ЛЕКЦИЯ 1

C++ программалау тіліне, алгоритмдер және деректер құрылымдарына кіріспе

C++ тілінің базалық құралдары

Кез келген табиғи тілдегі мәтінде 4 негізгі элементті бөліп қарастыруға болады:
- символдар
- сөздер
- сөз тіркестері
- сөйлемдер.

Осындай элементтер алгоритмдік тілде де бар, алайда мұнда сөздерді ***лексемалар*** (элементар конструкциялар), сөз тіркестерін ***өрнектер***, ал сөйлемдерді ***операторлар*** деп атайды.

Лексемалар символдардан, өрнектер лексемалар мен символдардан, ал операторлар символдар, өрнектер және лексемалардан құралады

Тіл алфавиті немесе ***символдары*** – бұл тілдегі барлық мәтіндер жазылатын, ары қарай бөлінбейтін негізгі таңбалар;

Лексема немесе ***элементар конструкция*** – өзіндік мағынасы бар тілдің ең кіші бірлігі;

Өрнек белгілі бір мәннің есептелу ережесін береді;

Оператор кез келген бір әрекеттің аяқталған спатталуын береді.

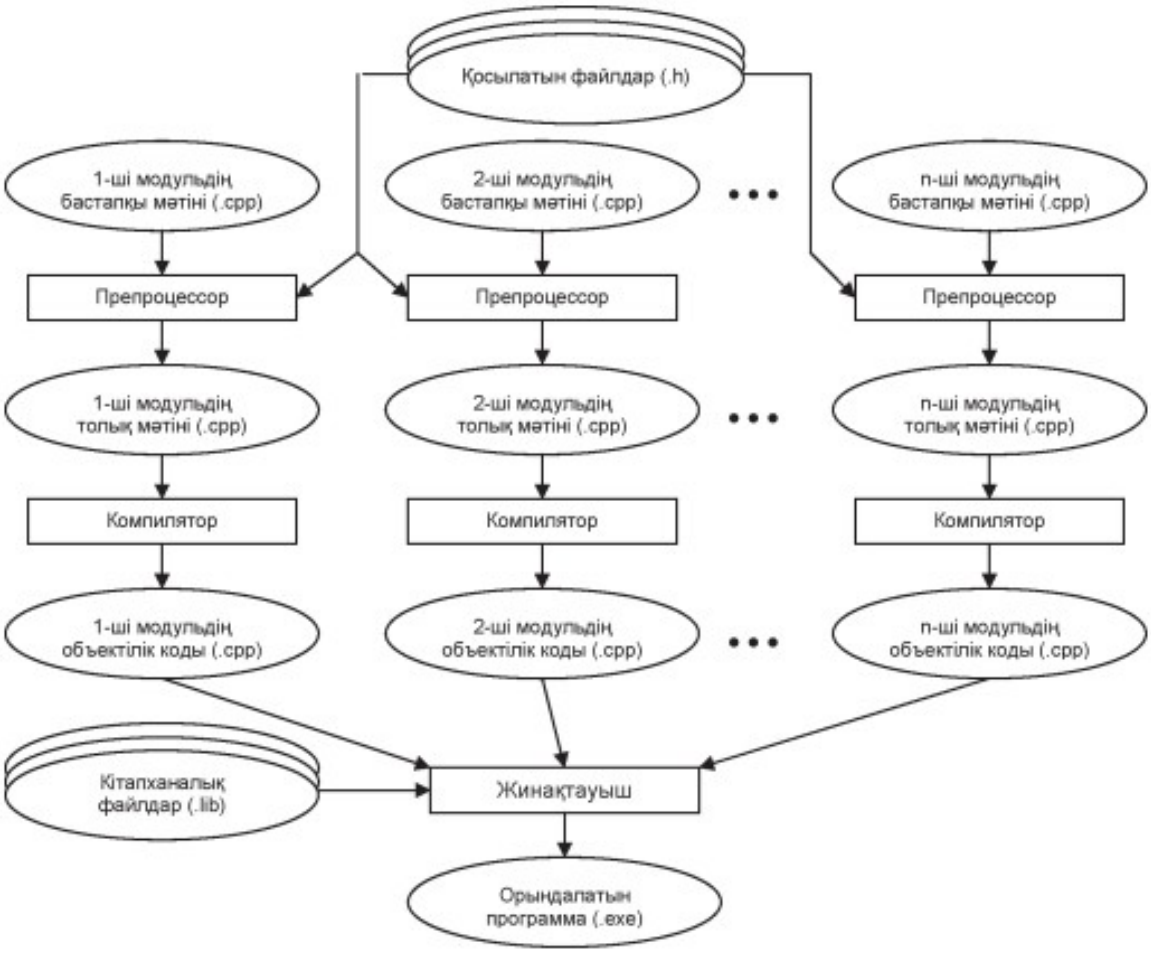
Операторлар ***орындалатын*** және ***орындалмайтын*** болып бөлінеді.

Орындалатын операторлар мәліметтермен атқарылатын әрекеттерді білдіреді.

Орындалмайтын операторлар мәліметтерді сипаттау үшін қажет, сондықтан оларды көбінесе сипаттау операторлары немесе жай ғана сипаттауыштар деп те атайды.

Тілдің әрбір элементі ***синтаксис*** және ***семантика*** арқылы анықталады. Синтаксистік анықтаулар тіл элементтерінің құрылу ережесін білдіреді, ал семантика олардың мағынасы мен қолданылу ережелерін анықтайды.

Біртұтас алгоритммен біріктірілген сипаттаулар жиыны мен операторлар алгоритмдік тілдің программасы болып табылады. Программаны орындау үшін, оны процессорға түсінікті машиналық кодқа аудару керек. Бұл процесс бірнеше сатылардан тұрады.



Программа алдымен ***препроцессорға*** беріледі, ол мәтін құрамындағы ***директиваларды*** (мысалы, мәтінге тақырыптық файлдарды – программада қолданылатын элементтердің сипаттамасы сақталған мәтіндік файлдарды қосуды) орындайды.

Алынған программаның толық мәтіні ***компиляторды*** кіру нүктесіне беріледі де, компилятор лексемаларды тауып алып, тіл грамматикасының негізінде, осы лексемалардан тұратын өрнектер мен операторларды анықтайды. Сонымен қатар, компилятор синтаксистік қателерді табады, егер олар жоқ болса, ***объектілік модуль*** құрады.

Жинақтауыш (компоновщик) немесе байланыстар редакторы бір объектілік модульге басқа объектілік модульдерді және де кез келген программада болатын (мысалы, экранға мәлімет шығаруды жүзеге асыру үшін) кітапхана функцияларын қамтитын модульдерді біріктіре отырып, программаның ***орындалатын модулін*** қалыптастырады.

Егер программа бірнеше бастапқы файлдардан құралатын болса, олар жеке-жеке түрлендіріліп, осы жинақтау сатысында біріктіріледі. Осының нәтижесінде шығатын атқарылатын модульдің кеңейтілуі .exe болады және ол әдеттегі тәсілмен орындауға жіберіледі.

Құжаттамаларда, кітаптарда тіл ережелерін сипаттау үшін көбінесе бір формальді метатіл қолданылады, мысалы, ***Бэкус-Наур формулалары*** немесе ***синтаксистік диаграммалар***

Баяндау тілінің қарапайымдылығы мен көрнекілігін сақтау үшін кең таралған сипаттаудың формальді емес түрін қолданамыз, мұнда синтаксистік конструкциялардың міндетті түрде қажет етілмейтін бөліктері тік жақшаға алынады да, белгілі бір нақты мәнмен

ауыстырылуы тиіс мәтін қазақ тілінде жазылған, ал бірнеше элементтің бірін таңдап алу шарты ***вертикаль сызықпен*** белгіленген. Мысалы,

[void | int] атауы();

түріндегі жазба С++ тілінің ережелеріне сәйкес ***атауы*** конструкциясының орнына нақты бір басқа ат қойылуы тиіс екенін, ал оның алдында ***void*** немесе ***int*** сөздерінің бірі болуы немесе ешбір түйінді сөз болмауы мүмкін екендігін білдіреді.

Жүйелі жақшалар бірнеше элементтің ішінен біреуі ғана таңдап алынуы тиіс элементтерді топтастырады. Ал тік жақшалардың өздері синтаксистік элемент ретінде қарастырылатын жағдайда ол ерекше түрде айтылады.

С++ тілінің алфавиті

С++ тілінің алфавиті келесі таңбаларды қамтиды:

- латын тілінің бас және кіші әріптері және астын сызу белгісі;
- 0-ден 9-ға дейінгі араб цифрлары, A және F арасындағы оналтылық цифрлар;
- арнайы белгілер: " { } , | [ ] ( ) + - / % *. \ ‘ : ? < = > ! & # ~ ; ^
- бос орын символдары: бос орын, табуляция символы, жаңа жолға көшу символы.

Алфавит символдарынан тіл лексемалары құралады. Олар:

- идентификаторлар;
- түйінді (резервтегі) сөздер;
- операциялар белгілері;
- тұрақтылар;
- ажыратқыштар (жақшалар, нүкте, үтір, бос орын симводары).

Лексемалардың шекараларық символдары ретінде ажыратқыштар немесе операциялар белгілері сияқты басқа лексемалар қолданылады.

Идентификатор – бұл программалық объектінің атауы. Оны құру үшін латын әріптері, цифрлар және астын сызу белгісін қолдануға болады. Бас әріптер мен кіші әріптер әртүрлі символдар болып саналады, мысалы, ***sysop***, ***SySoP*** және ***SYSOP*** – үш түрлі атау.

Идентификатордың алғашқы символы әріп немесе астын сызу белгісі болуы тиіс, алғашқы символ ретінде цифрды қолдануға болмайды. Атау символдарының ішіне бос орын қойылмайды.

Стандарт бойынша идентификатордың ұзындығы шектелмеген, бірақ кейбір компиляторлар мен жинақтауыштар (компоновщиктер) оған шектеу қояды. Идентификатор айнымалыларды, функцияларды, типтерді және т.с.с. жариялау кезеңінде құрылады, осыдан кейін оны программаның келесі операторларында қолдануға болады. Идентификаторды таңдауда төмендегі шарттарды ескерген жөн:

- идентификатор түйінді сөздермен және тілдің қолданылатын стандартты объектілерінің атауларымен сәйкес келмеуі тиіс;

- идентификаторларды астын сызу символымен бастамаған жөн, себебі олар жүйелік функциялардың немесе айнымалылардың атауларымен сәйкес келуі мүмкін, сонымен қатар бұл программаның орындалу жылдамдығын төмендетеді;

- сыртқы айнымалыларды анықтау үшін қолданылатын идентификаторларға жинақтауыштар шектеу қояды (әртүрлі жинақтауыштарды немесе жинақтауыш нұсқаларын қолдану сыртқы айнымалылар атауларына да әртүрлі талаптар қояды).

Түйінді сөздер – бұл компилятор үшін арнайы мағынасы бар алдын ала анықталған (резервтегі) идентификаторлар. Оларды тек өз анықталған мағынасында ғана қолдануға болады.

Операциялар таңбалары, яғни ***белгілері*** – бұл операндтармен орындалатын әрекеттерді анықтайтын бір немесе бірнеше символдар тізбегі. Операция таңбаларының ішіне бос орын белгісі қойылмайды. Құрамындағы операндтар санына байланысты операциялар унарлы, бинарлы, тернарлы болып бөлінеді. Мәтін ішіндегі орнына, яғни қолданылатын ортасына қарай бір таңбаның қызметі әртүрлі болуы мүмкін. Мынадай операциялар [ ], ( ) және ? таңбаларынан басқа белгілер жеке лексема қызметін атқарады.

Тұрақтылар деп программадағы мәні өзгермейтін шамаларды айтады. Олар бүтін, нақты, символдық және тіркестік тұрақтылар болып бөлінеді. Компилятор тұрақтыны лексема ретінде белгілеп алып, оны сыртқы бейнесіне қарай белгілі бір типке жатқызады

Бір символдан тұратын символдық тұрақтылар компьютер жадында бір байт орын алады және стандартты ***char*** типіне жатады. Қос символды тұрақтылар екі байт орын қажет етеді және ***int*** типіне жатады, мұндағы бірінші символ адресі кіші байтта орналастырылады (мәліметтер типтері туралы келесі бөлімде жазылған).

Кері қиғаш сызық белгісі келесі мақсаттарда қолданылады:

- графикалық кескіні жоқ кодтарды (мысалы, \a – дыбыстық сигнал, \n – кур сорды келесі жолдың басына көшіру) бейнелеу үшін;

- апостроф ('), кері қиғаш сызық (\), сұрақ белгісі (?) және тырнақша (")символдарын бейнелеу үшін;

- кез келген символды оналтылық немесе сегіздік кодтардың көмегімен бейнелеу үшін, мысалы, \073, \0xF5. Сандық мәндер 0 мен 255 аралығында болу керек.

Кері қиғаш сызықтан басталатын символдар тізбектерін ***басқару тізбектері*** немесе ***escape-тізбектер*** деп атайды. Жоғарыдағы кестеде олардың мүмкін болатын мәндері берілген. Басқару тізбегі жеке бір символ ретінде қабылданады.

Егер кері қиғаш сызық белгісінен кейін кестеде көрсетілмеген символ кездесетін болса, онда оның нәтижесі белгісіз болып саналады. Ал егер сандар тізбегінде цифрдан өзге символ кездессе, ол сандық кодтың соңы болып есептеледі.

Басқарушы тізбектер тіркестік литералдар деп аталатын тіркестік тұрақтыларда қолданылуы мүмкін. Мысалы, егер тіркес ішіне тырнақша жазу керек болса, оның алдына кері қиғаш сызық қойылады, сол арқылы компилятор бұл тырнақшаның тіркесті шектейтін тырнақшалардан басқа екенін ажырата алады:

"\"Питер\" баспа үйі"

Барлық тіркестік литералдарды компиляторлар әртүрлі объект ретінде қарастырады.

Программада бос орын символдарымен ғана бөлінген тіркестік тұрақтылар компиляция кезінде біртұтас тұрақты болып біріктіріледі. Бір жолға симайтын ұзын тіркестік тұрақтыны кері қиғаш сызық символы арқылы бөле отырып, бірнеше жолдарға орналастыруға болады. Мұндай бөлу символдарын компилятор жұмыс барысында ескермейді де, келесі жол алдыңғысының жалғасы ретінде қабылданады.

Әрбір тіркестік литералдың соңына компилятор \0 басқару тізбегімен берілетін нөлдік символды қосып қояды. Сондықтан тіркес ұзындығы оның жазуындағы символдар санынан әрқашанда бір символға артық болады. Осылайша, " " бос тіркестің ұзындығы 1 байтқа тең болады. Бір символдық тіркес ("A") пен символдық тұрақты ('A') арасындағы айырмашылыққа назар салыңыз. Бос символдық тұрақты қолданылмайды.

Түсініктемелер

Жол соңындағы түсініктемелер екі «тура қиғаш сызық» символдарынан (//) басталып, жаңа жолға көшу символымен аяқталады, ал түсініктеменің екінші түрі /* және */ жақшалық символдар ішіне жазылады. Түсініктемелер ішінде тек C++ тілі алфавитінің символдарын ғана емес, компьютерде теруге болатын кез келген символдарды енгізе беруге болады.

Қабатталған түсініктеме-жақшаларды (/*) пайдалану стандарт бойынша рұқсат етілмейді, бірақ кейбір компиляторда оларды пайдалануға болады.

C++ тіліндегі мәліметтер типтері

Кез келген программаның негізгі мақсаты мәліметтерді өңдеу болып табылады. Әр типтегі мәліметтер әртүрлі жолмен өңделеді және сақталады. Кез келген алгоритмдік тілдегі әрбір тұрақтының, айнымалының, өрнектің немесе функциялардың есептеу нәтижелері белгілі бір типке сәйкес келуі керек.

Мәліметтер типі бойынша төмендегілер анықталады:

- мәліметтердің компьютер жадындағы ішкі бейнесі;
- осы типтегі шамалар қабылдай алатын мәндер жиыны;
- осы типтегі шамаларға қолдануға болатын операциялар мен функциялар

C++ тілінің барлық типтері ***негізгі*** және ***құрама*** болып екіге бөлінеді. C++ тілінде бүтін, нақты, символдық және тіркестік шамаларды бейнелеу үшін алты негізгі мәліметтер типі анықталған. Осы типтерді негізге ала отырып, программалаушы құрама типтер сипаттамасын енгізе алады. Оларға жиымдар, тізбелер, функциялар, құрылымдар, сілтемелер, нұсқауыштар, біріктірмелер және кластар жатады.

Мәліметтердің негізгі типтері

Мәліметтердің негізгі (стандартты) типтерін көбінесе арифметикалық типтер деп те айтады, өйткені оларды арифметикалық операцияларға қолдануға болады.

Негізі типтерді сипаттау үшін келесі түйінді сөздер анықталған:

- int (бүтін);
- char (символдық);
- wchar_t (кеңейтілген символдық);

- bool (логикалық);
- float (нақты);
- double (екі еселенген дәлдіктегі нақты сандар).

Алғашқы төрт тип бүтін сандық (***бүтін***), соңғы екеуі жылжымалы нүктелі типтер (***нақты***) деп аталады. Компилятордың бүтін шамаларды өңдеу кезінде қалыптастыратын коды жылжымалы нүктелі шамаларға арналған кодтан өзгеше болып келеді.

Стандартты типтердің компьютердегі ішкі бейнеленуі мен олардың мәндерінің диапазонын нақтылайтын төрт тип спецификаторы бар, олар:

- short (қысқа);
- long (ұзын);
- signed (таңбалы);
- unsigned (таңбасыз).

Бүтін тип (int)

Компьютер жадындағы ***int*** типіндегі шамалардың алатын орны, яғни ұзындығы немесе өлшемі стандарт бойынша анықталмайды, ол компьютер мен компиляторға байланысты болып келеді.

Осы типтегі шамалар үшін 16 разрядты процессорда 2 байт, ал 32 разрядты процессорда 4 байт бөлінеді.

Тип атауының алдында тұратын ***short*** спецификаторы компиляторға процессордың разрядтылығына қарамастан, санға 2 байт бөлу қажет екендігін білдіреді. Ал ***long*** спецификаторы бүтін шамаға компьютер жадынан 4 байт орын берілетінін көрсетеді. Сонымен, 16 разрядты компьютерде int және ***short int*** типтері бірдей, ал 32 разрядты компьютерде int және long int типтері бірдей (яғни эквивалентті) болып саналады.

Бүтін типтегі шаманың компьютер жадындағы ішкі бейнесі – екілік кодпен берілген бүтін сан. ***signed*** спецификаторын қолданғанда санның алғашқы биті оның таңбасын көрсетеді (0 – оң сан, 1 – теріс сан). ***unsigned*** спецификаторы тек оң сандарды көрсетуге мүмкіндік береді, өйткені мұнда алғашқы разряд сан кодының бөлігі ретінде қарастырылады. Осылайша, ***int*** типінің мәндер диапазоны спецификаторларға тәуелді болып келеді.

Алдын ала келісім бойынша барлық бүтін типтер таңбалы сандарды бейнелейді, яғни ***signed*** спецификаторын жазбауға болады. Программада кездесетін тұрақтылар сыртқы түріне қарай кез келген бір типке жатқызылады. Егер программалаушыға бұл типтен басқаша тип қажет болып жатса, онда ол ***L***, ***l (long)*** және ***U, u (unsigned)*** жалғауларының (суффикстерінің) көмегімен керекті типті көрсете алады.

Мысалы, 32L тұрақтысына long типі беріледі және ол компьютер жадынан 4 байт орын алады. L және U жалғауларын қатар қолдануға да болады, мысалы, 0x22UL немесе 05Lu.

Символдық тип (char)

Символдық типтегі шамаға компьютердегі символдар жиынының ішінен кез келген таңбаны орналастыруға жеткілікті орын бөлінеді, типтің атауы осыған байланысты түрде

берілген. Әдетте, бұл 1 байтқа сәйкес келеді. Басқа да бүтін типтер сияқты, ***char*** типі де таңбалы немесе таңбасыз болуы мүмкін.

Таңбасы бар шамаларда -128 бен +127 аралығындағы мәндерді сақтауға болады. Ал, ***unsigned*** спецификаторын қолданғанда, мәндердің өзгеруі 0 мен 255 сандары аралығында болады. Бұл 256-символдық ASCII кодтары құрамының кез келген символын сақтауға жеткілікті. char типіндегі шамалар осы көрсетілген диапазоннан шықпайтын басқа бүтін сандарды сақтау үшін де қолданыла береді.

Кеңейтілген символдық тип (wchar_t)

wchar_t типі символдарды кодтау кезінде 1 байт жеткіліксіз болатын жағдайларда, мысалы, Unicode ортасында жұмыс істеуге арналған. Бұл типтің ені жүзеге асырылуына тәуелді болады; әдетте ол ***short*** типіне сәйкес келеді. ***wchar_t*** типіндегі тіркестік тұрақтылар ***L*** префиксімен жазылады, мысалы, L"Gates".

Логикалық тип (bool)

Логикалық типтегі шамалар тек ***true*** және ***false*** мәндерін ғана қабылдай алады. ***false*** мәнін бейнелеудің ішкі формасы – 0 (нөл). Кез келген басқа мән ***true*** болып қабылданады. Бүтін типке түрлендірілген жағдайда ***true*** мәні бірге сәйкес келеді. Жылжымалы нүктелі типтер (***fl oat, double, long double***)

C++ тілінің стандарты нақты мәндерді сақтауға арналған мәліметтердің үш типін анықтайды, олар: ***float***, ***double*** және ***long double***. Жылжымалы нүктелі мәліметтер типтері компьютер жадында бүтін санды типтерден басқаша түрде сақталады. Компьютер жадындағы нақты санның ішкі бейнесі 2 бөліктен – мантисса мен дәрежеден тұрады.

Программа құрылымы

C++ тіліндегі программа функциялардан, сипаттамалардан және препроцессор директиваларынан тұрады. Функциялардың біреуінің атауы ***main*** болуы тиіс. Программаның орындалуы осы функцияның бірінші операторынан басталады.

Функцияны анықтаудың қарапайым форматы төмендегідей болады:

қайтарылатын_мән_типі атауы ([параметрлер]){

функция тұлғасын құратын операторлар

}

Көбінесе функция кез келген бір мәнді есептеу үшін қолданылады, сондықтан функция атауының алдында оның типі көрсетіледі.

- Егер функция мән қайтармауы тиіс болса, void типі көрсетіледі;
- функцияның тұлғасы блок болып табылады, сондықтан ол жүйелі жақшаға алынады;
- функциялар қабаттасқан түрде берілмейді;
- әрбір оператор нүктелі үтірмен аяқталады (құрама операторлардан басқасы).

Құрамында ***main***, ***f1*** және ***f2*** сияқты функциялары бар программа мысалы:

Препроцессор директивалары

```
сипаттамалар
int main(){
    негізгі функция операторлары
}
int fl(){
    fl функциясының операторлары
}
int f2(){
   f2 функциясының операторлары
}
```

Программа бірнеше ***модульден*** (бастапқы файлдардан) құралуы мүмкін. Енгізу/шығару туралы бірқатар алдын ала ескертулер. C++ тілінде құрамдас енгізу/шығару құралдары жоқ, ондай операциялар стандартты кітапханалардағы функциялар, типтер және объектілер көмегімен жүзеге асырылады. Көбінесе екі тәсіл қолданылады: C тілінен мұраланған функциялар және C++ тілінің объектілері.

C тілі стиліндегі негізгі ***енгізу/шығару функциялары***:

```
int scanf (const char* format, ...) // енгізу
int printf(const char* format, ...) // шығару
```

Бұлар шамалардың кез келген санын ***format*** форматы тіркесіне сәйкес форматталған түрде енгізу және шығару әрекеттерін орындайды. Формат тіркесінде мәлімет шығару кезінде ағымға (экранға) көшірілетін немесе мәлімет енгізу кезінде ағымнан (пернетақтадан) шақырылатын символдар және % белгісінен басталатын, енгізу және шығару кезінде нақты шамалармен алмастырылатын ***түрлендіру спецификациялары*** болады.

C тілі стиліндегі енгізу/шығару функцияларын пайдаланатын программаның мысалы:

```
#include <stdio.h>
int main(){
   int i;
   printf("Бүтін сан енгізіңіз\n");
   scanf("%d", &i);
   printf("Сіз %d санын енгіздіңіз, рахмет!", i);
   return 0;
}
```

Айнымалылар және өрнектер

Кез келген программада есептеулер жүргізу қажет болады. Мәндерді есептеу үшін операндтардан, операция белгілері мен жақшалардан тұратын ***өрнектер*** қолданылады. Операндтар есептеуге қажетті мәліметтерді береді. Операциялар орындалуы тиіс әрекеттерді анықтайды. Әрбір операнд, өз кезегінде, өрнек немесе оның дербес бір түрі, мысалы, тұрақты

немесе айнымалы болып табылады. Операциялар приориттеріне (басымдылығына) сәйкес орындалады. Операциялардың орындалу ретін өзгерту үшін жай дөңгелек жақшалар қолданылады.

Айнымалы – бұл белгілі бір типтегі мәліметтерді сақтай алатын, атау берілген компьютер жады аймағы. Айнымалылардың аты және мәні болады. Оның аты мәні сақталатын жады аймағын пайдалану үшін қызмет етеді. Программаны орындау барысында айнымалының мәнін өзгертуге болады. Кез келген айнымалыны қолданудың алдында оны сипаттау керек.

Аты a болатын бүтін және аты x болатын нақты айнымалыларды сипаттау мысалы:

int a; float x;

Айнымалыларды сипаттау операторының жалпы түрі:

[жады класы] [const] типі аты [инициалдаушы];

Осы оператордың құрамындағы бөліктерді жазу ережелерін қарастырайық:

- Міндетті түрде көрсетілмейтін жады класы ***auto, extern, static*** және ***register*** мәндерінің біреуін қабылдай алады. Олар туралы төменде айтылады.

- ***const*** модификаторы программада айнымалының мәнін өзгертуге болмайтынын көрсетеді. Мұндай айнымалыны атаулы ***тұрақты*** немесе ***жай ғана тұрақты*** деп атайды;

- Сипаттау кезінде айнымалыға бастапқы мән меншіктеуге болады, бұл ***инициалдау*** деп аталады. Инициализаторды екі формада жазуға болады, алғашқысы – теңдік белгісімен:

= мәні

немесе дөңгелек жақшада:

(мәні)

Тұрақты жариялану кезінде инициалдануы керек. Бір операторда бір типтегі бірнеше айнымалыны үтір арқылы бөле отырып, сипаттауға болады.

Егер инициалдаушы мәннің типі айнымалы типімен сәйкес келмесе, белгілі ережелер бойынша ***типтерді түрлендіру*** орындалады.

Тип пен жады класынан басқа айнымалы сипаттамалары айқын түрде немесе келісім бойынша оның ***әрекет ету аймағын*** береді. Жады класы және әрекет ету аймағы тек сипаттаманың өзіне ғана емес, сонымен қатар оның программа мәтініндегі орналасуына да тәуелді болып келеді.

Идентификатордың әрекет ету аймағы – идентификатормен байланысқан жады аймағын пайдалану (онымен қатынас құру) үшін осы идентификатордың өзін қолдануға болатын программа бөлігі. Әрекет ету аймағына байланысты айнымалы жергілікті (локалды) немесе ауқымды (глобалды) болуы мүмкін.

Егер айнымалы блок ішінде анықталған болса (блок жүйелі жақшамен қоршалып тұрады), ол ***жергілікті*** айнымалы болып табылады да, оның әрекет ету аймағы сипатталу нүктесінен блок соңына дейінгі аралықты қамтиды. Ал егер айнымалы кез келген блоктан тыс анықталған болса, ол ***ауқымды*** айнымалы болып саналады да, оның әрекет ету аймағы болып осы айнымалы анықталған сипаттау нүктесінен файл соңына дейінгі аймақ есептеледі.

Жады класы программалық объектінің (жеке жағдайда, айнымалының) өмірлік уақытын, яғни пайдаланылу кезеңін және көріну аймағын анықтайды. Егер жады класы айқын түрде көрсетілмесе, оны компилятор жариялану мәтініне тәуелді түрде (контекстіне байланысты) анықтайды. Айнымалының өмірлік кезеңі (уақыты) тұрақты (программаны орындап біткенше) және уақытша (блокты орындап біткенше) болуы мүмкін.

Идентификатордың көріну аймағы деп идентификатормен байланысты жады аймағына қарапайым қатынас құру мүмкін болатын программа мәтінінің бөлігін айтады. Көбінесе идентификатордың көріну аймағы оның әрекет ету аймағымен бірдей болып келеді. Мұндағы ерекше жағдай – ішкі блокта да дәл осындай атпен сипатталатын айнымалы болған кезде туындайды. Мұндайда сыртқы айнымалы ішкі блоктың әрекет ету аймағына кіргенменен, ол онда, яғни ішкі блокта көрінбейді. Дегенмен, егер бұл айнымалы ауқымды болса, онда оның көріну аймағына қатынас құру операциясын :: пайдалана отырып, оны қолдана беруге болады.

Жады класын беру үшін келесі спецификаторлар қолданылады:

- ***auto*** – автоматтық айнымалы. Ол үшін жады стектен бөлінеді және қажет болғанда айнымалының анықтамасы орналасқан оператордың әрбір орындалуы кезінде ол инициалданып отырады. Жадыны босату айнымалы сипатталған блоктан шығатын кезде орындалады. Оның өмірлік кезеңі – сол айнымалы сипатталған сәттен блок соңына дейінгі аралық болып табылады. Ауқымды айнымалылар үшін бұл спецификатор қолданылмайды, ал жергілікті айнымалылар үшін ол алдын ала келісім бойынша қабылданады, сондықтан оны айқын түрде берудің онша қажеті жоқ.

- ***extern*** – айнымалы программаның басқа жерінде (басқа файлда немесе мәтіннің төменгі бөлігінде) анықталатынын көрсетеді. Олар өздері жарияланған программаның барлық модульдерінде қолдануға болатын айнымалыларды құру үшін қолданылады

- ***static*** – статикалық айнымалы. Өмірлік кезеңі – тұрақты. Ол айнымалының анықтамасы орналасқан оператордың алғашқы орындалуы кезінде бір рет инициалданады. Сипаттау операторының орналасуына байланысты статикалық айнымалылар жергілікті немесе ауқымды болуы мүмкін. Ауқымды статикалық айнымалылар өздері сипатталған модульде ғана көрінеді (яғни пайдаланылады).

- ***register*** – auto спецификаторына ұқсас, бірақ мұнда айнымалыға, мүмкіндігінше, процессор регистрлерінен орын (жады) бөлінеді. Егер компиляторда мұндай мүмкіндік жоқ болса, айнымалылар auto сияқты өңделеді.