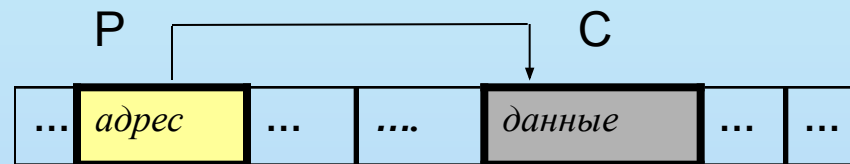


# УКАЗАТЕЛИ.



## Объявление

<тип> \*P;

```
int k=10, n, *ptr_k;  
float a=6.1, *ptr_a ,b;
```

## Операции

### 1.Получение адреса

**&**

$P = \&C$  – занести в ячейку P адрес ячейки C

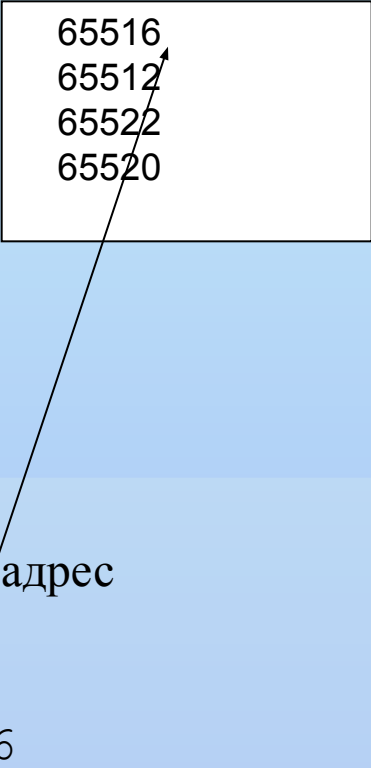
Пример: *Рассмотрим расположение переменных в оперативной памяти при их объявлении*

```
int n, k;  
float x, y;
```

65512	65513	65514	65515	65516	65517	65518	65519	65520	65521	65522	65523
y				x				k		n	

```
% u
printf("адрес %u ",
&var);
```

```
printf(" %u ", &x);
printf(" %u ", &y);
printf(" %u ", &n);
printf(" %u ", &k);
```



65516  
65512  
65522  
65520

используем указатель, например для x и получим его адрес

```
float *p;
p=&x;
printf("%u",p);
```

65516

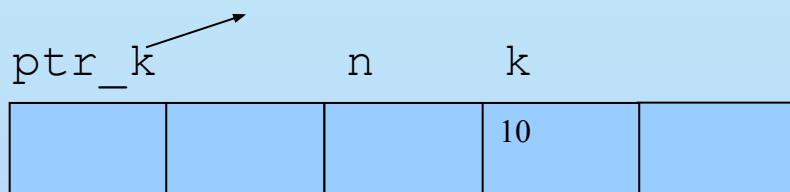
## 2. Получение значения по адресу.

\*  
\*p

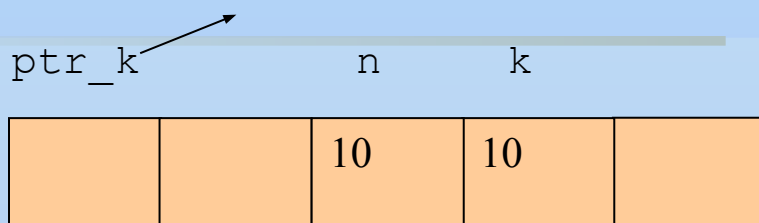
$t = *p$  - занести в ячейку  $t$  данные, расположенные в ячейке, на которую указывает указатель  $p$

Пример *Рассмотрим расположение переменных в оперативной памяти при объявлении указателя*

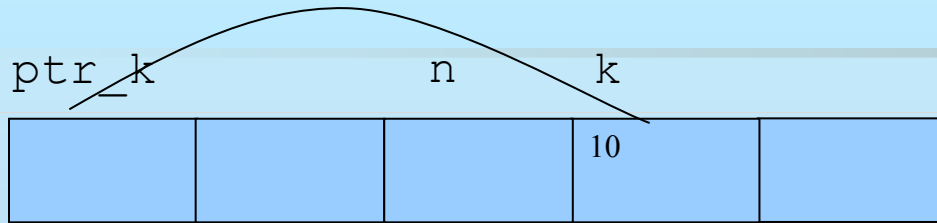
```
int k=10, n, *pt_k;
```



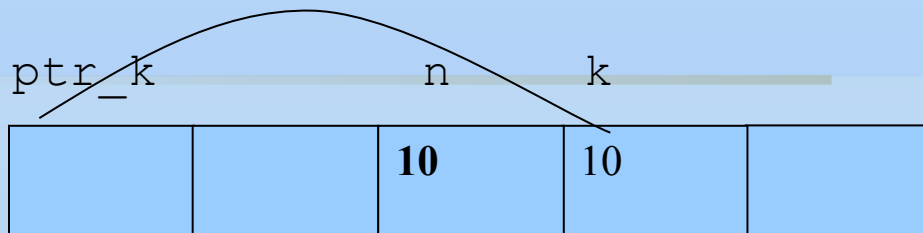
n=k



```
ptr_k=&k
```



```
n = *ptr_k;
```



```
void main()
{
    int *pt_k, n, k=10;
    pt_k=&k;
    n=*pt_k;
    printf("\n k=%d", k);
    printf("\n адрес_k=%u", &k);
    printf("\n pt_k=%u", pt_k);
    printf("\n
k_по_указателю=%d", *pt_k);
    printf("\n n=%d", n);
}
```

На экране  
k=10  
адрес\_k=65520  
pt\_k=65520  
k\_по\_указателю=10  
n=10

Аналогично для действительных чисел

```
float a=6.1, *ptr_a, b;  
ptr_a=&a;  
b=a;  
b=*ptr_a  
printf("a=%5.2f  b=%5.2f ",a,b );
```

На экране

```
a=6.1  b=6.1
```



# Указатели и массивы.

## 1. Одномерные массивы

Известное статическое описание

```
int a[10]
```



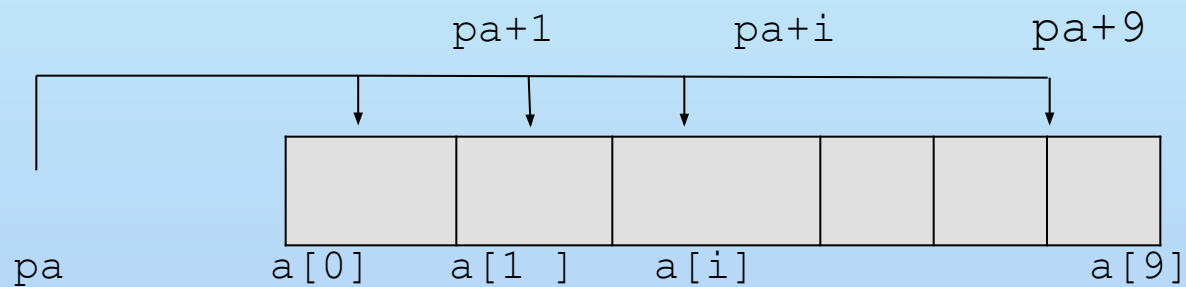
a[0]    a[1]                                    a[9]

a[i] - соответствует i-  
элементу.

## Описание с помощью указателя

```
int *pa
```

\*pa - указатель на тип int



\*pa → a[0]

\*(pa+1) → a[1]

**\*(pa+i) → a[i]**

```
float b[10]
```

```
printf(" %u ", &b);
```

65122

```
printf(" %u ", &b[0]);
```

65122

```
printf(" %u ", &b[1]);
```

65126

```
printf(" %u ", &b[2]);
```

65130

```
printf(" %u ", &b[3]);
```

65134

## 2. Двухмерные массивы

Известное статическое описание

```
int t[5][3]
```

$t_{00}$	$t_{10}$	$t_{20}$
$t_{01}$	$t_{11}$	$t_{21}$
$t_{02}$	$t_{12}$	$t_{22}$
$t_{03}$	$t_{13}$	$t_{23}$
$t_{04}$	$t_{14}$	$t_{24}$

$t[0][0]$	$t[0][1]$	$t[0][2]$	$t[1][0]$	$t[1][1]$	$t[1][2]$	$t[2][0]$	...		$t[4][2]$
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----	--	-----------

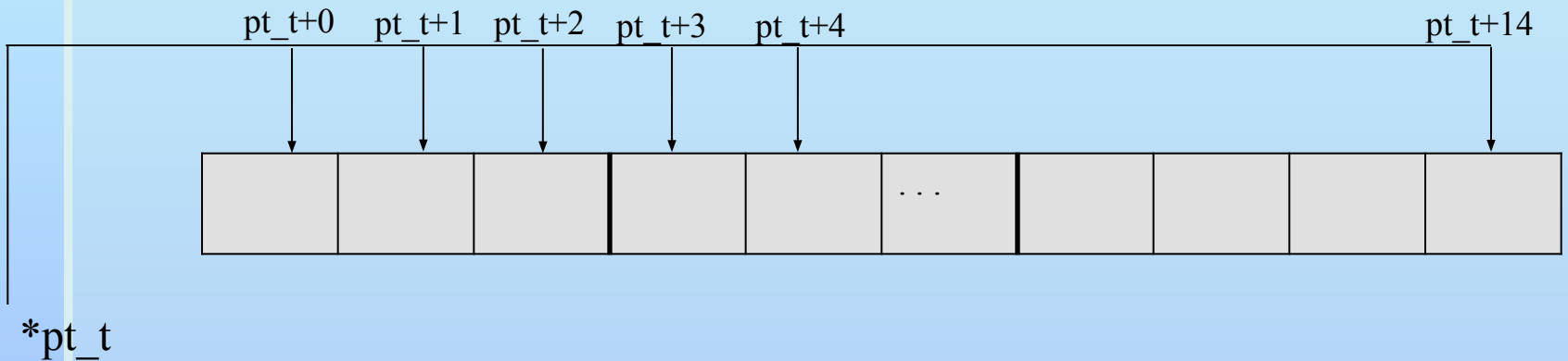
$t[i][j]$  — соответствует элементу,  
расположенному в  $i$ -той строке и  $j$ -том  
столбце.

## Описание с помощью указателя

```
int *pt_t;
```

\*pt\_t - указатель на тип

```
int
```



```
*pt_t      → t[0][0]  
*(pt_t+1) → t[0][1]  
*(pt_t+2) → t[0][2]  
*(pt_t+3) → t[1][0]  
*(pt_t+4) → t[1][1]
```

**$*(pt\_t+i*m+j) \rightarrow t[i][j]$**

<code>printf("%u", &amp;t);</code>	65498
<code>printf("%u", &amp;t[0]);</code>	65498
<code>printf("%u", &amp;t[0][0]);</code>	65498
<code>printf("%u", &amp;t[0][1]);</code>	65502
<code>printf("%u", &amp;t[0][2]);</code>	65506
<code>printf("%u", &amp;t[1]);</code>	65510
<code>printf("%u", &amp;t[1][0]);</code>	65510
<code>printf("%u", &amp;t[1][1]);</code>	65514
<code>printf("%u", &amp;t[1][2]);</code>	65518

# ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

---

## 1. Функции выделения памяти

`malloc()` (от *memory allocation*, выделение памяти),  
`calloc()` (от *clear allocation*, чистое выделение (памяти))  
`realloc()` (от *reallocation*, перераспределение памяти).  
`free()`  
<stdlib.h>

Например

```
malloc(20*sizeof(int));  
calloc(20, sizeof(int));  
free();
```

Пример. Сформировать массив целых чисел.

Определить адреса элементов массива.

Сформировать массив вещественных чисел.

Определить адреса элементов массива.

```
#include<stdlib.h>
```

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

```
#include<alloc.h>
```



```
void main()  
{  
    int i;  
    int *x=(int*) malloc (5*sizeof(int));  
    for (i=0;i<5;i++)  
    {  
        *(x+i)=int(0.002*rand());  
        printf("\n % 2d", *(x+i));  
        printf("      %u", (x+i));  
    }  
    free(x) ;
```

```
float *r=(float*) calloc (5,sizeof(float));
for (i=0;i<5;i++)
{
*(r+i)=(0.001*rand());
printf("\n %8.2f",*(r+i));
printf(" %u", (r+i));
}
free(r) ;
}
```

## На экране

0 2472

21 2474

14 2476

5 2478

23 2480

7.12 2472

17.56 2476

6.41 2480

22.95 2484

31.13 2488

## 2. Операторы **new** и **delete**

### Выделение памяти

```
<указатель> = new <тип>;
```

```
<указатель> = new <тип> [K] ; -выделение памяти  
для массива из K
```

ЭЛЕМЕНТОВ

### Освобождение памяти

```
delete <указатель> ;
```

```
delete [ ] <указатель> ;
```

Пример.

```
void main()
{
int *x,i;
x=new int[10];
for (i=0;i<5;i++)
{
*(x+i)=12+int(random(50));
printf("\n % 2d",*(x+i));
printf("      %u", (x+i));
}
delete[]x;
```

```
float *a;
a=new float[10];
for (i=0;i<5;i++)
{
*(a+i)=0.01*rand();
printf("\n % .2f",*(a+i));
printf("  %u", (a+i));
}
delete[]a;
```

На экране

25 3734

12 3736

15 3738

47 3740

20 3742

227.37 3734

123.45 3738

65.37 3742

345.34 3746

47.56 3750