Лекция 1: Начало программирования на языке Си

Сергей Мыц

кафедра Информатики, БГУИР

предмет “Программирование”, ИиТП, первый курс,
весенний семестр 2015

Содержание лекции

Содержание

Откуда брать теорию

- Слушать лекции.
- Выполнять практические задания.
- Читать рекомендованные книги - это неотъемлемая часть курса.

Задача лекций и практических

- Материала много - времени на занятиях мало
- Лекции и практические задания дают посмотреть и пощупать часть основных вещей.
- Они задают направление, искать и изучать информацию по ключевым словам надо самостоятельно.

Применение изученного

- Лабораторные дают возможность применить изученное, которой надо пользоваться.
- Кроме лабораторных надо самостоятельно экспериментировать и писать программы.
- Пробовать разные штуки из теории.

Постоянная практика

- Программированию нельзя научиться исключительно в теории.
- Надо постоянно писать код, и это надо делать часто и много.
- Применять новое, ошибаться, анализировать свои ошибки и снова за дело.

Содержание

Определение

Компьютер - устройство или система, способная производить вычисления и выполнять действия по принятию логических решений.

Производительность

- И делает он это чрезвычайно быстро - квадриллионы операций в секунду.
- 10^{15}
- 1 000 000 000 000 000
- И это не предел.

Производительность

- И делает он это чрезвычайно быстро - квадриллионы операций в секунду.
- 10^{15}
- 1 000 000 000 000 000
- И это не предел.

Производительность

- И делает он это чрезвычайно быстро - квадриллионы операций в секунду.
- 10^{15}
- 1 000 000 000 000 000
- И это не предел.

Производительность

- И делает он это чрезвычайно быстро - квадриллионы операций в секунду.
- 10^{15}
- 1 000 000 000 000 000
- И это не предел.

Данные, программы и программиста

- Компьютеры обрабатывают данные.
- Используя наборы инструкций, называемые программами.
- Эти действия определяются программистами.

Структура

- Аппаратная часть (hardware, железо) - различные устройства, составляющие компьютерную систему.
- Программное обеспечение (software, софт) - программы, выполняющиеся на компьютере.

Части компьютера

- Процессор
- Память
- Дисковая система
- Блоки ввода-вывода
- Видеоподсистема
- Периферийные устройства

Варианты моделей архитектур

- Гарвардская архитектура
- Архитектура фон Неймана

Гарвардская архитектура

- Хранение инструкций и данных разделено
- Каналы инструкций и данных разделены

Принципы архитектуры Фон Неймана

- Однородность памяти
- Адресность
- Программное управление
- Двоичные код

Архитектура Фон Неймана: однородность памяти

- Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы.
- Значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему.
- А это значит, что программы могут быть получены как результат исполнения другой программы

Архитектура Фон Неймана: адресность

- Структурно основная память состоит из пронумерованных ячеек.
- Процессору в произвольный момент доступна любая ячейка.
- Двоичные коды команд и данных разделяются на единицы информации, называемые словами.
- Они хранятся в ячейках памяти.
- Доступ по номерах ячеек – адресам.

Архитектура Фон Неймана: программное управление

- Все вычисления есть последовательности управляющих слов – команд.
- Каждая команда – некоторая операция из набора операций, реализуемых вычислительной машиной
- Команды программы хранятся в последовательных ячейках памяти вычислительной машины и выполняются последовательно.
- При необходимости, с помощью специальных команд, эта последовательность может быть изменена.
- Решение об изменении порядка выполнения команд программы принимается на основании анализа результатов предшествующих вычислений.

Архитектура Фон Неймана: двоичный код

- Вся информация (данные и команды) кодируется двоичными цифрами 0 и 1.
- Каждый тип информации представляется двоичной последовательностью и имеет свой формат.

Содержание

Основные этапы подготовки и запуска

- Редактирование - редактор
- Препроцессорная обработка - препроцессор
- Компиляция - компилятор
- Компоновка (линковка) - компоновщик
- Загрузка - загрузчик
- Исполнение - процессор

Язык ассемблера

Язык ассемблера (assembly language, assembler) – машинно-ориентированный язык низкого уровня с командами, обычно соответствующими командам машины, который может обеспечить дополнительные возможности вроде макрокоманд.

Си как надстройка над ассемблером

- Объединяет идеи ассемблера в более высокоуровневые конструкции.
- Добавляет свои новые концепции.
- Делает более удобным работу с программой и её разработку.
- Более читабельный код.
- Больший контроль над типами данных.
- Идеи структурного программирования.

Си как надстройка над ассемблером

- По сути, все данные и код - всего лишь кусок памяти.
- Типы данных явно существуют только в момент написания и компиляции программы.
- См. идеи фон Неймановской архитектуры.

Содержание

Первая программа на Си

```c
/*
 * Simple C helloworld program.
 */
#include <stdio.h>

// Execution entry point
int main()
{
    // Output line of text to stdout
    printf("Hello, World!\n");

    // return result code, 0 if success
    return 0;
}
```

Комментарии

- // - однострочный
- /* */ - многострочный

Задаваемые идентификаторы

- Переменные и константы
- Функции
- Типы данных

Неизменяемые идентификаторы

Зарезервинованные ключевые слова языка.

Идентификаторы

- могут состоять из букв и цифр
- знак подчёркивания _ считается в этом контексте буквой
- начинаются только с буквы
- регистрозависимы

Именование идентификаторов

- зависит от соглашений о написании кода
- не начинать с _ – может использоваться библиотечными функциями

Именование идентификаторов

- по-умолчанию: переменные_и_функции_пишутся_так
- A_КОНСТАНТЫ_ВОТ_ТАК

Именование идентификаторов

- давать понятные имена
- если маленькая область видимости и контекст - возможно использование более коротких, при широких - более длинных

Этапы начала жизни идентификаторов

- Объявление (declaration)
- Определение (definition) - и отсюда one definition rule
- Инициализация

Переменные

- int number; // объявление переменной типа int
- int number = 0; /* объявление и инициализация переменной */

Простые стандартные типы данных

- значения
- указатели

Больше типов и пользовательские типы

Структуры, перечисления и т.п. будут в дальнейших лекциях

Начальное значение

- По-умолчанию переменная - это всего лишь какой-то кусок памяти.
- Внутри - какое-то произвольное значение.
- Перед использованием надо инициализировать.

Базовые типы

- char - байт, один символ из локального символьного набора.
- int - целое число, размер (количество бит) соответствует разрядности среды (процессор и ОС, распространены 32-битные и 64битные).
- float - вещественное число с плавающей точкой, одинарной точности.
- double - вещественное число с плавающей точкой, двойной точности.

Модификаторы

- short, long - может влиять на размер (количество бит).
- signed, unsigned - для типов-чисел отрицательные и положительные, или только положительные.
- const - значение переменных один раз инициализируется и нельзя менять.

Константы

- переменные с модификатором const
- значения, заданные явно

Константные значения

- int - 100500
- float - 1f или 1.0f
- double - 1.0
- символьная - 'Y'
- строковая - "Hello, World" (совокупность символов и управляющих последовательностей)

Константное выражение

- выражение, содержащее только константы
- могут вычисляться при компиляции, а не при выполнении
- можно использовать везде, где допускается одна константа

Стандартные потоки ввода-вывода

- stdin - стандартный ввод
- stdout - стандартный вывод
- stderr - стандартный вывод для ошибок

Простой пример ввода-вывода

```c
#include <stdio.h>

// Execution entry point
int main()
{
    int number;

    // read int value into memory address of
        variable number from stdin
    scanf("%d", &number);

    // print formatted string using int value from
        variable number
    printf("Your number is %d\n", number);
    return 0;
}
```

Содержание

Операции

- унарные знаковые (+ и -)
- арифметические (+, -, *, /, % и т.д.)
- инкремент и декремент (++, --)
- отношение и логические (<, >, <=, >=, ==, &&, || и т.д.)
- преобразование типов ((double)10)
- порязрядные (побитовые) (&, |, ^, <<, ~ и т.д.)
- присваивание (=)
- присваивание с операцией (+=, *=, |= и т.д.)
- разыменование и взятие адреса (* и &)
- тернарный (?:)

Выражения

- несколько операций
- условные выражения
- приоритет и порядок вычисления выражнеий

Содержание

Операторы и блоки

- Выражение становится оператором, если поставить после него ;.
- Составные операторы - блоки кода - заключаются в {}
- Важно правильное форматирование кода, чтобы было просто читать.

Условные операторы

- if (условие) {код}
- if (условие) {код} else {код}
- if (условие) {код} else if (условие) {код} else {код}
- тернарный оператор - (условие ? выражение : выражение)
- switch

Циклы

- for (выражение; выражение; выражение) {}
- while (выражение) {}
- do {} while (выражение);

break и continue

- break - прерывание выполнение и переход в конец конструкции.
- continue - прерывание отдельной итерации выполнения.

Оператор goto и метки

- Ставим метку в коде label:
- Переходим на метку - goto label;
- Использование сильно нежелательно, разве что при обработке ошибок в отдельных случаях.

Содержание

Керниган и Ритчи, Программирование на C

- Глава 1: Вводный урок (всё до массивов)
- Глава 2: Типы данных, операции и выражения
- Глава 3: Управляющие конструкции

Содержание

Керниган и Пайк, Практика программирования

- Глава 1: Стиль программирования

Содержание

Керниган и Пайк, Unix. Программное окружение

Для знакомства с Linux (семейство Unix) можно почитать первые несколько глав.

Содержание

Эрик Реймонд, Искусство программирования для Unix

Можно почитать для знакомства с базовыми идеями, логикой и подходами.