

# Программирование

## 2 | Указатели и массивы

13 февраля 2021

# Указатель

## Адрес

- Номер ячейки памяти

# Указатель

## Адрес

- Номер ячейки памяти

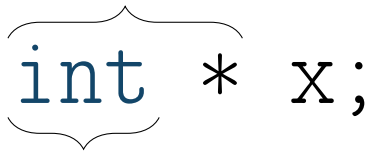
## Указатель

- Адрес
- Указывает (ссылается) на значение в памяти
- Заданного типа
- Взятие, разыменованение, арифметика

# Типы указателей

## Указатель на int

тип указателя

  
`int * x;`

тип значения

на которое указывает

# Типы указателей

## Указатель на указатель на int

тип указателя

$\underbrace{\text{int}}_{\text{тип значения}}$   $**$   $x;$

тип значения  
на которое указывает

# Операции с указателями

## Взятие указателя

(Address-of)

```
int x;
```

```
int * p = &x;
```

# Операции с указателями

## Взятие указателя

(Address-of)

```
int x;  
int * p = &x;
```

## Разыменование

(Dereference)

```
int * p;  
int x = *p;
```

# Операции с указателями

## Взятие указателя

(Address-of)

```
int x;  
int * p = &x;
```

## Разыменование

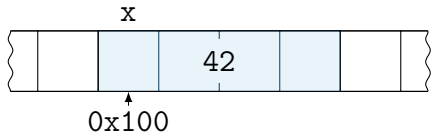
(Dereference)

```
int * p;  
int x = *p;
```

$*(&x) == x;$

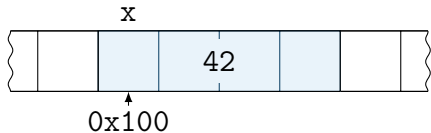


# Операции с указателями



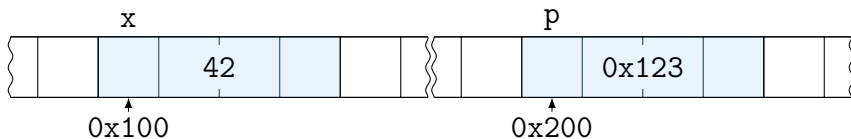
```
int x = 42;
```

# Операции с указателями



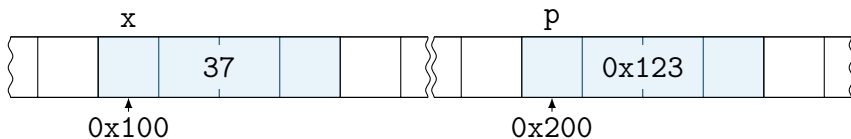
```
int x = 42;  
int * p = &x;
```

# Операции с указателями



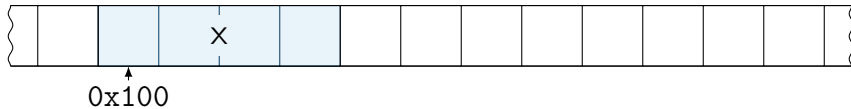
```
int x = 42;  
int * p = &x;
```

# Операции с указателями



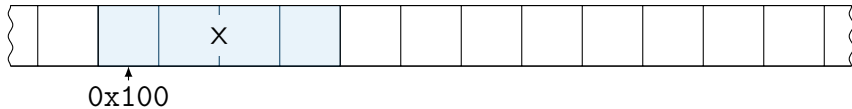
```
int x = 42;  
int * p = &x;  
*p = 37;
```

# Адресная арифметика



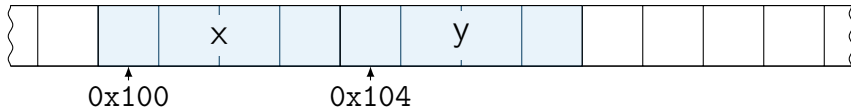
```
int x;
```

# Адресная арифметика



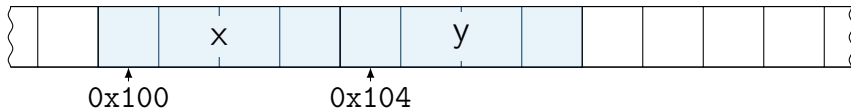
```
int x;  
int * p = &x;
```

# Адресная арифметика



```
int x;  
int * p = &x;  
int * s = p + 1;
```

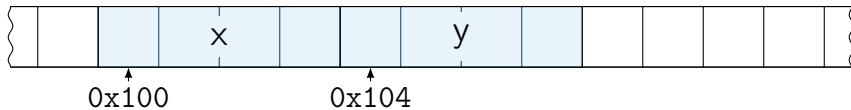
# Адресная арифметика



```
int x;  
int * p = &x;  
int * s = p + 1;  
long int diff = s-p;
```



# Адресная арифметика



```
int x;  
int * p = &x;  
int * s = p + 1;  
long int diff = s-p; // == 1
```

# Адресная арифметика

Для указателей определены операции:

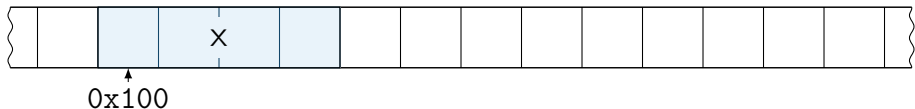
- ➊ Добавление (вычитание) целого числа
- ➋ Сравнение с другим указателем
- ➌ Вычитание указателей

# Массив

Массив – упорядоченный набор элементов одного типа.

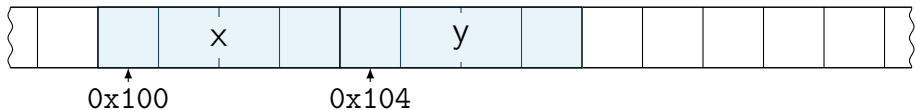
# Массив

Массив – упорядоченный набор элементов одного типа.



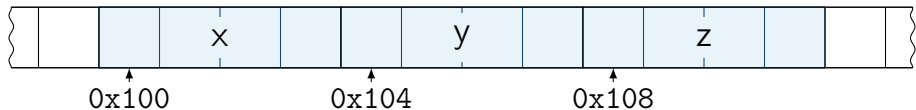
# Массив

Массив – упорядоченный набор элементов одного типа.



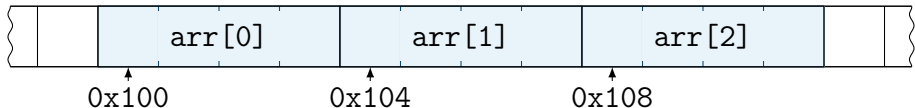
# Массив

Массив – упорядоченный набор элементов одного типа.



# Массив

Массив – упорядоченный набор элементов одного типа.

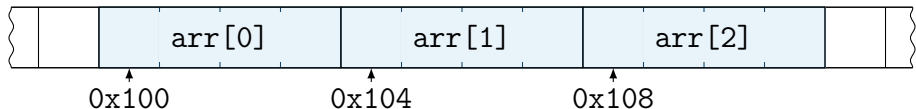


```
int arr[3];
```

тип элементов                      число элементов  
  (константа)

# Массив

Массив – упорядоченный набор элементов одного типа.



Запись:

```
arr[i] = 42;
```

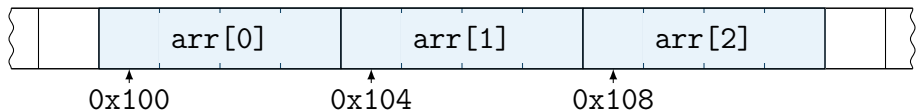
Чтение:

```
int x = arr[i];
```



# Массив

Массив – упорядоченный набор элементов одного типа.



Запись:

```
arr[i] = 42;  
*(arr + i) = 42;
```

Чтение:

```
int x = arr[i];  
int x = *(arr + i);
```

# Объявление массива

- Имена одномерных массивов в C — неизменяемые указатели на первый элемент

# Объявление массива

- Имена одномерных массивов в C — неизменяемые указатели на первый элемент
- Размер статического массива — константа

# Объявление массива

- Имена одномерных массивов в C — неизменяемые указатели на первый элемент
- Размер статического массива — константа
  - Целое число

```
int arr[42];
```

# Объявление массива

- Имена одномерных массивов в С — неизменяемые указатели на первый элемент
- Размер статического массива — константа

- Целое число

```
int arr[42];
```

- Константа препроцессора

```
#define ARR_SIZE 42  
int arr[ARR_SIZE];
```

# Инициализация массива

- Без инициализации

```
int arr[4]; // {?, ?, ?, ?}
```

# Инициализация массива

- Без инициализации

```
int arr[4]; // {?, ?, ?, ?}
```

- С выводом размера

```
int arr[] = {4, 8}; // {4, 8}
```

# Инициализация массива

- Без инициализации

```
int arr[4]; // {?, ?, ?, ?}
```

- С выводом размера

```
int arr[] = {4, 8}; // {4, 8}
```

- Частичная инициализация

```
int arr[4] = {15, 16}; // {15, 16, ?, ?}
```



# Инициализация массива

- Без инициализации

```
int arr[4]; // {?, ?, ?, ?}
```

- С выводом размера

```
int arr[] = {4, 8}; // {4, 8}
```

- Частичная инициализация

```
int arr[4] = {15, 16}; // {15, 16, ?, ?}
```

- Инициализация строкой

```
char arr[] = "Hello"; // {'H', 'e', 'l', 'l', 'o', 0}
```

# Двумерные массивы

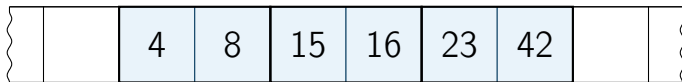
```
#define N 3
#define M 2

int main() {
    int arr[N][M] = {{4, 8}, {15, 16}, {23, 42}};
    ...
}
```

# Двумерные массивы

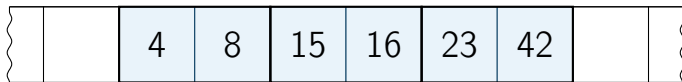
```
#define N 3
#define M 2

int main() {
    int arr[N][M] = {{4, 8}, {15, 16}, {23, 42}};
    ...
}
```



# Двумерные массивы

```
int main() {  
    int arr[N][M] = {{4, 8}, {15, 16}, {23, 42}};  
  
    arr[0][0] == 4  
    arr[0][1] == 8  
    arr[2][1] == 42  
}
```



Q & A