



# С++ для начинающих программистов

Длительность курса: 22 академических часа

## 1 Что такое современный C++

### Цели занятия:

#### 1.1.1 Современный C++

Поговорим об особенностях языка C++ - как он развивался, и каков он сейчас. Познакомимся со стандартом языка, научимся отличать `undefined behavior` от `unspecified behavior`.

#### 1.1.2 Hello, World!

Скомпилируем свою первую программу! Разберём процесс сборки приложений на C++, научимся не бояться ошибок и внимательно читать предупреждения. Также узнаем о существовании `online`-инструментов, с помощью которых можно начать работать уже сейчас.

#### 1.1.3 CMake

Разберем, что такое CMake и зачем нам нужен этот инструмент. Напишем свой первый `CMakeLists.txt` и научимся собирать свои проекты с помощью CMake.

---

## 2 Базовый синтаксис языка C++

### Цели занятия:

#### 1.2.1 Синтаксис C++

Посмотрим, как в наших программах представить числа, символы и массивы данных. Разберём много примеров. Познакомимся с более сложными (и более удобными) типами в C++.

#### 1.2.2 Ветвления

Продолжим изучать синтаксис. На этот раз изучим синтаксис ветвлений `if-else` и конструкцию `switch-case`. Это поможет сделать наши программы более сложными и интересными.

#### 1.2.3 Циклы

Разберём на примерах синтаксис циклов в языке C++. Научимся использовать `for`, `while` и даже `do while`.

---

### 3 Такие разные функции. Модульность

#### Цели занятия:

##### 1.3.1 Функции

Разберемся в функциями - что это, зачем они нужны и как их можно использовать. Посмотрим синтаксис и потренируемся на нескольких примерах.

##### 1.3.2 Модульность

Как создавать сложные программы? В этом нам поможет модульность языка C++ (которой, правда, почти нет). И снова разберём несколько примеров.

##### 1.3.3 Разное про функции

Разберём несколько особенностей использования функций в языке C++. Познакомимся с особенностью функции main, разберём, зачем нужен стек вызовов, а также поэкспериментируем с рекурсией.

## 1 Структуры и классы

### Цели занятия:

2.1.1 часть 1. Привет, структура.

Структуры. Вкратце про полиморфизм, наследование и инкапсуляцию

2.1.1 часть 2. Подробнее о структуре, привет объединение. Структура как отдельный тип данных. Объединения.

2.1.2 часть 1. Здравствуй, класс. Привет, объект.

Класс и объект. Функция-член класса. Доступ к членам класса

2.1.2 часть 2. Конструкторы и деструкторы.

Конструктор и деструктор. Сравнение со структурой.

2.1.3 часть 1. Немного о наследовании.

Наследование. Доступ к членам базового класса. Защищённые члены.

2.1.3 часть 2. Еще немного о наследовании.

Параметры конструктора. Чтение графов наследования

---

## 2 Полиморфизм и все-все-все

### Цели занятия:

2.2.1 часть 1. Дружба и перегрузка.

Действия с объектом. Дружественные функции.

Перегрузка конструктора.

2.2.1 часть 2. Передача и возврат.

Передача и возврат объекта функции. Конструктор копии.

2.2.2 часть 1. Производные типы.

Виртуальные функции. Указатели и ссылки на производные типы.

2.2.2 часть 2. Виртуальные функции и абстрактные класс. Виртуальные функции. Абстрактные классы.

2.2.3 часть 1. Перегружаем операторы.

Перегрузка операторов. Унарных и бинарных. Общие правила.

---

### 3 Шаблоны классов и функций

#### Цели занятия:

2.2.3 часть 2. Еще о перегрузке операторов.

С использованием функций-членов и не членов. Советы по перегрузке операторов.

2.3.1 часть 1. Обобщенные функции.

Обобщенные функции. С двумя обобщенными типами.

2.3.1 часть 2. Больше про обобщенные функции.

Явная перегрузка обобщенной функции. Перегрузка шаблона.

2.3.2 часть 1. Обобщенный класс

Обобщенный класс. Обобщенный класс с двумя типами.

2.3.2 часть 2". Больше про обобщенный класс.

Обобщенный класс безопасного массива. Аргументы, не являющиеся типами, для обобщенного класса.

## 1 Общий обзор стандартной библиотеки

### Цели занятия:

#### 3.1.1 Стандартная библиотека C++

Мы познакомимся со стандартной библиотекой C++, посмотрим, из каких частей она состоит, а так же разберемся, зачем нам нужно ее изучить.

#### 3.1.2 Ввод/вывод в C++

На этом занятии мы познакомимся с концепцией потоков, посмотрим, какие стандартные потоки есть в C++, для чего они нужны и какие у них есть особенности. Вы научитесь эффективно применять стандартные потоки ввод-вывода при написании программ на C++.

#### 3.1.3 `std::string`

Мы разберемся с понятием строки в C/C++, посмотрим, какие проблемы есть при использовании сишных строк, и как эффективно их можно заменить с помощью `std::string`. Вы освоите основные операции с `std::string` в части создания строки, поиска и изменения ее подстрок.

---

## 2 Подробнее о контейнерах и вводе-выводе

### Цели занятия:

#### 3.2.1 `std::array`

На занятии мы разберемся, какие проблемы есть у стандартных массивов в C/C++, и зачем нужен `std::array`. Вы поймете, в каких случаях можно его применять и научитесь делать это эффективно.

#### 3.2.2 `std::vector`

Разберемся с самым часто применяемым контейнером в STL. Мы разберемся, какие проблемы могут быть при бездумном его применении и как использовать `std::vector` максимально эффективно.

#### 3.2.3 `std::list`

На занятии посмотрим, что такое двусвязный список и как с ним работать. Так же разберемся какие накладные расходы несет в себе его применение и как `std::forward_list` может от них избавить.

#### 3.2.4 `std::deque`

На этом занятии познакомимся с двусторонней очередью, посмотрим, как она устроена и какими свойствами обладает. Вы освоите интерфейс этого контейнера.

#### 3.2.5 Упорядоченные ассоциативные контейнеры STL

Мы рассмотрим упорядоченные ассоциативные контейнеры, какими они бывают, как устроены внутри и какими свойствами обладают. Вы освоите приемы эффективной работы с такими контейнерами.

#### 3.2.6 Неупорядоченные ассоциативные контейнеры STL

На занятии посмотрим на неупорядоченные ассоциативные контейнеры. Разберемся, чем они отличаются от упорядоченных, как устроены внутри и как и когда их применять.

---



## 3 Алгоритмы, которые лучше знать лучше

### Цели занятия:

#### 3.3.1 Итераторы в STL

Это занятие построит мост от контейнеров к алгоритмам. Мы разберемся с важным понятием итератора, посмотрим на то, какой он бывает и почему может стать не валидным.

3.3.2 Модифицирующие и немодифицирующие алгоритмы STL На занятии рассмотрим некоторые из множества алгоритмов в STL: `any_of`, `copy`, `for_each` и другие. Отучимся от повсеместного применения сырых циклов и освоим методы создания более выразительного кода.

#### 3.3.3 Алгоритмы поиска и сортировки STL

На этом занятии мы посмотрим на алгоритмы сортировки и поиска в упорядоченных последовательностях. Узнаем, какие есть алгоритмы помимо `find` и насколько они эффективно работают.

#### 3.3.4 Алгоритмы STL: на множествах, на куче, перестановки, числовые

Этим занятием наше знакомство с алгоритмами в STL. Мы посмотрим, как можно работать с множествами, что такое куча как структура данных и как с ней работать, а также освоим полезный алгоритм `accumulate`.

## 1 Многопоточность или зачем в процессоре больше одного ядра

### Цели занятия:

#### 4.1.1 1\_1\_ThermsTherms

Терминология многопоточности. Поймём, что такое поток выполнения, как операционная система справляется с ситуацией, когда их несколько. Поговорим о том, что же такое процесс.

#### 4.1.2 1\_2\_How\_it\_works

В этом видео разберёмся с понятиями многопоточность, многозадачность, параллельность и псевдопараллельность. Разберем на житейском примере, как может быть реализована многопоточность - чем отличается параллельность от псевдопараллельности.

#### 4.1.3 1\_3\_Threads

Рассмотрим инструменты работы с потоками, которые предоставляет программисту стандартная библиотека C++. Посмотрим на примерах, как создавать потоки и что с ними делать потом.

#### 4.1.4 1\_4\_Mutex

В этом видео обсудим проблему синхронизации потоков. Зачем она нужна, и что может произойти, если её нет.

---

## 2 **Исключения есть всегда, и С++ - не исключение.**

### **Цели занятия:**

#### 4.2.1 2\_1\_Base

Исключения: начало. Обсудим механизм исключений, поймем, зачем они нужны. Изучим базовый синтаксис работы с исключениям, на примерах посмотрим, как они работают.

#### 4.2.2 2\_2\_Exception\_vs\_errors

Зачем что-то менять, если и так хорошо? Рассмотрим на примерах отличие механизма исключений от классической обработки кодов ошибок. Много примеров и понимание преимуществ исключений гарантировано.

#### 4.2.3 2\_3\_Niceties

Тонкости и детали - что может быть лучше? На примерах разберем - как построить иерархию исключений, почему лучше перехватывать исключения по ссылке, а также принципы работы с исключениями при использовании стандартной библиотеке.

---

**Цели занятия:**

4.3.1 Network

Зачем изучать работу с сетью, что даёт нам возможность соединить несколько компьютеров по сети. Понятие сервера и клиента, понимание отличий. Небольшой обзор RESTfull API.

4.3.2 Boost\_asio

Какие средства для работы с сетью предоставляет C++. Какое место в работе с сетью занимает Boost.Asio. Рассмотрим много примеров и освоим эту библиотеку.

4.3.3 Boost\_beast

Библиотека Boost.Beast - в чем отличие от Boost.Asio и как её использовать. Рассмотрим на примерах.

4.3.4 POCO

Библиотека POCO. Познакомимся с ещё одной библиотекой для разработки сетевого программного обеспечения. Рассмотрим достоинства и недостатки, изучим функционал на примерах.