8. НАРЕДБИ ВО C++

Наредби за редоследна контролна структура

Наредбите од оваа структура се запишуваат во големи загради:

контролна структура	*наредба*	*пример:*

почеток
чекор-1;
...
чекор-k;
крај
	{
	наредba-1;
	...
	наредба-k;
	}
		{
		a=5;
		b=7;
		c=a+b;
		}

Празна наредба (**null**-наредба)

```
;     // Ова е празна наредба
```

Пример:

```
while (a==b)
;
```

Наредби за контролните структури за избор

Наредба **if** (Избор на една можност **ако-тогаш**)

контролна структура	*наредба*

ако услов
тогаш
чекор-1;
...
чекор-k;
крај_ако {услов}
	if (услов)
	{
	наредba-1;
	...
	наредба-k;
	}

услов може да биде израз од било кој податочен тип. Ако **услов** има вредност различна од 0, се третира како **true**, а инаку како **false**.

Пример:

```
xaps=x:
if (x<0)
    xaps=-x;
```

Пример:

```
int x;
if (x)
    cout<<x;
```

Наредба if-else (избор од две можности **ако-тогаш-инаку**)

контролна структура	*наредба*

ако услов
тогаш
{
чекори-A;
}
инаку
{
чекори-B;
}
крај_ако {услов}
	if (услов)
	{
	наредби-A;
	}
	else
	{
	наредби-B;
	}

1. Заградите { и } се користат кога блоковите наредби-A; и наредби-B; содржат повеќе од една наредба.

2. Наредбите **if-else** можат да се вгнездуваат една во друга. Ако вгнездувањето се врши во else, се добива контролната структура **ако-илиако-инаку**.

Пример А:

```
if (услов1)
        наредба-1;
else if (услов2)
            наредба-2;
    else if (услов3)
            наредба-3;
        else
            наредба-4;
```

Пример Б:

```
if (услов1)
            наредба-1;
else if (услов2)
    {       наредба-2;
            if (услов3)
                наредба-3;
    }
    else
                наредба-4;
```

1. else оди со оној претходен if кој се наоѓа во истиот блок.
2. Во Пример-А, наредба-4 ќе се изврши само ако не се исполнети сите 3 услови.
3. Во Пример-В, наредба-4 ќе се изврши само ако не се исполнети првиот и вториот услов, независно дали е исполнет или не третиот услов.

Пример: Да се состави програма во C++ со која ќе се утврди дали зададен агол е остар или не.

```
#include <iostream>
using name space std;

void main ()
{
    int alfa;
    cout<<”Vnesi agol vo stepeni”<<endl;
    cin>>alfa;
    if ((alfa>0) && (alfa<90))
        cout<<”Agolot e ostar”<<endl;
    else
        cout<<”Agolot ne e ostar”<<endl;
}
```

Пример: Да се состави програма во C++ со која ќе се утврди дали зададен број е поголем, еднаков или помал од 5.

```
#include <iostream.h>

main ()
{
    int broj;
    cout<<”Vnesi broj i pritisni Enter”<<endl;
    cin>>i;
    cout<<”i=”<<i<<endl;

    if (i>5)
        cout<<”Brojot e pogolem od 5”<<endl;
    else
        if (i=5)
            cout<<”Brojot e ednakov na 5”<<endl;
        else
            cout<<”Brojot e pomal od 5”<<endl;
}
```

Пример: Да се состави програма во C++ со која ќе се погоди број помеѓу 1 и 10, што го “замислил” компјутерот.

```
#include <iostream>
using namespace std;

void main ()
{
    int broj=7, i;
    cout<<”Vnesi broj pomegu 1 i 10”<<endl;
    cin>>i;

    if (i==broj)
        cout<<”Bravo, pogodi! Zamisleniot broj e: ”<<i<<endl;
    else
    {
        cout<<”Zalam, ne pogodi! ”<<endl;
        cout<<”Zamisleniot broj e: ”<<broj<<endl;
    }
```

Пример: Да се состави програма во C++ со која ќе се одреди најголемиот од три дадени броја.

```
#include <iostream>
using namespace std;

void main ()
{
    int a,b,c;
    cout<<”Vnesi tri broja”<<endl;
    cin>>a>>b>>c;

    if (a>=b)
        if (a>=c)
            cout<<”Najgolem broj e a= ”<<a<<endl;
        else
            cout<<”Najgolem broj e c= ”<<c<<endl;
    else
        if (b>=c)
            cout<<”Najgolem broj e b= ”<<b<<endl;
        else
            cout<<”Najgolem broj e c= ”<<c<<endl;
    }
```

Условен оператор (услов**?** израз-Т **:** израз=Н;)

Условниот оператор се користи за претставување на едноставни **if-else** наредби.

Пример:

```
if (m>n)
    s=m-n;          ⇔          s=(m>n) ?  (m-n) : (m+n);
else
    s=m+n;
```

Пример: Што ќе се отпечати со програмскиот сегмент:

Cout<<”Brojot ” <<broj<<” e ”<<(broj%2) ? “paren.” : “neparen.”;

за broj=5?

Решение:

Бројот 5 е непарен.

Наредба Switch

Со оваа наредба се реализира контролната структура *избор од повеќе можности* (**case**).

контролна структура	*наредба*

случај израз
а: чекори-А;
излез;
b: чекори-В;
излез;
...
k: чекори-К;
излез;
инаку чекори-Х;
крај_случај {израз}
	switch (израз)
	{
	case a: наредби-А;
	break;
	case b: наредби-В;
	break;
	...
	case k: наредби-К;
	break;
	default: наредби-Х;
	}

1. Вредностите на *израз* се целобројни.
2. Лабелите a,b, ... , k се константи или целобројни изрази.
3. Наредбата **break** извршува скок на крајот на наредбата **switch**.
4. Ако не се наведе **break** извршувањето продолжува со следниот **case**.
5. Ако *израз* не добие вредност на било која лабела, се извршуваат наредбите по **default**.

Пример: Во зависност од знакот да се определи за која аритметичка операција се работи:

```
switch (znak)
{
    case  '+':    cout<<"znakot "<<"+:"<<" sobiranje";
                  break;
    case  '–':    cout<<"znakot "<<"–:"<<" odzemanje";
                  break;
    case  '*':    cout<<"znakot "<<"*:"<<" mnozenje";
                  break;
    case  '+':    cout<<"znakot "<<"/:"<<" delenje";
                  break;
    default:      cout<<znak<<endl;
                  break;
}
```

Пример: Да се внесе едноцифрен број помеѓу 1 и 9 и да се отпечати дали бројот е парен или непарен. Доколку сме внеле број што не помеѓу 1 и 9 да се отпечати: "Погрешен податок".

```
#include <iostream>
using namespace std;
void main ()
{
	int broj;
	cout<<”Vnesi broj”<<endl;
	cin>>broj;

	switch (broj)
	{
		case 1:
		case 3:
		case 5:
		case 7:
		case 9:
			cout<<”Brojot e neparen”<<endl; break;
		case 2:
		case 4:
		case 6:
		case 8:
			cout<<”Brojot e paren”<<endl; break;
		default:
			cout<<”Pogresen podatok<<endl; break;
	}
}
```

Оператори за инкрементирање и декрементирање ++, --

Тоа се оператори за зголемување/намалување на вредноста на податокот за 1.

оператор	форма	инкремент	декремент
++	++x	префиксен	
	x++	постфиксен	
––	––x		префиксен
	x––		постфиксен

Пример:

Вредност на x пред пресметувањето	израз	Вредност на изразот	Вредност на x по пресметувањето
5	++x	6	6
5	x++	5	6
5	– –x	4	4
5	x– –	5	4

Инкрементирањето и декрементирањето може да се користи како наредба.

Пример: Програмски сегмент за збир на првите n природни броеви.

```
int broj=1;
int zbir=0;
while (broj<=n)
{
    zbir+=broj;
    ++broj;                    // isto so broj=broj+1; ili broj+=1;
}
```

Пример:

```
#include <iostream>
using std::cout;
using std::endl;
{
    int a,b,c;
    a=b=c=1;
    cout<<a<<"  "<<b<<"  "<<c<<endl;
    a=b+c++;
    cout<<a<<"  "<<b<<"  "<<c<<endl;
    a=b=c=1;
    a=b+ ++c;
    cout<<a<<"  "<<b<<"  "<<c<<endl;
}
```

Излез:

```
1   1   1
2   1   2
3   1   2
```

Наредби за контролните структури за повторување

Наредба while

Со оваа наредба се реализира контролната структура **додека–извршувај**.

контролна структура	*наредба*

додека услов **извршувај**
чекор-A;
чекор-B;
...
чекор-K;
крај_додека {услов}
	white (услов)
	{
	наредба-A;
	наредба-B;
	...
	наредба-K;
	}

1. *услов* е **логички израз** кој може да има вредност **true** или **false** или **аритмеички израз** со вредност **различна од нула** или **нула**.
2. Ако *услов* не е исполнет уште при првиот циклус, тогаш наредбата while не се извршува ниту еднаш.

Пример: Да се пресмета бројот на знаци во една линија на внесен текст.

```
#include <iostream>
using namespace std;

main ()
{
	char znak;
	cout<<”Vnesi tekst”;
	int broj=0;
	cin.get (znak);
	while (znak != ′\n′)
	{
		++broj;
		cin.get (znak);
	}
	cout<<”Broj na vneseni znaci: ”<<broj<<endl;
}
```

Пример: Да се пресмета бројот на знаци во една линија на внесен текст.

```
#include <iostream>
using namespace std;

main ()
{
      char znak;
      cout<<”Vnesi tekst”;
      int broj=0;
      cin.get (znak);
      while (znak != ′\n′)
      {
            ++broj;
            cin.get (znak);
      }
      cout<<”Broj na vneseni znaci: ”<<broj<<endl;
}
```

Пример: Да се најде НЗД на два броја.

```
#include <iostream>
using namespace std;

void main ()
{
      int a,b,nzd;
      cout<<”Vnesi gi broevite a i b:”<<endl;
      cin>>a;
      cin>>b;
      if (a<b)
            nzd=a;
      else
            nzd=b;
      while ((a % nzd != 0) || (b % nzd != 0))
            nzd=nzd-1;

      cout<<”NZD na ”<<a<<” i ”<<b<<” e ”<<nzd<<endl;
}
```

Излез:

```
Vnesi gi broevite a i b:
80
120
NZD na 80 i 120 e 40.
```

Наредба do–while

Со оваа наредба се реализира контролната структура **извршувај–додека**.

контролна структура	*наредба*

извршувај
чекор-A;
чекор-B;
...
чекор-K;
додека {услов};
	do
	{
	наредба-A;
	наредба-B;
	...
	наредба-K;
	}
	while (услов);

Наредбата се користи кога некоја секванца (низа) наредби треба да се изврши барем еднаш.

Повторувањето на низата наредби се врши сè додека е исполнет *услов*. Кога *услов* нема да биде исполнет, повторувањето прекинува.

Пример: Да се пресмета вредноста на бројот e=2.718281 ... со дадена точност eps, користејќи ја формулата:

```
#include <iostream>
using namespace std;

void main ()
{
	int i;
	double e,eps,clen;
	cout<<”Vnesi ja tocnosta, eps: ”;
	cin>>eps;
	e=1;
	clen=1;
	do
	{
		clen=clen/i;
		e=e+clen;
		i++;
	}
	while (clen>=eps);
	cout<<”Brojot e so tocnost ”<<eps<<” iznesuva: ”<<e<<endl;
}
```

Наредба for

Со оваа наредба се реализира контролната структура за повторување со броење на циклусите **за–до–чекор**.

контролна структура	*наредба*

за бројач←почеток **до** крај **чекор** износ
чекор-А;
чекор-В;
...
чекор-К;
крај_за {бројач};
	for (*иницијализација*; *услов*; *ажурирање*)
	{
	наредба-А;
	наредба-В;
	...
	наредба-К;
	}

Најнапред, вредноста на контролната променлива *бројач* се поставува на вредност *почеток*. Потоа, пред започнување на циклусот се испитува дали вредноста на *бројач* е помала или еднаква на вредноста *крај*. Ако овој услов е исполнет циклусот се изведува, а вредноста на *бројач* се зголемува за *износ*. Ако условот не е исполнет, тогаш циклусот не се изведува

Една можна форма на наредбата for, т.е. на структурата **за–до–чекор**.

for (***бројач = почеток*** **;** ***бројач<=kraj*** **;** ***бројач+=износ***)
{
наредба-A;
наредба-B;
...
наредба-K;
}

Пример: Збир на непарни броеви помали од n.

```
#include <iostream>
using namespace std;

main ()
{
    int n,broj,zbir=0;
    cout<<”Vnesi n: ”;
    cin>>n;
    for (broj=1; broj<=n; broj+=2)
        zbir+=broj;
    cout<<”Zbirot na neparnite broevi do ”<<n<<” e ”<<zbir<<endl;
 }
```

Наредбата **for** во C++ дозволува иницијализација и ажурирање на повеќе променливи.

Пример: Збир на непарни броеви помали од n.

```
#include <iostream>
using namespace std;

main ()
{
      int n;
      cout<<”Vnesi n: ”;
      cin>>n;
      for (int broj=1, zbir=0; broj<=n; broj+=2)
            zbir+=broj;
      cout<<”Zbirot na neparnite broevi do ”<<n<<” e ”<<zbir<<endl;
 }
```

услов може да биде кој било логички израз во кој можат да се користат релациони и логички оператори.

Пример: Збир на непарни броеви помали од n.

```
#include <iostream>
using namespace std;

main ()
{
      int n;
      cout<<”Vnesi n: ”;
      cin>>n;
      for (int broj=1, zbir=0; broj<=n && (број % 2 ==1); broj++)
            zbir+=broj;
      cout<<”Zbirot na neparnite broevi do ”<<n<<” e ”<<zbir<<endl;
 }
```

Во делот за ажурирање во наредбата for може да се ажурираат повеќе променливи:

```
{
      int n;
      cin>>n;
      for (int i=1, j=n ; i<=n && j>=1 ; ++i, –-j)
            cout<<i+j<<endl;
}
```

Делот за иницијализација и ажурирање во наредбата for може да биде празен:

```
{
	int n;
	cin>>n;
	int i=1, int j=n;
	for ( ; i<=n && j>=1 ; )
	{
		++i; --j;
		cout<<i+j<<endl;
	}
}
```

Наредбите for можат да се вгнездуваат една во друга:

```
{
	int n;
	cin>>n;
	for (int i=1 ; i<=n ; ++i)
	{
		for (int j=n ; j>=1 ; --j)
			cout<<i+j<<endl;
	}
}
```

Наредби за контролните структури за излез и скок

Наредби за контролната структура излез **break, continue**

Контролните наредби за **излез** се користат за скок од местото каде што се наоѓаат во некоја контролна структура за повторување, на крајот од циклусот. Со нив настанува прекин на повторување на циклусот.

Наредбата **break** се користи за излез од контролните структури за повторување: **додека–извршувај** (while), **извршувај–додека** (do–while) и **за–до–чекор** (for) и структурата **случај** (case).

Наредбата **break** најчесто се користи за излез од структури со бесконечен број циклуси.

Пример: Погодување на замислен број од 1 до 10. Постапката прекинува откако ќе се погоди замисленият број.

```
#include <iostream>
using namespace std;

main ()
{
    const int zamislen_broj=7;
    for (int i=1 ; ; i++)
    {
        cout<<”Vnesete broj pomegu 1 i 10: ”;
        cin>>broj;
        if (broj==zamislen_broj)
        {
            cout<<”Pogodivte vo ”<<i<<”-tiot obid”;
            break;
        }
    }
}
```

Пример: Програма за пресметување квадратен корен од внесени броеви. Процесот да се прекине кога ќе се внесе негативен број.

```
#include <iostream>
#include <cmath>
using namespace std;

main ()
{
    double x;
    while (true)
    {
        cout<<”Vnesete realen broj x: ”;
        cin>>x;
        if (x<0)
        {
            cout<<”Brojot e negativen”; break;
        }
        cout<<”Kvadraten koren od ”<<x<<” e ”<<sqrt(x)<<endl;
 }
```

Со наредбата **continue** не се излегува од контролните структури за повторување како со наредбата **break**, туку само се прекинува тековниот циклус со скок на крајот на циклуот, а потоа се продолжува со наредниот циклус.

Пример: Програма за пресметување квадратен корен од внесени броеви. Доколку се внесуваат негативни броеви процесот да не се прекинува, туку за нив да не се пресметува квадратен корен.

```
#include <iostream>
#include <cmath>
using namespace std;

main ()
{
    double x;
    while (true)
    {
        cout<<”Vnesete realen broj x: ”;
        cin>>x;
        if (x<0)
        {
            cout<<”Brojot e negativen”;
            continue;
        }
        cout<<”Kvadraten koren od ”<<x<<” e ”<<sqrt(x)<<endl;
 }
```

Програмата не обезбедува регуларно завршување на повторувањето.