1.1 Алгоритм туралы түсінік

Кез келген бағдарламаны жазу үдерісін қарастырсақ, екі негізгі кезеңді бөліп алуға болады – есепті шешу алгоритмін құру және алгоритмді іске асыру кезеңдері.

Есепті шешу алгоритмін құру кезеңін түсіну үшін алгоритм ұғымын анықтап алу керек.

«Қазіргі кездегі алгоритм сөзінің мағынасы рецепт, үдеріс, әдіс, тәсіл, рәсім, бағдарлама сөздерінің мағынасына өте ұқсас, бірақ «алгоритм» сөзінің қосымша сипаты бар. Алгоритм белгілі бір есепті шығаруда операциялардың орындалу ретін анықтайтын ережелер жиыны ғана емес, оның басқа ең негізгі бес қасиеті бар:» [4, 29 бет].

Алгоритм ұғымының ең негізгі бес қасиеті бар, олардың әрқайсысы оның ерекшелігін сипаттап, әрекеттер ретін анықтайтын басқа сөздердің мағынасынан бөледі.

Алгоритмнің негізгі қасиеттері:

- алгоритмнің аяқталуы;
- анықтылық;
- тиімділік;
- деректерді енгізу;
- деректерді шығару.

Алгоритм қасиеттерін анықтаған кезде авторлардың көпшілігі бірінші үш қасиеттерімен ғана шектеледі. Осы бөлімде біз алгоритмнің барлық бес қасиеттерін қарастыратын боламыз. Алгоритмнің бірінші қасиеті - алгоритмнің аяқталуы, яғни алгоритм қадамдардың соңғы санында аяқталуы тиіс. Сонымен бірге қадамдар саны өте көп болуы мүмкін, бірақ олар аяқталуы тиіс.

«Алгоритмнің аяқталу қасиетінен басқа алгоритмнің барлық қасиеттеріне ие процедураны есептеуіш әдіс деп атауға болады» [4, 30 бет].

Алгоритмнің екінші қасиеті - анықтылық, яғни алгоритмнің әрбір қадамы нақты анықталуы тиіс. Мысалы, рецепте «Бір салым тұзды қосыңыз» [32 бет, 4], нұсқауы саусақтардың өлшемі мен азық-түліктің көлемін ескермейді.

Алгоритмнің үшінші қасиеті - тиімділік, яғни шешім қолданылатын ресурстардың ең аз шығынын жұмсап, тез және дұрыс орындалуы тиіс. Алгоритмнің тиімділігін бағалайтын көптеген әдістемелері бар, оның өзі -тұтас бір ғылым, бірақ, әзірше тек қана анықтамасымен ғана шектелейік.

Деректерді енгізуді ұйымдастыру мен бағдарламаны басқару процесі алгоритмнің қазіргі кездегі ұғымында алгоритмнің жеке қасиеті болып бөлінді. Бұл қасиет көбінесе «Пайдаланушының деректерді енгізу интерфейсін» анықтайды, яғни бағдарламаны пайдалану барысында деректерді енгізу мүмкіндігі.

Алгоритмнің соңғы қасиеті бағдарлама жұмысының нәтижелерін шығаруды анықтайды. Олар технологиялық процестер бойынша суреттер,

кесте, график, кейбір цифрлық немесе аналогты мәндер, апаттық жағдайлардың дыбыстық сигналдары болуы мүмкін, т.б..

1.2 Есепті шешу алгоритмін құру кезеңі

Есепті шешу алгоритмін құру есепті талдаудан басталады, яғни есептің мәнін анық түсіну керек. Есепті талдаудан кейін ғана алгоритмді дайындауды – есепті шешетін ретті қадамдарды құруды бастауға болады.

Есепті шешу алгоритмін құру бойынша соңғы жұмыс – алгоритмді тексеру.

Есепті шешу алгоритмін құру кезеңінде алгоритм айқын болмаса, онда есептің шешімін бір-бірімен байланысқан жеке бағыныңқы есептерге немесе блогтарға бөлу керек. Әрбір блог үзінділерге бөлінуі мүмкін және ол әрбір үзіндінің орындау алгоритмі айқындалғанға дейін қайталанады. Есепті шешу кезінде жеке үзінділердің алгоритмдері есептің жалпы шешімін сипаттайтын бір алгоритмге біріктіріледі.

Алгоритмді құрудың бұл кезеңі ең маңызды болып табылады, өйткені осы кезеңде есепті шешудің барлық мүмкін нұсқалары қарастырылуы керек. Әсіресе ол шешмнің алгоритмі анық емес, ауқымды немесе логикалық күрделі есептерді шешу кезінде керек.

Қарапайым, оқу бағдарламаларын шешу кезінде есепті шешу алгоритмі интуитивті түрде анықталады немесе сіз оның алгоритм екенін білмей-ақ «есіңізде ұстап жүресіз». Осындай есептерді қарастырған кезде біз оның орындалуын ғана қарастырамыз.

Алгоритмді құру кезінде есеп шешімінің түрлі нұсқалары болуы мүмкін, сондықтан шешімнің ең дұрыс нұсқасын таңдау үшін «алгоритмнің талдауын» жүргізе білу керек [33 бет, 4] .

Алгоритмді құру нәтижесі болып алгоритмнің сипаттамасы немесе оның құрылымдық схемасы есептеледі.

Алгоритм жұмысының дұрыстығын тексеретін бірнеше әдістемелер бар, мысалы, деректердің белгіленген мәндері беріледі және есептің шешімі қолмен қаралады.

Алгоритмдерді тексерудің ең жақсы нұсқасы - әлдебір бағдарламалау тілінің көмегімен орындау, яғни дайын алгоритмді бағдарламалау тілінің бағдарламалық коды арқылы сипаттау (бағдарлама мәтіні) және оның жұмысын компьютерде тексеру. Алгоритмді құру кезеңіне көшейік.

1.3 Алгоритмді жүзеге асыру кезеңі

Алгоритмді құру кезеңі мыналардан тұрады

– бағдарламалау тілдерінің бірінде құрылған алгоритм бойынша бағдарлама кодын жазу;

– бағдарламаны тестілеу – компьютерде бағдарламаны іске қосу және оның жұмысының нәтижелерін белгілі бір бақылау мәліметтері бойынша немесе логикалық түрде тексеру.

Алгоритмді құру кезеңдерін орындау үшін кем дегенде бір бағдарламалау тілін білу керек.

Алгоритмдеу негіздерін білмей бағдарламалауды үйрену мүмкін емес және алгоритмдеу негіздерін үйрену бағдарламалаусыз мүмкін емес.

Барлық бағдарламалаудың алгоритм тілдері, әдетте, құрылымдық немесе процедуралық бағдарламалау технологиясын қолдануға негізделген.

Қазіргі кезде объекті-бағытталған бағдарламалау (ОББ) технологиясы негізге алынып отыр, ол үшін арнайы визуалды бағдарламалау орталары дайындалған, мысалы, Delphi, VISUAL C++, түрлі VISUAL STUDIO және т.б..

Бұл оқулықта Visual Studio.NET визуалды бағдарламалау ортасында объекті-бағытталған C# тілін қолданып бағдарламалау сұрақтары жан-жақты қарастырылған.

Оқулықтың бірінші бөлімінде C# тілінде бағдарламалаудың негіздері қарастырылған, онсыз Visual Studio.NET визуалды бағдарламалау ортасы ұсынатын бағдарламалау технологиялары мен объекті-бағытталған бағдарламалау технологияларын меңгеру мүмкін емес.

Әлбетте, бағдарламалау тілін меңгерудің бастапқы кезеңінде алгоритмдеудің кейбір сұрақтарын жан-жақты қарастырамыз, өйткені есеп шешімінің алгоритмдерін құру үдерісі мен бағдарлама кодын жазу бір-бірімен байланысты. Алайда, бұдан былай есеп шешімі алгоритмінің идеясы мен бағдарламалық жүзеге асырылуы ғана қарастырылады.

1.4 .NET платформасының құрылымы

.NET платформасының идеясы - кез келген тілде жазылатын қосымшаларды дайындауға және орындауға арналған бірыңғай жүйені жасау.

Қосымшаларды жазу үшін NET платформасына бірнеше бағдарламалау тілдеріне арналған, Visual Studio.NET деп аталатын құру ортасы қосылды. Оның ішінде қосымша жобасының кодын енгізіп, түзету жасауға арналған мәтіндік редактор және жобаны жөндеу мен іске қосу құралдары, анықтама жүйесі, т.б. элементтері бар.

Әр түрлі бағдарламалау тілдері бойынша деректер типтерінің үйлесімділігі үшін .NET платформасы бағдарламалау тілінің әрбіріне типтердің ортақ жүйесін (Common Type System – CTS) – компьютер жадысында деректерді сақтаудың бірыңғай түрін талап етеді.

Қосымшалардың түрлі типті компьютерлерге тасымалдауды қамтамасыз ету үшін .NET платформасында біріңғай аралық компиляция тілі (Common Intermediate Language – CIL) қарастырылған. Оған платформаның кез келген тілінде жазылған қосымшалар түрлендіріледі.

Осы тілдің командалары нақты операциялық жүйеге, компьютер типіне, қосымша кодына тәуелді емес. CIL тіліндегі бағдарлама өз бетімен орындалмайды, ол кез келген компьютерге, кез келген операциялық жүйеге орнатуға болатын, жалпытілді орындау ортасы (Common Language Runtime,– CLR) деп аталатын жүйенің бақылауымен орындалады. Жалпытілді орындау ортасының CIL тіліндегі кодты нақты бір процессордың машиналық командасына аударатын JIT компиляторы бар.

JIT компилятордың атауы өз жұмысының принципін сипаттайды, яғни қосымшаның осы сәтте орындауды қажет ететін бөлігін ғана компиляциялау (just in time – дер кезінде).

.NET платформасында қауіпсіздікті қамтамасыз ету үшін жүйелік көзқарас қолданылады - қосымша компиляциясы кезеңінде exe немесе dll кеңейтуі бар арнайы файл құрылады, яғни CIL тілінде коды және метадеректері бар құрылым. Пайдаланушы компьютерінде қауіпсіздік пен қосымшаны орнату мен өрбітудің жеңілдігін қамтамасыз ететін метадеректер құрастырудың аты мен нұсқасын, қосымшада қолданылатын объект туралы мәліметтер мен деректер типін, құрастыру тәуелді файлдар т.б. өзіне қосады.

Кез келген .NET. тілдерінде бағдарламалауда қолдануға болатын .NET платформасының үлкен кітапхана класы (Framework Class Library – қысқаша .NET Framework) бар.

Материал мазмұнында қосымша, жоба, бағдарлама терминдері жиі қолданылады. «Қосымша» терминін «бағдарлама» терминінің синонимі сияқты қабылдануы мүмкін. «Консоль» үшін құрылатын қосымшаларды бағдарламалар деп атаймыз. «Windows» (Windows- қосымшалар) үшін құрылатын бағдарламаларды қосымшалар деп атаймыз.

Зерттеу сатысындағы қосымша жоба деп аталады.

1.5 Visual Studio.NET ортасы

Кез келген визуалды бағдарламалау ортасы сияқты Visual Studio.NET құру ортасы .NET үйлесімді тілдер қолданатын қосымшаларды жазу, түзету, компиляциялау, дұрыстау және іске қосу құралдарын ұсынады..NET платформасының Visual Studio.NET құру ортасы төрт тілмен жұмыс істеуге арналған: C#, VB.NET, C++ және J#, бірақ қазіргі уақытта басқа да бағдарламалау тілдерін .NET платформасына қосу үшін құрулар белгілі.

Visual Studio.NET ортасы әр түрлі типті жобаларды жасауға мүмкіндік береді, мысалы:

– Windows-қосымшалар Windows интерфейсінің дәстүрлі басқару элементтерін қолданады;

– консольді қосымша шығаруды командалық процессор терезесінде орындайды;

– басқа қосымшаларда пайдалану үшін қолданылатын кластар кітапханасы (Dll модульдері) ;

– веб-қосымшалар, яғни әдістерін Интернет арқылы шақыруға болатын қосымшалар, т.б.

Visual Studio.NET ортасы технологиясының басым бөлігі Windows және веб-қосымшаларды құруға арналған, бірақ әзірлеушілер консольді қосымшалармен жұмысты да ескерген.

Консольді қосымшамен жұмысты орындау барысында пайдаланушы Windows операциялық жүйесінің командалық жолының терезесіне немесе Турбо Паскаль ортасының мәтіндік тәртіптегі терезесіне ұқсас мәтіндік терезеде жұмыс істейді.

«Консольді қосымшалар тілді оқу үшін ең қолайлы болып келеді, өйткені оларда графикалық интерфейсті дайындау үшін көптеген стандартты объекттер қолданылмайды» [Павл. 15 бет].

Сондықтан оқулықтың бірінші бөлімінде деректерді өңдеудің түрлі алгоритмдерін зерттеуге көбірек уақыт бөлу мақсатында және C# тілінің базалық қасиеттерін қарастыру үшін біз консольді косымшаларды ғана құраймыз.

Оқулықтың екінші бөлімінде Windows-қосымшаларын және объекті-бағытталған бағдарламалау негіздерін қолданып, бағдарламалау технологиясы қарастырылатын болады.

Visual Studio.NET ортасында жұмыс жасау үшін жобаларды жасау, оларды магнитті дискте сақтау, жобаны іске қосу, т.б. бойынша білім керек, сондықтан C# тілін үйренуді Visual Studio.NET ортасын оқудан бастайық.

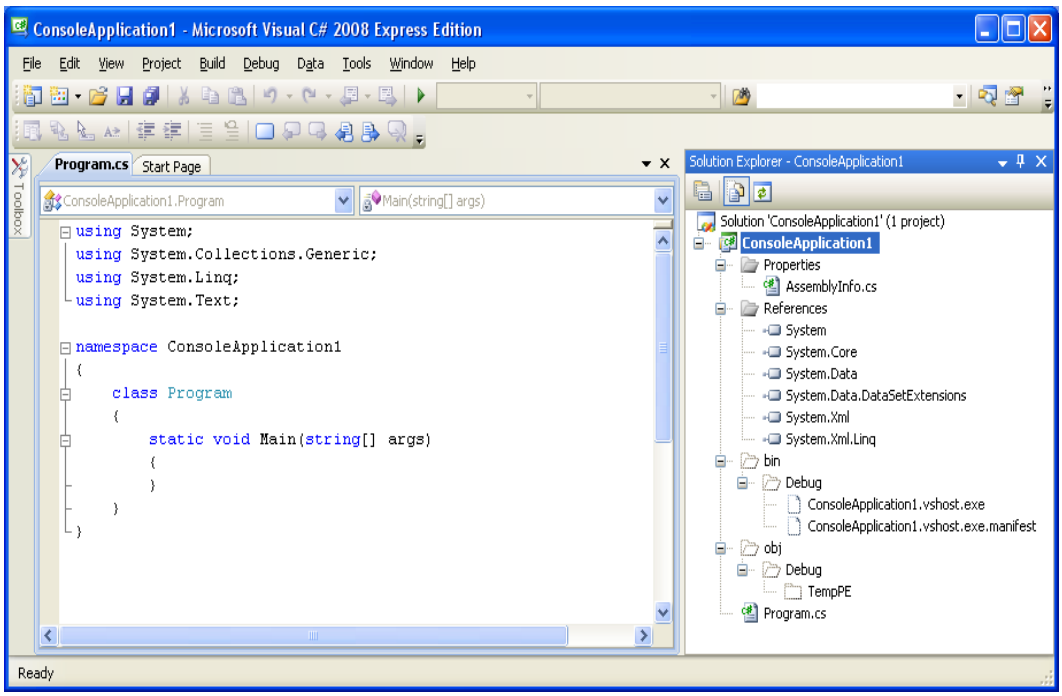
1.6 Жобаны жасау. Ортаның негізгі терезелері

Жобаны жасау үшін Visual Studio.NET ортасы іске қосылғаннан кейін, басты менюде File ->New -> Project командасын таңдау керек.

Ашылған диалогтық терезенің сол жағында Visual C# Windows тармағын, ал оң жағында Console Application тармағын таңдау керек.

Name өрісіне жоба атын жазуға болады, бірақ жобаның берілген атын қалдыруды ұсынамыз. Жобаның орналасу орны ретінде Location өрісінде (жобаның дискіде сақтау орны) компьютердің жұмыс орнын жазыңыз.

Берілген мәндер расталғаннан кейін ортада жоба берілген атаумен құрылады. Бағдарлама кодының үлгісі 1.1-суретінде көрсетілген.



1.1-сурет – Консольді қосымша жобасы құрылғанынан кейінгі Microsoft Visual Studio ортасының үлгісі

1.1-суреттің жоғарғы бөлігінде бас меню (File, Edit, View режимдерімен, т.б.) және орта режимдерінің командаларын жылдам таңдауға арналған құрал-саймандар панелі орналасқан.

Экранның жоғарғы оң жағында жобаны басқару терезесі - Solution Explorer орналасқан (егер ол көрінбей тұрса, бас менюдің View -> Solution Explorer командасын пайдалану керек).

Терезеде жобаға қатысты барлық ресурстар жазылған, мысалы, System, System.Data, System.Xml атаулар кеңістігіне сілтемелер және т.б.

Экранның негізгі бөлігін редактор терезесі алып тұр, онда ортада автоматты түрде құрылған бағдарлама коды (мәтін) орналасқан. Бағдарлама коды үлгі ретінде құрылады, оған Сіз керекті жағдайда өз кодыңызды жаза аласыз.

Бағдарлама коды көрсетілген кезде оның әртүрлі құрамдас бөліктерінің түрлі-түсті түстері қолданылады. Қызметтік сөздерге (кодтың резервте сақтаулы сөздері) көк түс арналған. Түсініктеме сұр немесе қара жасыл түсте, ал қалған мәтін қара түсте көрсетіледі.

Мәтіннің сол жағында құрылым символдары орналасқан, ішінде минус немесе плюсы бар вертикаль сызықтар мен квадраттар қосылған. Егер минусы бар кез келген квадратты басса, онда сәйкес код фрагментін (блогын) «жинауға» болады, бұл ретте минус плюске ауысады. Егер плюс квадраты басылса, онда сәйкес код фрагменті (блогы) экранның сәйкес бөлігіне «жайылады». Ақпаратты көрсетудің осындай құрылымдары Windows жүйесінің бумаларын көрсету кезінде қолданылады және бағдарлама кодының керекті фрагментіне назар аударуға мүмкіндік береді.

Магнитті дискіде жобаны жазу үшін File->Save All команда тізбегін қолдану керек (көптеген дискеттер салынған құрал-саймандар панелінде). Редакцияланатын файлда өзгерістерді ғана сақтау үшін File->Save Program.cs командасын таңдауға немесе аспаптық панелінде бір дискета салынған батырманы басу керек.

Жобаны ашу үшін келесі File->Open Project команда тізбегін қолдану керек.

1.7 C# тілінің символикасы

C# бағдарламалау тілі жаңа .NET технологияларын қолдайтын объекті-бағытталған бағдарламалау тілі болып табылады. Кез келген тілде сияқты C# тілінде алфавит– символдар жиыны бар, олардың көмегімен бағдарлама коды (мәтін) жазылады. C# тілінде бағдарламаны жазу үшін келесі символдар қолданылады:

– кез келген әріптерді, C# тілі бас және кіші әріптерді ажыратады;
– 0 - 9 дейінгі цифрлар;
– қызметтік белгілер;
– қызметтік сөздер;
– бағдарлама мәтіні бойынша түсініктемелер.

Әріптер мен цифрлар бойынша барлығы түсінікті, ал қызметтік белгілер мен сөздерді (олардың саны өте көп) тілді оқу барысында меңгереміз.

Бағдарлама кодын жазу барысында оның әр түрлі конструкциясының (операторлар, функциялар, объекттер, жазулар, айнымалылар, т.б.) аттары, яғни идентификаторлары анықталуы тиіс. Олар әріптерден (латын алфавитін қолдану қажет), цифрлардан (0 – 9), астын сызу ‘_’ символынан тұруы мүмкін, бірақ идентификатордың бірінші символы әріп немесе астын сызу символы болуы керек, сан болмауы тиіс.

Мысалы:

– дұрыс – `A_9, B34, TYTI, Max;`
– дұрыс емес – `A-9, B 34, 13FIG.`

Егер қызметтік сөздердің алдында ‘@’ символы тұрса, оны идентификатор ретінде де қолдануға болады. Мысалы , @for, бірақ бағдарламалауда қызметтік сөздерді қолдану жағымсыз әдеттің нышаны деп есептеледі. ‘@’ символын қызметтік сөздерден өзге сөздермен бірге қолдануға болмайды.

1.8 Айнымалыларды сипаттау

Бағдарламалау тілінің «айнымалы» ұғымының алуан түрлі анықтамасы бар, бірақ компьютердің аппараттық қамтамасыз ету тұрғысынан ең дұрыс анықтамасы мынандай – бағдарламаны орындау уақытының әр түрлі мезетінде

түрлі мәндерді болуы мүмкін компьютер жадысының аймағы. Осы жадының аймағына сәйкес қолданылатын символикалық ат айнымалы аты немесе айнымалы идентификаторы деп аталады.

Айнымалы - идентификатормен белгіленген компьютер жадысының аймағы, онда бағдарламаның жұмысы кезінде өзгеретін деректер сақталады.

Бағдарламада айнымалыны қолданбас бұрын оны жариялау керек – оның атын және типін көрсету керек.

Айнымалы типі оған қандай мәндерді жазуға болатынын көрсетеді – бүтін, нақты, символдық, т.б.

Айнымалы аты идентификатор болып келеді.

Мысалы:

```
int TYT;
float cym, kol;
```

Мұнда, `TYT` айнымалысы – бүтін, ал `cym`, `kol` айнымалылары – нақты (бөлшек) типте екені көрсетілген.

Бір типті айнымалылар нүктелі үтірмен аяқталуы тиіс.

Айнымалыны жариялау кезінде жариялау операторы қолданылады - нүктелі үтірмен аяқталатын әрекет. Бірақ әдебиеттерде бұл амалды жариялау операторы деп атаудың орнына жай ғана бағдарлама айнымалысын жариялау деп атайды.

Айнымалыны бірінші рет қолдану жариялау болып есептелетін BASIC бағдарламалау тілінен немесе айнымалылары бағдарлама басында, арнайы бөлімде жарияланатын PASCAL бағдарламалау тілінен өзгеше **C#** бағдарламалау тілінде айнымалы бағдарламаның кез келген орнында жариялана алады. Сонымен қатар айнымалыны PASCAL тіліне ұқсас бағдарламаның басында немесе BASIC тіліне ұқсас бірінші рет қолдану барысында жариялауға болады.

1.9 C# тілінің деректер типі

.NET бағдарламалау технологиясының жаңа бір міндеті CTS (Common Type System) – жалпы типтер жүйесін қолдану болып табылады. Ол осы технологиямен жұмыс істейтін кез келген бағдарламау тілі үшін компьютер жадында деректер көрсетімін стандарттауға мүмкіндік береді. Шартты түрде барлық CTS деректер типі мәнді (бүтін, нақты, т.б.) және сілтемелік (массивтер, кластар, т.б.).болып бөлінеді. Компьютер жадысы мәнді типтегі айнымалыларға бағдарлама компиляциясы кезінде, ал сілтемелік типтегі айнымалыларға `new` операторының көмегімен бағдарлама орындалу барысында бөлінеді.

.NET технологиясында деректер типін белгілеу күрделі иерархиялық құрылымға ие, мысалы, System.Int32. Сондықтан C# тілінде кейбір жиі қолданылатын типтердің жазуын жеңілдету үшін қарапайым типтер ұғымы енгізілді. Қарапайым типтер бұл – кейбір мәнді және сілтемелік типтердің

қысқартылып жазылған түрі. 1.1-кестесінде C# тілінің қарапайым типтері және осы типтерге сәйкес CTS типтері көрсетілген.

1.1-кестесі – C# тілінің қарапайым типтері

C# тілінің қарапайым типтері	.NET платформасының CTS типтері
Byte	System.Byte
Sbyte	System.SByte
Short	System.Int16
Int	System.Int32
Long	System.Int64
Usort	System.UInt16
Uint	System.UInt32
Ulong	System.UInt64
Float	System.Single
Double	System.Double
Object	System.Object
Char	System.Char
String	System.String
Decimal	System.Decimal
Bool	System.Boolean

Бағдарламада деректер типін таңдауда есептеу дәлдігінің талаптары және компьютердің айнымалыларға бөлінетін жады көлемі ескеріледі.

Бағдарламалауды меңгерудің бірінші кезеңінде біз тек қана қарапайым типтерді ғана қолданамыз, ал қалған басқа деректер типтерін материалды меңгеру барысында қарастырамыз.

1.10 C# тілінің тұрақтылары

C# тілінде тұрақтыларды жариялау айнымалыларды жариялаумен бірдей, бірақ. `const` қызметтік сөзі қосылып жазылады.

Мысалы:

```
const char CIMV = 'y';
const int MAX = 640;
```

Тұрақты бұл – идентификатормен белгіленетін, бағдарлама жұмысының барысында өзгермейтін деректердің мәні сақталатын компьютер жадысының аймағы.

Тұрақтыны жариялаған кезде оның типі ғана емес, сонымен бірге оның мәні де көрсетілуі керек.

1.11 C# тілінің атаулар кеңістігінің ұғымы

C# тіліндегі кез келген бағдарлама `using` операторының көмегімен бағдарлама кодына кейбір атаулар кеңістігін көрсетуден басталады. Мысалы:

```
using System;
using System.Windows.Forms; и т.д.
```

Бұл ретте бір `using` операторына тек бір атаулар кеңістігі тиісті бола алады.

Әрбір атаулар кеңістігі .NET платформасына тиісті кластардың белгілі бір тобына сәйкестенеді (әрбір класс - белгілі бір тип). Сонымен, .NET платформасының барлық типтер жиынтығы (CTS-те 4000 аса түрлі типтер белгілі) өзінің функционалды міндеті бойынша логикалық байланысқан топтарға біріктірілді, олар атаулар кеңістігі деп аталады. Кейбір міндеттері шешу үшін кейбір кластар, әдістер, функциялар немесе деректер керек болса, онда сіз бағдарламаға тиісті атаулар кеңістігін қосуыңыз керек**.** Сонымен, **C#** тілінде NET платформасының бұрында жазылған кластар кітапханасын қолдану мүмкіндігі орындалады.

Ескере кететін бір жағдай, **C#** тілі бойынша оқулықтардың көптеген авторлары кітапхана терминін қолданбай, атаулар кеңістігі ұғымымен шектеледі. 1.2-кестесінде ең жиі қолданылатын атаулар кеңістігі көрсетілген

Бағдарламалауға жаңадан келген бағдарламашылардың келелі мәселелерінің бірі - керекті атаулар кеңістігінің атын және оны қосу жолдарын анықтау. Атаулар кеңістігімен жұмыс жасау технологиясын C# тілін меңгеру барысына қарай қарастырамыз.

Айта кететін жәйт, түрлі Visual Studio орталарының «қосымшаларды дайындау шеберлері» керекті атаулар кеңістігін автоматты түрде таңдап алады, қажеттілік болмаса оларды өзгертпеген дұрыс. Мысалы, егер Visual Studio 2008 ортасында консольде орындалатын қосымша (Fail->New->Project->Console Application) таңдалса, онда қосымшаны дайындау шебері 1.1-суретінде көрсетілген атаулар кеңістігін автоматты түрде қосады.

1.2-кестесі – **.**NET платформасының кейбір атаулар кеңістігі

.NET платформасының кейбір атаулар кеңістігі	Тағайындалуы
System	Object класы бар түпкі атаулар кеңістігі және қарапайым типтегі деректермен, математикалық функция жинағымен, деректерді енгізу–шығарумен, қоқысты жинау операциясымен жұмыс жасауға арналған кластар жиынтығы, т.б.
System.Data System.data.SqlClient және т.б..	Бұл атаулар кеңістігі деректер базасымен жұмыс жасауға арналған
System.IO	Бұл атаулар кеңістігі файлға деректерді енгізу–шығаруға жауап береді, т.б.

System.Drawing System.Drawing.Drawing2D	Бұл атаулар кеңістігінің кластарында графикалық қарапайым құралдары, қаріптер жиыны, сызықтар түрлері, монитор экранында графикалық ақпаратты көрсету құралдарының жиындары бар.
System.Net	Кластар жиыны желілер бойынша деректерді табыстауға жауап береді.
System. Security	Кластар жиыны желілер бойынша деректерді табыстау қауіпсіздігін артыру үшін қолданылады.
System.Web	Кластар жиыны web-қосымшаларда жұмыс жасауға арналады.
System.Windows.Form	Бұл атаулар кеңістігінің кластары Windows интерфейсінің элементтерімен - терезелер, батырмалар, басқа да басқару элементтерімен жұмыс жасауға арналған .

Бағдарламаның атаулар кеңістігін анықтағаннан кейін Visual Studio 2008 ортасында консольді қосымшаны дайындау шебері `namespace Console Application { . . .}` арнайы нұсқауымен бағдарлама кодының аумағын анықтады, онда құрылған қосымшаның деректер типін қолдануға болады.

Айта кететін жәйт, ортаның бас терезесінің сыртқы көрінісі терезені күйге келтіруге байланысты, оны бағдарламашы «өзіне керекті нұсқада» орындайды. Ортаны меңгеру кезеңінде күйге келтіру сипаттамаларын өзгертпеуді ұсынамыз.

1.12 Өзін-өзі тексеру сұрақтары

1 Алгоритм ұғымы.

2 Есептерді шешу алгоритмін құру кезеңдері неден тұрады?

3 Алгоритмді құру қандай кезеңдерден тұрады?

4. NET платформасы не үшін арналған?

5 Visual Studio.NET қосымшаларын дайындау ортасы дегеніміз ...

6 Құрастыру (сборка) файлы неден тұрады?

7 Түрлі тілдерде жазылған бағдарламаларды тасымалдау .NET платформасында қалай қамтамасыз етіледі?

8 .NET платформасының .NET Framework неден тұрады?

9 .NET платформасының CTS неден тұрады?

10 C# тілінде айнымалы ұғымы