Основы алгоритмизации и программирования

Пашук Александр Владимирович

pashuk@bsuir.by

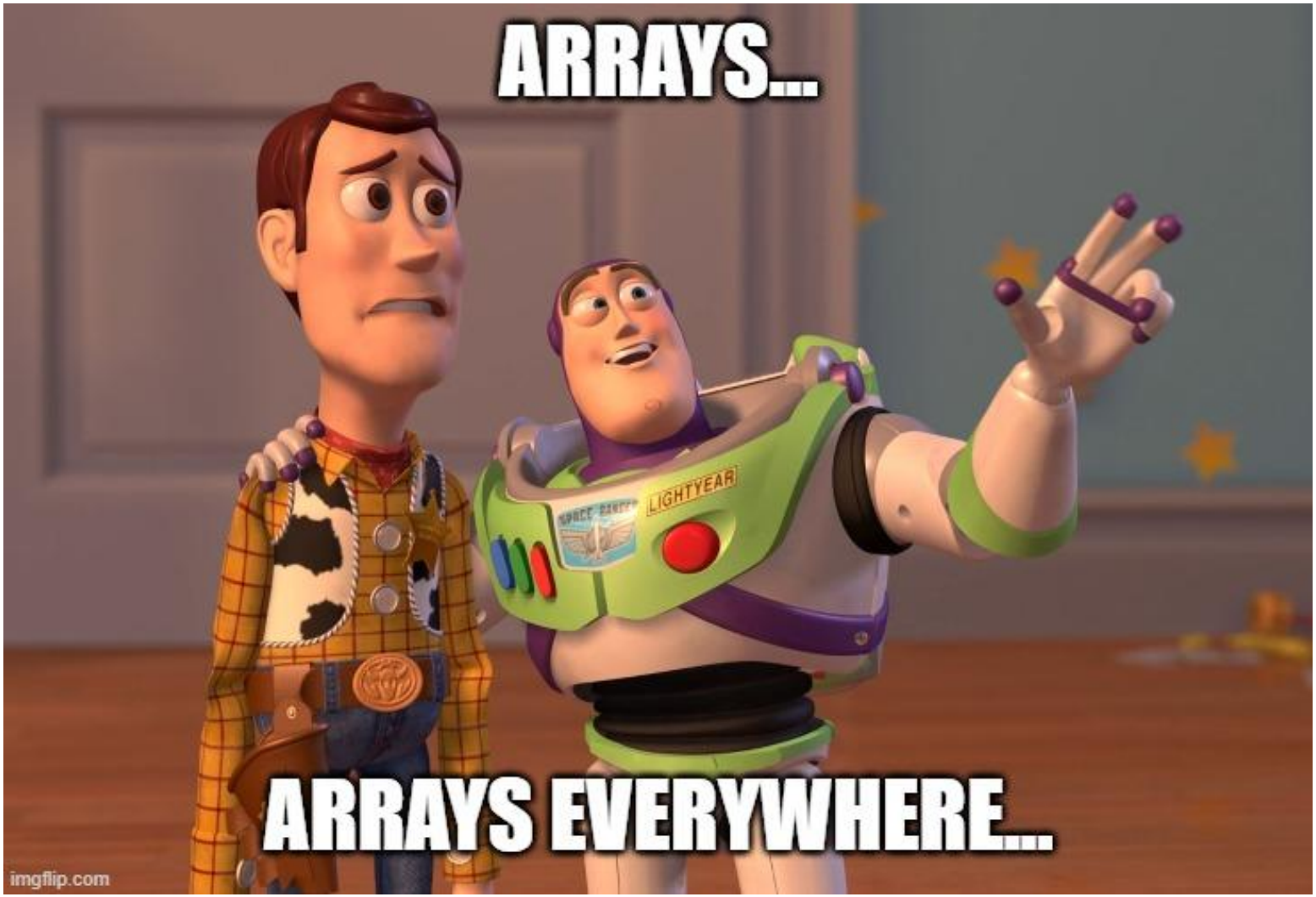
Содержание блока

Сложные типы данных (массивы)

1. Одномерные массивы
2. Многомерные массивы
3. Указатели, динамическое распределение памяти

Содержание лекции

1. Одномерные массивы
2. Генерация массивов
3. Поиск элементов в массиве
4. C-style строки. Операции со строками
5. Вопросы из теста


ARRAYS...
LIGHTYEAR
ARRAYS EVERYWHERE...
imgflip.com

Что такое массивы?

Массивы – это механизм, позволяющий группировать вместе данные одного типа.

Данные, сгруппированные в массиве, могут быть как **основных типов** (int ,char), так и **определенных пользователем** (структуры, объекты).

Доступ к элементам массива осуществляется по **индексу**.

В C++ не только массивы используются для группирования элементов одного типа.

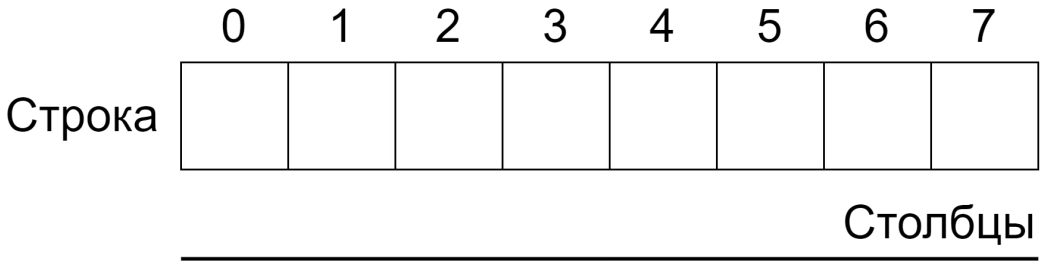
Что такое массивы?

Массив – именованная последовательность областей памяти, хранящих однотипные элементы. Каждая такая область памяти называется **элементом массива**.

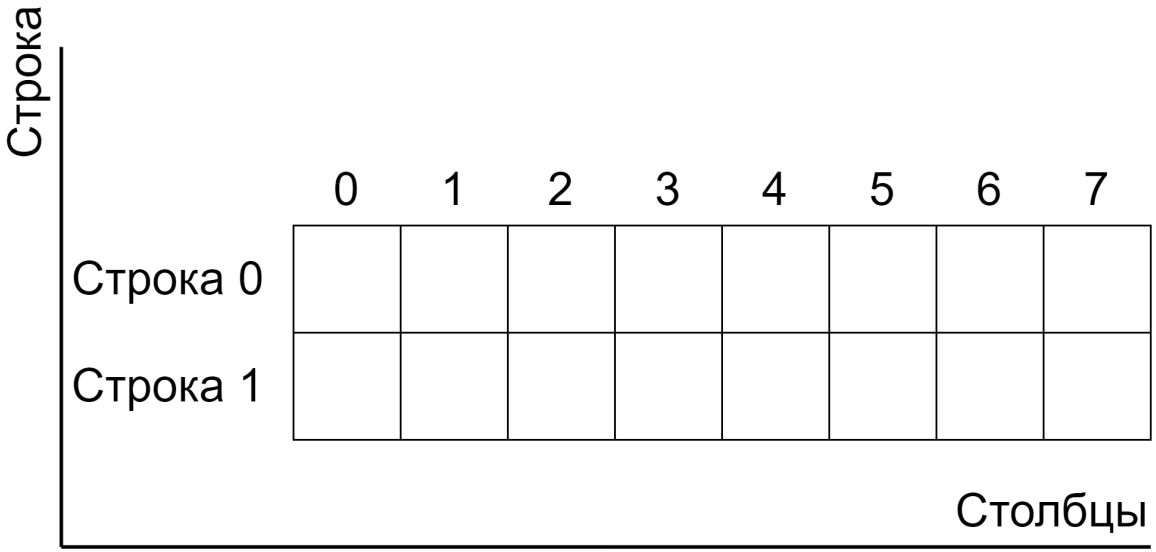
Массивы обладают **размерностью** (большей или равной единице), которой задается число элементов, содержащихся в них, а также **измерением**, что предполагает возможность описания в программе одно- и многомерных массивов.

Количество элементов в массиве называется его **размером**.

Что такое массивы?

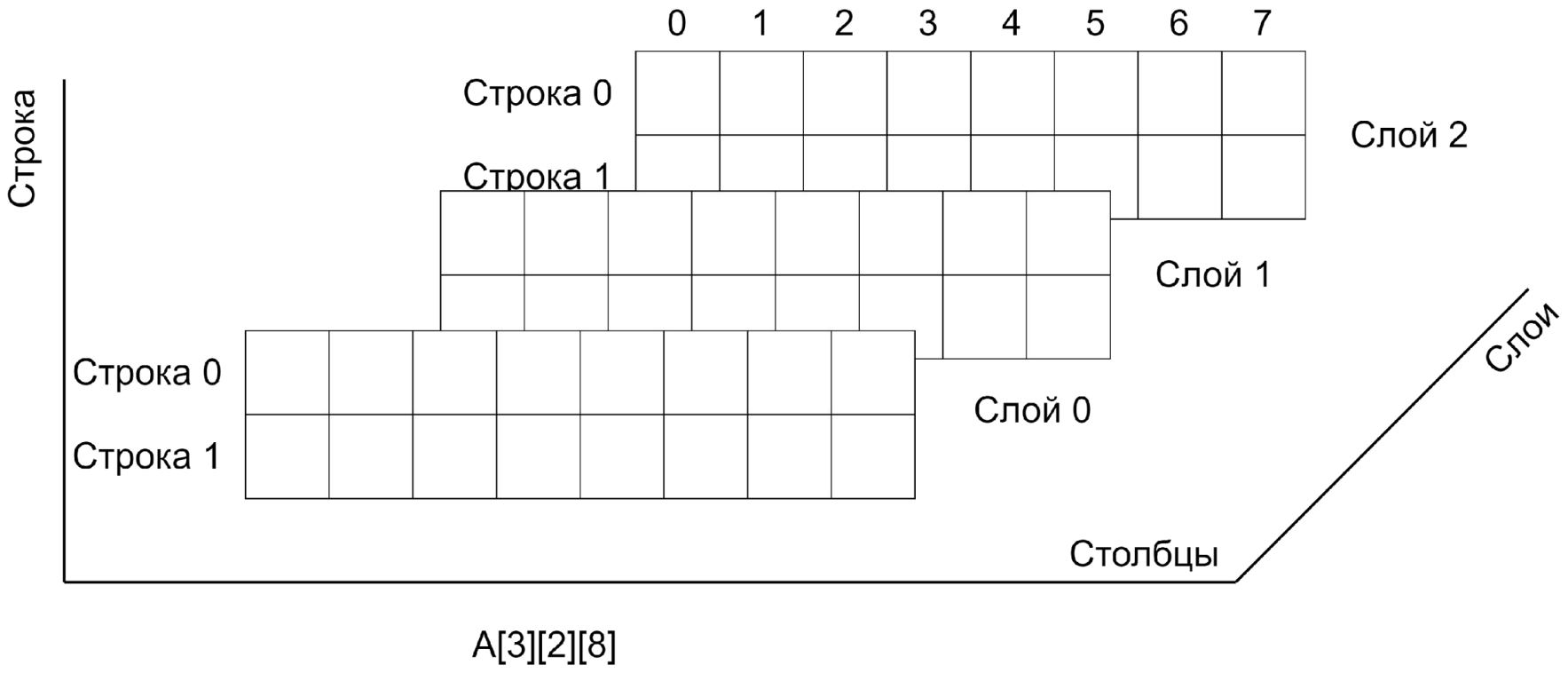


A[8]



A[2][8]

Что такое массивы?



Объявление массивов

Синтаксис определения массива без дополнительных спецификаторов и модификаторов имеет два формата:

```
<type> <array_name>[<size_of_array>];
```

или

```
<type> <array_name>[];
```

Объявление массивов

Элементами массива **не могут быть** функции, файлы и элементы типа void.

Размерность массива может быть опущена в случаях если:

- при объявлении массив инициализируется;
- массив объявлен как формальный параметр функции;
- массив объявлен как ссылка на массив, явно определенный в другом файле.

Примеры

```
int a[100];
double d[14];
char s[]="Программирование";

const int t=5, k=8;
float wer[2*t+k];

const int N_max=853;
int sample[N_max];
```

Инициализация массивов

В C++ одновременно с объявлением массива можно задать начальные значения всех элементов массива или только нескольких первых его компонент.

```
float t[5]={1.0, 4.3, 8.1, 3.0, 6.74};
char b[7]={'П','р','и','в','е','т'};
int d[10]={1, 2, 3};
char a[10]="Привет";
```

Если в определении массива явно указан его размер, то **количество начальных значений не может быть больше количества элементов в массиве**.

Пример

```
int main() {
    char str[] = "Hello world";

    printf(
        "Length of `%s` is equal %d\n",
        str,
        sizeof (str)
    );

    return 0;
}
```

Инициализация

```
int arr1[10] = {1, 2, 3};
for (int i=0; i < 10; i++)
    cout << arr1[i] << " ";
// 1 2 3 0 0 0 0 0 0 0

cout << endl;

char arr2[10] = {'a', 'b', 'c'};
for (int i=0; i < 10; i++)
    cout << arr2[i] << " ";
// a b c
```

Обращение к элементам массива

```
d[55] // индекс задается как константа
s[i] // индекс задается как переменная
w[4*p] // индекс задается как выражение
```

Компилятор в процессе генерации кода задает начальный **адрес** массива, который в дальнейшем не может быть переопределен. **Начальный адрес массива – это адрес первого элемента массива.**

Имя массива считается константой-указателем, ссылающимся на **адрес** начала массива.


Does your dog bite?
No, but it can hurt you in other ways.
Array indexing starts at 1

Обращение к элементам массива

- Индексация элементов массива начинается с нуля.
- Первому элементу массива соответствует значение индекса 0, второму - значение индекса 1, элементу с порядковым номером k - значение индекса k-1.

Справка: E.W. Dijkstra Archive: Why numbering should start at zero (EWD 831)

Определение размера памяти

Массив занимает **непрерывную** область памяти. Для одномерного массива полный объем занимаемой памяти в байтах вычисляется по формуле:

```
Байты = sizeof (тип) * размер массива
```

Массив представляет собой набор однотипных данных, расположенных в памяти таким образом, чтобы по индексам элементов можно было легко вычислить адрес соответствующего значения.

```
адрес(A[i]) = адрес(A[0]) + i*k
```

Пример

```
int main() {
   char s;
   char str[] = "Hello world";

   cout << sizeof(s) << endl; // 1
   cout << sizeof(str) << endl; // ???

   return 0;
}
```

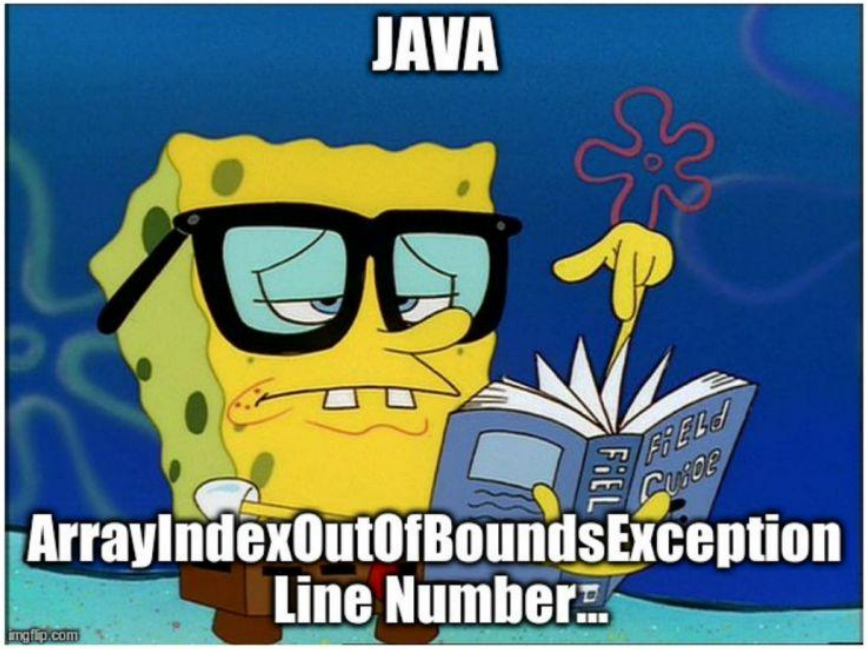
Что произойдет?

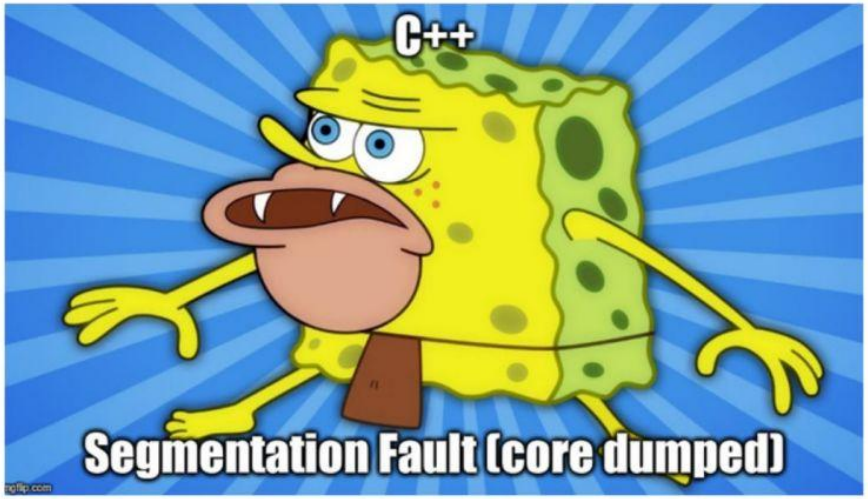
```
int main() {
   int arr[10];
   for (int i=0; i < 100; i++)
      arr[i] = i;

   for (int i=0; i < 100; i++)
      cout << arr[i] << endl;

   return 0;
}
```

Что произойдет?





Массивы

В языке C++ **не производится проверки границ массивов**: таким образом, исполнение кода не остановится при выходе за границы массива.

Если переполнение массива происходит во время выполнения оператора присваивания, то лишние значения могут присвоиться другим переменным или включиться в текст программы.

Ответственность за корректную работу с элементами массива лежит на разработчике.

Интересный факт

Операция [] является коммутативной, т.к. допускает обмен операндов местами:

```
int main(){
    int n = 3;
    int arr[5] = {1, 2, 3, 4, 5};

    cout << arr[1] + arr[n+1] << endl; // 7
    cout << 1[arr] + (n+1)[arr] << endl; // 7
}
```

Генерация массивов

Генерация массивов – автоматическое формирование значений элементов. Генерацию массива (массивов) в программе оформляют в виде отдельной функции.

Стандартными способами генерация массивов являются:

- ввод данных с клавиатуры
- формирование значений через генератор случайных чисел
- вычисление значений по формуле
- ввод данных из файла.

Ввод данных с клавиатуры

```
int main() {
    int n = 5; int a[n];

    printf("\nEnter %d elements: \n", n);

    for (int i=0; i<n; i++){
        printf("x[%d] = ", i);
        scanf("%d", &a[i]);
    }

}
```

Генерация случайных чисел

```
int main(){
    int n=10, arr[n];
    // cout << RAND_MAX << endl;
    for (int i=0; i < n; i++)
        arr[i] = rand();

    for (int i=0; i < n; i++)
        cout << arr[i] << " ";
    // 41 18467 6334 26500 …

    return 0;
}
```

Генерация случайных чисел

Формула генерации случайных чисел по заданному диапазону:

```
n = <first_value> + rand() % <last_value>;
n = <first_value> + rand()*1.0 / (RAND_MAX)*<last_value>
```

Например:

```
for (int i=0; i < n; i++)
   arr[i] = rand() % 5 + 1;
// 2 3 5 1 5 5 4 4 3 5
```

Генерация случайных чисел

Функция **rand()** один раз генерирует случайные числа, а при последующих запусках программы всего лишь отображает сгенерированные первый раз числа.

Такая особенность нужна для того, чтобы можно было **правильно отладить разрабатываемую программу**: при отладке программы, внеся какие-то изменения, необходимо удостовериться, что программа срабатывает правильно, а это возможно, если входные данные остались те же, то есть сгенерированные числа.

Генерация случайных чисел

```
#include <ctime>

int main(){
    int n=10, arr[n];

    srand( time(0) );

    for (int i=0; i < n; i++)
        arr[i] = rand() % 5 + 1;

    // ...
    // 4 5 5 5 2 2 2 3 4 5
    // 5 2 4 4 1 2 1 5 3 1
}
```

Вычисление по формуле

```
int main(){
    int n=20, arr[n]={1, 1};

    for (int i=2; i < n; i++)
        arr[i] = arr[i-2] + arr[i-1];

    for (int i=0; i < n; i++)
        cout << arr[i] << " ";

    // 1 1 2 3 5 8 13 21 34 55 89 144 233 …
    return 0;
}
```

Вывод массивов

```
int main(){
    int a[max], n;

    do {
        printf("\nEnter n (n<=20):");
        scanf ("%d", &n);
    } while (n > max);

    printf("\nEnter %d elements: \n", n);
    for (int i=0; i<n; i++){
        printf("x[%d] = ", i);
        scanf("%d", &a[i]);
    }

    for (int i=0; i<n; i++)
        printf("%d\t",a[i]);
}
```

Поиск в массиве

Существует две основных формулировки задачи поиска:

- найти элемент массива (первый или последний), удовлетворяющий заданному условию;
- найти все элементы массива, удовлетворяющие некоторому условию.

Любой поиск связан с последовательным просмотром элементов массива и проверкой их соответствия условию поиска

Поиск единственного элемента

В случае поиска единственного элемента основу алгоритма решения задачи составляет цикл, содержащий в качестве условия продолжения отрицание условия поиска.

Например: требуется проверить, есть ли среди элементов массива A длиной n элемент со значением, равным заданному значению x

Поиск единственного элемента

Возможны две ситуации:

- такое элемент существует, тогда при некотором значениии индекса i выполняется условие `A[i] = x`
- такого элемента в массиве нет.

В первом случае поиск нужно завершать при обнаружении искомого элемента, в втором - при достижении конца массива.

Поиск единственного элемента

```
int n = 10, A[n] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int i = 0, x = 9;

while (A[i] != x && i < n)
    i++;
// OR
while (i < n) {
    if (A[i] == x)
        break;
    i++;

cout << i << endl;
```

Строковые переменные

Обычно в C++ используется два вида строк: **строка как массив типа char** и **строка как объект класса string**.

```
const int MAX = 80;
char name[MAX];

cout << "Enter name: ";
cin >> name;
cout << "You entered: " << name << endl;
```

Пример (Лекция 6)

```
#include <cstdio>

int main() {
    char name[80];

    printf("Enter the name: ");
    scanf("%s", name);
    printf("Hello, %s", name);
    // Enter the name: test user
    // Hello, test

    return 0;
}
```

Пример

```
int main() {
   char s;
   char str[] = "Hello world";

   cout << sizeof(s) << endl; // 1
   cout << sizeof(str) << endl; // ???

   return 0;
}
```

Нулевой символ

- Каждый символ в строке занимает 1 байт (таблица ASCII).
- Все строки должны завершаться байтом, содержащим 0.
- В символьном виде такой байт представляется в виде `\0`, код которой в ASCII равен 0. Завершающий ноль называется **нулевым символом**.

Пример

```
cout.setf(ios::boolalpha);

char name[] = "Username";

cout << (name[8] == '\0') << endl; // true
cout << name << endl; // Username

name[4] = '\0';

cout << name << endl; // User
```

Пример

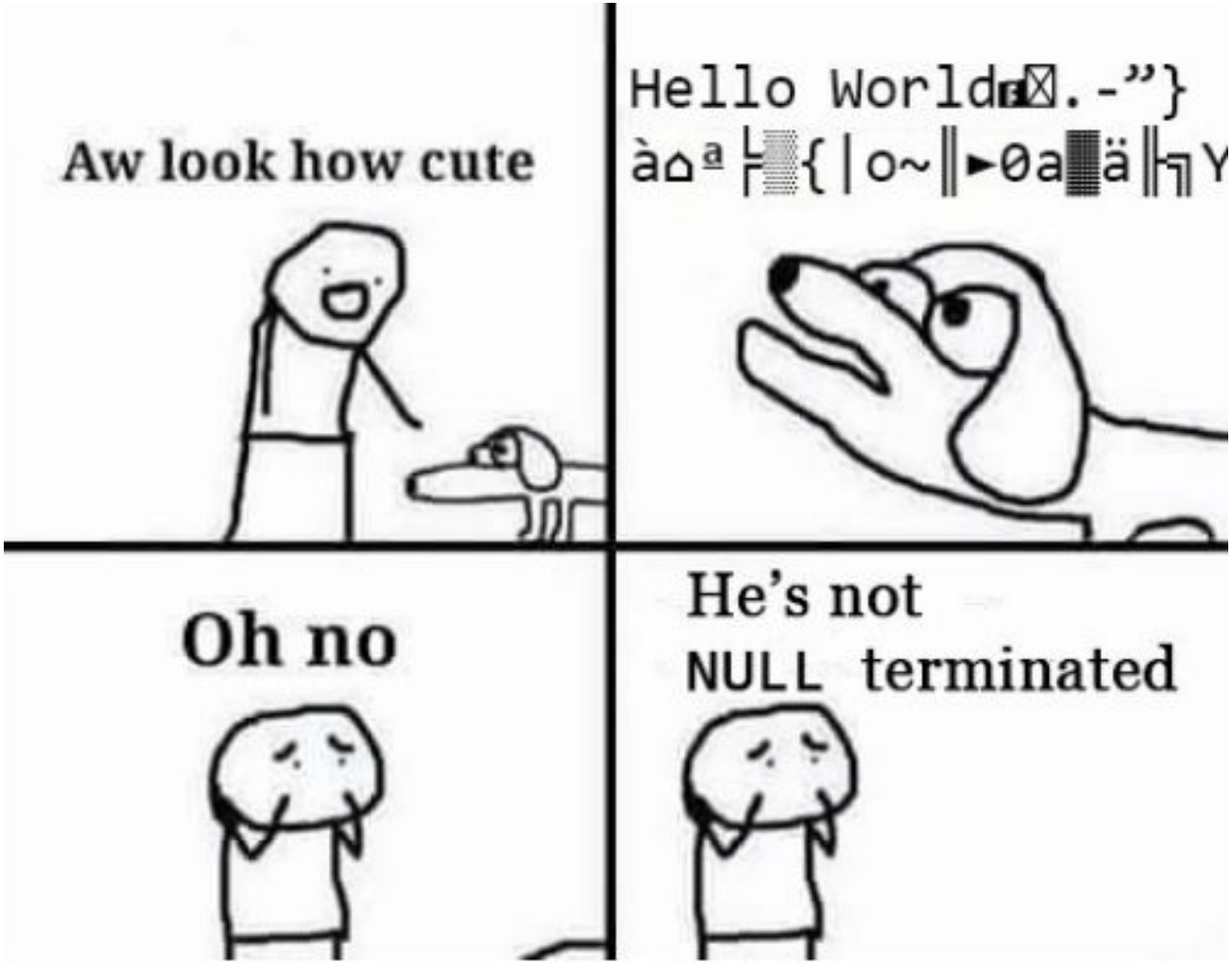
```
int main(){
    char name[] = "Hello, world";

    cout << name << endl; // Hello, world

    name[12] = '!';

    cout << name << endl; // ???

    return 0;
}
```



It never gets old...

Пример

```
int main(){
    char name[100];
    cout << "Enter the name: ";

    // cin >> name; // Not safe!
    // cin >> setw(100) >> name; // Better!
    cin.get(name, 100); // Good!

    // cin.get(name, 100, '$'); // Multiline input
    cout << name << endl;
    return 0;
}
```

Копирование строк

```
int main(){
    char str1[] = "Hello", str2[12];

    int i = 0;
    for (i=0; i < strlen(str1); i++)
        str2[i] = str1[i];

    str2[i] = '\0'; // Important!

    cout << str2 << endl;
    return 0;
}
```

Операции над строками

```
#include <cstring>

char str1[] = "Hello World", str2[12], str3[6];

// strcpy(<destination>, <source>);
strcpy(str2, str1);
cout << str2 << endl; // "Hello World«

// strncpy(<destination>, <source>);
strncpy(str3, str1, 5);
str3[5] = '\0'; // Important!
cout << str3 << endl; // Hello
```

Операции над строками

```
char str1[] = "Hello World", str2[12], str3[6];

// strcmp (<str1>, <str2>)
cout << strcmp(str1, str2) << endl; // 0

strcpy(str2, "ABCDE");
cout << strcmp(str1, str2) << endl; // 1

strcpy(str2, "abcde");
cout << strcmp(str1, str2) << endl; // -1
```

Операции над строками

```
char src[50] = "efghijkl";
char dest[50]= "abcd";

strcat(dest, src);
cout << dest << endl; // abcdefghijkl

strcpy(dest, "abcd");
strncat(dest, src, 5);
cout << dest << endl; // abcdefghi
```

Операции над строками

```
char str[] = "ABC 123 #";

cout << str[0] << ": " << isalpha(str[0])
     << endl; // A: 1

cout << str[4] << ": " << isdigit(str[4])
     << endl; // 1: 1

cout << str[0] << ": " << ispunct(str[0])
     << endl; // A: 0
```

Пример вопроса на экзамене

Доступ к элементам массива осуществляется с помощью:

- Подхода FIFO
- Операции точки
- Имени элемента
- Индекса элемента

Пример вопроса на экзамене

Что выведет данная программа?

- -15
- -30
- Случайное значение
- Ошибку компиляции

```
#include < stdio.h >

using namespace std;

int main() {
    int array[] = {10, 20, 30};

    cout << -2[array];

    return 0;
}
```

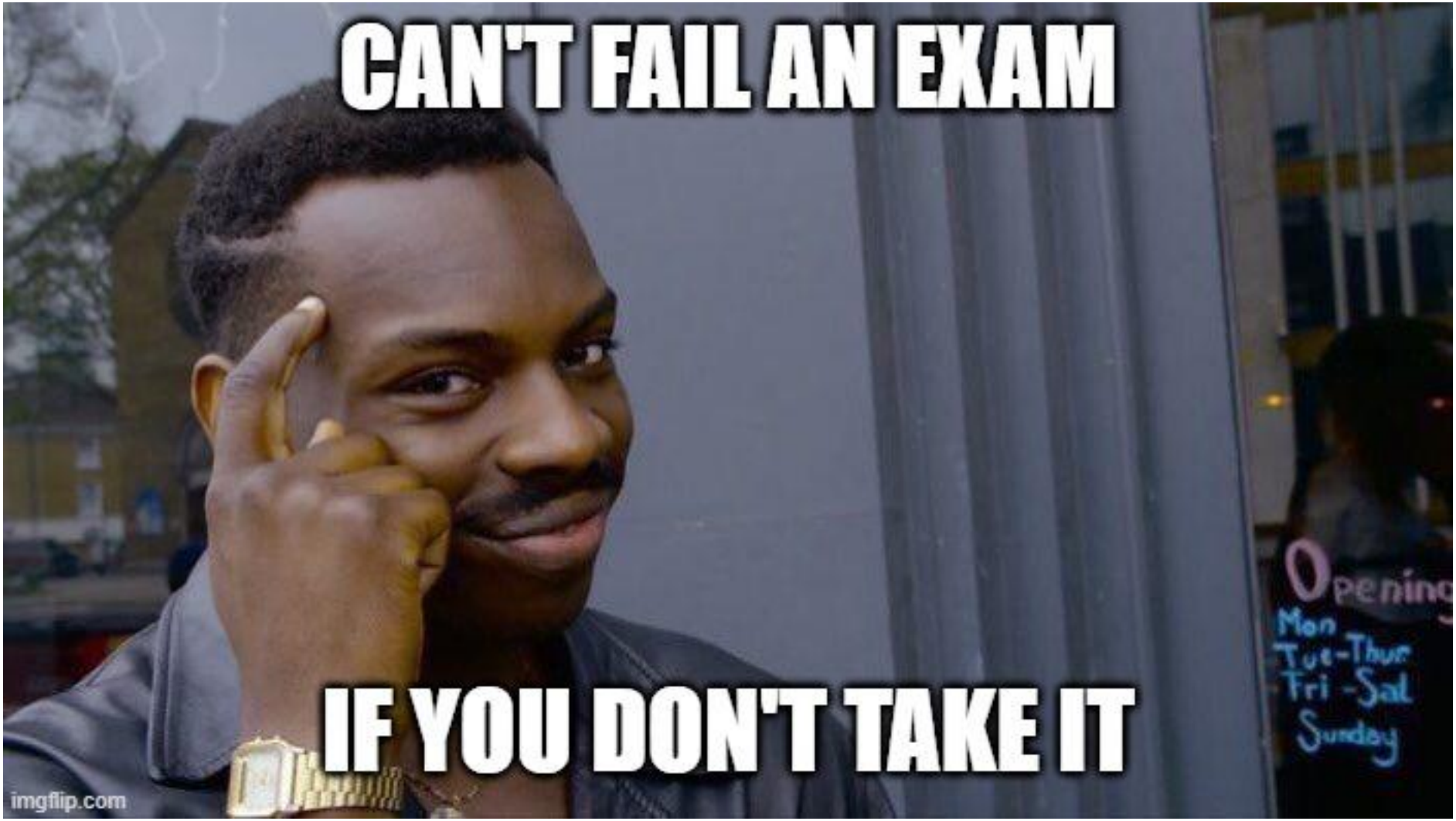
Пример задачи на экзамене

Объявите одномерный целочисленный массив, в котором не более 100 элементов.

Выполните генерацию массива первыми 100 простыми числами.

Вывод массива на экран организовать в строку (или в строки по 10 элементов в каждой).

Генерация и ввод массива должны быть оформлены в виде функций.


CAN'T FAIL AN EXAM
IF YOU DON'T TAKE IT
Opening
Mon
Tue-Thur
Fri -Sat
Sunday
imgflip.com