Строки

Алтайский государственный университет
Факультет математики и ИТ
Кафедра информатики

Барнаул 2014

Лекция 12

- Строки: общие сведения
- Функции для работы со строками
- Строки как параметры функций

Несколько заданий для самопроверки

Задание 1

- **Есть ли в следующем фрагменте кода ошибка и, если есть, в чем она состоит?**

```
struct outer{
    int a;
    struct inner{
        char c;
    };
};
```

(a) В Си не разрешены вложенные структуры

(b) Необходимо инициализировать поля структуры

✔ **(c) Внутренняя структура должна иметь имя**

(d) Внешняя структура должна иметь имя

(e) Нет никаких ошибок

Задание 2

- Что выведет на экран следующая программа?

```
#include <stdio.h>

struct node {int a; int b; int c;};

void main() {
  struct node s={2,5,7};
  struct node *p=&s;
  printf("%d",*((int*)p));
}
```

```
2
```

Строки: общие сведения

- Массивы символов
- Символьные строки
- Объявление строк
- Указатели и строки
- Ввод и вывод строк

Чем плох массив символов?

Это массивы символов:

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

Для массива:

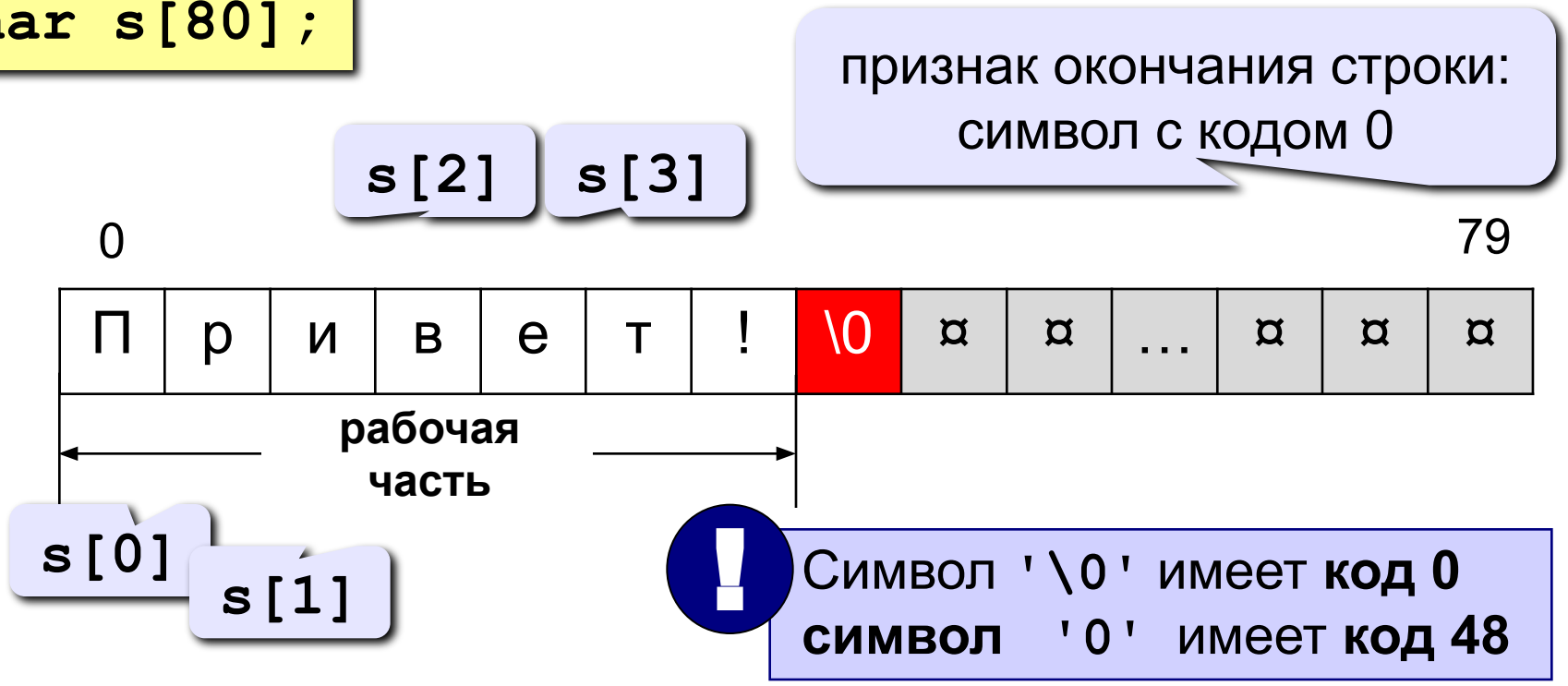
- каждый символ – отдельный объект;
- массив имеет длину N, которая задана при объявлении

Что нужно:

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

Символьные строки

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



Символьная строка – это последовательность символов, которая заканчивается символом `'\0'`.

Объявление символьных строк

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

```
char s[80];

char s1[80] = "abc";

char qqq[] = "Вася";
```

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

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

выделяется 5 байт (с учетом '\0')

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

Указатели и символьные строки

```
char str[10] = "0123456";
char *p;                    /* указатель на символ */
p = str;                    /* или & str[0] */
*p = 'A';                   /* "A12345" */
p ++;                       /* перейти к str[1] */
*p = 'B';                   /* "AB2345" */
p ++;                       /* перейти к str[2] */
strcpy ( p, "CD" );         /* "ABCD" */
strcat ( p, "qqq" );        /* "ABCDqqq" */
puts ( p );
```

Ввод и вывод символьных строк

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

```
void main()
{
  char q[80];
  int i;
  printf("Введите с
  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

- Длина строки
- Сравнение строк
- Копирование строк
- Объединение строк
- Поиск в строке

Функции для работы со строками

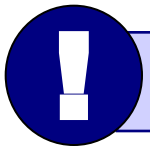
Подключение библиотеки:

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

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

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

n = 6

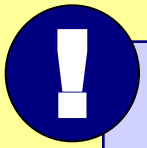


При определении длины символ `'\0'` не учитывается!

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!

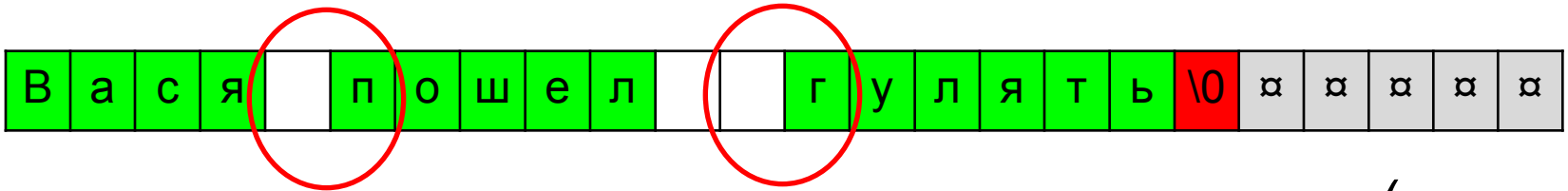
q1	q2	n
"AA"	"AA"	

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>
void 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>
void 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

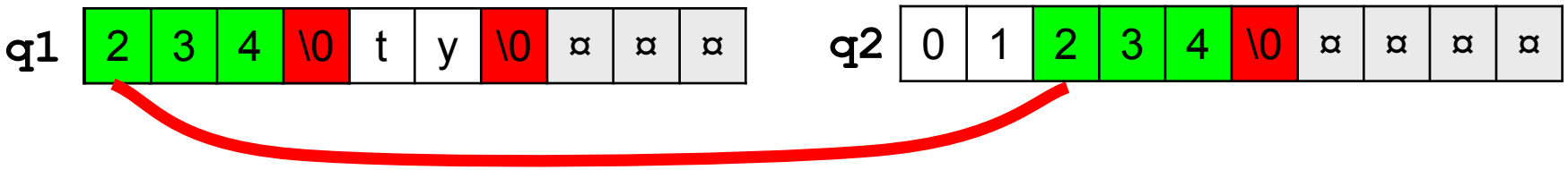
Старое значение q1 стирается!

копирование «хвоста» строки

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

q2 = &q2[0]

q2+2 = &q2[2]

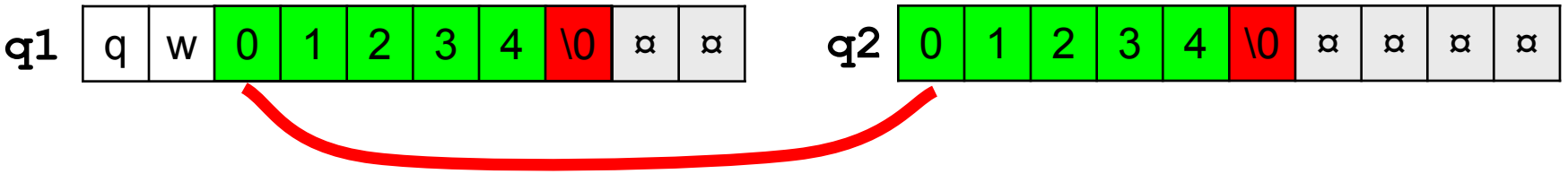


Копирование строк

копирование в середину строки

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

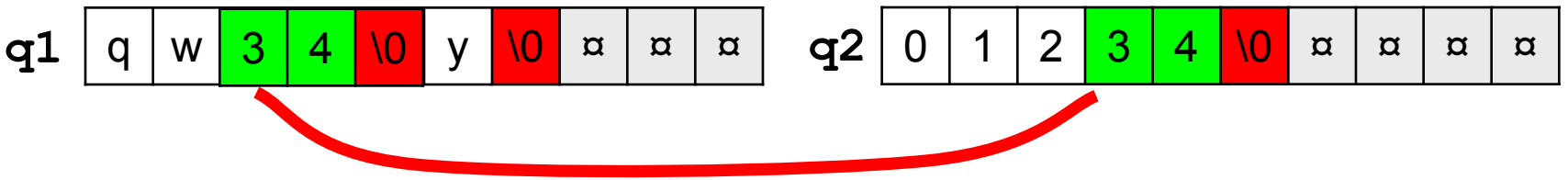
q1+2 = &q1[2]




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

q1+2 = &q1[2]

q2+3 = &q2[3]

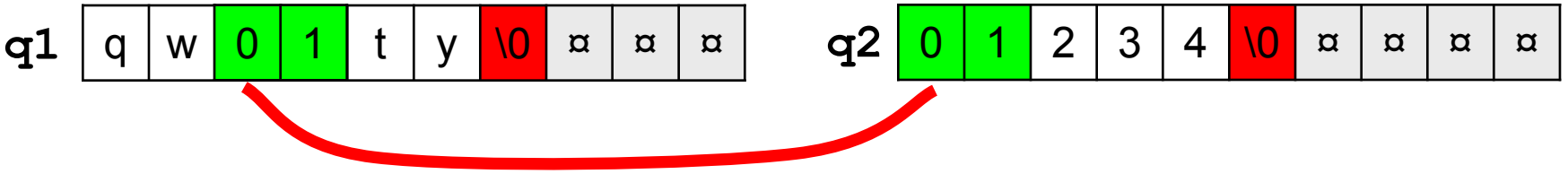


Копирование строк

strncpy – копирование нескольких символов

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

q1+2 = &q1[2]

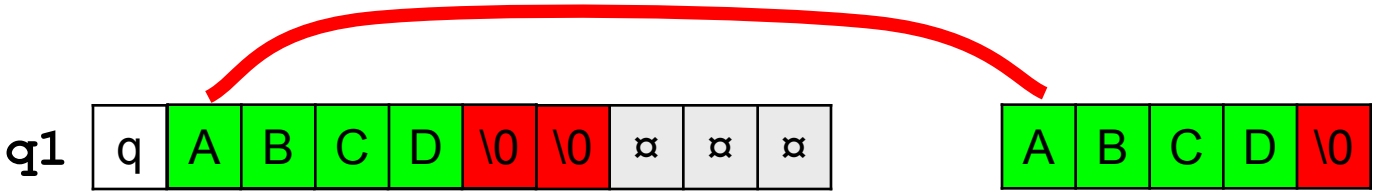


Функция `strncpy` не добавляет символ `'\0'` в конце строки!

Копирование строк

копирование строки-константы

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



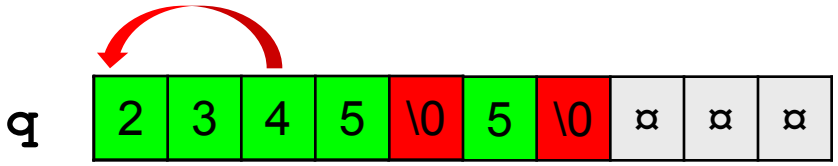

```
char q1[10] = "qwerty";
strcpy ( "ABCD", q1+2 );
```

Первым параметром НЕ может быть константа!

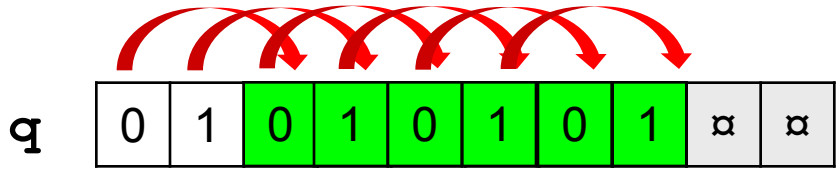
Копирование строк

копирование внутри одной строки

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




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

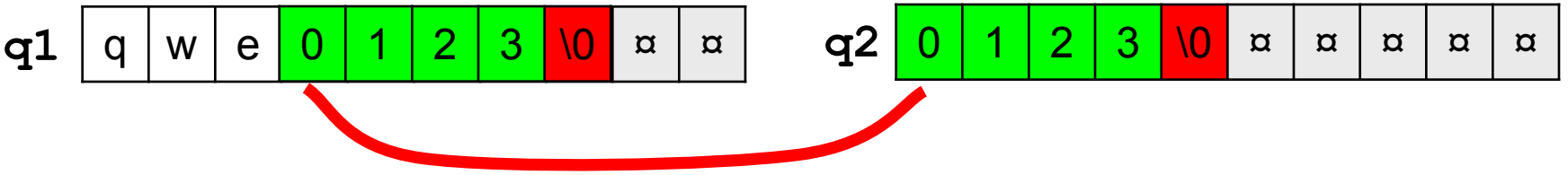


Зацикливание и зависание компьютера!

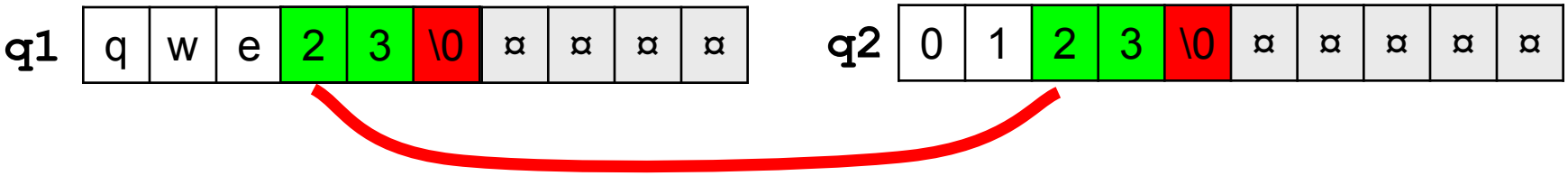
Объединение строк

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

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



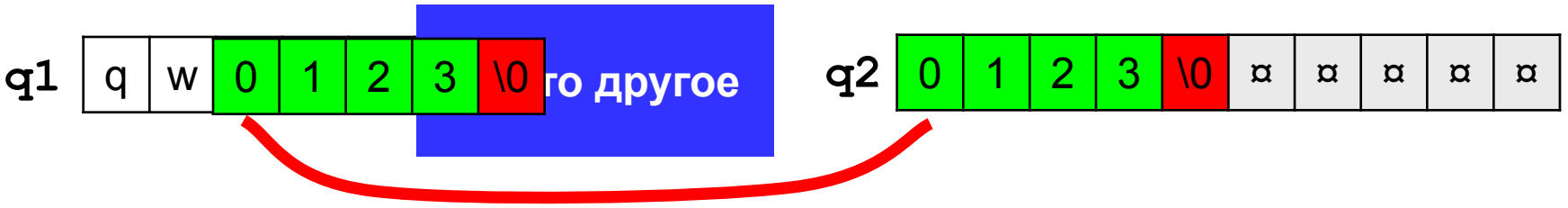

```
char q1[10]="qwe", q2[10]="0123";
strcat ( q1, q2+2 );
```



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 );
```

Транслятор не сообщает об этих ошибках!

Пример решения задачи

Задача: ввести имя файла (без пути) и поменять его расширение на **".exe"**.

Пример:

```
Введите имя файла:          Введите имя файла:
vasya.html                  vasya
Результат:                  Результат:
vasya.exe                   vasya.exe
```

Алгоритм:

- найти точку в имени файла
- если она есть, скопировать в это место строку-константу "**.exe**"
- если точки нет, добавить в конец строки "**.exe**"

Программа

```
void main()
{
char  fName[80];
int  i;
printf("Введите имя файла\n");
gets ( fName );
i = 0;
while ( fName[i] != '.' )  {
    if ( fName[i] == '\0' ) break;
    i ++;
    }
if ( fName[i] == '.' )
      strcpy ( fName+i, ".exe" );
else strcat ( fName, ".exe" );
puts ( "Результат:" );
puts ( fName );
}
```

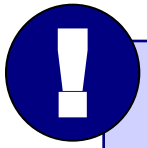
поиск точки

дошли до конца строки

меняем или добавляем расширение

Поиск в символьных строках

Задача: найти заданный символ или сочетание символов (**подстроку**) в символьной строке.



Функции поиска в Си возвращают адрес найденного символа или подстроки!

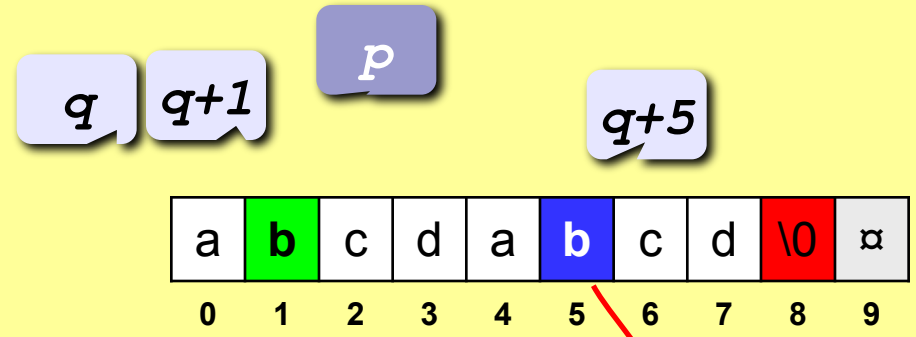
Если образец не найден, возвращается **NULL** (нулевой адрес).

Указатель – это переменная в которую можно записать адрес другой переменной заданного типа.

Поиск символа

strchr: найти первый заданный символ с начала строки

```
char q[10] = "abcdabcd";
char *p;
int nomer;
p = strchr(q, 'b');
if ( p == NULL )
  printf ( "Не нашли..." );
else {
  nomer = p - q;
  printf ( "Номер символа %d", nomer );
}
```



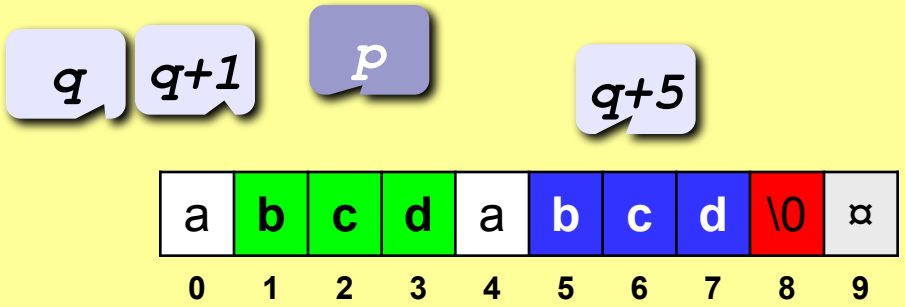
reverse

strrchr: найти последний заданный символ в строке

Поиск подстроки

strstr: найти первую подстроку с начала строки

```
char q[10] = "abcdabcd";
char *p;
int nomer;
p = strstr(q, "bcd");
if ( p == NULL )
  printf ( "Не нашли..." );
else {
  nomer = p - q;
  printf ( "Номер первого символа %d", nomer );
}
```



Пример решения задачи

Задача: ввести предложение и определить, сколько раз в нем встречается имя «Вася».

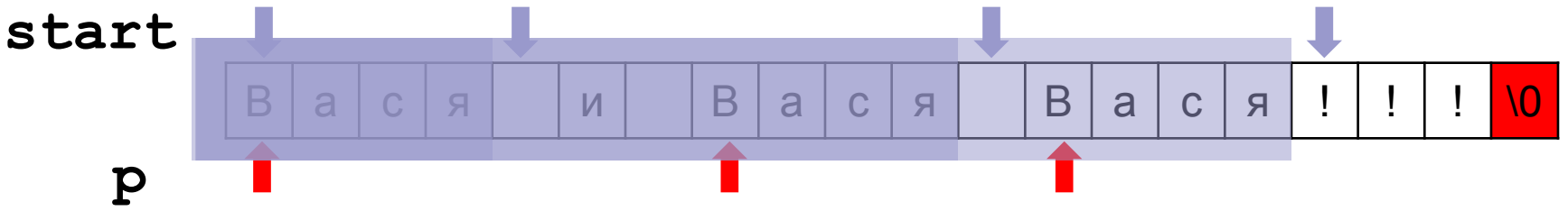
Проблема: функция ***strstr*** ищет только с начала строки.

Алгоритм:

1. Записать адрес начала строки в указатель **start**.
2. Искать подстроку «Вася», начиная с адреса **start**.

```
p = strstr( start, "Вася");
```

3. Если не нашли, выход из цикла.
4. Увеличить счетчик найденных слов.
5. Переставить **start** на адрес после найденного слова.
6. Перейти к шагу 2.



Программа

```
void main()
{
  char q[80], *start, *p;
  int count = 0;
  puts ( "Введите предложение" );
  gets ( q );
  start = q; /* ищем с начала строки */
  while ( 1 ) {
    p = strstr ( start, "Вася" );
    if ( p == NULL ) break;
    count ++;
    start = p + 4; /* отсюда ищем следующее слово */
  }
  printf ( "Имя 'Вася' встречается %d раз", count );
}
```

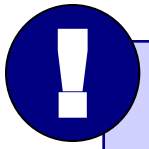
начало поиска

адрес найденного слова

Строки как параметры функций

- Передача параметров-строк
- Примеры функций со строковыми параметрами

Символьные строки в функциях



- строки передаются в функции так же, как и массивы;
- функции могут изменять строки-параметры.

Задача: составить процедуру, которая переставляет символы строки в обратном порядке.

Алгоритм:

- определить длину строки **len;**
- все символы первой половины переставить с соответствующими символами второй половины:

`s[i]` ⟺ `s[len-1-i]`

```
c = s[i];
s[i] = s[len-i-1];
s[len-1-i] = c;
```

Программа

```
void Reverse ( char s[] )
{
  int len = strlen(s);
  char c;
  for ( i = 0; i < len/2; i ++ ) {
    c = s[i];
    s[i] = s[len-i-1];
    s[len-1-i] = c;
  }
}

void main(){
  char s[] = "1234567890";
  Reverse ( s );
  puts ( s );
  Reverse ( s + 5 );
  puts ( s );
}
```

длину строки определяем на месте

Как сделать инверсию **любой** части строки?

```
0987654321
```

```
0987612345
```

Символьные строки в функциях

Задача: составить функцию, которая находит количество цифр в строке.

```
int NumDigits ( char s[] )
{
  int i, count = 0;
  for ( i = 0; i < strlen(s); i ++ )
    if( strchr ( "0123456789", s[i] ) )
      count ++;
  return count;
}
```

```
if ( strchr ( "0123456789", s[i] ) != NULL )
```

или

```
if ( '0' <= s[i] && s[i] <= '9' )
```

Символьные строки в функциях

Основная программа

```
int NumDigits ( char s[] )
{
...
}
void main()
{
  char s[80];
  int n;
  printf ( "Введите строку\n" );
  gets ( s );
  n = NumDigits ( s );
  printf ( "Нашли %d цифр.", s );
}
```

Вопросы?

- Строки: общие сведения
  - Массивы символов
  - Символьные строки
  - Объявление строк
  - Указатели и строки
  - Ввод и вывод строк
- Функции для работы со строками
  - Длина строки
  - Сравнение строк
  - Копирование строк
  - Объединение строк
  - Поиск в строке
- Строки как параметры функций
  - Передача параметров-строк
  - Примеры функций со строковыми параметрами

Дубовая роща. Девочка и апельсин