Системное программирование на C++

Иван Ганкевич

2018

Это курс о программировании

- на языке C++ (2011 и выше),
- под ОС на основе ядра Linux,
- с использованием современных инструментов разработки,
- с большим количеством примеров из реальных программ.

Цели курса

Научиться

- программированию на C++11,
- отображать объекты из предметной области на абстракции языка C++,
- разработке в среде Linux,
- работать с системами сборки, диагностики ошибок, отладки и профилирования кода.

Структура курса

- Лекции + практики.
- Вопросы, предложения, домашние задания: i.gankevich@spbu.ru.

Раздел 1

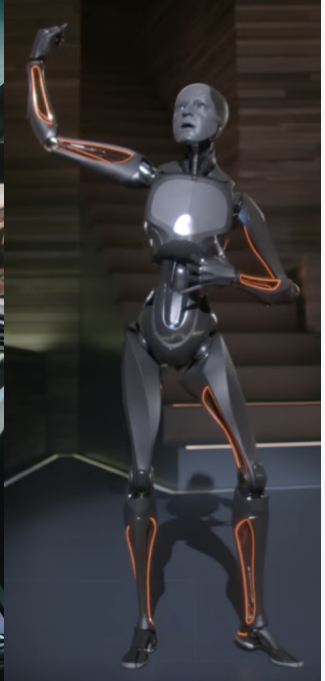
C++

Что такое C++?

Язык для создания и использования легковесных абстракций, применяемый, в основном, для разработки инфраструктуры и приложений, функционирующих в условиях ограниченности ресурсов.

Бьерн Страуструп





С современными инструментами дешевле проверить правильность работы C++, чем C.

Рассел Найт, НАСА

Новые возможности языка

C++11

- безымянные функции
- выведение типов
- правые (rvalue) ссылки
- константные выражения
- вариативные шаблоны
- списки инициализации
- литералы
- атрибуты
- `thread_local`
- ...

STL

- потоки, мьютексы, семафоры, атомарные операции
- `std::future`, `std::promise`
- регулярные выражения
- ГПСЧ
- моменты и интервалы времени
- хэш-таблицы
- умные указатели
- ...

Универсальная инициализация (1)

```
// c++03
std::vector<int> x;
x.push_back(1);
x.push_back(2);
x.push_back(3);

// c++11
std::vector<int> x{1,2,3};
std::vector<int> x = {1,2,3};
x = {1,2,3};
```

Универсальная инициализация (2)

```
void f(std::vector<int>);

// c++03
std::vector<int> x;
x.push_back(1);
x.push_back(2);
x.push_back(3);
f(x);

// c++11
f({1,2,3});
```

Универсальная инициализация (3)

```
template <class T> class vector {
public:
    vector(initializer_list<T>);
};

// без std::initializer_list
vector<int> x(10); // 10 элементов
vector<int> x{10}; // 10 элементов

// с std::initializer_list
vector<int> x(10); // 10 элементов
vector<int> x{10}; // 1 элемент
```

Универсальная инициализация (4)

```
vector<int> x{1,2,3};    // ок
vector<int> x{1.0,2,3};  // ошибка: усечение
```

Выведение типов (1)

```
std::vector<int> x{1,2,3,4,5,6,10};

// c++03
std::vector<int>::iterator result =
    std::find(x.begin(), x.end(), 7);

// c++11
auto result = std::find(x.begin(), x.end(), 7);
```

Выведение типов (2)

```
// выведение типа ссылки
const auto& vertices = ship.hull().vertices();
for (int i=0; i<vertices.size(); ++i) {
    std::cout << vertices[i] << '\n';
}
```

Поэлементный цикл (1)

```
std::vector<int> x{1,2,3,4,5,6,10};

// c++03
typedef std::vector<int>::size_type size_type;
size_type n = x.size();
for (size_type i=0; i<n; ++i) {
    std::cout << x[i] << '\n';
}

// c++11
for (int num : x) { std::cout << num << '\n'; }
```

Поэлементный цикл (2)

```
// привести символы к нижнему регистру
std::string s{"ABC"};
for (auto& ch : s) {
    ch = std::tolower(ch);
}

// вызвать функцию для каждого элемента
for (auto i : {1,2,3}) {
    func(i);
}
```

Лямбда-функции (1)

```
std::vector<int> x{1,2,3,4,5,6,10};

// поиск первого элемента > 5
auto result = std::find_if(x.begin(), x.end(),
    [] (int i) {return i > 5;}
);
```

Лямбда-функции (2)

```
// [=] передача по значению
// [&] передача по ссылке

int cnt = 0;
std::generate(x.begin(), x.end(),
    [&cnt] () {return ++cnt;}
);
```

Лямбда-функции (3)

```
// объект-функция
int cnt = 0;
std::function<int()> gen = [&cnt] () {
    return ++cnt;
};
// или auto gen = ...
std::generate(x.begin(), x.end(), gen);
```

Лямбда-функции (4)

```
struct wave {
    std::string name;
    std::function<float(float)> func;
};
std::vector<wave> genwaves(float a) {
    return {
        {
            "cosine_wave",
            [=] (float x) {return a*std::cos(x);}
        }
    };
}
```

Функции по-умолчанию

```
// запретить копирование
class X {
private:
    std::ofstream out;
public:
    X() = default;
    ~X() = default;
    X(const X&) = delete;
    X& operator=(const X&) = delete;
};
```

Строго типизированные перечисления

```
enum class Colorspace: int {RGB, YUV, CMYK};
void f(Colorspace cs, void* data);
f(Colorspace::RGB, ...);
```

Раздел 2

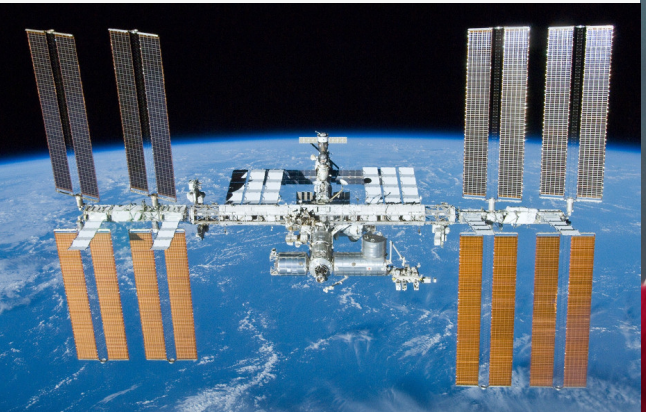
Linux

Как и наука, открытый исходный код позволяет людям строить на прочной основе текущих знаний, не делая из них тайну.

Линус Торвальдс

TOP 500

The List.



```
1.9 Ext4 file system parameters
------------------------------
```

"That letter [the last s] is sad because all the others have those things [=] below them and it does not."

This patch fixes the tragedy so all the letters can be happy again.

Linux для программиста C++

- Компиляторы: GCC и Clang (C++ 2011, 2014, 2017).
- Статический анализ кода: `clang-tidy`, `cppcheck`.
- Проверка корректности программы: `valgrind`, `-fsanitize=...`
- Покрытие кода тестами: `lcov`.
- Профилирование: `perf`.
- Сборка кода: `cmake`, `meson`, `ninja`.
- Автоматизация скриптами.

Выводы

- C++ стал более кратким.
- C++ стал использоваться в критически важных программах вместо C.
- Linux расширяет возможности среды разработки.

Ссылки

- C++11 FAQ (домашняя страница Страуструпа).