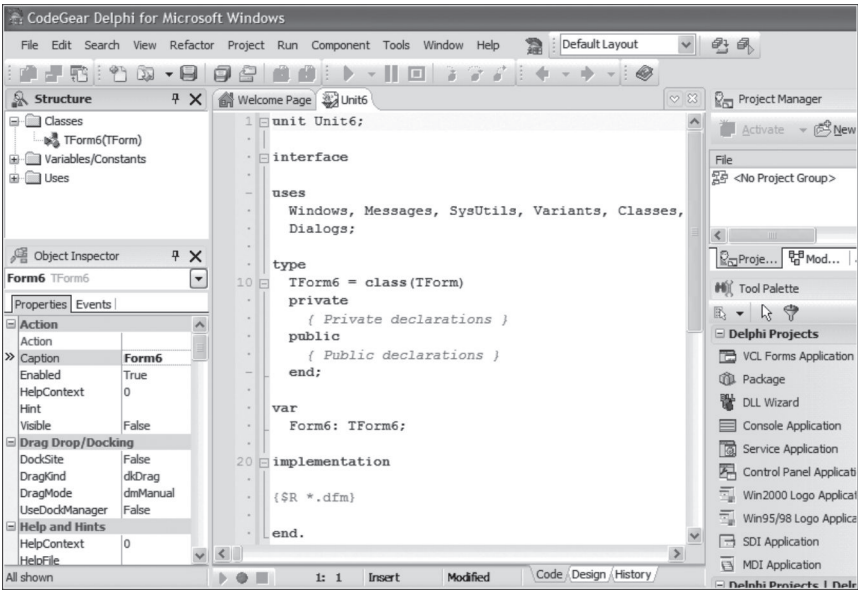
Осы жақын арада Паскальда бағдарламалау үшін КӘО айтарлықтай танымал болған PascalABC жүйесі болды. Берілген жүйе Object Pascal тілі негізінде бағдарламалауды оқыту кезінде қолдануға арналған. Жүйе MS Windows операциялық жүйе негізінде әзірленді және стандартты графикалық интерфейске ие болады (2.2 сурет).

ОББ тілдерін және Windows-қосымшаларды әзірлеудің визуалды технологиясының дамуымен КӘО жаңа түрі пайда болады, олардың мақсаты – терезелік графикалық интерфейс құру үшін бағдарламашыға қолайлы құралдарды ұсыну. Мұндай жүйелердің құрамына класстардың, объектілердің және әдістердің кітапханалары кіреді. Олардың қатарына Object Pascal бағдарламалау тілі негізінде терезелік Windows-қосымшаларды әзірлеуді жүзеге асыратын Delphi ортасы кіреді (2.3 сурет). Delphi аналогы ретінде еркін-таратылатын Lazarus КӘО болады.

Visual Basic ортасы Basic тілінде Windows-қосымшаларды құру үшін қолданылады.



2.2.сурет.PascalABC жүйесінің интерфейсі



2.3. –сурет. Delphi жүйесінің интерфейсі

2000 жылдары бағдарламалаудың кіріктірілген ортасының жаңа түрі дами бастады: NET Framework платформасы. Оның басты артықшылығы – берілген жүйемен қолдау табатын, бағдарламалаудың түрлі тілдерінде жазылған, модульдерден бағдарламалық өнім, сонымен қатар тілдерге қатысты жалпы, өзгермейтін кітапханалардың класстарын, әдістерін және объектілерін құру мүмкіндігінің болуы. Берілген платформа үшін кең таралған әзірлеу орталарының бірі ретінде Microsoft Visual Studio болды. NET- технологиялар платформасында PascalABC. NET бағдарламалаудың кіріктірілген орталарының жаңа нұсқасы жұмыс істейді.

2.3. ЖОҒАРЫ ДЕҢГЕЙЛІ БАҒДАРЛАМАЛАУ ТІЛДЕРІН

Бағдарламалаудың барлық тілдерінде деректерді ұйымдастыру тәсілдері мен олармен әрекет ету анықталды. Бұдан басқа, құрамына көптеген таңбалар (алфавит), лексемалар және бағдарламалаудың өзге де бейнелеу құралдары кіретін тіл элементтері болады. Бағдар-

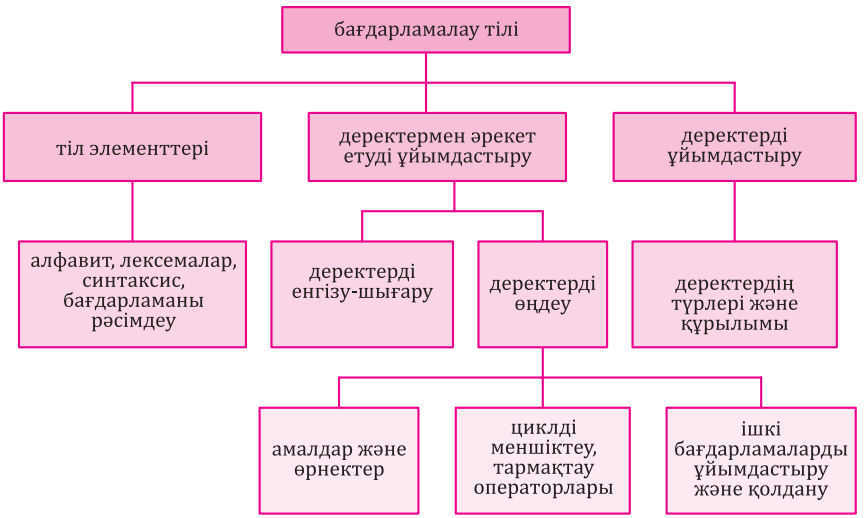
ламалау тілдерінің алуан түрлігіне қарамастан, оларды оқып-үйрену шамамен бір схема бойынша жүргізіледі. Бұл жоғары деңгейлі тілдердің құрылымының ортақтығымен шартталған (2.4-сур.).

Берілген оқу құралында Паскаль және Delphi тілдерін сипаттау осы схемаға сәйкес орындалады.

Табиғи тілдер мен бағдарламалау тілдері арасындағы ұқсастықты атап өткен жөн. Біріншіден, шет тілінде оқып, жазу үшін сол тілдің *алфавитін* білу керек. Екіншіден, сөздер мен сөйлемдерді жазу ережелерін, яғни тілдің *синтаксисін* білу керек. Үшіншіден, оларға сәйкес әрекет ету үшін сөздер мен фразалардың мазмұнын түсіну керек. Мысалы, ұшақтың салонында «Fasten belts!» (Белбеуді тағыңыз!) көрсеткіш тақтасы жанды. Ағылшын грамматикасын біле тұра, бұл фразаны дұрыс оқуға, бірақ оның мағынасын түсінбеуге болады, және сәйкесінше, тиісті әрекеттерді орындамауға болады. Сауатты жазылған сөздерден мүлде мағынасыз фразаны құруғаболады. Тіл құрылымының мағыналық мазмұны *семантика* деп аталады.

Бағдарламалаудың кез келген тілі үш негізгі құрамдасты: алфавит, синтаксис және семантиканы құрайды.

Бағдарламалау тілінде ережелерді сақтау ауызекі сөйлеу тіліне қарағанда қаталырақ болуы тиіс. Адам тілінде артық ақпараттың айтарлықтай зор көлемі болады, яғни қандай да бір сөзді естімей, фразаның тұтас мағынасын түсінуге болады.



2.4. –сурет. Жоғары деңгейлі бағдарламалау тілінің құрылымы.

Тыңдаушы немесе оқушы адам қабылданатын мәтінді өзі әбден ойлап, толықтырып, қателерін түзей алады.

Компьютер болса — бұл бәрін «шындық» ретінде қабылдайтын автомат. Бағдарламалардың мәтінінде еш артықтық жоқ және компьютер тіпті көзге көрінетін (адамның көзқарасымен) қатені де өзі түзете алмайды. Ол тек өзі «түсінбеген» жерді көрсетуі және мүмкін болатын қатенің сипаты туралы ескертуді көрсете алады. Қатені тек бағдарламашы түзейді.

Бағдарламалау тілінің синтаксисін сипаттау үшін басқа тілдерді сипаттауға арналған, қандай да бір тіл, яғни метатіл (тілүсті) қажет. Бағдарламалау бойынша әдебиеттерде аса танымал метатл ретінде Бэкус-Наур формулары (БНФ тілі) және синтаксистік диаграммалар болады. Бұдан әрі біз негізінен синтаксистік диаграммалар тілін қолданатын боламыз, өйткені олар айтарлықтай көрнекі және оңай қабылданады. Алайда, қолайлы болған жағдайда, БНФ тілінің кейбір элементтерін де енгізетін боламыз.

БНФ –да кез келген синтаксистік ұғым мағынасы «анықтама бойынша бар» фразасымен бірдей болатын, «::=» белгісімен жалғанған, оң жақ және сол жақ бөліктерден тұратын формула болады. Формуланың сол жақ бөлігі <>бұрыштық жақшаларға алынған, анықталатын ұғымның атынан (метаайнымалыдан), ал оң жақ бөлігі – мета айнымалыны қабылдай алатын мәндердің барлық жиынтығын анықтайтын, формуладан немесе диаграммадан тұрады. Тілдің синтаксисі ұғымдарды жүйелі күделендіру арқылы қалыптасады, яғни алдымен қарапайым ұғымдар, содан соң құрамдас ретінде алдыңғы ұғымдар кіретін, бірте-бірте біршама күрделі ұғымдар анықталады. Мұндай жүйелікте соңғы анықталатын ұғым «бағдарлама» болатыны мәлім.

Метаформулаларды жазу кезінде белгілі бір келісімдер қабылданады. Мысалы, «екілік цифр» ұғымын анықтайтын БНФ, мынадай түрде беріледі:

<екілік цифр> : : = 0|1

2.4. ПАСКАЛЬ ТІЛІМЕН АЛҒАШҚЫ ТАНЫСУ

Паскаль бағдарламалау тілінің тарихы 2.1.-тармақшасында айтылды. 40 жылдан аса уақыт ішінде бұл тіл дамуға ұшырады. Паскальдің бірінші стандартын оны құрушы Н.Вирт сипаттады.

Borland фирмасы ТурбоПаскаль деген атаумен өз нұсқасын құрып,

тілді кеңейтті. Тіл оқу бағдарламасы шегінен шығып, практикалық бағдарламалау үшін кеңінен қолданыла бастады. Бұғанбағдарламашылар үшін қолайлы болған MS DOC операциялық жүйе негізінде жұмыс істеген, әзірлеменің кіріктірілген жүйесі ықпал етті. Тілдің келесі нұсқасы - Object Pascal - Windows арқылы жұмыс істейтін, Delphi визуалды бағдарламалаудың кіріктірілген орта негізіне енді. Паскаль тілінің әрбір келесі нұсқасы алдыңғы нұсқаның барлық мүмкіндіктерін қолдайды. Object Pascal негізінде Ресейдің оқу орындарында кеңінен тараған PascalABC оқу жүйесі құрылды.

Оқулықтың бұл тарауында Паскаль тілін сипаттау құрылымдық бағдарламалау әдістемесін қолдайтын,ТурбоПаскаль тұрақты стандартының негізінде жүргізілетін болады. Сондықтан мұндағы берілген бағдарламалар PascalABCда, сонымен бірге Turbo Pascal-бағдарламалау орталарында да, сондай-ақ Delphi немесе Lazarus бағдарламалау ортасының консолды тәртіптемесінде атқарылатын болады.

Паскаль тілінде бағдарламалаудың құрылымы. Паскаль стандартты тілінің анықтамасы бойынша бағдарлама соңында бағдарлама соңының белгісі – нүктесі бар тақырыптан және денеден (блоктан) тұрады. Өз кезегінде блокта сипаттама бөлімдері және операторлар бөлімі болады:

Program <бағдарламаның аты>;
Label <белгілер бөлімі>;
Const <константа бөлімі>;
Type <типтердің бөлімі>;
Var <айнымалылар бөлімі>;
Procedure (Function) <ішкі бағдарламалар бөлімі>;
Begin
<операторлар бөлімі>
End.

Операторлар бөлімі әрбір бағдарламады болады және ол негізгі болып саналады. Сипаттау бөлімдерінің барлығы әрбір бағдарламада бола бермейді.

Стандартты тілмен салыстырғанда ТурбоПаскальда, мүмкін болады:

- бағдарлама тақырыбының болмауы;
- **Const**, **Type**, **Var**, **Label** бөлімдерінің суреттеу бөлімінде бірінің артынан бірі кез келген бағытта және бірнеше рет келуі.

Бағдарламалар мысалдары. Жоғарыда сөз болғандай,Паскаль түрін Н.Вирт оқу тілі ретінде әзірлеген. Онда берілген негізгі қағида - бағдарламалаудың құрылымдық әдістемесін қолдау.

Осы қағидаға бұл жерде алгоритм тілі деп атайтын жалған код сүйенеді. Шын мәнінде, АТ мен Паскаль тілінің арасындағы айырмашылық мынада болады: АТ — орыс тілінде, Паскаль — ағылшын тілінде; Паскаль тілінің синтаксисі қатаң түрде және бір мәнді анықталған, ал АТ синтаксисі – біршама еркін түрде бірлген.

Паскаль тіліндегі бағдарламаның жазбасы алгоримт тілінде жазылған алгоритмнің ағылшын тіліндегі аудармасына ұқсайды. АТ жазылған жай бөлшектерді бөлу алгоритмін Паскальдағы тиісті бағдарламада жазылғанмен салыстырыңдар:

```
алг бөлшектерді бөлу
тұт a, b, c, d, m, n
бас енгізу a,b,c,d
    m : = a * d
    n : = b * c
    шығару m, n
соң
```

```
Program Division;
Var a,b,c,d,m,n : Integer;
Begin ReadLn(a, b, c, d);
    m : = a * d;
    n : = b * c;
WriteLn(m, n)
End.
```

Бұл жерде келесі математикалық теңдік ескерілген:

$$\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c}.$$

Бұл бағдарламаны Паскаль тілінің кітабына қарамай-ақ, әсіресе ағылшын тілін біле тұрып, түсінуге болады.

Бағдарламаның тақырыбы бағдарламашы ойлап тапқан (Division— бөлу) ерікті түрдегі есім келетін, **Program** (бағдарлама) сөзінен басталады. Айнымалыларды сипаттау бөлімі содан кейін айнымалылардың тізімі келетін **Var** (variables — айнымалылар) сөзінен басталады. Түрі екі нүктеден соң **Integer** (тұтас) сөзі арқылы көрсетіледі. Бағдарлама операторлары бөлімінің басы мен соңы **Begin** (басы) и **End** (соңы) сөздерімен белгіленеді. Бағдарлама соңында *міндетті түрде нүкте қойылады*.

Пернетақтадан бастапқы деректерді енгізу ReadLn (read line — жолды салыстырып оқу) процедурасының көмегімен жүргізіледі. Пернетақтада дисплей экранында көрсетілетін, бір-бірінен аралық арқылы бөлінетін төрт санды тереді, және енгізу пернесін басады.

Меңгеру операторлары Паскальда АТ-дегідей жазылады. Көбейту

таңбасы жұлдызшамен (*) белгіленеді.

Дисплей экранына нәтижелерді шығару жолға m жәнеn екі тұтас санды шығаратын WriteLn (*write line*— жолға жазу) процедурасының көмегімен жүргізіледі. Содан соң экрандағы меңзер келесі бос жолдың басына ауысады, және бағдарламаның жұмысы аяқталады.

Бағдарламаның синтаксисі -дұрыс жазу ережелерін қатаң сақтау қажет. Әсіресе, Паскальда тыныс белгілерінің мақсаты қатаң түрде анықталған:

- нүктелі үтір (;) бағдарлама тақырыбының соңында, айнымалыларды суреттеу бөлімінің соңында, әрбір оператордан соң қойылады. **End** сөзінің алдында нүктелі үтірді қоймауға болады;
- үтір (,) әр түрлі тізімдерде (суреттеу бөлімінде айнымалылар тізімінде, енгізілетін және шығарылатын шамалар тізімінде) элементтерді бөлгіш болып табылады.

Бағдарламалау тілінде қатаң синтаксистің болуы, ең алдымен, транслятор үшін қажет. Транслятор — бұл ресми орындалатын бағдарлама. Егер айнымалылар тізімінде бөлгіш ретінде үтір болуы керек болса, онда кез келген өзге таңба онда қате ретінде қабылданады. Егер нүктелі үтір операторларды бөлгіш болатын болса, онда транслятор оператор ретінде бағдарлама мәтінінің барлық бөлігін бір нүктелі үтірден екіншісіне дейін тұтас қабылдайды. Сәйкесінше, егер қандай да бір екі оператордың арасына нүктелі үтір қойылмаса, онда транслятор оларды біреу ретінде ғана қабылдайтын болады, яғни қате жіберілуі әбден мүмкін.

Синтаксистік ережелердің негізгі мақсаты – тілдік құрылымдарға бір мәнді мағына беру болып табылады. Егер қандай бір құрылым екі түрлі мағынада берілетін болса, онда бір қатебар деген сөз. Сондықтан ішкі түйсікке сенбей, тіл ережелерін оқып-білу керек.

Бұдан әрі Паскальдің синтаксис ережелерін қатаң түрде сипаттау беріледі, әзірше тіл туралы бастапқы түсінік алу үшін күрделі емес алгоритмдерді бағдарламалаудың бірнеше мысалын қарастырайық. .

Факториалды ($N!$) есептеу алгоритмін Паскаль тіліне «транслциялайық»:

алг факториал	**Program** Factorial;
тұт N, I, F	**Var** N, I, F: Integer;
бас енгізу(N);	**Begin** ReadLn(N);
F := 1	F := 1

```
I : =1
әзірше  I <= N
цб      F := F * I
        I := I + 1
цс
шығару F
соңы
```

```
I := 1;
While I <= N Do
Begin F := F * I;
    I := I + 1
End;
WriteLn(F)
End.
```

Бұл мысалдан біріншіден, шарталды циклі операторының («Әзірше» циклі) Паскальда қалай жазылатыны көрінеді:

While<орындау шарты>**Do**<циклдің денесі>

While— әзірше, **Do**— жасау. Егер циклдің денесінде операторлардың жүйелігі болса, онда ол *құрама оператор* құрады деп айтады, оның басында және соңында **Begin** және **End** деп жазу керек.**Begin** және **End** көмекші сөздерді көбінебірнеше операторларды бір құрамаға біріктіретін *операторлық жақшалар* депте атайды.Егер циклдің денесі — біроператорболатын болса (құрама емес), онда операторлық жақшалар қойылмайды. Онда транслятор циклдің денесі ең жақын белгіде «;» аяқталады деп есептейді.

Екіншіден, мысалдан Паскальда циклдің басын (**цб**) және циклдің соңын (**цс**) белгілейтін арнайы сөздердің жоқ екендігі көрінеді. Бұл жағдайларда **Begin** және **End** көмекші сөздер қолданылады.

Квадрат теңдеуді шешу бағдарламасының тағы бір мысалын қарастырайық:

```
алг түбірлер;
зат a, b, c, d, x1, x2
бас

    қайталау
    шығару «a,b,c енгізіңіз;
        a ≠ 0»)
    енгізу a, b, c
    a  дейін ≠  0
    d:= b^2 - 4ac

    егер d ≥ 0
    онда
    x1:=(-b + sqrt(d))/(2a)
    x2:=(-b - sqrt(d))/(2a)
```

```
Program Roots;
Var a,b,c,d,x1,x2 : Real;
Begin{деректерді бақылаумен
    енгізу}
    Repeat
    WriteLn('Введите a, b, c;
        a<>0');
      ReadLn(a,b,c)
    Until a <>0;
{Дискриминантты есептеу}
    d := b*b - 4*a*c;
    If d >= 0
Then Begin {түбірбар}
x1:=(-b + sqrt(d))/2/a);
x2:=(-b - sqrt(d))/2/a);
```

```
    шығару x1, x2                    WriteLn('x1 =', x1
                                     'x2 =', x2);
    басқаша шығару «заттық             End
                 түбірлер жоқ »-      ElseWriteLn('заттық түбірлер жоқ')
    кв
соң                                  End.
```

Бұрын мысал келтірілгендермен салыстырғанда, бұл бағдарламада көптеген жаңа элементтер пайда болды. Паскаль — Real деректердің заттық түрінің аты.

Шарттан кейінгі цикл («Дейін» циклі) операторымен бағдарламанады

Repeat <циклдің денесі> **Until** <аяқталу шарты>

Repeat — қайталау, **Until** — дейін. Циклдің денесі дара да, сондай-ақ құрама оператор түрінде де болуы мүмкін, алайда **Begin** және **End** сөздерін қолданудың қажеттігі жоқ, өйткені **Repeat** және **Until** сөздері операторлық жақшалардың рөлін өздері атқарады.

≠ (тең емес) белгісі Паскальда <> белгісін білдіреді, ал ≥ (артық немесе тең) белгісі >= түрінде белгіленеді.

Арифметикалық өрнектерді жазу ережесі толығырақ кейін қарастырылатын болады. Түбірлерді есептеу формулаларында қолданылатын квадрат түбірдің стандартты функциясы (), паскальда sqrt (*x*) түрінде жазылады. Өрнектерде амалдарды орындау тәртібі жақшалармен және амалдардың үлкендігімен анықталады. Амалдар үлкендігі алгебрадағы сияқты анықталады. Үлкендігі бойынша бірдей оларды жазу тәртібі бойынша орындалады (солдан оңға қарай).

Паскальда тармақталу келесі форма түрінде берілген *шартты операторды*ң көмегімен бағдарламаланады:

If <шарты> **Then** <1 оператор > **Else** <2 оператор >

If— егер, **Then**— онда, **Else**— басқаша. 1 және 2 операторлар дара да, құрама да бола алады. Құрама операторды **Begin** және **End** операторлық жақшаларға алу керек.

Алгоритмдік тілдерде сияқты шартты операторды нысанын толықтай емес қолдану мүмкін:

If <шарты>**Then** <оператор>

Берілген бағдарламаның өзіне тән ерекшелігі ретінде мәтінде түсініктемелерді қолдану болады.

Түсініктеме — бұл фигуралық жақшаларға алынған {...} таңбалардың кез келген жүйелігі. Сондай-ақ түсініктемелердің шектегіші ретінде жұлдызшасы бар дөңгелек жақшаларды (*...*) қолдануға болады. Түсініктеме бағдарламаның ешқандай әрекетін анықтамайды, тек түсініктеме мәтін ретінде қолданылады. Ол аралық қоюға болатын бағдарламаның кез келген жерінде берілуі мүмкін. Бағдарламашы түсініктемені компьютер үшін емес, бағдарлама мәтіні мейлінше түсінікті болу үшін, өзіне арнап жазады.

Жақсы түсініктеме берілген бағдарламаларды өзін-өзі құжаттаған бағдарлама деп атайды. Кейбір бағдарламаларда түсініктемелердің көлемі есептеу операторларының санынан асып кетеді.

Түсініктемелерді тиімді қолдану – бағдарламалаудың жақсы стиліниңбелгісі.

Бағдарламаны ЭЕМ-да орынду үшін оны *жадыға енгізу, трансляциялау* және *атқару* қажет. Ол үшін компьютерде ПК-де ТурбоПаскаль немесе Pascal ABC және т.б. жүйесін құрайтын, бағдарламалық қамсыздандырудың арнайы құралдары болуы тиіс.

Жаттығулар

АТ-ден Паскальға «трансляциялау» керек:

а) Евклид алгоритмін;

б) үшеуінен ең үлкен мәнді таңдау алгоритмін;

в) қабырғаларының ұзындықтары берілген үшбұрыштың болуын анықтау алгоритмін;

г) қосу және азайту амалдарымен ғана шектеліп, екі тұтас санды көбейту алгоритмін;

д) тұтас санды бөлуден бөлінді мен қалдықты есептеу алгоритмін.

2.5. ТІЛ ЭЛЕМЕНТЕРІ ЖӘНЕ ДЕРЕКТЕР ТҮРЛЕРІ

Алфавит. ТурбоПаскаль тілінің алфавитіне әріптер, таңбалар, цифрлер және арнайы символдар кіреді:

- *A* -дан *Z* дейінгі(бас әріптер) және *a*-дан *z* дейінгі (кіші әріптер) латын әріптері;
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 цифрлері;
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F ан алтылық цифрлер;

■ + - * / = <> [ ] . , ( ) : ; { } @ $ # арнайы таңбалар.

Арнайы таңбалардың қиыстырулары аралықтармен бөлуге болмайтын, тұтас таңбалар болып саналады:

:= меңгеру белгісі;

>= артық немесе тең;

<= кем немесе тең;

<> тең емес;

(* *) түсініктемелерді шеттегіштер ({ } қатар);

(. .) балама [ ].

Аралықтар — бұл аралық таңбасы (ASCII-32) және барлық басқарушы кодтың таңбалары ASCII(0 -ден 31 дейін).

Арнайы таңбаларға сондай-ақ мағынасы бір мәнде анықталған және басқа мақсаттар үшін қолданылмайтын *көмекші сөздер* жатады. Тіл үшін – бұл бірыңғай таңбалар.

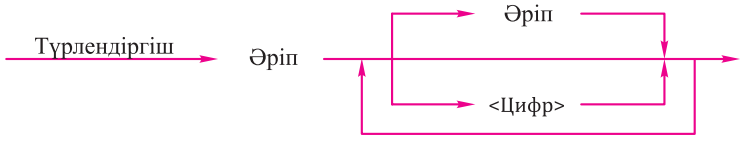
ТурбоПаскаль тілінің көмекші сөздері:absolute, and, array, be-gin, case, const, div, do, downto, else, end, external, file, for, forward, function, goto, if, implementation, in, inline, interface, interrupt, label, mod, nil, not, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor.

ТурбоПаскаль тілінің соңғы нұсқасында объектілермен жұмыс жасауға және кіріктірілген ассемблерге жататын бірқатар көмекші сөздер бар.

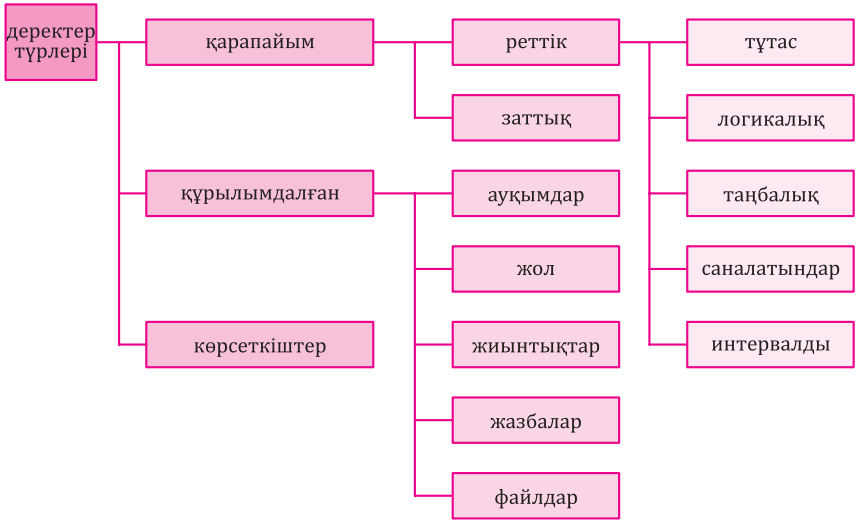
Түрлендіргіштер. Түрлендіргіш деп белгілі бір бағдарламалық объектінің: константаның, айнымалылардың, деректер түрлерінің, процедураның, функцияның, бағдарламаның таңбалық атауын айтады. Синтаксистік диаграмманың көмегімен түрлендіргішті 2.5.суретте көрсетілген түрде беруге болады.

Диаграмманың шифрін былай ашуға болады: түрлендіргіш – бұл әріптен басталатын әріптер мен цифрлардың кез келген жүйелігі. ТурбоПаскальда астын сызу таңбасыда әріптерге жатады.

Бас және кіші әріптердің түрлендіргіштер мен қызметтік сөздерде айырмашылығы жоқ. Мысалы, max, MAX, MaX, mAx — бұл бір атауды білдіреді.



2.5. –сурет. Түрлендіргіштің синтаксистік диаграммасы



2.6.-сурет. ТурбоПаскальдың деректер түрлерінің құрылымы.

Түсініктемелер. Келесі түрдегі құрылымдар түсініктемелерден тұрады және сондықтан компилятор арқылы ескерілмейді:

{«}»} таңбасы жоқ кез келген мәтін
(*«*)»*)таңбалары жоқ кез келген мәтін

Орыс алфавитінің әріптері тек түсініктемелерде, литерлік және мәтіндік константаларда ғана қолданылады.

«{$» немесе «(*$» таңбаларынан басталатын жол *компилятордың директивасы* болып табылады Бұл таңбалардан кейін компиялтор командасының мнемоникасы келеді.

Деректер түрлерінің тұжырымдамасы. Бұл тұжырымдама бағдарламалаудың кез келген тілінде негізгілердің бірі болып табылады. Шама түрімен оның үш қасиеті байланысты: *ішкі көрінісінің формасы, қабылданатын мәндердің жиынтығы және рұқсат етілетін амалдар жиынтығы.* ТурбоПаскаль деректер түрінің сан түрлігімен сипатталады (2.6-сурет).

Стандартты Паскальда деректердің жолдық түрі болмайды. Бұдан басқа, ТурбоПаскальда тұтас және заттық – бұлар деректер түрлерінің топтары. ТурбоПаскальдың кейініректегі нұсқаларында деректердің процедуралық түрі және «объекті» деректер түрі болады.

Деректердің әрбір түрінің өз түрлендіргіші болады.

2.2. кесте. ТурбоПаскальдың деректер типі

Түрлендіргіш	Ұзындығы, байт	Мәндердің диапазоны(жиынтық)
Тұтас		
Integer	2	-32768..32767
Byte	1	0..255
Word	2	0..65535
Shortint	1	-128..127
Longint	4	-2147483648..2147483647
Заттық		
Real	6	$2{,}9 \cdot 10^{-39}$—1,7 · 1038 (11—12)
Single	4	$1{,}5 \cdot 10^{-45}$—3,4 · 1038 (7—8)
Double	8	$5 \cdot 10^{-324}$—$1{,}7 \cdot 10^{308}$ (15—16)
Extended	10	$3{,}4 \cdot 10^{-4932}$—$1{,}1 \cdot 10^{4932}$ (19—20)
Логикалық		
Boolean	1	True, False
Таңбалық		
Char	1	ASCII кодының барлық таңбалары

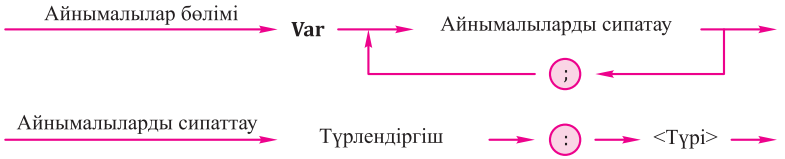
ТурбоПаскальда анықталған деректердің қарапайым түрлері туралы ақпарат 2.2.кестеде берілген. Деректердің заттық түрлері үшін жақша ішінде санның ондық берілуіндегі мантиссаның сақталатын маңызды цифрларының саны көрсетілген.

Стандартты Паскальда заттық түрлерден тек Real түрі, ал тұтастардан - Integerтүрі анықталған.

Деректердің Single, Double, Extended түрлері Паскаль-бағдарламаларда тек ДК «қалқымалы арифметиканың» қосымша процессорымен жабдықталған жағдайда ғана қолданылады (Intel-80486 және одан әрілерден бастап, IBM PC процессорлары үшін, бұл шарт әрдайым орындалады)

Деректер түрі егер нөмірлеуге болатын мәндердің салыстырып санау сандарынан тұратын болса, *реттік* деп аталады. Бұдан мәндердің бұл жиынтығы үшін «келесі» және «бұдан бұрынғы» ұғымдары болатынын көруге болады.

Айнымалыларды сипаттау. Бағдарламада қолданылатын барлық айнымалы шамалар үшін айнымалылар бөлімінде олардың түрлері көрсетілуі керек. Айнымалылардың **бөлімінің құрылымы** 2.7. суретте берілген.



2.7.-сурет. Айнымалылар бөлімінің құрылымы

Бағдарламаның айнымалылар бөлімінің мысалы:

```
Var m, n, k : Integer;
    x, y, z : Real;
    Symbol : Char;
```

Константалар. Константа түрі мәнмәтін бойынша, яғни оны бағдарламада жазу нысаны бойынша анықталады.

Тұтас ондық костанталардытұтас санның кәдімгі нысанында таңбамен немесе таңбасыз жазады. Мысалы: 25, -24712, 376.

Тұтас он алтылық константаларды «$» префиксімен жазады. Олар $00000000 -ден$FFFFFFFF дейінгі ауқымда болуы тиіс.

Белгіленген нүктемен берілген заттық константаларды бөлшектік бөлігі бар ондық санның кәдімгі нысанында жазады. Тұтас және бөлшек бөліктердің айырмасы ретінде нүкте болады. Мысалы: 56.346, 0.000055, -345678.0.

Қалқымалы нүктемен берілген заттық константалардың келесі нысаны болады:

<мантисса>E<тәртіп>

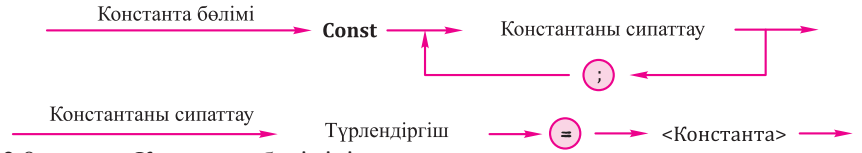
Мұндағы<мантисса> — белгіленген нүктемен берілген тұтас немесе заттық сан, <тәртіп> — таңбамен немесе таңбасыз берілген тұтас сан. Мысалы: 7E-2 (... 10^{-2}), 12.25E6 (12,2...10^{6}), 1E-25 (10^{-25}).

Таңбалық константа — алфавиттің апострофқа (дәйекшеге) алынған кез келген таңбасы. Мысалы: 'W', '!', '9'.

Логикалық константа — бұл: True, False сөздері.

Жолдық константа— бұл апострофқа (дәйекшеге) алынған таңбалар жолы. Мысалы: 'TurboPascal', ' Жауабы: ', ' 35-45-79'. Жолдық константаның максималды ұзындығы 255 таңба.

Константаға тағайындалуы бағдарлама константалары бөлімінде берілетін, белгілі бір есім сәйкес келуі мүмкін.



2.8. –сурет. Константа бөлімінің құрылымы

```
Const
    Max= 1000;
    G= 9.81;
    Cod = 'Қате';
```

Константа бөлімінің құрылымы 2.8.-суретте берілген. ТурбоПаскальда сондай-ақ *типтелген константаларды* қолдануға болады. Типтелген константа алдымен бастапқы мән берілетін айнымалыларға ұқсас болады. Және бұл компиляция кезеңінде жүзеге асырылады. Мысалы:

```
Const   NumberCard   :   Integer  =  1267;
                Size   :   Real     =  12.67;
              Symbol   :   Char     =  '*';
```

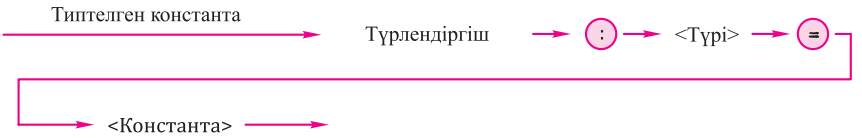
Типтелген константаны сипаттау 2.9.-суретте берілген.

ТурбоПаскальда бағдарламада алдын ала анықтаусыз қолдануға болатын белгілі бір константалар мәндеріне қатысты кейінге сақталған бірқатар аттар бар (2.3.-кесте).

Пайдаланушы деректерінің түрлері. Паскаль тілінің түбегейлі сәттерінің бірі - бұл пайдаланушыға деректердің өз түрлерін анықтауға рұқсат беру. Сонымен бірге пайдаланушының деректерінің түрлері Паскаль деректерінің стандартты түрлеріне негізденеді.

Пайдаланушының деректер түрін сипаттау үшін Паскальда құрылымы 2.10.-суретте берілген түрлер бөлімі бар.

Түгендеу түрлерінің деректері (2.11.-сурет) берілген түрдің айнымалысы қабылдай алатын барлық мәндерді түгендеу арқылы беріледі.



2.9.-сурет. Типтелген константаны сипаттау

2.3-кесте. ТурбоПаскальдың кейінге сақталған константалары		
іш	Түрі	Мәні
True	Boolean	АҚИҚАТ
False	Boolean	ЖАЛҒАН
MaxInt	Integer	32767

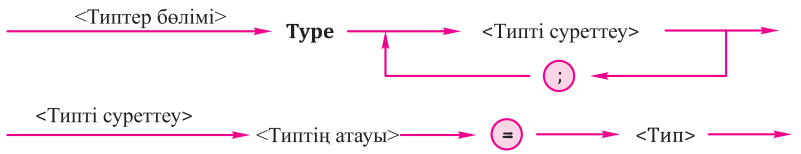
Деректер түрінің анықталған аты айнымалыларды суреттеуде қолданылады. Мысалы:

```
Type Gaz = ( C, O, N, F) ;
     Metal = (Fe, Co, Na, Cu, Zn) ;
Var  G1, G2, G3: Gaz;
     Metl, Met2: Metal;
     Day:(Sun, Mon, Tue, Wed, Thu, Fri, Sat;
```

Мұндағы Gaz және Metal—G1, G2, G3 и Met1, Met2 айнымалыларға үйлесімге қойылатын, түгенделетін деректер түрлерінің аттары. Day айнымалысына ат берілмеген түгенделетін деректер түрі тағайындалады.

Түгенделетін деректер түріне кіретін мәндер константалар болып табылады. Олармен жасалатын әрекеттер константаларға қолданылатын ережелерге бағынады. Түгенделетін типтегі әрбір мән жадыда 2 байтты алады, сондықтан элементтер саны 65535 аспауы тиіс.

Түгенделетін деректер түрі— тәртіптелген жиынтық. Оның элементері 0-ден бастап сипаттауда берілген тәртіп бойынша нөмірленген.



2.10.-сурет. Деректер типтері бөлімінің құрылымы

2.11.-сурет. Түгенделетін деректер типін суреттеу

2.12. –сурет. Деректердің интервалды типін сипаттау

Бұрын берілген сипаттамасы бар бағдарламада келесі үзіндінің болуы мүмкін:

```
If Day = Sun Then WriteLn('Алақай! Бүгін демалыс!');
```

Интервалды типтің деректері (2.12-сурет) кейбір реттік типтің реттелген шектеулі ішкі жиынтығы ретінде беріледі.

Бірінші константаның реттік нөмірі тиісті базалық негізгі деректерде екінші константаның нөмірінен артық болмауы тиіс.

Бағдарламаны атқару кезінде интервалды типтегі айнымалының мәндерінің орнатылған диапазонға тиесілі болуы автоматты түрде бақыланады. Диапазоннан шығу кезінде бағдарламаны атқару үзіледі. Мысалы:

```
Type  Numbers = 1..31;
      Alf = 'A'..'Z';
Var   Data: Numbers;
      Bukva: Alf;
```

2.6. АРИФМЕТИКАЛЫҚ АМАЛДАР, ФУНКЦИЯЛАР, ӨРНЕКТЕР. МЕНШІКТЕУ ОПЕРАТОРЫ

Деректердің арифметикалық типтеріне деректердің заттық және тұтас топтары жатады. Оларға арифметикалық амалдар және қатынас амалдары қолданылады.

Деректерге қолданылатын амалдар *унарлы* (бір операндаға қолданылатын) және *бинарлы* (екі операндаға қолданылатын) болып бөлінеді.

Бір унарлы арифметикалық амал – бұл келесі пішімдегі таңбаны өзгерту амалы:

- <шама>.

Паскаль стандартты тілінің бинарлы арифметиалық амалдары 2.4.-кестеде берілген, мұндағы I — тұтас, ал R — заттықтиптің деректері.

2.4.-кесте. Паскальдың бинарлы амалдары

Белгі	Өрнек	Операнд типтері	Нәтиже типі	Амал
+	A + B	R, R I, I I, R; R, I	R I R	Қосу
-	A - B	R, R I, I I, R; R, I	R I R	Азайту
*	A * B	R, R I, I I, R; R, I	R I R	Көбейту
/	A/B	R, R I, I I, R; R, I	R R R	Заттық бөлу
div	A div B	I, I	I	Тұтас бөлу
mod	A mod B	I, I	I	Тұтас бөлудің қалдығы

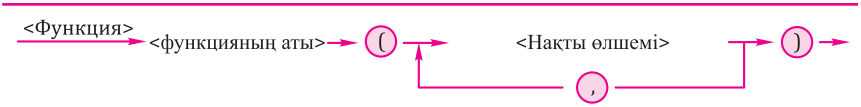
Арифметикалық амалдарға Паскаль тілінің стандартты функциялары қолданылулары мүмкін. Функцияға жүгіну құрылымы 2.13.-суретте көрсетілген.

Өрнектегі функция операнд түрінде беріледі. Мысалы: меншіктеу операторында

X := 2 * sin(A)/ln(3.5) + cos(C - D)

операнд ретінде математикада сияқты жазылатын үш функция: sin, ln, cos беріледі. Аргументтер нақты параметрлер деп аталады, ортақ жағдайда арифметиалық типтегі өрнек түрінде болады және дөңгелеке жақшалардың ішінде жазылады. Функцияны есептеу нәтижесі ретінде тиісті типтің шамасы болады.

ТурбоПаскальдың стандартты математикалық функцияларын сипаттау 2.5.-кестеде берілген.



2.13.-сурет. Функцияға жүгінудің құрылымы

2.5.-кесте. ТурбоПаскальдың стандартты математикалық функциялары

Жүгіну	Аргумент типі	Нәтиже типі	Функция
Pi	—	R	π саны= 3.1415926536E+00
abs (x)	I, R	I, R	x аргументінің модулі
arctan (x)	I, R	R	Арктангенс *x*, рад
cos (x)	I, R	R	Косинус *x*, рад
exp (x)	I, R	R	e^x— экспонентатің *x*
frac (x)	I, R	R	Бөлшек бөлігі *x*
int (x)	I, R	R	Тұтас бөлігі *x*
ln (x)	I, R	R	Натуралды логарифм *x*
random		R	Интервалдағыы жалған кездейсоқ сан [0, 1)
random (x)	I	I	Интервалдағыы жалған кездейсоқ сан [0, *x*)
round (x)	R	I	Жуық тұтас санға *x* толықтыру
sin (x)	I, R	R	Синус *x*, рад
sqr (x)	I, R	I, R	Квадрат *x*
sqrt (x)	I, R	R	Квадрат түбір *x*-тен
trunc (x)	R	I	Модуль бойынша *x* аспайтын жуық тұтас

Арифметикалық өрнек сандық шамаларға амалдарды орындау тәртібін береді. Арифметикалық өрнектерде арифметикалық амалдар, функциялар, операндтар, дөңгелек жақшалар болады. Бір константа немесе бір анымалы – бұл арифметикалық өрнектің қарапайым түрі.

Мысалы, Паскаль тілінің ережесі бойынша жазылған математикалық өрнек мынадай түрде беріледі:

$$\frac{2a + \sqrt{0{,}5\sin(x + y)}}{0{,}2c - \ln(x - y)},$$

(2 * A+ sqrt(0.5 * sin(X+ Y)))/(0.2 * C - ln(X - Y)).

Арифметикалық өрнектерді дұрыс жазу үшін белгілі бір ережелерді сақтау қажет.Барлық таңбаларды бір жолға, яғни бір деңгейде жазу керек.

1. «*» таңбасын қалдырмай, амалдардың барлық белгілерін қою керек.

2. Екі амал таңбасын бірден жзуға болмайды, яғни $A + -B$ жазуға болмайды, $A + (-B)$ жазу керек.

3. Басымдылығы тым жоғары амалдарды басымдылығы сәл төмен амалдардан бұрын орындайды. Амалдар басымдылықтарының кему тәртібі мынадай:

- функцияны есептеу;
- (-) белгісін ауыстырудың унарлы амалы;
- *, /, div, mod;
- +, -.

4. Басымдылықтары бірдей бірдей жазылған бернеше амалды жүйелі түрде солдан оңға қарай орындайды.

5. Жақшаға алынған өрнектің бөлігі бірінші кезекте есептеледі. (Мысалы, (A + B) * (C - D) өрнегінде көбейту қосу мен азайтудан кейін орындалады).

6. Математикакалық мағынасы жоқ өрнектерді жазуға жол берілмейді, мысалы: нөлге бөлу, теріс санның лограифмі және т.с.с.

Мысал келтірейік. Берілген ережелер бойынша жазылған арифметикалық өрнек (дөңгелек ішіндегі цифрлермен амалдарды орындау тәртібі берілген),

① ⑦ ④ ⑤ ③ ⑥ ② ⑫ ⑪ ⑩ ⑧ ⑨

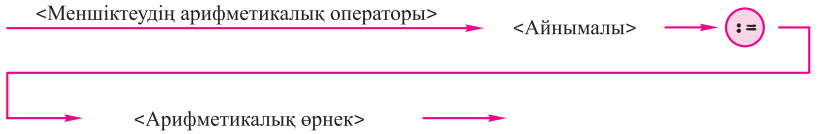
(1+y)*(2*x + sqrt(y) - (x + y)) / (y + 1/(sqr(x) - 4))

Келесі математикалық формулаға сәйкес келеді:

$$(1+y)\frac{2x+\sqrt{y}-(x+y)}{y+\frac{1}{x^2-4}}.$$

Паскальда санды кез келген дәрежеге айналдыру амалы немесе стандартты функция жоқ. x^y есептеу үшін келесідей жасау қажет:

- егеру — тұтас мән болса, көбейтуді орындау керек, мысалы: $x^3 \rightarrow x * x * x$. Одан да үлкен дәрежелер үшін циклдегі көбейтуді орындау керек;
- егеру — заттық мән болса, келесі математикалық формула қолданылады: : $x^y = e^{y\ln(x)}$, бұл жазба Паскальда түрінде беріледі:



2.14. –сурет. Меншіктеудің арифметикалық оператор құрылымы

exp (Y * ln(x))

Әлбетте, узаттық типте х нөлдік немесе теріс мәні болуы мүмкін емес. Тұтас типтегі үшін мұндай шектеу жоқ.

Мысалы Паскальда жазылған $\sqrt[3]{a+1} = (a+1)^{\frac{1}{3}}$, формуласы келесі түрде беріледі:

exp(1/3 * ln(A + 1))

Егер өрнекті есептеу нәтижесінде тұтас типті шама алынса, өрнек тұтас типті болады. Егер өрнекті есептеу нәтижесі ретінде заттық шама болса, өрнек заттық типке ие болады.

Меңгерудің арифметикалық операторы 2.14.-суретте берілген құрылымға ие болады.

Мысалы, ол мына түрдегі жазба болады

Y := (F * N - 4.5)/cos(X)

Меңгеру операторын орындау тәртібі бұрынырақ қарастырылған еді. Тек келесі ережеге назар аударған жөн: айнымалының типтері және өрнектер бірдей болулары тиіс. Бұл жерде шектен шығу тек өрнек тұтас типке, ал айнымалы – заттық типке ие болғанда ғана мүмкін болады.

Жаттығулар

1. Келесі формулалар үшін арифметикалық өрнектерді Паскальда жазу керек:

а) $a + bx = cyz$; б) $[(ax\text{-}b)x + c]x - d$ в) $\frac{a+b}{c} + \frac{c}{ab}$;

г) $\frac{x+y}{a_1}\frac{a_2}{x-y}$; д) $10^4\alpha - 3\frac{1}{5}\beta$; е) $\left(1+\frac{x}{2!}+\frac{y}{3!}\right) / \left(1+\frac{2}{3+xy}\right)$.

2. Келесі өрнектерге сәйкес келетін математикалық өрнектерді Паскальда жазу керек:

а) (p + q)/(r + s) - p * q/(r * s);

б) 1E3 + beta/(x - gamma * delta);

в) a/b * (c + d) - (a - b)/b/c + 1E-8.

3. Неліктен Паскальда функцияның аргументі әрқашан жақшада жазылады, мысалы ln5 емес, ln (5) деп жазады,?

4. Келесі формулалар үшін тиісті арифметикалық өрнектерді Паскальда жазу керек:

а) $(1 + x)^2$; б) $\sqrt{1 + x^2}$; в) $\cos^2 x^2$; г) $\log_2 \frac{x}{5}$;

д) $\arcsin x$; е) $\frac{e^x + e^{-x}}{2}$; ж) $x^{\sqrt{2}}$; з) $\sqrt[3]{1 + x}$;

и) $\sqrt{x^8 + 8^x}$; к) $\frac{xyz - 3{,}3\left|x + \sqrt[4]{y}\right|}{10^7 + \ln 4!}$; л) $\frac{\beta + \sin^2 \pi^4}{\cos 2 + \left|\operatorname{ctg} \gamma\right|}$.

5. Келесі өрнектердің мәндерін есептеу керек:

а) trunc(6.9);

б) trunc(6.2);

в) 20div6;

г) 2 div 5;

д) round(6.9);

е) round(6.2);

ж) 20mod 6;

з) 2 mod 5;

и) 3 * 7 div 2 mod 7/3 - trunc(sin(1)).

6. Келесі өрнектердің типін анықтау керек:

а) 1 + 0.0;

б) 20/4;

в) sqr(4);

г) sqrt(16);

д) sin(0);

е) trunc(-3.14).

7. Егер *y* – айнымалы, ал *n* – тұтас сан болса, келесі меңгеру операторларының қайсысы дұрыс екенін анықтау керек:

а) y := n+ 1;

б) n := y - 1;

в) n := 4.0;

г) y := trunc(y);

д) y := n div 2;

е) y := y div 2;

ж) n := n/2;

з) n := sqr(sqrt(n)).

8. Қосымша айнымалыларды қолданбай, *x* және *y* тұтас айнымалылардың орындарын ауыстыру керек. Үшінші айнымалы арқылы мәндерді айырбастау әдісімен салыстыру бойынша табылған алгоритмнің кемшілігін табу керек. Берілген алгоритмді заттық сандар үшін қолдануға бола ма?

9. *h* тұтас айнымалыға *k* оң тұтас сан жазбасында жүздіктер разрядында тұрған цифрдың мәнін беру керек (мысалы, *k* = 28 796, онда *h*= 7).

10. *S* тұтас айнымалыға к үш таңбалытұтас санның цифрлар қосындысының мәнін беру керек.

11. Келесі бағдарлама қандай есепті шығаратынын анықтау керек:

```
Program Test;
Type Natur = 1..MaxInt;
Var    N : Natur;
       X : Real;
Begin ReadLn(N);
            X := 0;
       While N > 0 Do
       Begin
           X := X + 1.0;
           N := N - 1
       End;
       WriteLn(X)
End.
```

Алынған нәтижені неғұрлым қарапайым тәсілмен алу мүмкіндігін көрсету керек.

2.7. ЕНГІЗУ МЕН ШЫҒАРУДЫ ҰЙЫМДАСТЫРУ

Пернетақтадан деректерді енгізу. *Деректерді* енгізу - бұл ақпаратты сыртқы құрылғылардан жедел жадыға жіберу.Әдетте, шығарылатын есептің бастапқы деректері енгізіледі.

Деректерді шығару — бұл деректерді жедел жадыдан сыртқы жүктемелерге жіберу (баспа, дисплей, магнитті құрылғылар және т.б.). Кез келген есепті шығару нәтижелері шығарылуы тиіс. Дербес компьютердің негізгі енгізу-шығару құрылғылары – пернетақта және дисплей (мониторлың экраны). Тек осы құрылғылар арқылы, ең алдымен, адам мен ДК арасындағы диалог жүзеге асырылады.

Пернетақтадан енгізу процедурасы келесі пішімде болады:

Read(<енгізу тізімі>)

Мұндағы <енгізу тізімі> — бұл үтірлермен бөлінген, айнымалы

аттарының жүйелігі, ал Read (оқу) — енгізудің стандартты процедурасына жүгінген оператор. Мысалы:

```
Read(a, b, c, d)
```

Бұл операторды орындау кезінде компьютердің жұмысы тоқтайды, содан соң пайдаланушы пернетақтада a, b, c, d айнымалылардың мәндерін, бір-бірінен аралық арқылы бөліп, береді. Бұл ретте енгізілетін мәндер экранда жанып тұрады. Теру соңында <Enter>: батырмасын шертеді. Мәндерді енгізу Паскаль тілінің синтаксисына қатаң түрде сәйкес келіп, орындалуы тиіс. Мысалы, бағдарламада енгізуді орындау кезінде

```
Var   T : Real;
      J : Integer;
      K : Char;
Begin
      Read(T, J, K);
```

пернетақтада

253.98 100 G [Enter] теру қажет.

Егер бағдарламада бірнеше Readоператорлары болса, онда олар үшін деректер ағынмен енгізіледі, яғни Read бір операторына арналған айнымалылардың мәндерін оқып алғаннан кейін келесі оператор үшін берілген деректер жол аяқталғанша, экранда оның алдындағы жолдан оқылады, содан соң келесі жолға ауысады. Мысалы, бағдарламада енгізуді орындау кезінде

```
Var    A, B : Integer;
       C, D : Real;
Begin
       Read(A, B);
       Read(C, D);
```

пернетақтада

18758 34[Enter] 2.62E-02 1.54E+01[Enter] теру қажет.

Пернетақтадан енгізу операторы сондай-ақ мына түрде болады

ReadLn(<енгізу тізімі>)

Мұндағы ReadLn (*read line*)— жолды оқып шығу.Read операторына қарағанда ReadLnбір операторына арналған мәндер соңғы тізімде оқылғаннан кейін, ReadLn келесі операторы үшін деректер жаңа

жолдан бастап оқылады. Егер осының алдындағы мысалдағы Read операторын ReadLn ауыстырса, яғни

ReadLn(A, B);
ReadLn(C, D) жазса, мәндерді енгізу екі жолдан тұрады:

18758 34 [Enter]
2.62E-02 1.54E+01 [Enter]

Экранға шығару. *Экранға шығару операторы* (шығарудың стандартты процедурасына жүгіну) келесі пішімде болады:

Write(<шығару тізімі>)

Мұнда шығару тізімінің элементтері ретінде түрлі типтегі өрнектер болуы мүмкін (әсіресе, константалар мен айнымалылар), мысалы:

Write(234);	{тұтас константа шығарылады}
Write(A+ B- 2);	{өрнекті есептеудің нәтижесі шығарылады}
Write(X, Summa, Arg1, Arg2);	{айнымалылардың нәтижелері шығарылады}

Экранға бірнеше санды бір жолға шығарғанда, олар аралықпен бөлінбейді, оны бағдарламашы қамтамасыз етуі тиіс. Мысалы, $I = 1$; $J = 2$; $K = 3$ болсын. Онда операторды орындау кезінде

Write(I, ' ', J, ' ', K);

Экранда келесі жолды қабылдаймыз: 1 2 3. Сонымен бірге соңғы таңбаны шығарғаннан кейін меңзер сол жолда қалады, және экранға келесі енгізу мңзердің осы позициясынан басталатын болады.
Экранға шығару процедурасы мынадай түрде болады

WriteLn(<шығару тізімі>)

Мұндағы WriteLn (*write line*)— жолды жазу. Бұл оператордың әрекеті Writetізімдегі соңғы мәнді шығарғаннан кейін мңзерді келесі жолдың басына ауыстыру болады. Параметрлерсіз жазылған WriteLn операторы жолды ауыстыруды орындайды.
Шығарудың пішімдері. Шығару тізімдерінде шығару пішім-

дерінің көрсеткіштері болуы мүмкін (пішімдер). Пішім экранға шығарылатын мәннің көрінісін анықтайды және оған сәйкес келетін элементтен қос нүкте арқылы бөлінеді. Егер пішімнің көрсеткіші болмаса, машина мәнді әдепкі қалпы бойынша қарастырылған, белгілі бір ереже бойынша шығарады. Бұдан әрі қысқаша анықтама ретінде түрлі типтегі шамаларды пішімді және пішімсіз шығарудың ережелері мен мысалдарын келтіреміз. Шығару тізімін ұсыну үшін келесі белгілеулерді қолданамыз:

I, P, Q — тұтас санды өрнектер;
R —заттық типтегі өрнек;
B — буль типтес өрнек;
Ch — таңбалық шама;
S — жолдық өрнек;
цифр;
* «+» немесе «-» белгісі;
_ аралық.

Write процедурасының пішімі

I — Меңзердің орналасу позициясынан бастап, I шамасының ондық көрінісі шығарылады:

I мәні	Оператор	Нәтиже
134	Write(I)	134
287	Write(I,I,I)	287287287

I:P — **I** шамасының ондық көрінісі ені **P** болатын өрістің шеткі оң жақ позициясына шығарылады.

I мәні	Оператор	Нәтиже
134	Write(I:6)	_134
312	Write((I + I):7)	_624

R — ені 18 таңбадан тұратын өріске қалқымалы нүктемен берілген пішімдеRшамасының ондық көрінісі шығарылады(егер R> 0.0 болса, #.##########E*## пішімі қолданылады; егер R< 0.0 болса, пішім келесі түрде болады_-#.##########E*##):

R мәні	Оператор	Нәтиже
715.432	Write(R)	_7.1543200000E+02
-1.919E+01	Write(R)	_-1.9190000000E+01

R:P — ені P болатын өрістің шеткі оң позицияларынаR мәнінің ондық көрінісі қалқымалы нүктемен берілген қалыпты пішімде шығарылады. Шығару өрісінің минималды ұзындығы оң сандар үшін - жеті, теріс сандар үшін –сегіз таңбаны құрайды. Нүктеден соң ең болмағанда бір нүкте шығарылады:

R мәні	Оператор	Нәтиже
511.04	Write(R:15)	5.110400000E+02
46.78	Write(-R:12)	-4.67800E+01

R:P:Q— таңбаларының ені P болатын өрістің шеткі оң жақ позициясынаR мәнінің ондық көрінісі белгіленген нүкте пішімінде шығарылады, сонымен бірге ондық нүктеден соң санның бөлшек бөлігі арқылы берілетін (егер Q= 0 болса, онда бөлшек бөлік те, ондық нүкте де шығарылмайды; егер Q> 24 болса, онда шығару кезінде қалқымалы нүктесі бар пішім қолданылады) Q цифрлардың (0<Q< 24) шығарылады:

R мәні	Оператор	Нәтиже
511.04	Write(R:8:4)	511.0400
-46.78	Write(R:7:2)	_-46.78

Ch:P— ені P болатын өрістің шеткі оң жақ позициясына Ch мәні шығарылады:

Ch мәні	Оператор	Нәтиже
'X'	Write(Ch:3)	X
'!'	Write(Ch:2,Ch:4)	! !

S — меңзер позициясынан бастапS мәні шығарылады:

S мәні	Оператор	Нәтиже
'Day N'	Write(S)	Day N
'RRDD'	Write(S,S)	RRDDRRDD

S:P —S мәні P таңбаларының ені P болатын өрістің шеткі оң жақ позициясына шығарылады:

S мәні	Оператор	Нәтиже
'Day N'	Write(S:10)	Day N
'RRDD'	Write(S:5,S:5)	RRDD RRDD

B — меңзердің ағымдағы позициясынан бастап, B өрнектің нәтижесі шығарылады B (True немесе False):

B мәні	Оператор	Нәтиже
True	Write(B)	True
False	Write(B, Not B)	FalseTrue

B:P — таңбаларының ені P болатын өрістің шеткі оң жақ позициясына бульдік өрнектің нәтижесі шығарылады:

B мәні	Оператор	Нәтиже
True	Write(B:6)	_ _True
False	Write(B:6, Not B:7)	_False_ _ _True

Экранға таңбалық шығаруды басқару. Экранға шығару үшін тек Write және WriteLn процедураларын қолдану бағдарламашыға шығарылатын мәтінді экранға орналастыруды басқаруға сәл ғана мүмкіндік береді. Мәтінді басып шығару тек жоғарыда төменге, солдан оңға қарай жүргізіледі. Алдында өтіп кеткен жолдарға қайтып келу, басылған мәтінді өшіру, таңбалардың түсін өзертуге және т.с.с. мүмкін емес.

Экранға шығаруды басқарудың қосымша мүмкіндіктерін процедулара мен CRT модулінің функциялары береді. Пайдаланушылық бағдарламаның модульмен байланыс орнату үшін сипаттау бөлімдерінің алдында жол қойылуы керек.

```
Uses CRT
```

Позицияның координаттары. Мәтіндік экрандағы әрбір таңбалық позиция (*X, Y*) анықталады. *X*-координата — жолдағы позицияны біліреді. Жолдағы шеткі сол жақ позицияда *X* = 1, ал шеткі оң жақта (стандартты тәртіптемеде) — *X* = 80 бар. *Y*-координата —таңба болатын жолдың нөмірі. Жолдар жоғардан төменге қарай нөмірленеді.

Экрандағы меңзерді CRT модулінде (*X, Y*) координаталарымен берілген позицияға бекіту үшін мынадай процедура бар:

```
GOTOXY(X, Y)
```

Мұнда меңзердің координаталары Byte типті өрнекпен беріледі.

Түсті басқару. Экранның мәтіндік режимінде түрлі-түсті дисс-

плелерде 16 түсті қолдануға болады. Түстер 0-ден 15 дейінгі реттім нөмірлермен, немесе CRT модулінде анықталған атаулы константалармен сәйкестендіріледі. Түстердің нөмірлері мен константалардың аттарының сәйкес келуі келесідей беріледі:

0	Black	6	Brown	12	LightRed
1	Blue	7	LightGray	13	LightMagenta
2	Green	8	DarkGray	14	Yellow
3	Cyan	9	LightBlue	15	White
4	Red	10	LightGreen		
5	Magenta	11	LightCyan		

Бағдарламада таңбалар бейнеленетін фонның түсін де, сол таңбалардың түстерін де басқаруға болады. Фон түстерін тағайындау процедурасы:

TextBackGround(Color)

Мұндағы аргумент — түстің нөмірін беретін Byte типті шама.
Таңба түсін тағайындау процедурасы:

TextColor(Color)

Егер фонның түсі мәтіндік терезені тазартудан бұрын тағайындалса, онда тазартудан кейін терезе түспен «боялады». Егер фон экранды тазалағаннан кейін орнатылса, онда тазартылған терезе қара түсте болады (әдепкі қалпы бойынша), ал фонның тағайындалған түсі таңбалар шығарылатын позицияда орнатылатын болады. Экранды тазарту және фонды белгіленген түспен бояу процедура арқылы жүзеге асырылады (параметрлерсіз)

ClrScr

Келесі бағдарлама бойынша экранға түрлі реңктегі алғашқы 15 түстің нөмірлері (өз түсіне сәйкес келетіндей) экран ортасындағы ақ фонға шығарылады:

Uses CRT;

```
Var I : Byte;
Begin
        TextBackGround(White);
        ClrScr;
        GoToXY(1, 12);
        For I := 0 To 14 Do
        Begin
            TextColor(I);
            Write(I : 5)
        End
End.
```

Процедуралар мен CRTмодулінің функцияларын қолданудың тағы бір мысалын қарастырайық. Таңбалық экранның 13-ші жолының ортасында циклді қайталау есептегіші «айналып жүретіндей» бағдарлама құру керек: 1, 2, 3, 4 және т.с.с сандар сендер компьютердің бір батырмасын басқанша, бір-бірін ауыстыра береді. Сонымен бірге 0-ден 15 дейінші цифрлердің түстері циклді түрде өзгеріп отырады:

```
Uses CRT;
var i: integer;
begin
      Clrscr;
      TextBackGround(White);
      i:= 1;
      Repeat
          GoToXY(40,13);
          TextColor(i mod 16);
          Write (i);
           i:= i+1
     until KeyPressed
End.
```

Бұл бағдарламада CRT модулінен *KeyPressed* функциясы қолданылады. Бұл функцияны атқару кезінде пернетақтадан сұрату жүргізіледі және қандай да бір батырманың басылу-басылмауы анықталады. Нәтижесінде егер қандай да бір батырма басылса функция *True* логикалық мәнін,кері жағдайда *False*— мәнін жібереді. Көбіне бұл функцияны экранда нәтижелер терезесінің кідіруін ұйымдастыру үшін қолданады (бағдарламаны орындаған соң ТурбоПаскаль экранға редакор терезесін шақырады). Бағдарлама соңының алдында

келесі оператор жазылады:

Repeat Until KeyPressed;

Бұл қандай да бір батырманы баспағанша, «бір орында айналып тұратын» бос цикл. Бұл кезде экранда – нәтижелер терезесі шығады. Батырманы басқан соң *KeyPressed* мәні *True* тең болады,цикл аяқталып, атқару бағдарламаның соңына шығады да экранға редактор терезесі қайта келеді. Бұл әдісті бағдарламаны орындауды оның кез келген жерінде кідірту үшін қолдануға болады. CRTмодулінің басқа да функциялары мен процедуралары туралы Паскальда бағдарламалау жүйелерінің анықтамасында оқуға болады.

Жаттығулар

1. Бастапқы деректер ретінде 1.0 және -2.0 сандары берілсе, келесі бағдарлама нені басып шығаратынын анықтау керек:

```
Program Roots;
Var B, C, D : Real;
Begin
        Read(B, C);
        D := Sqrt(Sqr(B) - 4 * C);
        WriteLn('x1 =', (-B + D)/2,
            'x2 =', (-B - D)/2)
End.
```

2. 3.4 және 7.9 мәндерін жүйелі түрде енгізу кезінде, келесі бағдарлама нені басып шығаратынын анықтау керек:

```
Program Less;
Var X : Real; T : Boolean;
Begin
        Read(X);
        T := X <Round(X);
        Read(X);
        T := T And (X <Trunc(X));
        WriteLn(T)
End.
```

3. 36,-6 және 2345 мәндерін жүйелі түрде енгізу кезінде, келесі бағдарлама нені басып шығаратынын анықтау керек:

```
Program ABC;
Var A, B : Integer;
Begin
        Read(A, B, A);
        WriteLn(A, B : 2, A : 5)
```

End.

4. Пайдаланушымен келесі түрде диалог жүргізетін екі тұтас санның қосындысын есептеу бағдарламасын құру керек (көп нүктенің орнына - енгізілетін және шығарылатын сандар):

Екі қосылғышты енгізіңдер
a= .....
b= .....
Есептеу нәтижесі:
a+ b= .....

2.8. ЛОГИКАЛЫҚ ШАМАЛАР, АМАЛДАР, ӨРНЕКТЕР

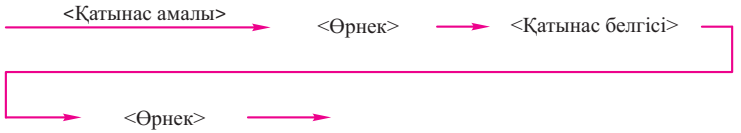
1.4-тармақшада логикалық шамалар, логикалық амалдар, логикалық өрнектер туралы айтылып өтті. Логикалық типтің шамасы тек екі мәнді: АҚИҚАТ және ЖАЛҒАН қабылдай алатынын естеріңізге саламыз.

Паскальда логикалық мәндер False (F) жәнеTrue (T) көмекші сөздерімен, ал логикалық типтегі түрлендіргіш – Boolean көмекші сөзбен белгіленеді.

Boolean түріндегі шамалардан басқа (константалар және айнымалылар), False, True логикалық мәндері қатынас амалдарының нәтижелерін қабылдайды.

Қатынас амалдары екі операндты салыстыруды жүзеге асырады және олардың арасындағы тиісті қатынастың ақиқат немесе жалған екендігін анықтайды.

Қатынас амалдарының құрылымы 2.15.суретте берілген, онда



2.15. –сурет. Қатынас амалдарының құрылымы

<қатынас белгісі> ::= = (тең) | <> (тең емес) | >
(артық) | < (кем) | >= (артық немесе тең) | <= (кем немесе тең).

Қатынастарды жазуға мысал келтірейік:

x <y; a + b >= c/d; abs(m - n) <= 1.

Және қатынастар мәндерін есептеу мысалы:

Қатынас	Нәтиже
12>= 12	True
56 > 10	True
11<= 6	False

Логикалық амалдар бульдік типтегі операндтарға орындалады. Төрт логикалық амал бар: **Not** — терістеу; **And** — логикалық көбейту (конъюнкция); **Or** — логикалық қосу (дизъюнкция). Бұл үш міндетті амалдан басқа ТурбоПаскальда **Xor** көмекші сөзбен белгіленетін «НЕМЕСЕ –ні жоққа шығаратын» тағы бір амал бар. Бұл екі операндтың екеуі де түрлі логикалық мәнге ие болған жағдайда нәтижесінде АҚИҚАТ мәнін беретін екіорындық амал.

Логикалық амалдар басымшылықтарының кему тәртібі бойынша аталды. Операндтардың түрлі мәндері үшін логикалық амалдарды орындау тәртібі 2.6.кестеде көрсетілген.

Қатынас амалдары ең төмен басымшылыққа ие, сондықтан логикалық амалдың операнды қатынас болған жағдайда, оларды шеңбер жақшаға алу керек. Мысалы, $1 \le x \le 50$ математикалық теңсіздікке келесі логикалық өрнек сәйкес келеді:

(1 <= x) And (x <= 5 0)

2.6. кесте. Логикалық амалдарды орындау нәтижесі

A	B	**Not** A	A **And** B	A **Or** B	A **Xor** B
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

Логикалық өрнек — бұл бағдарламалау тілінде жазылған логикалық формула. Логикалық өрнек логикалық амалдармен және шеңбер жақшалармен байланысқан логикалық операндтардан тұрады. Логикалық өрнекті есептеу нәтижесі ретінде бульдік шама болады (False немесеTrue). Логикалық операндтар ретінде логикалық константалар, айнымалылар, функциялар, қатынас амалдары болады. Бір жеке тұрған логикалық операнд логикалық өрнектің қарапайым түрі болып табылады.

d, b, c— логикалық сандар; *x, y*— заттық сандар; *k*— тұтас айнымалы болатын логикалық өрнекке мысал келтірейік:

1) x <2 * y; 2) True; 3) d;
4) Odd(k); 5) **Not Not** d; 6) **Not** (x >y/2);
7) d **And** (x <>y) **And** b;
8) (c **Or** d) **And** (x = y) **Or Not** b.

d = True, *b* = False, *c* = True, *x* = 3.0, *y* = 0.5, *k* = 5 болғанда, есептеу нәтижелері мынадай болады:

1) False; 2) True; 3) True; 4) True;
5) True; 6) False; 7) False; 8) True.

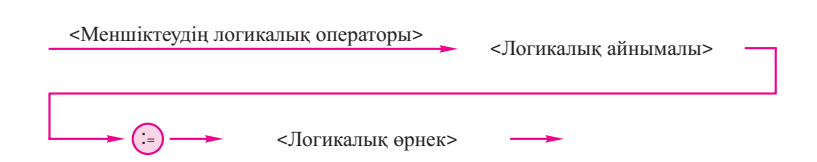
Берілген мысалда Odd (*x*) логикалық функциясы қолданылған. Бұл функция *x* тұтас аргументтің, егер х тақ болса True мәнін, және жұп болса – False мәнін қабылдайтын функция.

Меншіктеудің логикалық операторы 2.16.- суретте берілген құрылымға ие болады.

Меншіктеудің логикғалық операторына мысал келтірейік:

1) d := True;
2) b := (x > y) **And** (k <> 0);
3) c := d Or b **And Not**(Odd(k) **And** d).

Функция *Odd*(*X*) — *X* тұтас аргументтен логикалық типтегі стандартты функция. Функцияның нәтижесі аргументтің мәні тақ санға тең болса, True тең және аргумент тақ санға тең болса, нәтиже False тең болады.



2.16.-сурет. Меншіктеулогикалық операторының құрылымы

Функция Ord(*X*)— тұтас типтің стандартты функциясы, X аргументі — кез келген реттік типтің шамасы. Функцияны орындау нәтижесі – Өзінің типіндегі X мәнінің реттік нөмірі. Мысалы, егер *X* –логикалық шама болса, онда Ord(False) = 0; Ord(True) = 1.

Жаттығулар

1. Келесі логикалық өрнектердің мәнін табыңдар:
а) *K*= 15 болғанда Kmod 7 = *K* div 5 —1;
б) *P*= 0.182 болғанда Odd (trunc (10**P*));
в) *n*= 0 болғанда not Odd (*n*);
г) *t*= True болғанда, *P*= 10101 tand (*P* mod 3 = 0) и;
д) *x* = 2, *y*= 1 болғанда (*x** *y*<>0) and (*y* >*x*);
е) *a*= False, *b*= True болғандаа or not b.

2. Келесі меншіктеу операторларын орындағн соң a= True и x= 1 болғанда, *d* логикалық айнымалы қандай мәнге ие болатынын анықтау керек:
а) d := x < 2; б) d := not a or Odd(x); в) d := Ord(a)<> x?

3. Орындау нәтижесінде *t* логикалық айнымалы өрнек ақиқат болған жағдайда True мәніне, және жалған болған жағдайда - False мәніне ие болатын меншіктеу операторын жазу керек:
а) *x, y*, zсандарынантек екеуі ғана бір-бірімен тең;
б) *x*— оң сан;
в) *x, y, z* сандарының әрқайсысы оң;
г) *x, y, z* сандарының тек біреуі ғана оң;
д) *pq*-ге тұтас бөлінеді;
е) 5 цифры *k* үш таңбалы тұтас санның ондық жазбасына кіреді.

2.9. ТАРМАҚТАЛАТЫН АЛГОРИТМДЕРДІ БАҒДАРЛАМАЛАУ

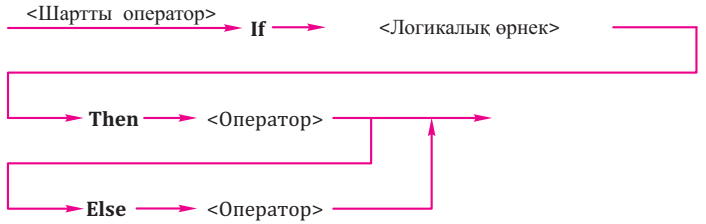
Тармақталудың алгоритмдік құрылымы шартты оператордың көмегімен Паскальда келесі түрде беріледі:

If<Шарт>**Then**<1 Оператор>**Else**<2 Оператор>;

Бұдан басқа, шартты оператордың толық емес түрін қолдану мүмкін болады:

If<Шарт>**Then**<Оператор>;

Шартты операторды қатаң түрде сипаттау синтаксистік диаграмма түрінде 2.17.-суретте берілген.



2.17.- сурет. Шартты операторлың синтаксистік диаграммасы

Шартты операторда шарт ретінде бірінші кезекте шығарылатын логикалық өрнек болады. Егер оның мәні True тең болса, онда <1 Оператор 1> (Then кейін) орындалады, ал егер оның мәні False тең болса, толық түрі үшін <2 Оператор > (Else кейін) немесе бірден толық емес нысаны үшін шарттыдан кейін келетін оператор (Else - сіз) орындалады.

2.1. мысал. a, b, c үш қабырғасының ұзындықтары бойынша үшбұрыштың ауданын есептеу бағдарламасын құру керек.

Есепті шығару үшін Герон формуласы қолданылады

$$\sqrt{p(p-a)(p-b)(p-c)},$$

мұндағыр = (a + b + c)/ 2 — үшбұрыштың жартылай периметрі.

Бастапқы деректер үшбұрыштың қабырғалары үшін негізгі ара қатынасты қанағаттандыруы тиіс: әрбір қабырғаның ұзындығы басқа екі қабырғаның ұзындығының қосындысынан кем боуы тиіс.

Бір шартты операторда айтарлықтай күрделі логикалық өрнектерді жазу мүмкіндігіне ие бола тұрып, бастапқы деректердің барлық дұрыс емес нұсқаларын бірден «сүзуге» болады:

```
Program Geron;
Var A, B, C, P, S : Real;
Begin
        WriteLn('үшбұрыш қабырғаларының ұзындығын
         енгізіңдер:');
        Write('a = '); ReadLn(A);
        Write('b = '); ReadLn(B);
        Write('c = '); ReadLn(C);
        If (A > 0) And (B > 0) And (C > 0) And (A + B > C)
                And (B + C > A) And (A + C > B)
        Then Begin
```

```
                P := (A + B + C)/2;
                S := sqrt(P * (P - A) * (P - B) * (P - C));
            End
Else WriteLn('Дұрыс емес бастапқы деректер')
End.
```

2.2. мысал. Квадрат теңдеуді шешу бағдарламасын құру керек.

Бұл есепті шешу алгоритмі 1.3-тармақшада толық қарастырылған (1.4.-суретті қараңыз). Алгоритмнің енгізілген тармақталу құрылымы бар. Алгоритм тілінде оның сипаттамасы сондай-ақ 1.3-тармақшада берілген. Паскальда бағдарлама келесі түрде беріледі:

```
Program Roots;
Var A, B, C, D, X1, X2 : Real;
Begin
      WriteLn('Квадрат теңдеудің коэффициенттерін енгізіңдер');
      Write(' a= );   ReadLn(A);
      Write('b = ); ReadLn(B);
      Write('c = ); ReadLn(C);
    If (A = = 0));
        Then If (B = 0)
              Then If (C = 0)
                    ThenWriteLn('кез келгенX — шешімі')
                    ElseWriteLn(Шешімі жоқ')
        Else
        Begin
              X := -c/b;
              WriteLn('X = ', X)
        End
    Else
    Begin d := b * b - 4 * a * c;
        If(d< 0)
        ThenWriteLn('Заттық түбірлер жоқ')
        Else
        Begin
              XI := (-b + sqrt(d))/2/a;
              X2 := (-b - sqrt(d))/2/a;
              WriteLn('X1 = ', X1);
              WriteLn('X2 = ', X2)
          End
```

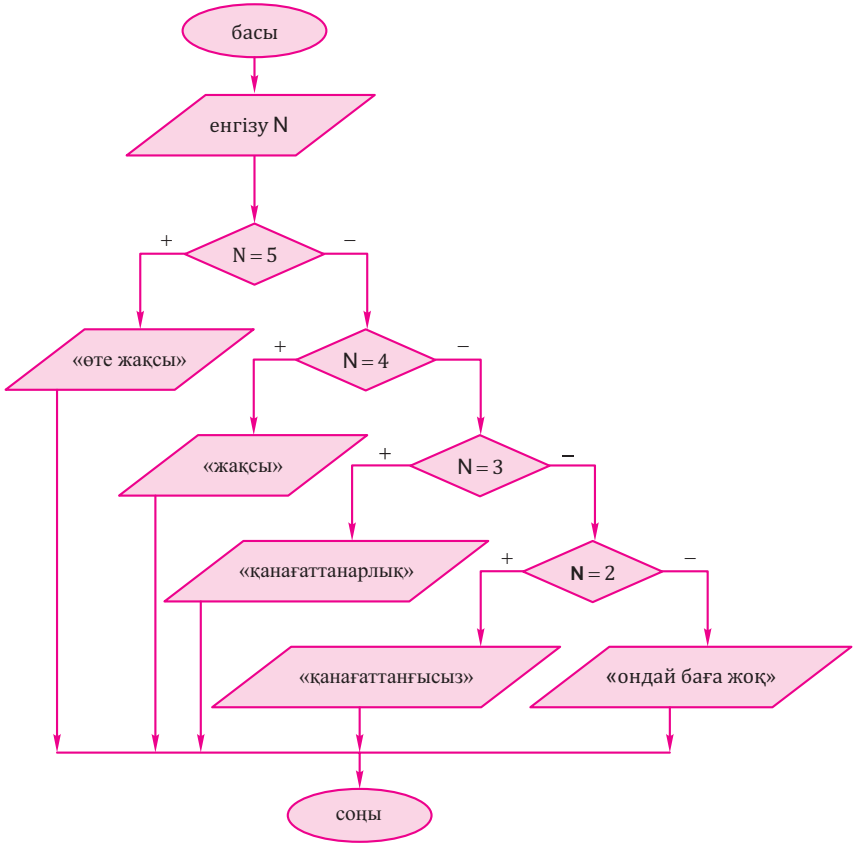
End

End

2.3. мысал. Бес баллдық бағаны оның атауына ауыстыру бағдарламасын құру керек: 5 — өте жақсы, 4 — жақсы, 3 — қанағаттанарлық, 2 — қанағаттанғысыз.

Есепті шығару алгоритмінің блок-схемасы 2.18. –суретте берілген. Бұл алгоритмнің салынған тармақталу құрылымға ие, және тиісті бағдарламаны Паскальда келесідей жазуға болады:

Program Marks-2;
Var N : Integer;
Begin
WriteLn(бағаны енгізіңіз:'); ReadLn(N);



2.18.-сурет.Бес баллдық бағаны оның атауына ауыстыру алгоримінің блок-схемасы

```
If(N = 5)
ThenWriteLn('Өте жақсы,)
ElseIf(N = 4)
        ThenWriteLn('Жақсы')
        ElseIf(N = 3)
                ThenWriteLn('Қанағаттанарлық')
                Else If (N = 2)
                        Then WriteLn('Қанағаттанғысыз');
                        Else WriteLn(' бұрыс баға')
End.
```

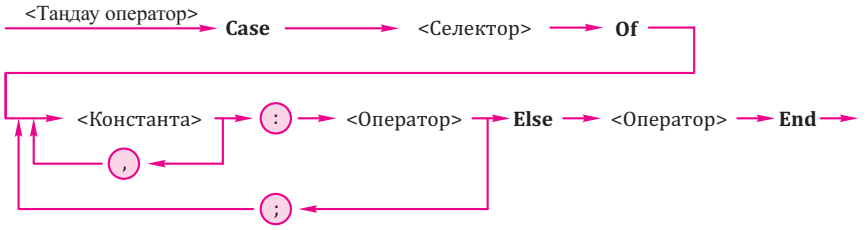
Салынған тармақталудың құрылымын Паскаль тілінде бар таңдау операторының бірінің көмегімен бағдарламалауға болады:

```
Program Marks;
Var N: Integer;
Begin
        WriteLn('Бағаны енгізіңіз:');
        ReadLn(N);
        Case N Of
                5: WriteLn('Өте жақсы');
                4: WriteLn('Жақсы');
                3: WriteLn(Қанағаттанарлық);
                2: WriteLn('Қанағаттанғысыз')
          ElseWriteLn('Ондай баға жоқ')
End.
```

Таңдау операторының пішімі 2.19.суретте берілген синтаксистік диаграммада сипатталады, мұндағы <Селектор> — бұл кез келген реттік типтің өрнегі, <Константа> — селектор тәріздес типтің тұрақты шамасы, ал <Оператор> — кез келген қарапайым немесе құрама оператор.

Таңдау операторын орындау келесідей жүзеге асырылады: өрнек-селектор есептеледі, содан соң константалар тізімінде селектордың алынған мәнімен сәйкес келетін мән анықталады, ал содан кейін берілген константамен белгіленген оператор атқарылады. Егер ондай константа табылмаса, **Else** кейін болатын операторды орындауға көшу жүзеге асырылады.

Таңдау операторында константалар тізімін қолдануға мысал келтірейік. Келесібағдарламастуденттің емтиханды тапсырған-тапсырмағаны, яғни ол 3.4. немесе 5 деген баға алса, емтихан тапсырылды, ал егер 2 деген баға алса, емтихан тапсырылмағандығын хабарлайды:




```
Case N Of
        3, 4, 5 : WriteLn('Емтихан тапсырылды')
              2 : WriteLn('Емтихан тапсырылған жоқ')
        Else WriteLn(Ондай баға жоқ');
```

Шартты оператор сияқты таңдау операторы да толық емес түрде, яғни Elseтармағысыз қолданылуы мүмкін.

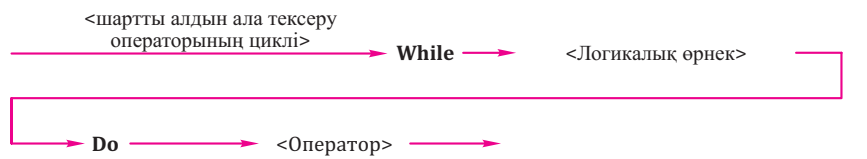
2.9. ТАРМАҚТАЛАТЫН АЛГОРИТМДЕРДІ БАҒДАРЛАМАЛАУ

Циклдік құрылымның үш түрі бар: шартты алдын ала тексеретін цикл, шартты соңынан тексеретін цикл және өлшем бойынша цикл.

Шартты алдын ала тексеретін цикл. «Әзірше», немесе шартты алдын ала тексеретін циклі операторының синтаксистік диаграммасын алайық (2.20-сурет). Мұнда алдымен <Логикалық өрнек> есептеледі. Оның мәніTrue тең болып тұрғанда, циклдің денесі -<Оператор> орындалады. Сонымен бірге <Оператор>қарапайым да, құрама да бола алады.

Мысал ретінде үйлесімді қатардың қосындысын берілген дәлдікпен есептейтін Паскальдағы бағдарлама үзіндісін

$$1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{I}+\cdots$$



2.20.-сурет. Шартты алдын ала тексеру циклінің синтаксистік диаграммасы

Кезекті қосынды ε кем болғанда немесе I тұтас айнымалы MaxInt мәніне жеткенде, қосу тоқтатылады:

```
S := 0;
I := 1;
While (1/I >= Eps) And (I < MaxInt) Do
Begin
    S := S + 1/I;
    I := I + 1
End.
```

Шартты соңынан тексеру циклі. «Дейін» циклінің операторының немесешартты соңынан тексеру циклінің синтаксистікдиаграммасы 2.21.-суретте берілген.

Берілген циклі атқару <Логикалық өрнек>True тең болған сәтке дейін қайталанады.

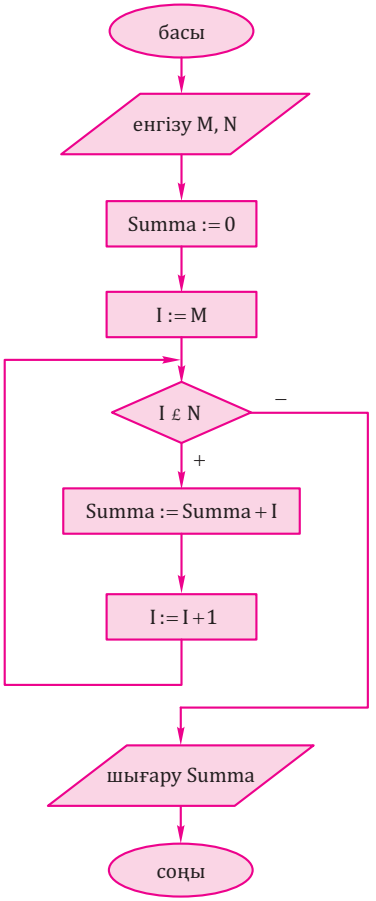
Шартты соңынан тексеру циклін қолданған осының алдындағы есеп былай шығарылады:

```
S := 0;
I := 1;
Repeat
    S := S + 1/I;
    I := I + 1
Until (1/I <Eps) Or (I >= MaxInt);
```

Өлшем бойынша цикл. M –нан N дейінгі тұтас сандардың қосындысын тікелей қосу арқылы есептеу есебін қарастырайық. Бұл есепті келесідей жазуға болады:

$$\text{Summa} = \begin{cases} \sum_{i=M}^{N} i, & \text{егер } M \le N; \\ 0, & \text{егер } M > N \end{cases}$$

Бұл есепті шығару алгоритмінің блок-схемасы 2.22.-суретте берілген, ал «Әзірше» циклінің құрылымын қолданатын тиісті бағдарламаны келесідей жазуға болады:



2.22. –сурет. Тұтас сандарды қосу алгоритмінің блок-схемасы

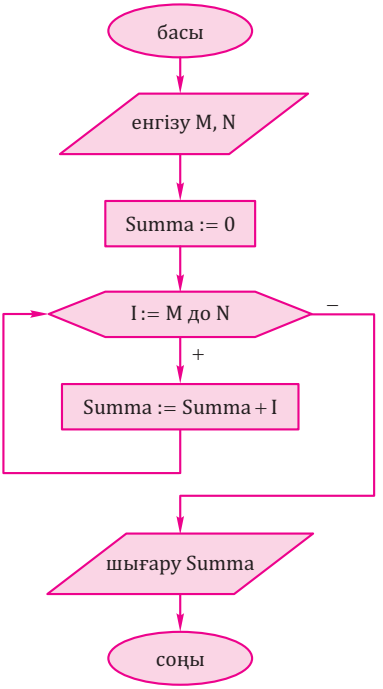
Бұл есепті шығару алгоритмінің блок-схемасы 2.22.-суретте берілген, ал «Әзірше» циклінің құрылымын қолданатын тиісті бағдарламаны келесідей жазуға болады:

```
Program Adding;
Var I, M, N, Summa : Integer;
Begin
    Write('M =');
    ReadLn(M);
    Write('N =');
    ReadLn(N);
    Summa := 0;
```

```
    I : = M;
    While I <= N Do
      Begin
      Summa := Summa + I;
      I := I + 1
  End;
 WriteLn('сомасы тең', Summa)
End.
```

Енді «өлшем бойынша цикл» деп аталатын циклдік құрылымның жаңа типін енгіземіз. Осы құрылымды қолданып, қосу алгоритмінің блок-схемасы 2.23-суретте берілген, ал осы есепті шығару алгоритмін Паскальда былайша жазуға болады:

```
Program Summering 2;
Var I, M, N, Summa: Integer;
Begin Write('M =');
```



2.23.-сурет. Өлшем бойынша циклді қосу алгоритмінің блок-схемасы

```
Write('N=');
ReadLn(N);
Summa := 0;
For I : = M To N Do
    Summa := Summa + I;
WriteLn('Сумма тең', Summa)
End.
```

Мұндағы *I* тұтас айнымалы *M*–нан *N* дейінгі диапазонда барлық мәндердің жүйелігін қабылдайды. *I* әрбір мәні кезінде циклдің денесі атқарылады. Циклдің соңғы атқаруынан кейін $I = N$ болғанда, циклдің алгоритмді жалғастыруға шығуы болады. Егер $M < N$, болса, және $M > N$ болғанда цикл бір де бір атқарылмаса, онда ол бір рет болса да орындалады.

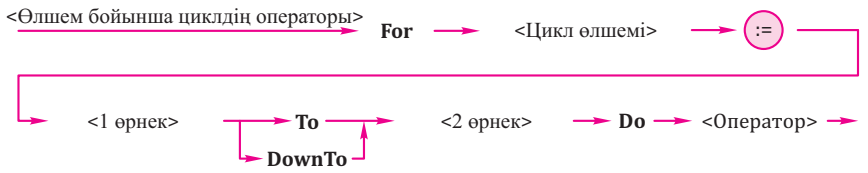
Берілген бағдарламада синтаксистік диаграммасы 2.24.-суретте берілген өлшем бойынша циклдің операторы қолданылады, мұндағы

<цикл өлшемі> ::= <реттік типтегі қарапайым айнымалының аты>

For операторының атқарылуы (**To** бірінші нұсқасында) келесі сызба бойынша жүргізіледі:

1. <1 өрнек> және <2 өрнек> мәндері есептеледі, және тек бір рет циклге ену кезінде ғана.
2. Циклдің өлшеміне <1 өрнек> мәні беріледі.
3. Цикл өлшемінің мәні <2 өрнек> мәнімен салыстырылады. Егер циклдің өлшемі осы мәннен кем немесе оған тең болса, циклдің денесі атқарылады, кері жағдайда циклдің орындалуы тоқтатылады.
4. Цикл өлшемінің мәні оның типіндегі келесі мәнге өзгереді (тұтас сандар үшін - бірге артады) және берілген сызбаның 3-тармағына көшу жүзеге асырылады.

For циклінің операторыөзіне **While** циклін қолдану кезінде түрлі операторларды орындайтын әрекеттерді біріктіреді:



2.24.-рет. Өлшеммен берілген циклдің синтаксистік диаграммасы

бастапқы мән өлшеміне меңгеруді, соңғы мәнмен салыстыру, мәнді келесіге өзгерту.

Әдетте, тұтас сандарды қосу нәтижесі қосу тәртібіне байланысты емес. Мысалы, қарастырылатын есепте сандарды кері тәртіпте де қосуға болады, яғни *N* -ден *M* дейін (*N*≥*M*). Ол үшін **For** циклі операторының екінші нұсқасын қолдануға болады:

```
Summa := 0;
For I : = N DownTo M Do
Summa := Summa + I;
```

DownTo тікелей «төменге дейін» деп аударылады. Берілген жағдайда циклдің өлшемі кему бойынша өзгереді, яғни циклдің әрбір қайталануы кезінде өлшем өз мәнін алдындағыға өзгертеді

(i := pred(i) бірдей). Сәйкесінше, *N* <*M* болған жағдайда, цикл бір де бір рет орындалмайды.

For операторымен жұмыс жасағанда келесі ережелерді ескерген жөн:

- циклдің өлшемінде Real типі бола алмайды;
- циклдің денесінде циклдің өлшемі – айнымалыны өзгертуге болмайды;
- циклден шыққандациклдің өлшемі – айнымалының мәні анықталмаған болып саналады.

Келесі мысалда **For** циклінің өлшемі ретінде таңбалық айнымалыны қолданамыз.Экранда латын алфавиті әріптерінің ондық кодтарын алу керек болсын. Латын әріптері кодтау кестесінде алфавит бойынша реттелгені белгілі. Тиісті бағдарламаның үзіндісін келесідей жазуға болады:

```
For C : = 'a' To 'z' Do
Write (C, ' - ', Ord(C));
```

Мұндағы C айнымалы Char типтес.

Енді өздерің латын алфавитінің кодталуын кері тәртіпте бағдарламаландар ('z' -'a' дейін).

Жаттығулар

1. Паскальда биквадратты теңдеудің толық шешуін құрыңдар.

2. Таңдау операторын қолдана отырып, енгізілген жылдағы айдың нөмірі бойынша тиісті жыл мезгілін шығарады (қыс, көктем, жаз, күз).

3. While, Repeat жәнеFor цикл операторларын қолдана отырып, N! есептеу бағдарламасының үш нұсқасын құру керек.

4. Таңбалардың жүйелігі «z» бас немесе кіші латын әрпі кездескенше енгізіле беретін бағдарлама құру керек. Енгізілетін таңбалар ішінде «W»

5. (ln x; ex), где x>1 интервалына кіретін барлық тұтас сандардың квадрат қосындысын есептеу керек.

6. Радиусы R (R > 0) болатын және координаталар басында ортасы бар шеңберге кіретін, тұтас санды координаталары бар нүктелер санын есептеу керек.

7. [0; 1] интервалында 0.1 қадаммен sinx және cosx функциялары мәндерінің кестесін келесі түрде басып шығу керек:

x	sin x	cos x
0.0000	0.0000	1.0000
0.1000	0.0998	0.9950
...	...	...
1.0000	0.8415	0.5403

8. Ондық жазбасында бірдей цифрлер жоқ барлық үш таңбалы сандарды арту тәртібінде басып шығу керек.

9. Тұтас $n > 2$ берілген. [2; n] интервалынан барлық жай сандарды басып шығу керек.

2.10. ІШКІ БАҒДАРЛАМАЛАР

Көмекші алгоритм ұғымы 1.5.бөлімдеқарастырылған болатын. Бағдарламалау тілдерінде көмекші алгоритмдер *ішкі бағдарламалар* деп аталады.Паскальда ішкі бағдарламаның екі түрі бар: *процедуралар* және *функциялар*.

Келесі мысалды қарастырайық: a және b екі натурал сан берілген. $a + b$, $|a - b|$, $a \cdot b$ үш шама үшін ең көп ортақ бөлгішті анықтау қажет. Мұны былай жазайық:ЕОБ ($a + b$, $|a - b|$, $a \cdot b$).

Берілген есепті шешу идеясы келесі математикалық факті бойынша түсіндіріледі: егерх, y, z — үш натурал сан болса, онда ЕОБ(x y, z) = = ЕОБ (ЕОБ x, y), z) болады. Басқаша айтқанда, алдымен екі шаманың ЕОБ, содан соң шыққан мәннің және үшінші санның ЕОБ табу керек (осыны дәлелдеп көріңіз).

Әлбетте, берілген есепті шешу үшін көмекші алгоритм ретінде екі санның ең көп ортақ бөлгішін табу алгоритмі болады. Сәйкесінше, бұл есеп өзімізге белгілі Евклид алгоритмінің көмегімен шығарылады (1.3-тарм. қараңыз), оны АТ-де процедура түрінде жазамыз:

```
процедура Евклид(тұтас M, N, K);
бас
    әзір M <>N
    цб
        егер M >N
        онда M : = M - N
        басқа N : = N - M
        кв
    цс;
    K := M
соңы
```

Мұндағы *M* және *N* процедураның нысандаулы параметрлері, яғни бұлар аргумент-параметрлер болады, ал K— нәтиже-өлшем. Алғашқы есепті шешудің негізгі алгоритмі келесі:

```
алг есеп
тұт a, b, c
бас енгізу (a, b)
    Евклид(a + b, |a-b|, c)
    Евклид(c, a * b, c) шығару (c)
соңы
```

Процедуралар. Шақырылатын бағдарламаға қатысы бойынша процедура сыртқы болатын АТ-ден Паскальда процедуралар ішкі бағдарламаларды сипаттау бөлімінде сипатталады. Бастапқы берілген есепті шешу бағдарламасы ТурбоПаскальда келесі түрде болады:

```
Program NOD1;
Var A, B, C : Integer;
Procedure Evklid(M, N : Integer; Var K : Integer);
Begin
    While M <> N Do
    If M > N
    Then M := M - N
    Else N := N - M;
    K := M
End;
Begin
    Write('A =');
```

```
    ReadLn(A);
    Write('B =');
    ReadLn(B);
    Evklid(A + B, Abs(A - B), C);
    Evklid(C, A * B, C);
    WriteLn('НОД=', C)
End.
```

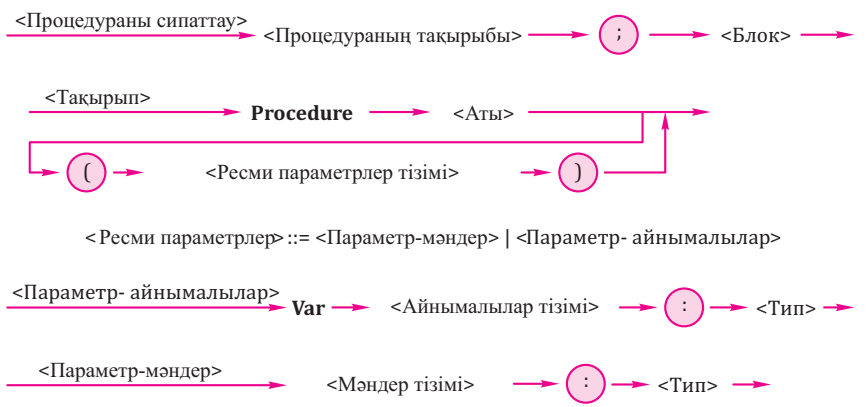
Бұл жағдайда негізгі бағдарлама мен процедура арасындағы аргументтер және нәтижелер алмасу параметрлер арқылы жүзеге асырылады (ресми және нақты). Алмасудың өзге де механизмі бар - ғаламдық айнымалылар, олар туралы бұдан әрі қарастырылады.

Процедураны сипатаудың синтаксистік диаграммасы 2.25. суретте берілген.

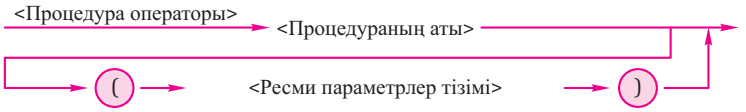
Диаграммадан процедураның параметрлері болуы да, болмауы да мүмкін екендігін көруге болады. Көбіне аргументер параметр-мәндер түрінде беріледі (бірақ олар параметр-айнымалылар болуы да мүмкін), нәтижелерді жіберу үшін параметр-айнымалылар пайдаланылады.

Процедура нәтиже ретінде шақырылатын бағдарламаға көптеген мәндерді беруі мүмкін (кей жағдайда – бір), бірақ бір де бір мәнді бермеуі де мүмкін.

Енді процедураға жүгіну ережесін қарастырайық. Процедураға жүгіну процедура операторының түрінде жүргізіледі (2.26сурет).



2.25.-сурет.Процедураны сипаттаудың синтаксистік диаграммасы



2.26.сурет. Процедураға жүгіну операторының құрылымы

Егер ресми параметрлері бар процедура сипатталатын болса, онда жүгіну де нақты параметрлерлі бар процедура операторы арқылы жүргізіледі. Ресми және нақты параметрлердің арасындағы сәйкестік ережесі мынадай болады: *саны бойынша, жүйелігі бойынша* және *типтер бойынша* сәйкес келу.

Ресми және нақты параметрлердің өзара әрекет етуінің бірінші нұсқасы *мәні бойынша жіберу* деп аталады: нақты параметрдің (өрнектің) мәні есептеледі және осы мән сәйкес келетін ресми параметрге беріледі. Ресми және нақты параметрлердің өзара әрекет етуінің екінші нұсқасы *аты бойынша жіберу* деп аталады: процедураны орындау кезінде ресми айнымалының аты сәйкес келетін нақты айнымалының атымен ауыстырылады (компиляцияланған бағдарламада айнымалының атына жады ұяшығының мекенжайы сәйкес келеді).

Қарастырылған мысалда *M* және *N* ресми параметрлері параметр-мәндер болады. Бұл процедураның аргументтері, оларға жүгінгенде бірінші рет оларға $A + B$ және Abs($A - B$) өрнектерінің мәндері, ал екінші рет - C және $A \cdot B$ мәндері сәйкес келеді. K параметрі процедура жұмысының нәтижесін алатын параметр-айнымалы болады. Процедураға жүгінудің екі жағдайында да сәйкес келетін нақты параметр ретінде ол арқылы негізгі бағдарлама нәтиже алатын C айнымалы болады.

Енді параметрсіз процедураны қолдану арқылы алғашқы берілген есепті шешетін бағдарламаның мысалын қарастырайық:

```
Program NOD2;
Var A, B, K, M, N : Integer;
Procedure Evklid;
Begin
     While M <> N Do
     If M > N
    Then M := M - N
    Else N := N - M;
    K := M
End.
```

```
Begin
    Write('A =');
    ReadLn(A);
    Write('B =');
    ReadLn(B);
    M := A + B;
    N := Abs(A - B);
    Evklid;
    M := K;
    N := A * B;
    Evklid;
    WriteLn('НОД тең', K)
End.
```

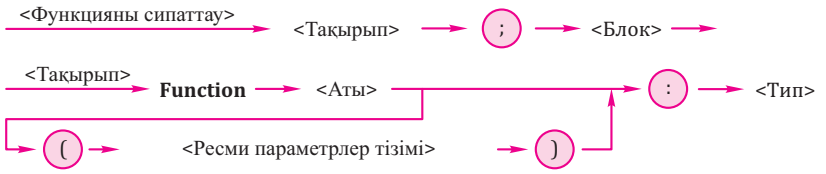
Бұл мысалды жете түсіну үшін сипаттаудың әрекет етусаласы ұғымын қарастыру қажет.

Кезкелген бағдарламалық объектінің (айнымалының, типтің, константаның және т.с.с.) сипаттаудың әрекет ету саласына сол сипаттау орналасқан блок жатады. Егер берілген блок басқасына салынған болса (ішкі бағдарлама), онда ондағы сипаттаулар *оқшауланған* болады және тек ішкі блоктың шегінде ғана әрекет етеді. Сыртқы блокта тұрған сипттаулар ішкі блокқа қатысы бойынша *ғаламдық* болып табылады. Егер ғаламдық түрде сипатталған объект ішкі блокта қолданыласа, онда оған сыртқы (ғаламдық) сипаттау таралады.

NOD1 бағдарламасында айнымалылар *M, N, K* - процедураның ішінде оқшауланған, ал *A, B, C*— ғаламдық болып саналады. Алайда процедураның ішінде *A, B, C* айнымалылар қоллданылмайды. Сырты блок пен процедураның арасындағы байланыс параметрлер арқылы жүзеге асырылады.

NOD2 бағдарламасында барлық айнымалылар ғаламдық болады. Evklid процедурасында бір де бір оқшауланған айнымалы (параметрлер де жоқ) жоқ, сондықтан ондағы қолданылатын айнымалылар *M* және *N* меңгеру операторы бойынша өз мәндерін негізінен бағдарламаның блогында алады. Нәтижені мәні экрнаға шығарылатын, *K* ғаламдық айнымалыда алады.

Параметрлер арқылы мәндерді беру механизмін қолдану процедураны ерекше әмбебап, негізгі бағдарламадан тәуелсіз етеді. Алайда кейбір жағдайларда, әсіресе ақпараттың үлкен көлемімен жұмыс істейтін процедуралар үшін мәндерді ғаламдық айнымалылар арқылы жіберу қолайлы болады. Бұл жағдайда ғаламдық өзара әрекет ету ЭЕМ жадын үнемдейді.



2.27. - сурет. Функцияны сипаттаудың синтаксистік диаграммасы

Параметрлер арқылы мәндерді беру механизмін қолдану процедураны ерекше әмбебап, негізгі бағдарламадан тәуелсіз етеді. Алайда кейбір жағдайларда, әсіресе ақпараттың үлкен көлемімен жұмыс істейтін процедуралар үшін мәндерді ғаламдық айнымалылар арқылы жіберу қолайлы болады. Бұл жағдайда ғаламдық өзара әрекет ету ЭЕМ жадын үнемдейді.

Функциялар. Енді ішкі бағдарлама-функциясын қарастырайық. Әдетте функция ішкі бағдарламаның нәтижесі ретінде скалярлы (қарапайым) шама болуы мүмкін жағдайда қолданылады. Нәтиженің типі функцияның типі деп аталады. ТурбоПаскальда жолдық типтегі функцияларды қолдануға болады. Функцияны сипаттаудың синтаксистік диаграммасы 2.27.-суретте берілген.

Процедурада сияқты, функцияда да ресми параметрлер тізімінде оның аргументтері болатын, параметр-айнымалылар және параметр-мәндер болуы мүмкін. Сонымен бірге аргументтер ғаламдық түрде берілетін болса, тізімдегі параметрлер болмауы мүмкін.

Функцияны қолдану арқылы алғашқы берілген есепті шығару бағдарламасы мынадай түрде болады:

```
Program NOD3;
Var A, B, Rez: Integer;
Function Evklid(M, N : Integer) : Integer;
Begin
      While M <> N Do
      If M > N
      Then M := M - N
      Else N : = N - M;
      Evklid : = M
End;
Begin
      Write('A = ');
      ReadLn(A);
     Write('B = ');
     ReadLn(B);
     Rez := Evklid(Evklid(A + B, Abs(A - B)), A * B);
     WriteLn('NODтең', Rez)
End.
```

Берілген мысалдан функцияның денесінде процедураға қарағанда *нәтиже функцияның атымен айнымалыға берілетіні* көрінеді.

Функцияға жүгіну өрнектегі операнд болып саналады және келесі түрде жазылады:

<функцияның аты> (<нақты параметрлер тізімі>)

Функцияда ресми және нақты параметрлер араснидағы сәйкес келу ережелері процедурадағыдай. Қарастырылған бағдарламаларды салыстыра отырып, NOD3 бағдарламасы басқаларына қарағанда біршама артықшылықтарға ие болатынын байқауға болады. Функция нәтижені бір меңгеру операторының орындауы арқылы алуға мүмкіндік береді. Мұнда сондай-ақ функцияға жүгіну кезінде нақты аргумент ретінде сол функцияның өзі болатыны көрсетілген.

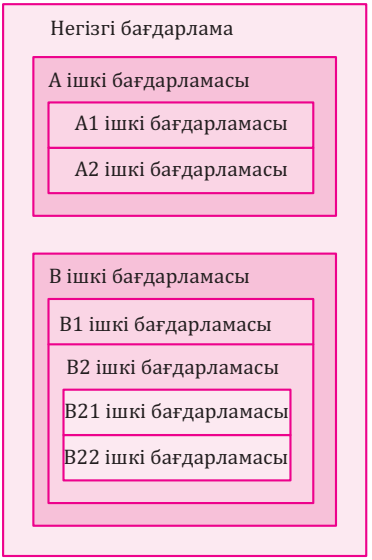
Стандартты Паскаль ережелері бойынша ішкі бағдарламадан шақырылатын бағдарламаға қайта келу бағдарламаны орындаудың соңына дейін барғанда болады (соңғы End). Алайда ТурбоПаскальда ішкі бағдарламадан оның кез келген жерінен шығуға мүмкіндік беретін құрал бар. Бұл оператор-процедура Exit. Мысалы, берілген екі заттық санның ең көбін анықтау функциясын былай сипаттауға болады:

```
Function Max(X, Y : Real): Real;
Begin
    Max := X;
    If X > Y Then Exit Else Max := Y
End.
```

Тағы да сипаттаудың әрекет ету саласы туралы. Паскальда келесі ереже бұлжытпай орындалады: кез келген бағдарламалық объект (константа, айнымалы, тип және т.б.бағдарламада қолданар алдында сипатталуы тиіс. Басқаша айтқанда, объектіні сипаттау оның бағдарламаның басқа үзінділерінде алғаш пайда болудың алдында болуы тиіс. Бұл ереже басқа ішкі бағдарламаларға да қатысты.

Ішкі бағдарламаларды сипаттаудың кейбір шарттық бағдарламада өзара орналасу құрылымы сызба түрінде 2.28.-суретте көрсетілген. Осы сызбаны пайдалана отырып, ішкі бағдарламаларды смпаттаудың әрекет ету саласы туралы мәселені қарастырып көрейік.

Кез келген ішкі бағдарлама оны сипатаудың әрекет ету саласының шегінде ғана қолданылуы мүмкін. Мысалы, *A* және *B* ішкі бағдарламаларының әрекет ету саласы - негізгі бағдарлама, сондықтан негізгі бағдарламадан *A* және *B* ішкі бағдарламаларына жүгінуге болады. Өз кезегінде, *B* ішкі бағдарламада *A* ішкі бағдарламаға жүгіну болуы ықтимал, алайда *A*-дан *B*-ға жүгінуге болмайды, өйткені *A* сипаттамасы *B* сипаттамасының алдынд аболады. *A*1 және *B*1 ішкі бағдарламалары *A* ішкі бағдарламасында оқшауланған және тек сол жерде ғана қолданылады, және *A*2-ден *A*1-ге жүгінуге болады, ал *A*1-ден *A*2-ге жүгінуге болмайды.



2.28.-сурет. Бағдарлама құрылымының мысалы

*B*1 ішкі бағдарламадан *A* ішкі бағдарламасына жүгінуге болады, өйткені оның сипаттауы *B*1 қатысты ғаламдық болып саналады, бірақ *A*1 жүгінуге болмайды, өйткені *A*1 сипаттаудың әрекет ету саласы *B* ішкі бағдарламаның блогына таралмайды.

*B*22 ішкі бағдарламадан тек *B*21, *B*1 және А ішкі бағдарламаларына ғана жүгінуге болады. (неліктен екенін өздерің табыңдар).

Егер бір ат сыртқы блокта та (ғаламдық) ішкі блокта да (оқшауланған) да сипатталатын болса, онда соңғы сипаттау (оқшауланған) бірінші сипатауды ішкі блок шегінде бөгейді. Екі бағдарламаны қарастырайық:

```
Program Examplel;
Var X :  Integer;
Procedure P;
Var X:   Integer;
Begin WriteLn('X = ', X)
End;
Begin X : = 1;
      P
End.
```

```
Program Example2;
Var X :  Integer;
Procedure P;
Begin
      WriteLn('X = ', X)
End;
Begin X : = 1;
      P
End.
```

Атқарған жұмыс нәтижесінде бірінші бағдарлама *X* = ... нәтижені береді, мұнда көп нүктенің орнына *x* белгісіз шамаға сәйкес бола-

тын кездейсоқ кез келген шама қойылады.

Екінші бағдарлама $X = 1$ деген нәтиже береді.

Бірінші ішкі бағдарламада X атымен берілген айнымалы ғаламдық түрде де, оқшауланған түрде де сипатталған, бірақ процедура ешқандай ат берілмеген оқшауланған айнымалының мәнін шығарады. Мұнда X сәйкестендіргішпен жадының екі әр түрлі ұяшығы сәйкес келетін екі мүлдем басқа шамалар белгіленеді.

Екінші бағдарламада X айнымалыбарлық бағдарлама үшін жалғыз болады және ол ғаламдық түрде сипатталады, сондықтан оған негізгі бағдарламада берілген 1 мәні ішкі бағдарламаға да беріледі.

Жаттығулар

1. Шеңбердің ауданын есептеудің ішкі бағдарламасын қолданып, процедурамен және функциямен ішкі және сыртқы радиустарының мәндері бойынша шеңбер ауданын есептеу бағдарламасының екі нұсқасын құру керек.

2. Екі нүктенің арасындағы кесіндінің ұзындығын есептеудің ішкі бағдарламасын қолдана отырып, үшбұрыш төбелерінің координаталары бойынша оның периметрін есептеу керек.

3. Үш тұтас сан берілген. Ішкі бағдарламаны қолдана отырып, олардың қайсысында цифр қосындысы көп екенін анықтау керек.

4. Кесіндінің ұзындығын есептеу ішкі бағдарлама-функциясын және үшбұрыш ауданын есептеудің ішкі бағдарлама-процедурасын қолдана отырып, Герон формуласы бойынша төбелерінің координаталлары бойынша дөңес үшбұрыштың ауданын анықтау керек.

2.11. РЕКУРРЕНТТІ ТІЗБЕКТЕРДІ ЕСЕПТЕП ШЫҒАРУ

Рекуррентті тізбегі. Рекуррентті тізбек ұғымы математика курсынан алынған. Ол келесі тәртіпте енгізіледі: сандық тізбектің басы болатын $a_1,..., a_k$ сандардың k белгілі болсын. Бұл тізбектің келесі элементтері мына формулалар бойынша есептеп шығарылады:

$$a_{k+i} = F(a_1,..., a_k);\ a_{k+2} = F(a_2,..., a_{k+1});\ a_{k+3} = F(a_3,..., a_{k+2})...,$$

мұндағы $F(...)$ — k аргументтерінен алынған функция.

$a_i = F(a_{i-1}, a_{i-2},..., a_{i-k})$ түріндегі формула

рекуррентті, ал k шамасы — рекурсияның тереңдігі деп аталады.

Басқаша айтқанда, *рекуррентті тізбек* — бұл әрқайсысы баста-

пқы k санамағанда, алдыңғылары арқылы есептеп шығарылатын сандардың шексіз қатары.

Реккурентті тізбектердің мысалы ретінде сәйкесінше арифметикалық және геометриялық прогрессиялар болады:

$$a_1 = 1,\ a_2 = 3,\ a_3 = 5,\ a_4 = 7,\ a_5 = 9,\ \ldots; \quad (2.1)$$

$$a_1 = 1,\ a_2 = 2,\ a_3 = 4,\ a_4 = 8,\ a_5 = 16,. \quad (2.2)$$

Берілген арифметикалық прогрессияның реккурентті формуласы:

$$a = a_{i-1} + 2.$$

Берілген геометриялық прогрессияның реккурентті формуласы:

$$a_i = 2a_{i-1}.$$

Екі жағдайда да рекурсия тереңдігі бірге тең (мұндай тәуелділікті бір қадамдық рекурсия деп те атайды). Тұтастай алғанда реккурентті тізбек бастапқы мәндердің жиынтығымен және реккурентті формуламен сипатталады. Мұның бәрін арифметикалық прогрессия үшін келесі түрде берілетін бір тармақтақталушы формулаға біріктіруге болады:

$$a_i = \begin{cases} 1, & \text{егер } i = 1; \\ a_{i-1} + 2, & \text{егер } i > 1, \end{cases}$$

ал геометриялық үшін -

$$a_i = \begin{cases} 1, & \text{егер } i = 1; \\ 2a_{i-1}, & \text{егер } i > 1. \end{cases}$$

Түрдің сандық тізбегі

1; 1; 2; 3; 5; 8; 13; 21; 34; 55 ...,

Математикада Фибоначчи сандары атауымен белгілі болған, онда үшінші элементтен бастап, әрбір сан алдыңғы екеуінің мәндерінің қосындысына тең болатын екіге тең тереңдігі бар реккурентті тізбек блып табылады (екі қадамдық рекурсия). Оны тармақталған түрде жазып көрсетейік:

$$f_i = \begin{cases} 1, & i = 1,\ i = 2; \\ f_{i-1} + f_{i-2}, & i > 2. \end{cases}$$

Реккурентті тізбектер туралы түсінікті енгізу бізге белгілі есептерге басқаша көзбен қарауға мүмкіндік береді.

Мысалы, *n*! тұтас саннан алынған факториалды келесі сандар қатарының *n*–дік элементінің мәні ретінде қарастыруға болады:

a0 = 1; a1= 1!; a2= 2!; a3= 3!; a4 = 4!; ...

Мұндай тізбектің рекуррентті жазылуы мынадай түрде болады:

$$a_i = \begin{cases} 1, & i = 0; \\ a_{i-1}\, i, & i > 0. \end{cases}$$

Реккурентті тізбектерді есептеп шығаруды бағдарламалау. Реккурентті тізбектермен келесі есептер байланысты болады:

1) берілген элементтің (*n*-дік) тізбегін есептеп шығару;

2) тізбектіңбелгілі бір бөлігін математикалық өңдеу (мысалы, алғашқы *n* мүшелердің қосындысын немесе көбейтіндісін есептеп шығару);

3) берілген жүйелік кесіндісінде белгілі бір қаситеттерді қанағаттандыратын, элементтердің санын есептеу;

4) белгілі бір талаптарды қанағаттандыратын бірінші элементтің нөмірін анықтау;

5) берілген санның жадында тізбек элементтерін есептеп шығару және сақтау.

Берілген тізбе толықтырылмай-ақ мейлінше көп кездесетін есеп түрлерін қамтиды. 1.4-есептерде сандық қатардың көптеген элементтерін бір мезгілде жадыда сақтау қажет емес: *оның элементтері бір айнымалыда бірін бірі ауыстырып, біртіндеп алынуы мүмкін.*

2.4. мысал. Арифметикалық прогрессияның *n*-ші элементін есептеп шығару керек (2.1).

Берілген есепті шығару бағдарламасы мынадай:

```
Var N, I: 0..MaxInt;
    A : Real;
Begin
   Write('N = ');
   ReadLn(N);
   A := 1;
   For I := 2 To N Do
   A := A + 2;
   WriteLn('A(', N:1, ')=', A:6:0)
End.
```

Рекуррентті формула $a_i = a_{i-1} + 2$ операторға көшті A:= A+ 2.

2.5. мысалПрогрессия қосындысына арналған формуланы қолданбай, геометриялықпрогрессияның (2.2) алғашқы элементтердің *n* қосындысын табу керек.

Берілген есепті шығару бағдарлама мынадай:

```
Var N, I: 0..MaxInt;
      A, S : Real;
Begin
      Write('N = '); ReadLn(N);
      A := 1;
      S : = A;
      For I := 2 To N Do
      Begin
            A : = 2 * A;
            S : = S + A
      End;
WriteLn('Сумма тең', S:6:0)
End.
```

Екіге тең тереңдікпен берілген рекуреннті тізбекті есептеп шығару кезінде бір айнымалымен шектеліп қалуға болмайды, мұны келесі мысалдан көруге болады.

2.6. мысал.Фибоначчи сандарының алғашқын ($n \geq 3$) баспаға шығару және олардың арасында қанша жұп сан бар екендігін есептеу қажет.

Берілген есепті шығару бағдарлама мынадай:

```
Var N, I, K, F, F1, F2 : 0..MaxInt;
Begin
      F1 := 1; F2 := 1;
      K := 0;
      WriteLn('F(1) = ', F1, ' F(2) = ', F2);
      For I := 3 To N Do
      Begin
            F := F1 + F2;
            WriteLn('F(', I : 1, ') = ', F);
            If Not Odd(F) Then K := K + 1;
           F1 := F2; F2 := F
End.
```

WriteLn('тізбектегі жұп сандардың саны ', K тең)

End.

Екі қадамдық рекурсияны жүйелі түрде есептеп шығару үшін үш айнымалы қажет, өйткені кезекті элементті есептеу үшін алдыңғы екеуінің мәндерін есте сақтау керек.

2.7. мысал. Берілген x заттық және ε шағын шама үшін (мысалы, ε = 0,000001) оған тек ε артық болатын қосындыларды ғана кіргізіп, келесі қатардың қосындысын есептеу керек

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...,$$

Математикадан мұндай шексіз қатардың қосындысы e^xтең болатын соңғы мәнге ие болатыны белгілі, мұндағы e= 2,71828... — натуралдық логарифмнің негізі. Бұл қатардың элементтері нөлге апаратын сандардың кему тізбегі арқылы берілгендіктен, қосуды ε-ден артық болмайтын абсолютті мән бойынша бірінші қосындыға дейін жүргізу керек.

Бұл өрнектегі қосылғыштарды

$$a_0 = 1;\ a_1 = x;\ a_2 = \frac{x^2}{2!};\ a_3 = \frac{x^3}{3!},\ ...,$$

Онда i –ші элемент үшін жалпылама формуласы келесідей болады:

$$a_i = \frac{x^i}{i!}.$$

Берілген тізбектің элементтері арасында рекуррентті тәуелділік бар екендігін бар екенін көру қиын емес. Оны тек түйсікпен ғана табуға болады, бірақ ресми түрде де шығаруға болады. Бірақ, ол үшін рекурсия – бір қадамды болатынын және әрбір келесі элемент алдыңғы элементті қандай да бір көбейткішке көбейту арқылы алынатынына көз жеткізу керек, яғни

$$a_i = Ka_{i-1}.$$

Жалпылама формуланы қолдана отырып, жазайық:

$$\frac{x^i}{i!} = K\frac{x^{i-1}}{(i-1)!},$$

бұдан

$$K = \frac{x^i/i!}{x^{i-1}/(i-1)!} = \frac{x}{i}.$$

Шын мәнісінде,

$$a_0 = 1;\ a_1 = a_0\frac{x}{2};\ a_3 = a_2\frac{x}{3}\ \ldots$$

Сәйкесінше, берілген рекуррентті тізбек келесідей берілуі мүмкін:

$$a_i = \begin{cases} 1, & \text{егері } i = 0; \\ a_{i-1}\dfrac{x}{i}, & \text{егері } i > 0. \end{cases}$$

Және, қойылған есепті шешетін бағдарламаны береміз:

```
Var   A, X, S, Eps : Real;
      I : Integer;
Begin
    Write('X = '); ReadLn(X);
    Write('Epsilon = '); ReadLn(Eps);
    A := 1; S := 0; I := 0;
    While Abs(A) > Eps Do
    Begin
        S := S + A;
        I : = I + 1;
        A := A * X/I
     End;
WriteLn('қатар сомасы тең', S : 10 : 4)
End.
```

Бір қадамды рекуррентті тізбектің бұл жерде де бұрынғыдай бір айнымалыда есептеледі.

Бұл бағдарламада циклді әрбір қайталап орындау *S* мәнін ізделіп отырған шамаға жақындатады (оның жазбасында мәнді сандарды нақтылайды). Мұндай есептеу үрдісі математикада *итерационды процесс* деп аталады. Сәйкесінше, итерационды есептеу процесін іске асырушы циклдер *итерационды циклдер* деп аталады. Оларды ұйымдастыру үшін While немесе Repeat операторлары қолданылады.

2.8. мысал. Берілген N натуралды және *x* заттық сан үшін (x > 0) өрнектің мәнін есептеу керек

$$\sqrt{x+\sqrt{x+...+\sqrt{x}}}.$$

Бұл жағдайда рекуррентілік аса байқалмайды. Оны индукция тәсілімен тауып көрейік. Іздеп отырған өрнек түр тізбегінің *N*-ші элементі деп санайық.

$$a_1=\sqrt{x};\ a_2=\sqrt{x+\sqrt{x}};\ a_3=\sqrt{x+\sqrt{x+\sqrt{x}}}\ \ ...,$$

бұдан келесі байланыс көрінеді:

$$a_2=\sqrt{x+a_1};\ a_3=\sqrt{x+a_2};\ ...;\ a_i=\sqrt{x+a_{i-1}}.$$

Енді берілген есепті шығару қиындық тудырмайды:

```
Var A, X : Real; I, N : Integer;
Begin
    Write('X = '); ReadLn(X);
    Write('N = '); ReadLn(N);
     A := Sqrt(X);
     For I := 2 To N Do
         A := Sqrt(X + A);
     WriteLn('OTBeT: ',A)
End.
```

Алайда барлық берілген есептерді шығаруға функцияны Паскальда рекурсивті анықтау мүмкіндігін қолдана отырып, өзге тұрғыдан да қарауға болады. Рекуррентті тізбек түріндегі арифметикалық прогрессияны сипаттаудан берілген элементті есептеп шығаруға арналған функцияны анықтау тәсілі тікелей туындайды.

Мұны бастапқы мәні a_0 және айырмасы *d* болатын арифметикалық прогрессияны анықтап, жалпы жағдай үшін орындайық:

$$a_i=\begin{cases} a_0, & \text{егер } i=1;\\ a_{i-1}+d, & \text{егер } i>1. \end{cases}$$

Тиісті ішкі бағдарлама-функция келесі түрде беріледі:

```
Function Progres(A0, D : Real; I : Integer): Real;
Begin
If I = 1
Then Progres := A0
```

```
        Else Progres := Progres(A0,D, I - 1) + D
End;
```

Келесі бағдарлама экранға мәндерін Fibon рекурсивті функция есептеп шығаратын, алғашқы 20 Фибоначчи сандарын шығарады:

```
Var K : Byte;
Function Fibon(N : Integer) : Integer;
Begin
    If (N = 1) Or (N = 2)
    Then Fibon := 1
    Else Fibon := Fibon(N - 1) + Fibon(N — 2)
End;
Begin
    For K := 1 To 20 Do WriteLn(Fibon(K))
End.
```

Рекурсивті функцияларды қолдану шоттың баяулауына әкелетінін атап өту қажет. Бұдан басқа, рекурсивті жүгінудің «жүру жолы» есте сақталатын стек ұзындығының жетіспеушілік проблемасы да болуы мүмкін.

Рекуррентті тізбектер көбіне әр түрлі эволюциялық міндеттерді, яғни уақыт барысында дамитын, қандай да бір үрдіс байқалатын міндеттерді шешу үшін де қолданылады. Осындай есепті қарастырайық.

2.9. мысал. Емдік ашығу кезінде науқастың салмағы 30 күн ішінде 96 кг – нан 70 кг түсті. Күнделікті салмақ жоғалту дене салмағына тепе–тең екені анықталды. Науқастың салмағы ашығу басталғаннан соң үшін к күннен кейін k = 1, 2,..., 29 үшін нешеге тең болатынын есепте шығару керек.

Науқастың-ші күнгі салмағын p_i (i= 0, 1, 2,., 30) деп белгілейік. Есептің шартынан p_0 = 96 кг, p_{30}= 70 кг болғаны белгілі. Егер K— бір күнде салмақ жоғалтудың тепе-теңдік коэффициенті деп алсақ, онда

$$p_1 = p_0 - Kp_0 = (1 - K)\, p_0;\ p_2 = (1 - K)p_1;$$

$$p_3 = (1 - K)p_2;\ ...;\ p_i = (1 - K)p_{i-1}$$

яғни келесі рекуррентті формула сипаттайтын тізбекті аламыз:

$$p_i = \begin{cases} 96, & \text{егер i} = 0; \\ (1 - K)p_{i-1}, & \text{егер 1} < i < 30. \end{cases}$$

К коэффициентті p_{30}= 70 шартын қолдану арқылы және «кері» алмастыруды орындау арқылы табуға болады:

$$p_{30} = (1 - K)p_{29}=(1 - K)^2p_{28}=(1 - K)^3p_{27} = ... =$$
$$= (1 - K)30p0 = (1 - K)^{30}96.$$

$(1 - K)^{30}96 = 70$ теңдеуінен табамыз:

$$1 - K = \left(\frac{70}{96}\right)^{\frac{1}{30}}.$$

Бұдан әрі қарай бағдарламалау белгілі бола бастайды:

```
Var     I : Byte;
        P, Q : Real;
Begin
    P := 96;
    Q := Exp(1/30 * Ln(70/96));
    For I := 1 To 29 Do
    Begin
      P := Q * P;
      WriteLn(I,'-ші күні— ', P : 5 : 3, ' кг')
    End
End.
```

Жаттығулар

1. Рекуррентті тізбек келесідей беріледі:

$$a_i = \begin{cases} 1, & \text{егер } i = 0; \\ a_{i-1}i + \dfrac{1}{i}, & \text{егер } i > 0. \end{cases}$$

Берілген n натурал сан үшін a_n мәнін табу керек.

2. Тізбек берілген

$$a_i = \begin{cases} 1, & \text{егер } i = 0,\ i = 1; \\ a_{i-2} + \dfrac{a_{i-1}}{i-1}, & \text{егер } i > 1. \end{cases}$$

1-ден 20 дейінгі элементтердің көбейтіндісін есептеу керек.

3. Рекуррентті тәсілдемені қолданып, 10-шы дәрежелі көпмүшеніңқосындысын Горнер формуласы бойынша есептеу керек:

$10x^{10}+ 9x^9+8x^8 + ... + 2x^2 + x =(((((((((10x + 9)\ x + 8)x + ... + 2)\ x + 1)x,$

мұндағы x — берілген заттық сан.

4. Берілген x заттық және N натурал сан үшін келесі тізбектік бөлшекті

есептеу керек:

$$x/\,(1 + x/\,(2 + x/\,(3 + x/\,(\ldots / (N + x))\ \ldots)\ .$$

5. 10^{-5}мәнінен асатын сандық қатардың барлық мүшелерін есептеу және шығару керек

$$1;\ \frac{1}{2!};\ \frac{1}{3!};\ \ldots;\ \frac{1}{N!},$$

6. $y = \sqrt{x}$ функциясын келесі рекуррентті формуламен анықталатын тізбектің шекті мәні ретінде есептеп шығаруға болады:

$$y_k = \frac{1}{2}\left(y_{k-1} + \frac{x}{y_{k-1}} \right) \quad k = 1.2\ \ldots \text{ үшін}$$

Мұнда кез келген бастапқы мән y_0 беріледі (k = $\sqrt{x}$ жуық болған дұрыс). ε дәлдігімен жақындатылған түбірдің мәніне $|y_k - y_{k-i}| < \varepsilon$ шарты орындалатын бірінші y_k жүзеге асырады. Есептеудің тиісті бағдарламасын құрыңдар.

2.13. ТАҢБАЛЫҚ ЖОЛДАРДЫ ӨҢДЕУ

Құрылымдандырылғандарға жататын деректердің түрі - жолдық (жол) деректер типін қарастырайық. Деректердің жолдық типіТурбоПаскальда бар және стандартты Паскальда жоқ екенін атап өткен жөн.

Жол — бұл таңбалардың тізбегі. Әрбір таңба жадының 1 байтын алады (ASCII коды). Жолдағы тағбалар саны оның ұзындығы деп аталады. Жолдың ұзындығы 0-ден 255 дейінгі диапазонда болуы мүмкін. Жолдық шамалар константалар және айнымалылар болады.

Жолдық константа — бұл апастрофқа алынған таңбалардың тізбегі. Мысалы:

```
' Бағдарламалау тілі ПАСКАЛЬ
' IBM PC — computer',
' 33-45-12'.
```

Жолдық айнымалы айнымалыларды сипаттау бөлімінде келесідей болып сипатталады:

Var <түрлендіргіш> : String[<жолдың максималды ұзындығы>].

Мысалы:

Var Name : String[20].

Жолдың ұзындығы сипаттауда берілмеуі де мүмкін. Бұл жағдайда ол максималды -255 таңбадан тұрады деп болжамдалады. Мысалы:

Var Slovo : String.

Жолдық айнымалы жадыда сипаттауда көрсетілген оның ұзындығына қарағанда 1 байт артық болады. Оның себебі бір (нөлдік) байт *жолдың ағымдағы ұзындығының* мәніне ие болатындығында. Егер жолдық айнымалыға ешқандай мән тағайындалмаса, онда оның ағымдағы ұзындығы нөлге тең болады.

Жолды таңбалармен толтыруға қарай оның ағымдағы ұзындығы арта түседі, бірақ ол сипаттау бойынша максималды мәннен артық болмауы тиіс.

Жол ішіндегі таңбалар бірден бастап индекстеледі (нөмірленеді). Әрбір жеке таңба төрт бұрышты жақшаға алынған индекспен берілген атпен сәйкестендіріледі. Мысалы:

Name[5], Name[i], Slovo[k+l].

Индекс оң константа, айнымалы шама және тұтас типтегі өрнек болуы мүмкін. Индекстің мәні бейнелеу шегінен шықпауы тиіс.

String типі және стандарттыChar типі үйлесімді келеді. Жолдар мен таңбалар тек бір ғана өрнектерде пайдаланыла алады.

Жолдық өрнектер жолдық константалардан, айнымалылардан, функциялардан және амал белгілерінен тұрады. Жолдық деректерге тіркесу амалдары мен қатынас амалдары қолданылуы мүмкін.

Тіркесу амалы (+) бірнеше жолды бір нәтиже беруші жолға біріктіру үшін қолданылады. Жолдық константаларға да, айнымалыларға да тіркесуді қолдануға болады. Мысалы:

'ЭВМ'+' IBM'+' PC'.

Нәтижесінде мынадай жол пайда болады:

'ЭВМ IBM PC'.

Нәтиже беруші жолдың ұзындығы 255 таңбадан артық болмауы тиіс.

Қатынастар амалы (=, <, >, <=, >=, <>) екі жолды салыстырады, нәтижесінде логикалық шама алынады (True немесе False). Қатынас амалының тіркесу амалына қарағанда басымшылығы төмен.Жолдарды салыстыру бірінші сәйкес келмейтін таңбаға дейін солдан оңға қарай жүргізіледі, және ең көп ретінде таңбалық кодтау кестесінде бірінші сәйкес келмейтін таңба ең көп нөмірге ие болған жол саналады.

Егер жолдардың ұзындығы әр түрлі болса, бірақ жалпы таңбалары сәйкес келсе, ең қысқа жол ұзын жолға қарағанда кем болып саналады. Ұзындықтары бойынша толықтай сәйкес келсе және бірдей таңбаларға ие болса, ондай жолдар тең болады. Мысалы:

Өрнек	Нәтиже
'cosml' < 'cosm2'	True
'pascal' > 'Pascal'	True
'Кілт_' <> 'Кілт'	True
'MS DOS' = 'MS DOS'	True

- Copy(S,Poz,N) функциясы—Poz позициясынан бастап, S жолынан N таңбалар N ұзындығының ішкі жолын шығарады. Мұндағы N және Poz—тұтас санды өрнектер. Мысалы:

Мәні s	өрнек	нәтиже
'ABCDEFG'	Copy (s, 2, 3)	'BCD'
'ABCDEFG'	Copy (s, 4, 4)	'DEFG'

- Concat (S1,S2,...,SN) функциясы S1,..., SNжолдарын бір жолға тіркеуді (конкатенация)орындайды. Мысалы:

Өрнек	Нәтиже
Concat('AA', 'XX', 'Y')	'AAXXY'

Length(S) функциясы—S жолдың ағымдағы ұзындығын анықтай-

ды. Нәтиже ретінде тұтас типті мәнді болады. Мысалы:

S мәні	Өрнек	Нәтиже
'test-5'	Length(S)	6
'(A+B)*C'	Length(S)	7

■ Pos (S1,S2) —функциясы S2 жолындаS1 шағын жолдың алғашқы көрінуін табады. Нәтижесінде S1 шағын жолдың бірінші таңбасы болатын, позиция нөміріне тең болатын тұтас сан алынады. Егер S1-де S2 шағын жол табылмаса, нәтиже 0-ге тең болады. Мысалы:

S2 мәні	Өрнек	Нәтиже
'abcdef'	Pos('cd', S2)	3
'abcdcdef'	Pos('cd', S2)	3
'abcdef'	Pos('k', S2)	0

■ Delete(S, Poz, N) процедурасы — Poz позициясынан бастап, Sжолынан N таңбаларды жояды. Мысалы:

S бастапқы мәні	Оператор	Соңғы мәні
'abcdefg'	Delete(S,3,2)	'abefg'
'abcdefg'	Delete(S,2,6)	'a'

Процедураны орындау нәтижесінде S айнымалыда жолдың ағымдағы ұзындығы кемиді.

■ Insert (S1, S2, Poz) процедурасы—Poz позициясынан бастап, S1 жолды S2 жолға қояды. Мысалы:

Бастапқы S2	Оператор	Соңғы S2
'ЭВМ PC'	Insert('IBM-',S2,5)	'ЭВМ IBM-PC'
'Сур.. 2'	Insert('N',S2,6)	'Сур. N2'

2.13. мысал. «ШАМА» сөзінен «БАР БОЛУЫ» сөзін алу бағдарламасы:

```
Program Slovo_1;
Var Sl1, Sl2 : String[10];
Begin
```

```
    Sl1 := 'ШАМА';
    Sl2 := Copy(Sl1,7,2) + Copy(Sl1,3,4) + Sl1[2];
    WriteLn(Sl2)
End.
```

2.14. мысал. «ЖОЛ» сөзінен «ТОР» сөзін алу бағдарламасы:

```
Program Slovo 2;
Var Sl : String[10];
Begin
    Sl := 'ЖОЛ';
    Delete(Sl,3,2);
    Insert('Е',Sl,2);
    WriteLn(Sl)
End.
```

2.15.мысалN жұлдызшалардан тұратын (мұндағы *N*— тұтас сан, 1 <*N*< 255) таңбалық жолды құрастыратын бағдарлама:

```
Program Stars;
Var  A : String;
     N, I : Byte;
Begin
     Write('Жұлдызша санын енгізіңіз');
     ReadLn(N);
     A := '';
For  I := 1 To N Do
     A := A + ' * ';
     WriteLn(A)
End.
```

Мұнда А жолдық айнымалыға алдымен бос жолдың мәні беріледі (' '), содан соң оған жұлдызшалар қосылады.

2.16. мысал цифрлердің таңбалық жолында «!» бірінші таңбаның алдында болатын санын есептеу бағдарламасы:

```
Program C;
Var  S : String;
     K, I : Byte;
Begin
     WriteLn('Жолды енгізіңіз');
     ReadLn(S);
```

```
K := 0;
I := 1;
While (I <= Length(S)) And (S[I] <> '!') Do
Begin
        If (S[I] >= '0') And (S[i] <= '9')
        Then K := K + 1;
        I := I + 1
End;
WriteLn( «!» таңбасына дейінгі цифрлердің саны K тең)
End.
```

Бұл бағдарламада K айнымалы цифрлерді есептегіш рөлінде болады, ал I айнымалы – цикл параметірінң рөлінде болады. Циклді орындау «!» таңбасына алғаш шыққан кезде немесе (егер жолда ондай таңба болмаса) жолдың соңына шыққанда аяқталады. Егер '0' ≤S[I] ≤'9' қатынасы ақиқат болса, S[I] таңбасы цифр болып табылады.

2.17. мысал. ' A ⊕ B =' түрінде таңбалық жол берілген. МұндағыA және B — ондық цифрлер, ⊕ белгісімен (+, -, *) амалдарының белгілерінің бірі белгіленген. Басқаша айтқанда, екі бір таңбалы санмен және тең белгісімен арифметикалық өрнек берілген, мысалы: '5+7 ='. Машина осы өрнектің мәнін тауып, «=» белгісінен кейін нәтижені шығару талап етіледі.Бұл жерде тек тұтас сандар болу үшін бөлу амалы қарастырылмайды.

Мұндай есепті шешу бағдарламасы *интерпретатор* деп аталады. Интерпретатор жолдың мазмұнын ашуы және тиісті арифметикалық амалды орындауы тиіс. Ол бастапқы таңбалық ақпараттан сандық ақпаратпен жұмыс жасауға көшуі керек.

Берілген пішімге сәйкес жол тек төрт таңбадан тұрады деп ұйғарылса, шешу бағдарламасы тым жеңіл болады:

```
Program Interpretator;
Var Str : String[4];
    A, B : 0..9;
    C : —100..100;
Begin
{---Бастапқы жолды енгізу---}
    WriteLn('өрнекті енгізіңдер!');
    WriteLn;
    Read(Str);
{---Цифрлік таңбаларды  сандарға түрлендіру---}
```

```
        A := Ord(Str[1]) - Ord('O');
        B := Ord(Str[3]) - Ord('O');
{---Арифметикалық амалды орындау---}
        Case Str[2] Of
                '+' : C : = A + B;
                '-' : C : = A - B;
                '*' : C := A * B
        End;
{---Нәтижені шығару ---}
        WriteLn(C:2)
End.
```

Берілген бағдарламада таңдау операторында селектордың рөлін Str[2] таңбалық шама атқарады. Егер ол «+» тең болса, C := A+ B операторды орындаймыз; егер ол «-» тең болса , C:= A–B операторды орындаймыз; егер ол «*» тең болса, C:= A* B операторды орындаймыз. Str[2] кез келген басқа мәндері үшін меңгерудің бір де бір операторы орындалмайды, және C айнымалының мәні анықталмаған болып қалады.

Бұл бағдарламада Case операторының толық емес түрі, яғни Else тармағысыз қолданылады. Дәл емес таңбаны алу туралы хабарламаны беру бағдарламасы Str[2] мынадай болады:

```
Case Str[2] Of
        '+' : C : = A + B;
        '-' : C : = A - B;
        '*' : C : = A * B
ElseWriteLn('операцияның дұрыс емес белгісі')
End;
```

Енді Interpretator бағдарламасы қалай орындалатынын қарастырайық.

Жолды енгізген соң цифрлік таңбалар тиісті ондық сандарға ауыстырылады. Содан соң амал белгісі түсіндіріледі. Осы түсіндіруге байланысты үш арифметикалық амалдың біреуі орындалады. Бұдан әрі нәтиже «=» таңбасынан кейін экранға шығарылады.

Жаттығулар

1. Тіркесу амалын және Copy функциясын қолданып, «дискжетек» сөзінен «балауыз» сөзін алу бағдарламасын құру.

2. Delete және Insert процедураларын қолданып, «амал» сөзінен «ереже»

сөзін алу бағдарламасын құру.

3. Берілген сөзде бірінші және соңғы таңбаларды «*» таңбасына ауыстыру.

4. Берілген сөзде бірінші және соңғы таңбаларды ауыстыру.

5. Берілген сөзге қанша әріп бар, «!» сонша таңба қосу керек (мысалы, «АЛАҚАЙ!» жолынан «АЛАҚАЙ!!!» алу).

6. Берілген жолда әрбір таңбадан кейін аралық қою.

7. Енгізілген сөздің әрбір әрпін екіге арттыру.

8. Енгізілген жолды аудару (мысалы «ДИСК» жолынан «КСИД» алу).

9. Берілген жолды барлық аралықты жою.

10. Тұтас санның жазбасы түрінде берілген жолды тұтас типтің тиісті шамасына ауыстыру бағдарламасын құру.

2.14. МАССИВТЕРМЕН ЖҰМЫСТЫ БАҒДАРЛАМАЛАУ

Күнделікті және ғылыми тәжірибеде кестелік түрде берілген ақпаратпен жиі ұшырасуға болады. Белгілі бір жылға берілген температуралардың орташа айлық мәндері 2.7.-кестеде берілген. Реттелген сандардың тізбегі түрінде берілген мұндай кесте сызықтық деп аталады. Бұл деректерді қандай да бір математикалық өңдеу қажет болған жағдайда, оларды белгілеу үшін әдетте индекстік символиканы енгізеді. Мысалы, T_1 қаңтардың температурасын (бірінші айды), ал T_5 — мамырдың және т.с.с температурасын белгілейді. Жалпы алғанда, кестеде берілген мәндердің жиыныы мынадай түрде белгіленеді:

$$\{T_i\},\ i = 1, ..., 12.$$

2.7. –кесте. Температуралардың сызықтық кестесі

Жылдың айы	1	2	3	4	5	6	7	8	9	10	11	12
Температура, °С	-21	-18	-7,5	5,6	10	18	22,2	24	17	5,4	-7	-18

2.8.-кесте. Температуралардың тік бұрышты кестесі

Жыл	Жылдың айы											
	1	2	3	4	5	6	7	8	9	10	11	12
1981	-23	-17	-8,4	6,5	14	18,6	25	19	12.3	5,6	-4,5	-19
1982	-16	-8,5	-3,2	7,1	8,4	13,8	28,5	21	6,5	2	-13	-20
1983	-9,8	-14	-9,2	4,6	15,6	21	17,8	20	11,2	8,1	-16	-21
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1990	-25	-9,7	-3,8	8,5	13,9	17,8	23,5	17,5	10	5,7	-14	-20

(1, 15, *i* және т.б.) элементтердің реттік нөмірлері индекстер деп аталады. Индекстелген шамалардыматематикалық өңдеу үшін жазу кезінде қолданған қолайлы. Мысалы, орташа жылдық температураны есептеу үшін келесі фрмула қолданылады:

$$T_{\text{ср}} = \frac{1}{12}\sum_{i=1}^{12} T_i.$$

Енді соңғы 10 жылға берілген орташа айлық температура туралы ақпаратты жинау керек болды делік, мысалы 1981 жылдан 1990 жылға дейін. Ол үшін бағандар жылдарға, ал жолдар – айларға сәйкес келетін, тік бұрышты кестені қолданамыз (2.8-кесте).

Ұтымды таңдалған деректерді жазу нысаны қажет ақпаратты алудың тиімділігін және жылдамдығын қамтамасыз етеді.Мысалы, 2.8-кестеден қай жылы қаңтар айы ең суық немесе қай айда берілген он жылдық бойынша ең тұрақсыз температуралық режим болғанын оңай білуге болады.

Осыған ұқсас кестеде сақталатын мәнддер үшін екі индексті белілеуді қолданған тиімді. Мысалы, $H_{1981,2}$— 1981 ж. ақпан айындағы температураны белгілеу және т.с.с. Сонымен бірге кестеде берілген барлық мәліметтердің жиынтығын математикалық тұрғыдан мынадай түрде жазуға болады:

$$\{H_{i,j}\};\ i=1981\ \ldots\ 1990;\ j=1\ \ldots\ 12$$

Мұндай деректерді өңдеу екі индексті шамаларды қолдану арқылы жүргізіледі. Мысалы, 10 жылдағы наурыз айының орташа температурасы

$$H_{\text{орт.наур}} = \frac{1}{10}\sum_{i=1981}^{1990} H_{i,3},$$

ал он жылдық кезеңге берілген орташа температура

$$H_{\text{орт.}} = \frac{1}{10}\sum_{i=1981}^{1990}\left[\frac{1}{12}\sum_{j=1}^{12}H_{i,j}\right] = \frac{1}{120}\sum_{i=1981}^{1990}\sum_{j=1}^{12}H_{i,j}.$$

Паскальда кестелердің баламасы ретінде *тұрақты тип* немесе *массив* деп аталатын деректердің құрылымдандырылған типі болады. Кесте сияқты массив те ортақ атқа ие болатын, нөмірленген бір типті мәндердің жиынтығы түріне болады. Массивтің элементтері индекспен берілген айнымалылар сияқты белгіленеді. Индекстер төрт бұрышты жақшалардың ішінде массивтің атынан кейін белгіленеді. Мысалы:

T[1], T[5], T[i], H[1981, 9], H[i, j]

Сызықтық кестені сақтайтын массив *бір өлшемді*, ал тік бұрышты кестені сақтайтын – *екі өлшемді* деп аталады. Бағдарламаларда бұдан да көп өлшемді массивдер қолданылады.

Массив элементтерінің типі оның *базалық типі* деп аталады. Әдетте, температуралардың қарастырылған массивтері үшін базалық тип ретінде заттық болады (Real).

Массивтерді сипаттау. Тұрақты типтегі айнымалы айнымалыларды сипаттау бөлімінде келесі түрде сипатталады:

Var <түрлендіргіш> : **Array** [<индекс типі>] **Of**
<құрамдас типі>

Көбіне индекс типі ретінде интервалды қолданылады. Мысалы, бұрынырақ қарастырылған орташа айлық температуралардың бір өлшемді массивін келесі түрде сипаттауға болады:

Var T : **Array** [1..12] **Of** Real;

Массивті сипаттау оның жадыда орналасуын және бағдарламада әрі қарай қолданылуын анықтайды. Массивтің тізбектік элементтері жадының тізбек ұяшықтарында орналасады (T[1], T[2] жіне т.с.), және индекс мәндері 1..12 диапазонынан шықпау керек. Индекс ретінде тиісті типтегі кез келген өрнек қолданылуы мүмкін. Мысалы, T[i + j], T[m div 2].

Индекстің типі Integer басқа, кез келген скалярлы реттік тип болуы мүмкін. Мысалы, бағдарламада келесі сипаттаулар болуы мүмкін:

Var Cod : **Array** [Char] **Of** 1..100;
L : **Array** [Boolean] **Of** Char;

Берілген бағдарламада массив элементтерінің келесі белгіленулеріне жол беріледі:

Cod['x']; L[True]; Cod[Chr(65)]; L[a>0].

Кейбір жағдайларда индекс ретінде аталатын деректер типін қолдану ыңғайлы болады. Мысалы, бір мектептің төрт онынші сыныптарының оқушылары жайлы деректер келесі ауқымда сақталуы мүмкін:

```
Type Index = (A, B, C, D);
Var Class 10 : Array [Index] Of Byte;
```

Және, егер, мысалы, Class_10[a] элементі 35 тең болса, бұл 10а сыныбында 35 адам бар дегенді білдіреді. Мұндай индекстеу бағдарламаны айтарлықтай жандандырады.

Көбіне деректердің құрылымдандырылған типіне типтер бөлімінде ат беріледі, ол кейін айнымалыларды суреттеу бөлімінде қолданылады:

```
Type Mas1 = Array [1..100] Of Integer;
     Mas2 = Array [-10..10] Of Char;
Var Num : Mas1; Sim : Mas2;
```

Бұған дейін элементтер типі скалярлы болатын бір өлшемді массивтер жайында сөз қозғалған еді.

Көп өлшемді массив Паскальда элементтерінің типі массив болатын (массивтердің массиві) бір өлшемді массив ретінде түсіндіріледі. Мысалы, бұрын қарастырылған тік бұрышты кестені келесі түде сипатталған массивте сақтауға болады:

```
Var H : Array[1981..1990] Of Array[1..12] Of
Real;
```

Осы массивтің кейбір элементтерін таңбалауға мысал келтірейік:

H[1981][1]; H[1985][10]; H[1990][12].

Алайда көбіне екі өлшемді массивтің элементтерін таңбалаудың келесі, балама түрі қолданылады:

H[1981, 1]; H[1985, 10]; H[1990, 12].

МұндағыайнымалыH[1981] кестенің бүкіл бірінші жолын, яғни 1981 ж. температуралардың барлық ауқымын таңбалайды.

Берілген екі өлшемді массивті сипаттаудың балама нұсқасы ретінде келесі болады:

```
Type Month = Array [1..12] Of Real;
     Year = Array [1981..1990] Of Month;
Var H : Year;
```

Берілген массивті сипаттаудың неғұрлым қысқа нұсқасы келесі түрде беріледі:

```
Var H : Array[1981..1990, 1..12] Of Real;
```

Осыған ұқсас элементтері екі өлшемді массив болатын үш өлшемді массивті бір өлшемді массив түрінде анықтауға болады. Мысалы:

```
Var A : Array[1..10, 1..20, 1..30] Of Integer;
```

Бұл $10 \cdot 20 \cdot 30 = 6\ 000$ тұтас сандардан тұратын және жадыда $6000 \cdot 2 = 12\ 000$ байт орын алатын массивтің сипатталуы. Паскальда массивтің өлшемділігіне жоғарыдан шектеу қойылмайды. Алайда Паскальдың әрбір нақты іске асыруында массивтерге бөлінетін жадының көлемі шектеледі. ТурбоПаскальда бұл шектеу 64 Кбайтты құрайды.

Математикадағы ұқсастықтар бойынша бір өлшемді сандық массивтерді көбіне векторлар, ал екі өлшемді масситверді – қалыптама деп атайды.

Паскальда динамикалық массивтерді, яғни өлшемді бағдарламаны орындау барысында анықталатын массивтерді қолдануға жол берілмейді. Массивтің өлшемін өзгерту бағдарлама мәтініндегі өзгерістер мен қайталама компиляция арқылы жүргізіледі. Мұндай өзгерістерді жеңілдету үшін константалар бөлімінде индекстік параметрлерді анықтайды:

```
Const Imax = 10; Jmax = 20;
Var Mas : Array [1.. Imax, 1.. Jmax] Of Integer;
```

Енді Mas массив өлшемдерін және осы өлшемдермен байланысты барлық бағдарлама операторларын өзгерту үшін бағдарламада тек бір жолды – константалар бөліміне түзету енгізу жеткілікті.

Массивпен бірыңғай тұтас ретінде әрекет ету. Мұндай әрекеттерді тек екі жағдайда ғана жүзеге асыру мүмкін болады:

- бір массивтің мәнін екіншісіне беру;
- «тең», «тең емес» қатынас амалдарында.

Екі жағдайд да массивтердің бірдей типтері болуы тиіс (индекстер типі және элементтер типі). Мысалы:

Var P, Q : **Array** [1..5,1..10] **Of** Real;

Меңгеру амалын орындау кезінде

P := Q

P массивтің барлық элементтері Q массивтікң сәйкес элементтеріне тең болады.

Бұрын атап өтілгендей, көп өлшемді массивтерде индекспен берілген айнымалы тұтас массивті білдіруі мүмкін. Мысалы, егер H кестеде 1989 ж. бойынша деректерді 1981 ж. деректердегідей жасау керек болса (тоғызыншы жолға бірінші жолдың мәнін беру), онда оны мынадай түрде жасауға болады:

H[1989] := H[1981].

Кестеде бұл жолдардың мәндерінің орындарын ауыстыруды сол типтегі үшінші айнымалы арқылы болады:

P := H[1989]; H[1989] := H[1981]; H[1981] :=P;

Мұндағы P келесідей сипатталған:

Var P : **Month**

Бағдарламаларда массивтерді өңдеу құрамдастары бойынша жүргізіледі.

Массивтерге мәндерді енгізуге мысал келтірейік:

```
For I : = 1 To 12 Do
      ReadLn(T[I]);
For I : = 1 To IMax Do
For J : = 1 To JMax Do
      ReadLn(Mas[I, J]);
```

Мұнда әрбір келесі мән жаңа жолдан енгізіледі. Жолдық енгізу үшін Readoператоры қолданылады.

Осыған ұқсас индекстік айнымалы бойынша циклде массивтің мәндерін енгізу ұйымдастырылады. Мысалы:

For I := 1 **To** 12 **Do** Write(T[I]:8:4);

Бағдарламаның келесі үзіндісі қалыптаманы экранға жолдық шығаруды ұйымдастырады:

```
For I := 1 To IMax Do
Begin
      For J : = 1 To JMax Do
```

```
            Write(Mas[I,J]:6);
        WriteLn
End.
```

Қалыптаманың кезекті жолына басып шыққан соң WriteLn операторы параметрлерсіз меңзерді жаңа жолдың басына көшіреді. Соңғы мысалда экрандағы қалыптама, егер JMax 12-ден аспаса (неліктен екенін өздерің түсіндіріп көріңдер), төртбұрышты кесте түрінде қалыпты түрде алынатынын ескере кеткен жөн.

Массивтерді өңдеудің типтік бағдарламаларының бірнеше мысалдарын қарастырайық.

2.18. мысал. Бұрын берілген орташа айлық температуралар массиві үшін T[1... 12] орташа жылдық температураны және осы мәннен айлық ауытқуларды есептеп шығару қажет.

Берілген есепті шешу бағдарлмасы мынадай болады:

```
Program Example;
Cont N = 12;
Type Vec = Array [1..N] Of Real;
Var T, Dt : Vec;
      St : Real;
      I : Integer;
Begin{ Бастапқы деректерді енгізу }
      WriteLn(Температуралар кестесін енгізіңдер);
      For I : = 1 To N Do
      Begin
            Write(I:2, ':');
            ReadLn(T[I])
      End.
{---Орташа температураны есептеу---}
      St:= 0;
      For      I := 1 To N Do
            St := St + T[I];
            St := St/N;
{---орташадан ауытқу кестесін есептеу---}
      For       I : = 1 To N Do
            Dt[I] := St - T[I];
{---Нәтижелерді шығару---}
      WriteLn('Орташа температуратең ', St:6:2);
```

```
        WriteLn;
        WriteLn('Орташа температурадан ауытқу');
        For I : = 1 To N Do
    WriteLn(I:1, ': ', Dt[I]:6:2)
    End.
```

Осы бағдарлама бойынша кез келген бір өлшемді заттық массив үшін орташа мәнді және мәннен ауытқу векторын есептеуге болады. Массивтің өлшемін реттемелеу тек константалар бөлімін түзету арқылы жүзеге асырылады.

2.19.мысал. Бұрын құрастырылған температуралар массивінен максималды элементті таңдауға, яғни ең жоғары температура мен оған сәйкес келетін айдың нөмірінтаңдауға болады.

Бұл есепті шешу алгоритмінің идеясы мынадай: Tmax заттық айнымалыда максималды температураны табу үшін алдымен T[1] массивтің бірінші мәні енгізіледі. Содан соң кезек бойынша TMax мәні температуралар массивінің қалған элементтерімен салыстырылады, және TMax артық болатын әрбір мәні осы айнымалыға беріледі. Ең жылы айдың нөмірін табу үшін NumMax тұтас айнымалыға температуралар массиві элементінің нөмірін TMax оның мәнін енгізудің әрбір жағдайында жіберу керек.

Берілген есепті шешу бағдарлмасы мынадай болады:

```
TMax := T[1];
NumMax := 1;
For  I := 2 To 12 Do
     If T[I]> Tmax
     Then
     Begin
              TMax := T[I];
              NumMax := I
End.
```

Температуралар массивінде максималдыға тең болатын бірнеше мән болса, онда NumMax осы элементтерден бірінші нөмір алынатынын айта кетейік. Соңғы күнді табу үшін If операторында «>» қатынас белгісін «> =» белгісіне ауыстыру керек.

2.20.мысал. Массивті іріктеуді орындау қажет, яғни бір өлшемді *X* массивінде *N* элементтерден мәндердің орнын ауыстыруды олар-

дың артуы бойынша: $X_1 \leq X_2 \leq ... \leq X_N$ орналастырып, жүргізу керек.

Іріктеу алгоритмдерінің біршама классы бар. Көпіршік әдіссі деп аталатын алгоритмді сипаттайық.

Массив элементтерінің іргелес жұптарын жүйелі түрде тәртіпке келтіру жүргізіледі: X_1 және X_2, X_2 және X_3, ..., X_{N-1}и X_N. Нәтижесінде максималды мән X_N ауысады.Содан соң дәл осы процедураны XN_ jжәне т.с.с. алғанға дейін, X_1 және X_2 екі элементтен тұратын тізбекке жеткенше, қайталайды. Мұндай алгоритм екі салынған циклдің құрылымына ие болады, сонымен бірге ішкі цикл ұзындықтың айнымалысы (қысқарылатын) болады.

Берілген есепті шешу бағдарлмасы мынадай болады:

```
For I := 1 To N _ 1 Do
For K := 1 To N _ I Do
    If X [K] > X [K + 1]
    Then
    Begin
            A := X[K];
            X[K]:=X[K + 1];
            X[K + 1] := A
    End.
```

2.21.мысал. Бұрын сипатталған10 жылға берілгенорташа айлық температуралардың екі өлшемді массиві үшін қай жылы ең жылы жаз болғанын, яғни қай жылы жазғы айлардың ең көп орташа температурасы болғанын анықтау қажет.

Алгоритмді шешу идеясы мынадай болады: алдымен S векторында 10 жылға берілген жаз айларының орташа температурасын алу керек, содан соң осы векторда ең көп элементтің нөмірін табу керек. Осы ізделіп отырған жыл болады.

```
Program Example 2;
Type Month = Array [1..12] Of Real;
     Year = Array[1981..1990] Of Month;
VarH: Year;
     S: Array[1981..1990] Of Real;
     I, J, K: Integer;
Begin {---Пернетақтадан деректерді енгізу---}
     For I := 1981 To 1990 Do
```

```
        For J := 1 To 12 Do
        Begin
        Write(J:2, '.', I:4,': ');
        ReadLn(H[I,J])
        End.
{---Жазғы орташа температураларды есептеп шығару---}
        For I : = 1981 To 1990 Do
        Begin
                S[I] := 0;
                For J:= 6 To 8 Do
                S[I] := S[I] + H[I, J];
                S[I] := S[I]/3
        End.
{---Ең жылы жаз болған жылды анықтау---}
        K : = 1981;
        For I : = 1982 To 1990 Do
                If S[I] >S[K] Then K := I;
WriteLn('Ең жылы жаз ', K,'-шы жылы болды')
End.
```

Жаттығулар

1. {zj}, i = 1, ..., 50 векторы берілген. Оның ұзындығын есептеу керек:

$$L = \sqrt{z_1^2 + z_2^2 + ... + z_{50}^2}.$$

2. 10-шы дәрежелі көпмүшені Горнер формуласы бойынша есептеу керек:

$a_{10}x^{10} + a_9x^9 + ... + a_1x + a_0 = ((... (a_{10}x + a_9) x + a_8) x + ... + a_1) x + a_0.$

3. $\{x_i\}$, i = 1, ..., 20 векторы үшін мәндері [0; 1] интервалында жатқан құрамдастар санын табу керек.

4. Артуы бойынша ретке келтірілген, берілген екі векторды олардың реттілігі сақталатындай етіп, $\{x_i\}$, $\{y_i\}$, i = 1, ..., 10 $\{z_i\}$, i=1, ., 20 бір векторға біріктіру.

5. 100 тұтас сандардан тұратын, берілген массив үшін алдымен онда бірнше рет кездесетін барлық сандарды, содан соң онда бір рет қана кездесетін барлық сандарды шығару керек.

6. Өлшемі 10 × 10 тұтас сандық қалыптаманың максималды элементінің мәнін және индексін табу керек.

7. Өлшемі 5 × 10 екілік қалыптаманың ең көп нөлі бар жолдардың нөмірін анықтау керек.

8. Өлшемі 5 × 10 екілік қалыптаманың барлық жолдарын олардың элементтерінің кемуі бойынша реттеу керек.

9. Өлшемі 5 × 5 тұтас сандық қалыптаманы тасымалдау, яғни оны басты диагоналына қатысты бейнелеу керек.

10. Өлшемі 5 × 10 екілік қалыптамада сәйкес елетін жолдарды табу керек.

2.15. ЖОБАЛЫҚ ТАПСЫРМА: «ХАНОЙ МҰНАРАСЫ»

Тапсырманың мақсаты— «Ханой мұнарасы» атауымен белгілі болған, комбинаторлық (есептелмейтін) есепке арналған рекурсивті процедураны қолдану мысалымен танысу.

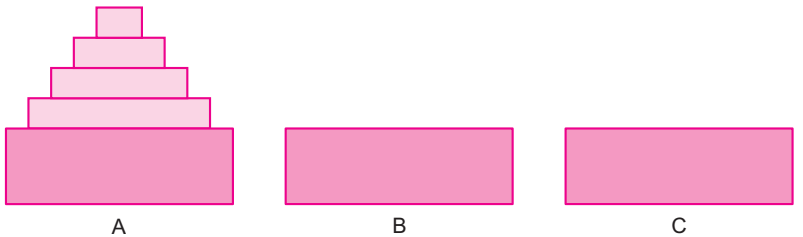
1-кезең. Міндет қоюды ұғыну.

Әдебиетте «Ханой мұнарасы» атауымен белгілі классикалық есепті қарастырайық (2.29 сурет).

Алаңқайда (оны А деп атайық) өлшемі іргетасынан төбесіне қарай кеми беретін, дисктерден тұратын пирамида бар. Осы пирамиданы дәл сол қалпында В алаңқайына көшіру перек. Бұл жұмысты орындау кезінде келесі шектеулерді сақату қажет:

- пирамиданың үстінен алынған тек бір дискті ғана ауыстыру керек;
- дискті тек алаңқайдың іргетасына, немесе одан гөрі үлкенірек дискке қою керек;

көмекші ретінде С алаңқайын пайдалануға болады.



2.29. – сурет. «Ханой мұнарасы» туралы міндет

«Ханой мұнарасы» атауы аңызбен байланысты, ол аңыз бойынша ерте заманда бір Ханой ғибадатханасының сопылары осы ережелер бойынша 64 дисктен тұратын мұнараны ауыстыруға бел буды. Сопылар әлі күнге дейін жұмыс істеуде және әлі ұзақ уақыт жұмыс істейді!

Берілген екі диск үшін берілген есепті шығару қиындық тудырмайды. Дискті ауыстыруды мысалы, А алаңқайынан В алаңқайына көшіруді А $\Rightarrow$ В түрінде белгілеп, осы жағдайға арналған алгоритмді жазайық:

$$\text{A} \Rightarrow \text{C};\ \text{A} \Rightarrow \text{B};\ \text{C} \Rightarrow \text{B},$$

Яғни, небары 3 жүрістен тұрады.

Үш дискіге арналған есепті шығару алгоритмі ұзынырақ:

$$\text{A} \Rightarrow \text{B};\ \text{A} \Rightarrow \text{C};\ \text{B} \Rightarrow \text{C};\ \text{A} \Rightarrow \text{B};\ \text{C} \Rightarrow \text{A};\ \text{C} \Rightarrow \text{B};\ \text{A} \Rightarrow \text{B},$$

яғни енді жеті жүріс.

k дисктер үшін *N* жүріс санын келесі рекуррентті формула бойынша анықтауға болады:

$$N(1) = 1;\ N(k) = 2 \times N(k - 1) + 1.$$

Мысалы, $N(10) = 1023$, $N(20) = 104857$, ал $N(64) = 18\,446\,744\,073\,709\,551\,615$, яғни ханойлық сопыларға сонша ауыстыру жасау қажет. Осы санды оқып көріңдер.

2-кезең. «Ханой мұнарасы» туралы есепті шығарудың рекурсивті алгоритмін ұғыну.

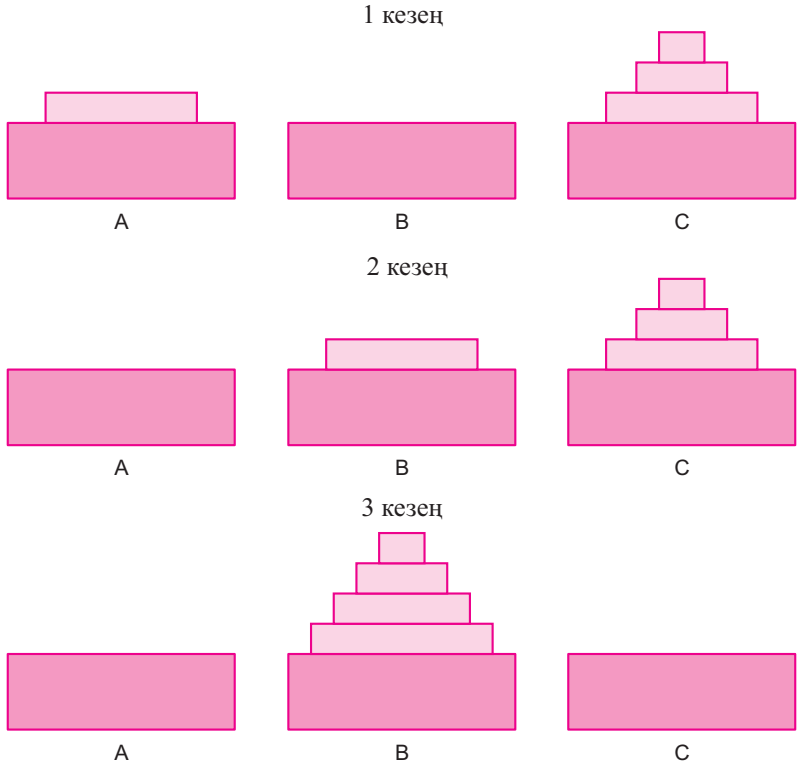
A алаңқайында *N* диск бар делік, сонда есепті шығару алгоритмі мынадай болады:

1. Егер $N = 0$, онда ештеңе жасаудың қажеті жоқ.
2. Егер $N > 0$, онда:

- *N* - 1 дискті *B* арқылы *C*-ғажылжыту керек;
- Дискті *A*-дан *B*-ға ауыстыру керек (А $\Rightarrow$ В);
- *N* - 1 дискті *A* арқылы *C*-дан *B*-ға жылжыту керек.

Алгоритмнің екінші тармағын орындау кезінде 2.30-суретте көрсетілген дисктерді ауыстырудың үш кезеңіне жүйелі түрде ие боламыз.

Рекурсивті әдістердің мәнісі - есепті өзіне өзін келтіру. Сендер Паскальда функциялар мен процедураларды рекурсивті анықтау мүмкіндігі бар екендігін білесіңдер. Бұл мүмкіндік рекурсивті алгоритмдерді бағдарламалық іске асыру әдісі болып табылады. Алайда есепті шығарудың рекурсивті жолын (рекурсивті алгоритм) көру көбіне оңайға соқпайды.



2.30. – сурет. Дисктерді жылжытудыңүш кезеңі.

Бұрын берілген дисктерді жылжытудың алгоритмін сипаттау рекурсивті қасиетке ие болады.

Nдисктерді жылжыту N- 1 дисктерді ауыстыру арқылы бейнеленеді. Осы алгоритмнің рекурсивті сілтемелерінің тізбегінен өзіне өзіқалай шығуға болады? Оның шешімі тривиал мазмұнының сонша лықты оғаш көрінгеніне қарамастан алгоритмнің бірінші тармағында болады.

3-кезең. Бұдан бұрын берілген бағдарламаны ұғыну. Оны компьютерде іске асыру. Нәтижелерін бұрын берілген дистердің ауысу алгоримдерімен салыстырып, №2, 3 мәндер үшін ретке келтіруші есептеулерді орындау.

Енді қарастырылатын есепті шығару алгоритмін Паскальда құрайық. Онда орындалуы тек N= 0 болғанда ғана аяқталатын Hanoi рекурсивті процедурасы бар. Осы бағдарламаға жүгіну кезінде олардың: 1, 2, 3 нөмірлерімен жауапты түрде берілген алаңқайлардың нақты

аттары қолданылады, сондықтан ауысу тізбегі шығар алдында мынадай түрде бейнеленеді:

1 ⇒ 2 1 ⇒ 3 2 ⇒ 3 және т.с.с.

```
Program Monahi;
Var N: Byte;
Procedure Hanoi(N: Byte; A, B, C: Char);
Begin
    If N > 0 Then
    Begin Hanoi(N - 1, A, C, B);
          WriteLn(A, '=>', B);
          Hanoi(N - 1, C, B, A)
    End
End;
Begin
    WriteLn('Дисктердің санын көрсетіңдер: ');
    ReadLn(N);
    Hanoi(N, '1', '2', '3')
End.
```

4-кезең.Бағдарламаға жүріс есептегіштерін, әрбір жүрістің шығуы оның реттік нөмірінен басталатындай етіп қосу. Есептеп шығаруды № 2, 3, 4, 5 және т.с.с. түрлі мәндер үшін орындау. Осы тәжірибеде бұрын берілген формуланың *N* мәндерінен дисктердің ауысу санына тәуелділігін дәлелдеу.

2.16. ЖОБАЛЫҚ ТАПСЫРМА: ІЗДЕСТІРУДІ БАҒДАРЛАМАЛАУ

Тапсырманың мақсаты — Ақпаратты іздестіру проблемасымен байланысты есептермен танысу.

1-кезең. Берілген мәтін бойынша немесе қосымша дереккөздерді қолану арқылы есептің қойылуымен танысу. Ақпаратты іздестіру проблемасымен байланысты кейбір есептерді қарастырайық. Бұл бағдарламалау бойынша классикалық әдебиетте айтарлықтай толық сипатталған есептердің ауқымды классы (мысалы, Н.Вирттің, Д.Кнуттың және т.б. кітаптарын қараңдар).

Іздестіру есептерінің жалпы мағынасы мынаған саяды: ЭЕМ жадында сақталатын ақпараттан белгілі бір талаптарды (шарттарды) қанағаттандыратын тиісті мәліметтерді таңдау.

Осыған ұқсас есептерді біз бұдан бұрын қарастырғанбыз. Мы-

салы: сандық массивте максималды санды іздестіру, деректер файлында қажетті жазбаны іздестіру және т.б. Мұндай іздестіру деректер құрылымының барлық элементтерін асып кетуімен және оларды іздестіру шарттарын сақталуын тексеру арқылы жүзеге асырылады. Құрылымның барлық элементтері байқалатын және *толық асып кету* деп аталатын асып кету іздестірудің *«маңдайлық»* тәсілі болып табылады, жәнекөптеген жағдайда ең үздік тәсілі бла бермейді.

Мысал қарастырайық. *X* бір өлшемді массивте заттық сан өсінде жататын, *N* нүктелерінің координаталары берілген. Нүктелер нөөмірленген. Олардың нөмірлері *X* массивіндегі тізбекке сәйкес келеді. Координаталардың басынан біршама шектетілген бірінші нүктенің нөмірін анықтау қажет.

Бұл бізге белгілі толық асып кету тәсілі арқылы шығарылатын массивтің элементінің модулі бойынша нөмірін анықтау есебі екенін түсіну қиын емес:

```
Const N = 100;
Var X : Array [1..N] Of Real; I, Number : 1..N;
Begin
{---X массивті енгіз---}
.....
Number := 1;
For I := 2 To N Do
    If Abs(X[I]) > Abs(X[Number])
    Then Number := I;
    WriteLn(Number)
End.
```

Мұнда бір өлшемді массив элементтерінің толық асып кетуі бір циклдік құрылымның көмегімен жүргізіледі.

Енді дәл сол бастапқы деректер бойынша ара қашықтығы ең көп болатын екі нүктелер жұбын анықтайық.

Осы есепті асып кету тәсілін қолдана отырып,былайша шығаруға болады: берілген *N* барлық нүктелер жұбын іріктеп, олардың арасынан ара қашықтығы ең көптерінің (координаталар айырмасының ең көп модулін) нөмірлерін анықтау керек. Мұндай толық асып кету екі салынған цикл арқылы жүзеге асырылады:

```
Number1 : = 1;
Number2 : = 2;
    For I : = 1 To N Do
        For J := 1 To N Do
            If Abs(X[I] - X[J] > Abs(X[Number1] -
             X[Number2])
```

```
            Then
            Begin
                    Numberl := I;
                    Number2 := J
End;
```

Алайда, есепті былайша шығару тиімді емес екені көрініп тұр. Мұнда әрбір нүктелер жұбы екі рет көрінетін болады, мысалы, $I = 1$, $J = 2$ және $I = 2$, $J = 1$. $N = 100$ жағдай үшін циклдерді орындау $100 \times 100 = 10\,000$ рет қайталанады.

Бағдарламаны орындау қайтлауларды алып тастағанда тездетіледі. Сонымен қатар I және J мәндерінің сәйкес келулерін де алып тастау қажет. Сонда циклдің қайталану саны $\frac{N(N-1)}{2}$, тең болады, яғни N= 100 болғанда 4 950 қайталану алынады.

Қайталануларды болдымау үшін алдыңғы бағдарламада ішкі циклдің басталуын 1-ден I +1 -ге өзгерту керек. Сонда бағдарлама келесі түрде берілетін болады:

```
Numberl : = l;
Number2 := 2;
For I := 1 To N Do
    For J := I + 1 To N Do
            If Abs(X[I] - X[J]) > Abs(X[Number1] -
                    X[Number2])
            Then
            Begin
                    Number1 : = I;
                    Number2 : = J
            End;
```

Алгоритмнің қарастырылған нұсқасын *қайталамасыз берілген асып кету* деп атаймыз.

Ескерту. Бұл есепті басқа әдіспен де шығаруға болар еді, бірақ берілген жағдайда бізді тек асып кету алгортимі қызықтырды. Түзуде емес, жазықтықта немесе кеңістікте орналасқан нүктелер үшін асып түсу алгоритміне балама іздестіру бірсыпыра қиындықтар тудырады.

Келесі есепте қосындылары онға тең болатын X массивінен қайталаусыз сандардың барлық үштіктерін іріктеп алу керек.

Бұл жағдайда алгоритм айнымалы ұзындығы бар үш салынған циклден тұратын болады:

```
For I := 1 To N Do
    For J := I + 1 To N Do
        For K := J + 1 To N Do
            If X[I] + X[J] + X[K] = 10
            Then WriteLn(X[I], X[J], X[K]);
```

Ал енді *X* массивтен қосындысы онға тең сандардың барлық топтарын таңдау керек деп ойлаңдар. Топтардағы сандардың саны түрлі, яғни 1-ден *N* дейін болуы мүмкін. Бұл жағдайда асып кету нұсқаларының саны күрт артады, ад алгортим болса тривиал емес болып қалады.

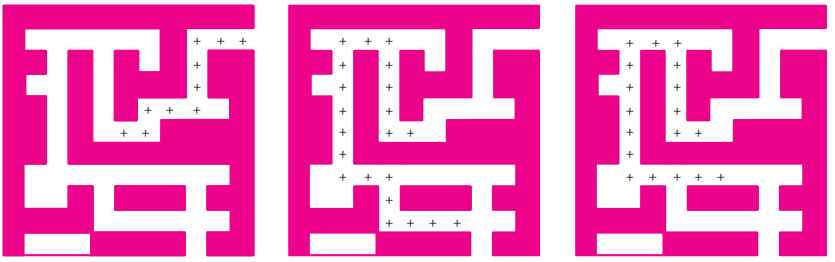
Мұнда тұрған не бар деп ойлайсыз ғой? Машина тез жұмыс істейді! Дегенмен санап көрейік. *N* объектілерден тұратын түрлі топтардың саны (босты да санағанда) 2^N құрады. $N = 100$ болғанда бұлт

2100 ≈ 1030 болады. Сәйкесінше, секундына миллиард амал жылдамдығымен жұмыс істейтін компьютер мұндай асып кетуді шамамен 10 жыл жүзеге асыратын болады. Тіпті ауыстыру қайталауларын алып тастау да мұндай асып кету алгоритмін іс жүзінде жүзеге асыра алмайды.

Мұндай есептерді шешудің жолы есеп шарты тұрғысынан алғанда нәтижесіз болып табылатын нұсқалардың асып кетуден алып тастау тәсілдерін табуда болады. Кейбір есептер үшін мұны әрі қарай сипатталатын алгоритмнің көмегімен іске асыру мүмкін болады.

2-кезең. Лабиринтті өту және қайталамамен берілген асып кету алгоритмі идеясымен.

Лабиринт берілген, оның ішіне кіріп, одан шығар жол іздеу қажет (2.31-сурет). Лабиринттің ішінде тек көлденең және тік бағыттарға ғана жүруге болады.



а *б* *в*

2.31.-сурет. Лабиринтті өту (а, б,в) нұсқалары

2.31-суретте *а... в* лабиринттің орталық нүктесінен шығудың барлық нұсқалары көрсетілген.

Осы есепті шешу алгортимін құру кезінде екі сұрақ туындайды: деректерді қалай ұйымдастыру керек және алгоритмді қалай құру керек?

Лабиринттің формасы туралы ақпаратты өлшемі $N \times N$ болатын таңбалық типтегі LAB квадрат қалыптамада сақтаймыз, мұндағы N — тақ сан (орталық нүкте болу үшін). Әрбір ұяшығында не қабырға, не өткел болатындай лабиринттің пішініне тор жабылады. Қалыптама тордың толтырылуын көрсетеді: өткел элементтеріне аралық, ал қабырға элементтеріне – қандай да бір таңба, мысалы «М» әрпі сәйкес келеді. Лабиринт бойынша жүру жолы «+» таңбасымен белгіленеді. Мысалы, 2.31, *б*–сурет LAB қалыптаманың келесі толтырылуына сәйкес келеді:

М	М	М	М	М	М	М	М	М	М	М
М		+	+	+			М			
М	М	+	М	+	М		М		М	М
М		+	М	+	М	М	М		М	М
М	М	+	М	+	М					М
М	М	+	М	+	+		М	М	М	М
М	М	+	М	М	М	М	М	М	М	М
М	М	+	+	+						М
М			М	+	М	М	М		М	М
М	М	М	М	+	+	+	+	+		М
М				М	М	М	М	+	М	М

Талдау үшін бастапқы деректер - лабиринттің пішіні (айқаршықсыз LAB бастапқы қалыптама); қажетті нәтиже - лабиринттің орталық нүктесінен шығатын мүмкін болатын траекториялар (әрбір жол үшін айқаршықпен белгіленген, траекториясы бар LABқалыптамасы шығарылады).

Қайталанумен берілетін асып кету алгоритмін сондай-ақ *сынама әдісі* деп те атайды, және оның мәнісі келесіде.

1. Траекторияның әрбір кезекті нүктесінен бір тізбек бойынша ғана мүмкін болатын жүру бағыттары қарастырылады. Көру әр жолы сағат тіліне қарсы бағытта жүргізіледі (оңға-жоғарыдан-солға-төменнен); қадам алғашқы табылған бос көрші торға жүргізіледі; қадам жасалған тор айқаршықпен белгіленеді деп келісейік

2. Егер кезекті торда әрі қарай жол болмаса (тығырық), бір қадам артқа жүріп, осы нүктеден мүмкін болатын жүру жолдарын қарастыру керек. Артқа оралғанда, артта қалған тор аралықпен белгіленеді.

3. Егер қадам жсалған кезекті тор лабиринттің шетінде (шыға берісте) болса, табылған жол басып шығаруға жіберіледі.

3-кезең. Рекурсивті процедураның тізбекті тәптіштеу әдісін лабиринтті жүріп өту туралы есепті бағдарламалау үшін қолдануды қарастыру. Алынған бағдарламаны компьютерде іске асыру. Тестік есептеп шығаруды орындау.

Бағдарламаны тізбекті тәптіштеу әдісімен құратын боламыз. Тәптіштеудің бірінші кезеңі:

```
Program Labirint;
Const N= 11; { лабиринттің өлшеміNx Nторлар}
Type Field = Array [1..N, 1..N] Of Char;
Var LAB : Field;
Procedure GO(LAB : Field; X, Y : Integer);
Begin
{Лабиринттің ортасынан шетіне дейінгі жолдарды іздестіру – әрбір табылған жол баспаға шығарылады}
End;
Begin
    {Лабиринтті енгізу}
    GO(LAB, NDiv2 + 1, NDiv2 + 1) {Ортасынан бастаймыз}
End.
```

GO процедурасы (*X, Y*) координаталары бар торға қадам жасауға талпынады. Егер бұл тор лабиринттің шыға берісінде болатын болса, баспаға жүріп өткен жол шығарылады, кері жағдайда – бұрын орнатылған тізбекке сәйкес көрші торға қадам жасалады. Егер тор тығырыққа тірелсе, артқа қадам жасалады. Жоғарыда айтылғандардан процедураның рекурсивті сипатта болатыны белгілі болды.

Алдымен процедураның жалпы сызбасын талдап тексерусіз жазамыз:

```
Procedure GO(LAB : Field; X, Y : Integer);
Begin
If {тор (X, Y) бос}
Then
```

```
Begin
    {Торға қадам жасау (X, Y)}
    If{Лабиринттің шетіне жеттік}
    Then{Табылған жол баспаға жіберіледі }
    Else{Шартталған тізбекте көрші торларға қадам жасау
        әрекеті}
        {Бір қадамға артқа шегіну}
    End
End;
```

Табылған қозғалыс траекториясының шығару үшін PRINTLAB процедурасы құрылады.

Ақырғы көрінісінде бағдарлма мынадай түрде беріледі:

```
Program Labirint;
Const N = 11; {Лабиринттің өлшеміNxN торлар}
Type Field = Array [1..N, 1..N] Of Char;
Var LAB : Field; X, Y : Integer;
Procedure PRINTLAB(LAB : Field);
{Лабиринтте табылған жолды баспаға жіберу}
Var X, Y : Integer;
Begin
    For X := 1 To N Do
    Begin
        For Y : = 1 To N Do
            Write(LAB[X, Y]);
            WriteLn
    End;
    WriteLn
End; {Баспалар}
Procedure GO(LAB : Field; X, Y : Integer);
Begin
    If LAB[X, Y] = ' ' {Егер тор бос болса}
    Then
    Begin
        LAB[X, Y] := ' + '; {Қадам жасалады}
        If (X = 1) Or (X = N) Or (Y = 1) Or (Y = N)
{Шеті}
        Then
```

```
            Else
                PRINTLAB(LAB) {табылған жол басылуда}
            Begin {Келесі қадамды іздестіру}
                GO(LAB, X + 1, Y);
                GO(LAB, X, Y + 1);
                GO(LAB, X - 1,Y);
                GO(LAB, X, Y - 1)
            End;
        LAB[X, Y] := ' ' {Артқа қарай оралу}
    End
End; { GO процедуралары}
Begin{Бағдарламаның негізгісі}
     {Лабиринтті енгізу}
    For X := 1 To N Do
    Begin
    For Y : = 1 To N Do
        Read(LAB[X, Y]);
        ReadLn
    End;
    GO(LAB,NDiv 2 + 1, NDiv 2 + 1)(Ортасынан бастаймыз}
End.
```

Бұл процедураны рекурсивті анықтауды қолданумен берілген әсем бағдарламасының тағы бір мысалы.

Берілген алгоритмнің схемасы қайталаумен берілген асып кету әдісіне тән. Осыған ұқсас алгоритмдер бойынша мысалы, фигуралармен шахмат тақтасын айналып өту туралы немесе тақтада фигураларды «бірін бірі» ұрмайтындай етіп орналастыру туралы, сондай -ақ оңтайлы таудаумен берілген көптеген есептер шығарылады (коммивояжер туралы, оңтайлы құрылыс туралы және т.б.).

Ескерту. LAB массивін қолдануға байланысты параметр-мән ретінде GO процедурасында ЭЕМ-да бағдарламаны атқару кезінде жадымен байланысты қиындықтар туындауы мүмкін. Бұл жағдайда массивті ғаламды түрде беруге ауысу керек.

4-кезең. Бұрынғы берілгендерден айырмашылығы бар, лабмиринттің өздік кескіндемесін ойлап табу. Лабиринттің кескіндемесін таңбалық қалыптама түрінде кодтау. Өздеріңнің құрған лабиринттеріңді жүріп өту бағдарламасы бойынша есептеуді орындау.

БАҚЫЛАУ СҰРАҚТАР ЖӘНЕ ТАПСЫРМАЛАР

1. Бағдарламалау деген не?
2. Алғашқы ЭЕМ-да бағдарламалар қандай түрде құрылды?
3. Автокодтар және Ассемблер тілдері неліктен бағдарламалаудың машинаға бағдарланған тілдері деп аталады?
4. Құрылу хронологиялық тізбегі бойынша бағдарламалаудың негізгі поцедуралық тілдерін атаңдар.
5. Бағдарламалаудың парадигмасы деген не?
6. Бағдарламалаудың негізгі парадигмалары қандай және олардың айырмашылықтары?
7. Құрылымдық бағдарламалау, визуалды бағдарламалау деген не?
8. Компиляция мен интерпретацияның айырмашылығы неде?
9. Бағдарламаны ретке келтіру деген не? Ретке келтіру қандай жүйелік құралдардың көмегімен жүргізіледі?
10. Әзірлеудің біріктірілген ортасына қалай құрамалар кіреді?
11. NET Framework платформасы бағдарламашыға қандай мүмкіндіктер береді?

ОБЪЕКТІЛІ-ВИЗУАЛДЫ БАҒДАРЛАМАЛАУ

3.1. ОБЪЕКТІЛІ – БАҒЫТТАЛҒАН БАҒДАРЛАМАЛАУДЫҢ НЕГІЗГІ ҰҒЫМДАРЫ

Бұл тармақшада сендер бағдарламалаудың объектілі бағдарланған парадигмасымен – ОБП танысасыңдар. 2.1- тармақшасында айтып өтілгендей, Паскаль тілі бастапқыда құрылымдық әдістемеге сүйеніп, тек процедуралық бағдарламалауға ғана арналған еді. Содан соң ТурбоПаскаль тілінің соңғы нұсқаларында ОБП элементтері пайда бола бастады. Бұл үрдіс Object Pascal атауына ие болған, Паскальдың объектілі-бағдарланған нұсқасын құруға әкелді. Object Pascal бағдарламалау тілінің бағдарламалаудың визуалды технологиясымен және визуалды құрамалар кітапханаларымен бірігу нәтижесінде Delphi бағдарламалау жүйесі пайда болды. Бұдан әрі Delphi негізі болып табылатын объектілік модель қарастырылады.

Класстар, объектілер, инкапсуляция. ОБП негізгі ұғымдары ретінде «класс» және «объект» болады. *Класс* — бұл бірдей қасиеттер жиынтығына және оларға әсер ету тәсілдеріне ие болатын, бағдарламашы сипаттайтын деректер типі. *Объект* — бұл қасиеттерінің нақты мәндерімен берілген белгілі класстың данасы.

Класс Object Pascal-да элементтері өрістер мен тәсілдер болатын, деректердің құрылымдық типіне ие болады. «Класс» типін сипаттау пішіні келесідей болады:

```
Type
    класстың атауы = class
    өрістерді сипаттау классы
    әдістерді хабарлау және қасиеттерді сипаттау
End;
```

Өрістер — бұл айнымалы шамалар (қарапайым немесе құрылымдандырылған), **әдістер** — осы класстың өрістерімен жұмыс жасайтын процедуралар мен функциялар.

Деректерді (өрістерді) бірыңғай құрылымға біріктіру және осы

деректермен (әдістермен) әрекет ету үрдісі ОБП-де *инкапсуляция* деп аталады. Инкапсуляция өріс мәндерін берілген класстан басқа, өзге тәсілдермен өзгерту мүмкіндігіне жол бермейді. Инкапсуляция — ОБП бірінші базалық қағидасы.

«Класс» деректер типі қолданылатын, түзу кесіндінің ұзындығын Object Pascal тілінде есептеу бағдарламасына берілген мысалды қарастырайық:

```
Program Geometry;
{-----Классты сипаттауTLine----------}
Type TLine = class
     FX1,FY1,FX2,FY2: real; // Өрістер— сызық ұштарының
     координаталары
     Function Length: real; // Әдіс— ұзындықты есептеу
     функциясы
     Procedure Input; // Әдіс— координаталарды енгізу
     процедурасы
End;
{---------Tline типті объектіні сипаттау----------}
Var Line: TLine;
{---------Әдістерді іске асыру----------------------}
Function Line.Length: real;
Begin
     Line.Length:=sqrt(sqr(Fx1-Fx2)+sqr(Fy1-Fy2))
End;
Procedure Line. Input;
Begin Line. Input;
Write('x1=' ); readln(Fxl);
Write('y1=' ); readln(Fyl);
Write('x2=' ); readln(Fx2);
Write('y2=' ); readln(Fy2)
End;
{----------Негізгібағдарлама!--------------------}
Begin
     Line:= TLine.Create; //динамикалық жадыда объектіні құру
     Line.Input;          //деректерді енгізу
     //Кесіндінің ұзындығын есептеу және енгізу
     Writeln('Кесіндінің ұзындығы=', Line. Length)
     Line.Destroy // динамикалық жадыдан объектіні өшіру
End.
```

Классты сипаттауда әдістер оларды іске асырушы процедуралар мен функциялардың тақырыптарын хабарлау түрінде берілген.Бұл процедуралар мен функцияларды толығырақ сипаттау төменде ішкі бағдарламалар бөлімінде берілген. Length функциялары мен Input процедураларын сипаттауда параметрлер көрсетілмейтіндігіне назар аударыңдар, өйткені олар үшін қол жетімді шамалар класс өрістері болады.

Конструктор (Create)— компьютердің динамикалық жадында объектінің деректеріне орын қалдыруға арналған стандартты функция. *Деструктор* (Destroy)динамикалық жадыны объектіден босататын стандартты процедура. *Динамикалық жадыдепжедел* жадының бөлімі аталады, ол жерде шамаларға орын бағдарламаны атқару кезінде (компиляция кезінде таратылатын статикалық жадыға қарағанда) қалдырылады.

Берілген класс объектілерінің өрістері мен әдістеріне жүгіну құрама атаудың көмегімен, жазбалардағыдай жүргізіледі:

<объектінің атауы>.<өрістің атауы>, <объектінің атауы>.<әдістің атауы>

Тұқым қуушылық және полиморфизм. Келесі бағдарламаның да геометриялық мазмұны бар. Онда екі класс хабарланған: дөңес төртбұрыштар классы (TFourAngl) және квадраттар классы (TKvadrat). Төртбұрыш — квадратқа қарағанда жалпылама ұғым. Квадрат төртбұрыштың жеке нұсқасы болып табылады. Кез келген төртбұрышқа тән ортақ қасиет — төрт төбесінің болуы, сондықтан TFourAngl класстарының өрістері ретінде төрт төбенің координатасы болады. Бұдан басқа, төртбұрыштың өрістерінің санына оның төрт бұрышының және екі диагоналының ұзындығын қосамыз. TFourAngl классының әдістері ішінде төбе координаталарын енгізуді, кесінді ұзындықтарын есептеуді (бұрыштары мен диагональдарының) және кез келген дөңес төртбұрыштың ауданын есептеуді хабарлаймыз.

TKvadrat классы TFourAngl классының ұрпағы ретінде хабарланады. Аталық класстың атауы ұрпақ-классытң хабарламасында class сөзінен соң дөңгелек жақшаның ішінде көрсетіледі. Ұрпақ-класс аталық-класстан оның барлық элементтерін, яғни өрістері мен әдістерін алады. Сондықтан ұрпақ-класстың хабарламасында оларды қайталаудың қажеті жоқ. Алайда, TfourAngl классында кез келген төртбұрыштың ауданын есептеу әдісі – Square функциясы болғанымен, Tkvadrat классында Square есіммен берілген функция квадрат үшін

ауданды есептеудің (қабырғасының ұзындығының квадраты ретінде) қарапайымдау әдісін қолдану мақсатында хабарланады, бұл есептеудің дәлдігін арттырады және машинаның уақытын қысқартады.

Бағдарламаның толық мәтінін қарастырайық:

```
    Program Geometry;
Type TKoord= record X,Y: real end; //Төбе координаталарының типі
{-----Негізгі классты хабарлау (атаны)----------}
Type TFourAngl=class
    P: array[0..3] of TKoord; //4-төбенің координаталары
    L: array[0..5] of real; //Қабырға мен диагональ ұзындықтары
    Procedure Init; //Әдісі: координаталарды енгізу
    Procedure Line; //Әдісі: кесінді ұзындықтарын есептеу
    Function Square: rea1;//Әдісі: ауданды есептеу
end;
{-----Туылған классты хабарлау (ұрпақты)-----}
Type TKvadrat=class(TFourAngl)
    Function Square: real; //Әдісі: квадраттың ауданы
end;
{-----ТөртбұрышжәнеКвадратобъектілерінсипаттау------}
Var FourAngl: TFourAngl;
    Kvadrat: TKvadrat;
{-----Әдістерді іске асыру процедураларын сипаттау--------}
ProcedureTFourAngl.Init;
Var i: integer;
begin
    for i:= 0 to 3 do
    begin
        write('Input X',i,':'); Readln(P[i]. X);
        write('Input Y',i,':'); Readln(P[i]. Y)
    end
end;
Procedure TFourAngl.Line;
Var i: integer;
begin
    for i:= 0 to 3 do
        L[i]:= sqrt(sqr(P[i].X-P[(i+1) mod 4].X)+ sqr(P[i].
        Y-P[(i+1) mod 4].Y));
        L[4]:= sqrt(sqr(P[0].X-P[2].X)+ sqr(P[0].Y- P[2].Y));
        L[5]:= sqrt(sqr(P[1]. X-P[3]. X)+ sqr(P[1].Y- P[3].Y))
end;
    Function TFourAngl.Square: real;
```

```
    Var pp1,pp2: real;
    begin
            Line;
            pp1:=(L[0]+L[1]+L[4])/2;
            pp2:=(L[2]+L[3]+L[4])/2;
            Square:= sqrt(ppl*(pp1-L[0])*(pp1-L[1])*(pp1-L[4]))+
            sqrt(pp2*(pp2-L[2])*(pp2-L[3])*(pp2-L[4]))
    end;
    Function TKvadrat.Square: real;
    begin
            Line;
            Square:= sqr(FourAngl.L[0])
    end;
{--------------------Негізгі бағдарлама--------------------}
    begin
            FourAngl:= TFourAngl.Create;
            {Объектіні құру FourAngl}
            Kvadrat:= TKvadrat.Create;
            {Объектіні құру Kvad rat}
            FourAngl.Init;{Төбе координаталарын енгізу}
            FourAngl.Line; {Қабырғалар мен
            диагональдардың ұзындықтарын есептеу}
            with FourAngl do
    begin
{--------Квадратты тану және ауданды есептеу----------------}
    if (L[0]=L[1]) and (L[1]=L[2]) and (L[4]=L[5]) then
    Writeln(Бұл квадрат, площадь= ', Kvadrat. Square)
    else
    Writeln(Бұл квадрат емес, аудан= ', Square)
    end
end.
```

Мәтінге кіріктірілген түсініктемелер бағдарламаның жекелеген фрагменттерін түсінуге мүмкіндік береді. Бұл бағдарламада тұқым *қуалаушылық* деп аталатын ОБП екінші базалық үрдісі қолданылған (инкапсуляциядан басқа). Тұқым қуалаушылық – бұл ұрпақ-класс өзінің аталық классынан барлық өрістер мен әдістерді мұраға алатын үрдіс. Тұқым қуалаушылық бұрын болған ұрпақтар ретінде жаңа класстарды құруға мүмкіндік береді.

Ұрпақ-класс өзінің аталық классынан барлық әдістер мен өрістерді мұраға алады. Сонымен бірге аталық класс та өз аталарынан кейбір элементтерді мұраға алған, және олар ұрпақтарына беріледі.

ОБП технологиясы тұқым қуалаушылықтың тармақталған сатыларын құрастыруға мүмкіндік береді, яғни бір ұрпақ бірнеше ата-анаға ие бола алады. Біздің келтірілген мысалда қарапайым тұқым қуалаушылық қолданлады: бір ата-ана – бір ұрпақ, бұл Object Pascal әрекет ететін шектеуге сәйкес келеді.

ОБП үшінші негізгі үрдісі – *полиморфизм* – әртүрлілік деп аталады. Бірдей интерфейстері бар класстардың элементтері түрлі іске асыруларға ие болады. Қарастырылған бағдарламада полиморфизм Square бірдей атауы бар функция екі ретінде екі түрде байқалу арқылы көрінеді: алдымен TfourAng аталық классында, содан соң ол Tkvadrat ұрпақ-классына *қайта анықталды*. Нәтижесінде кез келген төртбұрыштың ауданы және квадраттың ауданы әр түрлі алгоритмдер бойынша есептелетін болады.

Жаттығулар

1. ObjectPascal –да «класс» деген не» және «объект» деген не?
2. Инкапсуляция, тұқым қуалаушылық, полиморфизм ұғымдарына анықтама беріңдер.
3. Компьютерде (ObjectPascal бағдарламалау жүйесінде) Geometry бағдарламасын іске асыру.
4. TfourAngl классының сипаттамасына төртбұрыш бұрыштарының мәндері үшін өрістерді және бұрышты есептеу әдісін қосу. Бағдарламаның негізгі бөлігінде барлық бұрыштарды есептеуді іске асыру.
5. Жазық геометриялық фигураларды: шеңберді, шаршыны, тік бұрышты суреттеу үшін класстар жүйесін құру. Объектілерді құруға арналған әдістерді, жазықтықта қозғалуды есепке алуды, өлшемдердің өзгеруін және берілген бұрышта айналуды қарастыру. Тестілеу бағдарламасын іске асыру.
6. Ұымның пошталық мекен-жайы туралы ақпаратты сақтайтын класстың сипаттауын құру. Мекен-жайдың құрамдас бөліктерін жеке өзгерту, осы класстың объектілерін құру және өшіру мүмкіндіктерін қарастыру. Тестілеу бағдарламасын іске асыру.

3.2. DELPHI БАҒДАРЛАМАЛАУ ЖҮЙЕСІ

Delphi — бағдарламалаудың визуалды технологиясын қолдану жолымен объектілі-бағдарланған Windows қосымшаларын құруға арналған бағдарламалау жүйесі. Delphi визуалды ортасы RAD категориясының ортасына (Rapid Application Development — қосымшаларды тез әзірлеу ортасы) жатады. Сондай-ақ Delphi терминімен Паскальдың заманауи диалекті болатын, және ол да Delphi Pascal деп аталатын, бағдарламалау тілін атауға келісейік.

Мұндай орталарда бағдарламалық қамсыздандыруды әзірлеудегі негізгі тәсілдеме стандартты визуалды құрамаларды – бағдарламашы өзінің жобасына орналастыру арқылы, оларды бағдарламаға қосатын, алдын ала дайындалған класстарды қолдану блып табылады.

Delphi–де бағдарлама әзірлеу үрдісі біршама қарапайымдандырылған болуы мүмкін. Ең алдымен бұл әдетте бағдарлама әзірлеуде шамамен 80 % уақыткететін интерфейс құруға қатысты. Бағдарламашыа тек тиісті компоненттерді терезенің үстіне қою (Delphi-де ол *форма* деп аталады) және олардың қасиеттерін арнайы аспатың көмегімен (Object Inspector) реттемелеу жеткілікті. Оның көмегімен осы компоненттердің оқиғаларын *байланыстыруға* (батырманы басу, меңзердің көмегімен тізімде элементті таңдау және т.б), бағдарламашы өзі құрастыратын оны өңдеу процедурасын жүргізуге болады, — және қарапайым қосымша дайын болады. Бұл ретте бағдарламашының қарамағында ретке келтірудің қажетті құралдары, ыңғайлы мәнмәтіндік анықтама жүйесі, жобамен ұжымдық жұмыс жасауға арналған құралдар және т.с.с. бар.

Delphi бағдарламалау жүйесінің ортасы. Delphi бағдарламалау жүйесінің ортасы 3.1.-суретте берілген. Ол келесі элементтерден тұрады:

1. *Тақырып жолы* (терезенің жоғарғы жағында).

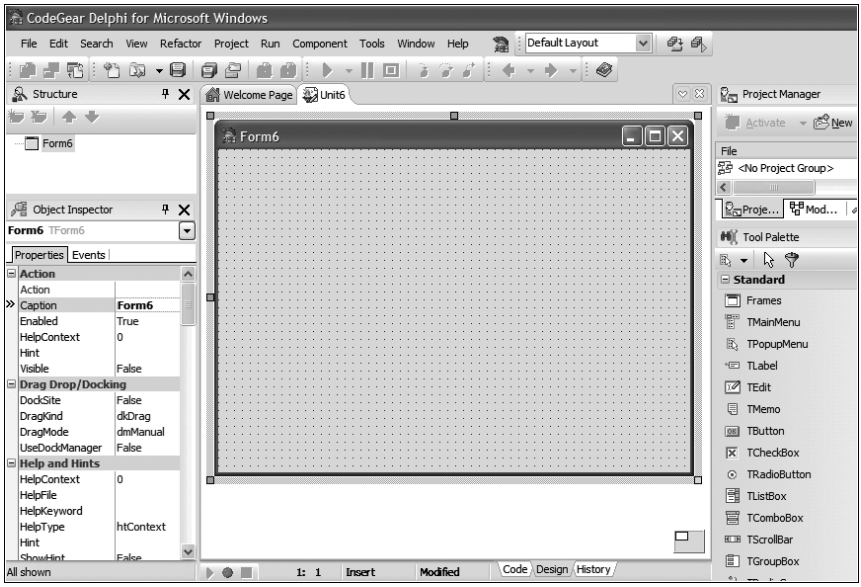
2. *Бас мәзір жолы* және командалық батырмалар (тақырып жолының астында).

3. *Формалар конструкторының терезесі*. Экранның ортасында Design қосымша бетте орналасады(3.1-сур.қараңыз). Форма басқару элементтерін (интерфейс элементтерін) орналастыру жолымен жобалатын қосымшаның интерфейсін құрастыру үшін қолданылады.

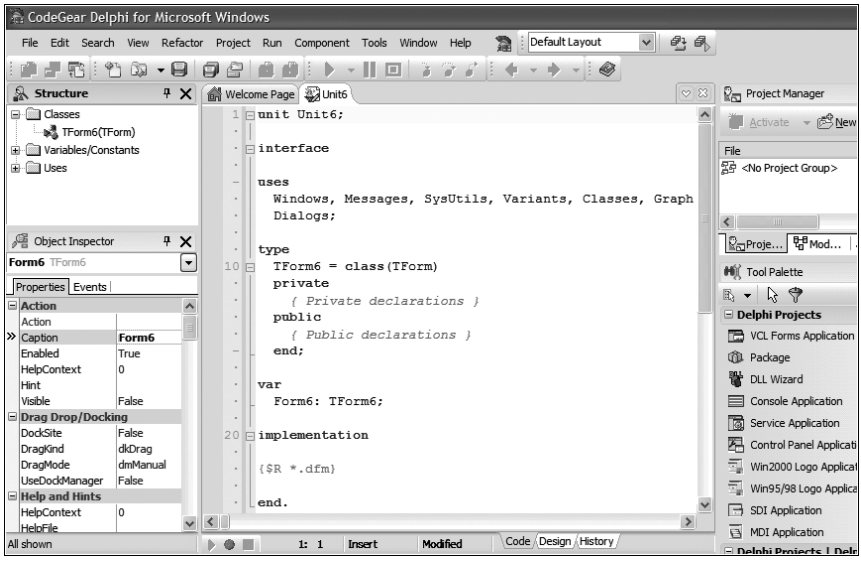
4. *Бағдарламалау кодының терезесі*. Экранның ортасында Codeқосымша бетте орналасады. Бағдарламаны құрудың басында бағдарламалау кодының терезесінде стандартты қалып орналасады (3.2-сурет). Бұдан әрі бағдарламалау үрдісінде ол бағдарлама мәтінімен толтырылады.

5. *Басқару элементтерінің терезесі (интерфейс компоненті)*. Оң жақта орналасады, ToolPalette деп аталған.Топтарға бөлінген басқару элементтерінің пиктограммаларынан тұрады. Пиктограммалардың стандартты тобы Standard, қосымша тобы — Additional, және т.б. деп аталады. Басқару элементтері тінтуірдің көмегімен жобаланатын қосымшаның формасына ауысады.

6. Объектілер инспекторы терезесі.Экранның сол жақ төменгі бөлігінде ObjectInspector атауымен орналасады Терезенің жоғарғы



3.1.-сурет. Форма Конструкторымен Delphi бағдарламалау ортасы



3.2.–сурет. Бағдарламалық код терезесімен Delphi бағдарламалау ортасы

жағында объектілер атауларымен түсіп қалатын тізім берілген (3.1-суретте *Form 6* атауымен жалғыз объект бар). Оның астында *Properties (Қаситтері)* қосымша беттеөрістердің тізімі — қолмен немесе бағдарламалық жолмен өзгертілетін, әдепкі қалпы бойынша қабылданған, олардың атауларымен және мәндерімен берілген объектінің қасиеттері орналасады. *Events (Оқиғалар)* қосымша беттетаңдалған объектіге қатысты оқиғалар тізімі бар. Әрбір оқиғаның аты және әдіске берген нұсқауы – осы оқиға туындаған сәттеорындалатын оқиғаны өңдеу бағдарламасы болады.

7. *Жобалар менеджерінің терезесі.* Экранның жоғарғы оң жақ бұрышында *Project Manager* атауымен орналасады. Ағымдағы жобадан тұратын компоненттердің файлдық ағашын бейнелейді (әрі қарай қараңдар). Тінтуірді шерту арқылы бір компоненттен екіншісіне көшуге мүмкіндік береді.

8. *Объектілер ағашы терезесі*. Экранның сол жақ жоғарғы бұрышында *Structure* атауымен орналасады.Құрылатын жобада графикалық интерфейс объектілерінің құрамын бейнелеп көрсетеді. Объектілер арасында тінтуірдің шертуімен ауысуға мүмкіндік береді.

Жоба. Форма. Delphi-де *жоба (Project)* деп атқарылатын бағдарлама құрылатын модульдер мен ресурстердің бүкіл кешені аталады. Жобаға бағдарламалық модульдер, экрандық формаларды, бағдарлама құрудың жалпы параметрлерін сипаттау және т.б. кіреді.

Форма— жұмыс терезелерін құруға арналған Delphi негізгі графикалық объекті. Форма дәстүрлі қосымша «терезелерінің»барлық нышандарына: белгі, тақырып, *Бүктеу, Ашу, Жабу* батырмалары, өлшемдік жиектемеге ие болады және тінтуірмен басқарылады (3.1-суретті қараңдар). Пайдаланушымен интербелсенді өзара әрекет етуді қарастыратын бағдарламада бағдарламаның негізгі терезесін сипаттайтын бас форма тағайындалады.

Форма өзінежобаның графикалық интрфейсінің барлық басқа компоненттері: батырмаларды, таңбаларды, енгізу терезелерін және т.с.с. кіргізетін өзіндік бір контейнер болып табылады. *Формалар конструкторы* жобаны әзірлеу кезіндекелесі әрекеттерді жасауға мүмкіндік береді:

- Компоненттерді формаға қосуға;
- Форманы және оның компоненттерін түрлендіруге;
- Компонент оқиғаларын код редакторында өңделінуге болатын процедурамен немесе функциямен байланыстыруға;

Жалпы алғанда Delphi-де бағдарлама әзірлеу үрдісі мынадай түр-

де беріледі: басқару элементтері терезсінен тінтуірдің көмегімен интерфейс компоненттері таңдалады (батырма, жазу, мәтінді түзету және т.б.), формаға орналастырылады және *Қасиеттері* саласында олардың қасиеттерінің мәндері беріледі.

Delphi ортасы форманың мазмұнын талдайды, формамен байланысты тиісті бағдарламалық модульді құрады (Unit), ал бағдарламашы оған процедуралардың бағдарламалық кодтарын – *оқиғаларды өңдеушілерді* енгізеді.

Жоба бірнеше формаларды және, сәйкесінше, бірнеше бағдарламалық модульдерді ұстап тұра алады. Формалардың бағдарламалық модульдерінен басқа, жобаға кейбір өзге бағдарламалық модульдермен берілген файлдар креді.

Басқару элементтері. Қасиеттері. Басқару элементтері — графикалық интерфейсінің компоненттері боып табылатын, олардың сыртқы көрінісі мен күйін анықтайтын, сондай-ақ пайдаланушы немесе бағдарлама жүргізетін оқиғаларға әрекеттесетін объектілер классы.

Басқару элементтерінің және олардың қасиеттерінің мысалдары:

TLabel — таңба. Экранда мәтінді бейнелеп көрсету қызметін атқарады. Қасиеттері қатарында: жазу (*Caption*) , аты (*Name*), шрифт параметрлері (*Font*), түс (*Color*), экрандағы өлшем, экранда орналасу координаталары және т.б.;

TEdit — түзетудің экрандық өрісі. Пайдаланушының бағдарламаны орындау барысында деректерді енгізу үшін қызмет етеді. Негізгі қасиеттері: аты (*Name*), енгізу/түзету өрісіндегі мәтін (*Text*), шрифт (*Font*), өлшемдер және т.б.;

TButton — командалық батырма. Бағдарламаны орындау кезінде батырманы басқанда қандай да бір әрекеттерді орындауға мүмкіндік береді. Негізгі қасиеттері: жазу (*Caption*), аты (*Name*), өлшемдері және экранда орналасуы.

Басқару элементтері. Оқиғалар. *Оқиға*— Пайдаланушының немесе бағдарламалық әсер етудің нәтижесінде объектінің қандай да бір үйінің өзгеруі. Әрбір объектілер классы үшін *Object Inspector* терезесінде *Events* қосымша беттебейнеленген оқиғалардың өз жиынтығы бар. Пайдаланушының әрекеттерімен іске асырылатын оқиғалар мысалы ретінде болады:

- *OnClick*— объектіде тінтуір көрсеткішімен шерту;
- *OnDblClick*—объектіде тінтуір көрсеткішімен екі рет шерту;

- *OnMouseMove*— объекті үстінде тінтуір көрсеткішімен жүру;
- *OnChange*— мазмұнды өзгерту (мысалы, түзету терезесіндегі мәтінді).

Әдістер — оқиғаларды өңдеу процедуралары. Delphi– оқиғаға өңдеушіге - оқиғаға процедураны шақыру арқылы жауап береді. Мұндай процедураны бағдарламаша өзі құрады. *Events* қосымша бетте оқиғаны таңдап,өрісте оқиға атына қарсы жерде екі рет шерту қажет.

Бағдарламалық код терезесінде процедураның дайындамасы пайда болады, оған осы оқиғаға жауап ретіндегі бағдарламалық кодты жазу керек.

Delphi жүйесінде құрылатын бағдарламалар *оқиғалық-басқарылатын бағдарламалар* деп аталады. Бұл бағдарламамен шығарылатын түрлі есептерді орындау белгілі бір оқиғалармен бастамашылық етеді дегенді білдіреді.

3.3. DELPHI БАҒДАРЛАМАЛАУ КЕЗЕҢДЕРІ

Delphi бағдарламалау жүйесінің көмегімен қосымшалардың бірнеше түрлі типтерін құруға болады. Соның ішінде *консолды қосымша* деп аталатын графикалық интерфейссіз берілген қосымшаны іске асыруға болады. Мұндай қосымшаларды сендер мысалы, ТурбоПаскаль немесе PascalABC жүйелерінде бұрын құрғансыңдар.

Консолды қосымша құру. Бағдарламалау үшін алғашқы мысал ретінде тұтас оңдық емес санды есептеп шығарудың оңдық жүйесіне ауыстыруға берілген есепті алайық. Бұдан әрі берілген бағдарламада келесі айнымалылар қолданылады:

p — есептеп шығарудың негізгі жүйелері (1 <p < 10);

Np —тұтаср-дық сан — бастапқы дерек;

N10 — оңдық сан - нәтиже.

Консолды қосымшаны ортада құруды қарастырайық. Ол келесі тәртіпте жасалады: после запуска системы Delphi жүйесін іске қосқаннан кейін бас мәзір арқылы жаңа жобаны: *File— New* құруға команда беріледі. Содан соң ашылған терезеде *Console Application* қосымшаның типін таңдау керек.

Дайын формамен берілген бағдарламалық код редакторының терезесінде бағдарлама мәтінін енгізу керек. Бағдарлама мынадай түрде беріледі:

```
program Projectl;
{$APPTYPE CONSOLE} {қосымша типі туралы
```

```
        компиляторға директива}
        uses SysUtils;{Жүйелік кітапхананы қосу}
        var     N10,Np,k: longint;
                p: 2..9;
        begin
        write('p='); readln(p);{Есептеп шығару жүйесінің
        негіздемесін енгізу}
        write('N',p,'='); readln(Np);{бастапқы р-дық санды
        енгізу}
        k:=1; N10:=0;
        while(Np<>0)do{ Np нөлге тең болғанға дейін цикл орында-
лады}
        beginN10:= N10+(Npmod10)*к;{Ашылған форманы қосу}
                k:= k*p{Базисті есептеу: p, p2, p3...}
                Np:=Npdiv 10 {Кіші цифрді алшақтату}
        end;
        writeln('N10=',N10); {ондық санды шығару}
        readln; {экранда нәтиже терезесін кешіктіру}
    end.
```

Бағдарламаны орындау бас мәзір арқылы *Run— Run* командасы бойыншажүзеге асырылады.

Бағдарламаны орнындаудың мысалы: берілген бағдарлама бойынша 1101_2 екілік санды ондық жүйеге асытырғанда экранда келесіні көреміз:

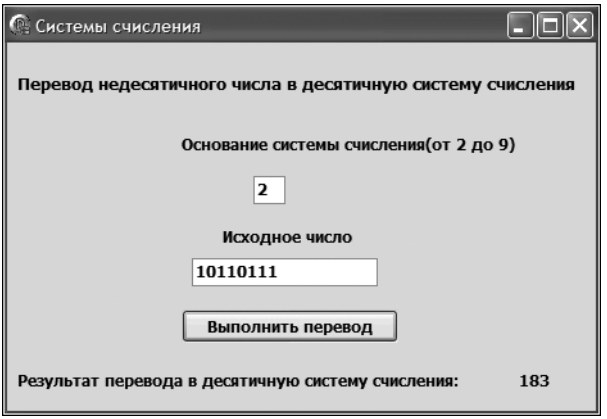
```
P=2
N2=1101
N10=13.
```

Сәйкесінше, нәтижесінде пайда болады: $1101_2 = 13_{10}$.

Терезелік қосымша құру. Енді сандарды ауыстыру туралы есепті шығаруды графикалық интерфейсі бар терезелік жобаның көмегімен іске асырамыз. Ол үшін, *File— New* командасынан кейін *VCL Forms Application* қосымшаның типін таңдаймыз.

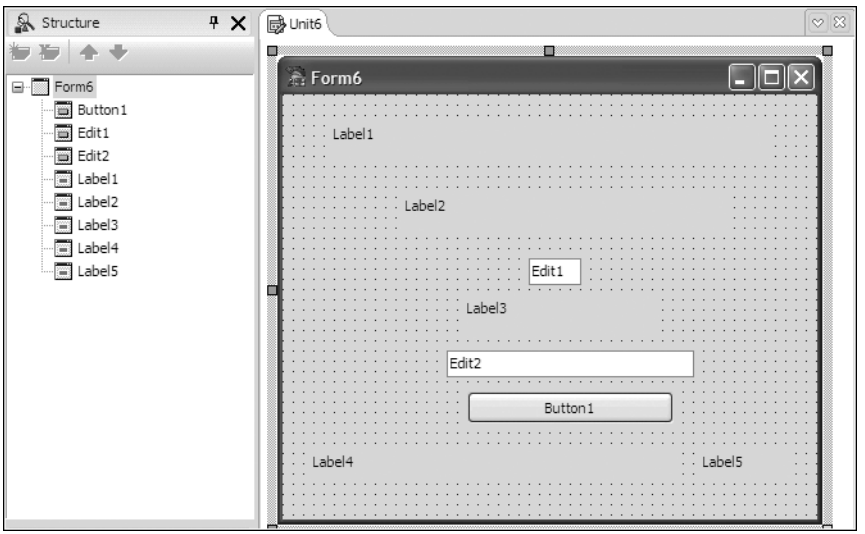
1. *Интерфейсті жобалау және құрастыру.* Жұмыстың басынан бастап соңғы нәтижені елестету керек: бағдарламаны орындау нәтижесін біз қандай түрде көргіміз келеді. Сандар туралы есеп үшін мұндай елестету 3.3-суретте көрсетілген.

Форма конструкторын және басқару элементтерінің панелін қолдана отырып, 3.4-суретте көрсетілгендей форма сұлбасына интерфейстің қажетті элементтерін саламыз. Бұл жерде Таңба (*Label*) типіниңбес элементі, екі редакциялау терезесі және бір командалық батырма болады.Редакциялау терезесінде бастапқы деректер: есептеп шығару жүйесінің негіздемесі және осы жүйедегі сан енгізіледі. Нәтижесі *Label5* өрісіне шығарылады.

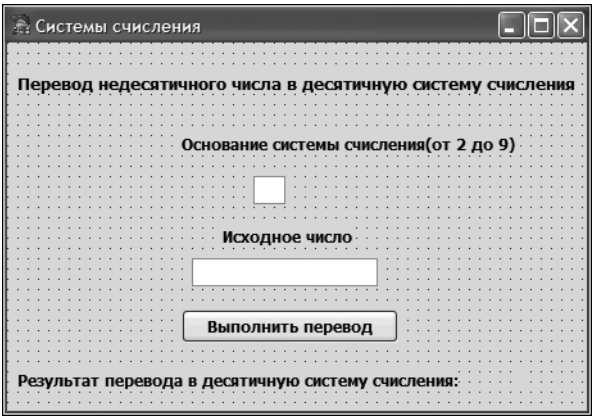


3.3.-сурет. Сандар туралы есептің интерфейсі

Содан соң форма объекті үшін Есептеп шығару жүйесінде жазу (Caption қасиеті)өзгертіледі. 1-ден 4-ге дейін таңбалар үшін тиісті таңбалар тағайындалады. *Label5* жазу өшіріледі. Командалық батырма үшін Ауыстыруды орындау жазуы тағайындалады. Интерфейс макетінің ақырғы түрі 3.5-суретте көрсетілген.



3.4. –сурет. Интерфейстің құрылымы



3.5.-сурет. Бағдарлама интерфейсінің макеті

2. *Оқиғаларды өңдеуді іске асыру.* Санды есептеудің ондық жүйесіне ауыстыруды орындау *Ауыстыруды орындау* батырмасында шерту бойынша жүргізілуі тиіс. OnClick *Button1* объектісі үшін OnClickоқиғаны өңдеу процедурасын шақыруға бастамашылық етеміз.Бағдарламалық код редакторы терезесінде тақырыппен берілген қалып пайда болады:

```
procedure TForm6.Button1Click(Sender: TObject);
```

Мұнда жақша ішіндеүрпақтары қалған барлық класстар болатын, Delphi негізгі классы - TObject классы көрсетіледі. Бұл процедураның денесіне ондық емес санды есептеудің ондық жүйесіне ауыстыру бағдарламасын жазамыз. Нәтижесінде құрылған формамен байланысты барлық бағдарламалық модуль (Unit), келесі түрде беріледі:

```
unit Unit6;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms, Dialogs, StdCtrls;
type
    TForm6 = class(TForm)
            Label1: TLabel;
            Label2: TLabel;
            Editl: TEdit;
            Label3: TLabel;
            Edit2: TEdit;
            Label4: TLabel;
```

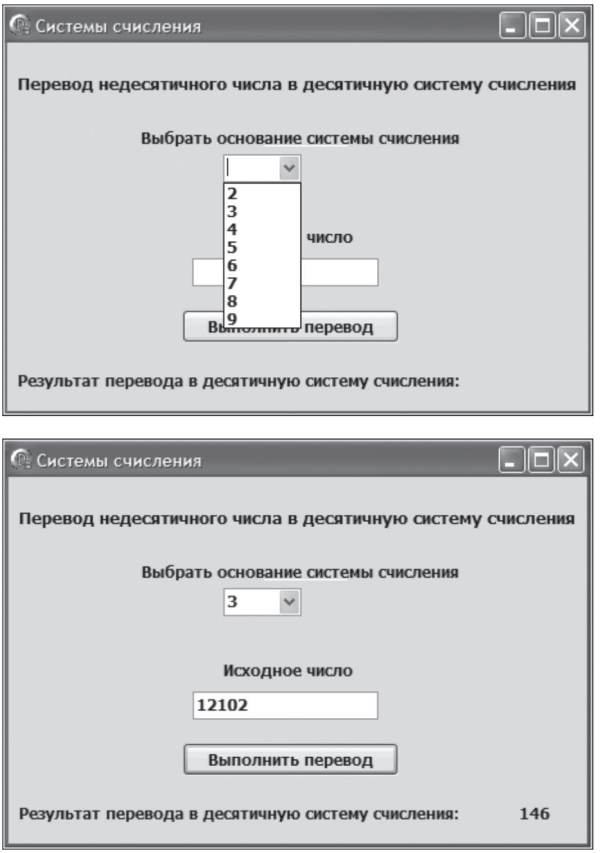
```
            Label5: TLabel;
            Buttonl: TButton;
        procedure Button1Click(Sender: TObject);
        private
            { Private declarations }
        public
            { Public declarations }
        end;
var
    Form6: TForm6;
implementation
{$R *.dfm}
{-------Оқиғаны өңдеу процедурасы----------}
procedure TForm6.Button1Click(Sender: TObject);
var N10,Np,k: longint;
p: 2..9;
begin p:= StrToInt(Edit1. Text); //Editl терезесінен енгізу
    Np:=StrToInt(Edit2. Text); //Edit2терезесінен енгізу
    k:= 1; N10:= 0;
    while (Np<>0) do
    begin N10:= N10+(Np mod 10)*k; k:= k*p;
        Np:= Np div 10
        end;
        Label5.Caption:= IntToStr(N10 //таңба өрісіне
        шығаруLabel5
    end; // Оқиғаны өңдеу процедурасының соңы
end.
```

Бағдарламаны орындау бас мәзір арқылы *Run— Run* командасы бойыншажүзеге асырылады. Экранда бағдарлама интерфейсінің терезесі пайда болады. Редакциялау терезесіне бастапқы деректер енгізіледі: есептеу жүйесінің негіздемесі және ауыстырылатын сан. Содан соң командалық батырмаға шертіп, *Label5* таңба өрісінде нәтижені аламыз (3.3-суретті қараңыз).

Бастапқы деректерді енгізу Edit интерфейс элементінің *Text* өрісінің көмегімен жүргізіледі. Пернетақтадан санды редакциялау терезесіне енгізгенде, ол таңбалық жол ретінде қабылданады. Егер тұтас сан енгізілсе, онда жол *StrToInt* (жол) функциясының көмегіментұтас типтегі шамаға түрлендіріледі. Егер заттық сан енгізу қажет болса, *StrToFloat* (жол)функциясы қолданылады.

Нәтижені шығару қорытынды санның мәнімен *Caption* таңбасына *Label5* жолдың қасиетін беру арқылы жүзеге асырылады. Тұтас санды таңбалар жолына ауыстыру *IntToStr* (*тұтас сан*) функциясының көмегімен жүргізіледі. Егер заттық санды шығару қажет болса, *FloatToStr* (*заттық сан*) функциясы қолданылады.

Егер мәндердің соңғы жиынтығының біреуін енгізу қажет болса, онда интерфейстің басқа элементтерін қолдану ыңғайлы: тізімдер (*ListBox*), тізіммен берілген өрістер (*ComboBox*), есептегіштер (*UpDown*), жылжымалар (*TracBar*), ауыстырып-қосқыштар (*RadioButton*). «Есептеу жүйелері» бағдарламасында жүйенің негіздемесін енгізу үшін тізіммен берілген өрісті қолдану 3.6-суретте көрсетілген.Батырмаға басу кезіндеоң жақтаолардың ішінен біреуін таңдау үшін берілген мәндер тізімі түсіп қалады. Мәні тінтуірді шерту арқылы таңдалады.



3.6.-сурет. Тізіммен берілген өріс үшін енгізу

Тізімге кіретін мәндер бағдарламалау кезінде ComboBoxl.Items қасиетін қолдану арқылы енгізіледі. Тізіммен берілген өріс элементтері индекстелген - яғни нөлден бастап, реттік нөмірлерге ие болады. Әрбір элемент элементтің индекстелген атымен: ComboBoxl.Items[индекс] сәйкестендіріледі. Тізімнен элементті таңдағаннан кейін оның индексі ComboBoxl.ItemIndex өрісіне беріледі. Сандарды ауыстыру бағдарламасында «р» айнымалының сәнін енгізу келесі операторомен іске асырылады:

p:= StrToInt(ComboBoxl. Items[ComboBoxl. ItemIndex]);

Тізіммен берілген өріс бір өлшемді массивті ұсыну үшін интрфейсті ұйымдастыру тәсілі болады.

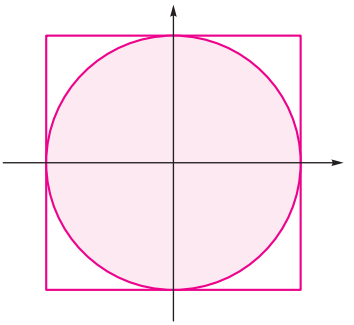
Жаттығулар

1. Параграфта берілген ондық емес санды есептеудің ондық жүйесіне ауыстыру бағдарламасын компьютерде консолды режимде іске асыру.
2. Параграфта берілген ондық емес санды есептеудің ондық жүйесіне ауыстыру бағдарламасын компьютерде терезелік режимде іске асыру.
3. Терезелік интерфейсті жобалау және желдің жылдамдығын «секундына метрден» «сағатына километрге» қайта есептейтін бағдарлама жазу.
4. Екі кедергіден тұратын, электр тізбегінің кедергісін есептейтін бағдарлама жазу. Кедергілер тізбектеліп немесе параллельді жалғануы мүмкін. Егер кедергінің шамасы 1 000 Ом асатын болса, нәтиже килоомда берілуі тиіс.
5. Интерфейсті жобалау және қабырғаларының ұзындығы бойынша үшбұрыштың ауданын есептейтін бағдарлама жазу. Үшбұрыштың тік бұрышты, тең бүйірлі, тең қабырғалы, еркін түрде болатынын байқау.
6. Интерфейсті жобалау және tc арқылы еркін құлайтын дененің жүріп өткен жолын написать есептейтін бағдарлама жазу. Бағдарламаны денеде алғашқы жүру жолы болғанда, жүрген жолды есептеу мүмкін болатындай құру керек.

3.4. СТАТИСТИКАЛЫҚ СЫНАҚ ӘДІСІН БАҒДАРЛАМАЛАУ

Статистикалық сынақ әдісі (Монте-Карло) — кездейсоқ шамаларды үлгілеуді және статистикалық бағалар мен ізделіп отырған шаманы алуды қолданатын, сандық әдіс. Бұл әдістің бірден-бір қолданылуы - фигуралардың аудандарын және дене көлемдерін есептеуде. Статистикалық сынақ әдісін қолдану арқылы π-Пифагор санын есептеу бағдарламасын жасайық.

3.7.сурет. Шеңберді шаршының ішіне салу



Әдістің идеясы мынада болады.

Бірлік шеңберінің айналасына қабырғаларының ұзындықтары 2-ге тең шаршы бейнеленеді (3.7-сурет).

Ықтималдықты бөлудің біркелкі заңымен берілген кездейсоқ сандардың құрылғысының көмегімен шаршы бойынша «ату», яғни шаршы ішіндегі нүктелерді кездейсоқ таңдау жүргізіледі. Әрбір мұндай таңдауды сынақ деп атайық. Сынақ *Random* функцияның көмегімен -1-ден 1-ге дейінгі мәндер шегінде (*X, Y*) нүктесінің координаталарын есептеу дегенді білдіреді. Содан соң бұл нүктенің шеңбер ішінде жату-жатпайтындығы анықталады. $X^2 + Y^2 \le 1$ болған жағдайда шарт орындалады. Егер нүкте шеңбер ішіне тиетін болса, онда тию есептегішіне бірлік енгізіледі.

P — сынақтардың жалпы саны делік. Олардың ішінен шеңберге тию *M* болды.Шаршының ауданы 4-ке тең. Шаршы ауданы сынақ нүктелерімен біркелкі жабылған жағдайда, шеңбер ауданы үшін келесі формула беріледі:

$$S_{kp} = 4 \lim_{P \to \infty} \frac{M}{P}.$$

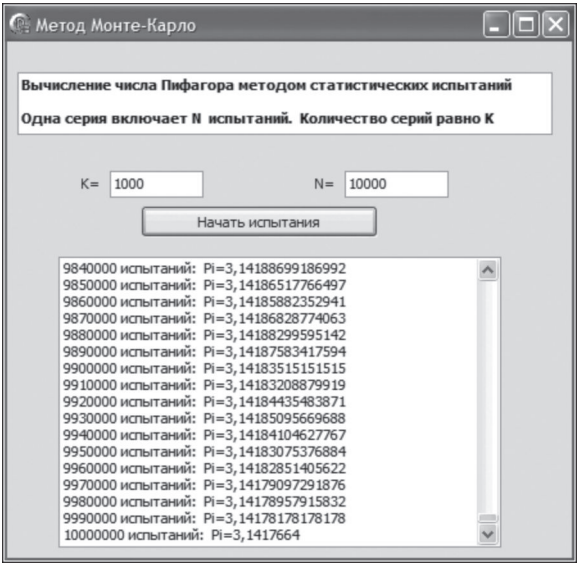
Формуланың мәні сынақ сандарының артуымен *M/P* қатынасы шеңбермен шаршының аудандарының қатынасына жақындай түседі және шексіздікке талпынған *P* болған шекте оған тең болады деп түсіндіріледі. Радиус шеңберінің ауданы 1 тең болғандықтан, π айтарлықтай көп мәнінде *P* жақындатылған теңдік орындалады:

$$\pi \approx 4 \frac{M}{P}.$$

P неғұрлым көп болса, соғұрлым осы теңдік дәл болады.

Осы есепті Delphi –де шығару бағдарламасының интерфейсі 3.8-суретте берілген. π санының мәнін шығаруды қадағалау үшін сынақтар серияларға бөлінеді. Бір серияда *N* сынақ жүргізіледі, алмұндай сериялардың саны *K* тең. Әрбір серия аяқталған соң экранға нәтиже шығарылады.

Формамен берілген терезеде интерфейстің келесі элементтері болады:



3.8.-сурет. « Монте-Карло әдісі» бағдарламасының интерфейсі

- *Memo1*— көп жолды мәтіндерді енгізу/редакциялауға арналған өріс (32 Кбайтқа дейін);
- *Edit1* және *Edit2* — *K* және *N* мәндерін енгізуге арналған өрістер;
- *Label1* және *Label2*— енгізу өрісіне берілген таңбалар;
- *Button1* — есептеуді жіберуге арналған командалық батырма;
- *ListBox1*— бұралатын тізімді енгізуге арналған өріс.

ButtonlClick оқиғаны өңдеу келесі прцедура арқылы жүргізіледі:

```
procedure TForm6.Button1Click(Sender: TObject);
Var i,j: integer; K,N,M: Int64;
    X,Y,Pi: double;
begin
    K:= StrToInt(Edit1.Text); {Серия сандарын енгізу}
    N:= StrToInt(Edit2.Text); {сынақ сериясының ұзындығын
    енгізу}
    Randomize;
    M:= 0; {Шеңберге  кіру есептегішін жүктеу}
    for i:= 1 to K do {Сынақ сериясының нөмірі бойынша цикл)
    begin
            for j:= 1 to N do{Сынақтарды қайталау)
```

```
begin
        X := 2*Random — 1; {X-нүктенің координатасын
        есептеу}
        Y := 2*Random — 1; {Y-нүктенің координатасын
         есептеу}
        if X*X+Y*Y< = 1
        then M:= M+1 {шеңберге тию санын есептеу}
end;
Pi:= 4*M/(i*N); {аяқталған соң π есептеу} серия}
{Сынақ санын және п шығару}
ListBoxl. Items. Add(IntToStr(i*N)+' сынақтар:
Pi='+ FloatToStr(Pi))
end;
end;
```

ListBox өріске шығару ListBox1Items.Add(жол) әдісін шақыру арқылы жүргізіледі.Осы әдіске әрбір жүгінген сайын *ListBox1* тізіміне жол қосады. Шығарылған жолдардың барлық тізімін көру үшін айналдыру сызғышы қолданылады.

3.8.-суретте көрсетілген нәтижелерді талқылайық. 1 000 серияға арналған есептеулер орындалды. Оның әрқайсысы 10 000 сынақтан тұрады. Соңғы нәтиже 10 млн сынаққа сәйкес келеді. Нәтижесінде π санының мәнінде тек төрт дұрыс цифр алынды: 3,141. Дәлірек мәні: π = 3,14159265..

π санының небәрі төрт цифры неліктен мұндай қымбат жолмен (100 млн сынақ!) алынды деген сұрақ туындайды? Және бұдан – келесі сұрақ туындайды: егер сынақ санын арттыратын болсақ, осындай әдіспен π санының қаншалықты дұрыс цифрын алуға болады? Жауабы: теориялық тұрғыдан –иә, практикалық тұрғыдан –жоқ! Мұның сыры неде?

Оның себебі заттық сандармен берілген *машиналық есептеулер ақаулығында*. Осы ақаулықтың нәтижесінде машиналық қате тәртібінің ара қашықтығына шеңбердің шекарасына жақын орналасқан нүктелер үшін шеңберге тию дагностикасы әр кезде дәл бола бермейді.Сынақтардың кейбір ауыспалы сыни саны болады, одан асып кеткен кезде, π санын есептеу дәлдігі артпай, керісінше төмендейді. Рны тәжірибе түрінде анықтап көріңдер!

Монте-Карло әдісінің көмегімен жазықтықта орналасқан күрделі формадағы фигуралардың ауданын немесе үш өлшемді денелердің көлемін есептеуге болады. Оның идеясы сол қалпында: фигура ауда-

ны немесе көлемі белгілі қарапайым түрдегі басқа фигураға орналастырылады. Мысалы, жазық фигура үшін – төртбұрыш немесе шеңбер, көлемді дене үшін - тікбұрышты параллелепипед немесе шар. Содан соң, таратудың біркелкі заңымен берілген кездейсоқ сандардың құрылғысының көмегімен нысанаға ату жүргізіледі. Сыналатын фигураға тиюлер қандай да бір тәсілмен диагностикадан өтеді. Бұдан әрі тию сандарының сынақтың (атулардың) жалпы санына қатысы бойынша ізделіп отырған аудан немесе көлем есептеледі.

Жаттығулар

1. Параграфта берілген Пифагор санын есептеу бағдарламасын компьютерде іске асыру. Сынақ санының артуымен нәтиженің қалай өзгергендігін байқау.

2. Интерфейсті жобалау және сызықтық өлшемдері бойынша дұрыс кеңістіктік фигуралардың (тетраэдр, куб, төртбұрышты пирамида) көлемін есептейтін бағдарламаны жазу.

3. Интерфейсті жобалау және таңбаның мәтінге неше рет кіргенін тексеретін бағдарламаны жазу; таңба мен мәтінді пайдаланушы береді.

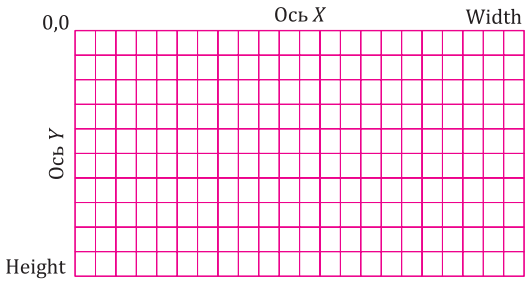
4. Интерфейсті жобалау жәнеенгізілген мәтінді: барлық бас әріптерді кіші әріптерге ауыстырып, бағдарламаны жазу

3.5. ЖОБАЛЫҚ ТАПСЫРМА: ФУНКЦИЯ КЕСТЕСІН ҚҰРУ

Тапсырманың мақсаты — енгізу терезінде $y = f(x)$ функциясының кестесін енгізуді алу бағдарламасын құру.

1-кезең. Бұдан әрі берілген мәтінді және қосымша дереккөздерді пайдаланып, құрастыру құралдарыменDelphi-де өз бетінше графикалық бейнелерді құру.

Delphi-де графикалық бейнелерді алу үшін кенеп – Canvas объекті қолданылады. Canvas объекті жеке компонент түрінде емес, форманың касиеті түрінде (Form.Canvas) қолданылады. Суретті кенепке шығару Paint формасының оқиғасы пайда болғанда жүргізіледі. Бұл оқиға қосымша терезесін шығару қажеттілігі туындаған әрбір жағдайда пайда болады (қосымшаны іске қосқанда немесе терезе түрін жаңартқанда). Бұл оқиғаны өңдеу FormPaint процедурасымен жүзеге асырылады.



3.9.-сурет. Графикалық тордың құрылымы және параметрлері

Кенепке сурет салу терезенің *графикалық бетінде* нүктелерді бояу арқылы жүргізіледі. Мұндай нүктелердің (пиксельдердің) жиынтығы графикалық тор құрайды. Әрбір тор нүктесінің жағдайы екі координатамен: көлденең (*X*) және тігінен (*У*) сипатталады. Координаталар сол жақ жоғарғы бұрыштан бастап саналатынын ескерген жөн. Графикалық тордың бір қадамы көршілес пиксельдердің арасындағы ара қашықтыққа тең болады (3.9-сурет).

X өсі бойынша қадамдардың максималды саны ClientWidth формасының қасиетін, ал *У* өсі бойынша - ClientHeight формасының қасиетін сақтайды.

Кез келген суретті салу үшін негізгі графикалық қарапайымдылар – нүктелер, сызықтар, шеңберлер, тікбұрыштар және т.с.с қолданылады. Мұндай қарапайымдыларды салу үшін Canvas объекітінің тиісті әдістері қолданылады.

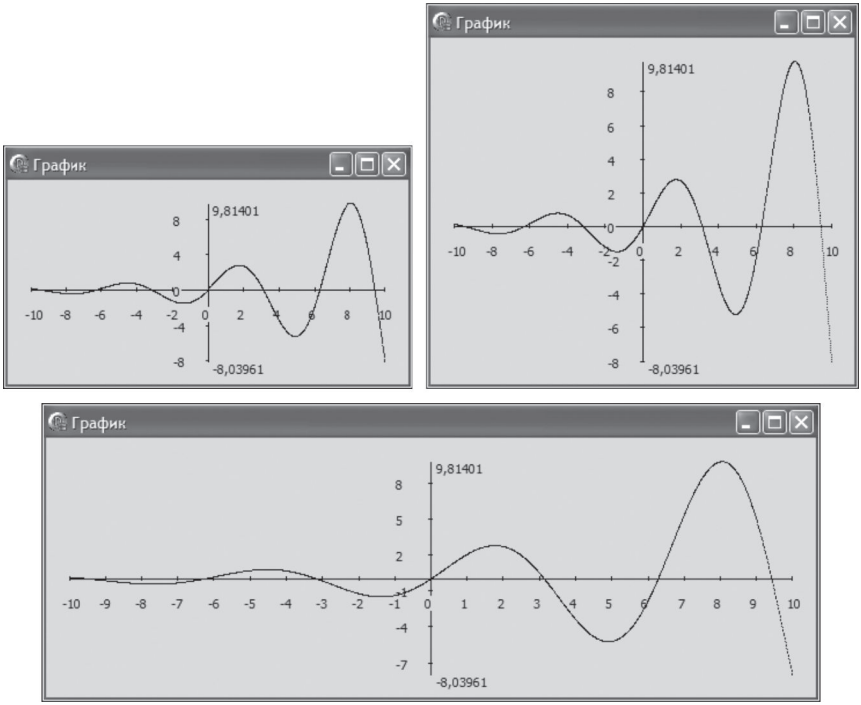
Бұдан әрі берілген функция кестесін салу бағдарламасында кенепте бейнені және жазуларды енгізу үшін келесі әдістер қолданылады:

- MoveTo (x,y: integer) — x,y координаталарының берілген мәндерімен поизцияда сурет салу үшін ағымдағы нүктенің сілтемесін орнатады;
- LineTo (x,y: integer) — ағымдағы нүктеден берілген координаталары бар нүктеге дейінгі сызықты сызады. Сызықтың түрін Pen (қаламұш) қасиеті анықтайды;
- TextOut (x,y: integer; s: string) — (x,y) координаталары бар нүктеден s жолын экранға шығарады. Қаріп мәтін шығарылатын (Canvas) жазықтығына Font қасиетін анықтайды. Мәтінді шығару бөлігін бояйтын түсті осы жазықтықтың Brush (қылқалам) қасиеті анықтайды. Басқа әдістер туралы мәліметтерді Delphi сипаттамаларында қараңдар.

Функцияның кестесі нүктелер бойынша

Pixels[x,y]:=z,

қасиетінің көмегімен шығарылады, мұндағы x, y — пикселдің координаталары; x — пикселдің түсі.



3.10.–сурет. GrOfFunc процедурасының көмегімен алынған шығару терезесінің түрлі өлшемдеріне арналған функцияның кестесі

2-кезең. Әрі қарай берілетін функция кестесін құру бағдарламасымен танысу. Бағдарлама мәтінін Delphi жүйесіне енгізу және оны ретке келтіру (егер шығару кезінде қателер жіберілетін болса). Бағдарламаны атқару және 3.10-суретте берілгенге ұқсас нәтижелерді алу.

Әрі қарай берілген бағдарлама $x \in [-10, 10]$ кесіндісінде берілген $f(x) = 2\sin x \cdot e^{x/5}$ функцияның кестесін құруды орындайды.

Функция мәндерін есептеу үшін ішкі бағдарлама -$f(x)$функциясы құрылды. Кестені құруды GrOfFunc атауымен берілген процедура орындайды. Бағдарлама мәтіні егжей-тегжейлі түсініктемелермен жабдықталған. Онымен мұқият танысып, бағдарлама мазмұнын ұғуға тырысыңдар!

```
{КЕСТЕЛІК ҰСЫНУҒА АРНАЛҒАН ФУНКЦИЯ}
    Function f(x:real): real;
    begin
```

```
      f:= 2*sin(x)*exp(x/5);
end;
{КЕСТЕНІ ҚҰРУ ПРОЦЕДУРАСЫ}
Procedure GrOfFunc;
var  x1,x2, {Функция аргументін өлшеу шекарасы}
      y1,y2, {Функция мәдерін өлшеу шекарасы}
           x, {Функция аргументі}
           y, {х нүктесіндегі функцияның мәні}
      dx: real; {Аргументің үстелуі}
   l,b:integer; {Кестені шығарудың сол жақ төменгі бұрышы}
      w,h:integer; {Кестені шығарудың ені және биіктігі}
   mx,my:real; { X жәнеY өстері бойынша масштабы}
x0,y0:integer; {Нүкте —координаталар басы}
n, sh, s: integer; {Өстердегі бірлік таңбалар саны}
begin
{кестені шығару бөлігін есептеу}
l:= 20;
b:= Form6.ClientHeight-20; {Y —сол жақ төменгі бұрыштың
координатасы}
h:= Form6. ClientHeight-4 0; {Биіктігі}
w:= Form6.ClientWidth-40; {Ені}
x1 := -10; {Аргумент диапазонының төменгі шегі}
x2 := 10; {Аргумент диапазонының жоғарғы шегі}
dx := 0.01; {Аргумент қадамы}
{Максималды жән минималды мәндерді есептеу}
{кесіндідегі функциялар [x1,x2]}
y1 := f(x1);{Минимумге арналған айнымалы}
y2 := f(x1);{Максимумге арналған айнымалы}
x := x1;
repeat
      y:= f(x);
      if y < y1 then y1:= y;
      if y > y2 then y2:= y;
      x:=x+dx;
until (x>=x2);
{Масштабты есептеу
my := h/abs(y2-y1);{ Y өсі бойынша масштаб}
mx := w/abs(x2-x1);{ X өсі бойынша масштаб }
x0:= l+abs(Round(x1*mx));{Басталудың қалпы}
```

```
        y0 := b-abs(Round(y1*my)); {координаталар}
            {ӨСТЕРДІҢ СУРЕТІН САЛУ ЖӘНЕ БЕЛГІЛЕУ}
                with Form6.Canvas do
                begin
                    MoveTo (x0, b); LineTo(x0,b-h); { Y өсін саламыз}
                    MoveTo(l,y0); LineTo(l+w,y0); {  Хөсін саламыз}
    { 0Y өсін белгілейміз: Y max және min өсіне функцияның мәндерін
енгіземіз}
    TextOut(x0+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
    TextOut(x0+5,b,FloatToStrF(y1,ffGeneral,6,3));
    n := Round(Form6.ClientHeight/40); {таңбалар саны
    {Функция мәндерінің өзгеру диапазонына байланысты таңбалар-
ды орналастыру қадамын таңдаймыз}
    If (y2-y1)<100 Then
        begin
            sh:= round((y2-y1)/n);
            s := Round(yl);
        end
    Else
        Begin
            {Тек ондықтарды белгілейміз}
            sh:= (Round((y2-y1)/n) div 10)*10;
            s := (Round(yl) div 10)*10;
        end;
    repeat
        MoveTo (Round(x0-2), Round(y0+s*my)); {тәуекелдерді
        саламыз}
        LineTo (Round(x0+2), Round(y0+S*my)); {және сандар}
        TextOut(x0-30,Round(y0-5+s*my),FloatToStrF(- 1*S,ffGeneral,
        6,3));
        S := S+sh;
    until (s > y2);
    // 0X өсін белгідлеу қою}
    n:= Round(w/40);
    If (x2 x1)< 100 Then { 0X өсі  үшін  ұқсас}
        begin
            sh:= round((x2-x1)/n) ;
            s:= Round(xl);
        end
```

```
Else
      begin
      sh : = (Round((x2-x1)/n) div 10)*10; s:=(Round(x1) div 10)*10;
end;
repeat
      MoveTo (Round(x0+s*mx),y0-2);
      LineTo (Round(x0+s*mx),y0+2);
      TextOut(Round(x0-5+s*mx),y0+15, FloatToStrF
       (S,ffGeneral,6,3));
      S := S+sh;
until (s > x2);
{ФУНКЦИЯ КЕСТЕСІН ҚҰРУ}
      x:= x1;
      repeat
         y:= f(x);
{графикалық тор нүктесін қызыл түспен бояу}
               Pixels [x0+Round(x*mx),y0-Round(y*my)]:= clRed;
               x:= x+dx;
      until (x >= x2);
end;
end;
{Форманы ашумен берілген кестені құруды жіберу}
procedure TForm6.FormPaint(Sender: TObject);
begin
      GrOfFunc;
end;
{Форма өлшемін өзгерту арқылы кестені қайта сызу}
procedure TForm6.FormResize(Sender: TObject);
begin
{Форманы тазарту}
Form6.Canvas.FillRect(Rect(0,0,ClientWidth,ClientH eight));
{Кесте құру}
GrOfFunc;
end;
```

Шығару терезелерінің өлшемдерін қолмен өзгертсе, онда Resize деп аталатын форманың оқиғасы пайда болады. Бұл оқиға форманы ескі суреттен тазартатын және терезенің жаңа өлшемі үшін кестені қайта сызуды қамтамасыз ететін FormResize процедурасымен өңделеді. Кенептің түрлі өлшемдеріне арналған кестені бейнелеудің үш

нұсқасы (шығу терезелері) 3.10-суретте берілген.

Кесте кестелік тордың 20 қадамына тең болатын, шығару терезесінде сол, оң, жоғарғы және төменгі өрістері бар, тікбұрыштың шегінде құрылады. *x* координатаның математикалық мәнінің өзгеру қадамы 0,01 тең. *X* өсі және *Y* өсі бойынша масштабтар (*mx* және *my* айнымалылар) *x* аргументінің және *y* функциясының математикалық мәндерінің бірлігіне келетін кестелік тордың қадам сандарына тең болады. Келесі терминологияны қолдану қабылданған: *x* аргументінің және *y* функциясының математикалық мәндері функция кестесінің қалпын координаталардың әлемдік жүйесінде анықтайды. Ал 3.9-суретте берілген құрылым координаталардың экрандық жүйесідеп аталады. *mx* және *my* масштабтары кесте нүктелерінің қалпын *әлемдік жүйеден координаталардың* экрандық жүйесіне қайта есептеу үшін қолданылады.

3-кезең. Төменде берілген бақылау сұрақтарға жауап беру және тапсырмаларды орындау. Сұрақтарға жауап беру және тапсырмаларды орындау нәтижелерін жазбаша есеп түрінде рәсімдеу.

БАҚЫЛАУ СҰРАҚТАР ЖӘНЕ ТАПСЫРМАЛАР

1. Форманың қандай қасиеті онда графикалық суреттер салуға мүмкіндік береді?
2. Қандай параметрлер кенептің өлшемін сипаттайды?
3. Координаталардың экрандық жүйесінің өстері қалай орналасқан?
4. Қандай әдістердің көмегімен кенепке суреттер пен жазбалар салуға болады?
5. Әлемдік координаталар жүйесі деген не?
6. GrOfFunc процедурасында *mx* және *my* шамалары не үшін қолданылады?
7. Қандай оқиға пайда болғанда кесте салу бағдарламасы пайда болады?
8. Кестені шығару терезелерінің (форманы) өлшемдерін өзгерту арқылы қайта сызу нені есебінен жүргізіледі?
9. *X* мәнінің басқа диапазонында басқа функцияның кестесін салу үшін бағдарламада нені өзгерту керек?
10. Координаталар өстерінің ұшында жебе салып, *X* және *Y* өстерінің белгіленуін жазу үшін бағдарламаға қандай операторларды қосу керек?
11. Кесте сызықтарының түсін қалайша өзгертуге болады?
12. Неліктен шығару терезесінің өлшемі өзгерген кезде өстерді белгілеу өзгереді? Бағдарламада ол қалай жүзеге асырылатынын талдаңдар.

ДЕРЕККӨР ТЕОРИЯСЫНЫҢ НЕГІЗДЕРІ

4.1. АҚПАРАТТЫҚ ЖҮЙЕЛЕР ЖӘНЕ ДЕРЕККӨР ТУРАЛЫ ТҮСІНІК

Қазіргі кезде адамдар анықтамалық ақпарат алу үшін компьютерді қолданады. Тіпті сауда орталықтарында сатушылар қажет тауар туралы мәліметті компьютердің көмегімен біліп ала алады. Кейбір орталықтарда анықтамалық компьютерлерге (оларды терминалдар деп атайды) қол жетімділік келушілерге беріледі. Теміржол немесе әуежайлық кассаларда кассирлер компьютерді сізге қажет билеттің бар-жоғын анықтау үшін қолданады. Қонақүйлерде компьютердің көмегімен бос орындардың болуы туралы, сонымен қатар қажет күнге нөмірді брондауға болады. Барлық жоғарыда аталған мысалдар компьютерлік технологияны қолданудың бір саласына ғана жатады, ол ақпараттық жүйелер деп аталады.

Ақпараттық жүйе — бұл дерекқордағы ақпараттың, сондай-ақ ақпаратты өңдеуді қамтамасыз ететін ақпараттық технологиялар мен ақпараттық құралдардың жиынтығы. Кез келген ақпараттық жүйенің белгілі бір қолданылу саласы болады.

Ақпараттық жүйелерді кеңінен тарату 3-ші буындағы ЭЕМ-дан басталады. Дәл сол кезден бастап компьютерлерде сыртқы жадының құрылғысы ретінде магнитті дисктердегі жүктегіштер қолданыла бастады. Магнитті дисктер алғашқы екі ұрпақтың машиналарында қолданылған, тізбекті қол жетімді құрылғы - магнитті жолақтарға қарағанда тікелей қол жетімді құрылғысы болып саналады. Соның нәтижесінде дисктердегі деректер жолақтағыға қарағанда тезірек өңделеді.

3-ші буынды машиналардың тағы бір маңызды ерекшелігі – ЭЕМ-да көппайдаланушылық режимінде жұмыс жасау мүмкіндігі, яғни бір машинаға бір уақытта ақпаратты дербес терминалдар - енгізу және шығару құрылғылары (пернетақта және монитор) арқылы көптеген пайдаланушылар қол жетімді болуы. Көп пайдаланушылық жұмыс

режимін қолдауды операциялық жүйелер қамтамасыз етеді.

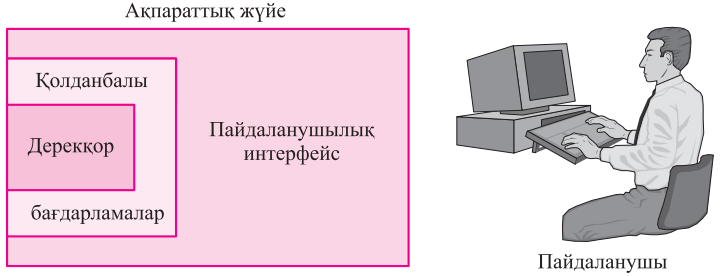
Желілік технологияларды дамыту ақпараттық жүйелерді таратуда орасан зор түрткі берді. Бір кәссіпорын, мекеме аясында ақпараттық жүйелер *корпоративті желі негізінде* жұмыс жасайды. Сонымен бірге барлық ақпарат бір торапқа жұмылдырылады, сондай-ақ көпшілікке қол жетімді деректердің түрлі бөліктері желінің түрлі тораптарында сақталуы да мүмкін болады.

Неғұрлым ірі ақпараттық жүйелер *ғаламдық компьютерлер желілері негізінде* жұмыс жасайды. Оның мысалы ретінде «Полет¬-Сирена» — ауа көлігінің ақпараттық жүйесін алуға болады. Бұл жүйеге қол жетімді терминал ретінде Интернетке қосылған кез келген компьютер бола алады. Алайда жалпыға жатпайтын, шектелген қол жетімділікпен және масштабпен берілген көптеген «ғаламдық» ақпараттық жүйелер – *корпоративті жүйелер* бар. Олар бір-бірімен бір ведомство кәсіпорындарының оқшауланған желілерін біріктіре алады және аймақ, министрлік және т.б. аясында олардың ортақ тиімді басқарылуына ықпал етеді.

Ақпараттық жүйенің құрылымы сызба түрінде 4.1-суретте бейнеленген. Кез келген ақпараттық жүйенің негізі ретінде дерекқор болады.

Дерекқор — белгілі бір тақырыптық салаға жататын, компьютердің сыртқы жадында сақталатын және үнемі қолданыста болатын, деректердің белгілі бір тәсілімен ұйымдастырылған жиынтығы.

Дерекқор — бұл небәрі тек сақталған ақпарат. Ал ақараттық жүйе деректерді мүдделі адамдардың – пайдаланушылардың осы сақтау қорынан қолдануын қамтамасыз етеді. Пайдаланушының ДҚ сұраныстырына қызмет көрсетуді: деректерді іздестіруді,



4.1. –сурет. Ақпараттық жүйенің құрамы

Оларды ыңғайлы түрде ұсынуды, өңдеуді және талдауды *дерекқор қосымшалары* деп аталатын бағдарлама орындайды.

Ақпараттық жүйенің пайдаланушысы есептеу техника саласында маман болуы міндеті емес. Сондықтан ақпараттық жүйелердің клиенттік қосымшалары пайдаланушыға ақпараттық жүйенің барлық мүмкіндіктерін іске асыруға және оның тарапынан жол берілмейтін әрекеттердің алдын алуға мүмкіндік беретін, қарапайым, көрнекі, интуитивті-түсінікті интерфейске ие болулары керек.

4.2. ДЕРЕКҚОРДЫ ӘЗІРЛЕУ КЕЗЕҢДЕРІ

Ақпараттық жүйенің негізі дерекқор болады. Көбіне бүкіл ақпараттық жүйе жұмысының тиімділігі ДҚ қалай ұйымдастырылғанына байланысты болады. Үлкен ДҚ әзірлеуге түрлі саладағы көптеген мамандар қатысады. Осындай жұмыстың негізгі кезеңдерін қарастырайық, негізгі түсініктерді енгізейік.

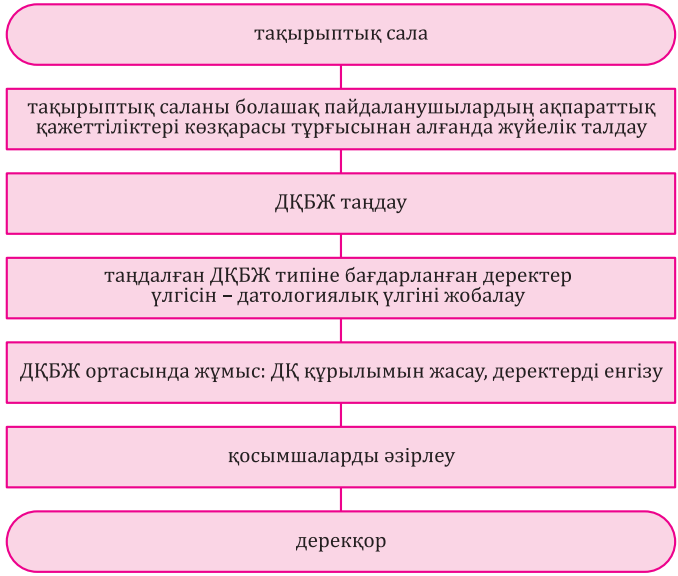
ДҚ тақырыптық саласы — бұл дерекқорда көрініс табатын, шынайы өмірдің бір бөлігі. Мысалы, егер ақпараттық жүйе үлкен кітапхананың оқырмандарына қызмет көрсету үшін арналған болса, онда оның тақырыптық саласы ретінде кітаптар мен мерзімдік басылымдардың кітапханалық қоры болады. Егер ақпараттық жүйе пйыздардың жүретін жолы, жүру уақыты, билеттердің болуы туралы және т.с.с. кез келген ақпаратты ұсынып, теміржол жолаушыларына қызмет көрсететін болса, онда тақырыптық сала ретінде теміржол бойымен жолаушыларды тасымалдау жүйесі болады.

Тақырыптық саланың инфологиялық үлгісі — бұл пайдаланушылардың тақырыптық саланың құрамы мен құрылымы туралы түсініктерін біріктіретін тақырыптық саланың жүйелік сипаттауы.

Дерекқорды басқарудың жүйесі(ДҚБЖ) — бұл дерекқорды құруға және қолдануға арналған бағдарламалық қамсыздандыру.

Деректер үлгісі (ДҚ схемасы) — қолданылатын ДҚБЖ типіне бағдарланған, дерекқордағы деректерді ұйымдастыру құрылымын сипаттау.

ДҚ құру кезеңдерінің жүйелігі 4.2.-суретт көрсетілген. Жұмыс ДҚ құрылатын тақырыптық саланы *жүйелік талдауынан* басталады. Бұл жұмыстың нәтижесі ретінде ақпараттық-логикалық (инфологиялық) үлгі болады.



4.2.-сурет. ДҚ құру кезеңдері.

Келесі қадамда ДҚ құру және қосымшаларды іске асыру үшін қолданылатын *ДҚБЖ таңдау* жүргізіледі.

Содан соң бірінші кезеңде құрылған, инфологиялық үлгіні бейнелейтін деректер үлгісін әзірлеу жүргізіледі. Деректер үлгісінің құрылымы қолданылатын ДҚБЖ енгізілетін деректерді ұсыну тәсіліне бағдарланған. Мұдай үлгі деректердің *датологиялық үлгісі* деп аталады.

Алдыңғы кезеңдер теориялық (жобалық) сипатта болады. Содан соң *ДҚБЖ ортасындағы жұмыс* басталады. ДҚ құрылымы жасалып, деректерді енгізу жүзеге асырылады.

Келесі кезең — *қосымшаларды әзірлеу.* Пайдаланушылардың ақпараттық қажеттіліктерін қамтамасыз ететін, деректерді айла-шарғылау тілінде бағдарламалар жазылады.

Курсымыздың келесі бөлімдерінде сендер ДҚ әзірлеудің аталған кезеңдерінің әрқайсысымен танысасыңдар, оларды оқу мысалдары арқылы жүзеге асыра аласыңдар.

Үлкен кәсіпорындарға немесе тіпті экономиканың тұтас салаларына қызмет көрсететін үлкен ақпараттық жүйелерді әзірлеу – көп еңбек етуді қажет ететін процесс. Мұндай жүйелерді жасауда өте көп адам қатысады, қомақты қаржылық ресурстер жұмылдырылады.

Әдетте, әзірлеу уақыты қатаң түрде шектелген, ал жүйе жұмысының сенімділігіне қойылатын талаптар өте жоғары болады. Жүйе ұзақ уақыт қолдану үшін жасалғандықтан, ол әзірлеуші тарапынан үнемі сүйемелдеуді қажет етеді. Әсіресе, тақырыптық салада шарасыз өзгерістер деректер құрылымына, қосымшаларға және т.с.с тиісті өзгерістер енгізуге мүмкіндік беретін, жүйенің белгілі бір икемділігін талап етеді. Бұл жұмыстың бәрін «қолмен» орындау іс жүзінде мүмкін емес.

1990 жылдардан бастап бағдарламалаудың технологиялары саласында Case-технологиялар (Computer-Aided Software/ System Engineering) деген атауға ие болған бағдар құрылды.

Case-технология — бұл күрделі жүйелерді талдау, жобалау, әзірлеу және сүйемелдеу үшін автоматтандыру әдістері мен құралдарының жиынтығы.

Case-технологияларда ақпаратты визуалды ұсыну әдістері, тақырыптық саланы үлгілеудің графикалық құралдары, соның ішінде деректерді ұйымдастырудың инфологиялық үлгісін, датологиялық үлгісін (ДҚ схемалар) құру үлкен рөл атқарады.Кез келген Case-жүйе әзірлеудің кез келген кезңінде өзара әрекет етуге және ақпараттық жүйені пайдалануға мүмкінік беретін, әзірлеушілерге де, тапсырыс берушіге де түсінікті болатын стандартты графикалық тілді қолданады. Case- жүйелер құрылған деректер үлгісі (құрылымдық немесе объектілік) мен тиісті ДҚБЖ арасындағы байланысты қамтамасыз етеді. Мұндай мүмуіндіктердің бәрі жүйені әзірлеу үрдісін тездетеді, қателердің ықтималдығын азайтады және оны жүйемелдеуді жеңілдетеді.

Міне, ақпараттық жүйелерді жобалауға қолданылған кейбір Case-құралдарға мысал келтірейік (жақшада өндіруші-компания көрсетілген): PowerDesigner (*Sybase*), Designer 200 (*Oracle*), Visio Enterprise (*Microsoft*) және басқалары.

4.3. ТАҚЫРЫПТЫҚ САЛАНЫҢ ИНФОЛОГИЯЛЫҚ ҮЛГІСІ

ДҚ оқу мысалын құруға көшейік. Бұл жүйенің тақырыптық саласы сендерге жақсы таныс: бұл мектеп. Егер мектепті оның толықтай қалпында зерттелетін жүйе ретінде қарастыратын болсақ, онда мұндай міндет біздің оқулығымыздың аясында өте күрделі болады. Сөзсіз, мектеп – бұл оны құрайтын көптеген құрама элементтерден тұратын, олардың арасындағы байланысы айтарлықтай күрделі құрылымнан

тұратын жүйе. Мектеп жүйесінде мынадай қосалқы жүйелерді бөліп қарауға болады:

- басқару қосалқы жүйесі (директор, директордың орынбасарлары);
- оқытушылардың қосалқы жүйелері;
- оқушылардың қосалқы жүйелері;
- мектептің материалдық базасы (ғимарат, оқу жабдықтары, кітапхана және т.б.);
- және т.б.

Тақырыптық салағажоғарыда аталған мектептің қосалқы жүйелерінің қызмет ететін барлық тараптары кіретін ДБ құру міндеті өте күрделі болады. Сондықтан біз олардың тек тек біреуін, бірақ мектептің ең басты функциясы – оқу процессін қарастырумен шектелеміз. Мектептегі оқу процессінің инфологиялық үлгісін құрайық. Мұндай үлгіні сипаттау үшін оның қандай мақсатта қолданылатынын анықтау қажет. Үлгінің қасиеттері оның мақсатына байланысты екені мәлім.

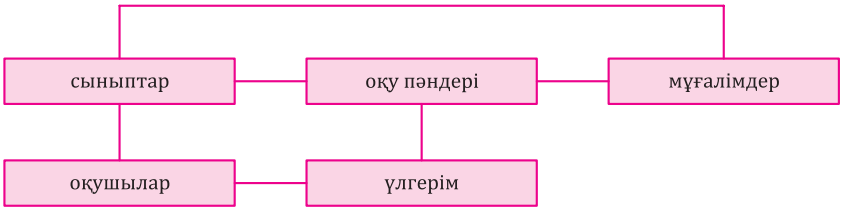
ДҚ құрылатын ақпараттық жүйенің мақсатын анықтайық. Ол пайдаланушыларды ақпараттандыруы тиіс:

- сыныптардың оқушылық құрамы туралы;
- мектептің оқытушылық құрамы туралы;
- оқытушылар мен сынып жетекшісінің оқу жүктемелерін бөлу туралы;
- оқушылардың үлгерімі туралы.

Есепті жеңілдету үшін оқушылардың түрлі пәндер бойынша бір оқу жылы ішіндегі тоқсандық және жылдық бағаларымен шектелейік.

Есептің қойылуынан объектілердің құрамы – мектеп жүйесінің элементтері белгілі болды. Біріншіден, бұл сыныптыр (5а, 7б және басқалары). Әрбір сынып — бұл оқушылар мен сынып жетекшісі кіретін мектептің қосылқы жүйесі. Объектілердң екінші типі ретінде оқушылар болады. Үшінші типі — мектептің мұғалімдері.

Мұғалім сыныпта белгілі бір пәнді оқытады. Оқу пәнін біздің жүйеде мұғалімдерді сыныптармен байланыстыратын, объектінің жеке типі деп қарастыруға болады. Тағы бір байланыстыратын объекті ретінде, оқушыларды оқылатын пәндермен байланыстыратын үлгерім табелін атауға болады. Жүйенің мүмкін болатын объектілері мен олардың арасындағы байланысты көрсететін сызбалық бейнесін салып көрсетейік (4.3-сурет). Мұндай сызба «граф» деп аталады – объектілердің (тікбұрыштардың) және олардың арасындағы байланыстардың (сызықтардың) жиынтығы.



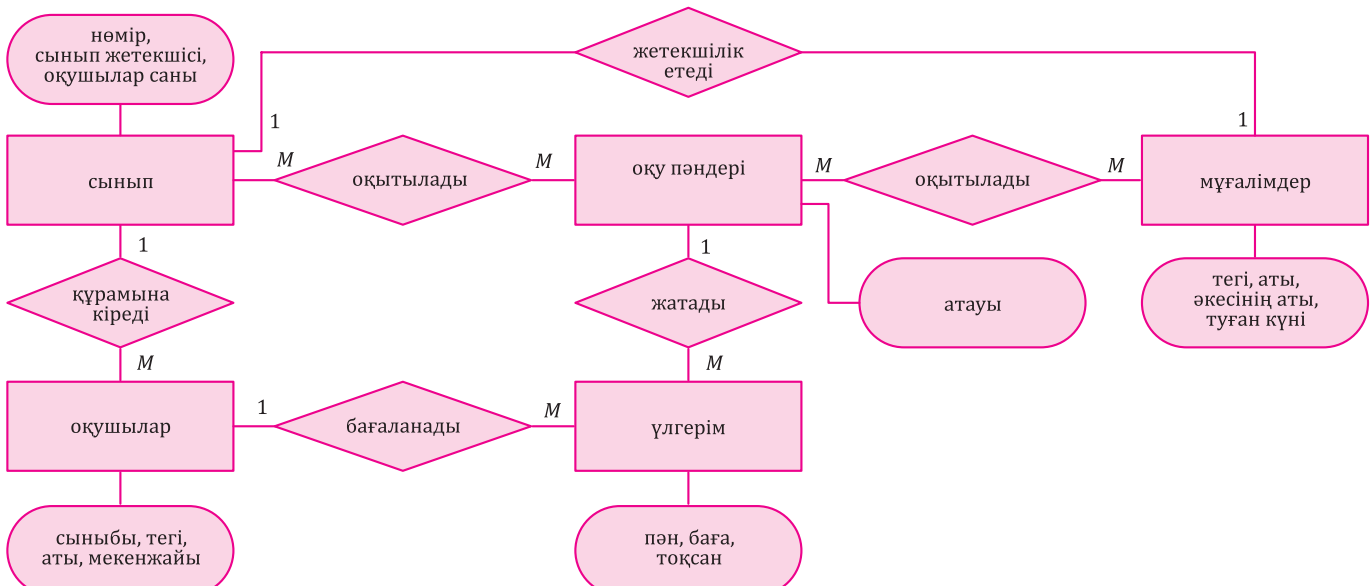
4.3.-сурет.Жүйеніің негізіг объектілері және олардың арасындағы байланыс.

Осы сызбадағы әрбір байланыс сызығының белгілі бір мәні бар. Мысалы, сыныптар мен оқушылар арасындағы сызық «құрамына кіреді» деген мағынаны, ал мұғалімдер мен оқу пәндерінің арасындағы байланыс «оқытады» және т.с.с. білдіреді. Мұғалімдер мен сыныптарды тікелей байланыстыратын сызық байланысты пәндер арқылы емес, сыныпқа жетекшілік ету арқылы берілетіндігіне назар аударыңдар.

Мұндай графты тақырыптық саланың ақпараттық-логикалық үлгісі, немесе қысқаша - *инфологиялық үлгісі* деп атайды. Инфологиялық үлгінібейнелеу үшін П.Чен (1976 ж.) *«мән—байланыс»* (ER-диаграммалар) типтегі диаграммаларды қолдануды ұсынды. 4.3.-суреттегі сызбамызға ER-диаграмманың тұрпатын беру үшін, оған байланыс атауы көрсетілген қиықшаларды және *объектілердің атрибуттары* көрсетілген сопақшаларды қосу керек. Оқу процессінің инфологиялық үлгісі 4.4.-суретте берілген.

Атрибуттар — бұл жүйеге кіретін объектілердің қасиеттері. Көріп отырғандай, барлық қасиеттер үлгіге кіре бермейді. Тек жүйені пайдалану көзқарасы тұрғысынан алғанда қажетті қасиеттер ғана ескеріледі.

Байланысқа берілген атау (қиықшада) оның мағынасын анықтайды. Байланыстың тағы бір сипаты бар – *байланыс типі* - (ДҚ теориясында ол түбегейліктің көрсеткіші деп аталады). Байланыс типтері келесідей болады: «бірге бір» (1-1), «көпке -бір» (1-М) және «көпке -көп» (М¬М). Мысалы, СЫНЫП, ОҚУШЫ объектілері арасындағы байланыс - «көпке-бір», себебі бір сыныпта оқушылар көп, ал бір оқушы бек бір сыныпқа ғана жатады. СЫНЫП және ОҚУ ПӘНІ объектілері расындағы байланыс - «көпке-көп», өйткені бір сыныпта көптеген оқу пәндері оқылады, және бір оқу пәні көптеген сыныптарда оқытылады. МҰҒАЛІМ және ПӘНДЕР объектілері арасындағы байланыс сондай ақ «көпке-көп» типіне ие болады: бір мұғалім



4.4.-сурет. Оқу процессінің инфологиялық үлгісі

бірнеше пәнді оқытады, және сол пәнді бірнеше мұғалымдер оқыт алады. «Сыныпқа жетекшілік ету» байланысы «бірге-бір» типіне ие болады, өйткені бір мұғалім тек бір сыныпта ғана сынып жетекшісі бола алады, және біп сыныпта тек бір сынып жетекшісі болады.

4.4. РЕЛЯЦИЯЛЫҚ ДЕРЕККОР. ДЕРЕККОРДЫ БАСҚАРУ ЖҮЙЕЛЕРІ

Тақырыптық саланың инфологиялық үлгісінің негізінде кейінкомпьютерлік ДҚ-да іске асырылатын, деректер үлгісі жасалады.

Дерекқорды жіктеу. ДҚ түрлі нышандары бойынша жіктеулер қолланылады. Жіктеудің бірінші нышаны - сақталатын ақпараттың мазмұны бойынша. *Фактографиялық* ДҚқатаң түрде белгіленген пішінде, қысқа түрде ұсынылатын деректерден тұрады. Мұндай ДҚ қағаз картотекаларының баламалары, мысалы, кітапханалық каталогтың немесе видеотеканың картотекасы. ДҚ басқа түрі - *құжаттамалық*. Мұнда балама ретінде құжаттардың мұрағаттары, мысалы, сот ісінің мұрағаты, тарихи құжаттар мұрағаты болады. Бұдан әрі біз тек фактографиялық ДҚ қарастырамыз.

Деректерді сақтау тәсілі бойынша жіктеу дерекқорды орталықтандырылғандарға және үлестірілгендерге бөледі. *Орталықтандырылған ДҚ* -да барлық ақпарат бір компьютерде сақталады. Бұл автономды ДК немесе пайдаланушы-клиенттер қолдана алатын желі сервері болуы мүмкін. *Үлестірілген ДҚ* оқшауланған және ғаламдық компьютерлік желілерде қолданылады. Мұндай жағдайда қордың түрлі бөліктері түрлі компьютерлерде сақталады.

Дерекқорды жіктеудің үшінші нышаны – деректерді ұйымдастыру үлгісі бойынша. *Деректердің құрылымдық үлгісінің* негізгі үш түрі бар: иерархиялық, желілік және кестелік. Сәйкесінше, құрылым нышаны бойынша ДҚ *иерархиялық, желілік* және *реляциялық (кестелік)* болып бөлінеді. Құрылымдық ДҚ-ға балама ретінде негізінде ОББ ұғымдары: класстар, объектілер, тұқым қуалаушылық, инкапсуляция және т.б. жататын *объектілі-бағдарланған ДҚ* болып табылады.

Біздің оқу құралда реляциялық ДҚ толығырақ тоқталамыз. 1970 жылдары американдық ғалым Э.Кодд деректердің кез келген құрылымын кестелік қалыпқа келтіруге болады деген ережеге негіз-

делген, реляциялық ДҚ теориясын құрды.

Деректердің реляциялық үлгісі. Реляциялық ДҚ негізгі ақпараттық бірлігі – кесте. Сәйкесінше, реляциялық ДҚ *деректердің кестелік үлгісін* қолданады. Дерекқор бір кестеден - *бір кестелі ДҚ*, немесе бір-бірімен өзара байланысқан көптеген кестелерден – *көп кестелі ДҚ* құрылуы мүмкін.

Кестенің құрылымдық құрамдаушысы ретінде жазбалар мен өрістер болады (4.1-сурет).

Әрбір *жазбада* жүйе құрайтын (тақырыптық сала) объектілердің жеке данасы жайлы: кітапханадағы бір кітап, кәсіпорынның бір қызметкері және т.с.с. ақпарат болады.

Ал әрбір *өріс* - бұл объектінің белгілі бір сипаты (қасиеті, нышаны): кітаптың аты, кітаптың авторы, қызметкердің аты-жөні, туған жылы және т.с.с Кестенің өрістері бір-бірінен айырмашылықтары бар атауларға болады. Кестеде бір-біріне ұқсас жазбалар болмайды.

Реляциялық ДҚ әрбір кестесі үшін жазбаны бір мәнді анықтайтын өріс немесе өрістер жиынтығы - *негізгі кілт* анықталуы тиіс. Басқаша айтқанда, негізгі кілттің мәні түрлі жазбаларда қайталанбауы тиіс. Мысалы, кітапханалық ДҚ- да мұндай кілтпен басқа кітаптарда сәйкес келмейтін кітаптың түгендеу нөмірі таңдалуы мүмкін.

Кесте құрылымын жолдық көрсету үшін келесі нысан енгізіледі:

КЕСТЕНІҢ_АТЫ (ӨРІСТІҢ АТЫ 1, ӨРІСТІҢ АТЫ_2, ..., ӨРІСТІҢ АТЫ_N)

Негізгі кілттен тұратын өрістердің асты сызылады.

Реляциялық ДҚ теориясында кесте *қатынас* деп аталады. Ағылшынша *relation*— қатынас. Осыдан «реляциялық дерекқор» атауының шығуы белгілі болды. Сондықтан КЕСТЕ_АТЫ — бұл қатынастың аты. Қатынастардың мысалдары:

КІТАПХАНА (ТҮГ. НӨМІРІ, АВТОРЫ, АТАУЫ, БАС._ЖЫЛЫ, БАСПА).

4.1.-кесте деректердің кестелік үлгісі

Жазба	1 өріс	2 өріс	3 өріс	...
1				
2				
3				
...				

АУРУХАНА (ПАЛАТА, ОРЫН НӨМІРІ, НАУҚАС, ТҮСКЕН_ КҮНІ, ДИАГНОЗЫ, АЛҒАШҚЫ)

Кестенің әрбір өрісінде белгілі бір *тип* болады. Типпен өрістің екі қасиеті байланысты:қабылдай алатын мәндердің жиыны және онымен орындауға болатын амалдар жиыны. ДҚ өрістері үшін төрт негізгі тип бар: таңбалық, сандық, логикалық және күн. «Кітапхана» және «Аурухана» кестелерінің өрістері үшін келесі типтер белгіленуі мүмкін:

Таңбалық тип: АВТОР, АТАУЫ, БАСПА, НАУҚАС, ДИАГНОЗЫ.
Сандық тип: ТҮГ. НӨМІРІ, БАС. ЖЫЛЫ, ПАЛАТАСЫ, ОРЫН НӨМІРІ.
Күні: ТҮСКЕН КҮНІ
Логикалық: АЛҒАШҚЫ

Соңғы жағдайда АЛҒАШҚЫ өрісі науқастың берілген диагнозбен ауруханаға алғаш рет ремесе қайта түскендігін білдіреді. Бұл өрістің мәнінде Те записи, где значение этого поля равно True (АҚИҚАТ) тең жазбасы болса, алғашқы науқастарға, ал False (ЖАЛҒАН) мәні қайта түскен науқасты білдіреді. Осылайша, логикалық типтегі өріс тек екі мәнді ғана қабылдай алады.

«Аурухана» кестесінде - екі өрістен: ПАЛАТА және ОРЫН НӨМІРІ тұратын - *құрама кілт* қолданылады. Тек олардың үйлесуі түрлі жазбаларда қайталанбайды (науқастардың аты-жөндері бірдей болулары мүмкін ғой).

Дерекқорды басқару жүйелері. Дерекқормен жұмыс жасауға арналған бағдарламалық қамсыздандыру дерекқорды басқару жүйесі деп аталады.

Құрылатын қорлардың құрылымына қарай иерархиялық, желілік, реляциялық, объектілі-бағдарланған ДҚБЖ бар. ДҚБЖ жіктеудің басқа белгісі – атқарылатын функциялары бойынша. Жартылай функционалды ДҚБЖ келесі әрекеттерді атқаруға арналған:

- ДҚ құрылымын құру;
- ДҚ ақпаратпен толтыру;
- құрылымды және ДҚ мазмұнын өзгерту (редакциялау);
- ДҚ ақпаратты іздестіру;
- деректерді іріктеу;
- ДҚ қорғау;
- ДҚ тұтастығын тексеру.

Жартылай функционалды ДҚБЖ мыналар жатады: MSAccess, MSFoxPro, Paradox және т.б. *ДҚ серверлері* және *ДҚ клиенттері* бағдарламалары функциялардың шағын жиынтығын орындайды.

4.5. ДЕРЕКТЕРДІҢ РЕЛЯЦИЯЛЫҚ ҮЛГІСІН ҚАЛЫПҚА КЕЛТІРУ

Енді реляциялық типтегі деректер үлгісін құру тәсілдеріне көшейік.

Деректердің реляциялық үлгісі – бұл өзара байланысқан көптеген қатынастар. Реляциялық үлгінің қарапайым түрі – бір қатынас. Дерекқорда – бір кесте. Бір кестелі дерекқорлармен информатиканың мектеп курсында таныстыңдар. Енді біздің танысатын тақырыбыз – көп кестелі ДҚ.

Деректерді сақтауды көп кестелі ұйымдастыруына әкелетін себептерді қарастырайық.

4.4.-суретте бейнеленген инфологиялық үлгіге толық түрде сәйкес келетін деректер үлгісін құруға біз біртіндеп, бірнеше кезең арқылы келеміз.

Бірінші кезеңде алдымызға бір жекелеген сыныптағы оқушылардың үлгерімі туралы мәліметтерден тұратын ДҚ құру мақсатын қояйық. Бізді оқушылардың әрбір тоқсан қорытындысы бойынша алатын бағалары, сондай-ақ барлық оқу пәндері бойынша алған жылдық бағалары қызықтыратын болады. Басқаша айтқан, ДҚ-ға сыныптың барлық оқушыларының бүкіл оқу жылына берілген үлгерім табельдерін енгізу керек. Бұдан басқа, ДҚ-да оқушылардың туған күндері мен үйлерінің мекенжайларын са сақтау қажет. Ер балаларды қыздардан айыру үшін әрбір оқушының жынысын көрсету керек.

Барлық аталған деректерді алатын қатынас келесі түрде беріледі:

ҮЛГЕРІМ (ТЕГІ, АТЫ, ПӘН, ЖЫНЫСЫ, ТУҒАН_КҮНІ_ МЕКЕН-ЖАЙЫ, 1_ТОҚС, 2_ТОҚС, 3_ТОҚС 4_ТОҚС, ЖЫЛ)

Бұл қатынаста кілт үш өрістен тұрады: ТЕГІ, АТЫ, ПӘН. Мұндай кілт *құрама* деп аталады.

Деректерді мұндай күйде сақтаудың айқын кемшілігі олардың артықтығы болады. *Артықтық* ұғымы - бір деректерді бірнеше рет қайталауды білдірді.

Әрбір оқушының ТЕГІ, АТЫ, ЖЫНЫСЫ, ТУҒАН_КҮНІ МЕКЕНЖАЙЫ өрістерінің мәндері түрлі пәндерге қатысты жазбаларда

қайталана береді. Бұл компьютер жадының орынсыз шығындалуына әкеледі (деректердің артықтығы). Бұдан басқа, қайталанатын өрістердің мәндерін қанда да бір жолдарға енгізгенде қателер жіберілуі мүмкін болады. Мысалы, бір мекенжай бірнше жерде түрлі жазылған. Мұндай жағдай *деректердің қарама-қайшылығы* деп аталады: бірдей деректер әр түрлі берілген.

Бұл мәселенеі шешу үшін берілген қатынасты екіге, яғни бір кестелі үлгіден екі кестелі үлгіге ауысу қажет болады. Бірінші кестені «Оқушылар» деп атаймыз. Онда оқушының аты-жөні, жынысы және мекенжайы сақталады. Бұл тізімдегі әрбір оқушыға өз нөмірі сәйкес қойылады (журналдағы нөмірі).

Міне. Осы нөмір кілттің міндетін атқаратын болады.

Екінші кестені «Үлгерім» деп атайық. Онда оқушыларды бірінші кестеде анықталған нөмірлері бойынша сәйкестендіруге болады. Оқушының аты-жөнін нөмірге ауыстыру жадының шығыны айтарлықтай қысқартады. «Үлгерім» кестесіне ПӘН өрісі және алынған бағалар туралы мәліметтер енгізіледі. ОҚУШЫНЫҢ НӨМІРІ және ПӘН өрістері құрама кілт құрайды. Нәтижесінде деректер үлгісі келесі екі қатынас түрінде ұсынылады:

ОҚУШЫЛАР (ОҚУШЫН_НӨМІРІ, ТЕГІ, АТЫ, ЖЫНЫСЫ, МЕКЕНЖАЙЫ)

ҮЛГЕРІМ (ОҚ_НӨМІРІ_ПӘН, 1_ТОҚС, 2_ТОҚС, 3_ТОҚС, 4_ ТОҚС, ЖЫЛ)

Бұл қатынастардың арасындағы байланыс «бірден -көпке» типіне ие болады. Ол ОҚ_НӨМІРІ (оқушының нөмірі)жалпы өрісі арқылы жүзеге асырлады. «Оқушылар» кестесінде бұл өріс негізгі кілт болып табылады. «Үлгерім» кестесінде ол құрама кілтке кіреді. Сәйкесінше, бұл өрістің бірінші кестедегі нақты мәні тек бір жазбада ғана болуы мүмкін, ал екіншісінде – жазбалардың көпшілігінде болады.

Біздің атқарған жұмысымыз деректерді *қалыпқа келтіру* деп аталады. Алынған деректердің екі кестелі құрылымы қалыпқа келтірілген құрылым болады. Қалыпқа келтірудің басты мақсаты – деректердің артықтығынан арылу. Нәтижесінде артықтығы жоқ ДҚ *әрбір фактіні бір данада* ғана сақтауы тиіс. Әрбір оқушы үшін оның: «тегі», «аты», «жынысы», «туған күні», «мекенжайы» атрибуттары қорға бір-ақ рет енгізіледі. Егер қандай да бір атрибуттардың мәндері өзгеретін болса, оларды дереу түзетуге болады. Мысалы, егер оқушының мекенжайы өзгерсе, онда деректер құрылымының бірінші нұсқасында оны

бірнше рет қайта жазу қажет болатын еді. Соңғы нұсқасында оны тек бір-ақ рет қана енгізеді.

Қалыпқа келтірудің мәні белгілі бір тақырыптық салаға жататын деректер үлгісін құру кезінде осы үлгіде берілуі тиіс сан түрлі *объектілер типтерін* (кейде – мәндерін деп айтады) айыра білу қажет. Біздің қарастырылған мысалда мұндай объектілер ретінде сауалнамалық деректерімен берілген ОҚУШЫЛАР және оқушылардың түрлі пәндер бойынша алған бағалары туралы мәліметтер бар ҮЛГЕРІМ (оқу қорытындысы) болады. Оқушылар туралы ақпарат «Оқушылар» кестесінде, оқу қорытындысы туралы ақпарат – «Үлгерім» кестесінде жинақталған.

Реляциялық ДҚ теориясында қатынастың «қалыпты нысаны» ұғымы қолднылады. Қатынастың барлық өрістері атомарлық болған жағдайда қатынас *бірінші қалыпты нысанда* болады. Атомарлық өріс әрі қарай бөлінбейді. Мысалы, «ТАӘ» бір өріске адамның тегін, атын, әкесінің атын біріктіру атомарлық қағидасын бұзады. Атомарлық ұғымы біркелкі. Мысалы, егер қосымшаларда көшенің, үйдің және пәтердің нөмірін жеке өңдеу қажет болмаса, онда мекенжайды құрамалаға бөлмеуге және оны атомарлық деп санауға болады.

Қатынас бірінші қалыпты нысанда болса және оның барлық негізгі емес өрістері тұтастай функционалды түрде негізгі кілтке тәуелді болса, қатынас *екінші қалыпты нысанда* болады. Басқаша айтқанда, негізгі емес өрістің мәні әрбір жазбада осы жазбадағы кілттің мәнімен тікелей байланысты болады. ОҚУШЫЛАР және ҮЛГЕРІМ қатынастары мұндай қасиетке ие болады. Берілген оқушының (кілт: ОҚ_НӨМІРІ) белгілі мекенжайы, тегі, туған күні және т.с.с. бар. Берілген оқушыда берілген пән бойынша (кілт: ОҚ_НӨМІРІ + ПӘН) тоқсан және жылдық бағалары бар.

Үшінші қалыпты нысанның талабы: екінші қалыпты нысанға қанағаттану және өрістерге қатысты кілттен тразитивті тәуелділікте болм С: А → С → В үшінші деңгейден тәуелділікті айтады. Біздің алынған қатынастарда транитивті тәуелділік жоқ. Мысалы, егер «Оқушы» қатынасында оқушы тұратын қаланың әкімшілік ауданын білдіретін АУДАН өрісі болса, онда транзитивті тәуелділік болуы мүмкін. Аудан тікелей мекенжаймен байланысты, сондықтан трназитивтілік келсідей беріледі:

ОҚ НӨМІРІ → МЕКЕН-ЖАЙ → АУДАН

Сөйтіп, біздің алынған деректердің екі кестелі үлгісі үшінші қа-

лыпты нысанның талаптарын қанағаттандырады.

4.6. MS ACCESS ДҚБЖ. КЕСТЕЛЕР ҚҰРУ

Жалпы сипаттамасы. Бағдарламашыға бағдарланған және соңғы пайдаланушыға бағдарланған ДҚБЖ болады. ДҚ-мен атқарылатын кез келген әрекеттер ЭЕМ-да бағдарламалардың көмегімен жүргізіледі.

Бағдарламашыларға бағдарланған дерекқорды басқару жүйелері, іс жүзінде өзінің мамандандырылған тілімен берілген бағдарламалау жүйелері болып табылады, олардың ортасында бағдарламашылар дерекқорларды өңдеу бағдарламаларын жасайды. Содан соң осы бағдарламалармен соңғы пайдаланушылар жұмыс істейді. Мұндай типтегі ДҚБЖ FoxPro, Paradox және басқалары жатады.

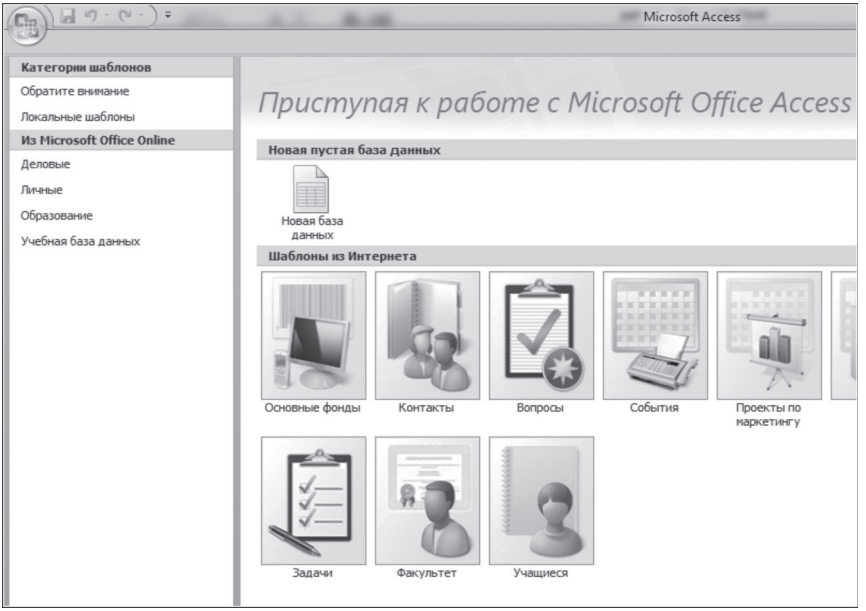
Бұдан басқа, бағдарламалаудың заманауи әмбебап тілдеріне ДҚБЖ кейбір функциялары кіріктірілген. Ондай тілдердің мысалы ретінде Delphi тілі болады.

Microsoft Access (MS Access) ДҚБЖ соңғы пайдаланушыға бағдарланған жүйелерге жатады. Ол пайдаланушыға бағдарламалауға жүгінбей-ақ, ДҚ негізгі амалдарды: құру, түзету, деректермен айлы-шарғылар жүргізулі оңай жүргізуге мүмкіндік береді. MS Access Windows операциялық ортада жұмыс істейді, автономды ДК де, сонымен қатар жергілікті компьютердлік желіде де қолданыла береді. Access көмегімен жеке ДҚ (кейде – «стол үстіндегі» деп те айтылады), сонымен қатар шағын деректер көлемімен ұйымның ДҚ құрылады және пайдаланылады. Ірі өнеркәсіптік ақпараттық жүйелерді құру үшін MS Access жарамайды.

MS Access негізгі объектілері ретінде кестелер, нысандар, сұраныстар, есептер, макростар, модульдер, сызбалар болады.

Кесте — бұл объектінің басты типі. Объектінің қалған барлық түрлері кестеден шығатын болады. Кестені құрайтын деректер элементтері – бұл жазбалар мен өрістер. Кесте элементтерінің қасиеттері типтермен, өріс пішімдерімен және өзге де басқа параметрлермен анықталады.

Нысан — бұл көмекші объект, іс жүзінде оны қолданбауға да болады. Нысандар пайдаланушы кестедегі деректерді көру, енгізу және түзетулер енгізу кезінде ыңғайлықты арттыру үшін жасалады.



4.5. –сурет. MSAccess бастапқы беті

Сұраныс — пайдаланушының деректерді іздеу, жазбаларды қосу, өшіру және жаңарту кезінде ДҚБЖ-не жүгіну нәтижесі. Деректерді іздестіру (іріктеу) нәтижесі кесте түрінде беріледі. «Сұраныс» терминімен сондай-ақ ДҚБЖ –не жүгіну командасын айтады.

Есеп — кестелер мен сұраныстарда болатын, ақпараттың негізінде құрастырылған, баспаға шығаруға арналған құжат.

Макростар — деректерді бағдарламалық басқару құралдары және әдетте қайталанатыцн амалдарды автоматтандыру үшін қолданылады.

Модульдерді VisualBasicforApplication (VBA) тілінде жазылатын, оқиғаларды өңдеу процедуралары деп те атайды.

Сызба — көп кестелі дерекқорда байланыс құрылымын сипаттау.

Жұмыстың басталуы. Бағдарламаны іске қосқаннан кейін экранда MS Access бастапқы бет ашылады (4.5-сурет). Мұнда ақпараттық жүйелердің кейбір үлгілік нұсқалары үшін ДҚ дайын нысандарын пайдалану ұсынылады.

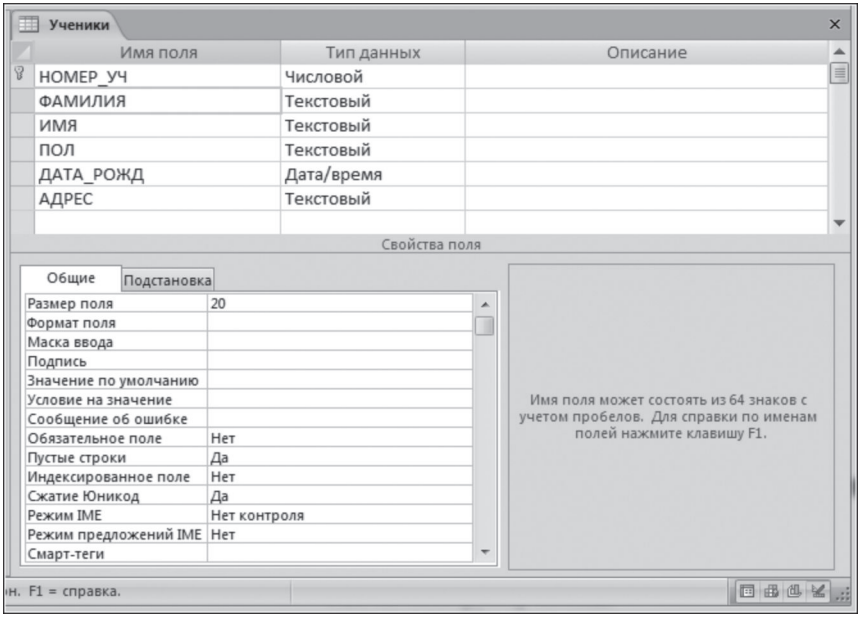
Келесі мысалдарда біз нысандарды қолданбаймыз. ДҚ құру оның өзі сақталатын файлды ашудан басталады. Ол үшін, MS Access бастапқы бетті ашқан соң. *Жаңа дерекқор құру* командасын орындау керек.Жүйе ДҚ сақталатын файлдың атын және дисктегі файлға барар жолды көрсетуді сұрайды. Біздің құратын ДҚ үшін файлдың атын

«Мектеп» деп көрсетейік.

Нәтижесінде экранда «Мектеп: дерекқор» тақырыбымен негізгі терезе ашылады. ДҚ құру бойынша одан арғы жұмыс екі кезеңнен тұралы: кестелер құрылымын суреттеу және кестелерге деректерді енгізу.

«Оқушылар» кестесін құру. Кесте құрылымын сипаттау – бұл: барлық өрістердің атын, сондай-ақ әрбір өрістің типін және қасиеттерін көрсету;негізгі кілтті тағайындау. Кестемен жұмыс істеу тәртіптемесінде *Құру* командасы беріледі. Ұсынылатын кестелерді құру тізімінен *Конструктор* таңдалады. Экранда кестелер конструкторы терезесі ашылады. «Оқушылар» кестесі үшін толтырылған Конструктор терезесі 4.6-суретте берілген.

Өрістердің аттары «Өрістің аты» бағанында, оларға сәйкес келетін типтер – «Деректер типтері» бағанында көрстеіледі. «Сипаттау» бағанын толтыру міндетті емес. Конструктор терезесінің төменгі жарты бөлігінде «Өрістің қасиеттері» кестесі бар. Мұнда өрістің өлшемі, өрістің пішімі және кейбір басқа да қасиеттер көрсетіледі. Әрбір параметрдің мағынасы түсініктеме мәтінмен түсіндіріледі. Бұдан басқа, әрдайым«F1» батырмасы бойынша анықтамаға жүгінуге болады.



4.6. –сурет. «Оқушылар» кестесінің конструкторы

4.6-суретте ТЕГІ өрісінің қасиеттері бейнеленге. Мәтіндік өрістің негізгі қасиеті ретінде оның ұзындығы болады. Ұзындықтың ең шекті мәні -255 таңба. Берілген жағдайда ұзындық - 20 таңба таңдалған. Бір жағынан, мәтіндік өрістің ұзындығын оған осы өрістің кез келген мүмкін болатын саны сиятындай беру керек, ал екінші жағынан, артық ұзындық – бұл шегі болатын компьютер жадындағы артық шығын екенін де ескерген жөн.

АТАУЫ өрісінің де мәтіндік типі бар. Ол үшін таңдалған ұзындық – 15 таңба.ОҚ_НӨМІРІ өрісіне тұтас санның типі берілген.

Негізгі кілтті тағайындау келесідей жүргізіледі: көрсеткіш ОҚ_НӨМІРІ негізгі өріске орнатылады және аспаптар панелінде таңбаланған *Негізгі өріс* командасы атқарылады.

Бұдан әрі әрбір кестенің құрылымы туралы ақпаратты кестелік түрде беретін боламыз. «Оқушылар» кестессінің мысалында ол 4.2.-кестеде көрсетілген түрде беірледі:

Барлық әрекеттерді орындағаннан кейін «Барлық кестелер» терезесінде «Оқушылар» кестенің атауы пайда болады.

Енді деректерді «Оқушылар» кестесіне енгізу ұйымдастырылады. Деректерді тікелей кестенің бланкіне немесе нысанды қолдану арқылы енгізуге болады. Енгізуге арналған нысанды және кестені көруді қолдану онда өрістер тым көп болғанда және ашық тұрғанда енгізілген жазба экранға сыймай қалған жағдайда жүзеге асырылады. «Оқушылар» кестесінің көлемі шағын, сондықтан нысаннық қажеті жоқ.

4.2-кесте. Оқушылар

Өрістің атауы	Өрістің типі	Ұзындығы (пішімі)
ОҚ НӨМІРІ	Сандық	Тұтас
ТЕГІ	Мәтіндік	20
АТЫ	Мәтіндік	15
ЖЫНЫСЫ	Мәтіндік	1
ТУҒАН КҮНІ	Күні/уақыты	—
МЕКЕНЖАЙЫ	Мәтіндік	30

Ученики

НОМЕР_УЧ	ФАМИЛИЯ	ИМЯ	ПОЛ	ДАТА_РОЖД	АДРЕС
1	Антонов	Кирилл	м	13.10.1999	Садовая 5, кв.56
2	Веткина	Ирина	ж	21.01.2000	Островского 3, кв.4
3	Вяткин	Иван	м	06.09.1999	Садовая 3, кв.14
4	Волочкова	Настя	ж	03.01.2001	Ладыгина 43, кв.23
5	Волегов	Кирилл	м	11.06.2000	Садовая 3, кв.4
6	Гилев	Валерий	м	04.11.1999	Лебедева 43, кв.4
7	Ежова	Марина	ж	23.02.2000	Малкова 76, кв.81
8	Зимина	Елена	ж	15.04.2001	Малкова 76, кв.2
9	Игошина	Наталья	ж	07.07.2000	Сибирская 8, кв.65
10	Ильиных	Михаил	м	31.12.1999	Лебедева 14, кв.3

4.7. –сурет. «Оқушылар» кестесі

«Оқушылар» кестесіне енгізуді бастау үшін кестенің атауын белгілеу керек, содан соң *Ашу* командасын орындау керек. Экранда тақырыбы мен бос жолы бар кестенің бланкі пайда болады. Одан әрі кестені толтыруды бастау керек. «Оқушылар» кестесінің толтырылған бірінші он жолы 4.7 суретте берілген.

Енгізілген ақпаратты сақтау үшін *Сақтау* командасын орындау керек.

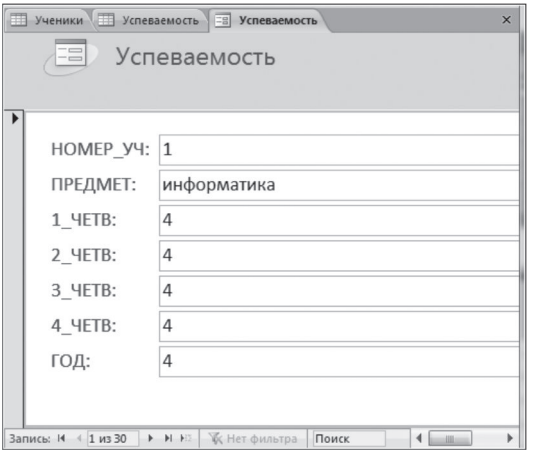
«Үлгерім» кестесін құру. Тағы бір «Үлгерім» қатынасын қайта жазайық:

ҮЛГЕРІМ (ОҚ_НӨМІРІ, ПӘН, 1_ТОҚС, 2_ТОҚС, 3_ТОҚС, 4_ ТОҚС, ЖЫЛ)

Бұдан әрі Конструктордың көмегімен «Үлгерім» кестесінің құрылымы сипатталады (4.3кесте).

4.3. –кесте. Үлгерім

Өрістің атауы	Өрістің типі	Ұзындығы (пішімі)
ОҚ НӨМІРІ	Сандық	Тұтас
ПӘН	Мәтіндік	30
1_ТОҚС	Сандық	Тұтас
2_ ТОҚС	Сандық	Тұтас
3_ ТОҚС	Сандық	Тұтас
4_ ТОҚС	Сандық	Тұтас
ЖЫЛ	Сандық	Тұтас



4.8. сурет. Кестені енгізуге, көруге және түзетуге арналған нысан

«Үлгерім» кестесін толтыру үшін деректерді нысан арқылы енгізу әдісін қолданамыз. Бас мәзірдің панеліндегі «Құру» қосымша парақта Нысандар конструкторы командасын таңдау қажет. Экранда сипатталған кесте құрылымына сәйкес келетін стандартты түрдегі нысан ашылады.

Ученики | Успеваемость

НОМЕР_УЧ	ПРЕДМЕТ	1_ЧЕТВ	2_ЧЕТВ	3_ЧЕТВ	4_ЧЕТВ	ГОД
1	информатика	4	4	4	4	4
1	история	4	4	4	4	4
1	математика	5	5	5	5	5
2	информатика	4	5	4	4	4
2	история	3	3	3	3	3
2	математика	4	5	5	5	5
3	информатика	3	3	3	3	3
3	история	5	5	5	5	5
3	математика	5	4	4	5	5
4	информатика	3	3	4	4	4
4	история	4	4	3	3	3
4	математика	4	4	4	4	4
5	информатика	5	5	5	5	5
5	история	5	5	5	5	5
5	математика	5	5	5	5	5

4.9.-сурет. «Үлгерім» кестесі

Енді нысан арқылы жүйелі түрде жазбаларды кестеге енгізу керек. Кейін нысанды кестені толықтыру және жазбаларға түзету енгізу үшін қолданған қолайды. «Үлгерім» кестесіне бірінші жазбалардың өріс мәндерін енгізу 4.8-суретте берілген.

Сынып тізімінің алғашқы бес оқушысына қатысты толтырылған кестенің үзіндісі 4.9-суретте көрсетілген. Бейнелерді қысқарту үшін оқылатын үш пәнмен: информатика, тарих, математика ғана шектелейік.

4.7. КЕСТЕДЕН ДЕРЕКТЕРДІ ІРІКТЕУГЕ БЕРІЛГЕН СҰРАНЫСТАР

ДҚ құрылған соң, оны ақпараттық анықтамалық ретінде пайдалануға болады. Бұл әрбір ақпараттық жүйенің негізгі мақсаты болып табылады.

Дерекқорда сақталатын ақпараттармен атқарылатын әрекеттер *айла-шарғы жасалған деректер* деп аталады.Оларға кейбір шарттар бойынша деректерді іріктеу, деректерді сұрыптау, деректерді жаңарту, жою және қосу жатады. Бұл әрекеттерді орындау сұраныстың көмегімен жүргізіледі.

Сұраныс — бұл деректермен айлы-шарғы жасаудың белгілі бір түрін орындауға берілген команда.

Түрлі ДҚБЖ-де көбіне сұраныстарды сипаттаудың негізгі екі тілі қолданылады:

1) QBE (Query by Example) тілі — үлгі бойынша сұраныстар тілі;

2) SQL (Structured Query Language) — құрылымдандырылған сұраныс тілі.

QBE типтегі тілдер ДҚ –ға сұранысты арнайы сұраныс нысандарын толтыру жолымен құрастыруға мүмкіндік береді. Мұндай әдіс тамаша көрнекілікті қамтамасыз етеді және сұраныс бойынша амалдарды орындау алгоритмін көрсетуді талап етпейді. Тек күтілетін нәтиженің үлгісі – сұранысқа жауап ретінде алынатын кестелер сипатталады.

SQL тілі кестелермен (құру, өшіру, құрылымын өзгерту) және осы кестелерде сақталатын деректермен (іріктеу, өзгерту, қосу, өшіру, сұрыптау) орындалатын амалдарды орындауға арналған.

MS Access ДҚБЖ-де SQL тілі сұраныстарды сипаттауда негізгі тіл болып саналады. QBE тілі сұраныс Конструкторы ұсынатын сұра-

ныс нысандары арқылы жүзеге асырылады. Конструктордың көмегімен сипатталған сұраныс SQL тілінде командаға жүгінетіндігін және сұранысты орындау осы тілде жүргізілетіні есте ұстау керек.

SQL командасымен тым жиі қолданылатын команда – іріктеу командасы болады. Бір кестелі ДҚ-дан іріктеуді жүзеге асыру үшін іріктеудің командалық пішімі келесідей болады:

SELECT <шығарылатын өрістер тізімі> FROM <кестенің аты>
WHERE <таңдау шарты> ORDER BY <сұрыптау кілті>[DESC]

Бұл команданың құрамдастарының барлығы бірдей міндетті болмайды. Таңдау шарты және сұрыптау параметрлерінің жоқ болуы мүмкін. Бұдан басқа, сұрыптау кілттері бірнше болуы мүмкін. Ндай жағдайда олар басымдылық тәртібі бойынша жазылады: алғашқы, екінші және т.с.с. Сұрыптау кілт мәнінің артуы және кемуі бойынша жүргізілуі де ықтимал.DECS опциясы болмаған жағдайда сұрыптау «артуы бойынша», ал болғанда – «кемуі бойынша» жүзеге асырылады.

4.1. мысал. «Оқушылар»кестесін қолдана отырып, 2000 ж. дейін туған оқушылардың аты-жөні және мекенжайы бар кестені алу. Кестені алфавит бойынша тегінен бастап сұрыптау керек

Сұраныс SQL тілінде мынадай түрде беріледі:

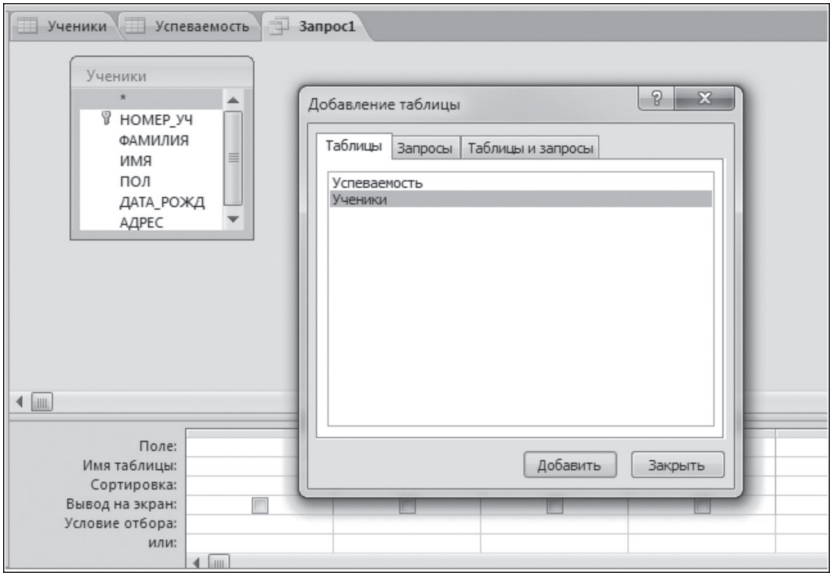
```
SELECT Оқушылар.ТЕГІ, Оқушылар.ИМЯ, Оқушылар.
    МЕКЕНЖАЙЫ
FROM Оқушылар
WHERE (((Оқушылар. ТУҒАН КҮНІ)<#1/1/2000#))
ORDER BY Оқушылар. ТЕГІ;
```

Қорытынды кестеге үш өріс шығарылады: оқушының тегі, аты және мекенжайы. Сұраныстарда *өрістердің құрамдастырылған атаулары* көрсетіледі:

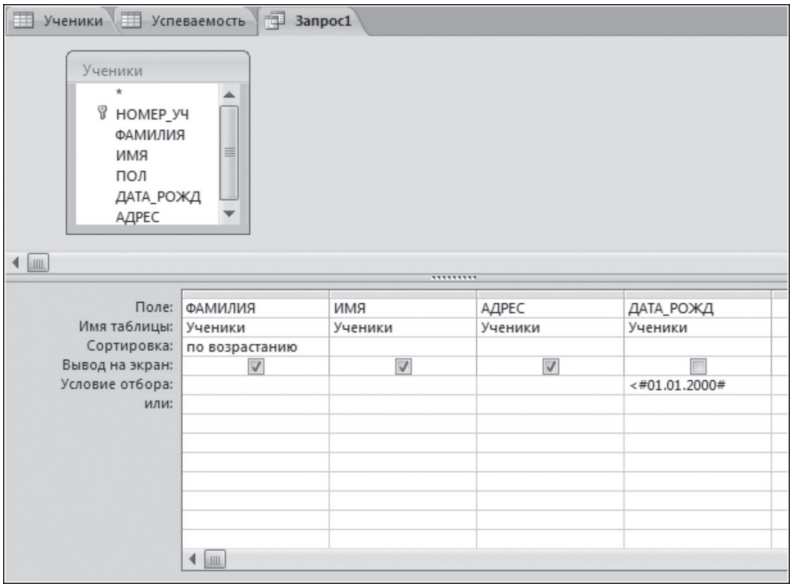
<кесте атауы>.<өріс атауы>

Таңдау шарты кестеден тек 2000 ж. 1 қаңтарға дейін туған (бұрын) оқушыларды ғана іздеу керктігін белгілейді. Қорытынды кестеде оқушылардың аты-жөндерінің алфавиттік түрінде ғана сұрыпталады.

Осы сұранысты Access сұраныстар Конструкторын қолданып, құруға болады. Конструкторды іске қосқан соң «Кесте қосу» терезесі ашылады. Содан соң «Оқушылар» кестесін белгілеп, *Қосу* командасын орындау қажет. (4.10-сурет).



4.10.-сурет. Кестені сұраныстар Конструкторына қосу



4.11.-сурет. 1 сұраныс конструкторы

ФАМИЛИЯ	ИМЯ	АДРЕС
Антонов	Кирилл	Садовая 5, кв.56
Вяткин	Иван	Садовая 3, кв.14
Гилев	Валерий	Лебедева 43, кв.4
Ильиных	Михаил	Лебедева 14, кв.3

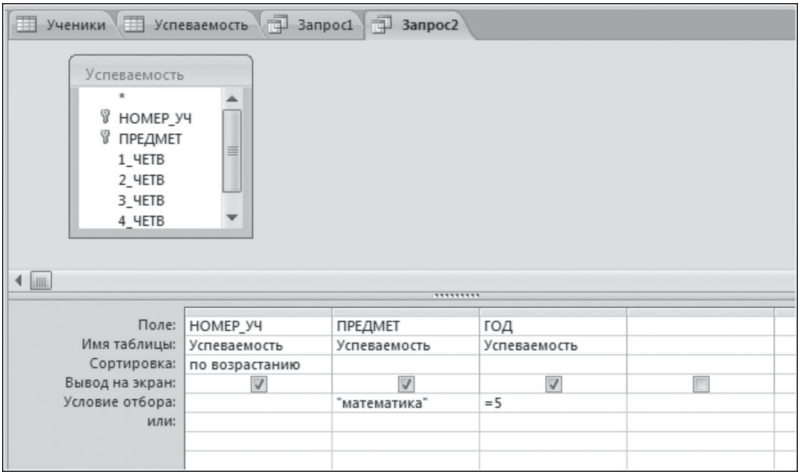
4.12.-сурет. 1 сұранысты орындау нәтижесі

Сұраныстар конструкторы кесте түрінде беріледі (4.11сурет), оның бірінші жолында сұранысты қлыптастыруға қатысатын өрістер көрсетіледі. Екінші жолда тиісті өріс алынатын, кестенің атауы болады. Үшінші жолда сұрыптау нышандары орналасады. Бесінші жолда жалаушалармен сұранысты орындау кезінде берілген өрісті экранға шығару қасиеті белгіленеді. Келесі жолдарда іріктеу шарты құрылады. Конструкторды толтыру аяқталғаннан кейін *Орындау* командасын атқару керек.

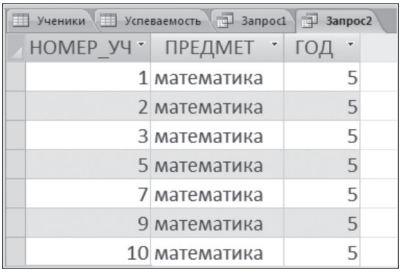
Бұл сұранысты орындау нәтижесі 4.12 суретте көрсетілген.

Сұранысты сақтау қажет. Сақтау кезінде диалогтік терезе ашылады, онда сұраныс үшін атау беріледі. Сұраныстың атауын стандартты түрде қалдырайық.

4.2. мысал. «Үлгерім» кестесін қолдана отырып,жыл бойы математикадан бестік алған оқушылардың нөмірін анықтау керек.



4.13-сурет. 2 сұраныстың Конструкторы



Ученики | Успеваемость | Запрос1 | Запрос2

НОМЕР_УЧ	ПРЕДМЕТ	ГОД
1	математика	5
2	математика	5
3	математика	5
5	математика	5
7	математика	5
9	математика	5
10	математика	5

4.14-сурет.2 сұранысты орындау нәтижесі

```
SELECT Үлгерім. ОҚ_НӨМІРІ, Үлгерім. ПӘН, Үлгерім. ЖЫЛ
    FROM Үлгерім WHERE (((Үлгерім. ОҚ НӨМІРІ)=
    "математика") AND ((Үлгерім.ЖЫЛ)=5))
ORDER BY Үлгерім. ОҚ НӨМІРІ;
```

Бұл командада таңдаудың екі шартын: ПӘН= "математика" ЖӘНЕ ЖЫЛ=5. өзара байланыстыратын AND (ЖӘНЕ) логикалық амалы қолданылады. Таңдаудың мұндай шартын күрделі шарт деп атаймыз. Сұраныс бойынша ақпарат кестенің осы екі шарт бір уақытта орындалатын жазбаларынан таңдалып алынады.

Күрделі шарттың жазу синтаксисіне назар аударыңдар: оның барлығы дөңгелек жақшаға алынады және AND басқа амалмен қосылатын әрбір жеке шарт сондай-ақ жақшаға алынады.

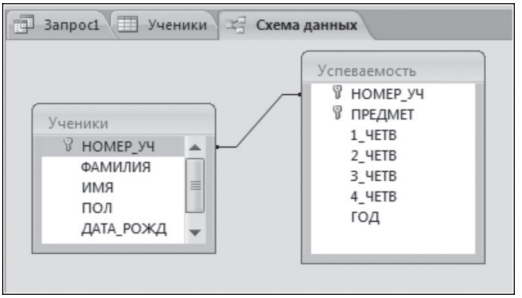
Конструкторды қолданған соң бұл сұраныс 4.13-суретте көрсетілгендей құрылады.

Сұранысты орындау нәтижесінде 4.14-суретте берілген кесте пайда болады.

4.8. КӨП КЕСТЕЛІ ДЕРЕККОРҒА СҰРАНЫС

4.14 суретті қараңдар. Бірінші бағанда оқушылардың нөмірлері емес, олардың аты-жөндері орналасса, бірсыпыра көрнекілеу болар еді. Бірақ нөмірлер мен фамилиялардың арасындағы байланыс «Оқушылар »кестесінде берілген.

Яғни, бізге деректер екі кестеден бір уақытта қолданылатын сұраныс құру керек.



4.15. –сурет. Екі кестелі ДҚ схемасы

4.5-бөлімде айтылғандай, бұл екі кесте бір-бірімен ОҚ_НӨМІРІ ортақ негізгі өріс арқылы байланысқан. Кестеле арасындағы байланысытың графикалық *бейнесі дерекқордың схемасы* деп аталады. Схеманы *Дерекқормен жұмыс жасау — Деректер* схемасы командасын беріп, экранға шығаруға болады. Біздің жағдайда 4.15-суретте берілген бейне көрсетіледі.

4.3. мысал. «Оқушы»және «Үлгерім» байланысқан кестелерді қолданып, жыл бойы математикадан бестік алған оқушылардың тегін және атын анықтау керек.

SQLкомандасы келесі түрде беріледі:

SELECT Оқушылар .ТЕГІ, Оқушылар.АТЫ, ҮЛГЕРІМ.ПӘН, ҮЛГЕРІМ. ЖЫЛ
FROM Оқушылар INNER JOIN ҮЛГЕРІМONОқушылар. ОҚ_НӨМІРІ = ҮЛГЕРІМ.ОҚ НӨМІРІ
WHERE (((ҮЛГЕРІМ.ПӘН)="математика") AND ((ҮЛГЕРІМ. ЖЫЛ)=5))
ORDER BY Оқушылар. ТЕГІ;

Бұл сұраныс бойынша қорытынды кестеге «Оқушылар» кестесінен екі өріс және «Үлгерім» кестесінен екі өріс қосылады. Мұнда сондай-ақ бұл екі кесте бір-бірімен ОҚ_НӨМІРІ (ON Оқушылар. ОҚ_НӨМІРІ = Үлгерім. ОҚ_НӨМІРІ) өрістері арқылы байланысқаны туралы ақпарат болады.

Сұранысты орындау нәтижесінде 4.16-суретте көрсетілген кесте алынады. Дәл осы сұраныс Конструкторда 4.17-суретте көрсетілген.

Запрос1 | Ученики | Схема данных | Запрос3

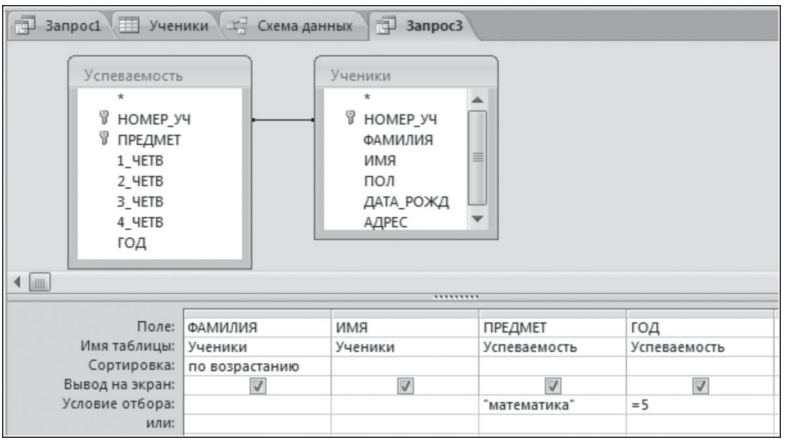
ФАМИЛИЯ	ИМЯ	ПРЕДМЕТ	ГОД
Антонов	Кирилл	математика	5
Веткина	Ирина	математика	5
Волегов	Кирилл	математика	5
Вяткин	Иван	математика	5
Ежова	Марина	математика	5
Игошина	Наталья	математика	5
Ильиных	Михаил	математика	5

4.16.-сурет. 3 сұраныстың орындаудың нәтижесі

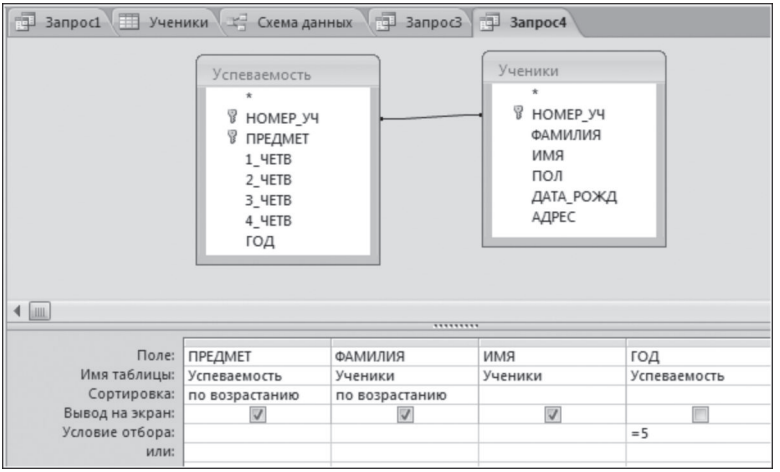
4. мысал. Қандай оқушылар және қандай пәндерден жылдық 5 деген баға алғандығы туралы мәліметтерді алу керек. Бұл мәліметтерді алфавит бойынша орналастырып, пәндер бойынша топтастыру керек.

SELECT Үлгерім ПӘН, Оқушылар.ТЕГІ, Оқушылар. АТЫ
FROM Оқушылар INNER JOIN Үлгерім ON Оқушылар. ОҚ НӨМІРІ = Үлгерім. ОҚ НӨМІРІ
WHERE (((Үлгерім.ГОД)=5))
ORDER BY Үлгерім. ПӘН, Оқушылар.ТЕГІ;

Дәл осы сұраныс Конструкторда 4.18суретте көрсетілген.
Сұранысты орындау нәтижесінде 4.19суретте көрсетілген кесте алынады.



4.17. – сурет. 3 сұраныстың конструкторы



4.18.-сурет. 4 сұраныстың конструкторы

Бұл сұраныста ПӘН өрісі сұрыптаудың *бастапқы кілті* ретінде, ал ТЕГІ өрісі – сұрыптаудың *екінші кілті* болғандықтан пәндер бойынша нәтижені топтастыруға қол жеткізілді.Мұндай жағдайда іріктеліп алатын жазбалар алдымен бастапқы кілт бойынша сұрыпталады. Егер бірнеше жазбаның бастапқы кілтінің мәндері сәйкес келетін болса, онда олардың тәртібі екінші кілт бойынша сұрыпталумен анықталады.

ПРЕДМЕТ	ФАМИЛИЯ	ИМЯ
информатика	Волегов	Кирилл
информатика	Ежова	Марина
информатика	Игошина	Наталья
история	Волегов	Кирилл
история	Вяткин	Иван
история	Ежова	Марина
история	Ильиных	Михаил
математика	Антонов	Кирилл
математика	Веткина	Ирина
математика	Волегов	Кирилл
математика	Вяткин	Иван
математика	Ежова	Марина
математика	Игошина	Наталья
математика	Ильиных	Михаил

4.19.-сурет. 4 сұранысты орындаудың нәтижесі

4.9. ЛОГИКАЛЫҚ ӨРНЕКТЕР ЖӘНЕ ІРІКТЕУ

Енді таңдауға сұраныс беру командасында іріктеу шарттарын тиянақтау ережелерін жан-жақты талқылаймыз.

Іріктеу шарттары — бұл ДҚ іріктелетін жазбалары үшін ақиқат болуы тиіс логикалық өрнек.

Логикалық өрнектер, элементтерімен сендер 1-тарауда танысқан, математикалық логика тілінде беріледі. Логиканың негізгі ұғымдарын есімізге түсірейік, оларды білу сендерге келешекте қажет болады.

1. Логикалық шама — бұл тек екі мәнді ғана - АҚИҚАТ (True) немесе ЛОЖЬ (False) қабылдайтын шама. Дерекқорларда логикалық типтің өрісі – бұл логикалық шама.

2. Логикалық өрнек — бұл ақиқат немесе жалған болатын пайымдау. Логикалық өрнек логикалық константалардан, логикалық айнымалылардан, логикалық амалдардан, логикалық қатынастардан тұрады.

3. Қатынас амалдары екі шаманың мәндерін салыстырады. Қатынас амалдарының белгілері: = (тең), <> (тең емес), > (артық), < (кем), >= (артық немесе кем), <= (кем немесе тең). Санлдық шамаларды салыстыруолардың арифметикалық мәнінде; таңбалық шамаларды салыстыру - таңбалар тәртібін ескере отырып, кодтау кестесіндежүргізіледі; «күн»және «уақыт» шамалары олардың уақыттағы жүйелігінің мәніне қарай салыстырылады.

4. Үш негізгі логикалық амал бар: терістеу -ЕМЕС (not), конъюнкция — ЖӘНЕ (and), дизъюнкция — НЕМЕСЕ (or). Оларды орындау тәртібі ақиқаттық кестесінде (4.4 кесте) бейнеленген. Мұндағы: АҚИҚАТ – А әрпімен, ЖАЛҒАН –Ж әрпімен белгіленген.

4.4.-кесте. Ақиқаттылық кестесі

А	В	А ЕМЕС	А ЖӘНЕ В	А НЕМЕСЕ В
И	И	Л	И	И
И	Л	Л	Л	И
Л	И	И	Л	И
Л	Л	И	Л	Л

5. Жоғары тұрушылыққа қарай кему бойынша логикалық амалдар келесі тәртіпте орналасады: ЕМЕС, ЖӘНЕ, НЕМЕСЕ. Логикалық өрнектерде амалдарды орындау реттілігіне әсер ету үшін дөңгелек жақшалар қолданылуы мүмкін.

Алдымен логикалық өрнектерді құрудың формалды мысалында – жазбаларды ДҚ-дан іріктеу шарттарында жаттығайық. A, B, C -сандық өріс, ал *R*1, *R*2 және т.б. — жазбалардың сәйкестендіргіші (кілті) болатын, бір кестелі ДҚ қарастырайық (4.5-кесте). Әрі қарай логикалық амалдардан және олардың нәтижелерінен, яғни осы шарттарды қанағаттандыратын жазбалардан тұратын, іріктеу шарттарының мысалдары берілген. Осы мысалдармен мұқият танысып,оларды түсінуге тырысыңдар.

Шарты	*Жауабы*
1) *A* = 1 ЖӘНЕ *B*= 2	:*R*1
2) *A* = 1НЕМЕСЕ *A* = 3	:*R*1, *R*2, *R*4, *R*5
3) *A* = 1 НЕМЕСЕ *B*= 2	:*R*1, *R*2, *R*3, *R*5
4) *A* = 1 НЕМЕСЕ *B*= 2 НЕМЕСЕ *C*= 3	:*R*1, *R*2, *R*3, *R*4, *R*5
5) *A* = 1 ЖӘНЕ *B*= 2 и *C* = 3	:*R*1
6) ЕМЕС *A* = 1	:*R*3, *R*4, *R*5

Реляциялық ДҚ-ға сұраныстар тілінің негізінде 1970 жылдары британдық ғалым Э.Кодд ұсынған, *реляциялық алгебраның* аппараты жатыр. Реляциялық алгебра логика алгебрасының дәл сол амалдарының теоретикалық-жиын интерпретациясын қолданады. ДҚ кестесінде жазбалардың жиынтығы жиын ретінде қарастырылады. ДҚ-ға сұраныс беру нәтижесі - сұраныс шарттарын қанағаттандыратын, жазбалардың кейбір қосалқы жиындары. Реляциялық алгебраның негізгі амалдары: біріктіру, қиылысу, айырма, кескін және іріктеу.

4.5. кесте. Бір кестелі ДҚ

Сәйкестендіргіш	A	B	C
R1	1	2	3
R2	1	3	1
R3	2	2	2
R4	3	3	3
R5	3	2	3

Соңғы үш амал ДҚ-ға сұраныс беру шарттарында қолданылмайды.

НЕМЕСЕ амалы әрбір екі шарттың (қатынастың) біреуін қанағаттандыратын жазбаларды бір іріктеуге біріктіреді. Реляциялық алгебра терминологиясында мұндай әрекет *бірігу амалына* сәйкес келеді. ЖӘНЕ амалының қызметі өзгеше: алдымен бірінші шартты қанағаттандыратын барлық жазбалар таңдалады, содан соң іріктелген жазбалардан екінші шартты қанағаттандыратындары таңдалып алынады. Реляциялық алгебра тілінде мұндай амал *қиылысу* деп аталады.

Келесі өрнектерде түрлі логикалық амалдар кездеседі, сондықтан оларды орындау кезінде амалдардың жоғары тұрушылығын ескерген жөн:

7) *A* = 1 ЖӘНЕ *B*= 2 НЕМЕСЕ *C* = 3	:*R*1, *R*4, *R*5
8) *A* = 1 НЕМЕСЕ *B*= 2 ЖӘНЕ *C* = 3	:*R*1, *R*2, *R*5
9) *A* ЕМЕС = 1 НЕМЕСЕ *B*= 2 ЖӘНЕ *C* = 3	:*R*1, *R*3, *R*4, *R*5
10) (*A* = 1 НЕМЕСЕ *B*= 2) ЖӘНЕ *C* = 3	:*R*1, *R*5

Және, соңында, бір өрістің мәндері екінші өрістің мәндерімен, сонымен қатар арифметикалық өрнектермен салыстырылатын мысалдар келтірейік:

11) *B*>= *A*	:*R*1, *R*2, *R*3, *R*4
12) *B*>= *A* ЖӘНЕ *B*>= *C*	:*R*2, *R*3, *R*4
13) *A* = *B* НЕМЕСЕ *A* = *C*	:*R*2, *R*3, *R*4, *R*5
14) *C*= *A* + *B*	:*R*1

MSAccess сұраныстар Конструкторында таңдау шартын ұсынудың кестелік әдісі қолданылады. Толығырақ осы әдісті талқылайық.

Конструктор кестесінің ұяшықтарында, «Іріктеу шарттары» деп аталатын жолдан бастап, тиісті өрістердің мәндеріне салынатын шарттар жазылады. Бір жолда тұрған шарттар бір уақытта орындалуы тиіс, яғни олар бір-бірімен ЖӘНЕ амалымен бірігеді, оны орындаудың нәтижесі ретінде жазбалар жиынының *қиылысуы* болады. Конструктордың әр түрлі жолдарына жазылған шарттар НЕМЕСЕ амалымен қосылады, және нәтижесі ретінде *жазбалардың бірігуі* болады.

Кесте жазбаларды дерекқордан іріктеу кезінде сүзгінің рөлін атқарады: алдымен бір уақытта бірінші жолдың шарттарын қанағаттан-

дыратын жазбалар іріктеледі, содан соң оларға екінші жолдың шарттарын қанағаттандыратын жазбалар қосылады және т.с.с.

Сұраныстар Конструкторында қолданылатын, логикалық өрнектерді кестелік әдіспен іске асыру мысалдары 4.6-кестеде берілген. Бұрын қарастырылған формалды мысалдан алынған іріктеу шарттарықолданылған.

4.6. –кесте. Логикалық өрнектерді іске асыру мысалдары

Шарты	A	B	C	Шарты	A	B	C
1) A = 1 ЖӘНЕ B = 2	= 1	=2		8) A = 1 НЕМЕСЕ B = 2 ЖӘНЕ C = 3	= 1		
						= 2	=3
2) A = 1 НЕМЕСЕ A = 3	= 1			9) A ЕМЕС = 1 НЕМЕСЕ B = 2 ЖӘНЕС = 3	<>1		
	= 3					= 2	=3
3) A = 1 НЕМЕСЕ B = 2	= 1			10) (A = 1 НЕМЕСЕ B = 2) ЖӘНЕ C = 3	= 1		=3
		=2				= 2	=3
4) A = 1 НЕМЕСЕ B = 2 НЕМЕСЕ C = 3	= 1			11) B>= A		>=[A]	
		=2					
			=3				
5) A = 1 ЖӘНЕ B = 2 ЖӘНЕ C = 3	=1	=2	=3	12) B>=A ЖӘНЕ B>= C		>= [A] AND >= [C]	
6) A ЕМЕС = 1	<>1			13) A = B НЕМЕСЕ A = C	= [B]		
					= [C]		
7) A = 1 ЖӘНЕ B = 2 НЕМЕСЕ C = 3	= 1	=2		14) C = A+B			
			=3				= [A] + [B]

10 шартқа назар аударыңдар. Кестеге жазу кезінде іс жүзінде жақшалар ашылды, және берілген логикалық өрнек балама өрнекпен ауыстырылды:

A = 1 ЖӘНЕ *C*= 3 НЕМЕСЕ *B*= 2 ЖӘНЕ *C*= 3

Квадрат жақшаға алынған өрістің атауы жазбадағы осы өрістің мәнін сәйкестендіреді. Мұндай белгілеуді, түптеп келгенде, Конструктордағы барлық шартты қолдануға болады. Мысалы, *A* = 1 қатынасысұраныс Конструкторында *A* бағанында екі нұсқада жазуға болады: 1) [*A*] = 1; 2) =1. Екінші нұсқа қысқарақ, сондықтан әдетте соны қолданады. Мысалдағы шартты былайша жазуға болар еді: [*A*] = [*B*] or [*A*] = [*C*].

4.10. ДЕРЕКТЕР ҮЛГІСІН КЕҢЕЙТУ

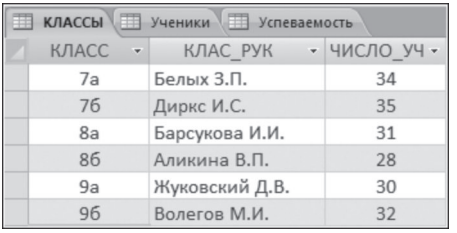
Қайтадан ДҚ жобалауға келейік. 4.3. қосалқы бөлімде сипатталған инфологиялық үлгіге сәйкес келетін деректердің толық үлгісін құруға келесі қадам жасайық. Бір жеке сыныпты қарастырудан тұтас мектепке көшейік. Енді ДҚ-ға мектептің барлық оқушылары жайлы және олардың үлгерімі туралы мәліметтер кіреді.

Мұндай кеңейтілген үлгіде объектінің жаңа типі пайда болады: СЫНЫП. Бұл объектінің атрибуттарына, әлбетте, сыныптың нөмірі (5а, 8б және т.с.с), сонымен қатар оның кейбір өзге де сипаттамалары жатады. Мысалы, мұндай атрибуттар ретінде сыныптағы оқушы саны,сынып жетекшісінің аты-жөні болуы мүмкін. «Сыныптар» қатынасын анықтайық:

СЫНЫПТАР(СЫНЫП, СЫН_ЖЕТ, ОҚ_САНЫ)

Тиісті кестенің үзіндісі 4.20-суретте көрстеілген.

Бұл кестені «Сыныптар» және «Үлгерім» кестелерімен байланыстыру керек. «Сыныптар» және «Оқушылар» арасындағы байланыс «бірі-көпке» типіне ие болады, өйткені әрбір сыныпта көптеген оқушы бар. Кестелер арасындағы байланыс ортақ өрістер арқылы орнатылады, сондықтан «Оқушылар» кестесіне енді СЫНЫП жаңа өрісті қосу керек.



КЛАССЫ | Ученики | Успеваемость

КЛАСС	КЛАС_РУК	ЧИСЛО_УЧ
7а	Белых З.П.	34
7б	Диркс И.С.	35
8а	Барсукова И.И.	31
8б	Аликина В.П.	28
9а	Жуковский Д.В.	30
9б	Волегов М.И.	32

4.20. –сурет. «Сыныптар» кестесі

ОҚУШЫЛАР(СЫНЫП, ОҚ_НӨМІРІ, ТЕГІ, АТЫ, ЖЫНЫСЫ, ТУҒАН_КҮНІ, МЕКЕНЖАЙЫ)

СЫНЫП өрісі ОҚ_НӨМІРІ өрісімен бірге құрама кілт құрайды. Шынында да, түрлі сыныптізімдеріндегі оқушылардың нөмірлері қайталанады (1-ші, 2-ші, 3-ші және т.с.с.). Сондықтан мектеп аумағында әрбір нақты оқушы сөзсіз тек сынып жиынтығымен және осы сынып тізіміндегі оқушы нөмірімен ғана анықталады. Мысалы, 7б + № 3 — бұл 7б сыныптан Герасимова Анна.

«Оқушылар» кестесінің мазмұны 4.21-суретте берілген.

Мұнда тек алты түрлі сыныптың: 7а, 7б, 8а, 8б, 9а, 9б алғашқы үш оқушысы туралы деректермен берілген кестенің үзіндісі көрсетілген. Жалпы мектеп үшін бұл өте үлкен кесте. Мысалы, мектепте 20 сынып болса, ал сыныптағы оқушылардың орташа саны - 30 болса, онда кесте 600 жазбадан тұратын болады.

Жоғарыда айтып кеткендей, «бірі-көпке» қатынасы кестелер арасында бірінші кестені негізгі кілті («бірі») екінші кесте өрістерінің арасында болғанда, және, сонымен бірге, оның негізгі кілтінің толықтай құрамында болмаған жағдайда байланыс орнатылады. Ол не негзгі кілт болып табылмайды, не екінші кестенің негізгі кілтінің бөлігі болады. Сондықтан «Үлгерім» кестесіне де енді СЫНЫП өрісін қосу керек.

ҮЛГЕРІМ (СЫНЫП, ОҚ_НӨМІРІ, ПӘН, 1_ТОҚС, 2_ТОҚС, 3_ТОҚС, 4_ТОҚС, ЖЫЛ)

Екі сегізінші сыныптың үш оқушысының үш пән бойынша үлгерімі туралы мәліметтер берілген «Үлгерім» кестесінің үзіндісі 4.22 кестеде көрсетілген.

КЛАССЫ | Ученики | Успеваемость

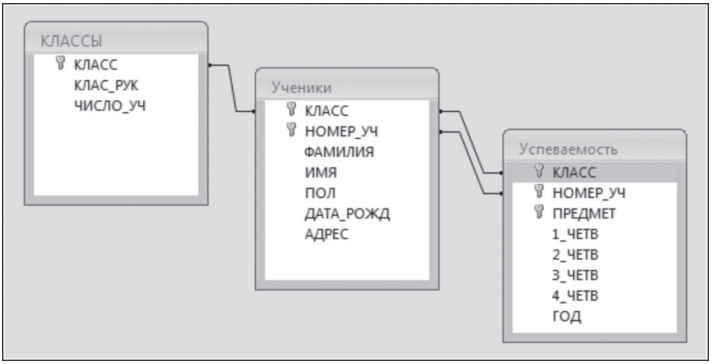
КЛАСС	НОМЕР_УЧ	ФАМИЛИЯ	ИМЯ	ПОЛ	ДАТА_РОЖД	АДРЕС
7а	1	Антипов	Петр	м	02.12.2001	Островского 3, кв.12
7а	2	Березин	Игорь	м	24.03.2001	Кирова 12, кв.56
7а	3	Буракова	Инна	ж	05.12.1999	Сормовская 1, кв.3
7б	1	Асмолова	Вера	ж	03.07.2000	Кирова 5, кв.87
7б	2	Вершинин	Олег	м	27.04.2000	Пушкина 2, кв.34
7б	3	Герасимова	Анна	ж	13.01.2001	Сормовская 4, кв21
8а	1	Антонов	Кирилл	м	23.09.2000	Садовая 5, кв.56
8а	2	Веткина	Ирина	ж	10.12.2000	Островского 3, кв.41
8а	3	Вяткин	Иван	м	01.11.1999	Садовая 3, кв.14
8б	1	Буркова	Светлана	ж	19.07.2000	Пушкина 14, кв 32
8б	2	Белкина	Нина	ж	19.06.1999	Кирова 3, кв.7
8б	3	Гилев	Николай	м	06.10.2001	Садовая 5, кв.8
9а	1	Астафьев	Семен	м	08.11.2000	Лебедева 5, кв.1
9а	2	Батуев	Василий	м	30.10.2000	Танкистов 21, кв 9
9а	3	Гурьевич	Елена	ж	03.01.2001	Пушкина 10, кв.4
9б	1	Быстров	Никита	м	07.09.2000	Кирова 4, кв 5
9б	2	Болотова	Лидия	ж	15.02.2000	Садовая 4, кв. 57
9б	3	Десяткин	Яков	м	03.11.2001	Сормовская 12, кв.6

4.21. сурет. «Оқушылар» жаңа кестесі

КЛАССЫ | Ученики | Успеваемость

КЛАСС	НОМЕР_УЧ	ПРЕДМЕТ	1_ЧЕТВ	2_ЧЕТВ	3_ЧЕТВ	4_ЧЕТВ	ГОД
8а	1	информатика	4	4	4	4	4
8а	1	история	3	4	4	4	4
8а	1	математика	3	3	3	3	3
8а	2	информатика	3	3	3	3	3
8а	2	история	4	4	4	4	4
8а	2	математика	5	5	5	5	5
8а	3	информатика	5	5	5	5	5
8а	3	история	5	5	5	5	5
8а	3	математика	5	5	5	5	5
8б	1	информатика	4	5	4	5	5
8б	1	история	4	5	4	5	5
8б	1	математика	4	4	4	4	4
8б	2	информатика	5	4	5	5	5
8б	2	история	4	3	4	3	3
8б	2	математика	5	5	5	5	5
8б	3	информатика	5	5	5	5	5
8б	3	история	5	4	5	5	5
8б	3	математика	5	4	5	5	5

4.22. сурет. «Үлгерім» кестесінің үзіндісі



4.23.сурет. Деректердің үш кестелік үлгісінің схемасы

Шын мәнісінде, мұндай кестенің өлшемі «Оқушылар» кестесінікіне қарағанда әлдеқайда үлкен. Егер мектептің 20 сыныбында орташа алғанда 30 адамнан оқыса, ал оқылатын пәнде саны 10 тең болса, онда «Үлгерім» кестесінде 6 000 жазба болуы тиіс.

Нәтижесінде келесі құрылымдағы үш кестелі ДҚ пайда болды:

СЫНЫПТАР (СЫНЫП, СЫН_ЖЕТ, ОҚ_САНЫ)
ОҚУШЫЛАР (СЫНЫП, ОҚ_НӨМІРІ, ТЕГІ, АТЫ, ЖЫНЫСЫ, ТУҒАН_КҮНІ. МЕКЕНЖАЙ)
ҮЛГЕРІМ (СЫНЫП, ОҚ_НӨМІРІ, ПӘН, 1_ТОҚС, 2_ТОҚС, 3_ТОҚС, 4_ТОҚС, ЖЫЛ)

MSAccess мұндай ДҚ құру кезінде оның схемасы 4.23. суретте берілген графикалық түрде бейнеленеді.

4.11. ІРІКТЕУДІҢ КҮРДЕЛІ ШАРТТАРЫМЕН БЕРІЛГЕН СҰРАНЫСТАР, ЕСЕПТЕЛЕТІН ӨРІСТЕР

Құрылған үш кестелі ДҚ ақпаратты іздестірудің бірнеше мысалын қарастырайық. Тиісті сұраныстарда іріктеудің күрделі шарттары қолданылатын болады.

4.5. мысал. Бір тоқсанда тарих пәнінен ең болмағанда бір үші бар 8-сынып оқушыларының тізімін алу. Сондай-ақ сынып жетекшісінің

тегін шығару.

Бұл сұраныста ДҚ барлық үш кестесі де қолданылады. Мұндай сұраныстың командасы SQL –да біршама ауқымды түрде беріледі:

```
SELECT Оқушылар.ТЕГІ, Оқушылар. СЫНЫП, СЫНЫПТАР.
       СЫН ЖЕТ
FROM   (СЫНЫПТАР INNER JOIN Оқушылар ON СЫНЫПТАР.
       СЫНЫП = Оқушылар.СЫНЫП) INNER JOIN Үлгерім
       ON (Оқушылар.ОҚ НӨМІРІ = Үлгерім.ОҚ НӨМІРІ)
       AND (Оқушылар. СЫНЫП = Үлгерім. СЫНЫП)
WHERE (((Оқушылар. СЫНЫП) Like "8?") AND
       ((Үлгерім. ПӘН)="СЫНЫП") AND ((Үлгерім.
       [1_тоқс])=3)) OR (((Оқушылар. СЫНЫП) Like "8?")
       AND ((Үлгерім. ПӘН)="тарих") AND ((Үлгерім.[2 тоқс])
       =3))OR (((Оқушылар. СЫНЫП) Like "8?") AND
       ((Үлгерім.ПӘН)="тарих") AND ((Үлгерім.[3 тоқс])
       =3)) OR (((Оқушылар. КЛАСС) Like "8?") AND
      ((Үлгерім.ПӘН)="СЫНЫП") AND ((Үлгерім.[4 тоқс])=3));
```

Мұндағы 8-сыныптарға қатысы бар жазбаларды іріктеу Like «8?» «бетперде» бойынша жүзеге асырылады. Бұл сыныптың атауы біріншісі «8» таңбасы болатын екі таңбадан тұратынын білдіреді.

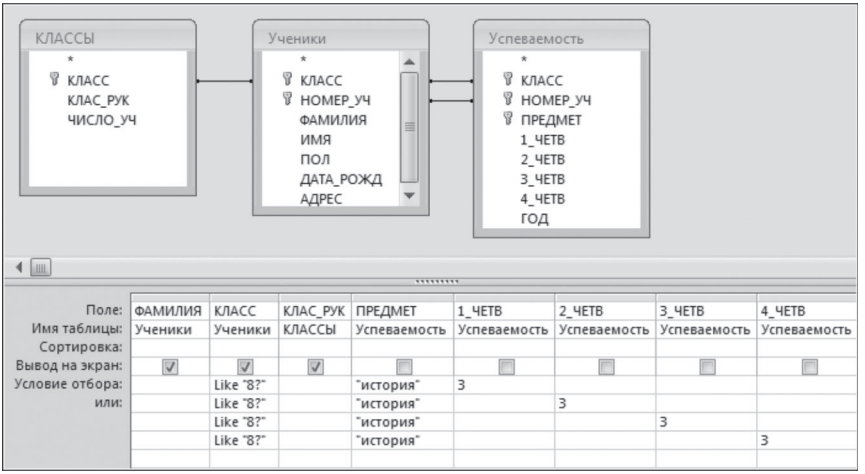
Ал дәл осы команда сұраныс конструкторында қалай құрылатынын көрейік (4.24 сурет).

Мұндай сұранысты орындаудың нәтижесі 4.25 суретте көрсетілген.

4.6. мысал. Төрт тоқсан бойынша математика пәнінен бағалар қосындысы 16- дан артық болатын 8-ші және 9-шы сыныптардың барлық оқушыларының тізімін алу.

Бұл сұраныста ҚОСЫНДЫ *есептелетін өріс* қолданылатын болады. Бұл өріс тек сұраныста ғана болады және ДҚ кестелеріне кірмейді. Команда на SQL команда келесідей беріледі:

```
SELECT Оқушылар. СЫНЫП, Оқушылар.ТЕГІ, [Үлгерім] .
       [1 тоқс] + [2 тоқс] + [3 тоқс]+[4 тоқс]
       AS ҚОСЫНДЫСЫ
FROM   Оқушылар INNERJOIN Үлгерім ON (Оқушылар.
       ОҚ_НӨМІРІ = Үлгерім.ОҚ_НӨМІРІ) AND (Оқушылар.
       СЫНЫП = Үлгерім. СЫНЫП)
WHERE(((Оқушылар. СЫНЫП) Like"8?") AND(([Үлгерім].
       [1_тоқс]+[2_тоқс]+[3_тоқс]+[4_тоқс])>16)
```



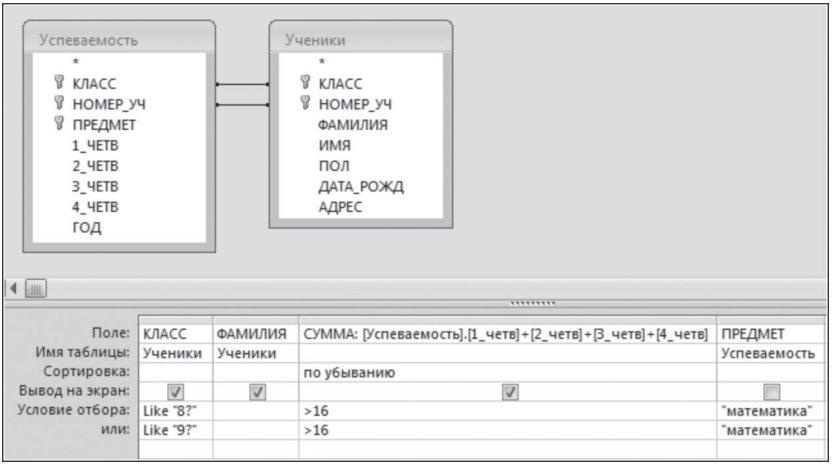
4.24. сурет. 1 есепке берілген сұраныстар Конструкторы

```
        AND((Үлгерім.ПӘН)= "математика")) OR(((Оқушылар.
        СЫНЫП) Like"9?") AND(([Үлгерім].[1 тоқс]+[2 тоқс]
        +[3 тоқс]+ +[4 тоқс])>16) AND((Үлгерім.ПӘН)=
        "математика"))
ORDER BY[Үлгерім].[1 тоқс]+[2 тоқс]+[3 тоқс]+
        +[4_тоқс] DESC;
```

Қорытынды кестеде ҚОСЫНДЫ тақырыбымен өріс шығарылады. Осы өріс бойынша сұрыптау жүргізуге, іріктеу шартын тағайындауға болады. Есептелетін өріс өрнек түрінде жазылады, берілген мысалда – бағалармен берілген төрт өрістің мәндерінің қосындысы. Бұл өрнектерде өрістердің аттары айнымалы шамалар ретінде қабылданады және шаршы жақшаға алынады. Өрнек бағдарламалауда және электронды кестелерде қолданылатын арифметикалық өрнектерге арналған дәстүрлі ереже бойынша жазылады.

ФАМИЛИЯ	КЛАСС	КЛАС_РУК
Антонов	8а	Барсукова И.И.
Белкина	8б	Аликина В.П.

4.25.сурет. 1 есепке берілген сұраныстың нәтижесі



4.26. 2 есепке арналған Конструктордағы сұраныс

КЛАСС	ФАМИЛИЯ	СУММА
9а	Гурьевич	20
9а	Астафьев	20
8б	Белкина	20
8а	Вяткин	20
8а	Веткина	20
8б	Гилев	19
9б	Десяткин	18

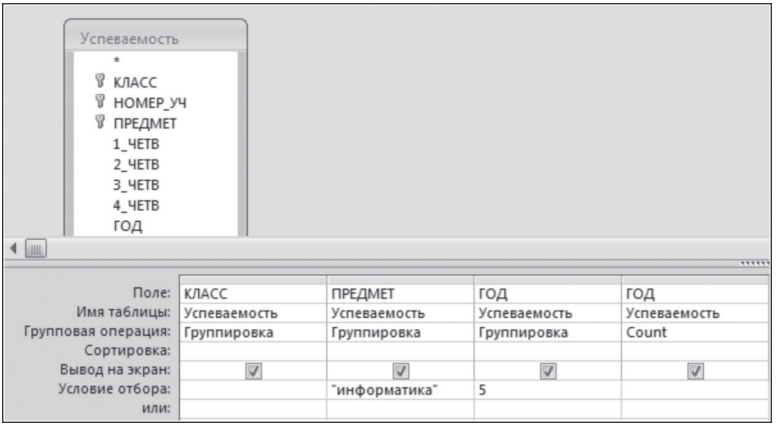
4.27. сурет. 1 есепке берілген сұраныстың нәтижесі

Берілген сұраныс конструктордың көмегімен 4.26 суретте көрсетілген құрырады.

Бұл сұранысты орындау нәтижесінде 4.27 суретте көрсетінген кесте шығарылады.

4.12. ҚОРЫТЫНДЫ СҰРАҚТАР ЖӘНЕ ЕСЕПТЕР

Жазба топтарында есептеулерді орындайтын сұраныстар *қорытынды жазбалар* деп аталады. Мұндай есептеулер: кейбір сандық өріс мәндерін қосу, еңкөп және ең аз мәнді табу, орташа мәнді есептеу, қандай да бір нақты мәндер санын есептеу және басқалары бола алады.



4.28.сурет. Конструктордағы қорытынды сұраныс

Келесі мысалды қарастырайық: мектептің әрбір сыныбы үшін информатика пәні бойынша үдіктер санын анықтау керек. Бұл тапсырманы орындау үшін «Үлгерім» кестесінен жазбаларды топтастыру, «Информатика» пәніне қатысты жазбаларды іріктеу керек, олардың ішінен тек жылдық бағасы «5» болатындарды таңдап, мұндай жазбалар санын есептеу керек.

Тапсырманы орындау тәртібі MS Access мынадай түрде беріледі:

1) Сұраныстар Коснтрукторы терезесін ашып, оған «Үлгерім» кестесін енгізу керек;

2) Сұраныс бланкінің бағандарында СЫНЫП, ПӘН, ЖЫЛ, ЖЫЛ өрістерін орналастыру керек;

3) Сұраныс бланкінің «Топтық амал» жолына қосу үшін *Топтық амалдар* орындау керек.(4.28 сурет).

Содн соң барлық өрістерде «Топтық амал» жолында «Топтастыру» мәні белгіленеді.

ПӘН өрісі үшін «= информатика» шартын енгізу қажет. ЖЫЛ өрісі екі рет қайталанған. Бірінші бағанда жылдық бағалардың таңдалатын мәндеріне берілген шарт беріледі. Бұл шарт « = 5». Екінші бағанда қорытынды функцияны орналастыру қажет. Қорытынды функция «Топтық амалдар» жолының және ЖЫЛ екінші өрістің қиылысындағы тордағы батырманы шерткенде ашылатын тізімнен таңдалады. Санын есептеу фнукциясы бұл тізімде «Count» атауына ие болады.Сұраныстың соңғы түрі 4.28 суретте берілген.

КЛАСС	ПРЕДМЕТ	ГОД	Count-ГОД
7а	информатика	5	1
7б	информатика	5	1
8а	информатика	5	1
8б	информатика	5	3
9а	информатика	5	2
9б	информатика	5	1

4.29. сурет. Қорытынды сұраныстың нәтижесі

Бұл команда SQL былайша жазылады:

SELECT Үлгерім. СЫНЫП, Үлгерім.ПӘН, Үлгерім. ЖЫЛ, Count (Үлгерім. ЖЫЛ) AS [Count-ЖЫЛ]
FROM Үлгерім
GROUP BY Үлгерім. СЫНЫП, Үлгерім.ПӘН, Үлгерім.ЖЫЛ
HAVING (((Үлгерімь.ПӘН)="информатика") AND ((Үлгерім.ЖЫЛ)=5));

Бұл сұраныстың нәтижесі ретінде 4.29 суретте берілген кесте болады. Нәтиженің мағынасы келесі: 8а сыныпта информатикадан бір бестік; 8б сыныпта – үш бестік және т.с.с.

Келесі сұраныста әрбір сынып үшін информатика бойынша жылдық бағалардың қосындысы анықталады. Мұнда топтастыру екі өріс бойынша жүргізіледі: СЫНЫП және ПӘН = «информатика». Бұл өрістердің мәндері бірдей болған жазба топтарында жылдық бағалар қосылады. Сандық өрістің мәндері жазбаларының топтары үшін қосу функциясы SUM атауына ие болады. Конструктордың қолданылуымен сұраныс 4.30 суретте көрсетілгендей құрылады.

Бұл сұранысты орындау нәтижесінде 4.31 суретте көрсетінген кестені аламыз.

Поле:	КЛАСС	ПРЕДМЕТ	ГОД
Имя таблицы:	Успеваемость	Успеваемость	Успеваемость
Групповая операция:	Группировка	Группировка	Sum
Сортировка:			
Вывод на экран:	☑	☑	☑
Условие отбора:		"информатика"	
или:			

4.30. Қосу функциясы бар қорытынды сұраныс

КЛАСС	ПРЕДМЕТ	Sum-ГОД
7а	информатика	12
7б	информатика	11
8а	информатика	12
8б	информатика	15
9а	информатика	14
9б	информатика	13

4.31. Қосу функциясымен берілген қорытынды сұраныстың нәтижесі

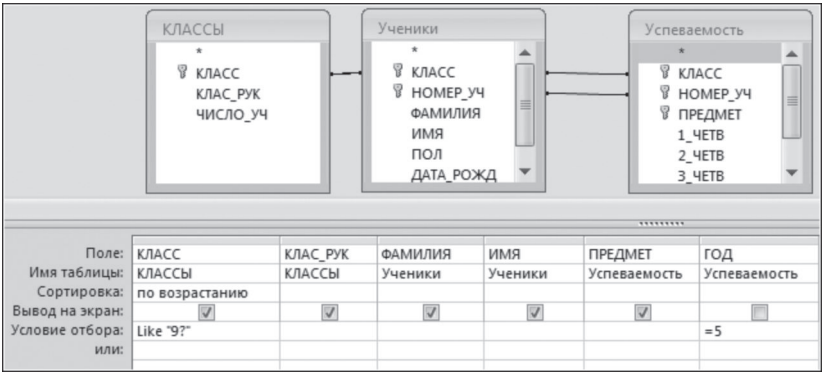
Жүзеге асырылуы үшін ДҚ-дың: «Сыныптар», «Оқушылар», «Үлгерім» барлық үш кестесініңдеректері қолданылатын сұранысты қарастырайық (4.32 сурет).

Әрбір пән бойынша 9-шы сынып оқушылары арасында жыл қорытындысы бойынша барлық үздіктердің тізімін алу қажет болды делік. Қорытынды кестеде сыныпты, сынып жетекшісін, оқушының аты-жөнін және «5» жылдық баға алған пәнді көрсету керек.

Мұндай сұранысты орындау нәтижесі ретінде 4.33 суретте берілген кесте болады.

Егер алынған тізім басылған құжат түрінде мектеп әкімшілігіне қажет болса, онда мұндай нысанда ол ыңғайсыз болады. Рәсімдеудің белгілі бір ережелеріне сәйкес келетін баспа құжаттар *есептер* деп аталады. ДҚБЖ-де есептерді алу мүмкіндігі қарастырылған. Есептер үшін ақпарат бастапқы кестелерден де, сонымен қатар сұраныс нәтижелеінен де алынады.

MS Access ДҚБЖ-де есеп құрудың неғұрлым оңтайлы тәсілі - Конструкторды пайдалану.



4.32. сурет. Үздіктерді іріктеу сұранысы

КЛАСС	КЛАС_РУК	ФАМИЛИЯ	ИМЯ	ПРЕДМЕТ
9а	Жуковский Д.В.	Гурьевич	Елена	математика
9а	Жуковский Д.В.	Гурьевич	Елена	история
9а	Жуковский Д.В.	Гурьевич	Елена	информатика
9а	Жуковский Д.В.	Астафьев	Семен	математика
9а	Жуковский Д.В.	Астафьев	Семен	информатика
9б	Волегов М.И.	Быстров	Никита	история
9б	Волегов М.И.	Десяткин	Яков	история
9б	Волегов М.И.	Десяткин	Яков	информатика
9б	Волегов М.И.	Болотова	Лидия	история

4.33. сурет. Үздіктерді іріктеу сұранысының нәтижесі

Есептер конструкторы құжаттың тиісті құрылымын жасауға, қаріпті, стильді, толтыруды және т.б. рәсімдеуге мүмкіндік береді. Есептер Шеберінің көмегімен құжатты бірнеше стандартты пішімде алуға болады. Шебердің көмегімен жасалған құжатты есептер Конструкторының көмегімен редакциялауға болады.

Бұрын алынған сұраныстың негізінде 4.34 суретте берілген есеп құрылды.

Отличники по предметам в 9 классах

КЛАСС	КЛАССНЫЙ РУКОВОДИТЕЛЬ	ФАМИЛИЯ	ИМЯ	ПРЕДМЕТ
9а	Жуковский Д.В.			
		Астафьев	*Семен*	
				математика
				информатика
		Гурьевич	*Елена*	
				математика
				история
				информатика
9б	Волегов М.И.			
		Болотова	*Лидия*	
				история
		Быстров	*Никита*	
				история
		Десяткин	*Яков*	
				история
				информатика

4.34.сурет. Үздіктердің тізімімен берілген есеп

4.13. ҚОРЫТЫНДЫ ЖОБА: МЕКТЕПТІҢ АҚПАРАТТЫҚ ЖҮЙЕСІН КЕҢЕЙТУ

Жобаның мақсаты — мектеп оқушыларының үлгерімі туралы мәліметтермен құрылған дерекқорды кеңейту және қосымша мәліметтер жасау.

1 кезең. 4.4 суретте бейнеленген инфологиялық үлгіге сәйкес келетін мектептің ақпараттық жүйесі үшін дерекқор үлгісін құрастыру.

Берілген кезеңді орындау үшін құрылған үлгіге мектептің оқытушылық құрамы туралы ақпаратты қосу қажет. ДҚ-да мұғалімдер туралы: «тегі», «аты», «әкесінің аты», «туған күні», «тұрғылықты мекенжайы», «оқыған оқу орны»және «жоо аяқтаған жылы» сияқты мәліметтер сақталуы тиіс делік. Бұдан басқа, ДҚ әрбір мұғалім қандай пәнді және қай сыныптарда оқытатыны жайлы мәліметтер болуы тиіс.

Жоғарыда аталған барлық деректерді екі қатынас арасында бөлу керек. Оларды «Мұғалімдер» және «Жүктеме» деп атайық. «Мұғалімдер» қатынасына тек мұғалімнің жеке атрибуты ғана кіруі тиіс:

МҰҒАЛІМДЕР (МҰҒ_КОДЫ, ТЕГІ, АТЫ, ӘКЕСІНІҢ АТЫ, ТУҒАН КҮНІ, МЕКЕНЖАЙЫ, ЖОО, ЖЫЛ,_ЖОО)

Бұл қатынасқа кілт ретінде «Мұғалімнің коды» өрісін енгізу «ОҚУШЫНЫҢ НӨМІРІ» өрісін «Оқушылар» қатынасына енгізумен бірдей мағынаға ие болады.

Мұғалімдер туралы мәліметтермен берілген бірінші жеті жазба 4.35 суретте көрсетілген.

Енді біз тағы бір маңызды проблемаға ұшырасамыз. 4.4 суреттегі схемада «көбі-көпке» қатынасымен «Мұғалімдер» және «Сыныптар» объектілерімен байланысқан «Пәндер» объекті берілген.

КОД_УЧ	ФАМИЛИЯ	ИМЯ	ОТЧЕСТВО	ДАТА_РОЖ	АДРЕС	ВУЗ	ГОД_ВУЗ
1	Аликина	Вера	Павловна	23.12.63	Смирнова 24, кв.16	ПГПУ	1996
2	Белых	Зинаида	Петровна	03.07.60	Кирова 56, кв.98	ПГУ	1989
3	Барсукова	Ирина	Ивановна	25.08.72	Мальцева 16, кв.34	ПГУ	1994
4	Волегов	Михаил	Ильич	09.01.70	Садовая 23, кв.14	ПГПУ	1994
5	Диркс	Иван	Семенович	29.12.65	Кирова 12, кв.8	МФТИ	1987
6	Доброва	Галина	Сергеевна	12.11.52	Садовая 12, кв.26	ПГПУ	1976
7	Жуковский	Дмитрий	Викторович	24.10.73	Пушкина 3, кв.49	НГУ	1999

4.35.сурет. «Мұғалімдер» кестесі

4.36. сурет. «Жүктеме» кестесі

КЛАСС	ПРЕДМЕТ	КОД_УЧИТ
7а	История	1
7а	Литература	3
7а	Математика	5
7б	История	1
7б	Литература	3
7б	Математика	6
8а	Информатика	7
8а	История	1
8а	Математика	5
8б	Информатика	7
8б	История	4
8б	Математика	5
9а	Информатика	7
9а	История	4
9а	Математика	6
9б	Информатика	7
9б	История	4
9б	Математика	5

Инфологиялық үлгіде бұл мүмкін болады.

Алайда, деректердің үлгісін құра отырып, қолданылатын ДҚБЖ жүктелетін шектеулерді ескеруге тиіспіз. Нақтырақ айтқанда, MS-Access ДҚБЖ деректер схемларында «көбі-көпке» қатынасын қолдануға мүмкіндік бермейді. Тек «бірі-бірге» және «бірі-көпке» қаатынастары ғана рұқсат етіледі.

Бұл мәселені келесідей шешуге болады: сызбаға «Пән» объектінің орнына «Жүктеме» деп атайтын объектіні енгіземіз. Тиісті қатынастың жазбаларында оқу жүктемесін мұғалімдер арасында бөлу туралы ақпарат, яғни берілген сыныпта берілген пәнді қандай мұғалім оқытатындығы туралы ақпарат болады. «Жүктеме» қатынасының құрылысы келесідей болады:

ЖҮКТЕМЕ(СЫНЫП, ПӘН, МҰҒ_КОДЫ)

«Жүктеме» кестесінің үзіндісінде алты сыныпта әрқайсысында үш пән бойынша мұғалімдердің оқу жүктемесін бөлу туралы ақпарат болады (4.36 сурет).

Берілген екі кестеден, мысалы, 8-ші және 9-шы сыныптарда информатика пәнін Жуковский Дмитрий Викторович оқытатыны көріеді.

«Жүктеме» қатынасы «Мұғалімдер» және «Үлгерім»,сонымен қатар «Мұғалімдер» және «Сыныптар» қатынастары арасындағы байланыс функциясын атқарады. Алынатын деректер схемасы 4.37 суретте көрсетілген.

«Мұғалімдер» және «Жүктеме» кестелерінің арасында «бірі-көпке» байланысы бар, өйткені, бір мұғалім әдетте бірнше сыныпта оқытады және бір ғана пәнді оқытып қоймайды. Байланыс МҰҒ_КОДЫ ортақ өрісі арқылы жүзеге асырылады. «Сыныптар» және «Жүктеме» кестелері арасындағы байланыс сондай-ақ «бірі-көпке» типіне ие болады, өйткені бір сыныпта көптеген пәндер оқытылады, бірақ берілген сыныпта берілген пәнді тек бір мұғалім оқытады. Байланыс СЫНЫП ортақ өріс арқылы орнатылады.

4.37. сурет. Қатынастар арасындағы байланыс схемасы

Ақыр соңында келесі құрылымы бар, бес кестелі ДҚ жобаланды:

МҰҒАЛІМДЕР (МҰҒ_КОДЫ, ТЕГІ, АТЫ, ӘКЕСІНІҢ АТЫ, ТУҒАН_КҮНІ, МЕКЕН-ЖАЙЫ, ЖЫЛЫ_ЖОО)
ЖҮКТЕМЕ (СЫНЫП, ПӘН, МҰҒ КОДЫ)
СЫНЫПТАР (СЫНЫП, СЫН ЖЕТ, ОҚ САНЫ)
ОҚУШЫЛАР (СЫНЫП, ОҚ_НӨМІРІ, ТЕГІ, АТЫ, ЖЫНЫСЫ, МЕКЕНЖАЙЫ)
ҮЛГЕРІМ (СЫНЫП, ОҚ НӨМІРІ, ПӘН, 1_ТОҚС, 2_ТОҚС, 3_ТОҚС, 4_ТОҚС, ЖЫЛ)

ДҚ жаңа нұсқасында «Оқушылар» және «Үлгерім» кестелері бұрынғы қалпында сақталды. Ал «Сыныптар» қатынасында СЫН_ЖЕТ өрісі енді «Мұғалімдер» кестесінде сынып жетекшінің реттік нөміріне (кодына) нұсқама беру енгізілген. Бұл өрістің типі мәтіндіктен сандыққа өзгерген. «Сыныптар» кестесі енді 4.38 суретте берілген көрініске ие болады.

Реляциялық ДҚ телриясында алынған деректер үлгісі *деректердің ғаламдық схемасы* деп аталады. Ғаламдық мағынасы схема жеке қосымшаларға, яғни деректерді өңдеудің жеке есептерді шешугебайланбағанын білдіреді. Сонымен бірге оның негізінде ДҚ енгізілген ақпарат шегінде көптеген осындай нақты есептерді шығаруға болады. ДҚ мұндай қасиеті *қосымшаларға тәуелді болмау* ұғымын білдіреді.

КЛАСС	КЛ_РУК	ЧИСЛО_УЧ
7а	2	34
7б	5	35
8а	3	31
8б	1	28
9а	7	30
9б	4	32

4.38. сурет. «Сыныптар» кестесі

Жобаның 1-кезеңін орындауға берілген тапсырмалар

1 тапсырма. ДҚ-ға «Мұғалімдер» кестесін қосыңдар және оны деректермен толтырыңдар.

2 тапсырма. ДҚ-ға «Жүктеме» кестесін қосыңдар және оны деректермен толтырыңдар.

3 тапсырма. 4.37 суреттегі кестеге сәйкес кестелер арасындағы байланысты анықтаңдар.

4 тапсырма. «Сыныптар» кестесінің құрылымы мен мазмұнына өзгерістер енгізіңдер.

2-кезең. Құрылған ДҚ сұраныстар сериясын іске асыру.

1-кезеңде құрылған бес кестелі деректер үлгісі ДҚ-ға салынған ақпарат шегінде дерекқордағы көптеген қосымшаларды (сұраныстарды) іске асыруға мүмкіндік береді. Ғаламдық схемада нақты қосымшаның (ұсыныстың) көзқарасы тұрғысынан алғанда «артық» ақпарат болады. Әдетте қандай да бір сұранысты қанағаттандыру үшін бүкіл ғаламдық схема қажет емес, оның *қосалқы схема* деп аталатын шағын бір бөлігі ғана қолданылады. Ғаламдық схема неғұрлым толық болса, оның негізінде сан түрлі қосымшалар құру мүмкіндігі соғұрлым арта түседі.

5 тапсырма. Сыныпты және сынып жетекшісінің ТАӘ көрсетіп, сынып жетекшілерінің тізімін алыңыз.

Мұндай сұранысты орындау үшін бір-бірімен байланысқан екі қатынастан тұратын: «Сыныптар» және «Мұғалімдер» ғаламдық схеманың тек бір бөлігі ғана қолданылады. Бұл берілген қосымша үшін қосалқы схема болып табылады.

6 тапсырма. Жеке пәндер бойынша «толық үздіктердің» (пән бойынша төрт тоқсанда да бес деген баға аоған оқушылар) тізімін алыңдар. Тізімде сыныпты, оқушының тегін, пәннің атын, осы пән бойынша мұғалімнің тегін көрсетіңдер.

Сұраныс құру кезінде келесі ерекшелікті ескеріңдер. Міндетті түрде сұраныс командасында үш кестеден: «Оқушылар», «Мұғалімдер» және «Үлгерімнен» тұратын өрістер қатысатын болады. Алайда қолданылатын қосалқы схемада төртінші кесте –«Жүктеме» болуы тиіс. Оны елемеуге болмайты, өйткені «Жүктеме» кестесі арқылы «Мұғалімдер» және «Үлгерім» арасында байланыс орнатылады.

7 тапсырма. Оқу жылының қорытындысы бойынша математика пәнінен қанағаттанарлық үштік баға алғандар туралы мәліметтер-

ден тұратын кестені алыңдар. Кестеде сыныпты, математика пәні мұғалімінің ТАӘ, оқушының аты-жөнін көрсетіңдер.

Бұл есепті шығару үшін 6-тапсырмадағыдай қосалқы схема қолданылады.

8 тапсырма. (қорытынды сұранысты алуға). Оқу жылының қорытындысы бойынша мектептің әрбір сыныбы үшін информатика пәні бойынша үздіктер санын анықтаңдар.

Список учителей

ФАМИЛИЯ	*ИМЯ*	*ОТЧЕСТВО*	*КЛАСС*	*ПРЕДМЕТ*
Аликина	*Вера*	*Павловна*		
			7а	История
			7б	История
			8а	История
Барсукова	*Ирина*	*Ивановна*		
			7а	Литература
			7б	Литература
Волегов	*Михаил*	*Ильич*		
			8б	История
			9а	История
			9б	История
Диркс	*Иван*	*Семенович*		
			7а	Математика
			8а	Математика
			8б	Математика
			9б	Математика

4.39. сурет. Қорытынды есептің көрінісі

Бұл тапсырманы орындау үшін «Үлгерім» кестесінен жазбаларды топтастыру, сыныптар бойынша «Информатика» пәніне қатысты жазбаларды іріктеу, олардың ішінен жылдық бағасы «5» болатындарын ғана таңдап алу керек, және мұндай жазбалардың санын анықтаңдар.

9 тапсырма. Мектеп мұғлімдері арасында оқу жүктемесін бөлу туралы деректер бар кестені алыңдар. Бұл кестеде әрбір мұғалім үшін қандай пәндерді және қай сыныптарда оқытатыны туралы ақпарат көрсетілкі тиіс.

Бізді қызықтыратын ақпарат «Мұғалімдер» және «Жүктеме» кестелерінде болады. Бұл сұраныста ешқандай іріктеу шарты жоқ.

10 тапсырма. 9-тапсырмаға берілген нәтижені есептер Шеберін пайдаланып, есеп түрінде рәсімдеңдер.

Қорытынды есеп 4.39 суретте көрсетілгендей болуы тиіс.

БАҚЫЛАУ СҰРАҚТАР ЖӘНЕ ТАПСЫРМАЛАР

1. Алғашқы буындағы ЭЕМ көбіне қай салаларда қолданылды?
2. Компьютерлік ақпараттық жүйелерді дамыту үшін техникалық негіз ретіндне болған не?
3. Ақпараттық жүйе деген не?
4. Ақпараттық жүйе қандай компоненттерден тұрады?
5. Инфологиялық үлгі деген не?
6. Инфологиялық үлгі қандай элементтерден құрылады?
7. Инфологиялық үлгіде қандай байланыс түрлері қолданылады?
8. Егер тақырыптық сала ретінде бір мектеп емес, қала мектептерінің бүкіл жүйесі болатын болса, 4.4 суреттегі үлгіге қандай өзгерістер нгізуге болады?
9. БҚ не үшін керек? Дұрыс жауапты таңдаңдар:
   а) компьютерде есептеулерді орындау үшін;
   б) деректерді сақтау, іздестіру және сұрыптауды жүзеге асыру үшін;
   в) басқару шешімдерін қабылдау үшін.
10. ДҚ жіктеудің қандай нұсқалары бар?
11. Реляциялық ДҚ-да деректерді ұйымдастырудың қандай негізгі әдісі қолданылады?
12. Реляциялық ДҚ –дағы жазба деген не?
13. Өріс, өрістің типі деген не? Қандай өріс типтері болады?
14. Жазбаның негізгі кілті деген не?
15. Келесі қатынастарда негізгі кілтті және жазба типтерін анықтаңдар:
    АВТОБУСТАР (МАРШРУТ НӨМІРІ, АЛҒАШҚЫ АЯЛДАМА, СОҢҒЫ АЯЛДАМА)
    КИНО (КИНОТЕАТР, СЕАНС, ФИЛЬМ, РЕСЕЙЛІК, ҰЗАҚТЫҒЫ,)
    САБАҚТАР (АПТА КҮНІ, САБАҚ НӨМІРІ, СЫНЫП, ПӘН, ОҚЫТУШЫ)
16. ДҚ үшін жазбалар құрылымын сипаттаңдар (өріс аттары, өріс типтері, негізгі кілт): ҰШАҚТАР САПАРЫ, ҚАЛА МЕКТЕПТЕРІ, ӘЛЕМ ЕЛДЕРІ.
17. Толық функционалды ДҚБЖ көмегімен қандай әрекеттерді орындауға болады?
18. Деректер үлгісін қалыптандыру қандай мақсатта жүргізіледі?
19. Бірінші, екінші және үшінші қалыпты нысан талаптары қандай?
20. Декомпозиция жолымен келесі қатынастарды қалыптандырыңдар (үшіші қалыпты нысанға келтіріңдер):

ЕМХАНА (НАУҚАСТЫҢ ТЕГІ, КЕЛУ КҮНІ ТУҒАН КҮНІ, УЧАСКЕСІ, ДӘРІГЕР, ДИАГНОЗЫ)

ҰШАҚ САПАРЫ (САПАРДЫҢ НӨМІРІ, ҰШУ КҮНІ, ҰШУ УАҚЫТЫ, БЕЛГІЛЕНГЕН ПУНКТ, ҰШАҚТЫҢ ТҮРІ, ОРЫН САНЫ, ЖОЛДА ҰШУ УАҚЫТЫ, КЕМЕ _КОМАНДИРІ, ҰШҚЫШТЫҢ_КЛАССЫ)

21. ДҚ құру неден басталады?
22. Кесте Конструкторында қандай ақпарат беріледі?
23. Деректерді енгізудің қандай әдістері қолданылады?
24. Схема не үшін керек? Ол қалай құрылады?
25. ДҚБЖ автоматты түрде «Оқушылар» және «Үлгерім» кестелері арасындағы байланыс типі - «бірі -көпке» болатынын қандай белгілері бойынша анықтады?
26. ДҚ-да деректерді айла-шарғылау ұғымы нені білдіреді? Іріктеу сұранысының мақсаты қандай?
27. БҚ -на сұранысты сипаттаудың қандай тілдері бар?
28. Іріктеуге беріген сұраныс командасы SQL –да құрылымы қандай болады?
29. Сұраныс Конструкторы кестесінің құрылымы қандай болады?
30. Жылдық бағалары ішінен үштік бағалары бар оқушылар нөмірлерін іріктеуге сұранысты SQL–да жазыңдар.
31. Access сұраныс Конструкторын қолданып, 30-тапрсыманың сұранысын суреттеңдер.
32. SQL-да тегі, сыныптың барлық ер балаларының аты және мекенжайы жазылған тізімін құрайтын іріктеу командасын жазыңдар. Жолдар тегі алфавит бойынша берілген тәртіпте орналасуы тиіс.
33. Access сұраныс Конструкторын қолданып, 32-тапрсыманың сұранысын суреттеңдер.
34. SQL-да жылдық бағалары ішінен үштік бағалары бар оқушыларды жазыңдар
35. SQL-да жылдық бағалары ішінен үштік бағаға ие болған барлық оқушылардың тегі мен атын іріктеуге сұраныс жазыңдар.
36. Access сұраныс Конструкторын қолданып, 30-тапрсыманың сұранысын суреттеңдер.
37. Логикалық өрнек деген не?
38. Қандай негізгі логикалық өрнектер бар? Ақиқаттық кестесі деген не?
39. Келесі шарттар бойынша 4.6 кестесі үшін жазбаларды іріктеу нәтижелерін анықтаңдар:
    1) $A = 2$ ЖӘНЕ $B = 2$
    2) $A = 2$ НЕМЕСЕ $B = 2$
    3) $A = 2$ ЖӘНЕ $B = 1$ НЕМЕСЕ $C = 3$
    4) $A>B$

5) $C = A + B$

6) $A = 1$ НЕМЕСЕ $A = 2$

7) $B > 1$ ЖӘНЕ $B < 3$

40. 38-тапсырмадағы барлық шарттарды кесте түрінде, яғни сұраныс конструкторы тілінде көрсетіңдер.
41. Дайын ДҚ-ға жаңа кестелерді қосуға бола ма?
42. «Сынып» жаңа кестесі мен «Оқушылар» және «Үлгерім» кестелері арасында байланыс қалайша орнатылады?
43. Мектеп туралы ақпарат қосылатын ДҚ құрылымын суреттеңдер: мектептің нөмірі, мектептің мекенжайы, яғни ДҚ-да бірнеше мектеп оқушыларының үлгерімі туралы ақпарат болады.
44. Есептелетін өріс деген не? Оны қайда қолдануға болады?
45. Сұраныс Конструкторы пішімінде әрі қарай аталатын A...B есептері үшін іріктеуге берген команданы көрстеіңдер. Барлығында сұрыптаманы бірінші өріс бойынша ұйымдастырыңдар.

    А. Оқу жылының барлық төрт тоқсанында тарихтан бес деген баға алған оқушылардың тегі, аты және сыныптары берілген тізімді алу.

    Б. Жыл бойы «біркелкі» оқыған, яғни барлық тоқсандық бағалары бірдей болған оқушылар үшін тегі, аты, сыныбы және математикадан жылдық бағасы көрсетілген тізімді алу.

    В. Математикадан жылдық бағасы олардың төрт тоқсан бойынша орташа бағасынан жоғары болған 9-шы сынып оқушыларының тізімін алу.
46. «Жазба топтары» деген не? Сұраныс Конструкторының құрлдарымен жазба топтары қалай құрылады?
47. Берілген жазба топтарымен қандай топтық амалдарды орындауға болады?
48. «Есеп» деген не жәнеMSAccess есеп қандай құралдармен құрылады?

Әдебиеттер тізімі

Абрамов В.Г. Тілге кіріспе Паскаль / В.Г. Абрамов, Н. П. Трифо¬нов, Г. Н. Трифонова. — М. : Ғылым, 1988.

Алексеев Е.Р. FreePascal, Lazarus/ Е.Р. Алексеев, О. В. Чеснокова, Т. В. Кучерт. — М. : ДМК-пресс, 2010.

Бакаревич Ю. Б. MicrosoftAccess2000 / Ю. Б. Бакаревич, Н. В. Пушкина. — СПб. : БХВ-Санкт-Петербург, 2004.

Ван Тассел Д. Бағдарламаларды стилі, әзірлеу, тиімділігі, ретке келтіру және сынау / Д. Ван Тассел. — М. : Мир, 1981.

Вирт Н. Деректердің алгоритмі және құрылымы / Н. Вирт. — М. : Мир, 1989.

Грогоно П. Паскаль тілінде бағдарламалау / П. Грогоно. — М. : Мир, 1982.

Дейт К. Дерекқор жүйесіне кіріспе / К.Дейт ; ағылш. тіл.ауд. — 6-е бас. — К. : Диалектика, 1998.

Епашников А.М. Ортада бағдарламалау ТурбоПаскаль 7.0 / А. М.Епашников, В.А. Епашников. — М. : МИФИ, 1994.

Иванова Г. С. Объектілі-бағдарланған бағдарламалау : оқулық / Г. С. Иванова, Т. Н. Ничушкина, Е. К. Пугачев ; Г. С. Ивановаың ред. — М. : Н. Э. Бауман ат. МГТУ басп., 2001.

Йенсен К. Паскаль — пайдаланушыларға арналған нұсқаулық және тілді сипаттау / К. Йенсен, Н. Вирт. — М. : Мир, 1982.

Касаткин В. Н. Ақпарат. Алгоритмдер. ЭЕМ / В. Н. Касаткин. — М. : Просвещение, 1991.

Мартин Дж. Дерекқорды және есептеу жүйелерін ұйымдастыру / Дж. Мартин. — М. : Мир, 1980.

Михеева В. MicrosoftAccess2002 / В. Михеева, И. Харитонова. — СПб. : БХВ-Санкт-Петербург, 2002.

Окулов С. М. Бағдарламалау бойынша есептер/ С.М. Окулов, Т. В. Ашихмина, Н.А.Бушмелева. — М. : БИНОМ. Білім зертханасы, 2006.

Окулов С. М. Бағдарламалау негіздері / С. М. Окулов. — М. : БИНОМ. Білім зертханасы, 2012.

Окулов С.М. Алгоритмдердегі бағдарламалау / С.М.Окулов. — М. : БИНОМ. Білім зертханасы, 2013.

Ушаков Д. М. Паскаль оқушылар үшін / Д. М. Ушаков, Т. А. Юркова. — СПб. : Питер, 2010.

Фаронов В. В. Delphi. Жоғары деңгейлі тілде бағдарламалау : оқулық / В. В. Фаронов. — СПб. : Питер, 2003.

Шупруга В. В. Delphi2006 мысалдарда / В. В. Шупруга. — СПб. : БХВ-Санкт-Петербург, 2006.

Мазмұны

Оқу басылымы

Семакин Игорь Геннадьевич

Бағдарламалау және дерекқор негіздері

Оқулық

Редакторы *Л. В. Толочкова*
Техникалық редакторы *Е. Ф. Коржуева*
Компьютерлік беттеу: *Р. Ю. Волкова*
Корректоры *А. П. Сизова*

Бас. № 101116448. Баспаға жіберілді 08.07.2014. Пішімі 60 х 90/16.
«Балтика» гарнитурасы . Офсеттік қағаз. Офсеттік баспа. П. б. шарт. 14,0.
Таралымы 2 000 дана. Тапсырыс №
«Академия» баспа орталығы» ЖШБ. www.academia-moscow.ru129085, Мәскеу, Бейбітшілік д-лы, 101В, құр. 1.
Тел./факс: (495) 648-0507, 616-00-29.
Санитариялық-эпидемиологиялық қорытынды № РОСС RU. АЕ51. Н16592 29.04.2014ж.
«Тверской полиграфический комбинат» ААҚ. 170024, Тверь қ., Ленин д-лы, 5. Телефоны: (4822) 44-52-03, 44-50-34. Телефон/факс (4822) 44-42-15.
Homepage— www.tverpk.ru
Электронды пошта(E-mail) — sales@tverpk.ru