

# Объектно-ориентированное программирование

Основы C++

Гришмановский Павел Валерьевич,  
кафедра автоматики и компьютерных систем, Политехнический институт, СурГУ

# Некоторые отличия от C

Некоторые конструкции, унаследованные от C, трактуются иначе:

## 1. Типы данных

- перечисление (enum) не эквивалентно типу int
- тэги перечислений, структур и объединений (смесей) являются самостоятельными именами типов и не требуют обязательного использования совместно с enum, struct, union
- недопустимо автоматическое приведение типов указателей кроме приведения к нетипизированному (но не обратно) и в пределах иерархии классов

## 2. Переменные

- переменная, определенная в секции инициализаторов цикла for, действует только в пределах цикла
- недопустимо определение переменной с типом int по умолчанию

## 3. Функции

- запрещен вызов функции без предварительного описания
- пустой список формальных параметров означает их отсутствие (эквивалентен указанию void)
- недопустимо описание параметров с типом int по умолчанию
- выполнение функции, возвращающей значение, обязательно должно завершаться оператором return с выражением

# Некоторые отличия от С (перенос кода)

Заголовочные файлы (C++11)

- Без “.h”
- `math.h` → `cmath` и т.п.

Стандартная библиотека C++:

- Стандартная библиотека С
- STL – стандартная библиотека шаблонов (контейнеры, алгоритмы и др.)
- Функции и классы (строки, потоки, исключения и др.)

Стандартные потоки ввода-вывода `cin` и `cout`

# Новые средства и возможности в C++

Пространства имен

Типы данных

- встроенный тип `bool`, ключевые слова `true` и `false`
- ссылки

Описание переменных

- в любом месте блока
- в секции инициализаторов оператора `for`, в условии оператора `if`
- инициализация неконстантным выражением

Перегрузка имен

- перегрузка функций и методов классов
- перегрузка операций

Значения параметров по умолчанию

Встраиваемые функции и методы

Управление динамической памятью

Исключения

**+ Классы**

# Пространства имен

## Описание

```
namespace <имя_пространства>
{
    <описания>
}
```

## Использование

```
<имя_пространства>::<идентификатор>
```

```
::<идентификатор>
```

## Включение

```
using namespace <имя_пространства>;
```

Пространства имен позволяют упростить управление внешними идентификаторами, разрешать конфликты

- Одно пространство можно открывать многократно, в т.ч. в разных файлах
- Пространства имен могут быть вложенными

Доступ к идентификаторам в пространстве имен – через операцию видимости (доступа) – ::

Доступ к идентификаторам в глобальном пространстве (неименованном)

Пространство можно включить в любую область (глобальную, локальную), в т.ч. в др. пространство, тогда использовать :: в этой области не требуется

# Ссылки

Ссылка как псевдоним объекта

Тип ссылки:

<тип> &

- Как локальные переменные
- Как формальные параметры
- Как возвращаемые значения

# Перегрузка имен

Использование одного имени (идентификатора) для нескольких программных объектов

- Перегрузка функций – использование одного имени для нескольких функций, отличающихся списком формальных параметров
- Перегрузка методов – аналогично (в пределах класса)
- Перегрузка операций – реализация операций C++ для операндов новых (не встроенных) типов данных (классов)

Сигнатура – имя функции (метода, операции) и список ее формальных параметров (количество и типы), однозначно определяющих функцию

При вызове функции (метода, операции) осуществляется поиск:

1. С тем же именем
2. С тем же количеством параметров
3. С совпадающими или приводимыми типами параметров (наиболее подходящая)

Возможны ошибки, если не подходит ни одной или подходит более одной

# Перегрузка функций

## Описание функций

```
void out(); // 1
void out(int x); // 2
void out(double x); // 3
void out(const char *s); // 4
void out(struct X &r); // 5
```

## Вызовы функций

```
out(12.34); // a
out(99); // b
out(5*5); // c
out("Hello"); // d
out("Hello" + 3); // e
out( ); // f
out(7, 2.33); // g
out(0); // h
```

# Значения параметров по умолчанию

## Описание

```
<Т.В.З.> <ИМЯ_Ф>(<ТИП> <ИМЯ> = <К.В.>)  
{  
    <реализация>  
}
```

## Использование

```
<ИМЯ_Ф>(<выр.>)    // явное значение  
<ИМЯ_Ф>()           // по умолчанию
```

Значение по умолчанию присваивается формальному параметру, если при вызове не указан соответствующий фактический параметр

Значение по умолчанию может быть задано для нескольких параметров подряд, начиная с последнего

При отбрасывании любого количества параметров сигнатура функции не должна совпадать ни с одной из перегруженных (одноименных) функций

При вызове может быть пропущено несколько фактических параметров подряд, начиная с последнего

# Встраиваемые функции

## Описание

**inline** <описание функции>

Код функции встраивается вместо вызова этой функции

Описание функции – реализация или прототип (тогда реализация должна быть ниже в той же единице компиляции)

# Управление динамической памятью

**new** <ТИП>

**new** <ТИП>[ <ВЫР.>]

Создание одиночного объекта и массива объектов (выражение определяет количество элементов массива)  
Возвращает указатель на объект (массив) типа <ТИП\*>

**delete** <указатель>;

**delete**[] <указатель>;

Уничтожение объекта и массива объектов, ранее созданных при помощи **new**.  
Указатель (его значение) должен быть тот же, который возвратила операция **new**.