Содержание

[ОБЩАЯ ИНФОРМАЦИЯ 2](#_heading=h.gjdgxs)

[Тема занятия](#_heading=h.30j0zll) 2

[Цели и задачи 2](#_heading=h.1fob9te)

[Ожидаемый результат 2](#_heading=h.3znysh7)

[Структура занятия 3](#_heading=h.tyjcwt)

[ОПИСАНИЕ ЗАНЯТИЯ 4](#_heading=h.3dy6vkm)

# ОБЩАЯ ИНФОРМАЦИЯ

### Тема занятия

Оператор switch. Цикл for.

### Цели и задачи

* Познакомиться с типами данных, применяемых в современных языках программирования
* Научиться работать с символьными переменными
* Познакомиться с понятием “строка”
* Познакомиться с множественным условием ветвления
* Научиться использовать условия ветвления и циклы в С++

### Ожидаемый результат

По результатам занятия Слушатель должен знать:

* Что такое типы данных
* Почему их несколько
* Что такое строка в языках программирования
* Что такое множественное условие ветвления
* Что такое цикл for (в чем его принципиальное отличие от while цикла)

По результатам занятия Слушатель должен уметь:

* Работать с переменными различных типов данных
* Применять на практике полученные знания в области работы с дробями и символами
* Работать с множественным ветвлением
* Понимать и разбираться в выборе оператора для множественного ветвления
* Использовать цикл for (в случаях, когда на это есть необходимость)

### Структура занятия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тайминг занятия** | |  |  |  | Таблица 1 |
| № | Этапы | Что делает преподаватель | Что делает Слушатель | Время | Общее время |
| 1 | Приветственное слово преподавателя |  |  | 5 мин. | 5 мин. |
| 2 | Повторение пройденного |  |  | 5 мин. | 40 мин. |
| 3 | Теоретическая часть |  |  | 30 мин. |
| 4 | Вопросы по теоретической части |  |  | 5 мин. |
|  | *Перерыв* |  |  | 15 мин. | 15 мин. |
| 5 | Практическая часть |  |  | 40 мин. | 45 мин. |
| 6 | Рефлексия и вопросы |  |  | 5 мин. |

# ОПИСАНИЕ ЗАНЯТИЯ

**Теоретическая часть**

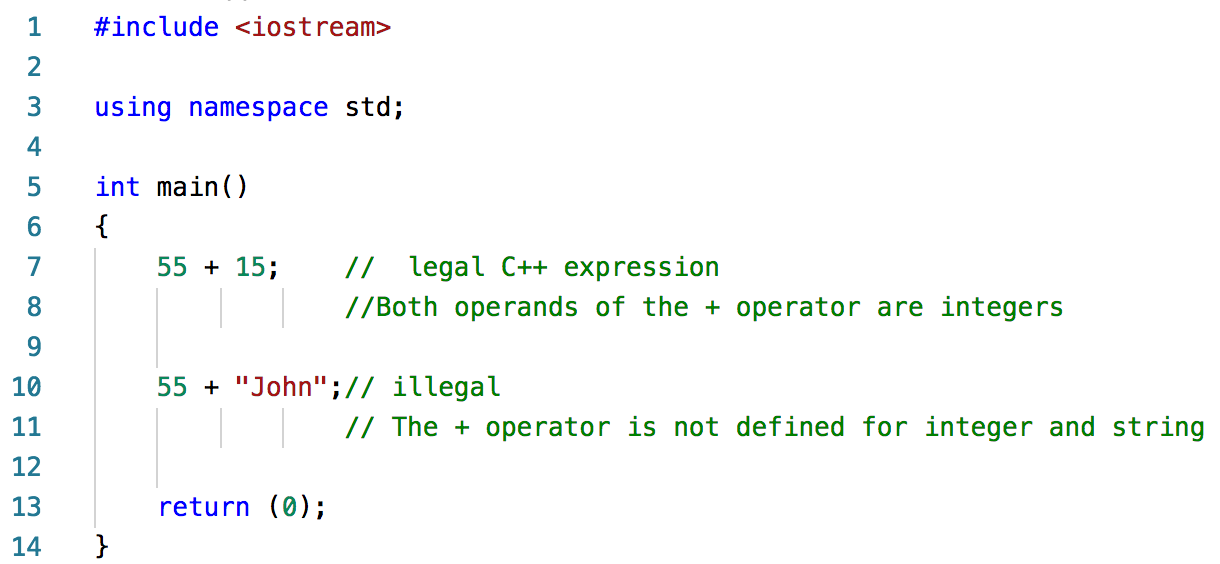
**Типы данных**

Операционная система распределяет память и выбирает, что будет сохранено в выделенную память, основываясь на типе данных переменной.

Тип данных определяет правильное использование идентификатора (имени), какой тип информации может быть сохранен, и какие типы операций могут быть выполнены.

**Выражения**

В приведенных ниже примерах показано правильное (legal) и неправильное (illegal) использование выражений языка C++.



**Числовые типы данных**

Числовые типы данных включают:

Целые числа, такие как -7, 42.

Числа с плавающей точкой (десятичные дроби), такие как 3.14, -42.67.

**Строки и Символы**

Строки состоят из чисел, букв, или символов. Строковые литералы размещаются в двойных кавычках; примеры "Hello", "My name is David".

Символы это одиночные буквы или символы, они должны быть закрыты между одинарными кавычками, например 'a', 'b', и т.д.

**Булевый тип**

Булевый тип возвращает только два возможных значения: true(истина) -> (1) и false(ложь) -> (0).

Условные выражения являются примером Булевого типа данных.

**Целочисленный тип**

Целочисленный тип хранит не дробные числа, которые могут быть положительными или отрицательными. Например, к целочисленному типу может относиться 42, -42, и подобные числа.

Размер целочисленного типа варьируется согласно с архитектурой системы, на которой запускается программа, однако 4 байта это минимальный размер в большинстве современных систем.

Используйте ключевое слово int для определения целочисленного типа.

**Целочисленный тип**

Некоторые из базовых типов, включая целочисленный, могут быть модифицированы используя один или несколько следующих спецификаторов:

**signed**: Знаковый целый тип может хранить и отрицательные и положительные значения.

**unsigned**: Беззнаковые целые могут хранить только положительные значения.

**short**: Половина стандартного размера.

**long**: Удвоенный стандартный размер.

Целочисленный тип данных резервирует 4-8 байт в зависимости от операционной системы.

**Числа с плавающей точкой**

Переменные типа с плавающей точкой могут хранить числа, такие как 420.0, -3.33, или 0.03325. Слова «плавающая точка» ссылаются на тот факт, что некоторое количество цифр может быть до и после десятичной точки. Вы можете сказать, что десятичная точка имеет способность "плавать".

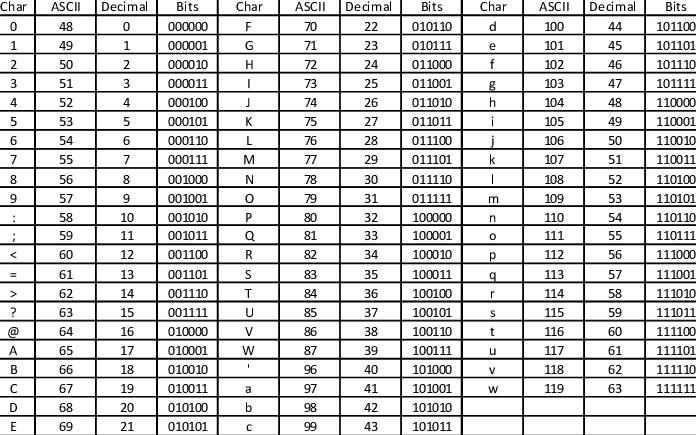
Существует три различных типа данных с плавающей точкой: float, double, и long double.

В большинстве современных архитектур, float равен 4 байтам, double равен 8, и long double может быть равен типу double (8 байт), или 16 байт.

Числа с плавающей точкой всегда являются знаковыми, что значит, что они могут хранить и положительные и отрицательные значения.

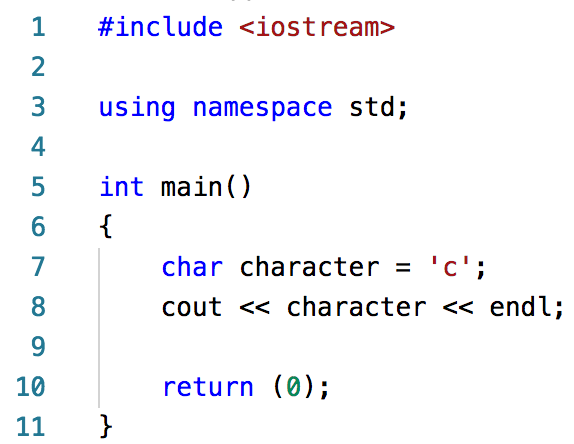
**Символьный тип**

Переменная типа char хранит целое число размером 1 байт. Однако, вместо интерпретирования значения типа char как целого, значение переменной типа char интерпретируется как символ набора ASCII.



**Символьный тип**

Символы представляются закрытыми между одиночными кавычками (например 'a', 'b', и т.д.).



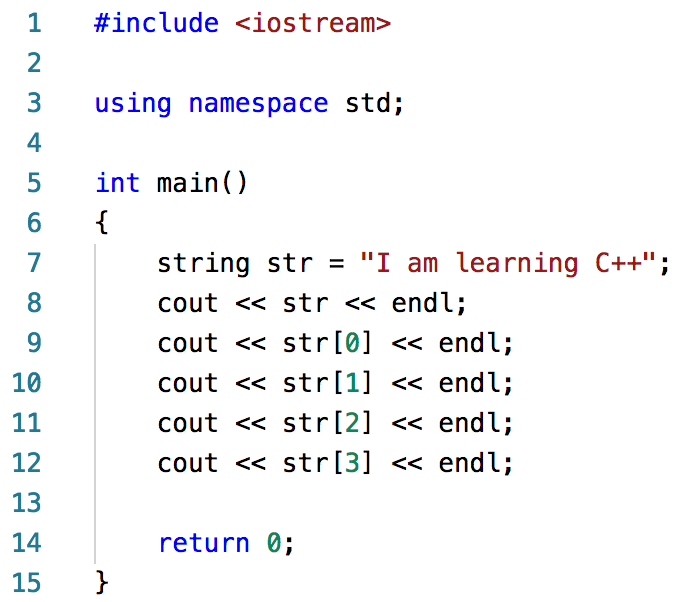
**Строки**

Строка -это упорядоченная последовательность символов, заключенная в двойные кавычки.

Строки являются частью Стандартной Библиотеки.

Вам необходимо включить библиотеку <string> чтобы использовать тип данных string. Либо вы можете использовать библиотеку, которая включает библиотеку с типом данных string.

Библиотека <string> включена в библиотеку <iostream>, так что вам не нужно отдельно включать <string>, если вы уже включили <iostream>.



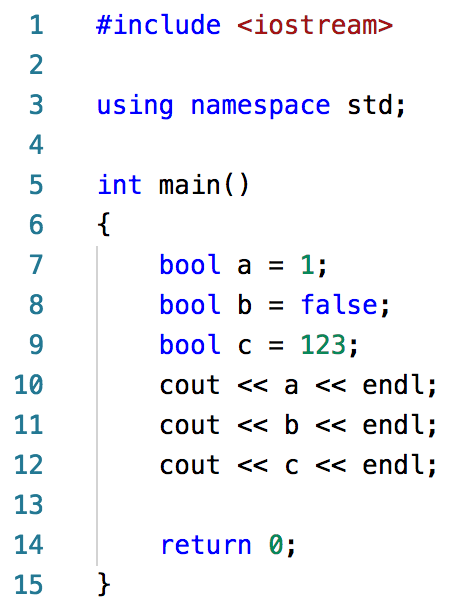
**Булевый тип**

Переменные булевого типа имеют только два возможных значения: true(истина) (1) и false(ложь) (0).

Для объявления булевой переменной используйте ключевое слово **bool**.

Если Булевое значение присвоено целому числу, то истинa становится 1, а ложь становится 0.

Если целое значение присвоено Булевому, то 0 становится false (ложью) и любое значение не равное нулю становится true (истинным).



**Правила наименования переменных**

Используйте следующие правила при наименовании переменных:

- Все переменные должны начинаться с буквы алфавита или с нижнего подчеркивания ( \_ ).

- После первой буквы имя переменной может содержать дополнительные буквы, а также и числа. Пробелы или специальные символы использовать в имени переменных нельзя.

**Чувствительность к регистру**

C++ чувствителен к регистру, это означает, что идентификатор написанный в верхнем регистре не эквивалентен такому же другому, но в нижнем регистре.

Например, **myvariable** это не тоже самое, что и **MYVARIABLE** и не то же самое, что и **MyVariable**.

**Это три разные переменные!**

**Выбирайте имена переменных, которые подходят к использованию, например: firstName, lastName.**

**Правила наименования переменных**

Ключевые слова C++ (зарезервированные слова) не могут быть использованы для наименования переменных.

Например, int, float, double, cout не могут быть использованы в качестве имени переменной.

Не существует предела длины имени переменных (в зависимости от оборудования), но старайтесь давать переменным практичные и осмысленные имена.

**Множество условий**

Иногда есть необходимость проверить переменную на равенство множеству значений. Это может быть достигнуто с помощью множества операторов if.

Ситуации, в которых необходимо использование нескольких операторов if, возникают часто, в связи с этим было принято решение о создании нового оператора.

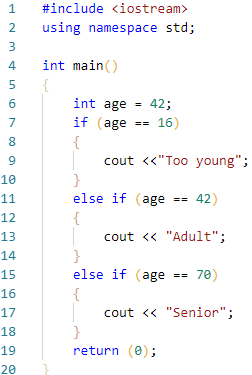
Оператор множественного условия ветвления switch (переключатель)

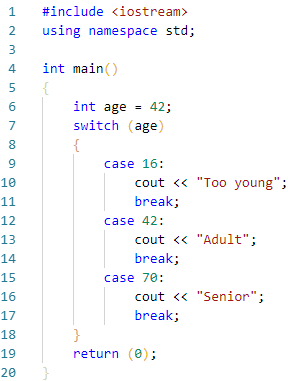
**Оператор switch**

Оператор switch проверяет переменную со списком значений, которые называются cases(случай), для определения которому из них оно равно.

Оператор switch вычисляет выражение для определения, которому из списка случаев оно соответствует. При нахождении соответствия выполняются выражения данного случая.

Оператор switch может содержать любое количество операторов case, за которыми следует значение и двоеточие





**Случай по умолчанию (default case)**

В операторе switch, может быть использован случай по умолчанию, если ни один из случаев не был реализован.

Код оператора default выполняется, если ни один из случаев не соответствует выражению оператора switch.

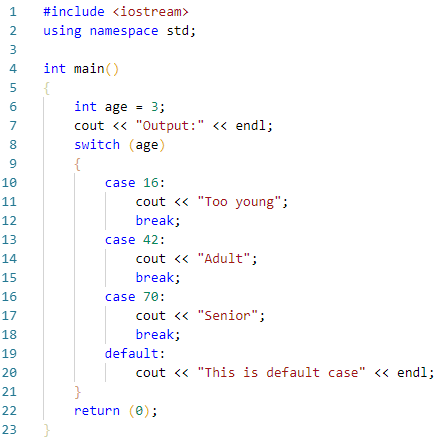
Оператор default должен находиться в конце оператора switch.

**Оператор break**

Роль оператора break - закончить оператор switch.

В образцах, в которых переменная соответствует значению в case, выражение, которое следует после оператора case, продолжает выполняться до оператора break. Другими словами, если опустить оператор break будут выполняться все выражения в следующих операторах case, даже те, которые не соответствуют выражению switch.

Оператору default не требуется оператор break, т.к. он является последним оператором case.



**Цикл for**

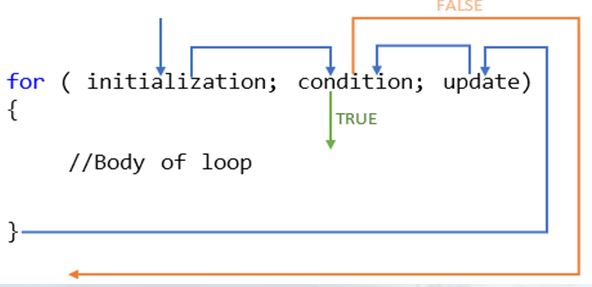
Цикл for является повторяющимся управляющим выражением, которое позволяет вам эффективно создать цикл, повторяющийся специальное количество раз.

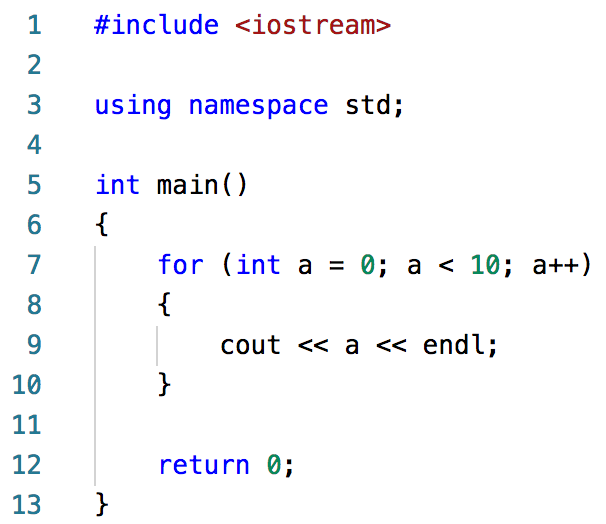
Шаг init выполняется первым и больше не повторяется.

Далее, высчитывается условие и, если оно верно, выполняется тело цикла.

Следующим шагом инкрементирующее выражение обновляет переменную, управляющую циклом.

Затем, повторяется тело цикла, останавливаясь только в случае, если условие станет ложным.





**Практическая часть**

**Задание 1**

Найти сумму чисел от 1 до 100.

**Задание 2**

Реализуйте таблицу умножения с помощью цикла for.

**Задание 3**

Напишите программу-калькулятор. Пользователь вводит 2 числа и выбирает арифметический оператор, ваша программа выдаёт результат.

\*

Добавьте поясняющие предложения

Например, “Please, enter first number”, ….

\*\*

Проверка на ноль