Программирование на языке C++

Вводный курс

Александр Морозов
gelu.speculum@gmail.com

ИТМО, весенний семестр 2022

Содержание

О чём этот курс?

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

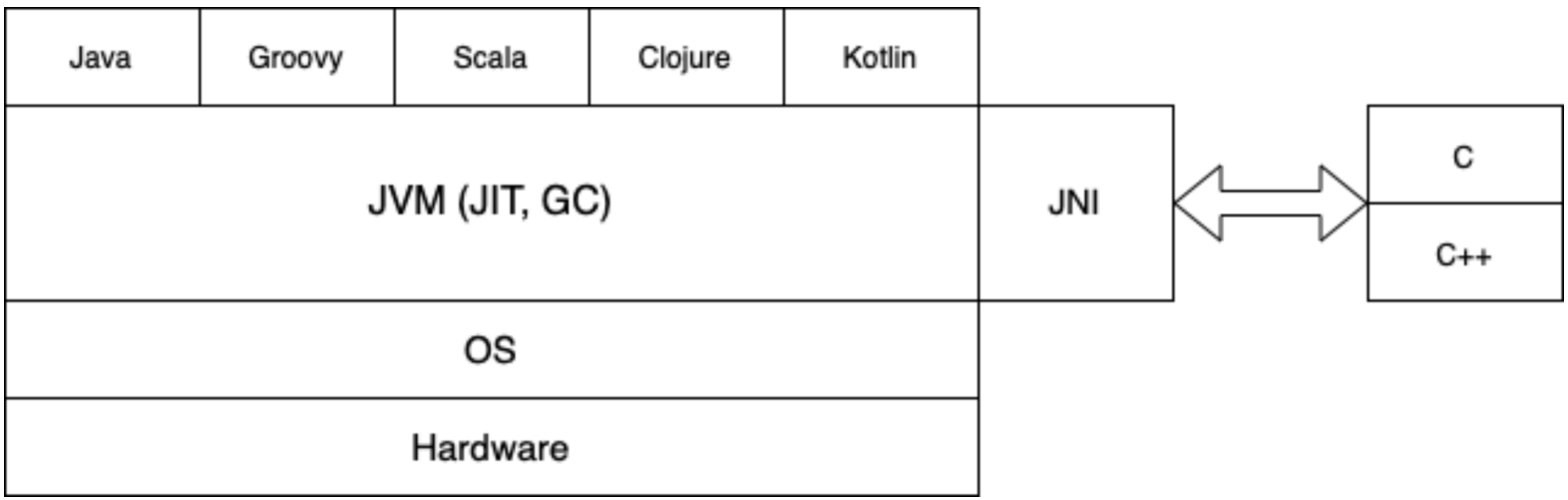
Почему C++?

- 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

Место C++ в современном мире

- 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)
- 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)

Что не так с Java?



Что не так с другими компилируемыми языками?

Интероперабельность C++ с другими языками

- с JVM – так себе, JNI
- с .NET – Managed/Unmanaged C++, есть подводные камни
- с различными интерпретируемыми – ?
- с различными компилируемыми – FFI (обычно, C интерфейс)

Что не так с C++?

- катастрофически сложен
- иногда слишком примитивен
- иногда недостаточно современен
- эволюция путём добавления элементов и механизмов

Сложности грамматики C++

```
1       // variables 'x', 'y', 'z'
2       int(x), y, *const z;
3
4       // expression '(int(x)), (y), (new int))'
5       int(x), y, new int;
6
7       // ??
8       B b(A());
```

Тонкости и нюансы в C++ важны

```
1       int a; // type of 'a' = int
2       decltype(a) b; // type of 'b' = int
3       decltype((a)) c; // type of 'c' = int &
```

В C тоже были весёлые моменты

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

Эволюционные сложности C++

```
1       static vs static vs static
2
3       struct vs class vs typename
```

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

Ещё больше веселья с инициализацией в C++20

```
1       struct A
2       {
3         int && r;
4       };
5
6       A a1{7}; // lifetime is extended
7       A a2(7); // lifetime is NOT extended
```

Формат курса

- Лекции
- Небольшие примеры-иллюстрации к лекциям
- Задачи по мотивам примеров
- Большие задачи
- Соревнование по скорости для 2-й большой задачи
- Сдача задач через code review на github.com
- Коллоквиум в середине семестра
- Экзамен

Некоторая 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

Содержание

Предшественники C++

- Simula: объектно-ориентированный язык, 1965
- C: эффективный процедурный язык, 1972

Появление C++: цели создателя

Бьярне Страуструп занимался моделированием распределенных аспектов операционных систем и ему нужен был язык:

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

Краткая история C++

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).
1998 - C++98, первый ISO стандарт языка.

Краткая история C++, продолжение

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.

Классификация языков программирования

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

Содержание

Нижний уровень языка

- целые числа и операции над ними в рамках двоичного представления
- дробные числа имеют определённую 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 этих 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

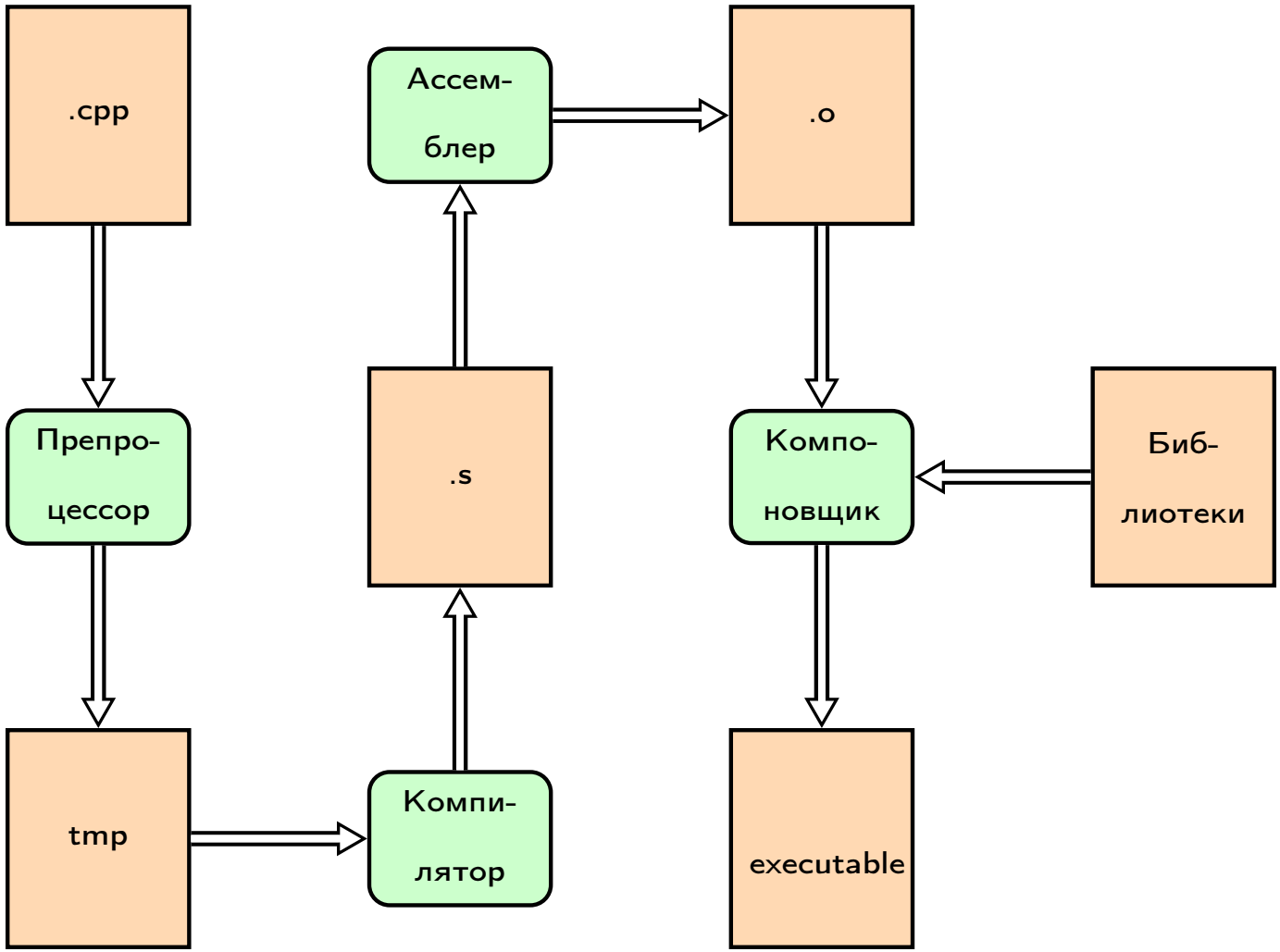
- Некоторые вещи язык гарантирует - вне зависимости от платформы, компилятора и иных внешних факторов. Например, `sizeof(int)<= sizeof(long)`.
- *Observable behaviour*: компилятор может менять программу, если её внешнее поведение не меняется.
- *Implementation defined behaviour*: поведение программы 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.

Содержание

Структура программ на языке C++

- набор текстовых файлов
- соглашение: заголовочные файлы и файлы кода
- соглашение: расширения **.h** и **.cpp**
- набор определений
- одно определение функции main

3 этапа трансляции C++



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`.

Online компиляторы

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

Содержание

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 $1 \ldots 3$

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 **F** 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
- оформление конспекта

Сложности курса

- язык сложный
- материала много
- студенты недооценивают сложность
- студенты переоценивают свои силы
- язык не похож на Java