C++

Лекция 1

Intro

Организационные вопросы

Активности

- Лекции (по вт, Егор)
- Семинары (по ср, Филипп)
  - Дорешки
  - Контрольные работы
- Домашние работы
- Экзамен
  - теормин
  - теория
  - задача

Популярность языка

- TIOBE September 2024: 2nd (link)
- PYPL September 2024: 5th (link)
- Octoverse: The state of open source
- StackOverflow Developer Survey 2022

Известность языка

- высокая эффективность и производительность
- уязвимости и UB

Сферы применения

- Системное ПО, операционные системы
- Базы данных
- Браузеры
- Компьютерные игры
- Машинное обучение …

История

C++ начал разрабатываться с начала 1980-х сотрудником Bell Labs Бьярне Страуструпом

"С с классами" и транслятор cfront

История

К 1983 было реализовано большое количество возможностей, поэтому язык был переименован в C++

Имя языка связано с оператором постфиксного инкремента

Стандартизация

Classic: C++98,03

Modern: C++11,14,17,20,23,…

- Поддержка стандартов компиляторами
- Драфт стандарта

Характеристики языка

- Высокая сложность изучения
- Свобода выбора стиля
- Эффективность
- Компилируемость
- Высокоуровневость и низкоуровневость
- Интероперабельность с языком C
- Кроссплатформенность

Высокая сложность изучения

- Огромный текст стандарта
- Сложный синтаксис
- Требует понимания системы, в которой запускается программа
- Знание идиом и "рецептов"

Свобода выбора стиля

- Выбор парадигмы программирования
- Выбор уровней абстракции

Эффективность

- Zero-overhead principle
- Возможность максимально оптимизировать участок кода

Компилируемость

Компиляция - преобразование текста программы в машинный код

- для каждой платформы отдельно
- позволяет отловить некоторые ошибки
- нет накладных расходов при выполнении программы
- при изменении программы нужно компилировать снова

Этапы компиляции

1. Preprocessor: Source Code Files → Translation Units
2. Compilation: Translation Unit → Object Files
3. Linker: (Object Files, Libraries) → Executable | Library

Высокоуровневость и низкоуровневость

Низкоуровневые особенности: - Работа с памятью - Использование платформенно-специфичных функций - Прямой доступ к ресурсам

Интероперабельность с языком C

- C++ отделился от C еще до стандартизации
- C не является подмножеством C++
- Language Linkage

Кроссплатформенность

- Код пришется один раз, компилируется на всех платформах
- Есть возможность написать непортируемый код

Программа

Программа - последовательность инструкций

Точка входа:

```cpp
// main.cpp
int main() {
  return 0;
}
```

Основные конструкции

```cpp
// main.cpp
#include <iostream>

bool isGood(int n) { return n == 42; }

int main() {
    int n = 32;
    for (int i = 0; i < n; ++i) {
        if (isGood(i)) {
            std::cout << i << " is good!" << std::endl;
        }
    }
    return 0;
}
```

Полезные сайты

- https://en.cppreference.com/
- https://godbolt.org/
- https://cppinsights.io/

Работа с файлами

```cpp
// main.cpp
#include <fstream>

int main() {
    std::ifstream in ("in.txt");
    std::ofstream out("out.txt");

    double value = 0.0;
    in >> value;
    out << value;

    return 0;
}
```

Многофайловая программа

```cpp
// main.cpp
#include <iostream>
#include "factorial.hpp"

int main() {
    std::cout << factorial(10);
    return 0;
}
```

Заголовочный файл

```cpp
// factorial.hpp

int factorial(int n) {
    // your code here...
}
```

```cpp
// main.cpp
#include <iostream>
#include "factorial.hpp"

int main() {
    std::cout << factorial(10);
    return 0;
}
```

Проблема 1

```cpp
// main.cpp
#include <iostream>
#include "factorial.hpp"
#include "factorial.hpp" // двойное включение

int main() {
    std::cout << factorial(10);
    return 0;
}
```

Решение проблемы 1

```
// factorial.hpp
#pragma once

int factorial(int n) {
   // your code here...
}
```

Проблема 2

Изменение фукнции `factorial` приводит к перекомпиляции `main.cpp`

Решение проблемы 2

```hpp
// factorial.hpp
#pragma once

int factorial(int n);
```

```cpp
// factorial.cpp
#include "factorial.hpp"

int factorial(int n) {
    // your code here...
}
```