Лекция 3

# Массивы и строки

# Массив

**Массив** – это группа однотипных элементов, имеющих общее имя и расположенных в памяти рядом.

**Особенности:**

- все элементы имеют один тип
- весь массив имеет одно имя
- все элементы расположены в памяти рядом

На рисунке показана структура целочисленного одномерного массива **a**. Размер этого массива — 16 ячеек:

**Int a[16];**

| 5 | -12 | -12 | 9 | 10 | 0 | -9 | 64 | 11 | -1 | 39 | 2 | -7 | 43 | 23 | 65 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] | a[10] | a[11] | a[12] | a[13] | a[14] | a[15] |

Заметьте, что максимальный индекс одномерного массива **a** равен **15**, но размер массива 16 ячеек. Нумерация элементов массива в Си начинается с НУЛЯ !

**Примеры:**

```
string  ListStudent [30];   //список студентов в группе
int  flat [180]; //квартиры в доме
float  x[10], y[10];  //координаты точек на плоскости
char  Name [10] = {‘v’, ‘a’, ‘l’, ‘e’, ‘n’, ‘t’, ‘i’, ‘n’};   //имя
```

# Инициализация массивов

1. Явное задание размеров массива:
   **int flat [180];**
2. Инициализация массива с присвоением начальных значений:

**string days[7]={“Sunday”, “Monday”, “Tuesday”,**
**“Wednesday”, “Thursday”, “Friday”, “Saturday”};**

Если начальные значения не заданы, в ячейках находится **«мусор»!**

Следующая инициализация выделит память под 5 целых чисел, а начальных значений задано только 3. В этом случае четвёртый и пятый элемент будут инициализированы (по умолчанию) значением 0.

**int testScore[5]= {74, 87, 91};**

Инициализация следующего вида не будет компилироваться:

**float milles[4]= {74.4, 87.2, , 91.7};**

3. Неявное задание размеров массива (размер массива определяется числом элементов справа от оператора присваивания):
   **char name[ ]={ ‘v’, ‘a’, ‘l’, ‘e’, ‘n’, ‘t’, ‘i’, ‘n’ };**
4. Размер определяется константой:

**const int N=15;**
**string ListStudent [N];**

# Что неправильно?

```
const int N = 10;
float A[N];
```

```
int X[4.5];
```

```
int A[10];
A[10] = 0;
```

выход за границы массива (стираются данные в памяти)

```
float X[5];
int n = 1;
X[n-2] = 4.5;
X[n+8] = 12.;
```

```
int X[4];
X[2] = 4.5;
```

дробная часть отбрасывается (ошибки нет)

```
float A[2] = { 1, 3.8 };
```

```
float B[2] = { 1., 3.8, 5.5 };
```

# Массивы

Объявление:

```
const int N = 5;
int A[N], i;
```

Ввод значении элементов массива с клавиатуры:

```
printf("Введите 5 элементов массива:\n");
for( i=0; i<N; i++) {
  printf ("A[%d] = ", i );
  scanf ("%d", &A[i] );
  }
```

```
A[0] = 5
A[1] = 12
A[2] = 34
A[3] = 56
A[4] = 13
```

По

Вы

```
for( i=0; i<N; i++) A[i]=A[i]*2;
```

```
printf("Результат:\n");
for( i=0; i<N; i++)
  printf("%4d", A[i]);
```

```
Результат:
  10   24   68 112   26
```

# Случайные числа

## Случайное целое число X в интервале от 1 до 20

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
   srand((unsigned)time(NULL));
   int x=rand()%20+1;      // генерируем числа в диапазоне от 1 до 20
   printf("%i",x);
   return 0;
}
```

**void srand(unsigned int)** - функция установки начального значения генератора псевдослучайных чисел.
**unsigned** - приведение аргумента к типу беззнакового целого
**time** - функция выдачи текущего времени
**NULL** - нулевой указатель, передаваемый функции time для выдачи текущего времени только как результата.

# Заполнение массива случайными числами

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

main()
{
    srand((unsigned)time(NULL));
    const int N = 10;
    int A[N], i;
    printf("Исходный массив:\n");
for (i=0; i<N; i++) {
  A[i] = random(100) + 50;
  printf("%4d", A[i]);
  }
...
}
```

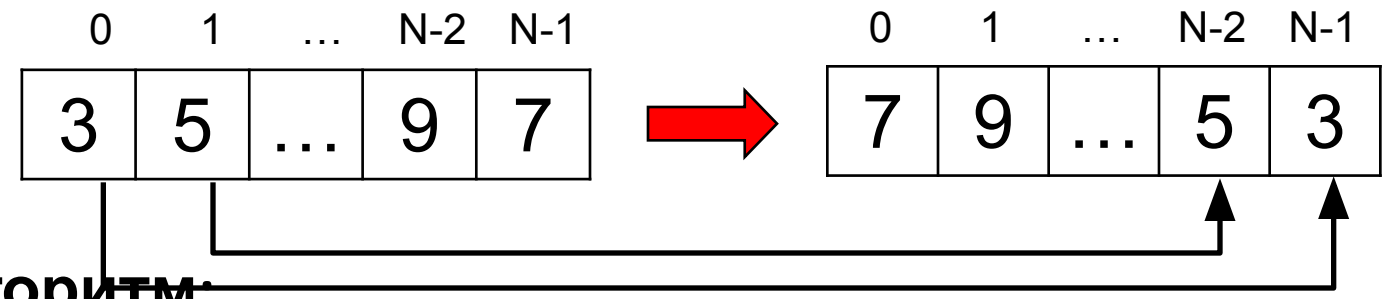
? Какой интервал?

`int random(int N)` – функция выдает случайное число от 0 до N-1

```
{ return rand() % N; }
```

# ПРИМЕР : Реверс массива

**Задача:** переставить элементы массива в обратном порядке (выполнить инверсию).




**Алгоритм:**

сумма индексов `N-1`

поменять местами `A[0]` и `A[N-1]`, `A[1]` и `A[N-2]`, …

**Псевдокод:**

```
for (i=0; i< N/2; i++)
 // поменять местами A[i] и A[N-1-i]
```

# ПРИМЕР : Циклический сдвиг элементов массива

**Задача:** сдвинуть элементы массива влево на 1 ячейку, первый элемент становится на место последнего.

**Алгоритм:**

`A[0]=A[1]; A[1]=A[2];… A[N-2]=A[N-1];`

**Цикл:**

почему не N?

```
for (i=0; i<N-1; i++)
  A[i]=A[i+1];
```


Что неверно?

```
main()
{
  const int N = 10;
  int A[N], i, c;
  // заполнить массив
  // вывести исходный массив
  c=A[0];
  for (i=0; i<N-1; i++)
    A[i]=A[i+1];
  A[N-1]=c;
  // вывести полученный массив
}
```

# Двумерный массив

В двумерном массиве, кроме количества элементов массива, есть такие характеристики как, количество строк и количество столбцов двумерного массива.

В объявлении  двумерного массива нужно указать:

- тип данных;
- имя массива.

В первых квадратных скобках указывается количество строк двумерного массива, во вторых квадратных скобках — количество столбцов двумерного массива. Двумерный массив визуально отличается от одномерного второй парой квадратных скобочек.

**// пример объявление двумерного массива:**

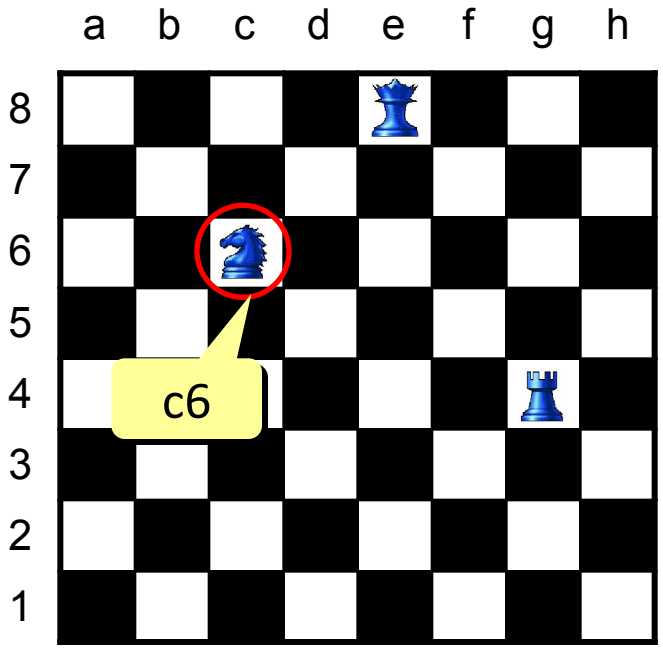
```
int a[5][3];
```

**// инициализация двумерного массива:**

```
int a[5][3] = { {4, 7, 8}, {9, 66, -1}, {5, -5, 0}, {3, -3, 30}, {1, 1, 1} };
```

# Матрицы

Задача: запомнить положение фигур на шахматной доске.


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 7 | 0 | 0 | 0 | 0 | **2** | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | **3** | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | | | | 0 | 0 | **4** | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

A[5][2]

# Двумерный массив (Матрица)

Объявление:

```
const int N = 3, M = 4;
int A[N][M];
float a[2][2] = {{3.2, 4.3}, {1.1, 2.2}};
char sym[2][2] = { 'a', 'b', 'c', 'd' };
```

Ввод с клавиатуры:

```
for (j=0; j<M; j++)
  for (i=0; i<N; i++) {
    printf("A[%d][%d]=",i,j);
    scanf("%d",&A[i][j]);
    }
```

i j

A[0][0]= 2
A[1][0]= 5
A[2][0]= 4
A[0][1]= 4
...
A[2][3]= 4

? Если переставить циклы?

# Двумерный массив (Матрица)

## Заполнение случайными числами

```
for (i=0; i<N; i++)
  for (j=0; j<M; j++)
    A[i][j] = random(25)-10;
```

цикл по строкам

цикл по столбцам

...тервал?

## Вывод на экран

```
for (i=0; i<N; i++) {
  for (j=0; j<M; j++)
    printf("%5d", A[i][j]);
  printf("\n");
  }
```

вывод строки

в той же строке

перейти на новую строку

```
   12   25    1   13
  156    1   12  447
    1  456  222   23
```

? Если переставить циклы?

# Пример: Нахождение суммы элементов матрицы

```
main()
{
  const int N=3, M=4;
  int A[N][M], i, j, S=0;
  ... // заполнение матрицы и вывод на экран
  for(i=0; i<N; i++)
    for(j=0; j<M; j++)
      S+=A[i][j];
  printf("Сумма элементов матрицы S=%d", S);
}
```

# Пример: Вычисление коэффициентов Бинома Ньютона

***Определение***

Многочлен вида

$$(a+b)^n = C_n^0 a^n + C_n^1 a^{n-1} b + \ldots + C_n^m a^{n-m} b^m + \ldots + C_n^{n-1} a b^{n-1} + C_n^n b^n$$

называется **биномом Ньютона**, а коэффициенты $C_n^m$ $(0 \le m \le n)$ называются **биномиальными коэффициентами**.

***Задание***

Используя два описанных ниже метода, найти значения коэффициентов разложения многочлена $(a + b)^n$ .

***Входные данные***

С клавиатуры вводится число $n$ — значение максимальной степени, для которой нужно посчитать коэффициенты бинома Ньютона.

***Выходные данные***

На экран выводятся $n + 1$ строк. Каждая $i$ – я строка ($0 \le i \le n$ ) содержит целые числа, записанные через пробел, — посчитанные биномиальные коэффициенты $C_i^k$ ($0 \le k \le i$).

# Треугольник Паскаля

Выпишем коэффициенты разложения в строчку, начиная с $n = 0, 1$ и так далее следующим образом:

| $n$ | Коэффициенты |
|---|---|
| 0 | 1 |
| 1 | 1 1 |
| 2 | 1 2 1 |
| 3 | 1 3 3 1 |
| 4 | 1 4 6 4 1 |
| 5 | 1 5 10 10 5 1 |
| 6 | 1 6 15 20 15 6 1 |
| 7 | 1 7 21 35 35 21 7 |
| ... | ..... |

Для каждого коэффициента можно записать следующие рекуррентные соотношения:

$$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}, \quad n > 1,\ 0 < m < n;$$

$$C_n^0 = C_n^n = 1, \quad n \geq 0.$$

# Определение числа сочетаний с помощью рекурсии

Биномиальные коэффициенты также можно вычислить по следующей формуле (число сочетаний из *n* по *k*):

$$C_n^k = \frac{n!}{k!(n-k)!}$$

где выражение *n!* (*n*-факториал) обозначает произведение всех натуральных чисел от 1 до *n*.

Исходя из соотношений:

$0! = 1,$

$n! = n*(n-1)! \quad (n > 0)$

можно написать рекурсивную функцию вычисления факториала, а затем использовать ее для вычисления биномиальных коэффициентов по приведенной выше формуле.

```
int fact(int n)
{
    if (!n)  return 1;
    return (n * fact(n-1));
}
```

# Чем плох массив символов?

Это массивы символов:

```
char A[4] = { 'A', '3', '[', 'Ж'};
char B[10];
```

Для массива:

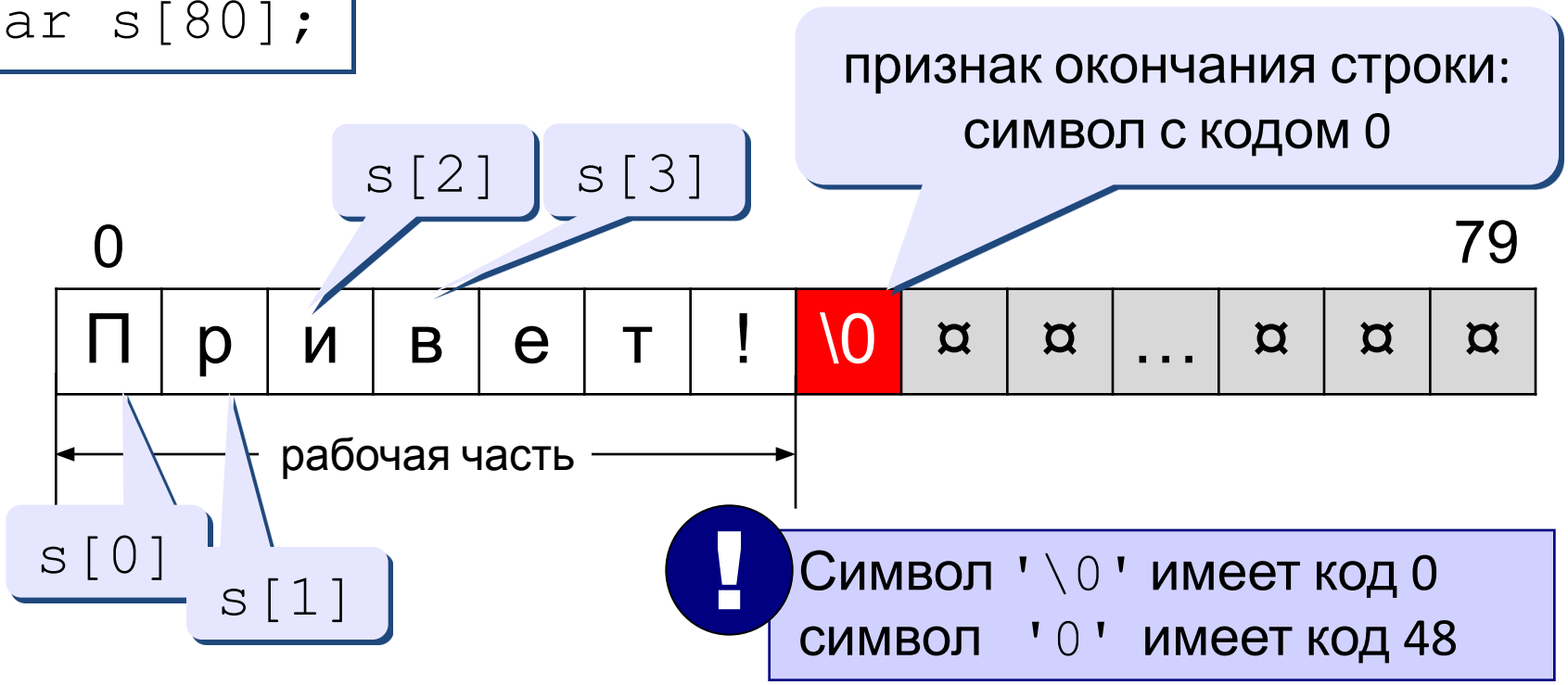
- каждый символ – отдельный объект;
- массив имеет длину N, которая задана при объявлении

Что нужно:

- обрабатывать последовательность символов как единое целое
- строка должна иметь переменную длину

# Символьные строки

```
char s[80];
```




**Символьная строка** – это последовательность символов, заключенная в двойные кавычки, которая заканчивается символом `'\0'`. Строка символов хранится в памяти ЭВМ как массив символов. Значение символьной строки - это адрес ее первого символа. При трансляции программы компилятор разместит все символьные строки в памяти, а в программу вместо них подставит соответствующие адреса (т.е. значения символьных строк!).

# Объявление символьных строк

Объявить строку = выделить ей место в памяти и присвоить имя.

```
char s[80];
```

выделяется 80 байт, в строке – «мусор» (если она *глобальная,* то нули '\0‘)

```
char s1[80] = "abc";
```

выделяется 80 байт, занято 4 байта (с учетом '\0')

```
char qqq[] = "Вася";
```

выделяется 5 байт (с учетом '\0')

!

- При выделении памяти надо учитывать место для символа '\0'.
- В строку нельзя записывать больше символов, чем выделено памяти.

# Ввод и вывод символьных строк

Задача: ввести слово с клавиатуры и заменить все буквы «а» на буквы «б».

```
main()
{
  char q[80];
  printf("Введите
  scanf( "%s", q);
  i = 0;
  while ( q[i] != '\0' ) {
    if ( q[i] == 'a' ) q[i] = 'б';
    i ++;
    }
  printf ("Результат: %s", q);
}
```

%s – формат для ввода и вывода символьных строк (выводится только часть до '\0'

начали с q[0]

не надо ставить &: &q[0]

пока не дошли до конца строки

переход к следующему символу

# Ввод символьных строк

Ввод одного слова:

```
char q[80];
printf("Введите текст:\n");
scanf("%s", q);
printf("Введено:\n%s", q);
```

```
Введите текст:
Вася пошел гулять
Введено:
Вася
```

Ввод строки с пробелами:

```
char q[80];
printf("Введите текст:\n");
gets(q);
printf("Введено:\n%s", q);
```

```
Введите текст:
Вася пошел гулять
Введено:
Вася пошел гулять
```

# Вывод символьных строк

Универсальный способ:

```
printf("Результат: %s", q);
```

- можно выводить сразу и другую информацию: надписи, значения переменных, …

Только для одной строки:

```
puts(q);
```

```
printf("%s\n", q);
```

- вывод только одной строки
- после вывода – переход на новую строку

# Функции для работы со строками

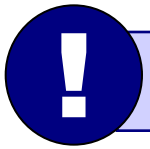
Подключение библиотеки:

```
#include <string.h>
```

Длина строки: *strlen* (*string length*)

```
char q[80] = "qwerty";
int n;
n = strlen ( q );
```

n = 6

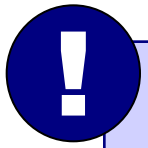


При определении длины символ `'\0'` не учитывается!

# Сравнение строк

***strcmp*** **(*string comparison*):**

```
char q1[80], q2[80];
int n;
gets ( q1 );
gets ( q2 );
n = strcmp ( q1, q2 );
```



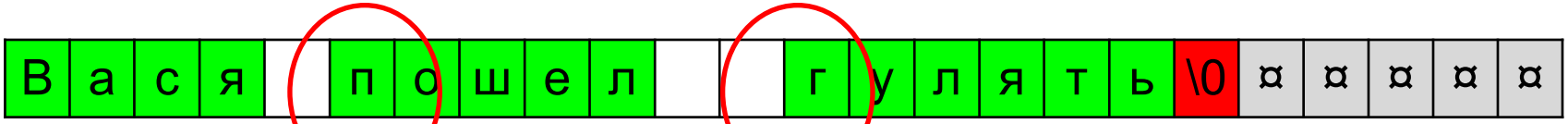
Функция вычисляет разность между кодами первых двух отличающихся символов!

# Пример решения задачи

**Задача:** ввести строку и определить, сколько в ней слов. Программа должна работать только при вводе правильного пароля.

**Идея решения:**

- проверка пароля – через ***strcmp***
- количество слов = количеству первых букв слова
- первая буква: пробел и за ним «не пробел»




- исключение: предложение начинается со слова (а не с пробела)

# Проверка пароля

```
#include <string.h>
main()
{
 char secret[]="123", pass[20];
 printf("Введите пароль\n");
 gets(pass);
 if ( strcmp(pass,secret) != 0 )
    {
    printf("Пароль неверный");
    getch();
    return 1;
    }
 ...
}
```

если пароль неверный...

сообщить об ошибке и выйти из программы

аварийное завершение, код ошибки 1

# Основная часть программы

```
#include <stdio.h>
#include <string.h>
main()
{
 char q[80];
 int i, len, count = 0;
 ...  // проверка пароля
 printf("Введите предложение\n");
 gets(q);
 len=strlen(q);
 if(q[0]!=' ')count++;
 for(i=0; i<len-1;i++)
   if(q[i]==' ' && q[i+1]!=' ')
     count++;
 printf("Найдено %d слов", count);
}
```

предыдущий слайд

особый случай

если нашли пробел, а за ним не пробел…

# Копирование строк

## *strcpy* (*string copy*)

```
char q1[10]="qwerty", q2[10]="01234";

strcpy ( q1, q2 );
```

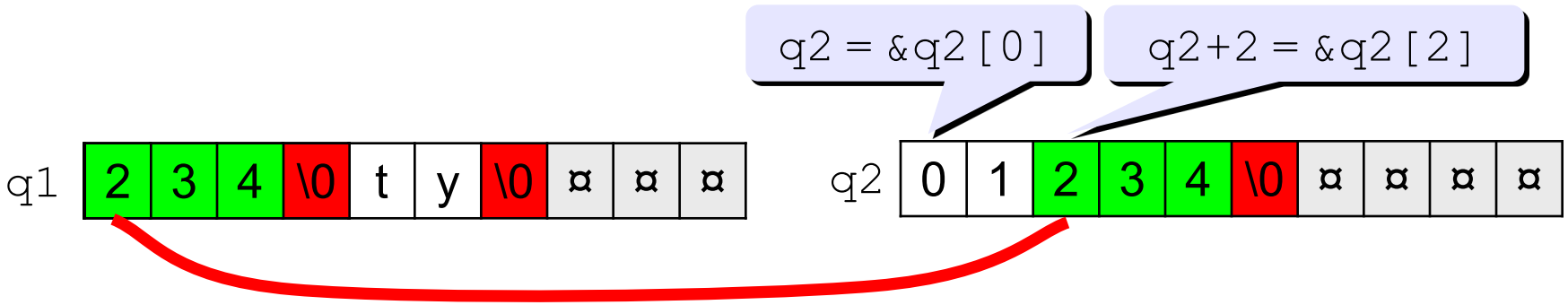
куда

откуда

Старое значение `q1` стирается!

копирование «хвоста» строки

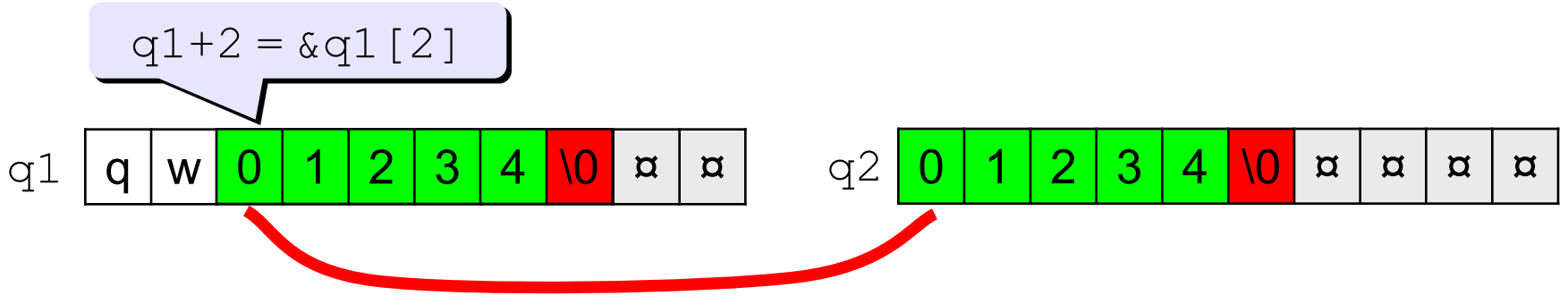
```
char q1[10]="qwerty", q2[10]="01234";
strcpy ( q1, q2+2 );
```



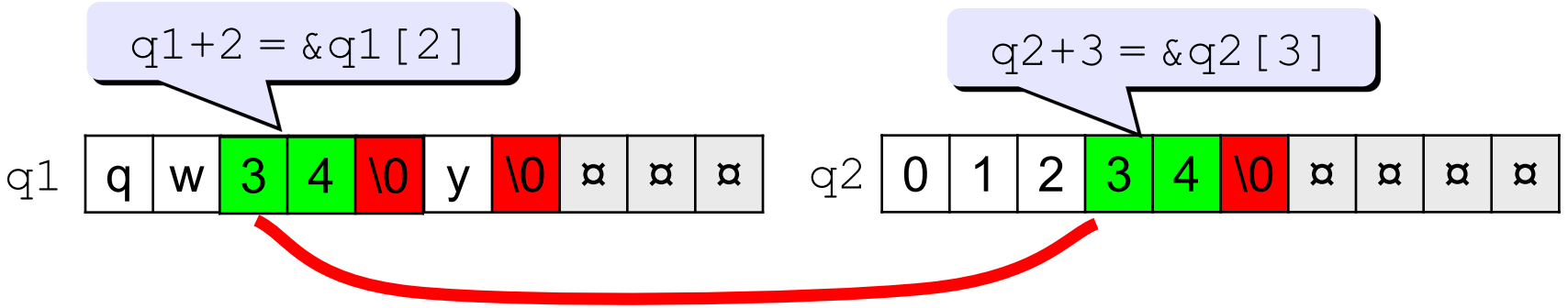
# Копирование строк

копирование в середину строки

```
char q1[10]="qwerty", q2[10]="01234";
strcpy ( q1+2, q2 );
```



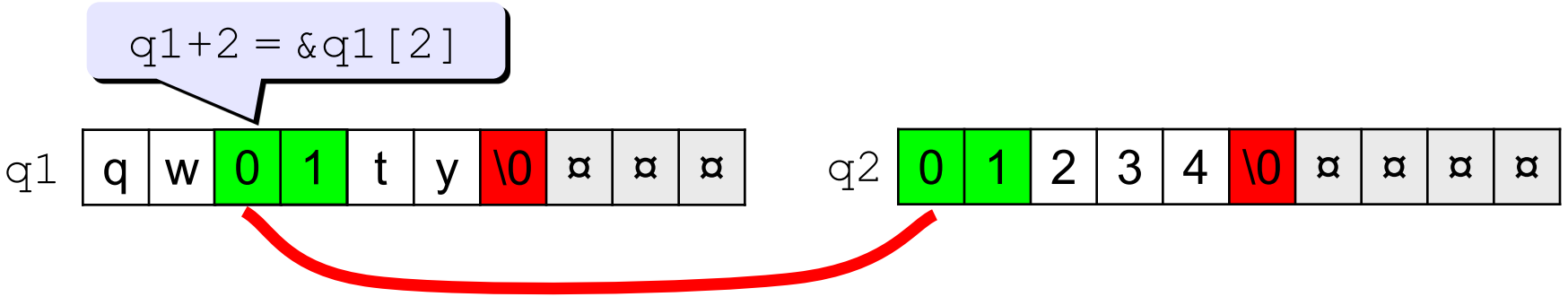

```
char q1[10]="qwerty", q2[10]="01234";
strcpy ( q1+2, q2+3 );
```



# Копирование строк

## *strncpy* – копирование нескольких

```
C char q1[10]="qwerty", q2[10]="01234";
  strncpy ( q1+2, q2, 2 );
```

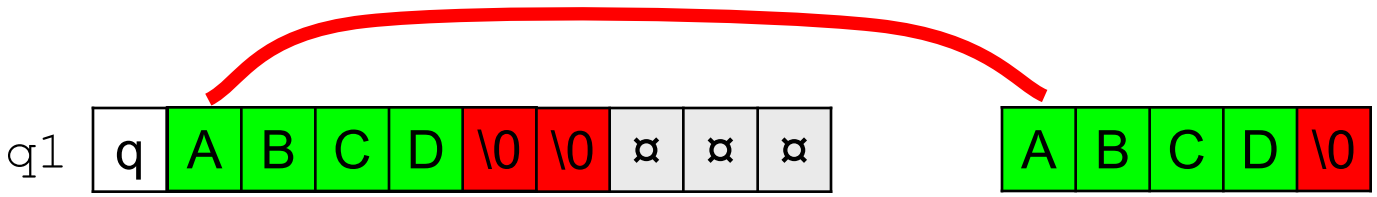



Функция `strncpy` не добавляет символ `'\0'` в конце строки!

# Копирование строк

копирование строки-константы

```
char q1[10]="qwerty";
strcpy ( q1+1, "ABCD");
```



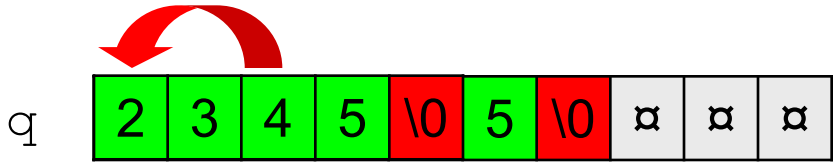

```
char q1[10]="qwerty";

strcpy ( "ABCD", q1+2 );
```

Первым параметром НЕ может быть константа!

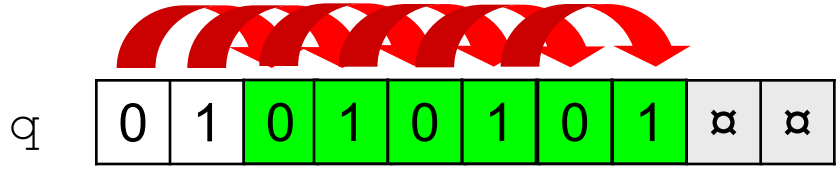
# Копирование строк

копирование внутри одной строки

```
char q[10]="012345";
strcpy ( q, q+2 );
```




```
char q[10]="012345";
strcpy ( q+2, q );
```

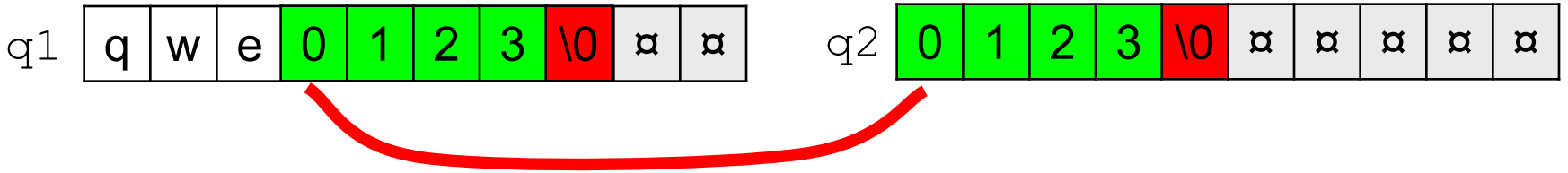



Зацикливание!

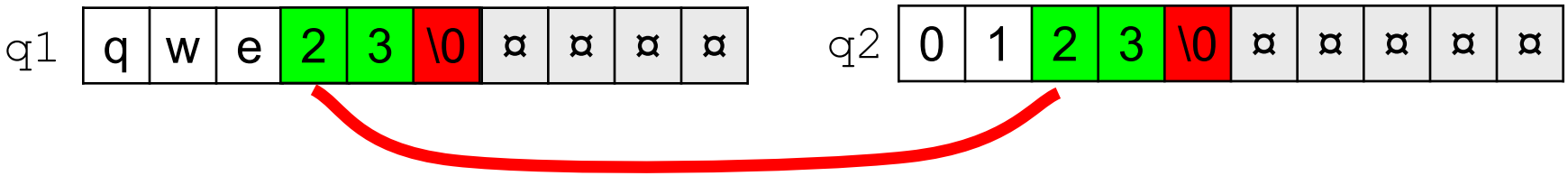
# Объединение строк

***strcat*** (***string concatenation***) = **копирование второй строки в конец первой**

```
char q1[10]="qwe", q2[10]="0123";
strcat ( q1, q2 );
```



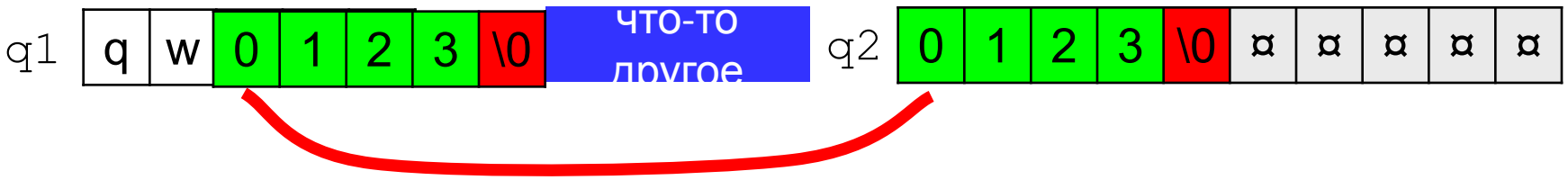

```
char q1[10]="qwe", q2[10]="0123";
strcat ( q1, q2+2 );
```



# Проблемы при копировании строк

- не хватает места для строки-результата

```
char q1[]="qwer", q2[10]="01234";
strcpy ( q1+2, q2 );
```




- зацикливание при копировании в ту же строку «слева направо»

```
char q[10]="01234";
strcpy ( q+2, q );
```