ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ - СОФИЯ

ИНФОРМАТИКА
част първа

лектор: ас. д-р Фани Томова
лекции: доц. д-р Атанас Атанасов
Катедра “Информатика”

Лекция 9

ИНСТРУКЦИИ В ЕЗИКА ЗА ПРОГРАМИРАНЕ С++

Инструкции в езика C++

- **Въведение - инструкции в C++ -**
- **Инструкция за присвояване**
  - предназначение
  - синтаксис
  - изпълненние
  - особености
- **Инструкция за въвеждане на информация**
- **Инструкция за извеждане на информация**
- **Описание на променливите програмата**
- **Празна инструкция**
- **Съставна инструкция (блок)**

Въведение –инструкции в C++

Инструкцията е предписание към компютърната система за изпълнение на точно определено действие, написано на език разбираем за нея.

В много литературни източници се използва понятието **оператор** вместо **инструкция**. Тук ще използваме понятието инструкция, защото по-точно изразява нейното предназначение.

Инструкциите, които се използват в C++, могат да бъдат за описание на обектите в една програма, за реализиране на изчисления и за осъществяване на входно-изходни операции.

Друга група от инструкции са предназначени за управление на изчислителния процес- реализиране на условни и безусловни преходи в една програма, организация на цикли, избор на варианти и управление на динамичната памет.

Инструкция за присвояване

-**Предназначение**.

Използва се за пресмятане на изрази от аритметичен, логически и символен тип.

Инструкцията за присвояване е предписание към компютърната система за изчисляване на израз и присвояване на получената стойност на дадена променлива.

-Синтаксис.

Общият вид на инструкцията за присвояване в езика C++ е следния: **<идентификатор> = <израз>;**

<идентификатор>, това името на проста променлива (от скаларен тип) или на променлива с индекс.

Знакът “**=**” е знак за присвояване, а **<израз>** е изразът, който трябва да се пресметне.

В частност изразът може да бъде константа, променлива или функция.

Инструкция за присвояване

Примери:

```
x1=(-b+ sqrt(d))/(2*a);
alfa=5.68;
x2=x1;
z=fabs(x1);
x=x+1;
```

От последния пример се вижда динамичния характер на инструцията за присвояване.

Променливата **x**, която е от дясната страна е стойността на промеливата преди да се изпълни инструкцията, а x от лявата страна е новата стойност на същата променлива след изпълнение на инструкцията.

Това означава, че знакът за присвояване “=” в C++, не е еднакъв със знака за равенство в математиката.

Инструкция за присвояване

-Изпълнение.

Физическият смисъл на тази инструкция е следният: Пресмята се изразът и получената стойност се присвоява на променливата, чийто идентификатор е записан от ляво на знака за присвояване.

На практика това означава получената стойност да се запише в тази клетка от паметта, която вече е отделена за променливата.

-Особености.

А. Всички променливи, участващи в израза трябва да са получили стойности до момента на изпълнение на инструкцията.

Ако това не е изпълнено, компилаторът не дава съобщение за грешка, но резултатите от изпълнението на инструкцията няма да са верни.

Инструкция за присвояване

Б. Ако в лявата страна стои променлива с индекс (елемент на масив) е необходимо всички променливи в индексният израз/променлива да са получили стойност, за да може да се локализира мястото в паметта, отделено за тази променлива с индекс.

В. При изпълнението на инструкцията за присвояване водещ е типът на лявата част, което означава, че изчислената стойност автоматично се преобразува до типа на променливата записана вляво.

Инструкция за въвеждане на информация

-Предназначение.

Това е инструкция за въвеждане на данни от клавиатурата в паметта на компютъра.

-Синтаксис.

Общият вид на инструкцията е следният:

cin >> <променлива> {>>променлива}; ,

където: **cin** е обект (променлива) от клас **iostream**, който е свързан с клавиатурата (входният поток от данни идва от клавиатурата).

>> - Операция за вход (въвеждане на данни);

<променлива> - това е идентификатор на променлива от допустим тип **(int, long, float, double**...), която вече е дефинирана и за нея трябва да въведе стойност от клавиатурата.

{>> <променлива>} С една инструкция за въвеждане могат да се въведът стойности за повече променливи.

Инструкция за въвеждане на информация

Примери:

cin>>a; cin>>b; cin>>c; или

cin>>a >>b>>c;

Операцията за вход **>>** се изпълнява от ляво на дясно, т.е. тя е лявоасоциативна.

-Изпълнение.

Инструкцията за въвеждане извлича чрез **cin** от клавиатурата поредната стойност и я прехвърля в оперативната памет в клетка, която вече е отделена за съответната променлива.

Изпълнението на инструкцията за въвеждане спира програмата, докато не бъдат въведени от клавиатурата толкова данни, колкото променливи фигурират в инструкцията.

Инструкция за въвеждане на информация

При въвеждането на данните от клавиатурата те се отделят една от друга с празен интервал, нов ред или табулация.

Въвежданите данни трябва да съответстват по тип на променливите, за които са предназначени.

Процесът на въвеждане на данните приключва с натискане на клавиша Enter и ако въведените данни са достатъчни програмата продължава своето изпълнение.

Пример:

Даден е следният фрагмент от програма:

```
double x1, x2, x3;
cin>>x1>>x2>>x3;
```

Неговото изпълнение изисква въвеждането от клавиатурата на три реални числа, които да се свържат с променливите **x1,x2,x3**, което означава да се запишат в паметта в отделените вече клетки.

Инструкция за въвеждане на информация

Това въвеждане може да стане по няколко начина:

1.) **1.1 2.2 3.3 <Enter>**

2.) **1.1 <Enter>**
2.2 <Enter>
3.3 <Enter>

3.) **1.1 <Enter>**
2.2 3.3 >Enter>

4.) **1.1 2.2 <Enter>**
3.3 <Enter>

И четирите случая на въвеждане на данни за x1, x2, x3 ще водят до един същи резултат: В паметта в клетките за x1 ще се запише 1.1, 2.2 за x2 и 3.3 за x3.

Инструкция за въвеждане на информация

Допустим е и следният случай на въвеждане на данни:

1.1 2.2 3.3 4.4 <Enter> В този случай последната стойност (**4.4**) ще остане необработена в буфера на клавиатурата и ще се използва от следващата инструкция за въвеждане. Това не се препоръчва, защото може да доведе до грешки. Когато една инструкция за въвеждане е приключила успешно, то променливата **cin** приема стойност **true**, в противен случай ще приеме стойност **false**.

Пример: Даден е следният програмен фрагмент:

```
int x1;        cin>>x1;
```

Ако при неговото изпълнение за **x1** вместо цяло число въведем: tri <Enter>, то в този случай съдържанието на клетката **x1** от паметта не се променя, а буферът на клавиатурата ще е в състояние **fail**, т.е. cin има стойност false, но системата не извежда съобщение за грешка.

Инструкция за извеждане на информация

-Предназначение.

Това е инструкция за извеждане на информация от оперативната памет на компютъра върху дисплея. Предвидена е възможност за извеждането на информацията по подходящ начин с използването на манипулатори.

-Синтаксис.

Общият вид на инструкцията за извеждане е следният:

cout << <израз> {<< <израз>};

където: << -оператор за изход;

cout – е обект (променлива) от клас **iostream** (изходен поток), който предварително е свързан с екрана.

<израз> – е израз от допустим за C++ тип (**bool, char, float, int, long, double,...**). Ако изразът е текстов трябва да се загради в кавички. С една инструкция за извеждане могат да се извеждат повече от един изрази.

Инструкция за извеждане на информация

-Изпълнение.

Операторът за изход **“<<”** изпраща към екрана стойността на израза.

Ако изразът е променлива, това означава, че на екрана трябва да се изведе нейната стойност, която се съхранява в паметта. Ако изразът е от аритметичен или логически тип, то той трябва да се пресметне и на екрана да се изведе неговата стойност. Ако е текстов израз (на кирилица или латиница), то той се извежда без обработка върху екрана. За форматирано извеждане на информация се използват следните манипулатори:

1.) setw(<целочислен израз>) – Стойността на целочисления израз задава широчината на полето за следващия изход. Пример:

```
int i=256, j=5678;
cout <<”i=” <<setw(6) << i <<”  j=” <<setw (8) << j;
```

Инструкция за извеждане на информация

В резултат на изпълнението на тази инструкция за извеждане на екрана ще се появи следната информация:

i= 256 j= 5678

Стойността на променливата е дясно ориентирана в полето, което е предварително специфицирано.

2.) setprecision(<целочислен израз>) - Стойността на израза задава броя на всичките позиции на реалното число (за цялата и за дробната част). Манипулаторът е в сила за всички извеждания след него. Пример:

```
double a=25.576, b=-562.425;
cout<<"a="<<setprecision(4)<<setw(8)<<a<<"
b="<<setw(8)<<b;
```

Изпълнението на инструкцията води до:

a= 25.58 b= -562.4, като **a** и **b** се извеждат в полета с широчина по 8 позиции с точност 4 цифри. Стойностите се закръглят по математически правилата.

Инструкция за извеждане на информация

3.) setiosflags(ios::fixed) – Използването на този манипулатор заедно със **setprecision** задава точността на извеждане за дробната част на реалните числа. Пример:

```
double a=25.5768, b= -562.4256;
cout<< setprecision(3) <<setiosflags(ios::fixed) ;
cout<<"a="<<setw(10)<<a <<"   b="<<setw(10)<<b;
```

След изпълнението на този фрагмент, на екрана ще се изведат стойностите на **a** и **b** закръглени до третия знак след десетичната точка.

a= 25.577 b= -562.426

4.) dec, oct, hex– Тези манипулатори задават, че всички следващи извеждания на цели числа ще бъдат в десетична, осмична или шестнадесетична бройна система.

Всеки от манипулаторите **dec, oct, hex** е в сила, докато не бъде отменен от друг..

Описание на променливите в програмата

Описанието на променливите в една програма на C++ е задължително и изпълнява следните функции:

-Запазва клетки в паметта за съответните променливи;

-В зависимост от типа данни, които приема дадена променлива, определя големината на отделената и клетка в байтове. Използват се два вида описания на променливите - с и без инициализация.

<тип> <списък от идентификатори>;

<тип> <идентификатор> = <израз>{,<идентификатор>=<израз>};

Примери:

```
int  k,l,m;
char p,f,q;
float x1,x2;
double z1=21.425, z2=3.24;
```

Описание на променливите в програмата

Тези описания предизвикват следните действия:

Запазват 3 клетки, всяка една с големина 2 байта (за 32-битов компилатор), в които ще се запишат стойностите на промеливите **k,l,m.**

След това ще се запазят още 3 клетки, всяка една с големина 1 байт, в които ще се записват стойностите на символните промеливи **p,f,q.**

След тях ще се запазят още 2 клетки за реалните променливи x1 и x2, всяка една с големина по 4 байта. Накрая ще се запазят и 2 клетки за реалните променливи z1 и z2, всяка една с големина по 8 байта и в първата ще се запише числото 21.425 за z1, а във втората 3.24 за z2. Променливите z1 и z2 се наричат инициализирани променливи.

С разгледаните до тук инструкции могат да се пишат програми на C++, реализиращи алгоритми с линейна структура:

Описание на променливите в програмата

Пример-1: *Да се състави програма на C++ за пресмятане на обема и пълната повърхнина на призма със страни на основата* ***a*** *и* ***b*** *височина* ***h.*** За пресмятането на обема и повъхнината на призмата са използвани следните зависимости:

V=a.b.h
S=2.(a+b).h +2.a.b

```
//Program  Pr2.cpp
# include<iostream>
using namespace std;
int main()
{ // Opisanie na promenlivite
double a,b,h,s,v;
// Vavejdane на stranite na osnovata
cout <<"a=";   cin >> a;
cout <<"b=";   cin >> b;
// Vavejdane na visochinata
cout <<"h=";   cin >> h;
/* Presmiatane na povarhninata i
obema na prizmata */
s=2*(a+b)*h+2*a*b;
v=a*b*h;
/* Izvejdane na rezultata za S i V */
cout <<"S=" << s  <<" V=" << v << "\n";
return 0;
}
```

Описание на променливите в програмата

Пример-2: *Да се състави програма за пресмятане обема и пълната повърхнина на цилиндър с радиус на основата* ***r*** *и височина* ***h*** *и теглото на цилиндъра, ако е направен от материал със специфично тегло* ***c.***

За извършване на изчисленията са използвани следните зависимости:

S=π.r2

V=S.h

Sp= 2.S + 2.π.r.h

G=V.c/1000

Стойностите за **r** и **h** са в см., а **c** е в кг. за куб.дециметър и това налага обемът да се превърне от куб. см в куб.дм., като се раздели на 1000.

```
//Program Pr3.cpp
#include <iostream>
#include <math.h>
using namespace std;
double const PI=3.141557;
int main()
{   double r,h,s,sp,v,g,c;
 /* stojnostite na r , h sa v sm, a za c sa v kg
*/
  cout <<"r, h, c=";
  cin >>r>>h>>c;
   s=PI*pow(r,2);
   sp=2*s+2*PI*r*h;
   v=s*h;
   g=v*c/1000;
   cout <<"******REZULTATI******" <<"\n";
   cout <<"Sp="<<sp <<" kv.sm.\n";
   cout <<"V=" <<v <<" kub.sm. \n";
   cout <<"G=" <<g <<" kg.\n";
   return 0;
}
```

Описание на променливите в програмата

Резултатите от изпълнението на програмата са:

r, h, c=4.56 10.00 7.68
********REZULTATI********
Sp=417.331 kv.sm.
V=653.513 kub.sm.
G=5.019 kg.

Празна инструкция

-**Синтаксис**.

Общият вид на инструкцията е следният:

< Празна инструкция> ::= < ; >

Празната инструкция не съдържа никакви символи и завършва със знака ”;”.

Използва се в случаите, когато синтаксиса на една инструкция изисква на определено място да присъства поне една инструкция, а логиката на задачата не го изисква.

Записването на няколко празни оператора един след друг е допустимо.

Например:

x1= -b/(2*a); ; ; ;

Този запис не води до грешка.

Съставна инструкция

Съставната инструкция е последователност от група от инструкции, заградени с големи скоби. В езика C++ това се нарича **блок**.

```
{
        S1;
        S2;
        S3;
        .......
        Sn;
}
```

където **S1,S2.....Sn** са инструкции.

Пример: Размяна на стойностите на A и B.

```
{
        C = A;
        A = B;
        B = C;
}
```

Съставна инструкция

- Съставната инструкция (блокът) се използва в случаите, когато синтаксисът на една инструкция изисква на определено място да присъства една инструкция, а логиката на задачата изисква присъствието на последователност от група от инструкции.
- Дефинициите, направени вътре в един блок са валидни само вътре в него.
- Желателно е скобите, които ограждат блока да са една под друга, защото това дава по-голяма прегледност на програмата.
- За разлика от другите инструкции, съставната (блокът) не завършва със знака “ **;** “.
- Блоковете се използват много често в условните инструкции и в инструкциите за организация на цикли.