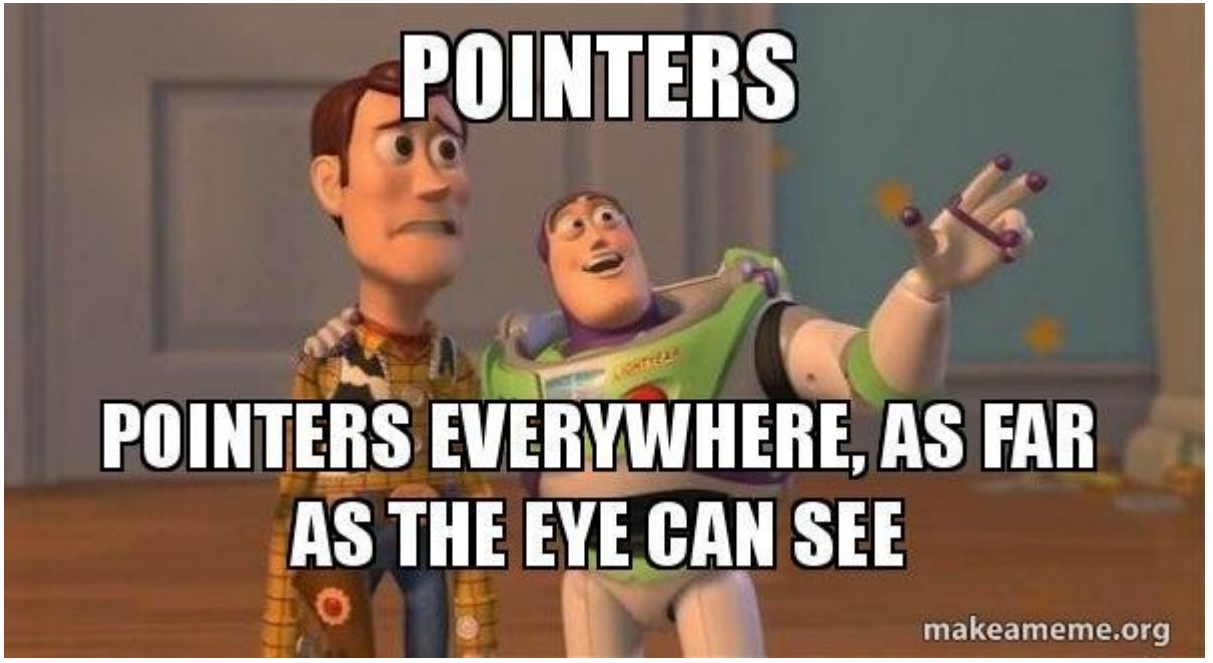
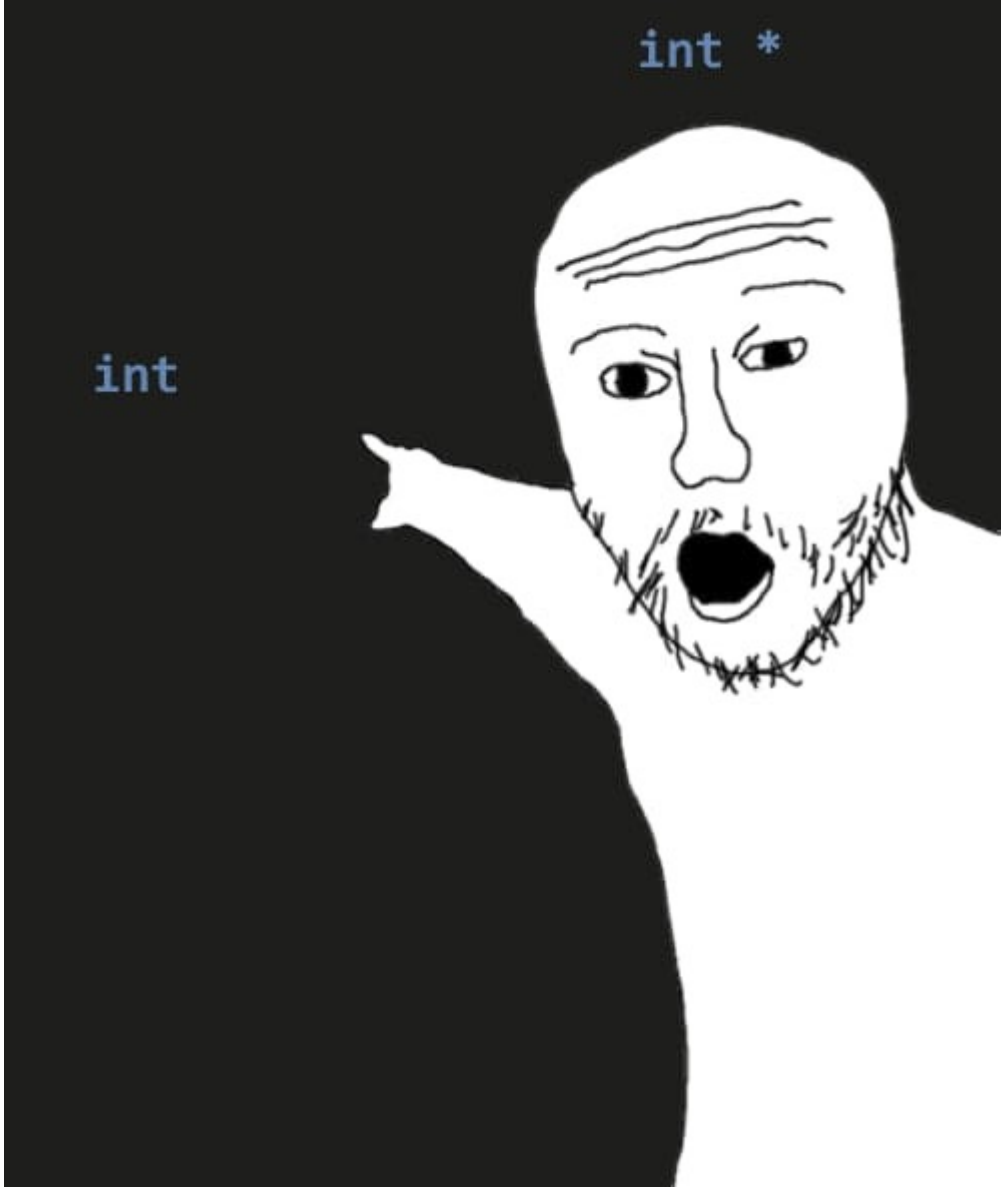
Семинар 6 по УП

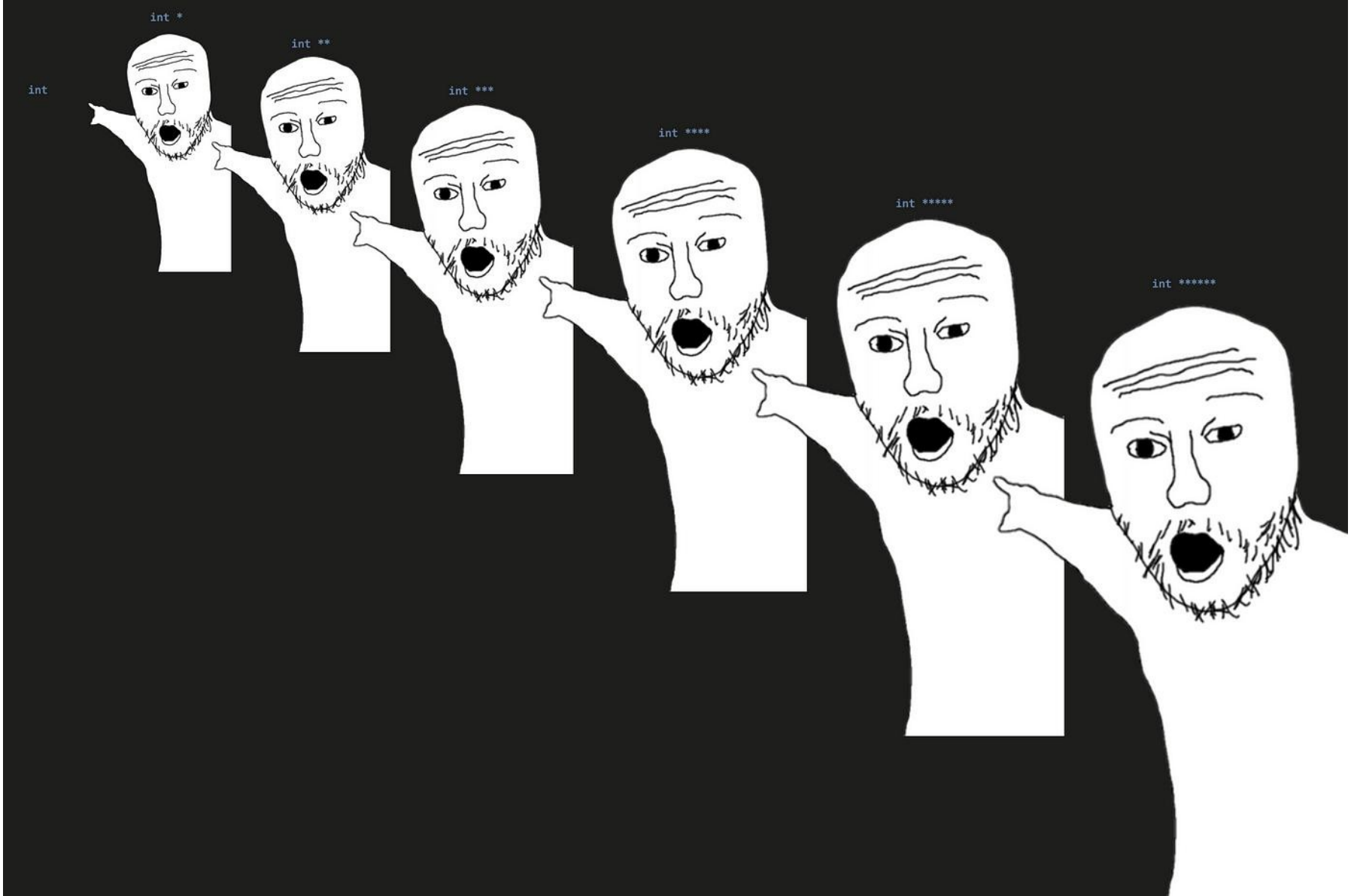
Указатели, динамични масиви, сортиране

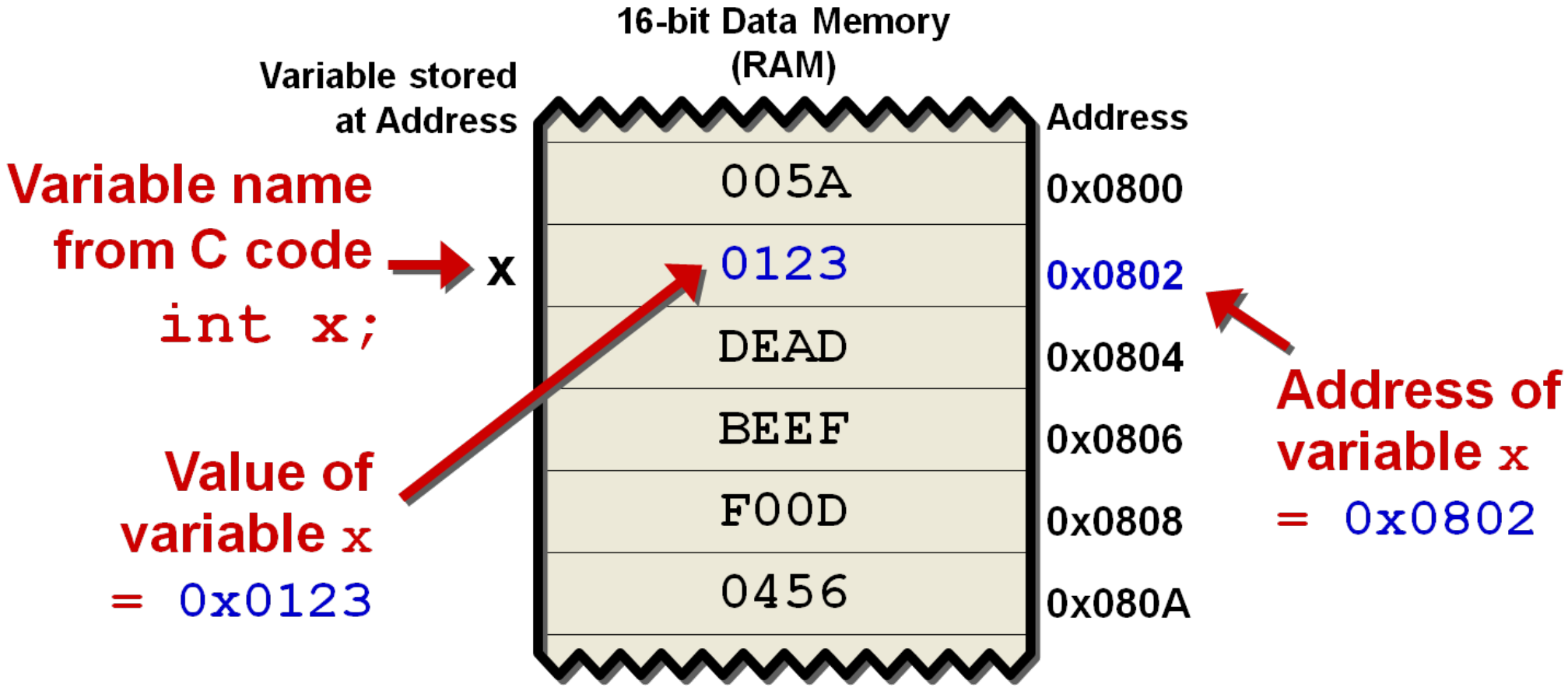


Указатели

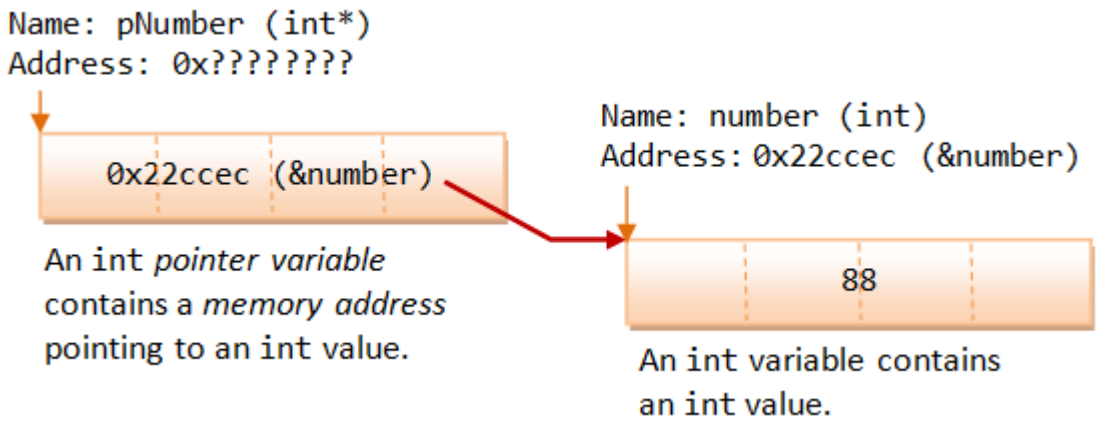
- Когато да сочиш даже е хубаво нещо.
- Указателя пази адрес в паметта
- Мислете си го като стрелкичка, която сочи нещо.
- Синтаксис <тип>* <име>;

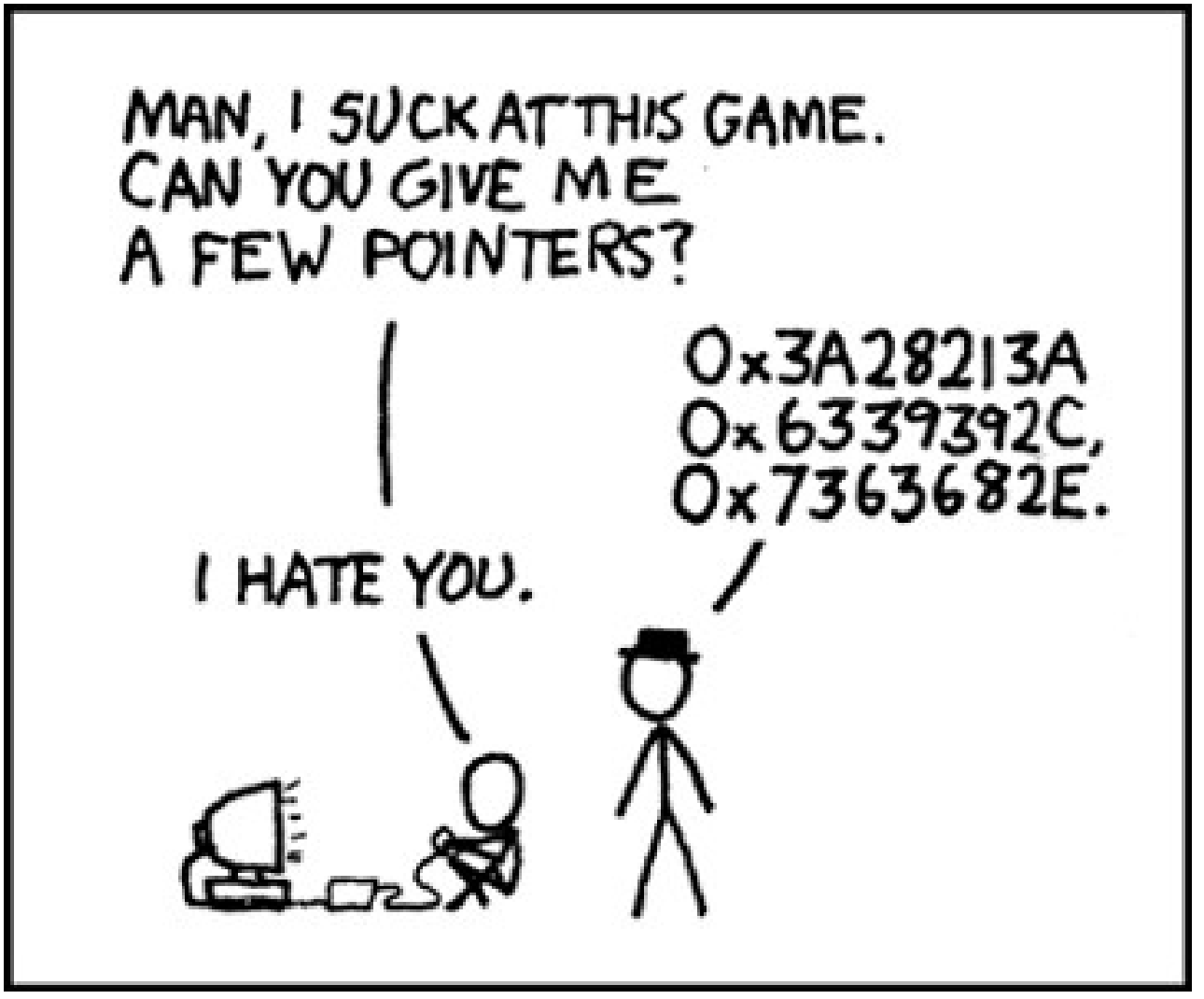

int *
int


int
int *
int **
int ***
int ****
int *****
int ******


16-bit Data Memory
(RAM)
Variable stored
at Address
Address
Variable name
from C code
int x;
x
005A
0123
DEAD
BEEF
F00D
0456
0x0800
0x0802
0x0804
0x0806
0x0808
0x080A
Value of
variable x
= 0x0123
Address of
variable x
= 0x0802

- Адресите са просто числа, които оказват клетка в паметта.
- Както знаете паметта се адресира (има биекция между естествените числа и клетките памет)




MAN, I SUCK AT THIS GAME.
CAN YOU GIVE ME
A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

Създаване и достъпване на елементи чрез указател

- Заделяме памет с new <тип>
- Достъпваме със *
- Взимаме адрес на променлива с &
- Съществува така наречения нулев указател или nullptr. Той се използва за указване, че даден указател не сочи към нищо.
- Важно!!! След като създавате памет по този начин вие отговаряте за това тя да се изтрие!!!! Това става с delete

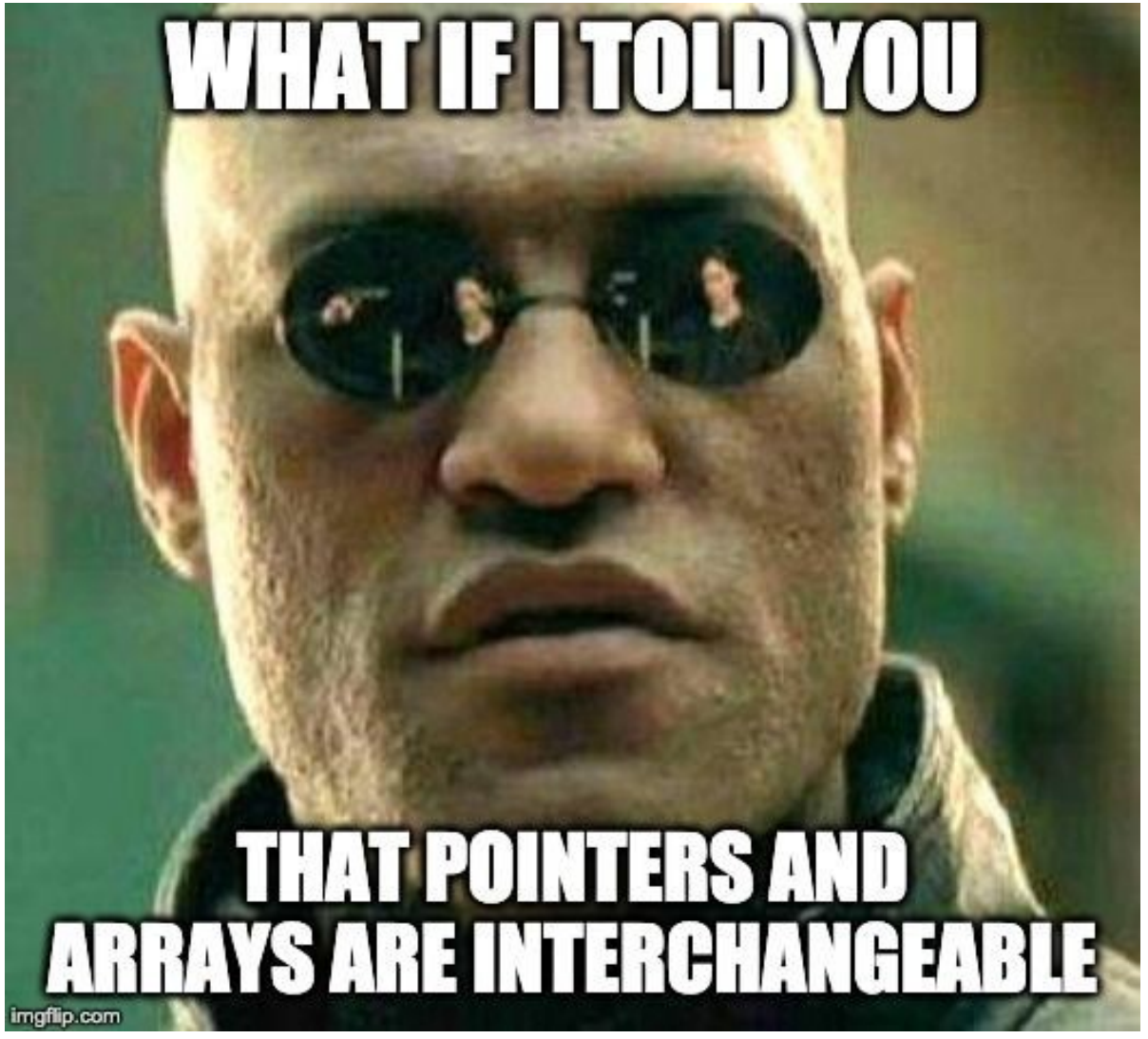
Пример

```cpp
#include <iostream>

int main()
{
    int a = 6;
    int* ptrToA = &a;
    std::cout << "The value of A is: " << *ptrToA;
    int *randomPtr = nullptr;
    randomPtr = new int;
    *randomPtr = 5;
    std::cout << "Random ptr points to value: " << *randomPtr;
    delete randomPtr;
}
```

За какво пък ми е това

- Предимство е, че указателите могат да се местят. Използвайте с удоволствие, мярка и повишено внимание да не увисне памет.
- Поради горната причина с тях могат да се правят динамични масиви.


WHAT IF I TOLD YOU
THAT POINTERS AND
ARRAYS ARE INTERCHANGEABLE
imgflip.com

Динамични масиви

- Реално самият масив представлява указател към първия си елемент.
- В: Как се дефинира масив с указател?
- О: Миии ей така: int* arr = new int[100];
- За да го изтрием delete[] arr;

Че какво му е динамичното на това?

- Аз май ви казах, че указателите се местят. Тоест във всеки момент може да сочи към нещо различно и е файн.
- Това значи, че ако сега сочи към масив със 100 елемента после може да сочи към такъв с 200.

Сега ще демонстрирам какво имам предвид

Малко за операциите с указатели

- Указателите могат да се инкрементират, да се декрементират и да се сравняват, но само с == и !=. Неща, като > < няма!
- Указателите могат да се увеличават и намаляват с число.

Примери

```
int arr[100];
int* ptr = arr;
ptr += 5;
std::cout << *ptr;
```

```
int arr[100];
int* ptr = arr;
ptr++;
std::cout << *ptr;
```

```
int arr[100];
int* ptr = arr + 10;
ptr--;
std::cout << *ptr;
```

Сортиране на масиви

- Внасяме малко ред в хаоса (Не не съм партиен).
- Сортираме по някакъв признак
- Има различни алгоритми

SelectionSort

- Разделяме масива на 2 части. Сортирана и несортирана.
- Намираме минималния елемент в несортираната част и го слагаме най-вляво
- Увеличаваме размера на сортираната с 1

Реализация

```
void selectionSort(int arr[], int size)
{
    for(int i = 0; i < size - 1; i++)
    {
        int curMinIdx = i;
        for(int j = i + 1; j < size; j++)
        {
            if(arr[j] <= arr[curMinIdx])
            {
                curMinIdx = j;
            }
        }
        std::swap(arr[i], arr[curMinIdx]);
    }
}
```

BubbleSort

- Обхождаме масива и на всяка стъпка проверяваме съседните елементи и поставяме по-големия отдясно.
- Така забелязваме, че най-големия от елементите винаги отива в края.
- После продължаваме така с втория и т.н.

Екстра илюстративност

- https://www.youtube.com/watch?v=lyZQPjUT5B4

Стига сме танцували. Време е да кодим!

```
void bubbleSort(int arr[], int size)
{
    bool swapped = true;
    for(int i = 0; i < size - 1 && swapped; i++)
    {
        swapped = false;
        for(int j = 0; j < size - i - 1; j++)
        {
            if(arr[j] > arr[j + 1])
            {
                std::swap(arr[j], arr[j + 1]);
                swapped = true;
            }
        }

    }
}
```

CountingSort

- При относително малки стойности на елементите в масива, но много елементи това е добър вариант, защото е много бърз.
- Просто броим в масив броя срещания на елемент

Код

```
void countingSort(int arr[], int size)
{
    int* counts = nullptr;
    int maxi = arr[0];
    for(int i = 1; i < size; i++)
    {
        if(maxi < arr[i])
        {
            maxi = arr[i];
        }
    }
    counts = new int[maxi + 1];
    for(int i = 0; i <= maxi; i++)
    {
        counts[i] = 0;
    }
    for(int i = 0; i < size; i++)
    {
        counts[arr[i]]++;
    }
    int idx = 0;
    for(int i = 0; i <= maxi; i++)
    {
        for(int j = 0; j < counts[i]; j++)
        {
            arr[idx++] = i;
        }
    }
}
```

```
for(int i = 0; i < size; i++)
{
    counts[arr[i]]++;
}
int idx = 0;
for(int i = 0; i <= maxi; i++)
{
    for(int j = 0; j < counts[i]; j++)
    {
        arr[idx++] = i;
    }
}
```

Задачи

- Даден е масив от числа. Да се напише функция, която връща указател към максималния елемент.
- Стек се нарича структура от данни, в която последния вкаран елемент е първи изкаран. Това значи, че може да се вкарва само отгоре и да се чете пак отгоре. Направете реализация на стек с динамичен масив
- Да се напише програма, която конкатенира 2 масива с указател.
- Да се напише функция, която проверява дали два низа от скоби са балансирани.
- Даден е масив със силите на N чудовища. На всяка минута I, I – тото чудовище влиза на арената и убива всички със сила по-малка от неговата. Напишете програма, която определя във всяка минута от 1 до N колко чудовища има на арената.