

# Семинар №9

## ФАКИ 2015

Бирюков В. А.

November 6, 2015

# Указатели

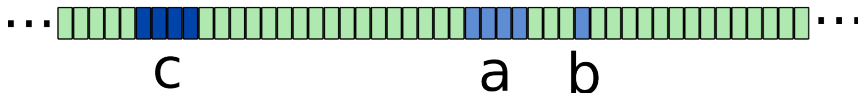
# Указатели

- Указатель – это переменная, содержащая адрес другой переменной.
- Указатели и массивы тесно связаны между собой

# Указатели

Указатели в памяти, объявление указателей

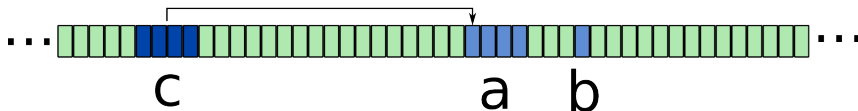
```
int a;  
char b;  
int * c;
```



# Указатели

Адрес переменной

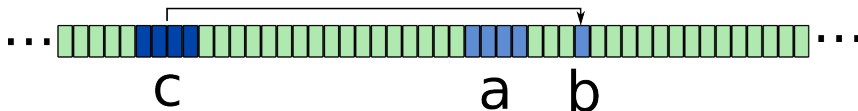
```
int a;  
char b;  
int * c = &a;
```



# Указатели

Адрес переменной

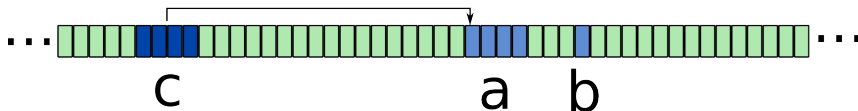
```
int a;  
char b;  
char * c = &b;
```



# Указатели

Ссылка по указателю

```
int a = 1;  
int * c = &a;  
*c = 5;
```



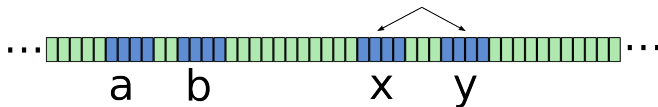
# Указатели и аргументы функций



# Указатели и аргументы функций

## Передача по значению

```
void swap(int x, int y) /* НЕПРАВИЛЬНО! */  
{  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}  
...  
swap(a, b);
```

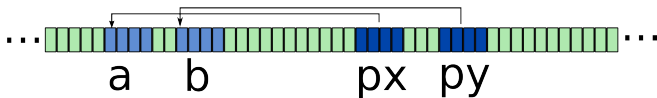


# Указатели и аргументы функций

Передача по адресу

```
void swap(int * px, int * py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}

...
swap(&a, &b);
```



# Указатели и массивы

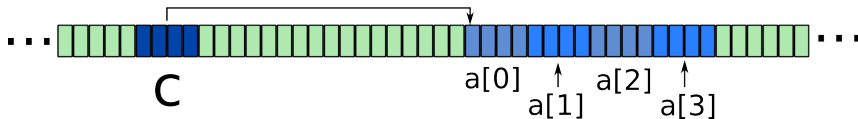
# Указатели и массивы

Указатель на элемент массива

```
int a[4] = {1, 2, 3, 4};
```

```
int *c = &a[0];
```

```
*c = 5;
```



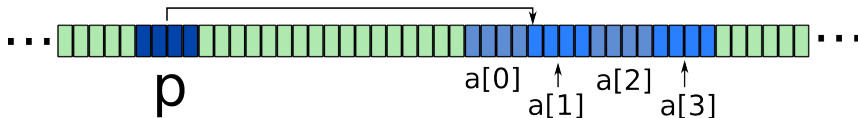
# Указатели и массивы

## Адресная арифметика

```
int a[4] = {1, 2, 3, 4};
```

```
int * p = &a[0];
```

```
p = p + 1;
```



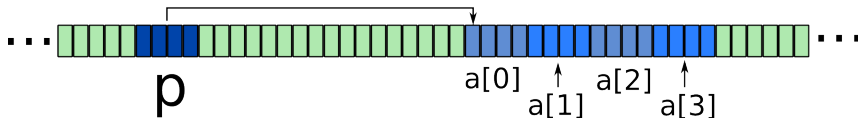
# Указатели и массивы

## Адресная арифметика

```
int a[4] = {1, 2, 3, 4};
```

```
int * p = &a[0];
```

```
int x = *(p + 2);
```



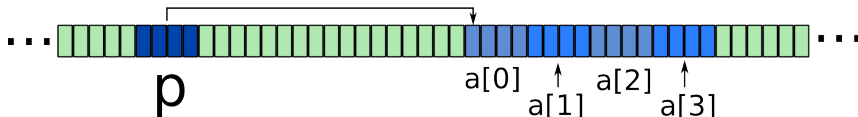
# Указатели и массивы

## Связь массивов и указателей

```
int a[4] = {1, 2, 3, 4};
```

```
int * p = &a[0];
```

```
int x = *(p + 2);
```



**\*(p+2) и a[2] - это одно и то же**

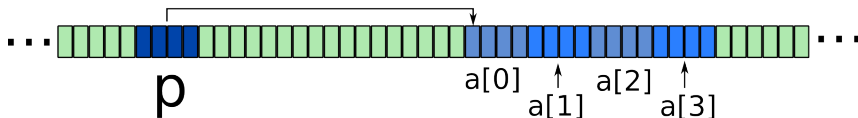
# Указатели и массивы

## Связь массивов и указателей

```
int a[4] = {1, 2, 3, 4};
```

```
int * p = &a[0];
```

```
int x = *(p + 2);
```



**Более того, в С массивы реализованы с помощью указателей**

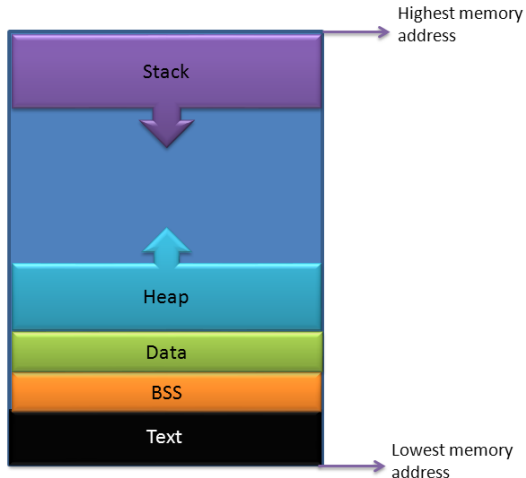
**Название массива - это указатель на первый элемент (т.е.  $a == p$ )**



# Mallos и free. Управление памятью.

# Управление памятью

## Сегменты памяти процесса



# Стек (Stack)

- Стек представляет собой обычный алгоритмический стек, применённый для управления памяти
- В нём хранятся локальные переменные
- Имеет фиксированный размер, определяется операционной системой, на порядок меньше чем Куча
- Немного быстрее, чем Куча

# Куча (Heap)

- Куча представляет собой обычную алгоритмическую кучу, применённую для управления памяти
- В ней можно динамически выделять память
- Размер, обычно, ограничен только доступными ресурсами
- Немного медленней, чем Куча

# Выделение памяти в Куче с помощью malloc и free

```
int *p;  
p = (int *)malloc(sizeof(int));  
if (p == 0)  
{  
    printf("ERROR: Out of memory\n");  
    return 1;  
}  
  
*p = 25;  
printf("%d\n", *p);  
  
free(p);
```

# Выделение памяти в Куче с помощью malloc и free

```
int *p;
p = (int *)malloc(100 * sizeof(int));
if (p == 0) {
    printf("ERROR: Out of memory\n");
    return 1;
}

for (int i = 0; i < 100; ++i) {
    *(p+i) = 123;
}
printf("%d\n", *(p+50));

free(p);
```

# Задание

# Задание

- Задачи на указатели – Начинаящие, Задание 2 (часть 1)
- Задачи на структуры – rect01
- \* МАЛО Калорий в картофеле ФРИ, задача mem\_1