Семинар по „Увод в програмирането“

Цикли

Нека имаме следната аритметична прогресния с начало a1 = 2 и разлика

d = 1.5. Нека имаме следната задача да изведем на екрана първите 10 числа от тази редица. С текущите ни знания това ще стане по следния начин.

```cpp
#include <iostream>

int main()
{
    double currentNum = 2;
    double d = 1.5;
    int i = 1;

    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;
    std::cout << "Element " << i << " is " << currentNum << std::endl;
    currentNum += d;
    i += 1;

    return 0;
}
```

И съответно резултата на конзолата е следния:

```
Element 1 is 2
Element 2 is 3.5
Element 3 is 5
Element 4 is 6.5
Element 5 is 8
Element 6 is 9.5
Element 7 is 11
Element 8 is 12.5
Element 9 is 14
Element 10 is 15.5

D:\C++\WorkSpace\Debug\WorkSpace.exe (process 16616) exited with code 0.
Press any key to close this window . . .
```

До тук добре. Желаният резултат е постигнат! Но в програмирането има концепция наречена **DRY principle**, която гласи „**Don't repeat yourself**“ или с други думи, повторението на един и същи код или логика трябва да бъде заместен чрез абстракция. Тогава този код може да бъде съкратен чрез **for цикъл** по следния наичн:

```
1   #include <iostream>
2
3   int main()
4   {
5       double currentNum = 2;
6       double d = 1.5;
7
8       for (int i = 1; i <= 10; i++)
9       {
10          std::cout << "Element " << i << " is " << currentNum << std::endl;
11          currentNum += d;
12      }
13
14      return 0;
15  }
```

И резултатът е абсолщтно същият в много по-съкратен вид.

```
Element 1 is 2
Element 2 is 3.5
Element 3 is 5
Element 4 is 6.5
Element 5 is 8
Element 6 is 9.5
Element 7 is 11
Element 8 is 12.5
Element 9 is 14
Element 10 is 15.5

D:\C++\WorkSpace\Debug\WorkSpace.exe (process 11592) exited with code 0.
Press any key to close this window . . .
```

В програмирането **циклите** ни помагат да повторим даден блок от код определен брой пъти. Днес ще разгледаме три типа цикли:

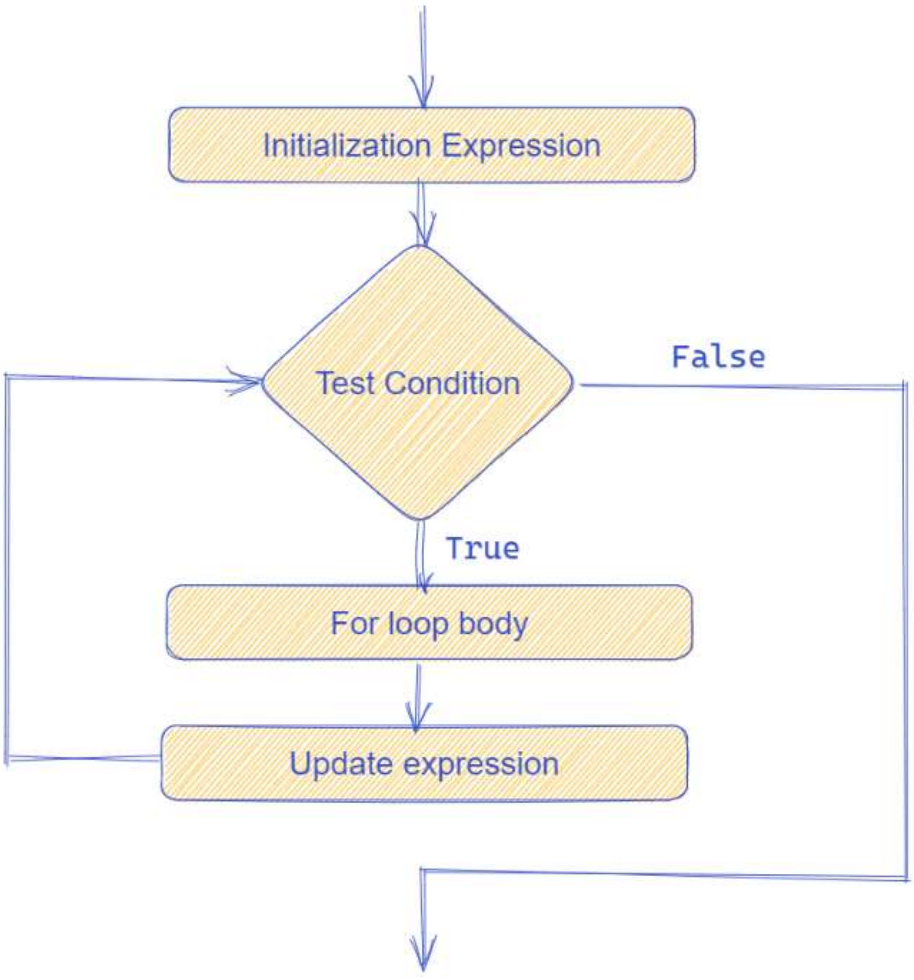
- **for loops**
- **while loops**
- **do-while loops**

1. For цикъл

1) Синтаксис

```
for (initialization; condition; update)
{
    //body of the loop
}
```

2) Последователност в стъпки на **for** цикъл:

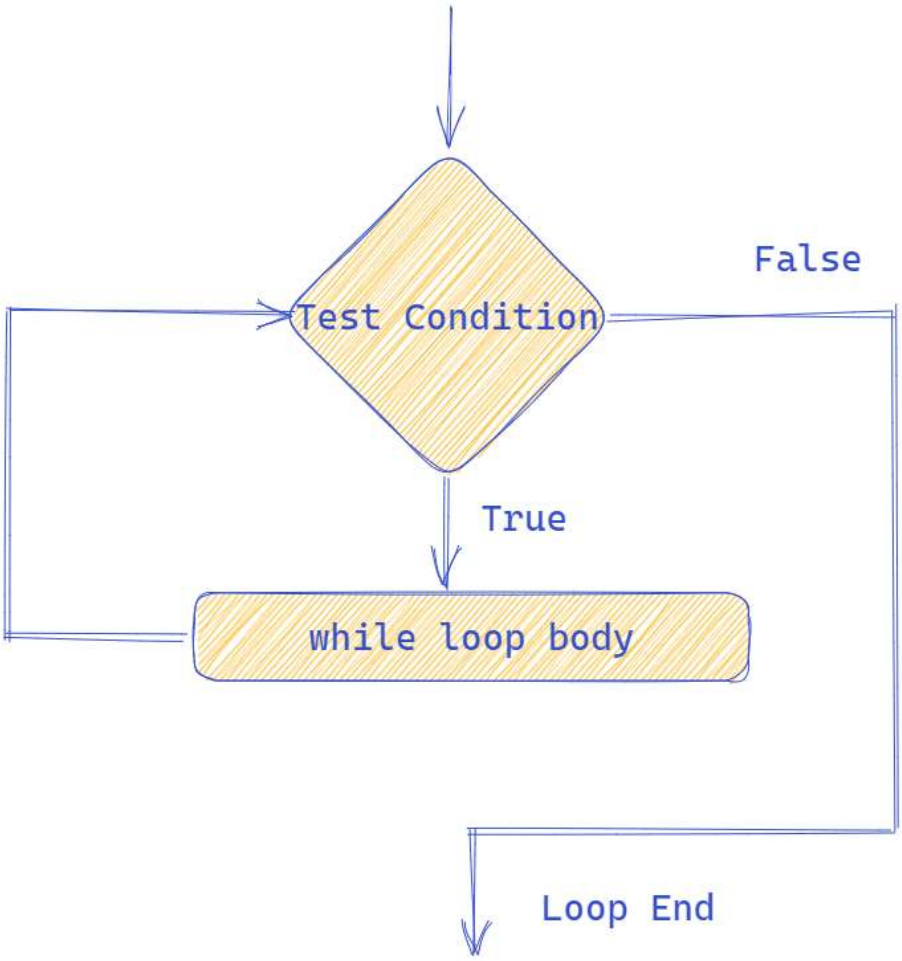


2. While цикъл

1) Синтаксис

```
while (condition)
{
    //body of the loop
}
```

2) Последователност в стъпки на **while** цикъл:

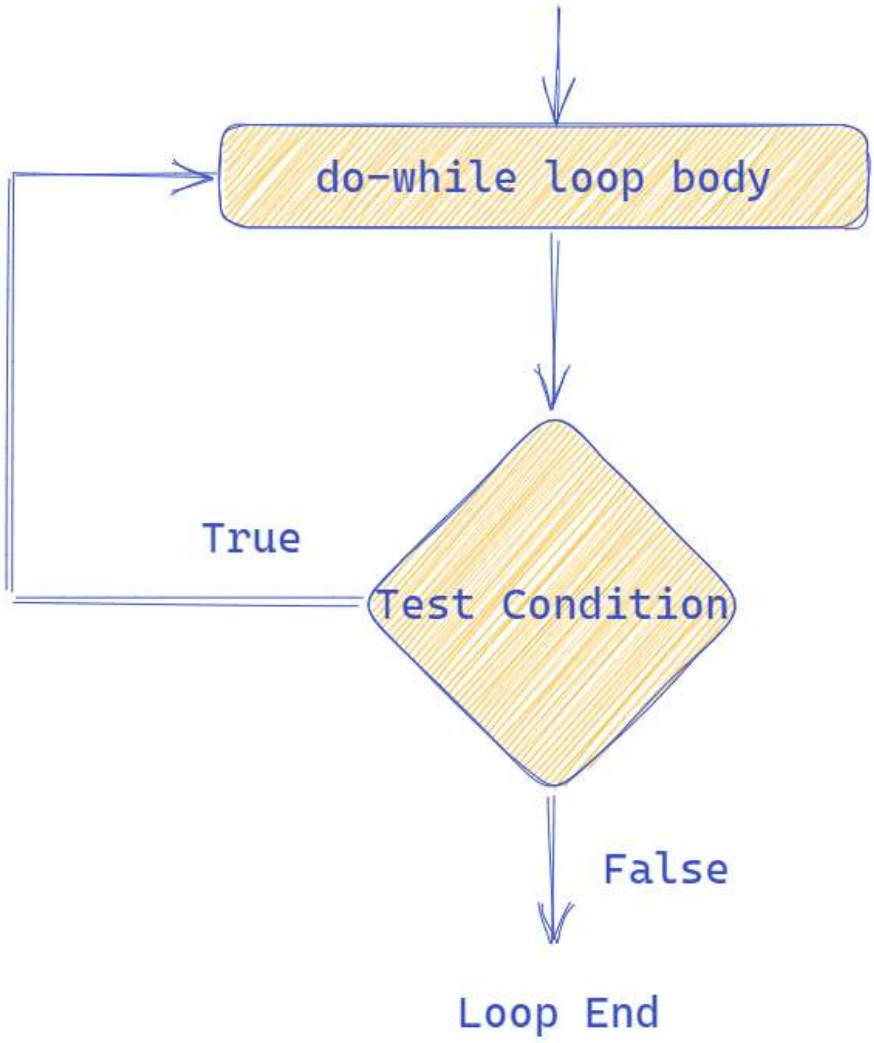


3. Do-while цикъл

1) Синтаксис

```
do
{
    //body of the loop
}
while (condition);
```

2) Последователност в стъпки на **do-while** цикъл:

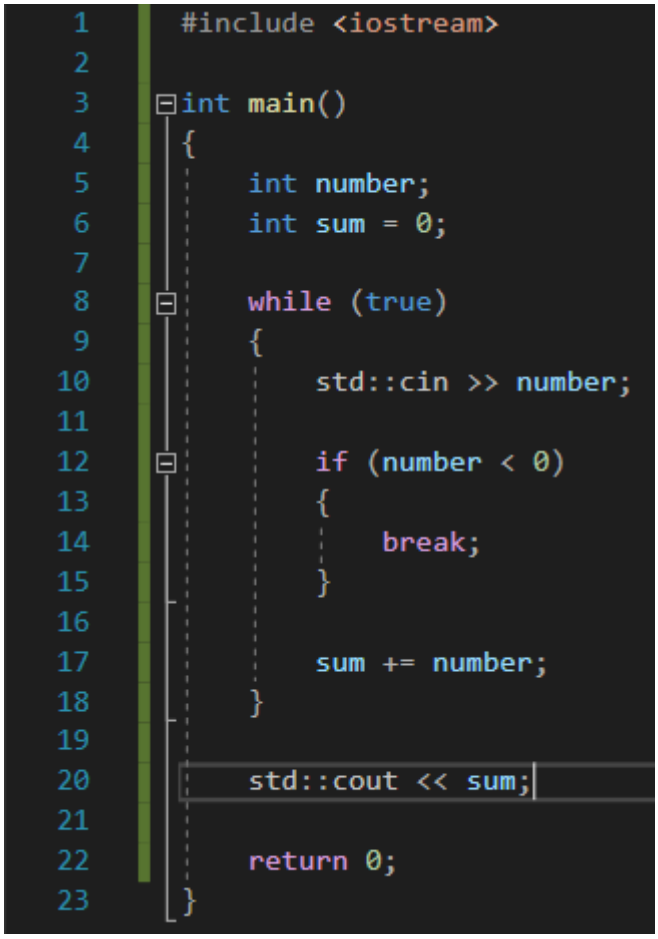


За разлика от **while** цикъла, в **do-while** се изпълнява първо блока от код и след това се прави проверка дали даденото условие е валидно.

3.Ключови думи при циклите

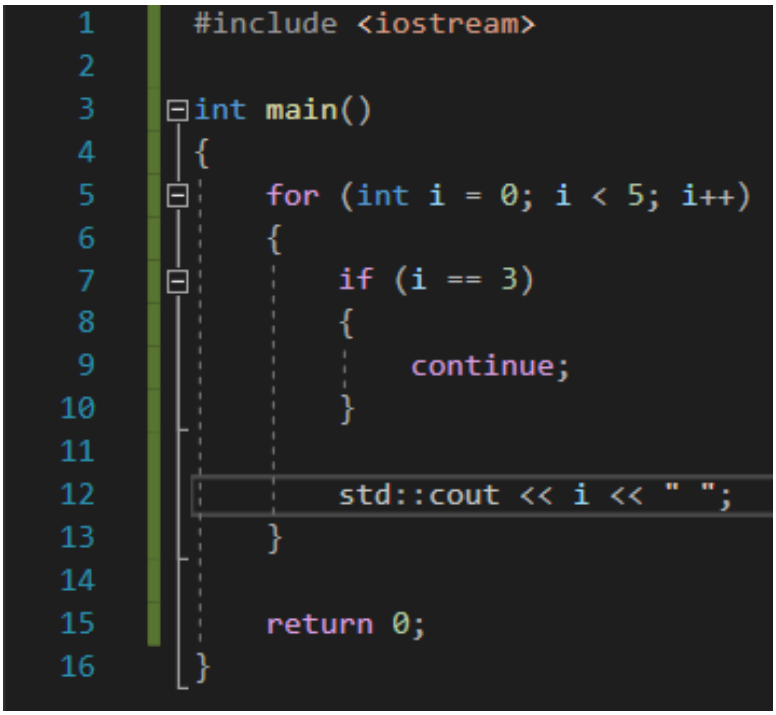
1) break

В езика C++ **break** е запазена дума или още ключова дума. Тя се използва, за да предотврати изпълнението на цикъла. Например нека въвеждаме числа докато не въведем отрицателно число, след което изведем на екрана сумата от предишните числа.



3) Continue

Continue се използва, когато искаме в даден момент да прескочим останалата част от кода в блока на цикъла и да го завъртим към следващата итерация. Нека изпечатаме на конзолата числата от 0 до 5 без 3:


#include <iostream>
int main()
{
for (int i = 0; i < 5; i++)
{
if (i == 3)
{
continue;
}
std::cout << i << " ";
}
return 0;
}