

Ravesli [Ravesli](#)


- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegEx](#)
- [Ассемблер](#)
- [Купить .PDF](#)


## Урок №68. Цикл do while

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 1 Авг 2020 |

 29407

[↑](#)  7

Одна интересная вещь в [цикле while](#) заключается в том, что если условие цикла изначально равно false, то тело цикла не будет выполняться вообще. Но иногда бывают случаи, когда нужно, чтобы цикл выполнялся хотя бы один раз, например, при отображении меню. Для решения этой проблемы C++ предоставляет цикл do while.

**Синтаксис do while в языке C++:**

```
do
    тело цикла;
while (условие);
```

Тело цикла do while всегда выполняется хотя бы один раз. После выполнения тела цикла проверяется условие. Если оно истинно, то выполнение переходит к началу блока do и тело цикла выполняется снова.

Ниже приведен пример использования цикла do while для отображения меню:

```
1  #include <iostream>
2
3  int main()
4  {
5      // Переменная choice должна быть объявлена вне цикла do while
6      int choice;
7
8      do
9      {
```

```
10     std::cout << "Please make a selection: \n";
11     std::cout << "1) Addition\n";
12     std::cout << "2) Subtraction\n";
13     std::cout << "3) Multiplication\n";
14     std::cout << "4) Division\n";
15     std::cin >> choice;
16 }
17 while (choice != 1 && choice != 2 &&
18        choice != 3 && choice != 4);
19
20 // Что-то делаем с переменной choice, например, используем оператор switch
21
22 std::cout << "You selected option #" << choice << "\n";
23
24 return 0;
25 }
```

Интересно, что переменная `choice` должна быть объявлена вне блоков `do while`. Почему так?

Если бы переменная `choice` была объявлена внутри блока `do`, то она была бы уничтожена при завершении этого блока еще до проверки условия `while`. Но нам нужна переменная, которая будет использоваться в условии `while`, следовательно, переменная `choice` должна быть объявлена вне блока `do`.

В целом, использовать `do while` вместо `while`, когда нужно, чтобы цикл выполнялся хотя бы один раз, является хорошей практикой.

Оценить статью:

★★★★★ (223 оценок, среднее: 4,93 из 5)



[← Урок №67. Цикл while](#)



[Урок №69. Цикл for →](#)

## Комментариев: 7



1. [metanastis:](#)  
[19 января 2019 в 06:52](#)

Дана вот такая задача:

Дана последовательность целых чисел, оканчивающаяся числом  $-1$ . Необходимо определить, присутствует ли в последовательности хотя бы одна пара одинаковых «соседних» чисел. Число  $-1$  членом последовательности не является.

Формат входных данных:

Вводится не более 215 целых чисел  $a_i$  ( $-215 \leq a_i \leq 215-1$ ;  $a_i \neq -1$ ). После последнего числа вводится  $-1$ . Гарантируется, что в последовательности есть хотя бы два числа.

Формат выходных данных:

Выведите «YES», если условие задачи выполняется, и «NO» в противном случае.

Решение:

```

1  #include<iostream>
2  using namespace std;
3                                     //Равные соседи
4  int main(){
5      int n,b;
6      bool flag=false;
7      cin>>n;
8
9      do{
10         cin>>b;
11         if(b==n)
12         {
13             flag=true;
14             n=b;
15         }
16
17     }while(b!=-1);
18
19     cout<<(flag?"YES":"NO")<<endl;
20
21     return 0;
22 }
```

Вопрос:

Почему "cin>> b" нужно обязательно вводить в цикле do? а не перед ним ? к примеру таким образом

```

1  int main(){
2      int n,b;
3      bool flag=false;
4      cin>>n;
5      cin>> b
```

??????????????

[Ответить](#)

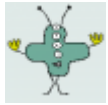


1. *Владимир:*

[11 декабря 2019 в 00:29](#)

Ну посмотрите сами внимательно — если вы так сделаете, откуда возьмется последовательность? Она и формируется введением нового числа в каждой итерации. Выносите ввод `b` за цикл — имеете разовый ввод двух чисел, их сравнение, результат флага, его вывод и все, конец программы.

[Ответить](#)



2. *Дмитрий:*

[8 июля 2018 в 15:41](#)

Добрый день! Подскажите, никак не могу догнать, как мне в этом примере организовать проверку деления на 0 ??

```
1  #include "stdafx.h"
2  #include <locale>
3  #include <string>
4  #include <iostream>
5  using namespace std;
6
7
8  double cal(double a, char z, double b)
9  {
10
11     switch (z)
12     {
13     case '+':
14         return a + b;
15         break;
16     case '-':
17         return a - b;
18         break;
19     case '*':
20         return a * b;
21         break;
22     case '/':
23         return a / b;
24         break;
25     default:
26         std::cout << "Incorrect operator " << std::endl;
27         break;
28
29     }
30 }
31
```

```
32 double getnum()
33 {
34     std::cout << "Input values : " ;
35     double x;
36     std::cin >> x;
37     return x;
38 }
39
40
41 char getOper()
42 {
43     char z;
44     std::cout << "input operator(+ or - or * or /): ";
45     std::cin >> z;
46     return z;
47 }
48
49 int main()
50 {
51     setlocale(LC_CTYPE, "rus");
52
53     std::cout << "Посчитаем?" << std::endl;
54
55     char answer='y';
56
57     while (answer == 'y' || answer == 'Y')
58     {
59         double a = getnum();
60         char z = getOper();
61         double b = getnum();
62
63
64         std::cout << cal(a, z, b) << std::endl;
65
66         std::cout << "Хотите продолжить? (y/n)";
67         std::cin >> answer;
68
69     }
70
71
72     return 0;
73 }
```

### Ответить



1. *Максим:*

[30 мая 2020 в 16:11](#)

Понимаю, что отвечать на комментарий 2-х летней давности тот еще моветон. Просто для данной "проблемы" идеально подойдет решение с циклом do while. И да, "вдруг кому-то пригодится".

Вместо строки 61:

```
1 double b = getnum();
```

нужно прописать эти строки:

```
1 double b;  
2 do  
3 {  
4     b = getnum();  
5     if (b == 0.0 && z == '/')  
6         std::cout << "На 0 делить нельзя!\n";  
7 }  
8 while (b == 0.0 && z == '/');
```

[Ответить](#)



3. *bm0802:*

[26 апреля 2018 в 01:35](#)

Пример так себе... вот тот же результат но без do:

```
1 #include <iostream>  
2  
3 int main()  
4 {  
5     int choice;  
6  
7     while (choice != 1 && choice != 2 &&  
8         choice != 3 && choice != 4)  
9     {  
10        std::cout << "Please make a selection: \n";  
11        std::cout << "1) Addition\n";  
12        std::cout << "2) Subtraction\n";  
13        std::cout << "3) Multiplication\n";  
14        std::cout << "4) Division\n";  
15        std::cin >> choice;  
16    }  
17  
18    std::cout << "You selected option #" << choice << "\n";  
19  
20    return 0;  
21 }
```

[Ответить](#)



1. *Юрий:*

[27 апреля 2018 в 23:54](#)

У вас сразу же выполняется проверка `choise`, прежде чем пользователю предоставится список, чтобы что-то выбрать. Т.е. пользователь еще ничего не выбрал, а вы уже проверяете то, что он ничего не выбрал — это первое действие. Затем вы уже показываете список — это второе действие в вашей программе.

В примере из урока, пользователю сначала предоставляется список для выбора — это первое действие, затем его выбор проверяется — это второе действие.

То, что оно работает у вашем примере и без `do` — то это задание можно реализовать еще проще: через `switch`, `if/else` и без `while` вообще. Напрашивается вопрос, зачем тогда придумали эти циклы `while`, `do/while`, `for`, если всё можно сделать и без них?

[Ответить](#)



1. *Игорь:*

[23 февраля 2019 в 20:29](#)

А разве не нужно в данном случае еще и присвоить значение переменной `choise`? Ведь значением не инициализированной переменной может быть всякий мусор из памяти, в том числе и 1,2,3 или 4... и тогда произойдет непреднамеренный автовыбор ))

[Ответить](#)

## Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены \*

Имя \*

Email \*

Комментарий






☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию

☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.

[TELEGRAM](#)  [КАНАЛ](#)  
[ПАБЛИК](#) 



## ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «SameGame» на C++/MFC](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте/Контакты](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020