Программирование на языке C++

Шаблов Анатолий
anatoliishablov@gmail.com

ИТМО, весенний семестр 2024

ITMOre than a UNIVERSITY

Вступление

О чем курс?

- Основные элементы языка C++
- Некоторые инструменты для разработки программ на C++
- Базовые навыки программирования

Почему C++?

- Язык сочетает черты низкоуровневого и высокоуровневого
- Позволяет как использовать сложные абстракции, так и прибегать к низкоуровневым оптимизациям и ручному управлению ресурсами
- Zero overhead abstractions
- Значительно более высокоуровневый, чем C, но в то же время может быть настолько же эффективным
- Один из самых распространенных прикладных языков

Где используется C++?

- Графические оболочки (MS Windows UI, Aqua, KDE)
- Офисные пакеты (MS Office, OpenOffice)
- Графические редакторы и среды 3D моделирования (Photoshop, Maya)
- Компьютерные игры (CryEngine, Frostbite, Gamebryo, id Tech 4-, Source, Unreal Engine) I CAD (Autodesk, Catia, FreeCAD)
- Браузеры и Javascript движки (Chrome, Firefox, V8, SpiderMonkey)
- Базы данных (MongoDB, частично MariaDB, MS SQL, Oracle, SAP DB, ScyllaDB)
- Системы информационного поиска, интернет поисковики (Google, Яндекс)
- Финтех (Bloomberg, Morgan Stanley, Itiviti)

Недостатки других языков

Интероперабельность

- JVM — JNI
- Python — Boost.Python
- Другие языки FFI (интерфейс на C)

Недостатки C++

- Катастрофически сложен
- Иногда слишком примитивен
- Иногда недостаточно современен
- Эволюция путём добавления элементов и механизмов

Сложности грамматики

```
// variables 'x', 'y', 'z'
int(x), y, *const z;
int* a, b;

// expression '(int(x)), (y), (new int))'
int(x), y, new int;
```

Тонкости и нюансы

```cpp
int a;
decltype(a) b;   // type of b == int
decltype((a)) c; // type of c == int&

int n = sizeof(0)["abcdef"];
// n == ??
```

Эволюционные сложности

```
static vs static vs static
struct vs class vs typename
const vs constexpr vs constinit vs consteval
```

Сложности правил языка

- 9 различных видов инициализации переменных
- 21 правило упорядочивания исполнения
- 13 правил выбора лучшего кандидата при перегрузке функций
- И ещё больше веселья в шаблонах

Формат курса

- Лекции
- Небольшие примеры-иллюстрации к лекциям
- Задачи по мотивам примеров
- Большие задачи
- Соревнование по скорости для одной из больших задач
- Сдача задач через code review на github.com
- Коллоквиум в середине семестра
- Экзамен

Литература

- Bjarne Stroustrup: Programming: Principles and Practice Using C++, 2014
  Программирование Принципы и практика использования C++
- Bjarne Stroustrup: The C++ Programming Language (4th edition), 2013
- Bjarne Stroustrup: The Design and Evolution of C++, 1994
  Дизайн и эволюция C++
- Stanley Lippman: C++ Primer (5th Edition), 2012
  Язык программирования C++. Базовый курс
- Herb Sutter: Exceptional C++, 1999; More Exceptional C++, 2001
  Решение сложных задач на C++
- и другие: Meyers, Josuttis, Alexandrescu, Vandevoorde

История

Классификация языков программирования

- Компилируемые / интерпретируемые
- Императивные / декларативные
- Поддержка различных парадигм: ООП, функциональные, логические
- Статическая типизация / динамическая типизация
- Сильная типизация / слабая типизация
- Энергичные / ленивые

Цели создания

- ООП
- пользовательские абстракции
- сильная типизация
- эффективность
- отсутствие “необоснованной стоимости” возможностей
- простота реализации (использование уже существующих инструментов)
- отсутствие излишних ограничений на стиль программирования

Краткая история

- 1979 - C with classes (расширение языка C классами, наследованием, более сильной типизацией, встраиваемыми функциями)
- 1983 - C++ (перегрузка функций и операторов, виртуальные функции, ссылки, типобезопасное управление памятью)
- 1985 - The C++ Programming Language (первое описание языка)
- 1989 - C++ 2.0 (множественное наследование, абстрактные классы, статические члены классов)
- 1990 - The Annotated C++ Reference Manual (шаблоны, исключения, пространства имен)
- 1992 - STL (обобщенная реализация различных структур данных и типовых алгоритмов)

Краткая история

- 1998 - C++98, первый ISO стандарт языка
- 1999 - Boost
- 2003 - C++03, второй ISO стандарт, незначительные изменения
- 2011 - C++11, новый стандарт, большие изменения и модернизация
- 2013 - 4-е издание The C++ Programming Language
- 2014 - C++14, дальнейшее развитие нового стандарта
- 2017 - C++17, текущий устоявшийся стандарт языка
- 2020 - C++20, текущий опубликованный стандарт языка
- 2023 - C++23, стандарт языка находящийся на последних стадиях принятия

Абстрактная вычислительная машина

Нижний уровень языка

- Целые числа и операции над ними в рамках двоичного представления
- Дробные числа имеют определенную точность и диапазон
- Типы данных вместо битов и байтов в памяти
- Линейная непрерывная память
- По умолчанию – последовательное исполнение (в рамках observable behaviour)
- Начиная с C++11 модель памяти учитывает параллельное исполнение
- Управление параллельным исполнением в стандартной библиотеке
- Исключения заданы высокоуровневым поведением

Ниже абстракций

- Особенности конкретной архитектуры, например, big-ending vs little-endian; битность процессора
- Целые и дробные числа представляются по-разному
- Много уровней памяти: регистры, кеш нескольких уровней, RAM
- Реальный и защищенный режим
- Виртуальная память
- Прерывания, системные вызовы, переключение контекста
- Параллелизм: внутри ядра процессора, между несколькими ядрами, между процессорами; разные гарантии синхронизации на разных архитектурах
- Инструкции различной сложности: RISC, CISC, векторные инструкции, сопроцессоры (“математический”, GPU)

Гарантии языка и undefined behaviour

Некоторые вещи язык гарантирует - вне зависимости от платформы, компилятора и иных внешних факторов.
Например, `sizeof(int) <= sizeof(long)`.

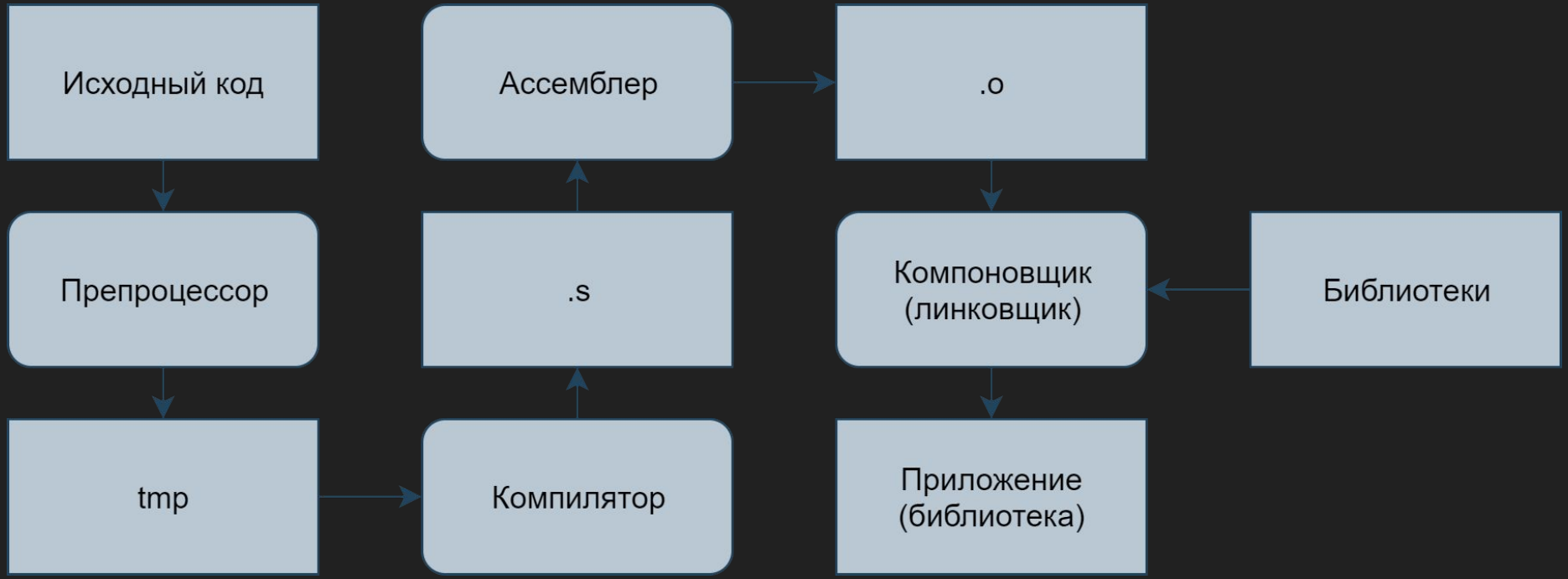
- **Observable behaviour**: компилятор может менять программу, если её внешнее поведение не меняется.
- **Implementation defined behaviour**: поведение программы может различаться в зависимости от реализации компилятора (но это должно быть задокументировано).
- **Unspecified behaviour**: поведение зависит от реализации, но это не требуется документировать; каждый возможный вариант поведение должен быть корректным.
- **Undefined behaviour**: стандарт не накладывает никаких ограничений на поведение в этом случае.

Трансляция программ

Структура программ

- Набор текстовых файлов
- Соглашение: заголовочные файлы и файлы кода
- Соглашение: расширения .h (.hpp .hxx) и .cpp (.cxx). Возможны дополнительные файлы, например, .ipp
- Набор определений
- Одно определение функции main

Этапы трансляции



Трансляция C++ на примере gcc

Трансляция программы из одного файла:

```
g++ -std=c++17 -o prog prog.cpp
```

Результат препроцессора:

```
g++ -std=c++17 -E prog.cpp
```

Ассемблерный код:

```
g++ -std=c++17 -S prog.cpp
```

Объектный файл:

```
g++ -std=c++17 -c prog.cpp
```

Дизассемблирование:

```
objdump -dS prog
```

Объектные файлы

Объектные файлы обычно состоят из различных секций. Например: заголовок, секция кода, секция данных, отладочная информация. Сущности ссылаются по именам, адреса в памяти не назначены.

Mangling: имена сущностей из текста программы не всегда могут быть перенесены в имена в объектном файле. Для C обычно соответствие точное (хотя некоторые реализации добавляют к имени дополнительную информацию). В C++ структура имен более сложная и они приводятся к уникальным строковым именам по определенному алгоритму (зависит от реализации).
Например, имя Space::Outer::Inner::code может быть преобразовано в _ZN5Space5Outer5Inner4codeE

Онлайн компиляторы

- Godbolt https://godbolt.org/
- CPP Insights https://cppinsights.io/
- Quick Bench https://quick-bench.com/
- Coliru https://coliru.stacked-crooked.com/
- Wandbox https://wandbox.org/

Организационные вопросы

План лекций

1. Вводная
2. Основы языка
3. Препроцессор, единицы трансляции, макросы
4. Сложные типы данных
5. Специальные методы классов
6. Работа с памятью
7. Предварительный обзор поздних тем
8. Коллекции и итераторы
9. Коллоквиум
10. Пространства имён
11. Перегрузка функций
12. ООП
13. Шаблоны
14. Исключения и безопасность

Практические задания

- Маленькие задачи – 4 варианта по мотивам примера из лекции

- Большие задачи
  - 4 набора по 3 задачи
  - все задачи стоят по-разному
  - наборы в целом сбалансированы

Дедлайны

- Маленькие задачи: март – май
- Большие задачи: апрель – начало июня
- Финальный дедлайн – экзамен, в первой половине сессии

Работа над задачей

- 1-й дедлайн – дедлайн оформления
- 2-й дедлайн – дедлайн приёмки
- 2 недели между двумя дедлайнами
- code review – несколько итераций
- работа строго индивидуальная
- по одной из больших задач возможно собеседование
- 14 “поздних” дней

Соревнование по скорости

- Параллельно с процессом ревью
- 2 прогона, можно внести изменения после 1-го
- места распределяются по перцентилям
- множитель оценки 1 . . . 3

Оценивание задач

- Базовая стоимость маленьких задач – 5-12 баллов
- Базовая стоимость больших задач – 15-40 баллов
- Суммарная базовая стоимость 3 больших задач – 80 баллов
- Штрафные баллы: число ревью, число существенных замечаний, число “пушей” на Github
- Соревнование по скорости – коэффициент масштабирования оценки за задачу

Теоретическая аттестация

- Коллоквиум в середине семестра

- Экзамен в конце
  - блиц
  - основная часть

Итоговая аттестация

- Финальная оценка = оценка за экзамен
- Блиц
  - провал = **F**
  - успех = **E** или переход к основной части экзамена
- Основная часть экзамена = оценка от **E** до **A**
- Успешно сданный коллоквиум заменяет блиц
- Допуск к экзамену – порог баллов за задачи
  - 70 баллов для блиц части
  - 90 баллов для основной части
- Автомат
  - 125 баллов = **B**
  - 150 баллов = **A**

Иные источники баллов

- улучшения тестов к задачам

Сложности курса