

Семинар №7

ФАКИ 2017

Бирюков В. А.

September 7, 2017

Многомерные массивы

```
type name[size1][size2]...[sizeN];
```

например объявление двумерного массива:

```
int array[5][10];
```

задание значений:

```
for (int i = 0; i < 5; ++i)
    for (int j = 0; j < 10; ++j)
        array[i][j] = something;
```

/* массив из 5-ти рядов и 2-х столбцов */

```
int array[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}
```

```
int i, j;
```

/* вывод всех элементов массива */

```
for ( i = 0; i < 5; i++ )
```

```
{
```

```
    for ( j = 0; j < 2; j++ )
```

```
    {
```

```
        printf("%d\n", array[i][j] );
```

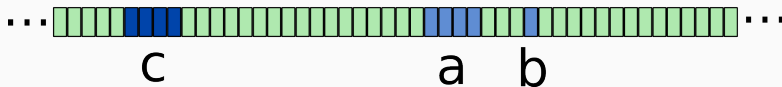
```
    }
```

```
}
```

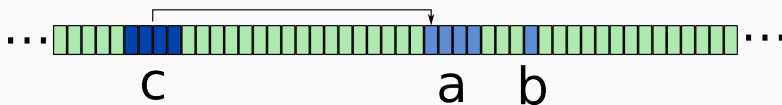
Указатели

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

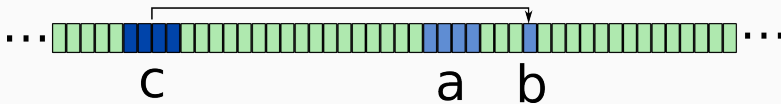
```
int a;  
char b;  
int * c;
```



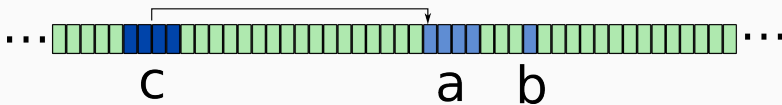
```
int a;  
char b;  
int * c = &a;
```




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



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

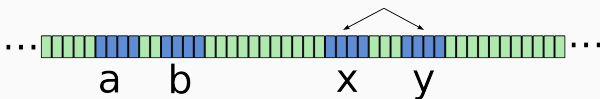


Указатели и аргументы функций

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);
```



```
void swap(int * px, int * py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
...
swap(&a, &b);
```

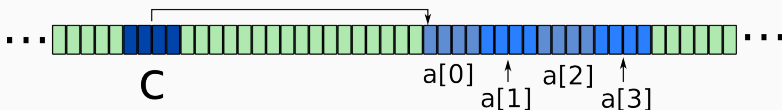


Указатели и массивы

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

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

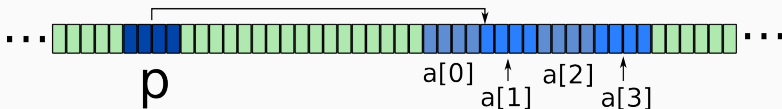
```
*c = 5;
```



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

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

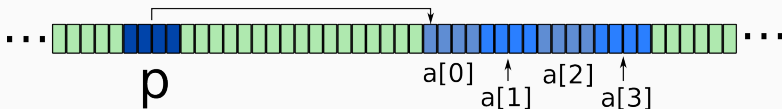
```
p = p + 1;
```




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

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

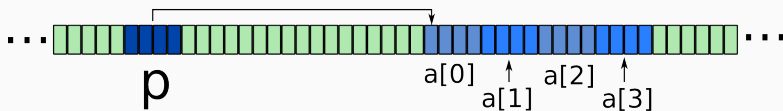
```
int x = *(p + 2);
```



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

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

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

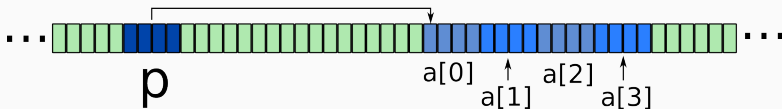


`*(p+2)` и `a[2]` - это одно и то же

```
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. $a == p$)

Маллос и free. Управление памятью.

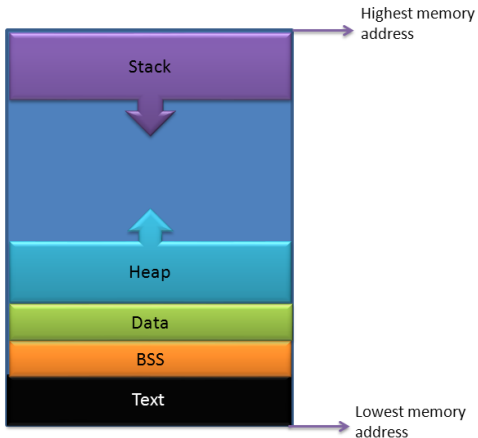


Figure : Process memory organization

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

Выделение памяти в Куче с помощью malloc и free

Выделение памяти на 1 переменную типа int

```
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

Выделение памяти на массив из 100 переменных типа int

```
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);
```

- Оператор sizeof
- Приведение типов:

```
float x = 5.2;  
int y = (int)x;
```

- Верно и для указателей:

```
int a = 42;  
float * p1 = &a;  
int * p2 = (int *)x;
```

- memcpu
- valgrind

Задание

- Задачи на указатели (МАЛО Калорий в картофеле ФРИ)