ОСНОВЫ ПРОГРАММНОГО КОНСТРУИРОВАНИЯ

Лекция № 1
5 сентября 2016

НЕФОРМАЛЬНАЯ ЧАСТЬ

- Я – Владимир Владимирович Парфиненко.
  - Бакалавр физики (ФФ), магистр математики (ММФ).
  - Профессиональный программист (Excelsior).
  - Регулярно чему-то учу (ОПК, ЛШ ФМШ).
- Контакт: vladimir.parfinenko@gmail.com

ФОРМАЛЬНАЯ ЧАСТЬ

О формате и оценках

УСЛОВИЯ РАБОТЫ

- Лекции: алгоритмы, структуры данных, практики программирования.
- Семинары: язык C, применение теоретических знаний на практике.
  - Второй семинарист: Илья Сергеевич Иванов.
- Самостоятельная работа: неотъемлемая часть практики.
- Материалы на сайте opk.afti.ru: база задач и лекции.

ФОРМА ОЦЕНКИ

- Курс ОПК является обязательным.
- Работа в семестре: оценивается двумя семинаристами, как-то коррелирует с баллами на opk.afti.ru.
- Проект: оценивается по множеству критериев на зачете.
- Теория: два вопроса по теории на зачете.
  - Бонус теория: теоретические задачки на лекциях.

СОДЕРЖАТЕЛЬНАЯ ЧАСТЬ

Крайне быстрое установление контекста
для последующей работы

КРАТКО О ПРОГРАММИРОВАНИИ

- Есть тупой кусок кремния (1), годный к повторяющимся, рутинным операциям.
- Есть пользователь (2).
- Программист (3) *отбирает* у пользователя (2) рутину, объясняя куску кремния (1), как ее выполнять.
- В основе труда программиста (3) лежит модель системы.
- Хорошая программа = хорошая модель.

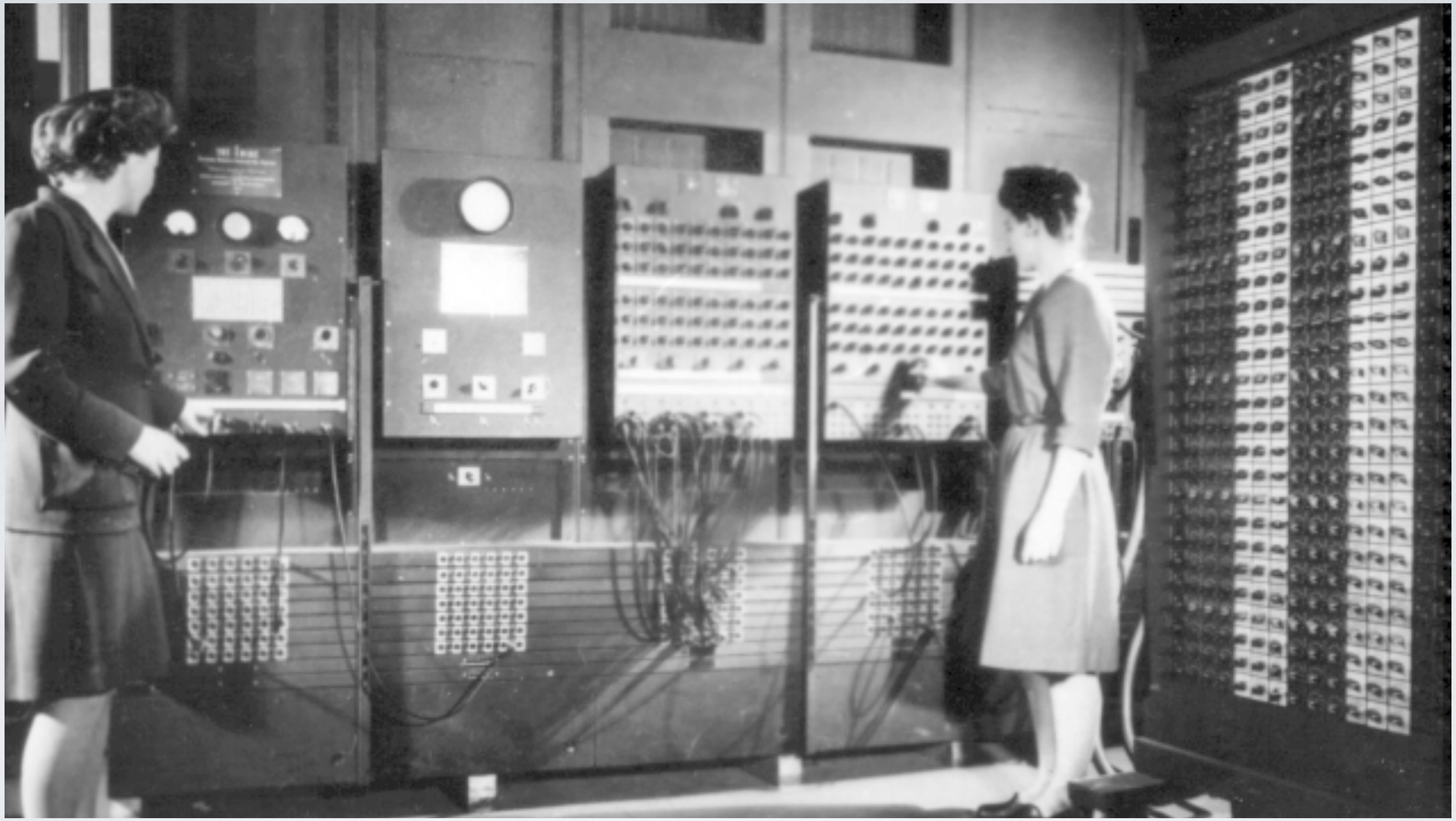
ЧТО ЕСЛИ...

- Программист отобрал у пользователя не ту рутину...
- Программист плохо объяснил задачу тупому куску кремния...
- Программист понял пользователя неправильно...
- Модель у программиста в голове не соответствует действительности…
- Программист решил, что он умный и сам правильно знает, как пользователю жить…
- Пользователю никто не объяснил, как пользоваться программой…

...TO

Пользователь всегда прав!

P. S. Пользователь есть всегда и у любой программы!



ИСТОРИЯ ПРОГРАММИРОВАНИЯ

Первые компьютеры (ENIAC, ...)

МАШИННЫЕ КОДЫ

0 и 1 хороши для кремния, но не для программистов

ПЕРВЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ

```
        .TITLE  HELLO WORLD
        .MCALL  .TTYOUT,.EXIT
HELLO:: MOV     #MSG,R1 ;STARTING ADDRESS OF STRING
1$:     MOVB    (R1)+,R0 ;FETCH NEXT CHARACTER
        BEQ     DONE    ;IF ZERO, EXIT LOOP
        .TTYOUT         ;OTHERWISE PRINT IT
        BR      1$      ;REPEAT LOOP
DONE:   .EXIT

MSG:    .ASCIZ /Hello, world!/
        .END    HELLO
```

Язык низкого уровня — язык Ассемблера

БАЗОВАЯ ИДЕЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

011001
110101

Компьютер

Машинные коды

program cool
begin
.....
end

Транслятор

011001
110101

Компьютер

Использование ЯП

СПОСОБЫ ТРАНСЛЯЦИИ

Программа (исходный код)

Компиляция

Компилятор

Исполняемый код

Интерпретация

Программа (исходный код)

Интерпретатор

Пользователь

ВИРТУАЛЬНЫЕ МАШИНЫ

Программа (исходный код)

Компилятор

Компиляция

Промежуточный (байт-) код

Интерпретация, иногда компиляция

Виртуальная машина

Пользователь

ЗАДАЧКА О ПРИРОДЕ ЯЗЫКА С

Русская Википедия утверждает, что язык С — компилируемый.
Возможен ли интерпретатор для С?
Существуют ли языки программирования, для которых не может существовать компилятора?
Ответы пояснить.

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

- Императивная. Программа – это и есть алгоритм. В каждый момент есть глобальное состояние (содержимое памяти), явно доступное для изменения. *Языки: Pascal, C, …*

- Функциональная. Программа – это функция. Нет явного состояния. *Языки: LISP, Haskell, Microsoft Excel (!).*

- Логическая. Программа – это набор предикатов и правил вывода. *Языки: Prolog, SQL.*

ИМПЕРАТИВНЫЙ C

```c
int fact(int n) {
    int result = 1;
    for (int i = 2; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

ФУНКЦИОНАЛЬНЫЙ HASKELL

```
fact 1 = 1
fact n = n * fact (n - 1)
```

ФУНКЦИОНАЛЬНЫЙ EXCEL

	A	B
1	n	fact(n)
2	1	1
3	2	2
4	3	6
5	4	24
6	5	=B5*A6
7	6	720
8	7	5040
9	8	40320
10	9	362880
11	10	3628800

ЛОГИЧЕСКИЙ PROLOG

```
fact(1, 1):-!.
fact(N, A):-
N1 = N - 1,
fact(N1, A1),
A = A1 * N.
```

ЯЗЫКИ ВЫСОКОГО УРОВНЯ

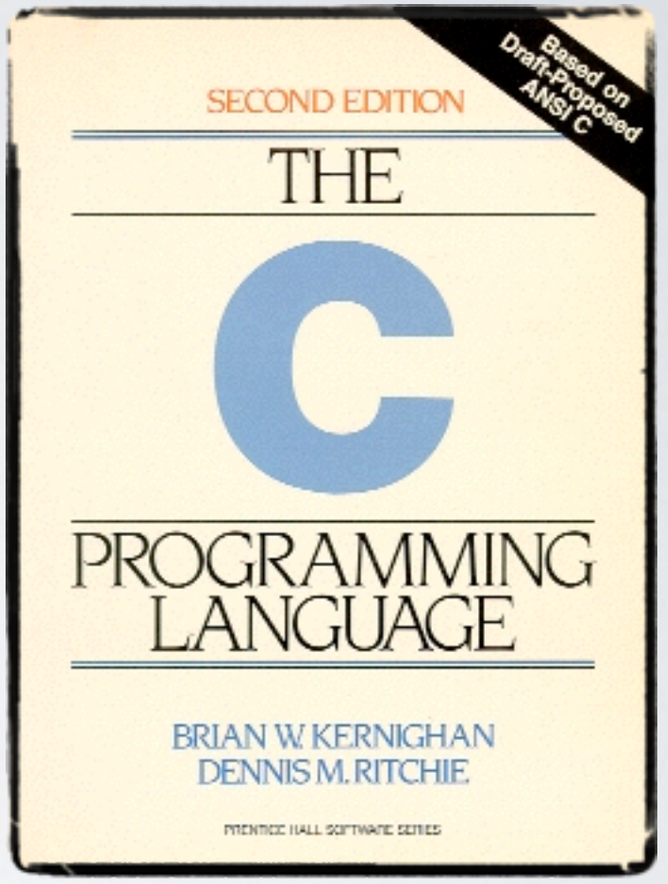
- FORTRAN (1950-е гг.)
- LISP (1950-е гг.)
- [Visual] BASIC (1964 г.)
- Pascal (1970 г.)
- C (1972 г.)
- C++ (1983 г.)
- Perl (1987 г.)
- Python (1991 г.)
- Ruby (1993 г.)
- Java (1995 г.)
- JavaScript (1995 г.)
- PHP (1995 г.)
- C# (2001 г.)
- Scala (2003 г.)
- Swift (2014 г.)
- …

ЯЗЫК С

- 1972 г., Деннис Ричи (Bell Labs).
- «Переносимый ассемблер» для системных целей:
  - Операционные системы: ядра, системные библиотеки.
  - Компиляторы, виртуальные машины, …
- Дает широкие возможности, но требует аккуратности.

ИСТОРИЯ

- 1972 г. – рождение.
- 1978 г. – выход книги, «K&R» C.
    - 1983 г. – отпочковался C++.
- 1989 г. – стандарт ANSI C или C89.
- 1999 г. – стандарт C99 (не совместим с C++!).
- 2011 г. – стандарт C11.



КОМПИЛЯТОРЫ

- GNU Compiler Collection.

- Microsoft Visual C++.

- Intel C++ Compiler.

- Clang + Low Level Virtual Machine.

HELLO_WORLD.C

```c
#include <stdio.h>

int main(void) {
    printf("Hello, world!\n");
    return 0;
}
```

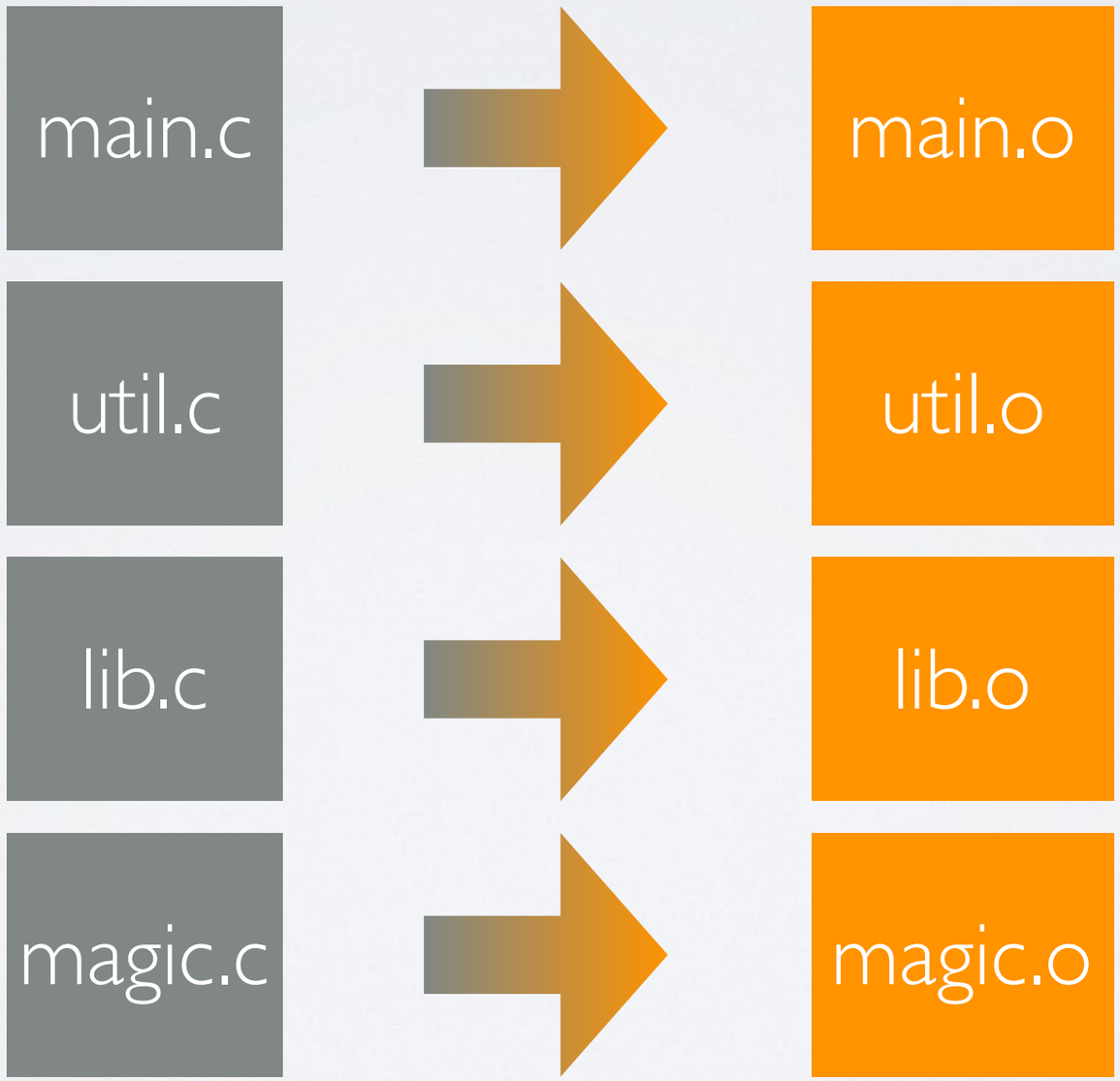
СТРУКТУРА ПРОГРАММЫ

- Программа состоит из функций, в т. ч. функции `main()`.
- Функции расположены в файлах *.c. Например:
  - main.c
  - util.c
  - lib.c
  - magic.c

ЭТАП 1. КОМПИЛЯЦИЯ

Исходные файлы (модули)

Объектные файлы



ОБЪЕКТНЫЙ ФАЙЛ FILE.O

- Машинный код функций, объявленных в file.c.
- Память под объявленные глобальные переменные.
- Ссылки на внешние функции.
- Ссылки на глобальные переменные.

- hello_world.c:
  - Машинный код функции `main()`.
  - Ссылка на внешнюю функцию `printf()`.

ЭТАП 2. ЛИНКОВКА

РАБОТА ЛИНКЕРА

- Операционная единица: **имя**. Каждый объектный модуль (в т. ч. библиотечный):
  - **Предоставляет** какие-то имена (функции, переменные, …)
  - **Требует** какие-то имена.
- Линкер удовлетворяет зависимости (все начинается с имени `main`).

ОШИБКИ ЛИНКЕРА

- Ошибки:
  - Имя требуется одним из модулей, но никаким не предоставляется.
  - Одно и то же имя предоставляется более, чем одним модулем.

```
Undefined symbols for architecture x86_64:
  "_foo", referenced from:
      _fact in fact-xGnG1z.o
ld: symbols(s) not found for architecture x86_64

duplicate symbol _fact in:
    /var/.../foo-ogM19a.o
    /var/.../bar-X6PLBX.o
ld: 1 duplicate symbol for architecture x86_64
```

ИСПОЛНЯЕМЫЙ ФАЙЛ

- Содержит все нужные имена (и ничего лишнего). Все ссылки на имена в объектных файлах были разрешены.

- По построению зависит от объектных файлов и библиотек (те зависят от исходных файлов).

- Для выполнения не нужно больше ничего (за исключением динамических библиотек).

ТЕНДЕНЦИИ

- Языки программирования становятся все мощнее за счет усложнения абстракций *(ср. кирпич vs. разборный дом)*.

- Тупые куски кремния не становятся умнее, но усложняются, поэтому «объяснять» им все сложнее.

- Трансляторы становятся все сложнее.

- Ожидания пользователей растут.

WTF?!

- Программист просто *обязан* постоянно учиться — новые технологии появляются ежедневно.

- Есть базовые вещи (алгоритмы, структуры данных, …).

  - Ими-то мы и займемся!

КОНЕЦ ПЕРВОЙ ЛЕКЦИИ