ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ - СОФИЯ

ИНФОРМАТИКА
част първа

лектор: доц. д-р Атанас Атанасов
Катедра “Програмиране и използване на компютърни системи”

Лекция 10

ИНСТРУКЦИИ В ЕЗИКА ЗА ПРОГРАМИРАНЕ С++

продължение

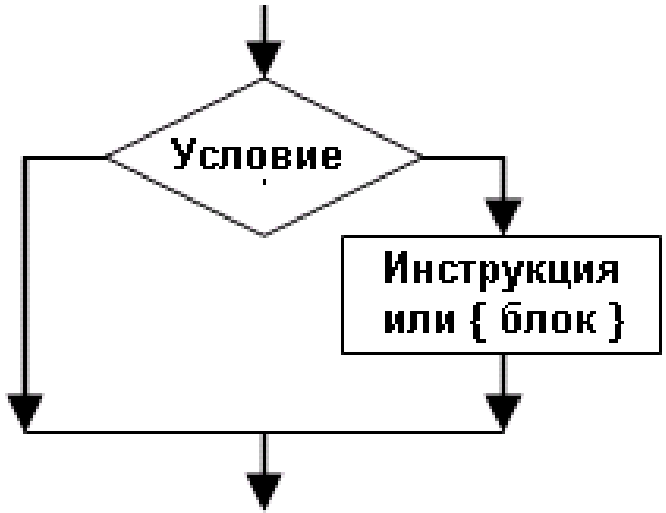
Инструкции в езика C++

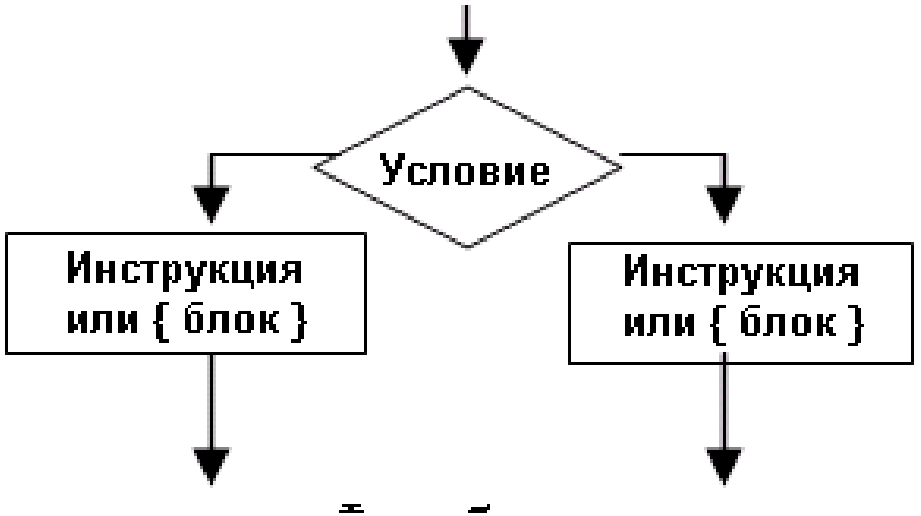
- Инструкции за условен преход
  - Условна инструкция “непълен if”
  - Условна инструкция “пълен if”
  - Вложени условни инструкции
- Инструкция за избор на вариант (switch/case)
- Инструкции за организация на цикли
  - Инструкция за цикъл с предусловие (while)
  - Инструкция за цикъл със следусловие (do - while)

Инструкции за условен преход

Инструкциите за условен преход дават възможност да се изпълни една или друга инструкция в зависимост от изпълнението или неизпълнението на дадено условие. С помощта на тези инструкции се извършват разклонения в изчислителният процес.

В езика C++ се използват две инструкции за условен преход (за непълен if и пълен if-else условен преход).

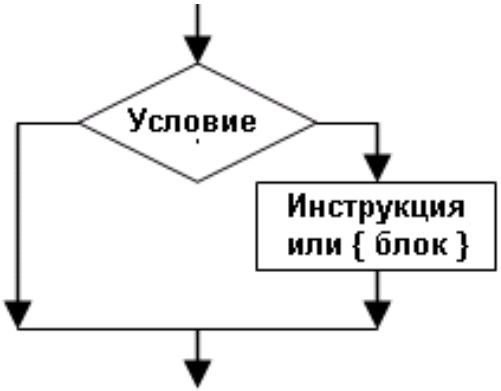




Условна инструкция “непълен if”

- Предназначение.

Използва се за организиране на разклонения в една програма на C++.



- Синтаксис.

if (<условие>) <инструкция>;

Където:

if - служебна дума в C++ и означава “**Ако**”;

< условие> - Това е условието за разклонение и по начин на записване е израз от логически тип (**bool**). Допуска се да бъде и аритметичен израз, но не се препоръчва, защото програмата става по-неразбираема за програмисти и потребители;

<инструкция> - Произволна инструкция от C++. Допуска се да бъде и съставна инструкция (блок).

Условна инструкция “непълен if”

-Изпълнение.

Пресмята се логическия израз (условието) и **ако** стойността му е **true** се изпълнява инструкцията, записана след него.

Ако стойността му е false, инструкцията се пропуска. Когато условието е записано като аритметичен израз, който приема стойност различна от нула се изпълнява инструкцията, а ако е 0, се пропуска.

-Особености:

-Логическият израз задължително се загражда в скоби;

-След логическия израз се записва една инструкция, а ако задачата изисква да са няколко е нобходимо да се обединят в блок (да се оформят като една съставна инструкция).

Условна инструкция “непълен if”

Пример.

Да се състави програма за пресмятане на разликата между максималното и минималното от три предварително зададени числа ***a, b, c****.*

Максималното число ще означим с **max**, а минималното с **min**, а разликата между тях с **raz.**

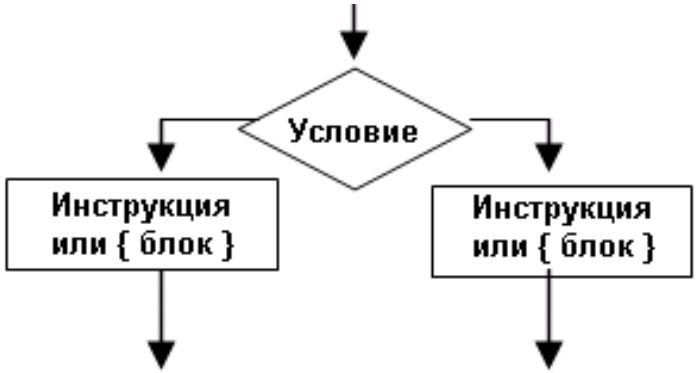
raz = max – min

```
#include <iostream.h>
int main()
{
  double a,b,c,max,min,raz;
  cout<<"Vavedete a,b,c=";
  cin>>a>>b>>c;
  max=a;
  if (b>max) max=b;
  if (c>max) max=c;
  min=a;
  if (b<min) min=b;
  if (c<min) min=c;
  raz=max-min;
  cout<<"max="<<max <<"\n";
  cout<<"min="<<min <<"\n";
  cout<<"razlika="<<raz;
  return 0;
 }
```

Условна инструкция “пълен if”

-Предназначение.

Използва се за организиране на разклонения в програмата.



-Синтаксис.

If (условие) <инструкция - 1> else < инструкция - 2>;

Където:

if , else– служебни думи в езика C++ (**Ако , Иначе**).

<условие> - Това е условието за разклонение и по начин на записване е израз от логически тип (**bool**). Допуска се да бъде и аритметичен израз, но не се препоръчва, защото програмата става по-неразбираема за програмисти и потребители;

<инструкция-1> и **<инструкция - 2>** - Произволни инструкции от C++. Допуска се да бъдат и съставни инструкции.

Условна инструкция “пълен if”

-Изпълнение.

Пресмята се логическият израз (условието) и ако той приеме стойност **true** се изпълнява инструкция-1, в противен случай (ако приеме стойност false), инструкция-1 се пропуска и се изпълнява инструкция-2.

Когато условието е записано като аритметичен израз, който приема стойност различна от нула, се изпълнява инструкция-1, а ако е 0 се изпълнява инструкция-2.

-Особености:

- Логическият израз трябва да бъде напълно опеделен;
- Логическият израз (условието) задължително се загражда в скоби;
-След условие и след **else** се записва по една инструкция, ако задачата изисква да са повече е необходимо те да се обединят в блок .

Условна инструкция “пълен if”

Пример

Да се състави програма за пресмятане кореният на пълно квадратно уравнение от вида: $A.x^2 + B.x + C = 0$

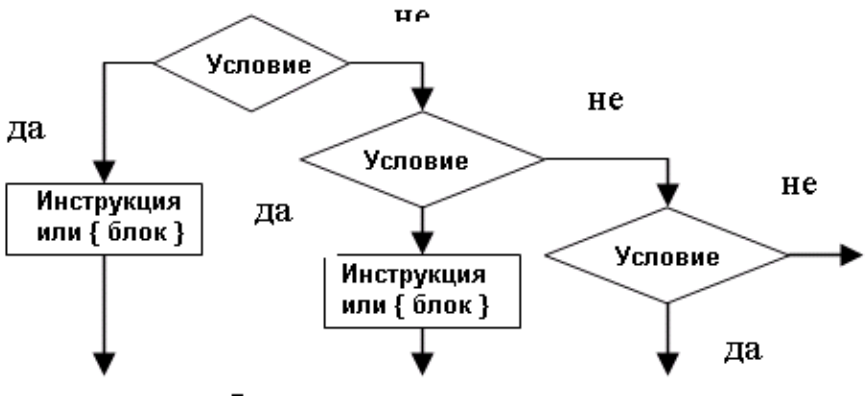
Алгоритъмът за решаване на квадратното уравнение е даден на фиг.1.11 в глава 1.

Реалните корени са означени с **x1** и **x2** , а реалната и имагинерната част на комплексните корени са означени с **re** и **im.**

```
#include <iostream.h>
#include <math.h>
void main()
{ double a,b,c,x1,x2,re,im,d;
   cout <<"a=";  cin >>a;
   cout <<"b=";  cin >>b;
   cout <<"c=";  cin >>c;
   d=pow(b,2)-4*a*c;
    if (d<0)
    {  re= -b/(2*a);
       im= sqrt(fabs(d))/2*a;
       cout <<"*Kompl. koreni*" <<"\n";
       cout <<"Real. chast="<<re <<"\n";
       cout <<"Imag. chast="<<im <<"\n";
     } else {
        x1=(-b+sqrt(d))/(2*a);
        x2=(-b-sqrt(d))/(2*a);
        cout <<"*Realni koreni***" <<"\n";
        cout <<"x1="<<x1 <<endl;
        cout <<"x2="<<x2 <<endl;
     }
}
```

Вложени условни инструкции

Това са условни инструкции, които съдържат в себе си други условни инструкции.



-Синтаксис:

if (<условие>) <инструкция

if (<условие>) <инструкция 1> else <инструкция 2>;

Това означава, че инструкция, инструкция 1, инструкция 2 са произволни инструкции (вкл. и условни инструкции) и в този случай се получават вложени условни инструкции. При образуване на вложени условни инструкции е в сила следното правило: В една условна инструкция всяко **else** се съчетава с най-близкия **if,** който е преди него.

Препоръка: Една условна инструкция да се влага в друга условна инструкция само след **else.** Ако се налага да се вложи след условието, то тя да се оформи като блок.

Условна инструкция “пълен if”

Пример

Да се пресметне стойността на функцията ***y****.*

y=x при x < 2

y=2 при x Э (2,3)

y=x-1 при x>3

```
#include <iostream.h>
int main()
 {
   double x,y;
    cout<<"x=";
    cin>>x;
    if (x<=2) {
       y=x;
    }
    else
   {
              if (x<=3) y=2;
              else y=x-1;
    }
    cout<<"**REZULTATI** \n";
    cout<<"x="<<x <<endl;
    cout<<"y="<<y <<endl;
    return 0;
 }
```

Условна инструкция “пълен if”

Пример.

Да се пресметне площта s на триъгълник със страни a, b и c по Хероновата формула. Да се определи дали той е равностранен, равнобедрен или разностранен.

Ако a, b и c не са страни на триъгълник , да се отпечата съобщение.

```
#include <iostream.h>
#include <imath.h>
void main()
 {  double a,b,c,p,s;
    cout << “a,b,c=";
    cin >>a>>b>>c;
    if ((a+b>c)&&(b+c>a)&&(c+a>b))
    {   cout<<“ a,b,c strani na triag \n”;
        p=(a+b+c)/2;
        s=sqrt(p*(p-a)*(p-b)*(p-c));
       cout<<“ Plosht s = “<<s<<endl;
        if ((a==b)&&(b==c))
        cout<<“ravnostranen \n”;
        else
           if ((a==b)||(b==c)||(c==a))
        cout<<“ravnobedren \n”;
        else cout<<“raznostranen \n”;
    } else  cout<<“a,b,c ne sa strani na
triag \n”;
}
```

Инструкция за избор на вариант

-Предназначение.

Използва се за избор на един от няколко предварително специфицирани варианти.

-Синтаксис.

switch (< израз>)

{ case <израз-1> : S1;
case <израз-2> : S2;
……
case <израз-n> : Sn;
} **,к**ъдето:

switch и **case** – служебни думи (**превключи, случай**).

<израз> - израз от точен тип (bool, int, char, enum);

израз-1, израз-2,… – константни изрази, с различни стойности, от тип който е съвместим с израз (наричат се етикети).

S1, S2,…,Sn – инструкции (прости или съставни);

Инструкция за избор на вариант

-Изпълнение.

Пресмята се **<израз>** и се сравнява последователно с етикетите **израз-1, израз-2,…..,** и се преминава към изпълнение на тази инструкция, чийто етикет съвпада с изчислената стойност на **<израз>**, а също така и всички инструкции, записани след нея, докато не се срещне инструкцията **break**.

Тази инструкция често се прилага при програмиране на менюта, при които потребителят на програмата трябва да въведе/избере цифра или буква, за да активира една или друга функционалност.

Например за активиране на някоя опция от меню лентата на произволен прозорец в Windows, потребителят трябва да селектира File, Edit, View, Insert, ... с мишката или с клавишна комбинация, което на ниво C++ програма е избор ‘F, за File, ‘E’ за Edit и т.н.

Условна инструкция “пълен if”

Пример.

Да се напише програма на C++, която да въвежда цифра, а след това да я изведе като текст.

```
# include<iostream.h>
void main()
 {  int i;
    cout<<"i=";  cin>>i;
    switch(i)
    { case 0 : cout<<"Nula \n"; break;
      case 1 : cout<<"Edno \n"; break;
      case 2 : cout<<"Dve \n"; break;
      case 3 : cout<<"Tri \n"; break;
      case 4 : cout<<"Chetiri \n"; break;
      case 5 : cout<<"Pet \n"; break;
      case 6 : cout<<"Shest \n"; break;
      case 7 : cout<<"Sedem \n"; break;
      case 8 : cout<<"Osem \n"; break;
      case 9 : cout<<"Devet \n"; break;
     default: cout<<"Nekorektni vhodni
              danni ! \n";
     }
} // krai na programata
```

Инструкции за организация на цикли

Под “цикъл” в програмите, написани на C++ се разбира *многократно повторение на една инструкция или група от инструкции (блок), за различни стойности на една променлива.*

Тази променлива е възприето да се нарича параметър на цикъла или управляваща променлива.

Циклите се разделят на две групи:

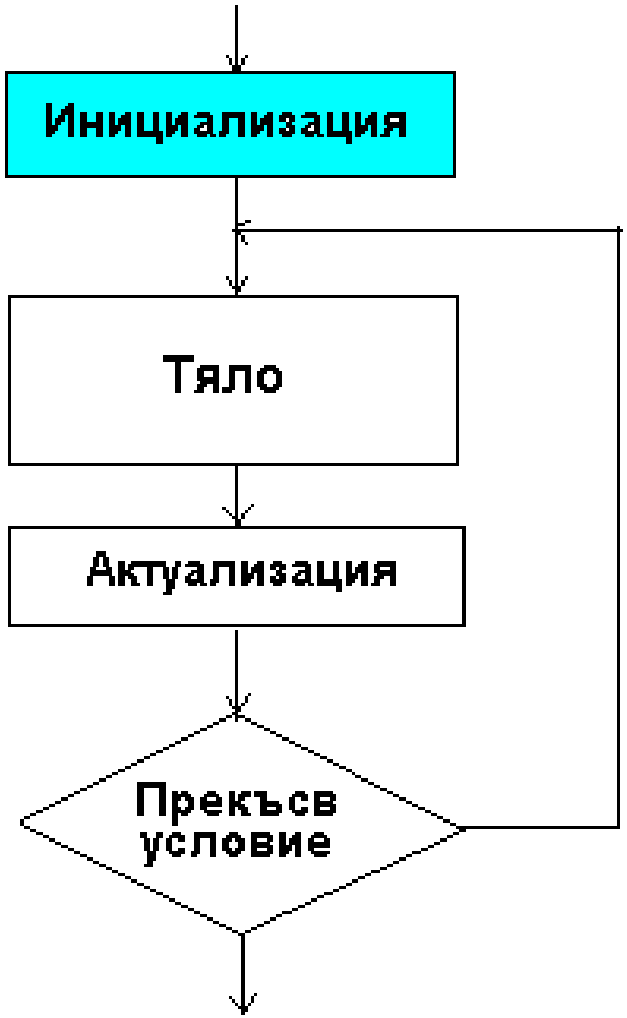
-**Индуктивни** - Това са цикли, при които броя на повторенията на инсрукциите в цикъла е предварително известен.

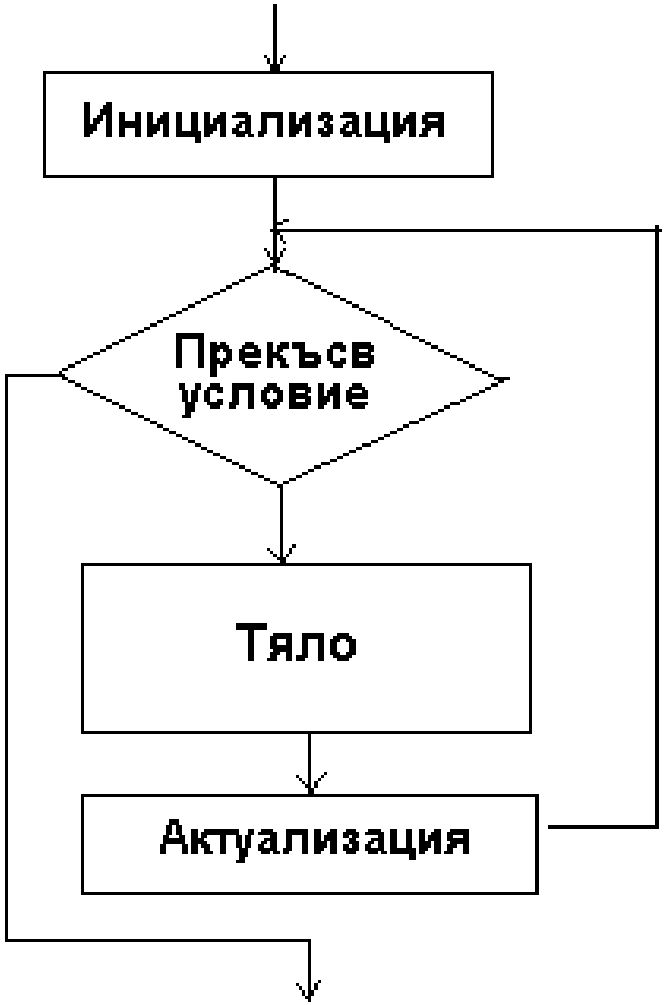
-Итеративни – Това са цикли, при които броя на повторенията на инсрукциите в цикъла е предварително известен. Условието за излизане от цикъла се формира вътре в него.

Циклите се делят още и на цикли с предусловие и цикли със следусловие.

Инструкции за организация на цикли

Блокови схеми на цикли със следусловие и цикли с предусловие.





Инструкция за цикъл с предусловие

-Предназначение.

Тази инструкция е удобна за организация, както на цикли от индуктивен, така и от итеративен тип.

-Семантика:

while (<условие>) <инструкция>;

Където:

while – Служебна дума на езика, която означава **”докато”;**

<условие> Това е условието за излизане от цикъла. То определя, кога трябва да се прекрати многократното повторение на тялото на цикъла (инструкцията). По начин на записване условието е логически израз. По изключение се допуска да бъде и аритметичен израз, но не се препоръчва, защото програмата става по-неразбираема.

<инструкция> - това е инструкцията, която тябва да се повтаря многократно. Може да бъде проста или съставна инструкция (блок). Нарича се още **тяло** на цикъла.

Инструкция за цикъл с предусловие

-Изпълнение.

Пресмята се условието и ако то приеме стойност **true** се изпълнява инструкцията (тялото на цикъла), записана след него. След което отново се пресмята условието и ако е **true** пак се изпълнява инструкцията. Това продължава, докато условието приеме стойност **false**, тогава се излиза от цикъла, т.е. прекратява се многократното изпълнение на инструкцията (тялото на цикъла).

-Особености:

-Ако условието приеме стойност **false**, още при първото пресмятане, тялото на цикъла няма да се изпълни нито един път (затова се нарича предусловие);

-Всички променливи, участващи в записа на условието, трябва да са получили стойност до момента на изпълнението на инструкцията за цикъл;

-Ако след условието, за всяка стойност на параметъра на цикъла трябва да се изпълняват повече от една инструкции, то те задължително се оформят като блок.

Инструкция за цикъл с предусловие”

Пример :

Да се пресметне изменението на тока в една ел. верига със съпротивление R и индуктивност L, след подаването на постоянно напрежение U на входа на веригата.

Изчисленията на тока да започнат от време t=0, и да се извършват през интервал от време Δt *и да се прекратят когато токът I достигне стойност 95% от стойността на тока Io.* Задачата се реализира с цикличен процес от итеративен тип.

```
# include<iostream.h>
# include<math.h>
int main()
 {double r,l,u,i,i0,t,dt,tau;
  cout<<"U="; cin >>u;
  cout<<"R="; cin >>r;
  cout<<"L="; cin >>l;
  cout<<"dt="; cin >>dt;
  i0=u/r;  tau=l/r;
  t=0.0;  i=0.0;
  while (i<=0.95*i0)
      {
         t=t+dt;
         i=i0*(1-exp(-t/tau));
         cout<<"t="<<t <<"  I="<<i<<"\n";
       }
  return 0;
 }
```

Инструкция за цикъл със следусловие

-Предназначение.

Тази инструкция също е удобна за реализация на цикли от индуктивен и от итеративен тип. Нарича се инструкция за цикъл със следусловие, защото първо се изпълнява тялото на цикъла, а след това се проверява условието за излизане от цикъла.

-Синтаксис.

do <инструкция>
while (<условие>);

Където:

do и **while** - служебни думи в езика (прави и докато);

<инструкция> - Това е инструкция или група от инструкции, оформени като блок, която ще се повтаря многократно,

<условие>- Това е условието за излизане от цикъла, аналогично на условието в цикъла с предусловие.

Инструкция за цикъл със следусловие

-Изпълнение.

Изпълнява се инструкцията (тялото на цикъла) и след това се пресмята условието. Ако условието приеме стойност **true** отново се изпълнява тялото на цикъла и това се повтаря, докато условието приеме стойност **false**. Тогава изпълнението на инструкцията **do/while** завършва.

-Особености:

-Ако условието приеме стойност **false**, още при първото пресмятане, то тялото на цикъла ще се изпълни един път и тогава се излиза от инструкцията за цикъл;

-Всички променливи, участващи в записа на условието, трябва да са получили стойност до момента на изпълнението на инструкцията за цикъл **do/while**.

-Инструкцията **do/while** завършва със символа “**;**”.

Инструкция за цикъл със следусловие

Пример.

Да се пресметне скоростната константа на една химическа реакция, като се използва зависимостта на Арениус:

*K=Z*exp(-E/(R*T)) , за различни стойности на температурата T от Тн до Ткр. със стъпка ΔT.*

Примерът е от циклични процеси с индуктивен характер, защото броят на пресмятанията на K за стойности на T от Тн до Ткр. със стъпка ΔT е предварително известен.

```
# include <iostream.h>
#i nclude <math.h>
double const R=8.314;
int main ()
{double t,tn,tk,dt,z,k,e;
  cout<<"Vavedete stoinosti za z,e=";
  cin >>z >>e;
  cout<<"Vavedete stoinosti za tn,tk,dt=";
  cin >>tn >>tk >>dt;
  t=tn;
  do
   {
      k=z*exp(-e/(R*t));
      cout<<"T="<<t<<“     K="<<k<<"\n";
      t=t+dt;
   }
  while (t<=tk);
  return 0;
}
```