НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Алгоритмизация и программирование

Программирование на C/C++

(ч.12 – указатели)

Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

Указатели

- Указатель – переменная, значением которой является адрес ячейки памяти.
- Указатель ссылается на блок данных из области памяти, причём на самое его начало.
- Указатель может ссылаться на переменную или функцию.
- Для этого нужно знать адрес переменной или функции.

Указатели

- Чтобы узнать адрес конкретной переменной в C++ **существует унарная операция взятия адреса &**.
- Такая операция извлекает адрес объявленных переменных, для того, чтобы его присвоить указателю.
- **Указатели используются для передачи** по ссылке **данных**, что намного ускоряет процесс обработки этих данных (в том случае, если объём данных большой), **так как их не надо копировать**, как при передаче по значению, то есть, используя имя переменной.

Указатели

- В основном **указатели используются для** организации **динамического распределения памяти**, например при объявлении массива, не надо будет его ограничивать в размере.
- Если **программист заранее не может знать, какого размера нужен массив** тому или иному пользователю, в таком случае используется динамическое выделение памяти под массив.
- Любой **указатель необходимо объявить перед использованием**, как и любую переменную.

Указатели

```
//объявление указателя
/*тип данных*/  * /*имя указателя*/;
```

- Принцип объявления указателей такой же, как и принцип объявления переменных.
- Отличие заключается только в том, что перед именем ставится символ звёздочки *.
- Визуально указатели отличаются от переменных только одним символом.
- Чтобы получить значение, на которое ссылается указатель, нужно воспользоваться операцией разыменования указателя *.

Указатели

Указатели можно сравнивать, причём не, только на равенство или неравенство, ведь адреса могут быть меньше или больше относительно друг друга.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Указатели

```
#include <iostream>
using namespace std;

int main()
{
    int var1 = 123;
    int var2 = 99;
    int *ptrvar1 = &var1;
    int *ptrvar2 = &var2;
    cout << "var1    = " << var1 << endl;
    cout << "var2    = " << var2 << endl;
    cout << "ptrvar1 = " << ptrvar1 << endl;
    cout << "ptrvar2 = " << ptrvar2 << endl;
    if (ptrvar1 > ptrvar2)
        cout << "ptrvar1 > ptrvar2" << endl;
    if (*ptrvar1 > *ptrvar2)
        cout << "*ptrvar1 > *ptrvar2" << endl;
    return 0;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Указатели

Указатели на указатели

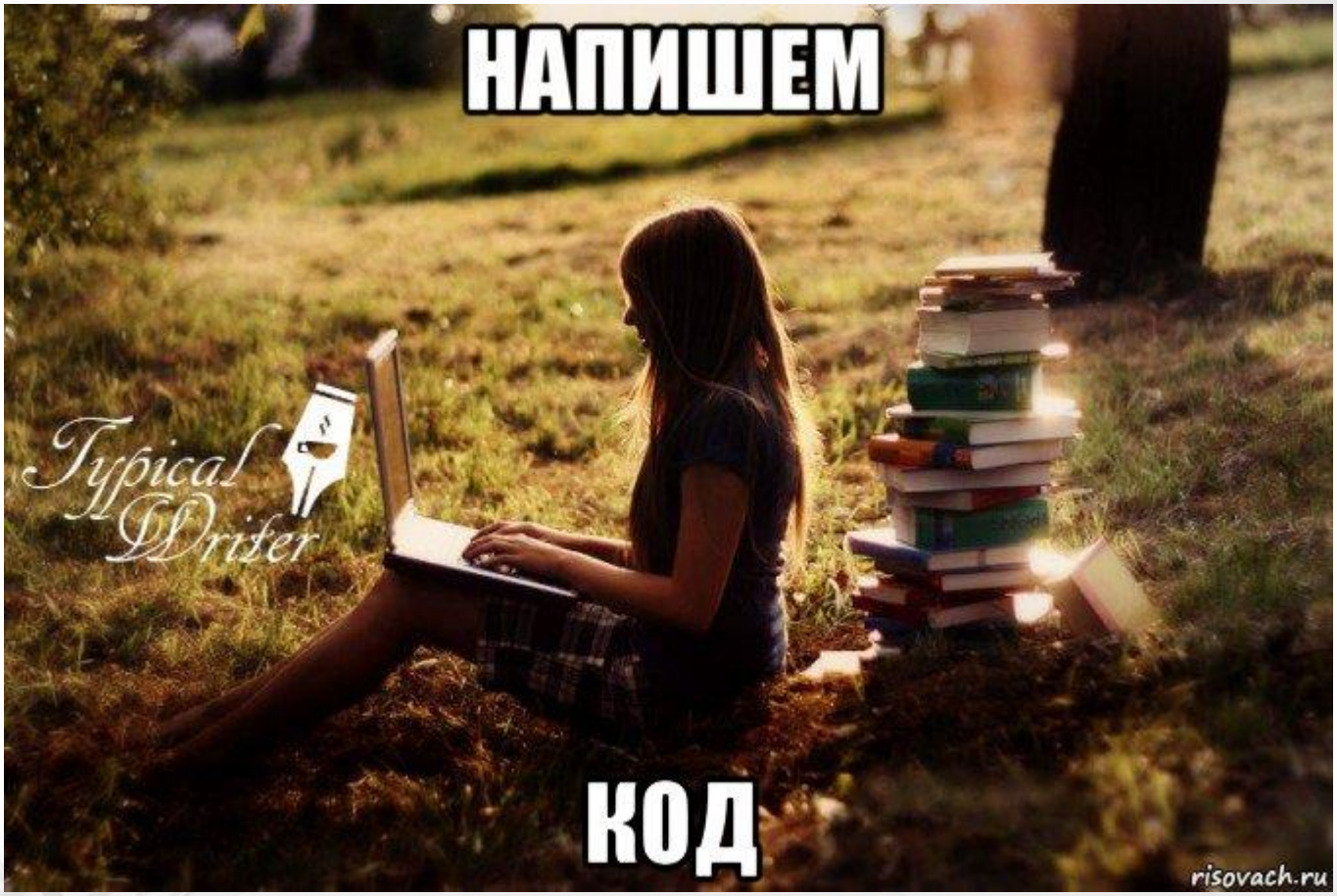
- Указатели могут ссылаться на другие указатели.
- При этом в ячейках памяти, на которые будут ссылаться первые указатели, будут содержаться не значения, а адреса вторых указателей.
- Число символов * при объявлении указателя показывает порядок указателя.
- Чтобы получить доступ к значению, на которое ссылается указатель его необходимо разыменовывать соответствующее количество раз.
- Разработаем программу, которая будет выполнять некоторые операции с указателями порядка выше первого.

Указатели

```
#include <iostream>
using namespace std;

int main()
{
    int var = 123;
    int *ptrvar = &var;
    int **ptr_ptrvar = &ptrvar;
    int ***ptr_ptr_ptrvar = &ptr_ptrvar;
    cout << " var\t\t= " << var << endl;
    cout << " *ptrvar\t= " << *ptrvar << endl;
    cout << " **ptr_ptrvar   = " << **ptr_ptrvar << endl;
    cout << " ***ptr_ptrvar  = "
         << ***ptr_ptr_ptrvar << endl;
    cout << "\n***ptr_ptr_ptrvar -> **ptr_ptrvar -> "
         << "*ptrvar -> var -> "<< var << endl;
    cout << "\t  " << &ptr_ptr_ptrvar<< " -> " << "    "
         << &ptr_ptrvar << " ->" << &ptrvar
         << " -> " << &var << " -> " << var << endl;
    return 0;
}
```

Указатели

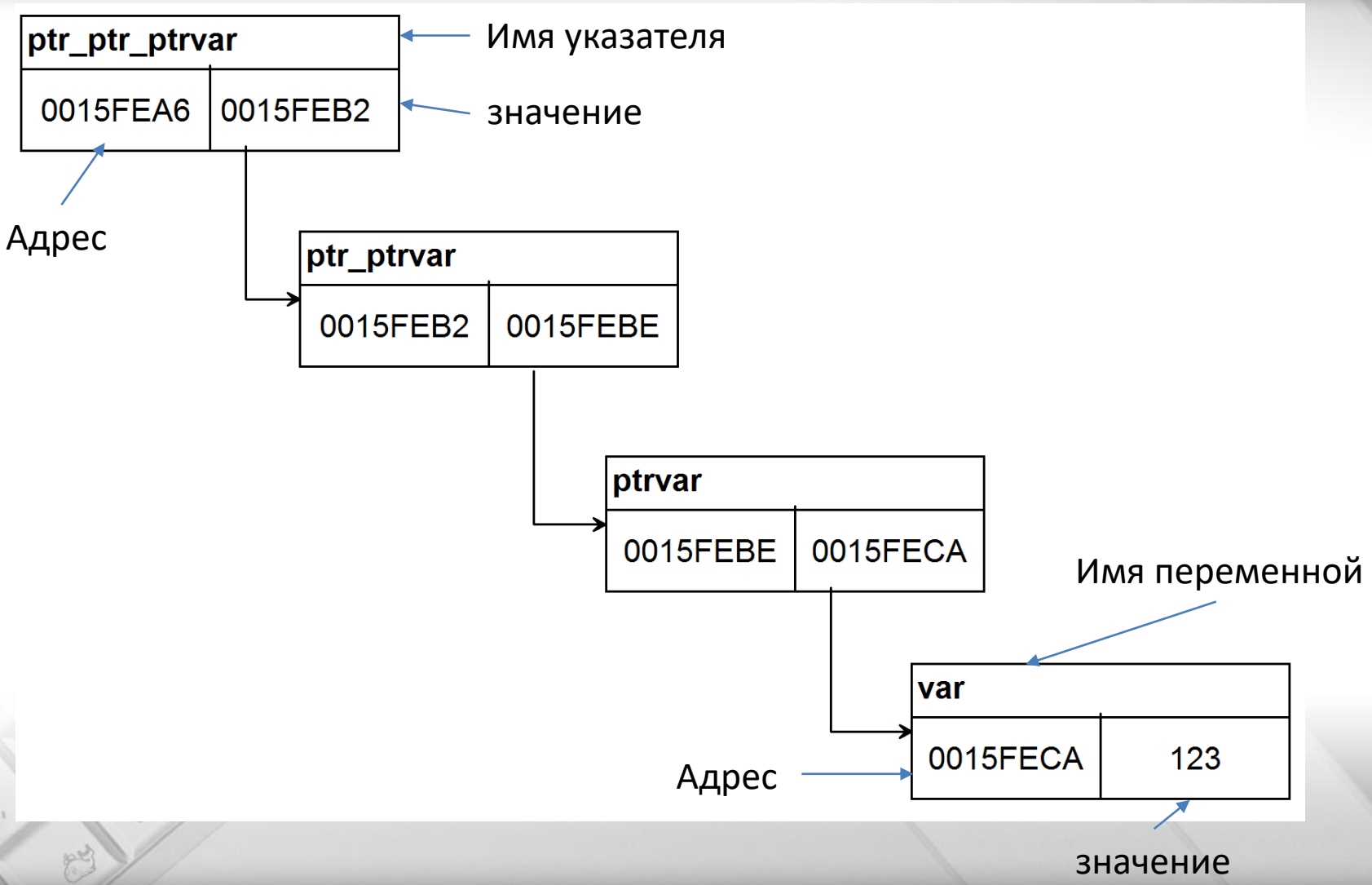


Указатели на указатели

- Данная программа доказывает тот факт, что для получения значения количество разыменований указателя должно совпадать с его порядком.
- В программе показана реализация указателя третьего порядка.
- Если, используя такой указатель (третьего порядка) необходимо получить значение, на которое он ссылается, делается 4 шага:
  1. по значению указателя третьего порядка получить адрес указателя второго порядка;
  2. по значению указателя второго порядка получить адрес указателя первого порядка;
  3. по значению указателя первого порядка получить адрес переменной;
  4. по адресу переменной получить её значение

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Указатели

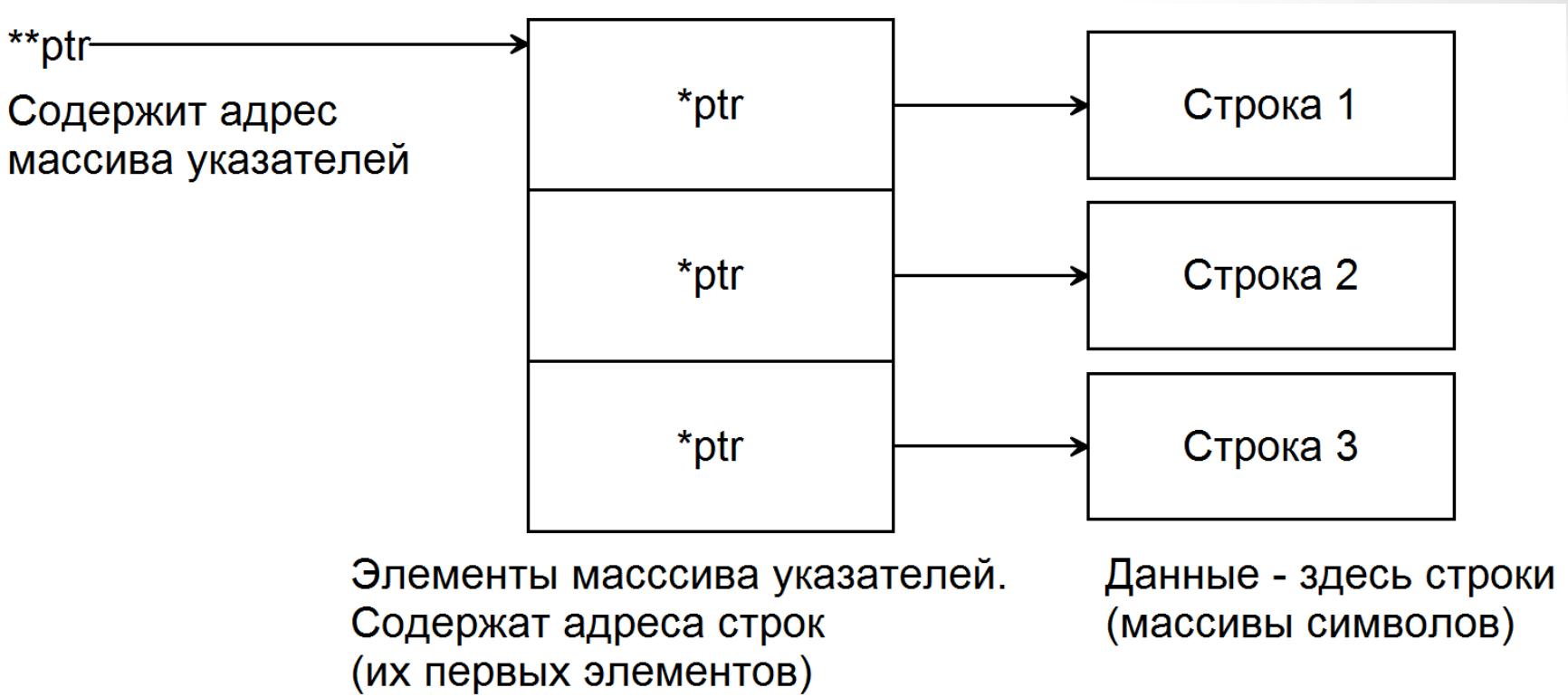


Указатель на указатель

- Указатель на указатель содержит в себе адрес, который ссылается на другой адрес, а он, в свою очередь, ссылается на адрес в памяти, где хранятся данные.
- Вроде бы и можно понять.
- Но как это применять на практике?
- И главный вопрос - Зачем оно надо???

- Например, для возможности работы с массивами указателей, которые указывают на память с данными (например, строками)
- Каждый элемент этого массива — это указатель, который содержит в себе адрес строки (первого элемента символьного массива):

Указатель на указатель



Указатель на указатель

- У нас есть указатель на указатель char **pp (он будет содержать адрес массива указателей на строки) и размер этого массива int size, который изначально равен 0.
- Нам надо написать функцию, которая будет выделять динамическую память для новых элементов массива указателей и для хранения символов новых строк.
- Эта функция будет принимать, как параметры, указатель на указатель, размер массива указателей и строку, которую надо будет записать в выделенную под нее память.
- Чтобы не усложнять задачу, в ней не будет диалога с пользователем.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Динамическое распределение памяти

- Динамическое выделение памяти необходимо для эффективного использования памяти компьютера.
- Например, мы написали какую-то программку, которая обрабатывает массив.
- При написании данной программы необходимо было объявить массив, то есть задать ему фиксированный размер (к примеру, от 0 до 100 элементов).
- Тогда данная программа будет не универсальной, ведь может обрабатывать массив размером не более 100 элементов.
- А если нам понадобятся всего 20 элементов, но в памяти выделится место под 100 элементов, ведь объявление массива было статическим, а такое использование памяти крайне не эффективно.

Динамическое распределение памяти

- В языке С для динамического распределения памяти компьютера можно использовать функции malloc(), calloc() и др.

```
void * malloc( size_t size )
void * calloc( size_t number, size_t size )
```

- Функция **malloc** выделяет блок памяти, размером **size** байт, и возвращает указатель на начало блока. Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.
- Функция **calloc** выделяет блок памяти для массива размером — **number** элементов, каждый из которых занимает **size** байт, и инициализирует все свои биты нулями.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Динамическое распределение памяти

- В языке С для освобождения памяти, выделенной ранее с помощью функций malloc(), calloc() и др. можно использовать функцию free()

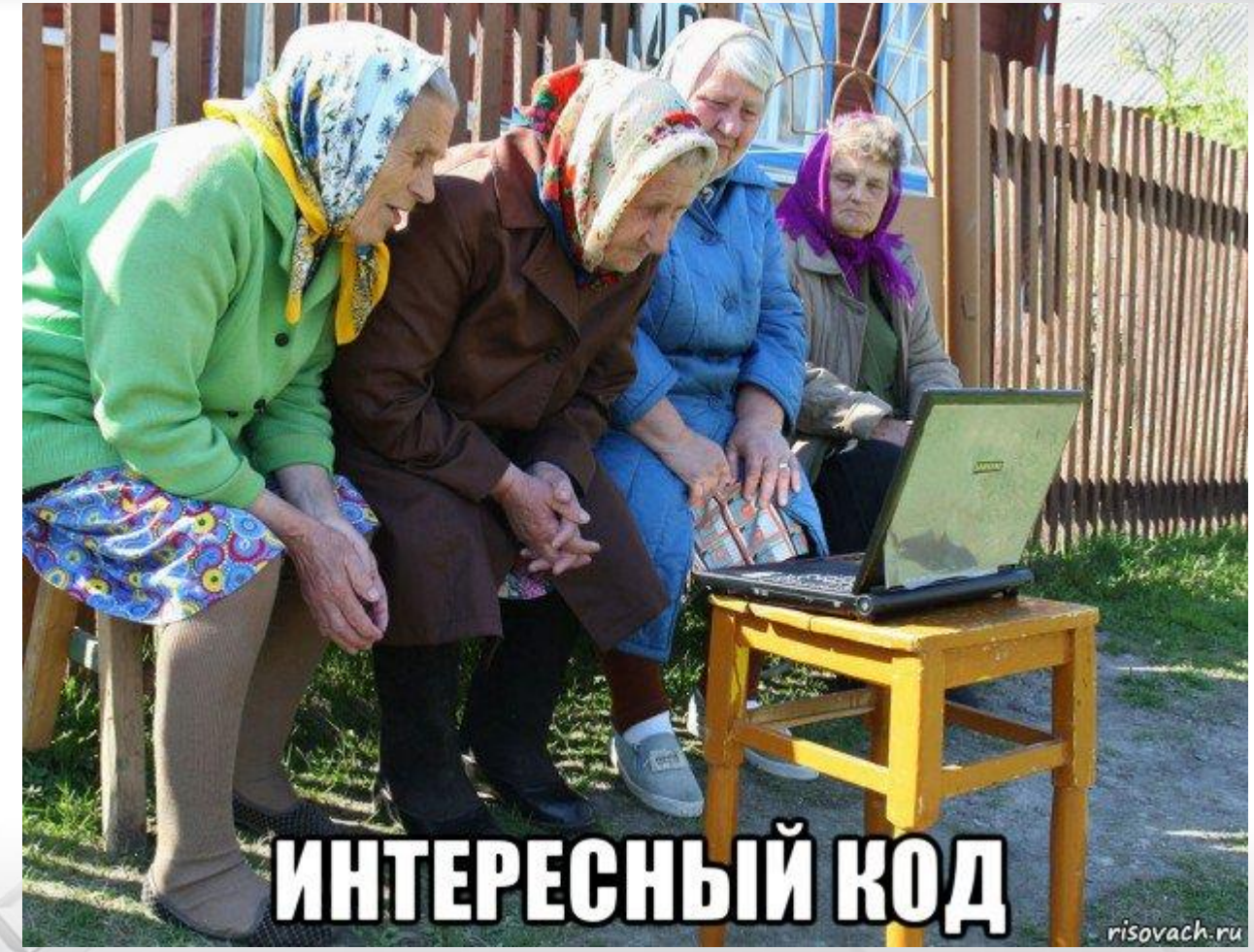
```
void free(void * ptr)
```

- Функция **free** освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или realloc освобождается. То есть освобожденная память может дальше использоваться программами или ОС.

Примечание. *Эта функция оставляет значение* ***ptr*** *неизменным, следовательно, он по-прежнему указывает на тот же блок памяти, а не на нулевой указатель.*

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Демонстрация



Динамическое распределение памяти

- В C++ для динамического распределения памяти компьютера предназначены операции **new** и **delete**.
- Операция **new** выделяет память из области свободной памяти, а операция **delete** высвобождает выделенную память.
- Выделяемая память, после её использования должна высвобождаться, поэтому операции **new** и **delete** используются парами.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Динамическое распределение памяти

```
// пример использования операции new
int *ptrvalue = new int;
// где ptrvalue – указатель на выделенный участок
// памяти типа int
// new – операция выделения свободной памяти под
// создаваемый объект.
```

- Операция **new** создает объект заданного типа, выделяет ему память и возвращает указатель правильного типа на данный участок памяти.
- Если память невозможно выделить, например, в случае отсутствия свободных участков, то возвращается нулевой указатель, то есть указатель вернет значение 0.
- Выделение памяти возможно под любой тип данных: int, float,double, char и т. д.

```
// пример использования операции delete:
delete ptrvalue;
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Создание динамических массивов

- Массивы также могут быть динамическими.
- Чаще всего операции **new** и **delete** применяются, для создания динамических массивов, а не для создания динамических переменных.
- Рассмотрим фрагмент кода, создания одномерного динамического массива.

```
// объявление одномерного динамического массива
float *ptrarray = new float[10];
// где ptrarray  – указатель на выделенный участок
// памяти под массив вещественных чисел  типа float
// в квадратных скобках указываем размер массива
```

После того как динамический массив стал ненужным, нужно освободить участок памяти, который под него выделялся.

```
// высвобождение памяти отводимой под массив:
delete [] ptrarray;
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Динамическое распределение памяти

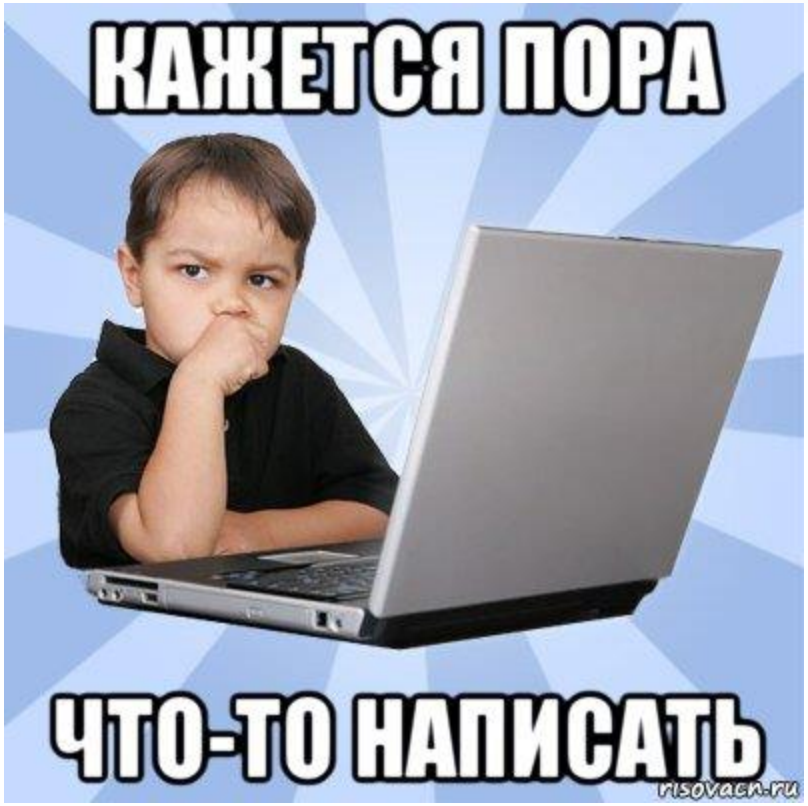
Теперь рассмотрим фрагмент кода, в котором показано, как объявляется двумерный динамический массив.

```
// объявление двумерного динамического массива:
float **ptrarray = new float* [2]; // две строки в массиве
   for (int count = 0; count < 2; count++)
      ptrarray[count] = new float [5]; // и пять столбцов
//  ptrarray  – массив указателей на выделенный участок
//  памяти под массив вещественных чисел  типа float
```

```
// высвобождение памяти отводимой под двумерный
// динамический массив:
    for (int count = 0; count < 2; count++)
        delete [] ptrarray[count];
//      где 2 – количество строк в массиве
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Динамические массивы



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Указатели на функции

Указатели могут ссылаться на функции. Имя функции, как и имя массива само по себе является указателем, то есть содержит адрес входа.

```
// объявление указателя на функцию
/*тип*/ (* /*имя указателя*/)(/*список аргументов ф-ции*/);
```

- Тип определяем такой, который будет возвращать функция, на которую будет ссылаться указатель.
- Символ указателя и его имя берутся в круглые скобки, чтобы показать, что это указатель, а не функция, возвращающая указатель на определённый тип данных.
- После имени указателя идут круглые скобки, в этих скобках перечисляются все аргументы через запятую, как в объявлении прототипа функции.
- Аргументы наследуются от той функции, на которую будет ссылаться указатель.

Указатели на функции

- Разработаем программу, которая использует указатель на функцию.
- Программа должна находить НОД – наибольший общий делитель двух целых чисел

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

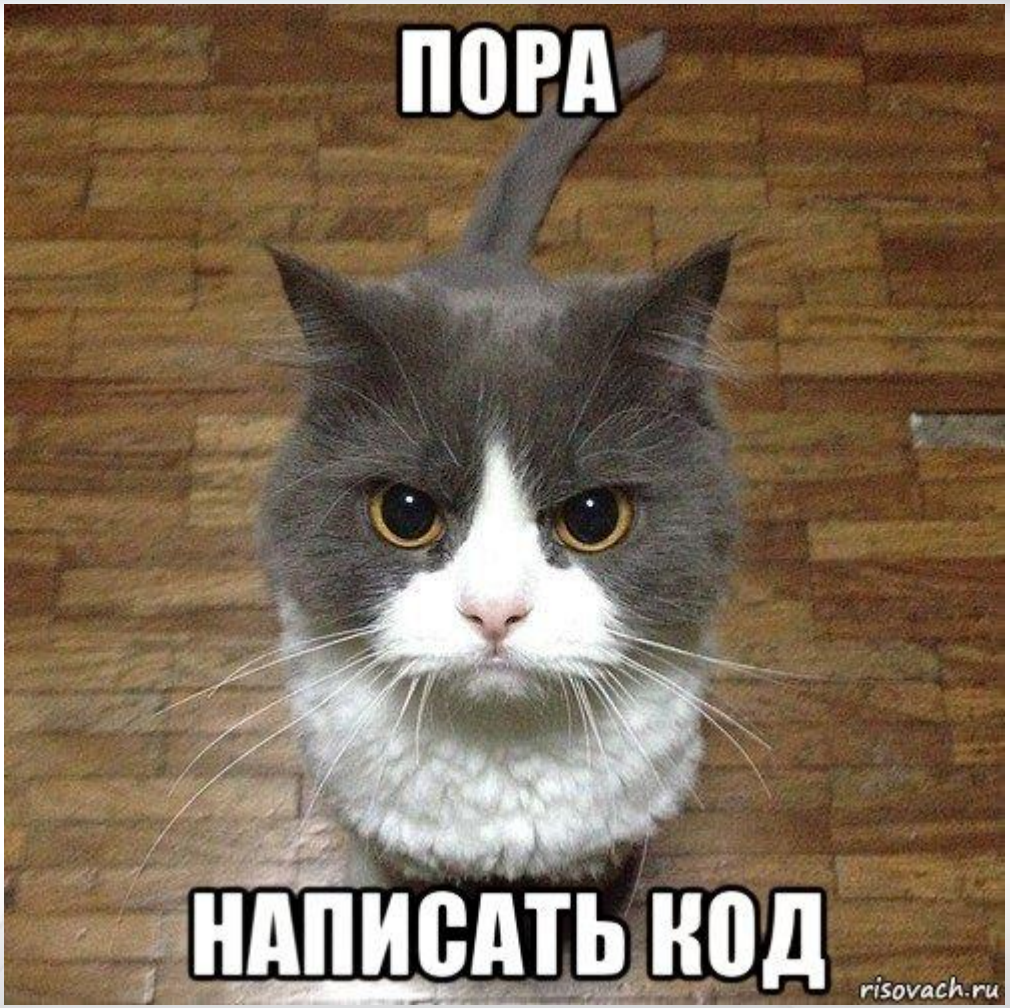
Указатели на функции

```
#include <iostream>
using namespace std;
int gcd(int number1, int number2) {
    if (number2 == 0)
        return number1;
    return gcd(number2, number1 % number2);
}

int main() {
    int (*ptrgcd)(int, int);
    ptrgcd = gcd;
    int a, b;
    cout << "Enter first number: ";
    cin >> a;
    cout << "Enter second number: ";
    cin >> b;
    cout << "GCD = " << ptrgcd(a, b) << endl;
    return 0;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Указатели



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Алгоритмизация и программирование

Программирование на C/C++

(ч.12 – указатели)

Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua