

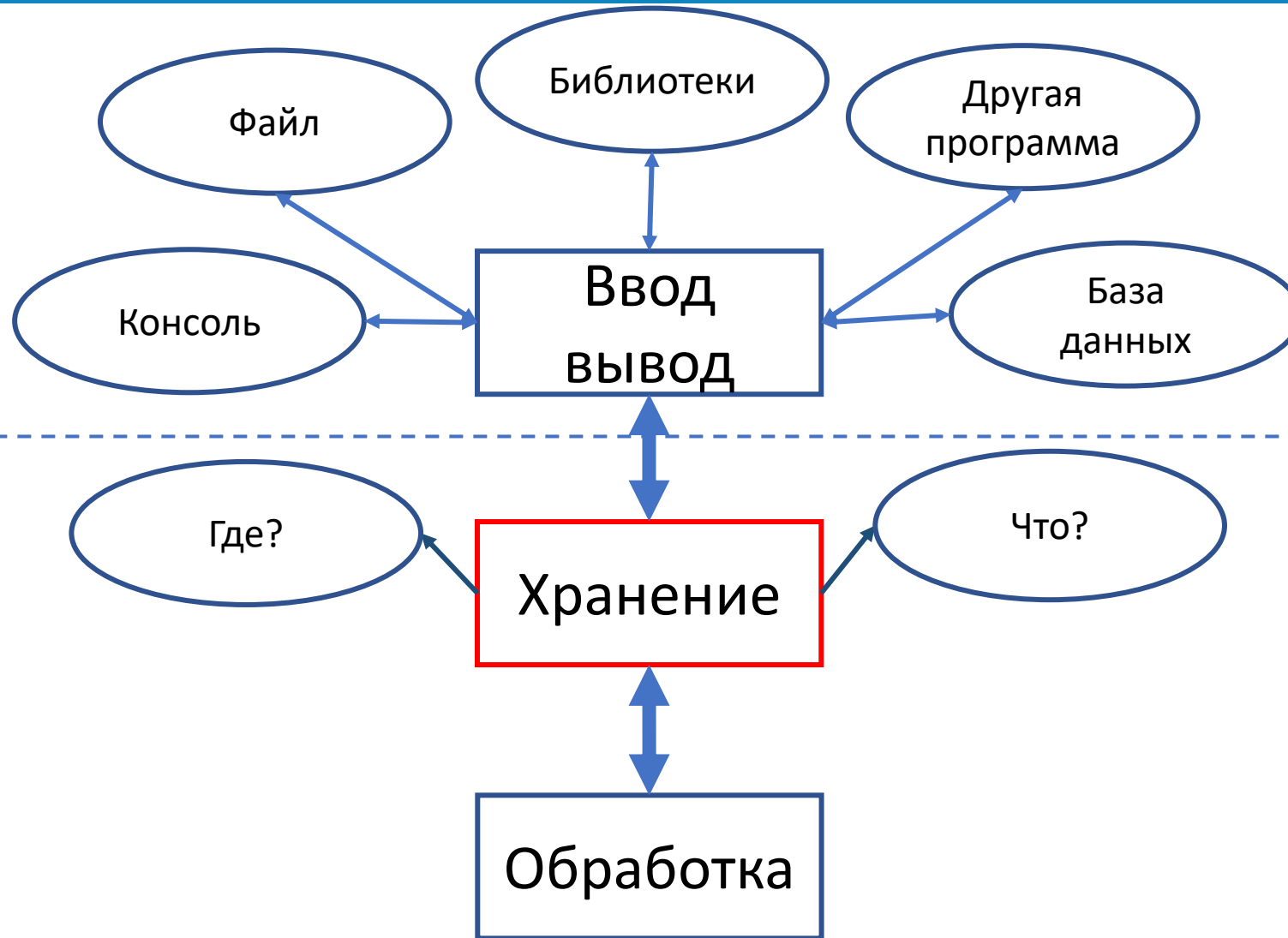


Основы программирования на C++

Занятие 5. Указатели



Дерево языка





Предисловие

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

```
void swap(int a, int b)
{
    int tmp = b;
    b = a;
    a = tmp;
}
```

```
int main()
{
    int a = 10, b = 15;
    printf("a = %d, b = %d\n", a, b);
    swap(a, b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```



Предисловие

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

```
void swap(int a, int b)
{
    int tmp = b;
    b = a;
    a = tmp;
}
```

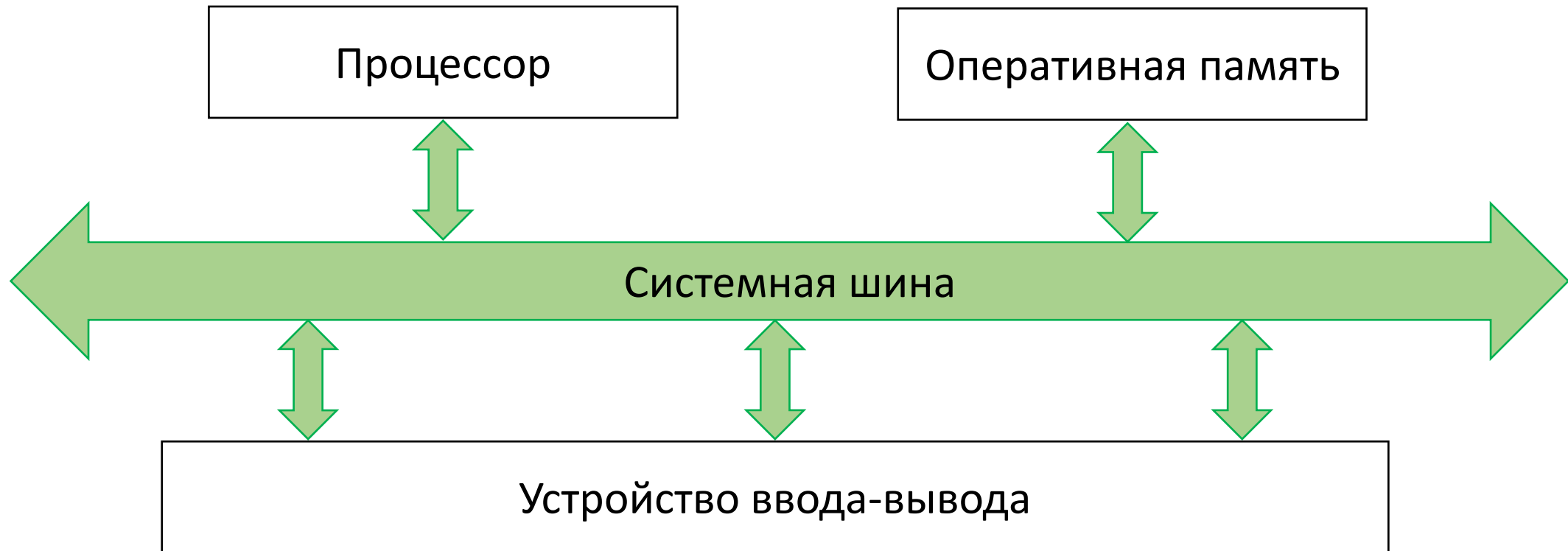
```
int main()
{
    int a = 10, b = 15;
    printf("a = %d, b = %d\n", a, b);
    swap(a, b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Результат

```
a = 10, b = 15
a = 10, b = 15
```

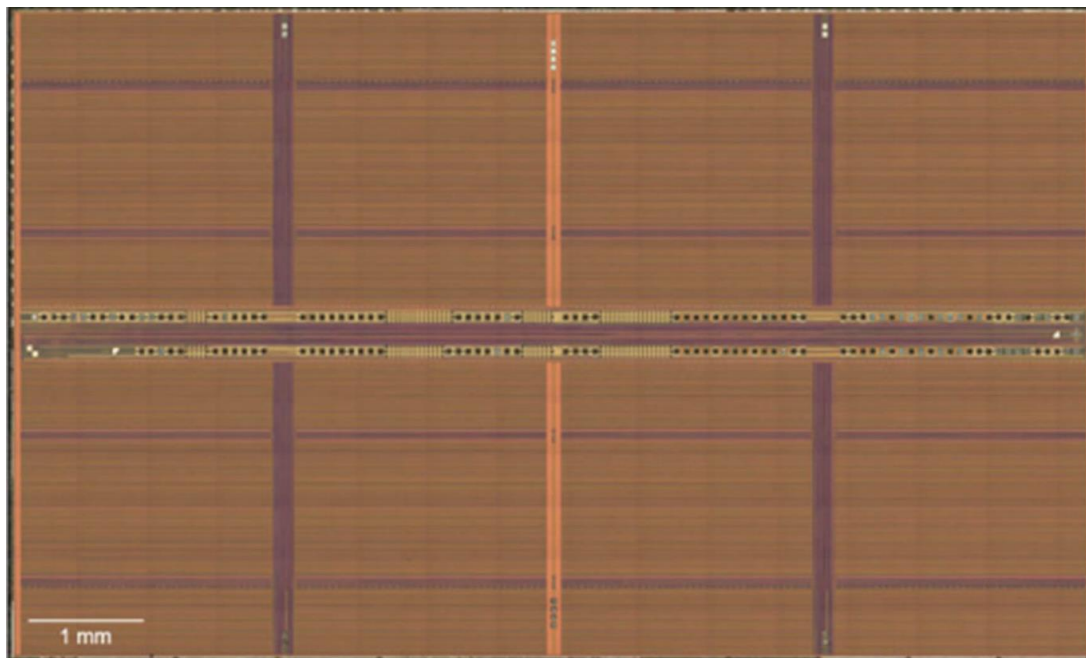


Схема организации ЭВМ

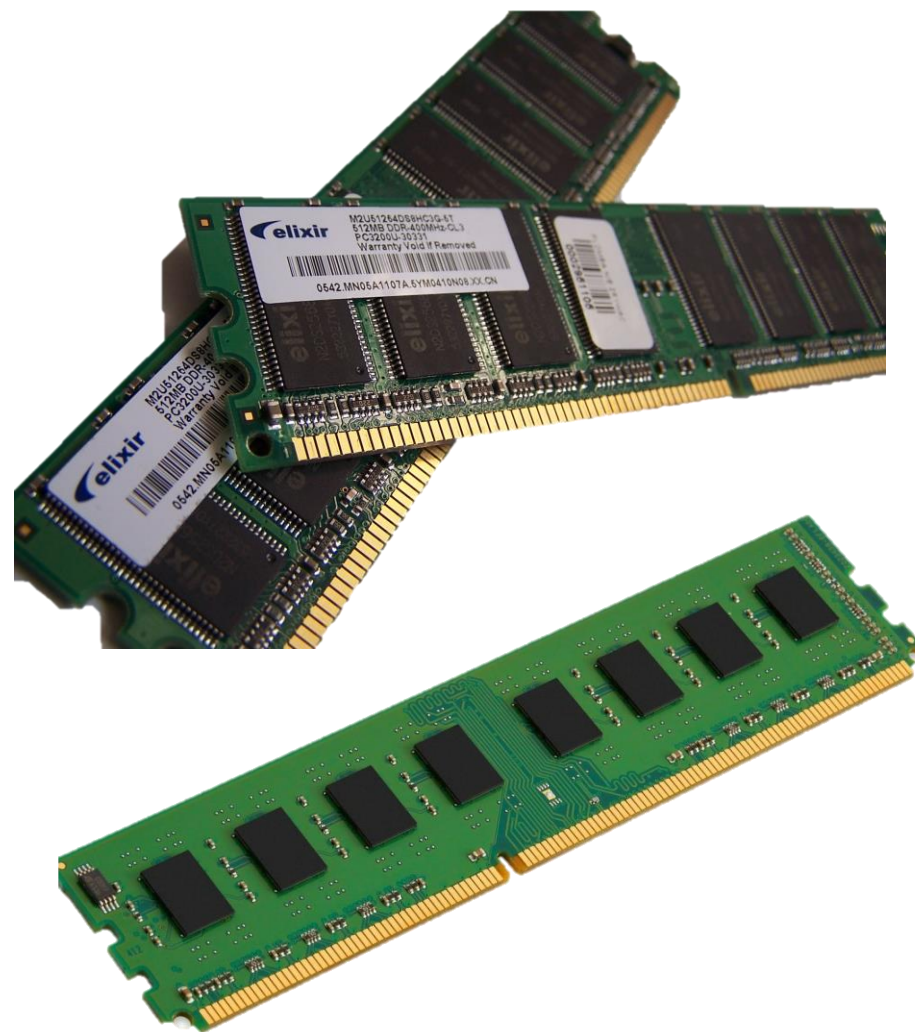




Оперативная память

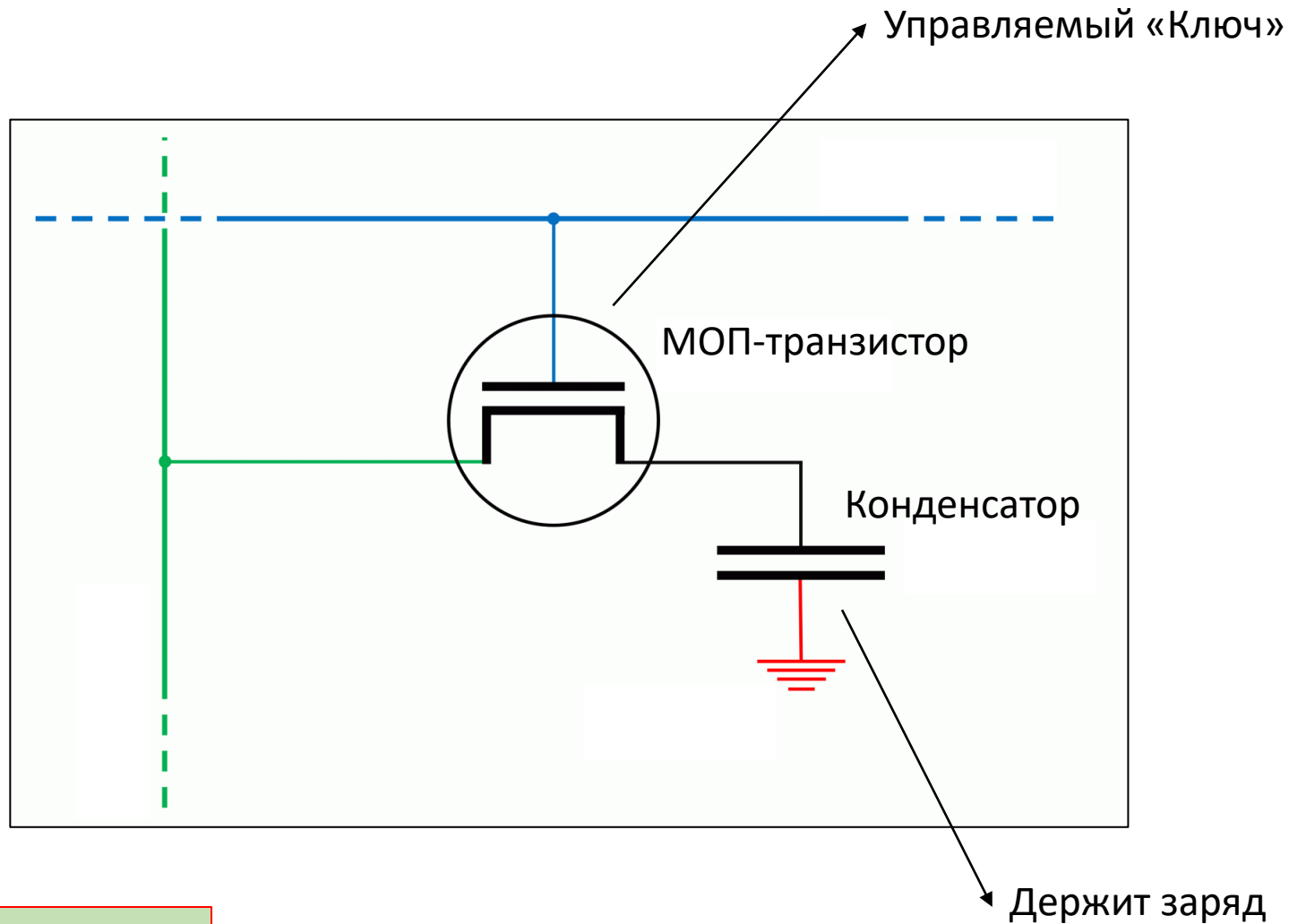
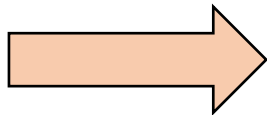
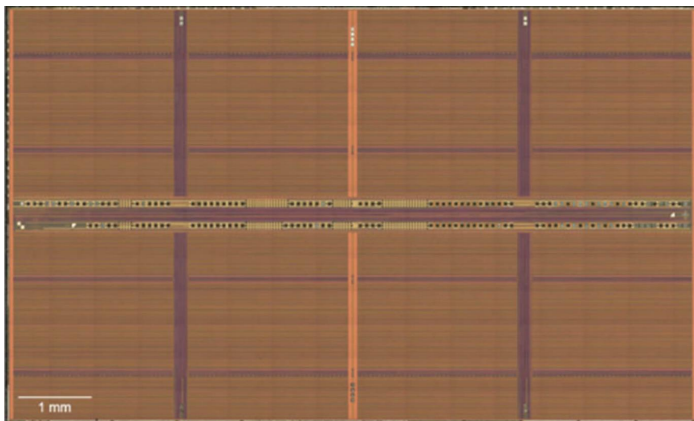


Чип DRAM под микроскопом





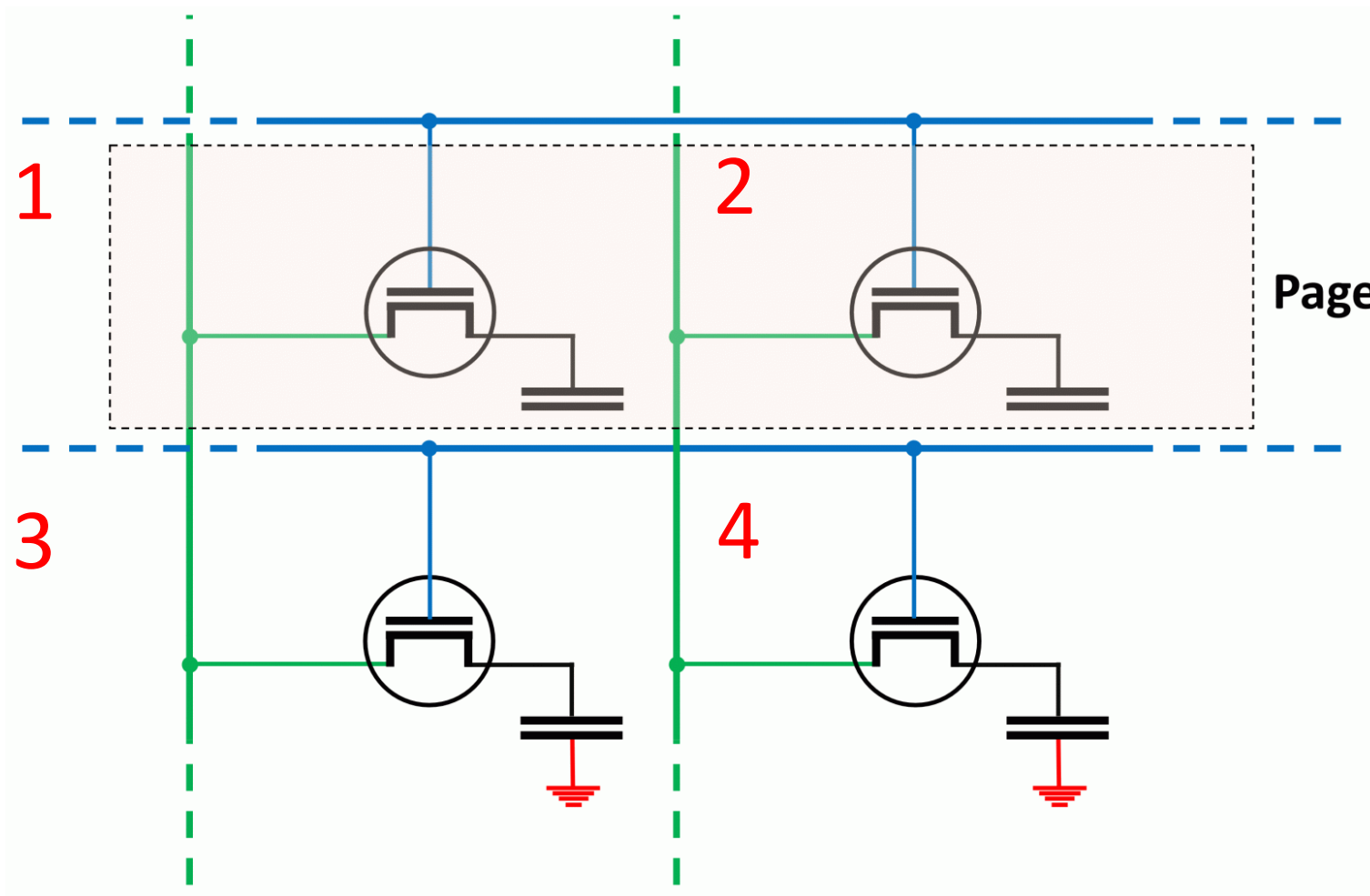
Оперативная память



Конденсаторы не способны хранить заряд вечно и каждую ячейку памяти нужно обновлять по 15-30 раз в секунду.



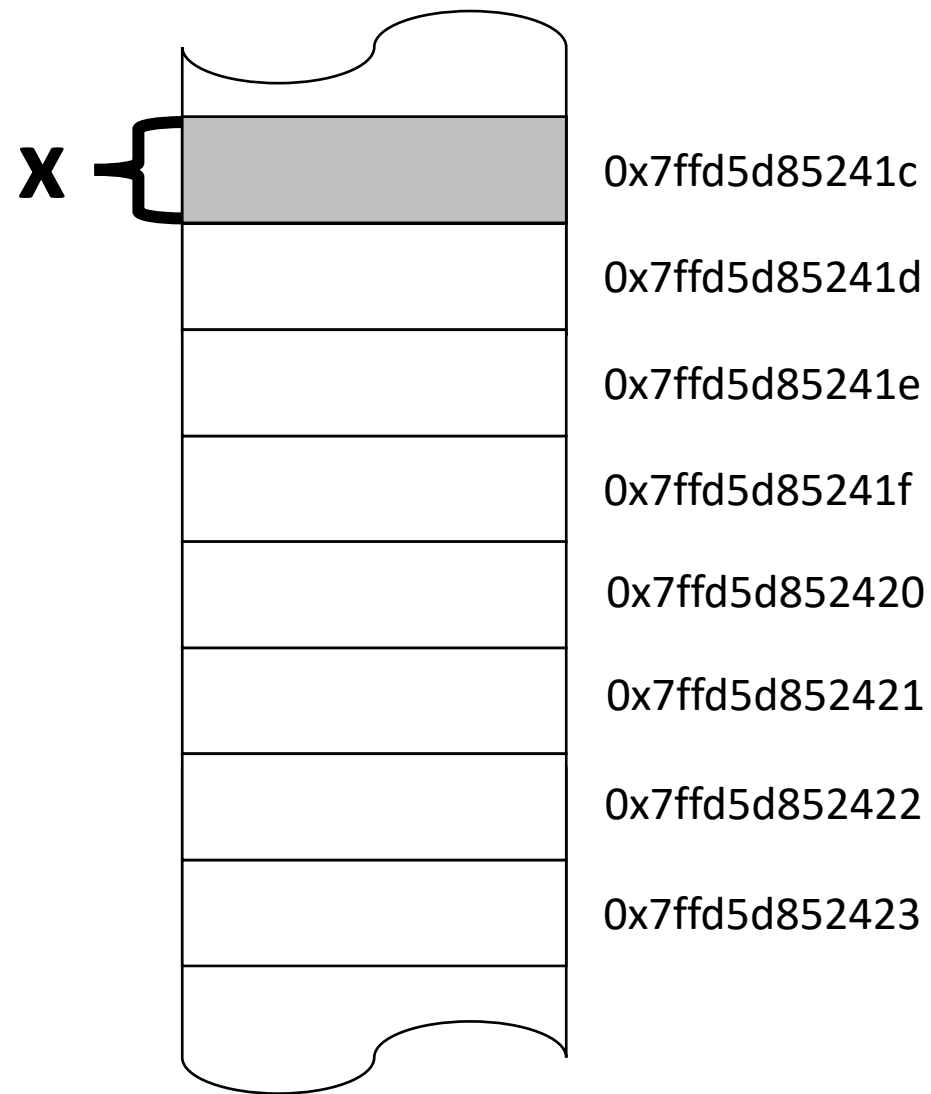
Оперативная память





Адреса

char **x**



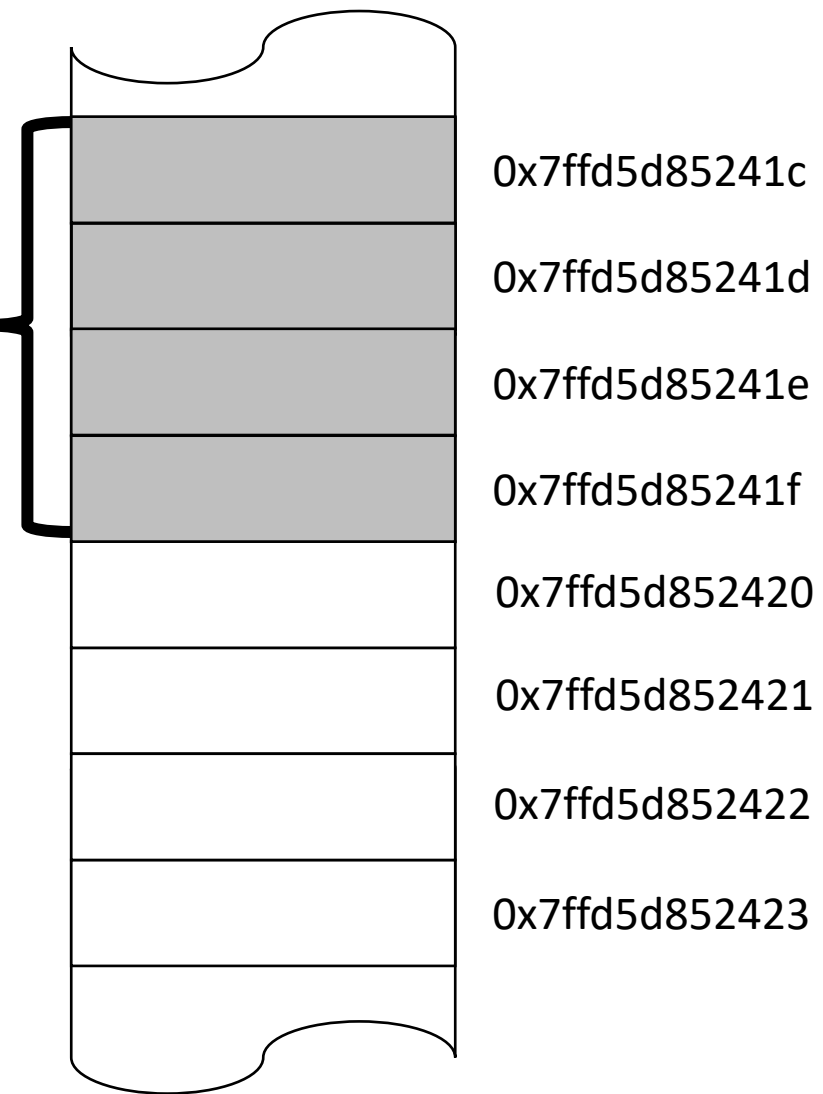


Адреса

int x



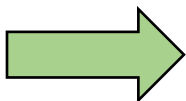
x



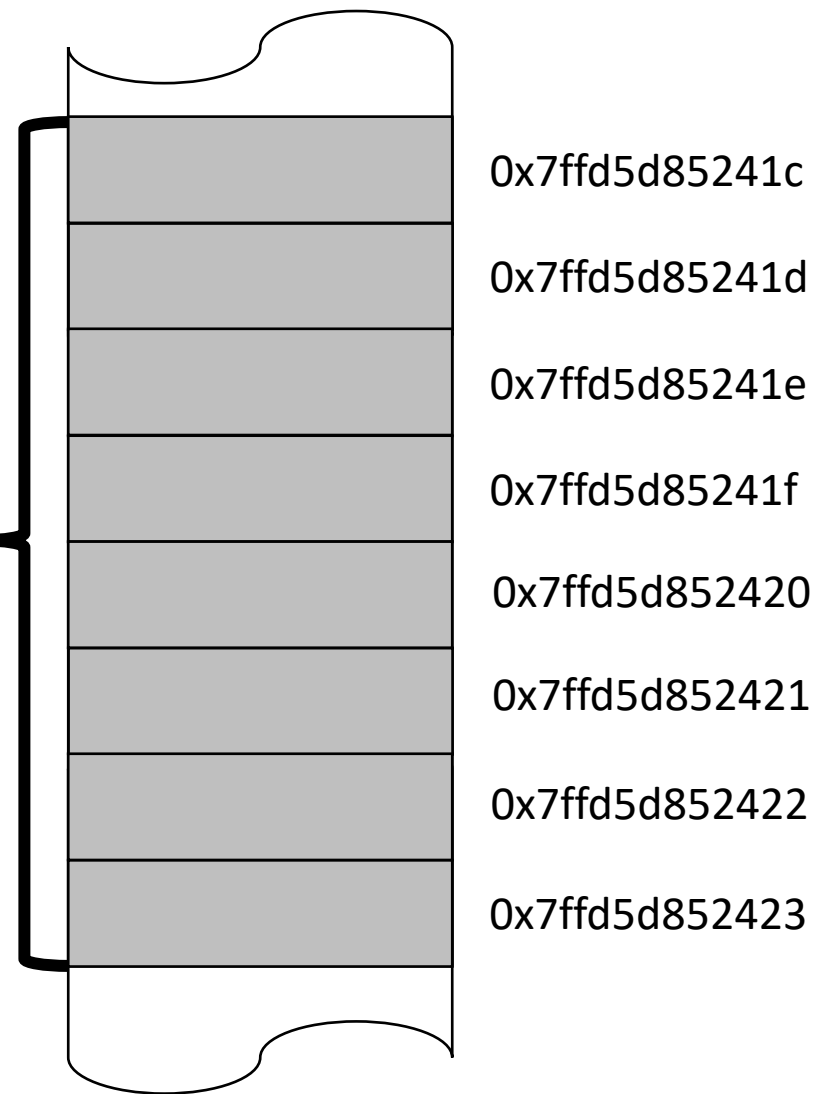


Адреса

double **x**







x



Как хранить адреса?



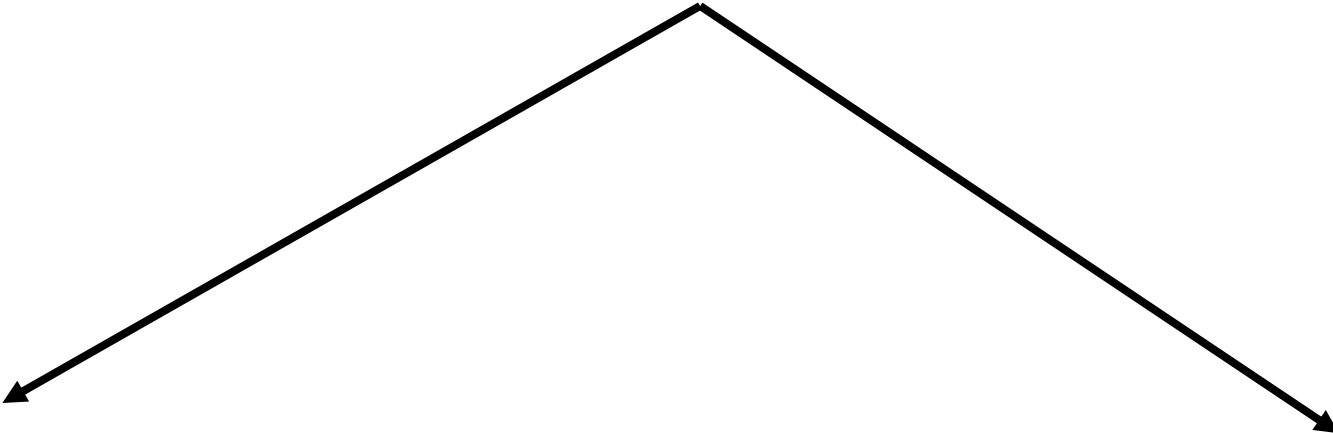
Указатели

char		храним 'char'
int		храним 'int'
double		храним 'double'
int*		храним адрес 'int'

Указатель - это адрес переменной в памяти.



Операции с указателями



Взятие адреса
 $\&x$

Разыменование
 $*x$



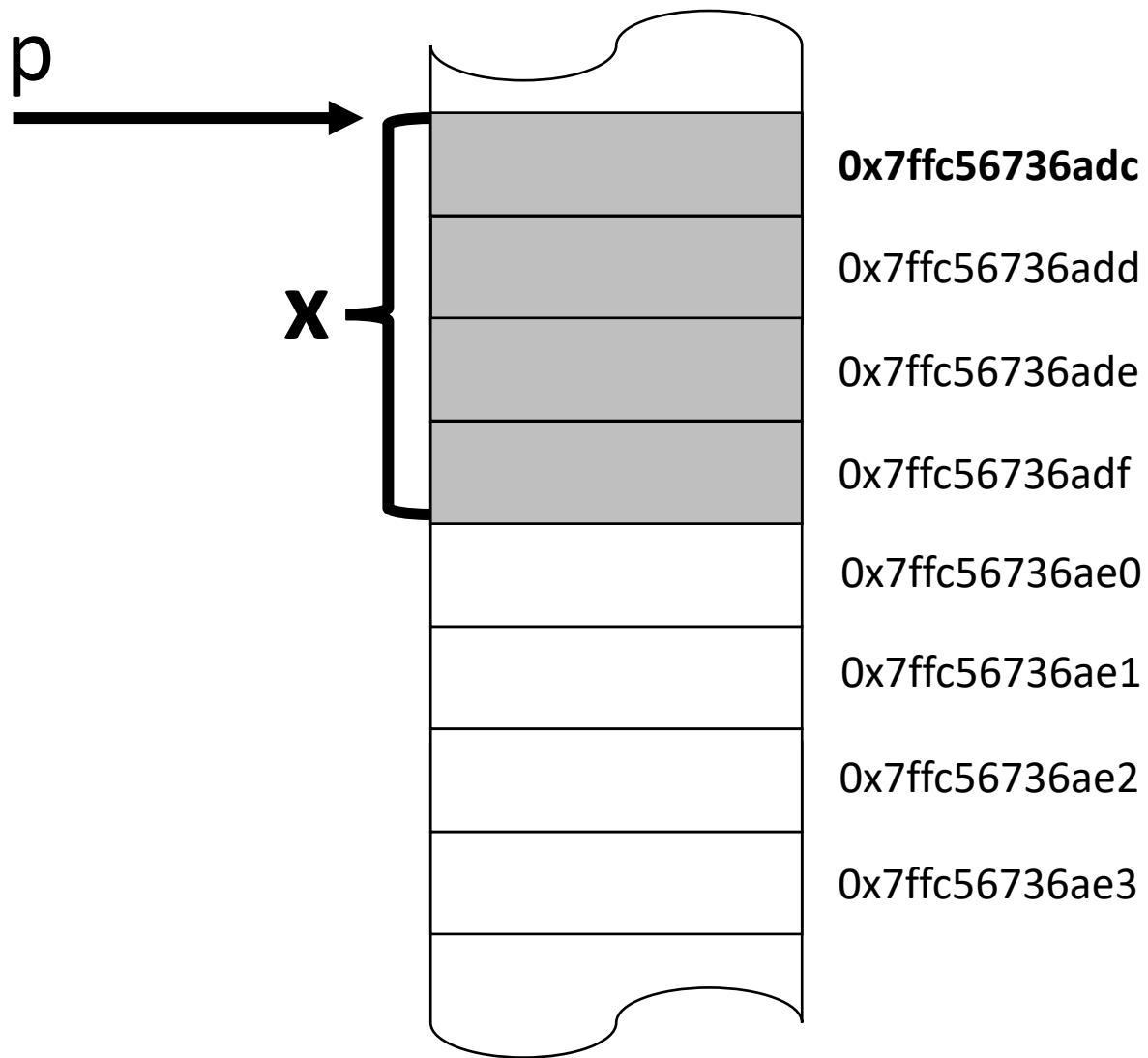
Указатели

```
int x = 10;  
int* p;  
p = &x;  
printf("%p\n", p);  
printf("%d\n", *p);  
printf("%p\n", &p);
```

p=0x7ffc56736adc

*p=10

&p=0x7ffc56736ad0

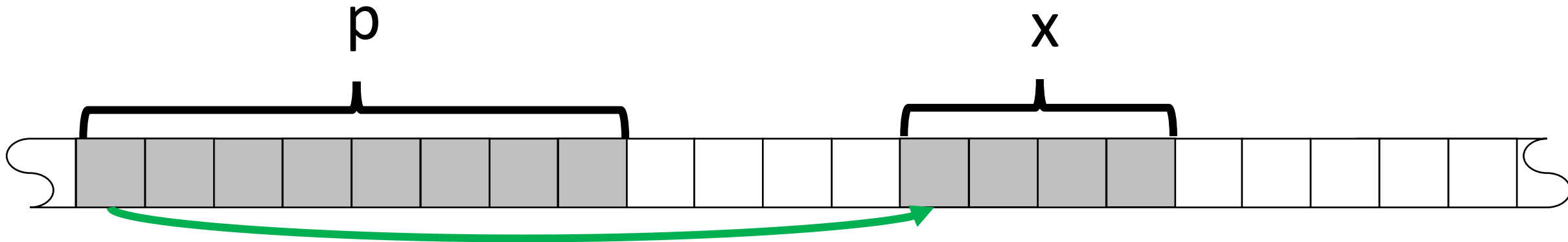


Чему равен размер указателя?



Указатели

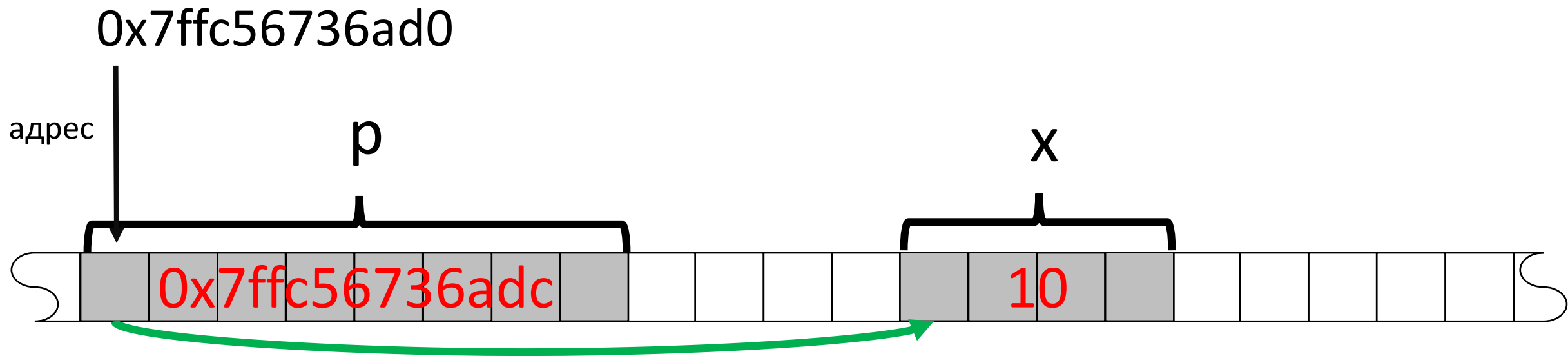
```
int x = 10;  
int* p;  
p = &x;
```





Указатели

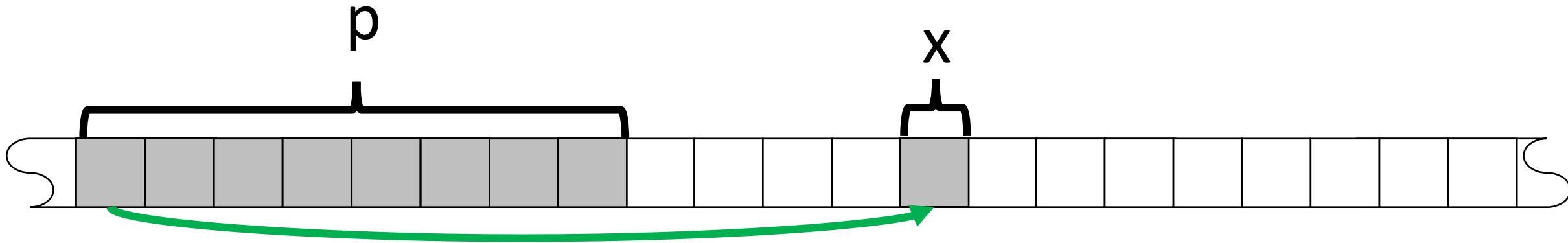
```
int x = 10;  
int* p;  
p = &x;
```





Указатели

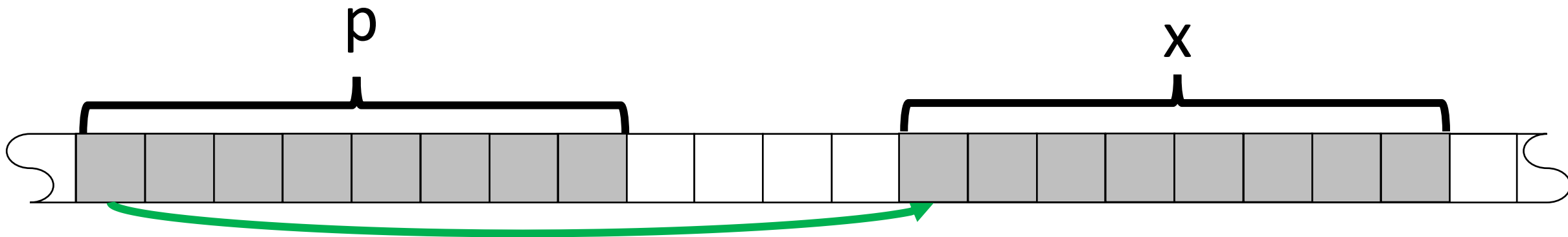
```
char x = 'A';  
char* p;  
p = &x;
```





Указатели

```
double x = 12.345678;  
double* p;  
p = &x;
```





Указатели

```
#include <stdio.h>
int main()
{
    int x = 10;
    int* y = &x;
    printf("x = %d\n", x);
    *y = *y + 1;
    printf("x = %d\n", x);
    return 0;
}
```

Что напечатает данная программа?



Указатели

```
#include <stdio.h>
int main()
{
    int x = 10;
    int* y = &x;
    printf("x = %d\n", x);
    *y = *y + 1;
    printf("x = %d\n", x);
    return 0;
}
```

Что напечатает данная программа?

```
x = 10
x = 11
```



Адресная арифметика

Сложение: (адрес в указателе) + (значение int_выражения)*sizeof(<тип>)

```
int x = 10;  
int* ptr;  
ptr = &x;  
ptr = ptr + 1;
```

Возвращаемое значение – адрес!



Адресная арифметика

Сложение: (адрес в указателе) + (значение int_выражения)*sizeof(<тип>)

```
int x = 10;  
int* ptr;  
ptr = &x;           //ptr = 0x0000002ABDBAF974  
ptr = ptr + 1;       //ptr = 0x0000002ABDBAF978
```

$$0x0000002ABDBAF978 = 0x0000002ABDBAF974 + 1 * 4$$



Адресная арифметика

Вычитание: (адрес в указателе) - (значение int_выражения)*sizeof(<тип>)

```
int x = 10;  
int* ptr;  
ptr = &x; //ptr = 0x0000003E8D2FF834  
ptr = ptr - 1; //ptr = 0x0000003E8D2FF830
```

$0x0000003E8D2FF830 = 0x0000003E8D2FF834 - 1*4$

Возвращаемое значение – адрес!



Адресная арифметика

Индексация: $*((\text{адрес в указателе}) + (\text{значение индекса}) * \text{sizeof}(<\text{тип}>))$

```
int x = 10;  
int* ptr;  
ptr = &x;           //ptr = 0x0000001F33AFFB24  
ptr = &ptr[1];      //ptr = 0x0000001F33AFFB28
```

1. ЧИСЛО = $*(0x0000001F33AFFB24 + 1 * 4)$

2. $0x0000001F33AFFB28 = \&(\text{ЧИСЛО})$

Напомнила ли вам что-нибудь
подобная запись?

Возвращаемое значение – значение по адресу!



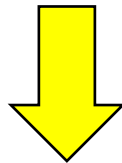
Указатели и массивы

Заметим!

```
ptr = &ptr[1];
```



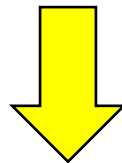
```
ptr = ptr + 1;
```



```
int x = ptr[1];
```



```
int x = *(ptr + 1);
```



Массивы

?

Указатели



Указатели и массивы

```
#include <stdio.h>
int main()
{
    int x[5];
    for(int i = 0; i < 5; i++)
        x[i] = i + 1;

    for (int i = 0; i < 5; i++)
        printf("%d ", x[i]);
    printf("\n");

    x[2] = 10;

    for (int i = 0; i < 5; i++)
        printf("%d ", x[i]);
    printf("\n");

    *(x + 2) = 3;
    for (int i = 0; i < 5; i++)
        printf("%d ", *(x + i));

    return 0;
}
```



Указатели и массивы

```
#include <stdio.h>
int main()
{
    int x[5];
    for(int i = 0; i < 5; i++)
        x[i] = i + 1;

    for (int i = 0; i < 5; i++)
        printf("%d ", x[i]);
    printf("\n");

    x[2] = 10;

    for (int i = 0; i < 5; i++)
        printf("%d ", x[i]);
    printf("\n");

    *(x + 2) = 3;
    for (int i = 0; i < 5; i++)
        printf("%d ", *(x + i));

    return 0;
}
```

Результат

1	2	3	4	5
1	2	10	4	5
1	2	3	4	5



The Ksplice Pointer Challenge

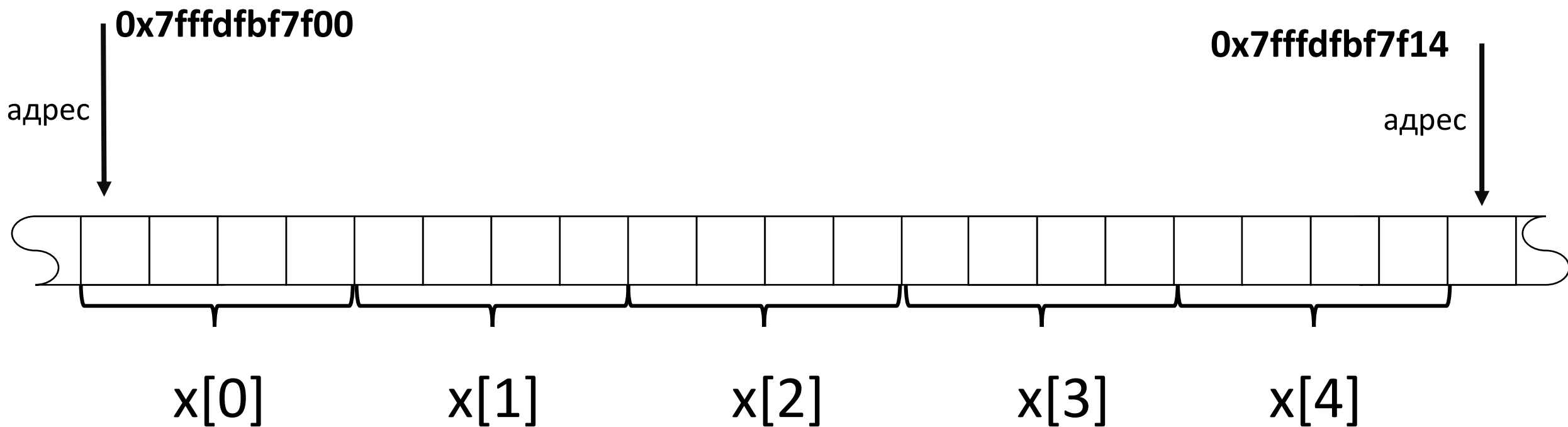
```
#include <stdio.h>
```

```
int main()
{
    int x[5];
    printf("x=\t%p\n", x);
    printf("x+1=\t%p\n", x + 1);
    printf("&x=\t%p\n", &x);
    printf("&x+1\t%p\n", &x + 1);
    return 0;
}
```

Что напечатает данная программа?



The Ksplice Pointer Challenge





The Ksplice Pointer Challenge

```
#include <stdio.h>

int main()
{
    int x[5];
    printf("x=\t%p\n",      x);
    printf("x+1=\t%p\n",    x + 1);
    printf("&x=\t%p\n",      &x);
    printf("&x+1\t%p\n",    &x + 1);
    return 0;
}
```



The Ksplice Pointer Challenge

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

```
int main()  
{
```

```
    int x[5];
```

```
    printf("x=\t%p\n", x);
```

```
    printf("x+1=\t%p\n", x + 1);
```

```
    printf("&x=\t%p\n", &x);
```

```
    printf("&x+1\t%p\n", &x + 1);
```

```
    return 0;
```

```
}
```

$x + 0 * \text{sizeof}(\text{int})$

$x + 1 * \text{sizeof}(\text{int})$

$\&x[0]$

$x + 1 * \text{sizeof}(x)$

Работает как
указатель

Работает как
массив



The Ksplice Pointer Challenge

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

```
int main()  
{
```

```
    int x[5];
```

```
    printf("x=\t%p\n", x);
```

```
    printf("x+1=\t%p\n", x + 1);
```

```
    printf("&x=\t%p\n", &x);
```

```
    printf("&x+1\t%p\n", &x + 1);
```

```
    return 0;
```

```
}
```

```
x=          000000139ACFFB48  
x+1=        000000139ACFFB4C  
&x=         000000139ACFFB48  
&x+1        000000139ACFFB5C
```




Указатели и массивы

```
#include <stdio.h>
int main()
{
    int x[5] = {1,2,3,4,5};
    int* ptr;
    ptr = x;
    printf("%p\n", x);
    printf("%p\n", &x);
    printf("%p\n", ptr);
    printf("%p\n", &ptr);
    return 0;
}
```

```
000000B412AFFAF8
000000B412AFFAF8
000000B412AFFAF8
000000B412AFFB28
```



Решение проблемы

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

```
void swap(int *a, int *b)
{
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
int main()
{
    int a = 10, b = 15;
    printf("a = %d, b = %d\n", a, b);
    swap(&a, &b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Результат

```
a = 10, b = 15
a = 15, b = 10
```



Задача

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
const int N = 5;

void input_array(int x[])
{
    for (int i = 0; i < N; i++)
        x[i] = rand()%100;
}

void print_array(int *x)
{
    for (int i = 0; i < 5; i++)
        printf("%d ", x[i]);
    printf("\n");
}

int main()
{
    srand(time(NULL));
    int x[5];
    input_array(x);
    print_array(x);
    return 0;
}
```



Домашнее задание 5

Доработать программу калькулятор следующим образом:

1. Ввод и вывод матриц из задания 4 осуществить с помощью отдельных функций. Сами матрицы должны быть определены в функции `main` и переданы в функции по указателю.
2. Добавить функцию поиска минимума и максимума в матрице.
3. Разработать отдельную функцию для вывода результата вычисления

Срок выполнения: 28.03.22