

# Динамическая память

## Лекция 1

# Размерности

1 байт = 8 бит

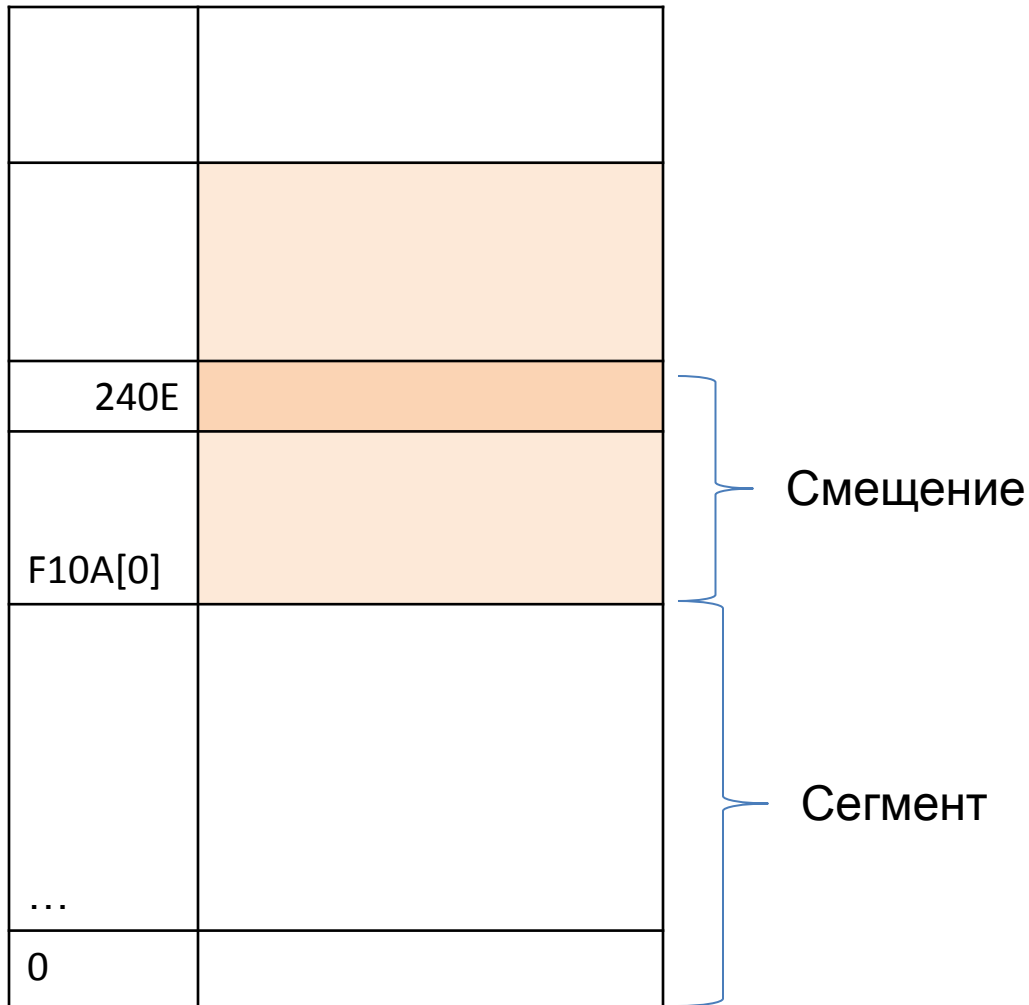
1 параграф =  $2^4$  байт

1 Кб =  $2^{10}$  байт

1 Мб =  $2^{20}$  байт

1 сегмент = 64 Кб =  $2^{16}$  байт

# Модель оперативной памяти ПК



адрес = (сегмент,  
смещение)

Абсолютный адрес =  
сегмент \* 16 + смещение

**Пример**

Адрес = (F10A, 240E)

Абс. адрес = F10A0 + 240E

$$\begin{array}{r} \text{F10A0} \\ + \\ \text{240E} \\ \hline \text{F34AE} \end{array}$$

# Модель карты памяти

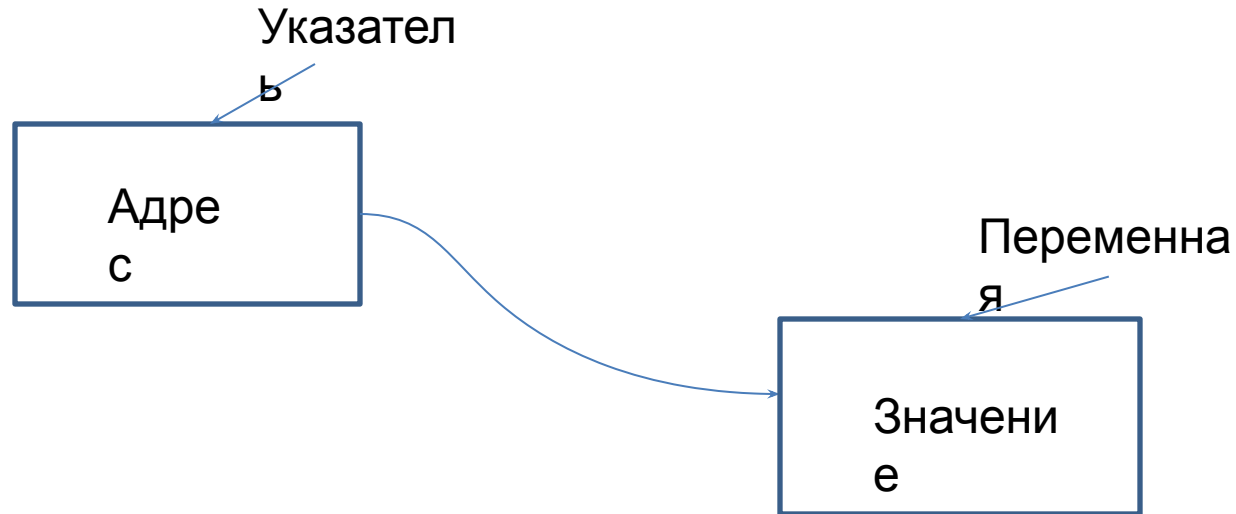


# Сравнение статической и динамической памяти

Параметры сравнения	Статические переменные	Динамические переменные
Способ распределения памяти	Автоматическое (во время компиляции)	Управляется программой
Место расположения	Глобальные переменные – в сегменте данных, локальные – в сегменте стека	В динамической памяти (куче)
Способ доступа	По имени (идентификатор)	По адресу (указатель на место расположения в памяти)

# Указатель

Указатель – это переменная, значением которой является адрес области памяти



# Описание указателей

На Паскале

```
var  
  p : pointer;  
  t : ^integer;  
  n: integer;
```

...

```
n := t^;
```

На Си

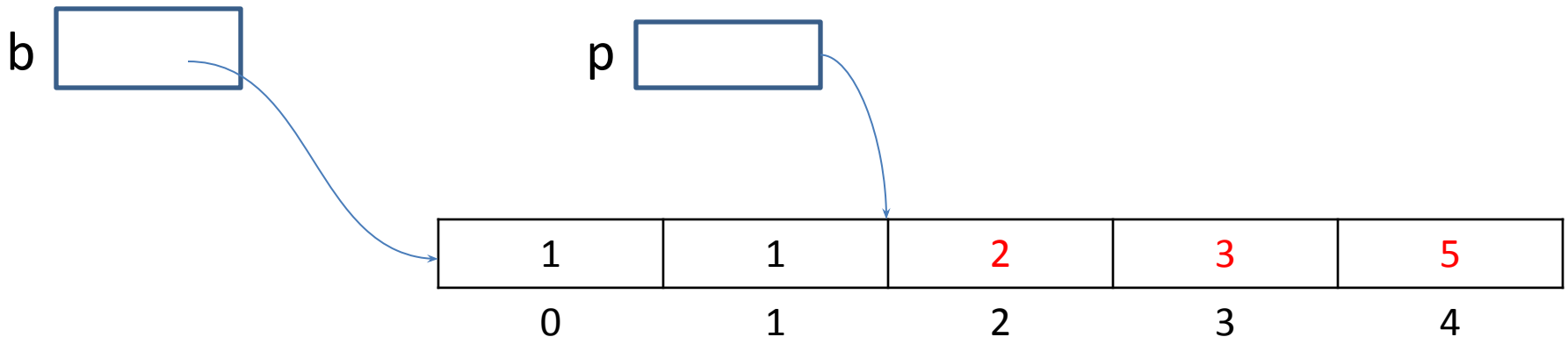
```
int *t;  
int n;
```

...

```
n = *t;  
//разыменование  
t = &n; //адрес
```

# Указатели и массивы

```
int b[5] = {1, 1};  
int *p, i;  
for (i = 2; i < 5; i++)  
    b[i] = b[i-1]+b[i-2];  
//-----  
for (p = b+2; p != b+5; p++)  
    *p = *(p-1) + *(p-2);
```





# Строки в Си

```
#include <string.h>
```

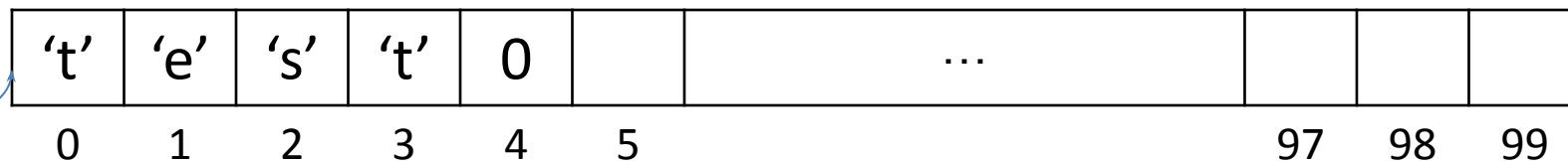
```
...
```

```
char S[100];
```

```
int l;
```

```
strcpy (S, "test");
```

```
l = strlen(S);
```



S

l

# Функции работы с динамической памятью

Функции	Прототипы и краткое описание
<b>malloc</b>	<b>void * malloc ( unsigned s );</b> Возвращает указатель на начало области динамической памяти длиной в <i>s</i> байт. При неудачном завершении возвращает значение <i>NULL</i> .
<b>calloc</b>	<b>void * calloc ( unsigned n, unsigned m );</b> Возвращает указатель на начало области обнуленной динамической памяти, выделенной для размещения <i>n</i> элементов по <i>m</i> байт каждый. При неудачном завершении возвращает значение <i>NULL</i> .
<b>realloc</b>	<b>void * realloc ( void * p, unsigned ns );</b> Изменяет размер блока ранее выделенной памяти до размера <i>ns</i> байт. <i>p</i> - адрес начала изменяемого блока. Если <i>p = NULL</i> (память раньше не выделялась), то функция выполняется как <i>malloc</i> .
<b>free</b>	<b>void * free ( void * p );</b> Освобождает ранее выделенный участок динамической памяти, адрес первого байта которого равен значению <i>p</i> .

# Пример работы с динамической памятью

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    float *t;
    int i, n;
    printf("\nn=");
    scanf("%d", &n);
    t= (float *)malloc(n*sizeof(float));
    for(i = 0; i < n; i++) {
        printf ("x[%d]=", i);
        scanf("%f", &(t[i]));
    }
    for(i = 0; i < n; i++) {
        if (i % 2 == 0) printf ("\n");
        printf("\tx[%d]=%f", i, t[i]);
    }
    free (t);
    return 0;
}
```

# Пример 2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char *s, *s1;
    int n;
    s = (char *)malloc(100);
    scanf("%s", s);
    for(n = 0; s[n]; n++);
    s1 = (char *)malloc(n*2 + 1);
    strcpy(s1, s);
    strcpy(s1 + n, s);
    printf("%s", s1);
    free(s);
    free(s1);
    return 0;
}
```

# Пример 3.

```
void swap (int *x, int *y)
{
    int a;
    a = *x;
    *x = *y;
    *y = a;
}
...
int main()
{
    int a, b;
    ...
    swap(&a, &b); // обмен значений двух переменных
    ...
    return 0;
}
```

# Структуры в Си

```
struct <ИМЯ ТИПА> {<ПОЛЯ>}
```

```
struct student {  
    char *name;  
    int age;  
};
```

```
struct student x, y, *z;
```

```
...
```

```
x.age = 19;
```

```
x.name = (char *)malloc(20);
```

```
scanf ("%s", x.name);
```

```
z = &x;
```

```
printf ("age = %d\n", (*z).age);
```

```
printf ("age = %d\n", z->age);
```