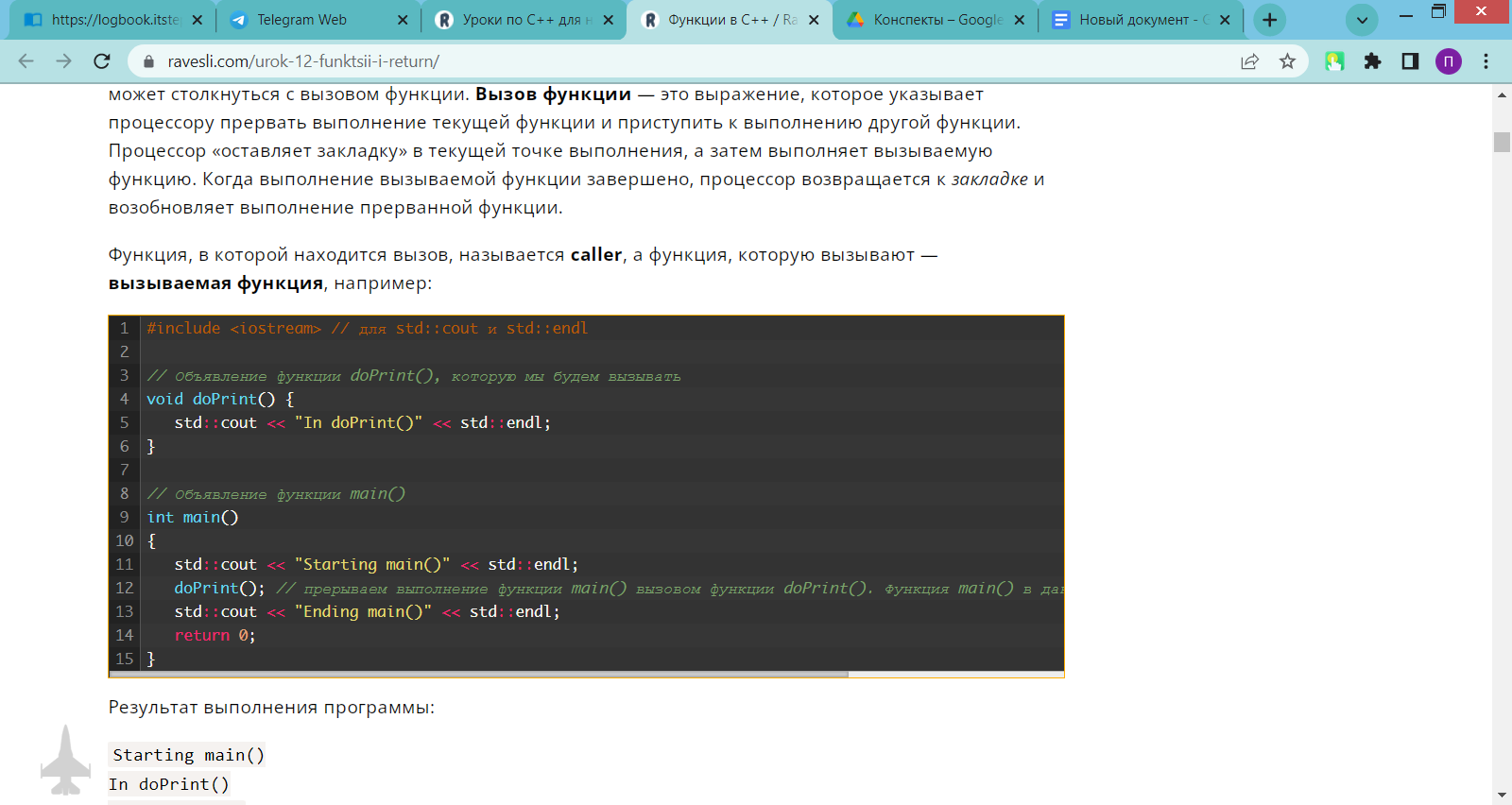
**Функции**

**Функция** — это последовательность стейтментов для выполнения определенного задания. Часто ваши программы будут прерывать выполнение одних функций ради выполнения других. Вы делаете аналогичные вещи в реальной жизни постоянно. Например, вы читаете книгу и вспомнили, что должны были сделать телефонный звонок. Вы оставляете закладку в своей книге, берете телефон и набираете номер. После того, как вы уже поговорили, вы возвращаетесь к чтению: к той странице, на которой остановились.

Программы на языке C++ работают похожим образом. Иногда, когда программа выполняет код, она может столкнуться с вызовом функции. **Вызов функции** — это выражение, которое указывает процессору прервать выполнение текущей функции и приступить к выполнению другой функции. Процессор «оставляет закладку» в текущей точке выполнения, а затем выполняет вызываемую функцию. Когда выполнение вызываемой функции завершено, процессор возвращается к *закладке* и возобновляет выполнение прерванной функции.

Функция, в которой находится вызов, называется **caller**, а функция, которую вызывают — **вызываемая функция**, например:



Результат выполнения программы:

Starting main()

In doPrint()

Ending main()

Эта программа начинает выполнение с первой строки функции main(), где выводится на экран следующая строка: Starting main(). Вторая строка функции main() вызывает функцию doPrint(). На этом этапе выполнение стейтментов в функции main() приостанавливается и процессор переходит к выполнению стейтментов внутри функции doPrint(). Первая (и единственная) строка в doPrint() выводит текст In doPrint(). Когда процессор завершает выполнение doPrint(), он возвращается обратно в main() к той точке, на которой остановился. Следовательно, следующим стейтментом является вывод строки Ending main().

Обратите внимание, для вызова функции нужно указать её имя и список параметров в круглых скобках (). В примере, приведенном выше, параметры не используются, поэтому круглые скобки пусты. Мы детально поговорим о параметрах функций на следующем уроке.

**Правило: Не забывайте указывать круглые скобки () при вызове функций.**

**Возвращаемые значения**

Когда функция main() завершает свое выполнение, она возвращает целочисленное значение обратно в операционную систему, используя **оператор return**.

Функции, которые мы пишем, также могут возвращать значения. Для этого нужно указать **тип возвращаемого значения** (или ***«тип возврата»***). Он указывается при объявлении функции, перед её именем. Обратите внимание, тип возврата не указывает, какое именно значение будет возвращаться. Он указывает только тип этого значения.

Затем, внутри вызываемой функции, мы используем оператор return, чтобы указать **возвращаемое значение** — какое именно значение будет возвращаться обратно в caller.

Рассмотрим простую функцию, которая возвращает целочисленное значение:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include <iostream>    // int означает, что функция возвращает целочисленное значение обратно в caller  int return7()  {      // Эта функция возвращает целочисленное значение, поэтому мы должны использовать оператор return      return 7; // возвращаем число 7 обратно в caller  }  int main()  {      std::cout << return7() << std::endl; // выведется 7      std::cout << return7() + 3 << std::endl; // выведется 10      return7(); // возвращаемое значение 7 игнорируется, так как функция main() ничего с ним не делает      return 0;  } |

Результат выполнения программы:

7

10

Разберемся детально:

   Первый вызов функции return7() возвращает 7 обратно в caller, которое затем передается в [**std::cout**](https://ravesli.com/urok-11-sout-cin-i-endl/#toc-0) для вывода.

   Второй вызов функции return7() опять возвращает 7 обратно в caller. Выражение 7 + 3 имеет результат 10, который затем выводится на экран.

   Третий вызов функции return7() опять возвращает 7 обратно в caller. Однако функция main() ничего с ним не делает, поэтому ничего и не происходит (возвращаемое значение игнорируется).

***Примечание:*** Возвращаемые значения не выводятся на экран, если их не передать объекту std::cout. В последнем вызове функции return7() значение не отправляется в std::cout, поэтому ничего и не происходит.

**Тип возврата void**

Функции могут и не возвращать значения. Чтобы сообщить компилятору, что функция не возвращает значение, нужно использовать [**тип возврата void**](https://ravesli.com/urok-29-void/). Взглянем еще раз на функцию doPrint() из вышеприведенного примера:

|  |  |
| --- | --- |
| 1  2  3  4  5 | void doPrint() // void - это тип возврата  {      std::cout << "In doPrint()" << std::endl;      // Эта функция не возвращает никакого значения, поэтому оператор return здесь не нужен  } |

Эта функция имеет тип возврата void, который означает, что функция не возвращает значения. Поскольку значение не возвращается, то и оператор return не требуется.

Вот еще один пример использования функции типа void:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #include <iostream>    // void означает, что функция не возвращает значения  void returnNothing()  {      std::cout << "Hi!" << std::endl;      // Эта функция не возвращает никакого значения, поэтому оператор return здесь не нужен  }  int main()  {      returnNothing(); // функция returnNothing() вызывается, но обратно в main() ничего не возвращает      std::cout << returnNothing(); // ошибка, эта строчка не скомпилируется. Вам нужно будет её закомментировать      return 0;  } |

В первом вызове функции returnNothing() выводится Hi!, но ничего не возвращается обратно в caller. Точка выполнения возвращается обратно в функцию main(), где программа продолжает свое выполнение.

Второй вызов функции returnNothing() даже не скомпилируется. Функция returnNothing() имеет тип возврата void, который означает, что эта функция не возвращает значения. Однако функция main() пытается отправить это значение (которое не возвращается) в std::cout для вывода. std::cout не может обработать этот случай, так как значения на вывод не предоставлено. Следовательно, компилятор выдаст ошибку. Вам нужно будет закомментировать эту строку, чтобы компиляция прошла успешно.

**Возврат значений функцией main()**

Теперь у вас есть понимание того, как работает функция main(). Когда программа выполняется, операционная система делает вызов функции main() и начинается её выполнение. Стейтменты в main() выполняются последовательно. В конце функция main() возвращает целочисленное значение (обычно 0) обратно в операционную систему. Поэтому main() объявляется как int main().

Почему нужно возвращать значения обратно в операционную систему? Дело в том, что возвращаемое значение функции main() является **кодом состояния**, который сообщает операционной системе об успешном или неудачном выполнении программы. Обычно, возвращаемое значение 0 (ноль) означает что всё прошло успешно, тогда как любое другое значение означает неудачу/ошибку.

Обратите внимание, по стандартам языка C++ функция main() должна возвращать целочисленное значение. Однако, если вы не укажете return в конце функции main(), компилятор возвратит 0 автоматически, если никаких ошибок не будет. Но рекомендуется указывать return в конце функции main() и использовать тип возврата int для функции main().

**Еще о возвращаемых значениях**

Во-первых, если тип возврата функции не void, то она должна возвращать значение указанного типа (использовать оператор return). Единственно исключение — функция main(), которая возвращает 0, если не предоставлено другое значение.

Во-вторых, когда процессор встречает в функции оператор return, он немедленно выполняет возврат значения обратно в caller и точка выполнения также переходит в caller. Любой код, который находится за оператором return в функции — игнорируется.

Функция может возвращать только одно значение через return обратно в caller. Это может быть либо число (например, 7), либо значение переменной, либо выражение (у которого есть результат), либо определенное значение из набора возможных значений.

Но есть способы обойти правило возврата одного значения, возвращая сразу несколько значений, но об этом детально мы поговорим на соответствующем уроке.

Наконец, автор функции решает, что означает её возвращаемое значение. Некоторые функции используют возвращаемые значения в качестве кодов состояния для указания результата выполнения функции (успешно ли выполнение или нет). Другие функции возвращают определенное значение из набора возможных значений. Кроме того, существуют функции, которые вообще ничего не возвращают.

**Повторное использование функций**

Одну и ту же функцию можно вызывать несколько раз, даже в разных программах, что очень полезно:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | #include <iostream>    // Функция getValueFromUser() получает значение от пользователя, а затем возвращает его обратно в caller  int getValueFromUser()  {     std::cout << "Enter an integer: ";     int x;     std::cin >> x;     return x;  }    int main()  {     int a = getValueFromUser(); // первый вызов функции getValueFromUser()     int b = getValueFromUser(); // второй вызов функции getValueFromUser()       std::cout << a << " + " << b << " = " << a + b << std::endl;       return 0;  } |

Результат выполнения программы:

Enter an integer: 4

Enter an integer: 9

4 + 9 = 13

Здесь main() прерывается 2 раза. Обратите внимание, в обоих случаях, полученное пользовательское значение сохраняется в переменной x, а затем передается обратно в main() с помощью return, где присваивается переменной a или b!

Также main() не является единственной функцией, которая может вызывать другие функции. Любая функция может вызывать любую другую функцию!

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #include <iostream>    void printO()  {     std::cout << "O" << std::endl;  }    void printK()  {     std::cout << "K" << std::endl;  }    // Функция printOK() вызывает как printO(), так и printK()  void printOK()  {     printO();     printK();  }    // Объявление main()  int main()  {     std::cout << "Starting main()" << std::endl;     printOK();     std::cout << "Ending main()" << std::endl;     return 0;  } |

Результат выполнения программы:

Starting main()

O

K

Ending main()

**Вложенные функции**

В языке С++ одни функции не могут быть объявлены внутри других функций (т.е. быть вложенными). Следующий код вызовет ошибку компиляции:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include <iostream>    int main()  {     int boo() // эта функция находится внутри функции main(), что запрещено     {        std::cout << "boo!";        return 0;     }       boo();     return 0;  } |

Правильно вот так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include <iostream>    int boo() // теперь уже не в main()  {     std::cout << "boo!";     return 0;  }    int main()  {     boo();     return 0;  } |