

# Семинар №1

ФАКИ 2017

---

Бирюков В. А.

October 19, 2017

- CS50 (просто загуглить)
- K&R: Брайан У. Керниган, Деннис М. Ритчи "Язык программирования C"
- Томас Кормен, Чарльз Лейзерстон, Рональд Ривест, Клиффорд Штайн "Алгоритмы: построение и анализ"

# Основы командной строки Linux

---

**pwd** (сокращение от **personal working directory**)

**ls** (сокращение от **list**)

Опции: **-l**, **-a**

**cd** (**change directory**)

Применение: **cd** <имя директории>

Особые директории: **.** **..** **~**

**man** (**manual**)

Применение: **man** <имя команды>

Например: **man ls**

### cp (copy)

Применение: cp <источник> <назначение>

### mv (move)

Применение: mv <источник> <назначение>

Можно переименовывать файлы

### rm (remove)

Применение: rm <имя файла>

Чтобы удалить директорию: опция -r

Будьте осторожны!

### **mkdir (make directory)**

Применение: `mkdir <название директории>`

### **nano - текстовый редактор**

`nano ./<имя_файла>`

Ctrl + X - закрыть редактор

Ctrl + O - сохранение файла

### **vim - продвинутый текстовый редактор**

`vim ./<имя_файла>`

i - перейти в режим редактирования

ESC - выйти из режима редактирования

:wq и :q! - выйти с сохранением и без

сохранения соответственно

# Hello world. Компилятор gcc.

---

# Пример простейшей программы на языке C

Программа, которая ничего не делает

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

Функция `main` – начальная точка выполнения для всех C и C++ программ



```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello world!");
```

```
}
```

*Подключаем библиотеку stdio.h*



```
#include <stdio.h>
```

```
int main()  
{  
    printf("Hello world!");  
}
```

*Подключаем библиотеку stdio.h*



```
#include <stdio.h>
```

*Функция main() начальная*

*точка выполнения программы*



```
int main()
```

```
{
```


```
    printf("Hello world!");
```

```
}
```

## Программа, которая печатает Helloworld на экран

Подключаем библиотеку `stdio.h`

  
**#include** <stdio.h>

**int** main()  *Функция `main()` начальная  
точка выполнения программы*

{

printf("Hello world!");

} 

*Функция из библиотеки `stdio.h`  
Печатает строку на экран*

Подключаем библиотеку `stdio.h`

**#include** <stdio.h>

**int** main() ← Функция `main()` начальная  
точка выполнения программы

{

printf("Hello world!\n");

}

↗  
Функция из библиотеки `stdio.h`  
Печатает строку на экран

↖  
`\n` - символ  
переноса строки

## gcc (GNU Compiler Collection)

gcc <имя файла для компиляции>

-o – задать имя файла

Имя файла по умолчанию: a.out

Примеры:

**gcc -o hello hello.c**

Скомпилирует файл hello.c и создаст исполняемый файл hello, который можно будет запустить исполнив ./hello

# Основы языка С. Базовые типы и операторы.

---

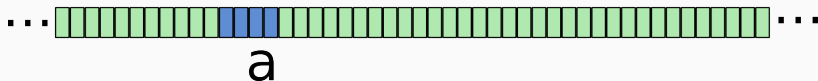
- Простой синтаксис
- Простой доступ к памяти, указатели
- Низкоуровневый
- Очень быстрый
- Небезопасный
- Сложно писать большие программы



- Именованная область памяти, адрес которой можно использовать для осуществления доступа к данным
- Название переменной может содержать латинские буквы, цифры и `_`, но не может начинаться с цифры

```
int a;
```

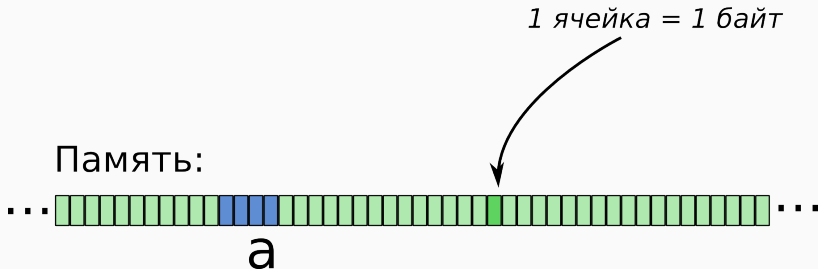
Память:



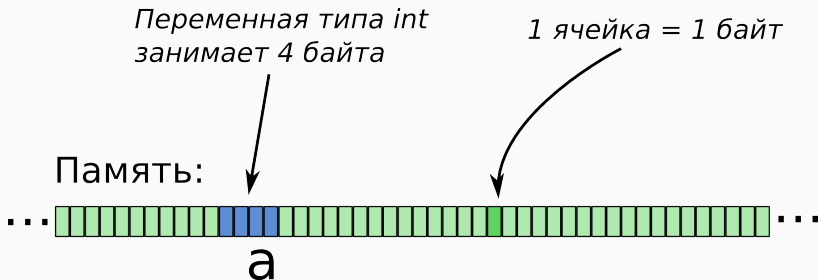
# Память и переменные

## Объявление переменной типа int(целое число)

**int** a;

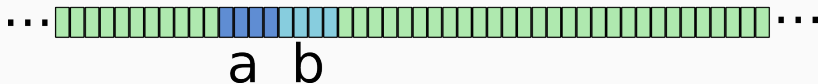


**`int`** `a`;



```
int a;  
float b;
```

Память:



# Память и переменные

Объявление `int`, `float` и `char`(целое число размером 1 байт)

```
int a;  
float b;  
char c;
```

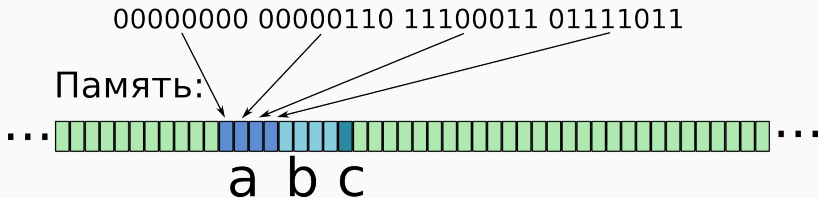
Память:



```
int a = 451451;
```

```
float b;
```

```
char c;
```

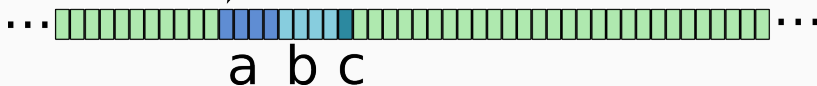


```
int a;  
float b;  
char c;
```

*Адрес переменной -  
номер ячейки памяти*

*&a - адрес этой  
ячейки памяти*

Память:

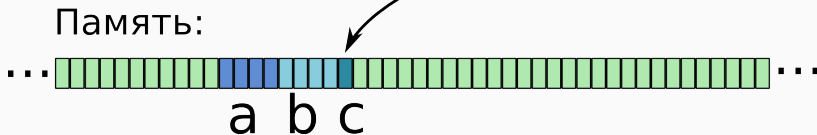


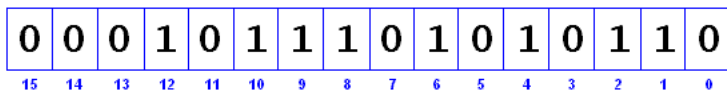


```
int a;  
float b;  
char c;
```

*Адрес переменной -  
номер ячейки памяти*

*&c - адрес этой  
ячейки памяти*





Число бит на тип зависит от системы. Для 64-х битных:

Название типа	Число бит	Макс. значения
char	8	-128..127 или 0..255
short	16	-32768..32767
int	32	$-2 \cdot 10^9 \dots +2 \cdot 10^9$
long int	64	$-2^{63} \dots +2^{63} - 1$
long long int	64	$-2^{63} \dots +2^{63} - 1$

# Беззнаковые целочисленные типы

Число бит на тип зависит от компилятора. Обычные значения такие:

Название типа	Число бит	Макс. значения
unsigned char	8	0..255
unsigned short	16	0..65535
unsigned int	32	0 ..+4 · 10 <sup>9</sup>
unsigned long int	64	0 ..+2 <sup>64</sup> – 1
unsigned long long int	64	0 ..+2 <sup>64</sup> – 1

sizeof() – размер файла в байтах

# Типы чисел с плавающей точкой



Название типа	Число бит	Макс. значения
float	32	$10^{-38}..10^{+38}$
double	64	$10^{-308}..10^{+308}$

Обычно используется double, так как float может быть недостаточно точен

# Оператор присваивания =

Присваивает переменной значение:

Пример:

```
int a, b;
```

```
float c, d;
```

```
a = 1;
```

```
b = a + 1;
```

```
c = 5.6;
```

```
d = 19;
```

```
a = 4.6;
```

# Математические операторы: + - / \* %

Примеры:

`a = 1 + 1;`

`b = 5.0 / 2.0;`

`c = 5 / 2;`

`d = 5 % 2;`

## Унарные операторы: + - ++ - -

Оператор инкремента ++ – увеличивает значение переменной на 1 и присваивает переменной

Примеры:

```
a = +5;
```

```
b = -a;
```

```
c = ++a;
```

```
d = c++;
```

Функции `printf()` и `scanf()`.

---



# Функции printf и scanf.

printf(строка форматирования, пер1, пер2, ...)

scanf(строка форматирования, &пер1, &пер2, ...)

Обозначение	Типы	Пример
i	Целочисленные типы	392
f	Типы с плавающей точкой	392.5