# Лабораторно упражнение № 6

## Тема: Оператори за цикли в езика C++. Реализиране на циклични алгоритми

### I. Цел на лабораторното упражнение

Да се затвърдят знанията на студентите за организацията и синтаксиса на програми написани на програмният език С. Да се усвоят нови знания и умения при създаването на циклични програми.

### II. Постановка на задачата

Многократното изпълнение на последователност от инструкции представлява важна алгоритмическа концепция. Един от методите за организация на повторение на изпълнението на поредица от команди (който се явява итерационна структура) се нарича `цикъл`. Характерно за циклите е, че в конструкцията им се предвиждат специални инструкции, осигуряващи управлението на цикъла набор от операции, които се изпълняват многократно. Управляващите инструкциите задават началото и края на цикъла и задават структурата и типа на цикъла, а многократно изпълняваните инструкции образуват тялото на цикъла. Инструкцията за начало на цикъла обикновено се нарича заглавен оператор.

### III. Теоретични сведения

Операторите за цикъл имат три основни елемента в организацията на програмния процес:

- Задаване на начални стойности на величини, участващи в управлението на цикъла. Те се наричат инициализиращи елементи и в някои случаи могат да се разполагат преди оператора за цикъл;
- Обновяване на стойностите на управляващите променливи. При всяко изпълнение на операторите от тялото на цикъла трябва да се обновява стойността на една или няколко променливи, от които зависи докога се изпълнява цикълът.
- Проверка на условие за край на цикъла. Повторението на изпълнение на операциите от тялото на цикъла се контролира от логическо условие. При всяко изпълнение на операторите от тялото трябва да се проверява това условие.
- Има три основни конструкции на цикли
- Цикъл тип `while` (докато е изпълнено условието - повтаряй тялото на цикъла);
- Цикъл тип `do...while` (повтаряй тялото на цикъла докато е изпълнено условието);

- Цикъл тип **for** (вариант на цикъла **while**).

### Цикъл с предусловие (while).

Този цикъл може да се представи логически с помощта на изречението:

**<Докато> <Израз> <повтаряй> <Оператор>.**

Тук **Израз** е логически израз, а **Оператор** е изпълним оператор (група оператори) представляващ тялото на оператора. Схемата на оператора с предусловие има вида (фиг. 1), където с пунктирана линия е показано представянето на цикъл с предусловие.

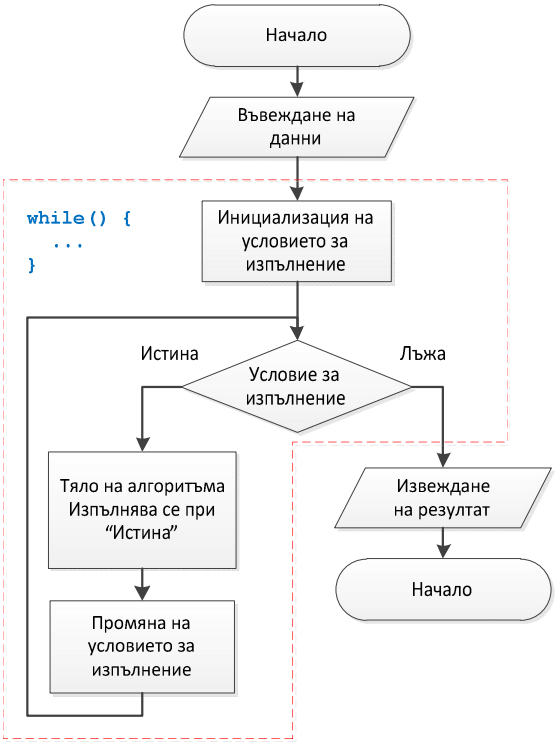
Синтактическата форма на оператора в C/C++ има вида:

```
while(<Израз>)
{
   <Оператор>
}
```

където:

Израз - е логически израз, който дава като резултат стойност истина или лъжа ( при математически изрази резултат 0 е лъжа, а за всички резултати различна от 0 е истина);

Оператор - един оператор или съставен оператор. Това е тялото на цикъла.



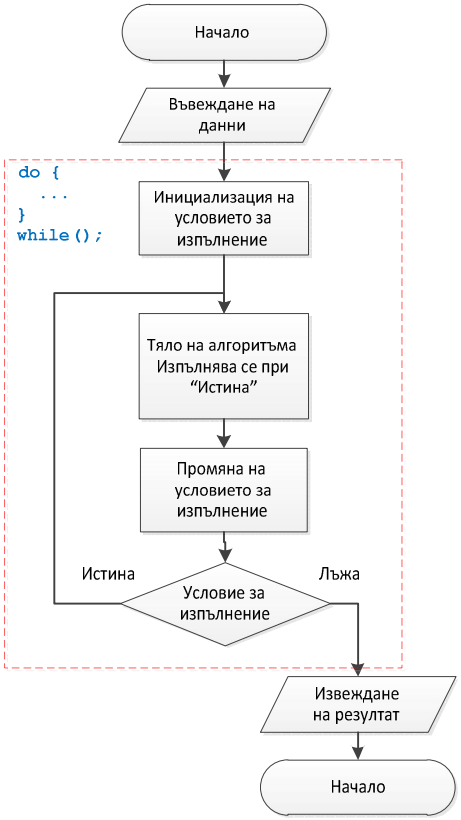

**Фиг.1.** Цикличен алгоритъм с предусловие

В тази конструкция на оператор за цикъл инициализиращата част се намира извън оператора за цикъл (преди оператора). Условието за проверка за продължаване на цикъла се задава с логическия израз Израз. Ако стойността на Израз е истина, се изпълнява тялото на цикъла Оператор, а ако е лъжа цикълът се напуска и управлението се предава на първия оператор, непосредствено следващ оператора за

цикъл. След изпълнение на операторите от тялото, управлението се предава в началото на цикъла за нова проверка на условието за продължаване. Цикълът се нарича с предусловие, защото най-напред се проверява условието и ако то е истина, се изпълняват операторите от тялото на оператора. **Това означава, че е възможно тялото на цикъла да не се изпълни нито веднъж, ако още при влизането в цикъла условието е неистина.** Обновяване на стойността на управляващите променливи трябва да се предвиди в тялото на цикъла.

### Цикъл с постусловие ( do...while)

Може да се представи логически с помощта на изречението:
**<Повтаряй> <Оператор> <докато> <Израз>.**
**<Израз>** е логически израз, а **<Оператор>** е изпълним оператор представляващ тялото на оператора. Алгоритмичната схема на оператора е показана на фиг. 2.




**Фиг. 2.** Цикличен алгоритъм със следусловие

Синтактическата форма на оператора в C/C++ има вида:

```
do {
    <Оператор>
} while (<Израз>);
```

където:
**Израз** - е израз, който дава като резултат стойност 0 или различна от 0;
**Оператор** - един оператор или съставен оператор.Това е тялото на цикъла.
Действието на оператора се изпълнява в следната последователност:

- Тялото на цикъла се изпълнява.
- Изчислява се **Израз**.
- Ако резултатът е стойност, различна от нула ("**истина**"), цикълът се изпълнява отново.
- Изпълнението на цикъла се прекратява, когато **Израз** получи стойност "**неистина**".

Както се вижда от схемата, проверката на условието за продължаване на цикъла се прави след изпълнение на операторите от тялото на цикъла.

Разликата между цикъл тип **while** и цикъл тип **do...while** е, че при **do...while** тялото на цикъла се изпълнява винаги поне един път, независимо от стойността на **Израз**.

## Цикъл с управляваща променлива ( цикъл for)

В много случаи броят на изпълненията на операторите от тялото на цикъл е известен предварително и тогава може да се използва цикъл, управляван с променлива. Той може да се представи с логическата фраза:

**за Id = St1 докато Условие2 прави Действие3**

където **Id** е идентификатор управляващ цикъла (индекс на цикъла), който представлява проста променлива; **St1** е израз чиято стойност е начално значение на **Id**; **Условие2** - израз указващ условието за край и **Действие3** - оператор (оператори), който представлява тялото на цикъла.

Синтактическата форма на оператора в език C/C++ има вида:

```
for([<инициализация>];[<условие за край>];[<инкремент>]) {
      <Оператор>
}
```

където:

**<оператор>** е един оператор или съставен оператор;

**<инициализация>** обикновено е израз, който присвоява начална стойност на променливата, която играе ролята на брояч на цикъла;

**<условие за край>** - е проверка на условие за продължаване на цикъла;

**<инкремент>** е израз, който реализира промяна на стойността на брояча;

Този формат е просто частен случай на цикъл **while**. Той е еквивалентен на следния запис:

```
<инициализация>; // инициализира брояча
while (<условие за край>)// проверка за  достигане краят на цикъла
 {
      <оператор>; // изпълними оператори
      <инкремент>; // модифицира брояча на цикъла
}
```

Квадратните скоби в синтаксиса показват, че всеки от изразите след **for** може да се пропусне, но точката и запетаята трябва да се запазят. Ако се пропусне **[<условие за край>]**, стойността му се приема за 1 ("истина") и цикълът става безкраен

Допустими типове за управляваща променлива на цикъла са всички числени типове, булевите променливи (стойности 0 и 1) и символните типове (като номера от кодовата таблица), както и някои други типове, които ще бъдат въведени по-късно.

## Вложени цикли

Вложените цикли представляват конструкция от няколко цикъла един в друг. Цикълът, в който се влага друг цикъл, се нарича външен цикъл, а този, който се влага, се нарича вътрешен. Най-вътрешния цикъл се изпълнява най-много пъти. В примерната конструкция по долу представлява вложен цикъл.

**Пример:**

```
for( int i=0;i<10;i++)
  for(int j= 10; j>0; j--)
      cout << " i = " << i << "   j = " << j << endl;
```

Вложените цикли, използвани необмислено, могат да влошат производителността на програмата.

## Преждевременно излизане от цикъл - break

Доста често се налага да се прекъсне изпълнението на даден цикъл, преди да са изпълнени зададения брой итерации. Това може да стане по два начина, задаване на стойност на индексната променлива извън горната граница за цикъла, или използването на оператора **break**.

Използването на първия случай не е удачно поради необходимостта да се знаят винаги граничните условия на цикъла. Това не винаги е възможно. **Например:**

```
void InputData()
{
 int i;
 for(;;)
  {
    . . . .
     if( i != 0 )
        break; // Излизане от цикъла
  }
}
```

В примера изразът **for(;;)** дефинира безкраен цикъл. Излизането от цикъла е възможно единствено с оператора **break**

## Прекъсване на текуща итерация continue

Има случаи, в които не желаем да напуснем изобщо даден цикъл, а искаме само да пропуснем останалата част от операторите в него и да продължим пак от началото му. Такъв начин на действие се осъществява от оператора **continue**.

При изпълнението на оператор **continue**, програмата пропуска останалата част от операторите в цикъла и започва от началото му. Поради това резултатите са различни от тези, получени без **continue**.

## Оператор за безуслоевен преход goto

Езикът C/C++ предлага и оператор **goto**, въпреки че повечето от случаите могат да се решат с останалите три оператора за предаване на управлението. Използвайте оператора внимателно и само ако се налага. Форматът му е:

```
goto етикет;
      . . .
етикет:  оператор
```

където "етикет" е идентификатор, свързан с даден оператор.

### IV. Задание за работа

**Зад 1.** Да се преобразуват в двоична, осмична и шестнадесетична бройна система десетичните числа 2567,367 и - 756,903.

запетая?

**Зад. 1..** Да се напише програма на C, която да изчислява минималната стойност на функцията:

F(x) = arctg( sin(n!) / n! ) + 0.22cos(x)

в интервала -y до y при произволна стойност на n. Факториела да се изчислява чрез итерация и чрез рекурсивна функция.

**Зад. 2.** Да се напише програма изчисляваща числото на Непер (e=2.7182818284...). За да се реши задачата се използва следното представяне:

e= 1/0!+1/1! +1/2! + .....

**Зад. 3** Да се напише програма пресмятаща интеграла $\int_a^b F(a)dx$ при дадени a,b и функция F.

Използвайте метода на Симсън. Съгласно този метод интервала [a,b] се разделя на 2.N+1 части. Всяка част има дължина H=(b-a)/2.N. Точките които разделят интервала на части, означаваме с X0, X1, X2 ... X2N. При това X0=a, X2n=b, X1=X0+H1. Нека означим с F1 стойността на функцията F(x) в точка X1: F1=F(x1). Интеграла се изчислява приближено съгласно следната формула.

$$\int_a^b F(x)dx = \frac{H(F_0 + 4.F_1 + 2.F_2 + 4.F_3 + \ldots + 4.F_{2n-1} + 2.F_{2n})}{2.n}$$

**Зад. 4** Да се напише програма, която намира сумата на целите числа от **1** до **n**, като **n** се въвежда от клавиатурата.

**Зад. 5** Да се напише програма, която намира произведението на целите числа от **1** до **n** (**n!**), като **n** се въвежда от клавиатурата (**n<34** ).

**Зад. 6** Да се напише програма, която въвежда **n** произволни числа от клавиатурата и намира тяхната сума ($1 \le v \le 1000$).

**Зад. 7** Да се напише програма, която въвежда **n** числа от

клавиатурата и намира средноаритметичната им стойност ($1 \le v \le 1000$).

**Зад. 8** Да се напише програма, която въвежда **n** числа от клавиатурата и намира сумата само на онези от тях, които са положителни ($1 \le v \le 1000$).

**Зад. 9** Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира средноаритметичната стойност само на онези от тях, които са положителни ($1 \le v \le 1000$).

**Зад. 10.** Да се напише програма, която въвежда **n** ($v \le 10$) на брой реални числа по-малки от 100 и намира тяхното произведение.

**Зад. 11.** Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира сумата само на онези от тях, които са отрицателни.

**Зад. 12.** Да се напише програма, която въвежда **n** ($v \le 10$) на брой реални числа по-малки от 100 и намира произведението на онези от тях, които са различни от нула.

**Зад. 13.**Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира сумата само на онези от тях, които са по-големи от предварително зададено число **к.**

**Зад. 14.** Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира сумата само на онези от тях, които са по-малки от предварително зададено число **к.**

**Зад. 15.**Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира сумата само на онези от тях, които са по-малки от предварително зададено число **u** и по-големи от предварително зададено число **v (u>v).**

**Зад. 16.**Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира средноаритметичната стойност само на онези от тях, които са по-големи от предварително зададено число **к.**

**Зад. 17.**Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира средноаритметичната стойност само на онези от тях, които са по-малки от предварително зададено число **к.**

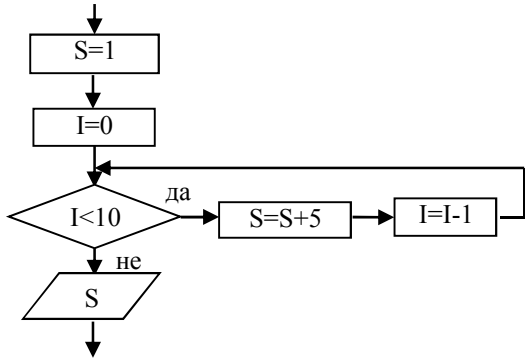
**Зад. 18.**Да се напише програма, която въвежда **n** на брой числа от клавиатурата и намира средноаритметичната стойност само на онези от тях, които са по-малки от предварително зададено число **u** или по-големи от предварително зададено число **v(u<v).**

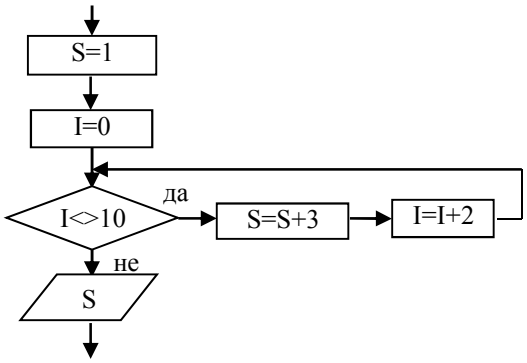
## V. Указания за работа

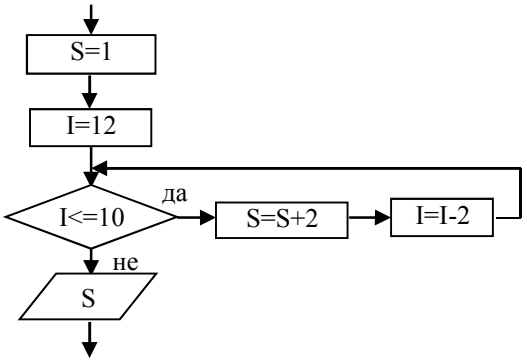
При изпълнението на задачите използвайте примерите приложени в теоретичната част?

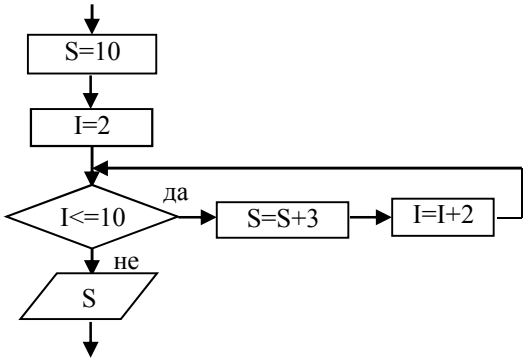
## VI. Контролни въпроси

1. Какви оператори за цикли познавате?
2. Направете сравнение между оператора **for** и **while**
3. Направете сравнение между оператора **for** и **do while**
4. Направете сравнение между оператора **while** и **do while**
5. Кои програми се наричат циклични?
6. Как се извършва обхождането на елементите от N мерен масив?
7. Колко цикъла са необходими за да се нулират елементите на 5-мерен масив?
8. Колко вложени цикъла са необходими за да се умножат две матрици?
9. Колко вложени цикъла са необходими за да се сумират две матрици?
10. Разгледайте блоковите схеми и отговорете за всяка една:
    ***a)*** колко пъти ще се изпълни тялото на цикъла;
    ***b)*** каква ще бъде стойността на променливата **I** след излизане от цикъла;
    ***c)*** каква стойност ще се изведе за **S**?




11. По-долу са написани фрагменти от програми на C++. За всеки един фрагмент отговорете:
    ***a)*** колко пъти ще се изпълни тялото на цикъла;
    ***b)*** каква ще бъде стойността на променливата **I** след излизане от цикъла;
    ***c)*** каква стойност ще се изведе за **S**?

```
s=10;
I=1;
while (I>6)
```

```
s=10;
I=5;
while (I>=1)
```

```
s=10;
I=5;
while (I>=1)
```

```
  {
    s=s + I;
    I=I - 1;
  }
   cout<< "s="<<s;
```

```
  {
    s=s + 10;
    I=I + 3;
  }
cout<< "s="<<s;
```

```
  {
    s=s + 10;
    I=I - 1;
  }
cout<< "s="<<s;
```

12. Поправете грешките

```
s=10;
j=5;
while j>=1
  {
    s=s + 10;
    j=j - 1;
  }
cout<< "s="<<s;
```

```
I=5;j=0;
while (j>=1)
  }
    s=s + 10;
    j=j - 1;
  {
cout<< "s="<<s;
```

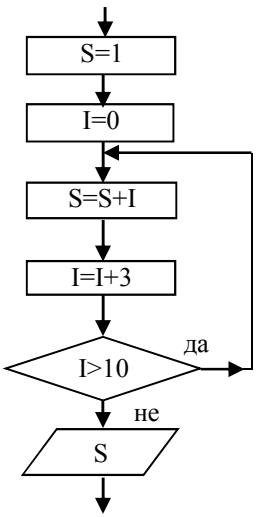
```
s=10;
I=5;
while (I>=1);
    s=s + 10;
    I=I - 1;
  }
cout<< "s="<<s;
```

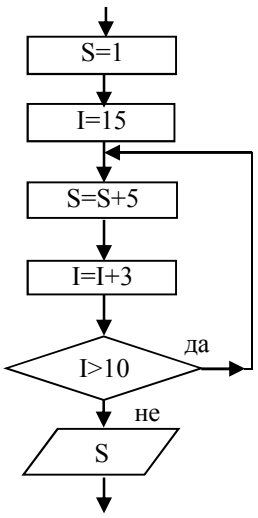
13. Разгледайте блоковите схеми и отговорете за всяка една:

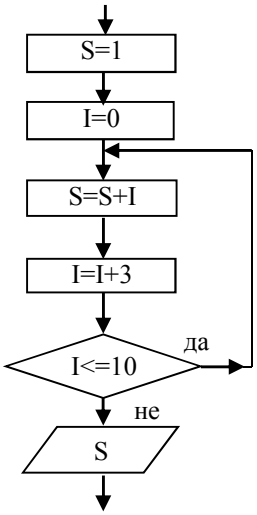
***a)*** колко пъти ще се изпълни тялото на цикъла;

***b)*** каква ще бъде стойността на променливата **I** след излизане от цикъла;

***c)*** каква стойност ще се изведе за **S**?




14. По-долу са написани фрагменти от програми на C++. За всеки един фрагмент отговорете:

***a)*** колко пъти ще се изпълни тялото на цикъла;

***b)*** каква ще бъде стойността на променливата **I** след излизане от цикъла;

***c)*** каква стойност ще се изведе за **S**?

```
    s=10;
    i=0;
    do
     { s=s + i;
      i=i + 1;
     }while (i>10);
cout<< "s="<<s;
```

```
    s=10;
    i=16;
    do
    {s=s + 2*i;
      i=i + 1;
    }while (i>10);
    cout<< "s="<<s;
```

```
    s=10;
       i=0;
       do
```

```
    s=10;
    i=0;
    do
```

```
        { s=s + i;
          i=i + 2;
        }while (i<=10);
      cout<< "s="<<s;
```

```
      {  s=s + 3*i;
        i=i + 1;
      }while (i<10);
    cout<< "s="<<s;
```

15.Поправете грешките:

```
   do
  {
  cout<<"Vuvedete broq na u4enicite\n";
  cout<<"(5<n<10) n = ";
  cin>>n
  }while (n<5)||(n>10);
```

```
   do
  {
  cout<<"Vuvedete broq na u4enicite\n";
  cout<<"(5<n<10) n = ";
  cin>>n
  }while (n<5)||(n>10);
```

```
   do
  {
  cout<<"Vuvedete broq na u4enicite\n";
  cout<<"(5<n<10) n = ";
  cin>>n
  }while (n<5)||(n>10);
```

## VII. Литература


1 Кай Хорстман. Принципи на програмирането със C++. ИК Софтех София 2000

2. Симеонов Г. Програмиране на C++. С. Техника 1993г.

3. Тед Фейсон. Borland C++. Обектно-ориентирано програмиране. NISOFT София 1994

4. Богданов Д., И.Мустакеров. Език за програмиране С. С. Техника 1989г.

5. А.Касткин. Профессиональное программирование на языке С. Системное программирование. Минск Вышэйшая школа 1993.