# Лекция 15

# C++

- C++ начал развиваться как расширение C
  - Cfront – препроцессор C++, генерировал C-код
  - Обеспечить совместимость снизу-вверх на уровне исходного кода
  - Использовать существующую инфраструктуру разработки на C
    - Заголовочные файлы, обрабатываемые препроцессором
    - Компоновщик, создающий исполняемые файлы из объектных
- Другой такой язык: Objective C

# C++ ABI

- Некоторые возможности C++ обрабатываются на этапе компиляции:
  - Параметры по умолчанию
  - Lambda
  - Template
- Наиболее сложная поддержка при выполнении требуется для exception handling

# Перегрузка функций

- В C имя функции уникально для программы
  - “func” в программе на C → “func” в объектном файле
- В C++ уникальны: пространство имен+классы+имя функции+параметры
  - void spc1::func(int);
  - void spc2::func(double);
- Декорирование имен (name mangling)
  - void spc1::func(int); → _ZN4spc14funcEi
  - void spc2::func(double); → _ZN4spc24funcEd

# Декорирование имен

- Взаимно-однозначное отображение уникальной совокупности атрибутов в идентификатор
- При компоновке вместо сравнения совокупности атрибутов сравнивается только идентификатор
- Удачная идея, используется повсеместно в разных языках
- Стандарта на декорирование имен в C++ нет: Linux, MacOS - “Itanium ABI”; Windows – Microsoft ABI

# Интерфейс C++ к C

- Чтобы отключить декорирование используется специальная форма extern
  extern “C” void func(int x);
- Такие функции не могут перегружаться, находиться в пространствах имен и т. П.
- Подключение стандартной библиотеки C:
  extern “C” {
  #include <unistd.h>
  }
- В стандартных заголовочных файлах это уже есть!

# Классы C++

- POD-типы (plain old data) – для них гарантируется раскладка по памяти, совместимая с C
  - http://en.cppreference.com/w/cpp/concept/PODType
  - Нет виртуальных функций и виртуального наследования
  - Нет пользовательского деструктора и конструктора копирования
  - Нет полей-ссылок

# Указатель this

- Передается как неявный первый параметр всех методов
- В Itanium ABI (Linux, MacOS...) – по стандартным соглашениям о вызовах
- В MS Visual Studio – THISCALL calling convention, this передается в %ecx

# Одиночное наследование

- Экземпляр базового класса находится в начале производного класса:
  struct Base { }; …
  struct Derived { struct Base b; … } d;
- Адрес экземпляра базового класса совпадает с адресом экземпляра производного
- Преобразование Derived* → Base* не меняет значение указателя

# Виртуальные вызовы

- Выполняются через таблицу виртуальных функций (VTABLE)
- Экземпляр таблицы – один на все экземпляры класса
- Указатель на таблицу находится в начале области памяти, выделенной под объект
- VTABLE[-1] – указатель на объект type_info для данного класса

# Таблица виртуальных функций

- Виртуальные функции вызываются по индексам