ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Урок №3

Програмиране на езика C

Съдържание

1. Понятието цикъл 3
2. Цикъл while.....6
3. Конструкция do while 10
4. Примери към урока..... 15
5. Домашна работа 18

1. Понятието цикъл

Много често и в живота и при писането на програми съществува необходимост от повтаряне на някои действия, няколко пъти. Например, да си представим алгоритъма за миене на чинии.

```
1. Взимаме чинията от мивката.
2. Насапунисване на чинията с препарат за миене на чинии.
3. Търкане на чинията с гъба.
4. Отмиване на сапунената пяна от чинията.
5. Изтриване на чинията.
6. Поставяне на чинията в шкаф.
7. Край на програмата.
```

В дадения на пръв поглед смислен алгоритъм има едно малко неразбирателство – ако чиниите бъдат повече от една, то измити ще се окаже така или иначе само една. Това е свързано с това, че програмата изпълнява действието по линеен начин – отгоре на долу по ред на номерата. Следователно, на нас ни трябва да измислим по какъв начин да накараме програмата да повтори набор от конкретни действия, и същевременно да определим нужното количество от повторения. Правилния алгоритъм ще изглежда така.

```
1. Взимане на чинията от мивката.
2. Насапунисване на чинията с препарат за миене на чинии.
3. Изтъркване на чинията с гъба.
4. Отмиване на пяната от чинията.
5. Изтриване на чинията.
6. Поставяне на чинията в шкаф.
7. Ако има още мръсни чинии- връщане към пункт 0.
8. Край на програмата.
```

Ще обърнем внимание на това, че за да определим дали да се повтаря действието от начало се използва условие „Ако има още мръсни чинии“. Ако има, е условие за вярност- действието се повтаря, ако е грешно, се изпълнява следващия 8-ми пункт от алгоритъма.

И така ние стигнахме до извода, че на нас ни е необходима някаква конструкция която съдържа в себе си набор от действия за повторения. В същото време количеството от повторения трябва да зависи от някакво условие, съдържащо се в същата конструкция.

Неволно, ние току що дадохме определение на така наречения ЦИКЪЛ. Да повторим още един път!!!

Цикъл — специализиран оператор на езика за програмиране с чиято помощ едно или друго действие може да се изпълни колкото пъти е необходимо в зависимост от условието.

Забележка: Друго наименование на цикъла – конструкция за повторение. А всяко повторение на действието – стъпка на цикъла или итерация.

В езика C съществуват няколко реализации на такава форма като цикъл. В този урок ще говорим за две такива реализации — **while** и **do while**.

2. Цикъл while

Общ синтаксис на порядъка на изпълнение на цикъла while

```
while(утвърждение)
{
                        Действие за повторение;
}
```

1. Преди всичко се осъществява проверка за утвърждение.
2. Ако утвърждението в кръглите скоби е вярно, се изпълнява действието което се намира във фигурните скоби.
3. Ако утвърждението в кръглите скоби е грешно, програмата ще премине към следващия ред след закриващата фигурна скоба цикъла.
4. Ако утвърждението в кръглите скоби е било вярно и действието се е изпълнило, следва отново проверка на утвърждението.
   Както виждате проверката на утвърждението се повтаря при всяко изпълнение на цикъла. Когато то престане да бъде вярно цикъла приключва. Обърнете внимание, че ако утвърждението е грешно от самото начало, действието в цикъла няма да бъде изпълнено нито веднъж.

Да разгледаме пример.

Да предположим че на някой човек му е необходимо да напише описание за 7-те чудеса на света. Преди да го направи на него му е необходимо да отиде и да види всяко едно от чудесата. И чак след това да пише за тях. Името на проекта **Miracles**.

```
#include <iostream>
using namespace std;
void main()
{
//обявяване на управляваща
 променлива
int counter=0;
while(counter<7)// проверка на утвърждението
              {
                  counter++;// промяна на управляващата променлива

                  //действие за повтаряне

                  // вие видяхте ... чудо на света
                  cout<<"You seen»<<counter<<" miracle of world!!!\n";

              }
cout<<"Now, you can start your work.\n";
}
```

Сега да разберем подробно как работи нашия пример.

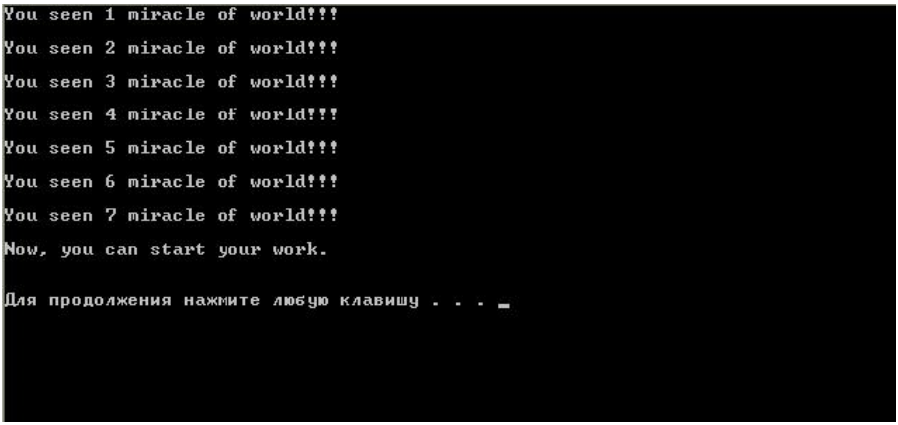
1. Обявяваме променлива изначално равна на 0
2. След това в условие на цикъла ние проверяваме значението на променливата. Защото именно от неговото значение зависи, ще се изпълни ли цикъла или не, такава променлива се нарича управляваща цикъла променлива.
3. Увеличаваме значението на променливата с единица.

Забележка: Даденото действие е задължително, тъй като ако не променим значението на променливата управляваща цикъла, резултата от проверката на утвърждението също никога няма да се промени. Това може да доведе до много разпространена грешка наречена- вечен цикъл. Ако утвърждението е вярно, а управляващата променлива винаги има едно и също значение, следователно - утвърждението е вярно винаги. Представете си, че мръсните чинии никога не свършват- тяхното число винаги е постоянно. За колко ще

стигне препарата за миене на чинии?! Няма да е за много, нали? Точно така и програмата няма да издържи на такъв натиск и след определено време след старта на вечния цикъл ще изкара грешка на етапа за изпълнение. За да се избегнат такива грешки, трябва внимателно да следи за това, че вътре в тялото на цикъла да се извършва промяна на управляващата променлива.

4. След това изкарваме на екрана текущото значение на нашата променлива под вида на съобщение за номера на прегледаното чудо на света.
5. След това отново се връщаме към условието и проверяваме значението на управляващата променлива.

Цикъла ще продължава своята работа докато значението на променливата не стане равно на 7. В този случай ще излезе на екрана надпис „You seen 7 miracle of world!!!", след това програмата ще се върне за проверка на условието. 7<7- е грешно. Програмата вече няма да влиза в цикъла и ще премине към съобщението „Now, you can start your work." В процеса на изпълнение на програмата ние ще видим следната картина:



Сега ние се запознахме с една от разновидностите за цикъл в езика C. Надяваме се, че не е сложно. В следващия раздел на урока, ще се запознаем с цикъла за алтернативната конструкция while..

3. Конструкцията do while

Общ синтаксис и принцип на работа с do while:

```
do
{
                    действие;
}
while(условие);
```

Цикъла do while е подобен на цикъла while. Разликата е в това, че в while проверката на условието протича веднага при начало на цикъла, и само след това, ако условието е вярно се изпълнява действие. В do while винаги първо се изпълнява действието и след това следва проверка на условието. Ако условието е вярно, то изпълнението се предава на оператора след while. С други думи, за разлика от while, в do while действието се изпълнява поне един път. Нека да разгледаме това на схемата:

Прилагане на do while на практика

Да предположим, че ни е необходимо да напишем програма в която потребителя да има право на избор да извърши някакво действие няколко пъти подред. Ще реализираме дадената задача първо с помощта на while, а след това с помощта на do while. Името на проекта е Calc.

```
#include <iostream>
using namespace std;
void main()
{
int answer,A,B,RES;

// молба за избор на операция
cout<<"\nSelect operation:\n";
cout<<"\n 1 - if you want to see SUM.\n";
cout<<"\n 2 - if you want to see DIFFERENCE.\n";
cout<<"\n 3 - if you want to exit.\n";
cin>>answer;

while(answer!=3){ // проверка на
        условието
 switch(answer){
                case  1:    // ако потребителя е избрал събиране
                cout<<"Enter first digit:\n";
                cin>>A;
                cout<<"Enter second digit:\n";
                cin>>B;
                RES=A+B;
                cout<<"\nAnswer: "<<RES<<"\n";
                break; // спиране switch
                case  2:     // ако потребителя е избрал изваждане
                cout<<"Enter first digit:\n";
                cin>>A;
                cout<<"Enter second digit:\n";
                cin>>B;
                RES=A-B;
                cout<<"\nAnswer: "<<RES<<"\n";
                break; // спиране switch
        case  3: // ако потребителя е избрал изход
                cout<<"\nEXIT!!!\n";
                break;
        default:    // ако избраното действие е некоректно
                cout<<"\nError!!! This operator isn't correct\n";
               }
```

```
// молба за избор на операция
        cout<<"\nSelect operation:\n";
        cout<<"\n 1 - if you want to see SUM.\n";
        cout<<"\n 2 - if you want to see DIFFERENCE.\n";
        cout<<"\n 3 - if you want to exit.\n";
                cin>>answer;
        }
       cout<<»\nBye....\n»;
}
```

В дадения пример на потребителя се предоставя избор за действие. След въвеждането, програмата проверява дали топва действие е изход от програмата (програмата приключва), а ако не, то протича начало на цикъл, анализ на действия и изпълнение на математическа операция. След това програмата отново ще попита потребителя какво той иска да направи.

Дадения код не представлява оптимално решение. Както виждате фрагмента

```
    // молба за избор на операция
       cout<<»\nSelect operation:\n»;
       cout<<»\n 1 - if you want to see SUM.\n»;
       cout<<»\n 2 - if you want to see DIFFERENCE.\n»;
       cout<<»\n 3 - if you want to exit.\n»;
       cin>>answer;
```

Се повтаря няколко пъти. В такъв случай трябва да се използва do while. Дадената конструкция ще приведе кода към належащия вид. Името на проекта е: CalcDoWhile.

```
#include <iostream>
using namespace std;
void main()
{
int answer,A,B,RES;

do{ // вход в цикъл

        // молба за избор на операция
        cout<<"\nSelect operation:\n";
        cout<<"\n 1 - if you want to see SUM.\n";
        cout<<"\n 2 - if you want to see DIFFERENCE.\n";
        cout<<"\n 3 - if you want to exit.\n";
                                cin>>answer;

        // анализ на
        действие
        switch(answer){
        case 1: // ако потребителя е избрал събиране
                  cout<<"Enter first digit:\n";
                  cin>>A;
                  cout<<"Enter second digit:\n";
                  cin>>B;
                  RES=A+B;
                  cout<<"\nAnswer: "<<RES<<"\n";
                  break; // спиране switch
        case 2: // ако потребителя е избрал изваждане
                  cout<<"Enter first digit:\n";
                  cin>>A;
                  cout<<"Enter second digit:\n";
                  cin>>B;
                  RES=A-B;
                  cout<<"\nAnswer: "<<RES<<"\n";
                  break; // спиране switch
        case 3: // ако потребителя е избрал изход
                  cout<<"\nEXIT!!!\n";
                  break;
        default:    // ако избраното действие е некоректно
                  cout<<"\nError!!! This operator isn't correct\n";
   }

        } while(answer!=3);
cout<<"\nBye....\n";

}
```

Изхождайки от гореспоменатото, разбирате, че и двете описания в днешния урок на конструкцията са полезни. Необходимо ви е само да се научите да избирате едната или другата, в зависимост от задачата.

Сега, когато сме горе долу запознати с цикъла, можем да преминем към следващия раздел на дадения урок. Приготвили сме ви няколко примера по днешната тема.

4. Примери към урока

Пример 1.

Условия на задачата.

Да се напише програма, която да намира сумата на всички цели числа от 1 до 5 включително. Името на проекта е Summ. **Код за реализация.**

```
#include <iostream>
using namespace std;
void main(){
    int BEGIN=1; // начало диапазона на събираните
    значения
     int END=5; // край на диапазона на събираните
    значения
    int SUMM=0; // променлива за натрупване на
    сума
     int i=BEGIN; // управляваща цикъла променлива

     // проверка на условието
     while(i<=END){ //(сравнение на урпавляващата променлива с края на диапазона)
     SUMM+=i;// натрупване на сума
     i++;// промяна на управляващата променлива
     }

     // извеждане на резултата
     cout<<"Result - "<<SUMM<<"\n\n";
}
```

Коментар към кода.

В качеството на коментар към кода, решихме да представим таблица която из основи описва всеки пункт от цикъла:

ВХОДНЫЕ ДАННЫЕ			
BEGIN=1		END=5	
РАБОТА ЦИКЛА			
номер итерации	i	условие	SUMM
1	1	1<=5 - true	0+1=1
2	2	2<=5 - true	1+2=3
3	3	3<=5 - true	3+3=6
4	4	4<=5 - true	6+4=10
5	5	5<=5 - true	10+5=15
6	6	6<=5 - false	X
SUMM=15			

При разучаване на таблицата не е трудно да се забележи, че управляващата променлива, също така изпълнява ролята на променлива, която последователно избира значения за сумиране.

Забележка: Разпространено заблуждение е, че управляващата променлива може да бъде променяна само с единица- това не е така. Главното е променливата да може да се променя по всякакъв логичен образ.

Пример 2.

Условие на задачата.

Да се напише програма, която да изкарва на екрана линия от 5 звезди. Името на проекта е Line.

Код за реализация.

```
#include <iostream>
using namespace std;
void main(){
       int COUNT=5; // количество звезди (дължина на
       линията)
        int i=0; // управляваща цикъла променлива

       while(i<=COUNT){ // проверка на

        условието

        cout<<"*";// въвеждане на звезди
        i++;// промяна на управляващата променлива
       }
       cout<<"\n\n";
}
```

Коментар към кода.

1. Управляващата променлива в момента на проверката на условието е равна на количеството вече нарисувани звезди. Това се случва, защото променливата i се увеличава с единица след всяко въвеждане на *.
2. Цикъла ще спре тогава, когато i=5, което ще съответства на количеството нарисувани *.

Сега следва да продължите с домашната работа!

5. Домашна работа

1. Направете програма която да изкарва на екрана хоризонтална линия от символи. Числото на символите, какъв символ да се използва и каква ще бъде линията – вертикална или хоризонтална – избира потребителя..
2. Напишете програма, която да намира сумата на всички цели нечетни числа в диапазон посочен от потребителя.
3. Дадено е натурално число n. Напишете програма, която да изчислява факториела на неотрицателните цели числа n (тоест цяло число по-голямо от 0) Формулата за изчисление на факториела е посочена долу.

```
n! = 1*2*3*....*n, (формула за изчисление на факториела n)
0! = 1 (факториел 0 е равен на 1 (според определението на
факториела))
```