

Ravesli [Ravesli](#)


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

Урок №48. Локальные переменные, область видимости и продолжительность жизни

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 11 Сен 2020 |

 29205

[| 20](#)

Этот материал является продолжением [урока №15](#).

Оглавление:

1. [Область видимости и продолжительность жизни](#)
2. [Соккрытие имен](#)
3. [Область видимости переменных](#)
4. [Параметры функций](#)
5. [Заключение](#)
6. [Тест](#)

Область видимости и продолжительность жизни

Прежде чем мы начнем, нам нужно сначала разобраться с двумя терминами: область видимости и продолжительность жизни. **Область видимости** определяет, где можно использовать переменную. **Продолжительность жизни** (или «*время жизни*») определяет, где переменная создается и где уничтожается. Эти две концепции связаны между собой.

Переменные, определенные внутри **блока**, называются **локальными переменными**. Локальные переменные имеют **автоматическую продолжительность жизни**: они создаются (и инициализируются, если необходимо) в точке определения и уничтожаются при выходе из блока. Локальные переменные имеют **локальную область видимости** (или «*блочную*»), т.е. они входят в область видимости с точки объявления и выходят в самом конце блока, в котором определены.

Например, рассмотрим следующую программу:

```
1 | #include <iostream>
```

```
2
3 int main()
4 {
5     int x(4); // переменная x создается и инициализируется здесь
6     double y(5.0); // переменная y создается и инициализируется здесь
7
8     return 0;
9
10 } // x и y выходят из области видимости и уничтожаются здесь
```

Поскольку переменные `x` и `y` определены внутри блока, который является главной функцией, то они обе уничтожаются, когда `main()` завершает свое выполнение.

Переменные, определенные внутри вложенных блоков, уничтожаются, как только заканчивается вложенный блок:

```
1 #include <iostream>
2
3 int main() // внешний блок
4 {
5     int m(4); // переменная m создается и инициализируется здесь
6
7     { // начало вложенного блока
8         double k(5.0); // переменная k создается и инициализируется здесь
9     } // k выходит из области видимости и уничтожается здесь
10
11     // Переменная k не может быть использована здесь, так как она уже уничтожена!
12
13     return 0;
14 } // переменная m выходит из области видимости и уничтожается здесь
```

Такие переменные можно использовать только внутри блоков, в которых они определены. Поскольку каждая функция имеет свой собственный блок, то переменные из одной функции никак не соприкасаются и не влияют на переменные из другой функции:

```
1 #include <iostream>
2
3 void someFunction()
4 {
5     int value(5); // value определяется здесь
6
7     // value можно использовать здесь
8
9 } // value выходит из области видимости и уничтожается здесь
10
11 int main()
12 {
13     // value нельзя использовать внутри этой функции
14
15     someFunction();
16
17     // value здесь также нельзя использовать
```

```
18 |  
19 |     return 0;  
20 | }
```

В разных функциях могут находиться переменные или параметры с одинаковыми именами. Это хорошо, потому что не нужно беспокоиться о возможности возникновения конфликтов имен между двумя независимыми функциями. В примере, приведенном ниже, в обеих функциях есть переменные `x` и `y`. Они даже не подозревают о существовании друг друга:

```
1  #include <iostream>  
2  
3  // Параметр x можно использовать только внутри функции add()  
4  int add(int x, int y) // параметр x функции add() создается здесь  
5  {  
6      return x + y;  
7  } // параметр x функции add() уничтожается здесь  
8  
9  // Переменную x функции main() можно использовать только внутри функции main()  
10 int main()  
11 {  
12     int x = 5; // переменная x функции main() создается здесь  
13     int y = 6;  
14     std::cout << add(x, y) << std::endl; // значение x функции main() копируется в  
15     return 0;  
16 } // переменная x функции main() уничтожается здесь
```

Вложенные блоки считаются частью внешнего блока, в котором они определены. Следовательно, переменные, определенные во внешнем блоке, могут быть видны и внутри вложенного блока:

```
1  #include <iostream>  
2  
3  int main()  
4  { // начало внешнего блока  
5  
6      int x(5);  
7  
8      { // начало вложенного блока  
9          int y(7);  
10         // Мы можем использовать x и y здесь  
11         std::cout << x << " + " << y << " = " << x + y;  
12     } // переменная y уничтожается здесь  
13  
14     // Переменную y здесь нельзя использовать, поскольку она уже уничтожена!  
15  
16     return 0;  
17 } // переменная x уничтожается здесь
```

Соккрытие имен

Переменная внутри вложенного блока может иметь то же имя, что и переменная внутри внешнего блока. Когда подобное случается, то переменная во вложенном (внутреннем) блоке «скрывает» внешнюю переменную. Это называется **сокрытием имен**:

```
1 #include <iostream>
2
3 int main()
4 { // внешний блок
5     int oranges(5); // внешняя переменная oranges
6
7     if (oranges >= 5) // относится к внешней oranges
8     { // вложенный блок
9         int oranges; // скрывается внешняя переменная oranges
10
11         // Идентификатор oranges теперь относится к вложенной переменной oranges.
12         // Внешняя переменная oranges временно скрыта
13
14         oranges = 10; // здесь мы присваиваем значение 10 вложенной переменной oranges
15
16         std::cout << oranges << std::endl; // выводим значение вложенной переменной
17     } // вложенная переменная oranges уничтожается
18
19     // Идентификатор oranges опять относится к внешней переменной oranges
20
21     std::cout << oranges << std::endl; // выводим значение внешней переменной oranges
22
23     return 0;
24 }
```

Результат выполнения программы:

```
10
5
```

Здесь мы сначала объявляем переменную `oranges` во внешнем блоке. Затем объявляем вторую переменную `oranges`, но уже во вложенном (внутреннем) блоке. Когда мы присваиваем `oranges` значение `10`, то оно относится к переменной во вложенном блоке. После вывода этого значения (и окончания внутреннего блока), внутренняя переменная `oranges` уничтожается, оставляя внешнюю `oranges` с исходным значением (`5`), которое затем выводится. Результат выполнения программы был бы тот же, даже если бы мы назвали вложенную переменную по-другому (например, `nbOranges`).

Обратите внимание, если бы мы не определили вложенную переменную `oranges`, то идентификатор `oranges` относился бы к внешней переменной и значение `10` было бы присвоено внешней переменной:

```
1 #include <iostream>
2
3 int main()
4 { // внешний блок
5     int oranges(5); // внешняя переменная oranges
6
7     if (oranges >= 5) // относится к внешней переменной oranges
```

```

8      { // вложенный блок
9          // Никакого определения внутренней переменной oranges здесь нет
10
11         oranges = 10; // это применяется к внешней переменной oranges, хоть мы и на
12
13         std::cout << oranges << std::endl; // выводим значение внешней переменной or
14     } // значением переменной oranges будет 10 даже после того, как мы выйдем из вл
15
16     std::cout << oranges << std::endl; // выводим значение переменной oranges
17
18     return 0;
19 } // переменная oranges уничтожается

```

Результат выполнения программы:

```

10
10

```

В обоих примерах на внешнюю переменную *oranges* никак не влияет то, что происходит с вложенной переменной *oranges*. Единственное различие между двумя программами — это то, к чему применяется выражение *oranges = 10*.

Соккрытие имен — это то, чего, как правило, следует избегать, поскольку оно может быть довольно запутанным!

Правило: Избегайте использования вложенных переменных с именами, идентичными именам внешних переменных.

Область видимости переменных

Переменные должны определяться в максимально ограниченной области видимости. Например, если переменная используется только внутри вложенного блока, то она и должна быть определена в нем:

```

1  #include <iostream>
2
3  int main()
4  {
5      // Не определяйте x здесь
6
7      {
8          // Переменная y используется только внутри этого блока, поэтому определяем ее
9          int x(7);
10         std::cout << x;
11     }
12
13     // В противном случае, переменная x может быть использована и здесь
14
15     return 0;
16 }

```

Ограничивая область видимости, мы уменьшаем сложность программы, поскольку число активных переменных уменьшается. Таким образом, легче увидеть, где какие переменные используются.

Переменная, определенная внутри блока, может использоваться только внутри этого же блока (или вложенных в него подблоков). Этим мы упрощаем понимание и логику программы.

Если во внешнем блоке нужна переменная, то её необходимо объявлять во внешнем блоке:

```
1 #include <iostream>
2
3 int main()
4 {
5     int y(5); // мы объявляем переменную y здесь, потому что она нам будет нужна как
6
7     {
8         int x;
9         std::cin >> x;
10        // Если бы мы объявили y здесь, непосредственно перед её первым фактическим
11        if (x == 4)
12            y = 4;
13    } // то она бы уничтожилась здесь
14
15    std::cout << y; // а переменная y нам нужна еще здесь
16
17    return 0;
18 }
```

Это один из тех редких случаев, когда вам может понадобиться объявить переменную до её первого использования.

Правило: Определяйте переменные в наиболее ограниченной области видимости.

Параметры функций

Хотя параметры функций не определяются внутри основного блока (тела) функции, в большинстве случаев они имеют локальную область видимости:

```
1 int max(int x, int y) // x и y определяются здесь
2 {
3     // Присваиваем большее из значений (x или y) переменной max
4     int max = (x > y) ? x : y; // max определяется здесь
5     return max;
6 } // x, y и max уничтожаются здесь
```

Заключение

Переменные, определенные внутри блоков, называются локальными переменными. Они доступны только внутри блока, в котором определены (включая вложенные блоки) и уничтожаются при завершении этого же блока.

Определяйте переменные в наиболее ограниченной области видимости. Если переменная используется только внутри вложенного блока, то и определять её следует внутри этого же вложенного блока.

Тест

Задание №1

Напишите программу, которая просит пользователя ввести два целых числа: второе должно быть больше первого. Если пользователь введет второе число меньше первого, то используйте блок и временную переменную, чтобы поменять местами пользовательские числа. Затем выведите значения этих переменных. Добавьте в свой код [комментарии](#), объясняющие, где и какая переменная уничтожается.

Результат выполнения программы должен быть примерно следующим:

```
Введите число: 4
Введите большее число: 2
Меняем значения местами
Меньшее число: 2
Большее число: 4
```

Подсказка: Чтобы использовать кириллицу, добавьте следующую строчку кода в самое начало функции `main()`:

```
1 | setlocale(LC_ALL, "rus");
```

Ответ №1

```
1 | #include <iostream>
2 |
3 | int main()
4 | {
5 |     // Используем кириллицу
6 |     setlocale(LC_ALL, "rus");
7 |
8 |     std::cout << "Введите число: ";
9 |     int smaller;
10 |    std::cin >> smaller;
11 |
12 |    std::cout << "Введите большее число: ";
13 |    int larger;
14 |    std::cin >> larger;
15 |
16 |    // Если пользователь ввел числа не так, как нужно
17 |    if (smaller > larger)
18 |    {
19 |        // То меняем местами эти значения
20 |        std::cout << "Меняем значения местами\n";
21 |
22 |        int temp = larger;
23 |        larger = smaller;
24 |        smaller = temp;
25 |    } // temp уничтожается здесь
```

```
26
27     std::cout << "Меньшее число: " << smaller << "\n";
28     std::cout << "Большее число: " << larger << "\n";
29
30     return 0;
31 } // smaller и larger уничтожаются здесь
```

Задание №2

В чём разница между областью видимости и продолжительностью жизни переменной? Какую область видимости и продолжительность жизни по умолчанию имеют локальные переменные (и что это значит)?

Ответ №2

Область видимости определяет, где переменная доступна для использования. Продолжительность жизни переменной определяет, когда переменная создается и когда уничтожается.

Локальные переменные имеют локальную (блочную) область видимости, доступ к ним осуществляется только внутри блока, в котором они определены.

Локальные переменные имеют автоматическую продолжительность жизни, что означает, что они создаются в точке определения и уничтожаются в конце блока, в котором определены.

Оценить статью:

★★★★★ (274 оценок, среднее: 4,95 из 5)



[← Урок №47. Блоки стейтментов \(составные операторы\)](#)



[Урок №49. Глобальные переменные →](#)

Комментариев: 20



1. Виктор:

[2 октября 2020 в 06:04](#)

```
1 #include <iostream>
2 #include <windows.h>
3
4 using namespace std;
5
6 int main(){
7     SetConsoleCP(1251);
8     SetConsoleOutputCP(1251);
9 }
```



```
10  int x;
11  cout<<"Введите целое число X: ";
12  cin>>x;
13  cout<<"Ваш X: "<<x<<endl;
14
15  int y;
16  cout<<"Введите целое число Y которое больше X: ";
17  cin>>y;
18  if (y>x)
19      cout<<"Ваш Y: "<<y<<" Больше чем X: "<<x<<endl;
20  else
21  {
22      cout<<"Меняем значения местами"<<endl;
23
24      int a(y);
25      cout<<"X = "<<a<<endl;
26
27      int b(x);
28      cout<<"Y = "<<b<<endl;
29  }
30
31
32  return 0;
33 }
```

Ответить



2. Александр:

17 июля 2020 в 23:05

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      setlocale(LC_ALL, "rus");
7      cout << "Введите число: " << "\n";
8      double number1; //объявление number1
9      cin >> number1;
10     cout << "Введите большее число" << "\n";
11     double number2; //объявление number2
12     cin >> number2;
13     if (number1 > number2)
14     {
15         double x = number1; //определяем x
16         number1 = number2;
17         number2 = x;
18         cout << "Меняем значения местами" << "\n";
19     } //переменная x уничтожается
20
21     cout << "Меньшее число: " << number1 << "\n";
```

```

22     cout << "Большее число: " << number2 << "\n";
23
24     return 0;
25 } //переменные number1, number2 уничтожаются

```

[Ответить](#)

3. Денис:

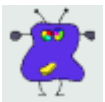
[11 июля 2020 в 22:36](#)

Думаю, так даже проще должно быть

```

1  #include <iostream>
2
3  int main()
4  {
5      setlocale(LC_ALL, "rus");
6
7      std::cout << "Введите первое число: ";
8      int a = 0;
9      std::cin >> a;
10
11     std::cout << "Введите второе число: ";
12     int b = 0;
13     std::cin >> b;
14
15     if (b > a)
16         std::cout << b << ">" << a << std::endl;
17     else
18     {
19         int x = a;
20         int y = b;
21         std::cout << x << ">" << y << std::endl;
22     } //уничтожаются переменные x и y
23
24     return 0;
25 } //уничтожаются переменные a и b

```

[Ответить](#)

4. Sagynysh:

[23 мая 2020 в 12:40](#)

```

1  #include<iostream>
2  using namespace std; // используем пространство
3  int main()
4  {
5      setlocale(LC_ALL,"rus"); // используем кириллицу
6      int x,y;
7      cout << "Введите меньшее число : ";
8      cin >> x ;

```

```
9      cout << "Введите большее число : ";
10     cin >> y;
11     if(x > y)
12     {                                     // начало вложенного блока
13         int z;                           // объявляем переменную z
14         z = x;
15         x = y;
16         y = z;
17     }                                     // конец вложенного блока,
18     cout << "большее число : " << y << endl;
19     cout << "меньшее число : " << x << endl;
20     system("pause");
21     return 0;
22 }
```

Ответить



5. Alex:

[29 апреля 2020 в 12:48](#)

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;    // используем именное пространство std
4
5
6  int main()
7  {
8      setlocale(LC_ALL, "rus");
9      began:
10     cout << "Введите число: ";
11     int x;
12     cin >> x;
13
14     cout << "Введите большее число: ";
15     int y;
16     cin >> y;
17
18     if (x == y)
19     {
20         cout << "Вы ввели одно и то же число дважды!" << endl;
21         goto began;
22     }
23
24     if (x < y)
25     {
26         cout << "Меньшее число: " << x << endl;
27         cout << "Большее число: " << y << endl;
28     }
29     else
30     {
31         int z = y;
```

```

32     cout << "Меняем значения местами" << endl;
33     cout << "Меньшее число: " << z << endl;
34     cout << "Большее число: " << x << endl;
35 } // z уничтожается здесь
36
37 } // x, y уничтожаются здесь

```

[Ответить](#)

6. Павел Карнов:
[11 февраля 2020 в 22:28](#)

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Введите число: ";           // выводим сообщение "Введите число: "
7      int x;                               // добавляем целое число x
8      cin >> x;                             // даем пользователю ввести значение x
9      cout << "Введите большее число: ";  // выводим сообщение "Введите большее ч
10     int y;                               // добавляем целое число y
11     cin >> y;                             // даем пользователю ввести значение y
12     int max;                             // добавляем целое число max
13     if (y > x)                            // если y больше x то...
14     {
15         max = y;                         // числу max присваивают значение y
16         y = x;                           // числу y присваивают значение x
17         x = max;                         // числу x присваивают значение max
18         cout << x << " " << y << endl;    // выводим переменные x и y через пробел
19     } else                                // если x больше или равен y то...
20     {
21         cout << x << " " << y << endl;    // выводим переменные x и y через пробел
22     }
23     return 0;
24 }

```

[Ответить](#)

7. Inviser666:
[8 января 2020 в 15:19](#)

```

1  #include <iostream>
2  #include <cstdlib>
3
4  int entfirstnum();
5  int entsecondnum();
6  void result(int x, int y);
7
8  int main() {

```

```

9     setlocale(LC_ALL, "rus");
10
11     int firstnum = entfirstnum();      //firstnum объявляется
12     int secondnum = entsecondnum();   //secondnum объявляется
13     result(firstnum, secondnum);
14
15     system("pause");
16 }                                     //уничтожаются переменные firstnum и secondnum
17 int entfirstnum() {
18     std::cout << "Введите число: ";
19     int x;                            //x объявляется
20     std::cin >> x;
21     return x;
22 }                                     //x уничтожается
23 int entsecondnum() {
24     std::cout << "Введите большее число: ";
25     int x;                            //x объявляется
26     std::cin >> x;
27     return x;
28 }                                     //x уничтожается
29 void result(int x, int y) {           //x и y определяются
30     if (x < y) {
31         std::cout << "меньшее число: " << x << '\n';
32         std::cout << "большее число: " << y << '\n';
33     }
34     else {
35         std::cout << "меняем значения местами" << '\n';
36         std::cout << "меньшее число: " << y << '\n';
37         std::cout << "большее число: " << x << '\n';
38     }
39 }                                     //x и y уничтожаются

```

Ответить



8. *armus1:*

30 сентября 2019 в 23:02

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      setlocale(LC_ALL, "rus");
7      int a;
8      cout << "Введите целое число:";
9      cin >> a;
10     int b;
11     cout << "Введите целое число, которое больше предыдущего:";
12     cin >> b;
13
14     if (a < b)

```

```

15     {
16         cout << "Меньшее число:" << a << endl;
17         cout << "Большее число:" << b << endl;
18     }
19     else
20     {
21         int z = b;
22         cout << "Меньшее число:" << z << endl;
23         cout << "Большее число:" << a << endl;
24     } // переменная z уничтожается
25
26     return 0;
27 } // переменные a и b уничтожаются

```

[Ответить](#)

9. Елена:

[9 июля 2019 в 22:04](#)

Еще вариант:

```

1 | a > b ? (cout << a << " , " << b << endl) : (cout << b << " , " << a << endl);

```

[Ответить](#)

10. Дмитрий:

[9 ноября 2018 в 23:19](#)

```

1 | #include <iostream>
2 |
3 |     /* такие маленькие программы МНЕ гораздо СПОДРУЧНЕЕ писать в ОДНОЙ функции
4 |
5 | int main()
6 | {
7 |     setlocale(0, "");
8 |
9 |     std::cout << "Введите целое число: ";
10 |    int a;
11 |    std::cin >> a;
12 |
13 |    std::cout << "Введите целое число побольше: ";
14 |    int b;
15 |    std::cin >> b;
16 |
17 |    // проверка ЕСЛИ введено ДВА РАЗА ОДНО И ТОЖЕ ЧИСЛО
18 |    if (a == b)
19 |    {
20 |        std::cout << "Вы ввели одно и то же число дважды!" << std::endl;
21 |        system("pause");
22 |        return 0;
23 |    }

```

```

24
25     // замена значений если Значения были введены НЕ так как требовалось
26     if (b < a)
27     {
28         std::cout << "Меняем значения местами" << std::endl;
29         int c;    // временная переменная C (область видимости локальная - внутри функции)
30         c = b;
31         b = a;
32         a = c;
33     }
34
35     std::cout << "Меньшее число: " << a << std::endl;
36     std::cout << "Большее число: " << b << std::endl;
37
38     system("pause");
39     return 0;
40 }

```

[Ответить](#)



11. **Игорь:**

[16 сентября 2018 в 22:41](#)

```

1  #include <iostream>
2
3  using namespace std;
4  int input();
5  void exchange(int little, int big);
6
7  int main()
8  {
9      setlocale(LC_ALL, "rus");
10     int little, big;
11     little = input();
12     big = input();
13     exchange(little, big);
14     return 0;
15 }
16 int input()
17 {
18     static int count = 0;
19     if(count == 1)
20     {
21         cout<<"Введите большее число:";
22     }
23     else
24         cout<<"Введите число:";
25
26     int number;
27     cin>>number;

```

```
29     count++;
30     return number;
31
32 }
33 void exchange(int little, int big)
34 {
35     if(little > big)
36     {
37         cout<<"Меняем значение:"<<endl;
38         int temp;
39         temp = little;
40         little = big;
41         big = temp;
42     }
43     cout<<"Меньшее число: "<<little<<endl;
44     cout<<"Большее число: "<<big<<endl;
45
46 }
```

[Ответить](#)



12. Максим:

[11 сентября 2018 в 21:30](#)

```
1  #include <iostream>
2
3  using namespace std;
4
5  int inputSmaller()
6  {
7      int x;
8      cout<<"Введите число: ";
9      cin>>x;
10     return x;
11 }//Уничтожается x функции inputSmaller
12
13 int inputLarger()
14 {
15     int x;
16     cout<<"Введите большее число: ";
17     cin>>x;
18     return x;
19 }//Уничтожается x функции inputLarger
20
21 void output(int smaller, int larger)
22 {
23     cout<<"Меньшее число: "<<smaller<<"\n";
24     cout<<"Большее число: "<<larger<<"\n";
25 }//Уничтожаются smaller, larger функции output
26
27 int main()
```



```

28 {
29     setlocale(LC_ALL, "rus");
30     int smaller(inputSmaller());
31     int larger(inputLarger());
32     if (smaller>larger)
33     {
34         cout<<"Меняем значения\n";
35         int intermediateVar(larger);
36         larger=smaller;
37         smaller=intermediateVar;
38     }//Уничтожается intermediateVar
39     output(smaller,larger);
40
41     return 0;
42 }//Уничтожаются smaller, larger функции main

```

Ответить



13. **Георгий:**

29 июля 2018 в 22:14

хм, сделал немного наоборот, но вроде смысл тот же

```

1  #include <stdafx.h>
2  #include <iostream>
3
4  int Enter()
5  {
6      std::cout << "Введите число: ";
7      int x;
8      std::cin >> x;
9      return x;
10 }
11
12 int EnterMax()
13 {
14     std::cout << "Введите большее число: ";
15     int x;
16     std::cin >> x;
17     return x;
18 }
19
20 int main()
21 {
22     setlocale(LC_ALL, "rus");
23     int a = Enter();
24     int b = EnterMax();
25     if (b > a)
26     {
27         std::cout << "Большее число: " << b << "\n";
28         std::cout << "Меньшее число: " << a << "\n";

```

```

29     }
30     else
31     {
32         std::cout << "Меняем значения \n";
33         std::cout << "Большее число: " << a << "\n";
34         std::cout << "Меньшее число: " << b << "\n";
35     }
36     system("pause");
37     return 0;
38 }

```

[Ответить](#)14. *master114:*[23 апреля 2018 в 11:48](#)

У меня получился такой вариант — ibb.co/jJLATc. Сделал пару функций. Перед выполнением думал, что получится поменять без временной переменной (в моем случае Z).

[Ответить](#)1. *Юрий:*[28 апреля 2018 в 00:00](#)

Вы правильно сделали, разбив программу на функции. А насчет временной переменной, то без неё здесь не обойтись 😊

[Ответить](#)1. *master114:*[4 мая 2018 в 15:38](#)

Нашел вариант без временной переменной — onlinegdb.com/B1E-4RKpM.

[Ответить](#)1. *Юрий:*[5 мая 2018 в 22:42](#)

Хоть и работает, но у вас здесь вместо одной временной переменной используются две временные переменные: xFunc и yFunc. + еще глобальные переменные, которые нежелательно лишний раз использовать.

Но в общем, реализация интересная однозначно.

2. *Mirovengil:*[19 августа 2019 в 12:39](#)

В каком смысле "без дополнительной переменной не обойтись"? А как же классика?

```
1 | a += b;  
2 | b = a - b;  
3 | a -= b;
```

PS: имена переменным поставил простейшие, чтобы было понятно, что происходит.

[Ответить](#)



15. *Sergey Groysman:*

[16 апреля 2018 в 15:50](#)

Юрий, доброго дня.

При решении теста №1, как сюда ввести вариант, если ввели равные числа?

Спасибо.

[Ответить](#)



1. *Юрий:*

[17 апреля 2018 в 22:56](#)

Если ввести правильные числа сразу, то они и выведутся после, без выполнения условия if. Дополнительно ничего прописывать не нужно. В условии if ведь и указывается, что менять значение нужно только при условии что значение smaller, которое ввел пользователь — больше значения large. Если же это не так, то ничего менять и не нужно.

[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

Электронная почта



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 «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

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