Презентация по информатике. 1 курс МФТИ. Часть 1.

Переменные, операторы, функции, массивы, простые сортировки, строки, структуры, динамическое программирование.

Бирюков В. А.

October 3, 2019

Содержание

Терминал

Среда программирования на Linux

Для этого курса вам понадобится:

- Хороший терминал - на Linux уже установлен.
- Компилятор - gcc или clang. gcc скорей всего уже установлен, но если нет, то:

```
$ sudo apt install gcc
```

- Хороший текстовый редактор. Советую Sublime Text: sublimetext.com
  Другие редакторы, которые можно попробовать - VS Code и Atom.

Среда программирования на Windows

Для этого курса вам понадобится:

- Хороший терминал - стандартный терминал Windows очень плох, поэтому советую установить cmder:
  cmder.net
- Компилятор - MinGW (аналог gcc на Windows).
  mingw.org
  Не забудьте установить переменную PATH.
- Хороший текстовый редактор. Советую Sublime Text:
  sublimetext.com
  Другие редакторы, которые можно попробовать -
  VS Code и Atom.

Работа с терминалом

- pwd - печатает текущую директорию

```
$ pwd
> /home-local/student
```

- cd - переходим в другую директорию.

  Перейти в папку workspace текущей директории:

```
$ cd workspace
```

  Перейти в папку выше:

```
$ cd ..
```

Работа с терминалом

- ls - печатает все файлы в текущей директории.

```
$ ls
workspace/ prog.c data.txt a.out
```

- ls -l - то же самое, но больше информации

```
$ ls -l
drwxr-xr-x student 0 Oct 3 workspace/
-rw-r--r-- student 450 Nov 16 prog.c
-rw-r--r-- student 620 Nov 16 data.txt
-rwxr-xr-x student 12654 Nov 16 a.out
```

Слева направо - права доступа (read, write, execute), владелец, размер в байтах, дата изменения и название файла.

Работа с терминалом

- cp - копирование

  Копируем файл `prog.c` в папку `workspace`:

  ```
  $ cp prog.c workspace/
  ```

- mv - перемещение.

  Перемещаем файл `prog.c` в папку `workspace`:

  ```
  $ mv prog.c workspace/
  ```

  Переименовываем файл `prog.c` в `new.c`:

  ```
  $ mv prog.c new.c
  ```

Компилятор gcc.

- gcc - компилятор
  Компилируем файл `prog.c`. В результате создаётся исполняемый файл `a.out`:

```
$ gcc prog.c
```

  Запускаем созданную программу:

```
$ ./a.out
```

  Удобно объединить эти операции:

```
$ gcc prog.c && ./a.out
```

Компилятор gcc. Опции.

- Опция `-o`

  Задаём имя выдаваемого исполняемого файла:

  ```
  $ gcc -o prog prog.c
  ```

  Будет создан файл `prog`, а не `a.out`.

- Опция `-std=c99`

  Используем более новую версию языка C (99-го года):

  ```
  $ gcc -std=c99 prog.c
  ```

- Опция `-lm`

  Эта опция нужна для линковки математической библиотеки `math.h`.

  ```
  $ gcc -lm prog.c
  ```

Функция main(), переменные, printf и scanf

Функция main

Простейшая программа на языке C:

```
int main() {
}
```

Любая программа на языке C должна содержать main.

Выполнение программы начинается с этой функции.

Данная программа ничего не делает.

Код возврата

Функция `main` возвращает код возврата.

```
int main() {
    return 0;
}
```

`0` - программа завершилась без ошибок.

не `0` - программа завершилась с ошибками.

Многие компиляторы не требуют писать `return 0;` на конце `main`. В этой презентации эта строка будет опускаться.

Hello world

Программа, которая печатает Hello world:

```
#include <stdio.h>
int main() {
    printf("Hello world");
}
```

Библиотека `stdio.h` (standard input/output) содержит функции для работы вводом данных в программу или выводом их из программы.

В частности, `printf` печатает на экран.

`printf` = `print formatted` = форматированная печать.

Hello world

Программа, которая печатает Hello world:

```
#include <stdio.h>
int main() {
    printf("Hello world\n");
}
```

с переносом на новую строку на конце.

"\n" = перенос на новую строку (Enter)

"\t" = табуляция (Tab)

"\b" = передвигаем каретку влево (стрелочка влево)

"\b \b" = удалить 1 символ (backspace)

Комментарии

В код можно вставлять комментарии.

```
int main() {
        // Комментарий в 1 строку

        /* Комментарий
                в несколько
                        строк   */
}
```

При компиляции этот текст отбрасывается и никак не влияет на работу программы.
Комментарий нужны чтобы сделать код более понятным.

Переменные типа `int`

`int` = integer = целые числа. `int` - переменная, которая хранит целые числа.
Создадим переменную под именем a, предназначенную для хранения целых чисел:

```
int main() {
    int a;
}
```

Кстати, функция `main` также возвращает переменную типа `int` - код возврата.

Переменные типа int

int - переменная, которая хранит целые числа.

```
int main() {
    // Объявление:
    int a;
    // Объявление и инициализация:
    int b = 451;
    // Присваивание:
    b = 233;
}
```

Имена переменных

- Имена переменных могут содержать любые латинские буквы и верхнего и нижнего регистров, цифры и символ подчёркивания _
- Но не могут начинаться с цифры

```
int main() {
    int ab$c; // Ошибка
    int 5x;   // Ошибка
    int _fj374834oJR_394t;  // Ок
    // ( но лучше выбирать осмысленные имена )
}
```

Печать значений переменных типа int

```c
#include <stdio.h>
int main() {
    int a = 451;
    printf("The password is %d.\n", a);
}
```

Вместо %d напечатает 451

Печать значений переменных типа int

```c
#include <stdio.h>
int main() {
    int a = 451;
    // Часто нужно напечатать только число:
    printf("%d\n", a);
    // %5d - печатает как минимум 5 символов
    // Если число меньше, то дополнит пробелами
    printf("%5d\n", a);
    // %05d - печатает как минимум 5 символов
    // Если число меньше, то дополнит нулями
    printf("%05d\n", a);
}
```

Печать значений переменных типа int

```c
#include <stdio.h>
int main() {
    int a = 451;
    // Печать в шестнадцатеричной системе:
    printf("%x\n", a);
}
```

Адрес и размер переменной

Расположение переменных в памяти характеризуется их адресом(номером первого байта) и размером переменной.

```
#include <stdio.h>
int main() {
    int a = 451;
    &a; // так можно найти адрес переменной
    sizeof(a); // так можно найти размер
    // Напечатаем их:
    printf("Address of a = %x.\n", &a);
    printf("Size of a = %d.\n", sizeof(a));
}
```

Приоритет операторов

1. (), []
2. ++, –, +, -(унарные), sizeof
3. *, /, %
4. +, -
5. >,<,<=,>=
6. ==, !=
7. &, |, &&, ||
8. =, +=, и т.д.

Приоритет операторов C подробнее:
ru.cppreference.com/w/c/language/operator_precedence

Управляющие конструкции: if else, циклы while и for

```c
#include <stdio.h>
int main() {
    int x;
    scanf("%d", &x);
    if (x < 5)
        printf("%d is less than five!!!\n", x);
}
```

```c
#include <stdio.h>
int main() {
    int x;
    scanf("%d", &x);
    if (x < 5)
       printf("%d\n", x);
    else if (x == 5)
        printf("Equal to 5\n");
    else
        printf("More than 5\n");
}
```

Базовые управляющие конструкции

Тернарный оператор :?

```
z = (x > 0) ? x : -x;
min = (x < y) ? x : y;
```

Базовые управляющие конструкции

Цикл while

```
#include <stdio.h>
int main() {
    int i = 1;

    while (i < 10) {
        printf("%d ", i);
        i++;
    }
}
```

Напечатает 1 2 3 4 5 6 7 8 9

Базовые управляющие конструкции

Цикл do while

```
int i = 1;

do
{
        printf("%d ", i);
        i++;
} while (i < 4)
```

Напечатает 1 2 3

```
for (int i = 0; i < 3; ++i)
        printf("%d ", i);
```

Напечатает 1 2 3

В условии циклов может стоять любой оператор:

```
int i = 0;

while (i++ < 3)
        printf("%d ", i);
```

```
for(int i = 0; i < 10; ++i){
        if(i == 6){
                break;
        }
        printf("%d ", i);
}
```

```
for(int i = 0; i < 10; ++i){
        if(i == 6){
                continue;
        }
        printf("%d ", i);
}
```

```
switch(i) {
        case 1:
                printf("It's one!\n");
                break;
        case 2:
                printf("It's two!\n");
                break;
        default:
                printf("It's something
    else!\n")
}
```

- Оператор goto передает управление на оператор, помеченный меткой
- Оператор goto в языках высокого уровня является объектом критики, поскольку чрезмерное его применение приводит к созданию нечитаемого кода
- Использование goto в практике программирования на языке C настоятельно не рекомендуется

Массивы

Строки

Динамическое программирование

Указатели

Стиль кода

Комментарии

```
// One line comment
/* Another one */
/* Multi-
        line
        comment*/
```

Стиль кода

- Отступы. В программе должна быть структура. Количество отступов соответствует уровню вложенности. Уровень вложенности увеличивается внутри , а также в телах операторов if, for, while, do-while, switch
- Каждый отступ - это ЛИБО TAB, ЛИБО n пробелов (лучше всего n = 4). Мешать их нельзя.
- Скобка { должна быть на следующей строке, под началом ключевого слова if/for/.

Стиль кода

- Скобка } должна быть СТРОГО под соответствующей {. После неё не должно быть ничего, за исключением комментариев.
- Каждый оператор (особенно, содержащий ключевое слово) должен быть с новой строки, после оператора и знака ; не должно быть ничего, кроме комментариев.
- Пробелы должны быть ПОСЛЕ , и ;(в цикле for) , а до них они НЕ нужны.