# Работа со строками и файлами

Тема 8

# Строки в Си

- Строка – это массив символов, заканчивающийся символом конца строки `'\0'`

```
char r[]={'A','B','C','D','E','F','\0'};
char s[] = "ABCDEF";
```

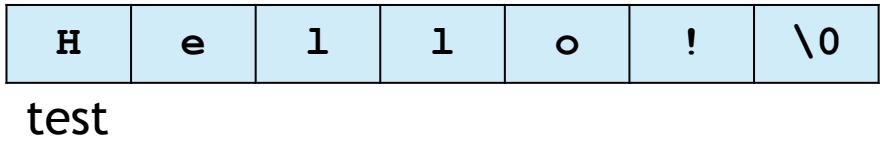
Объявление строк в формате Си

Объявления r и s одинаковы, но s - короче

# Разница в объявлении

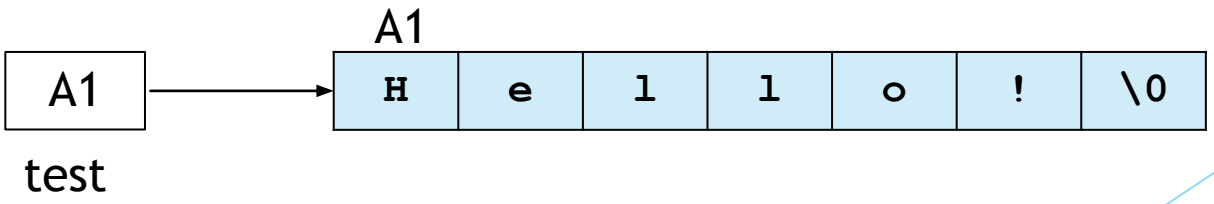
- Объявляем символьный массив

```
char test[] = "Hello!";
```




- Объявляем указатель

```
const char *test = "Hello!";
```



# Разница в объявлении

- Объявляем символьный массив

```
char test[] = "Hello!";
test[0] = 'h';          Можно!!!
test = "world";         Нельзя!!!
puts(test);
```

hello!

- Объявляем указатель

```
const char *test = "Hello!";
test = "world";         Можно!!!
test[0] = 'W';          Нельзя!!!
puts(test);
```

world

# Функции для работы со строками в Си (<string.h>)

- Возвращает длину строки без учета символа конца строки

```
int strlen(const char* s);
```

- Добавляет строку **source** в конец строки **dest** (всю или **n** символов)

```
char * strcat(char * dest, const char * source);
char * strncat(char * dest, const char * source, int n);
```

- Копирует строку **source** в строку **dest** (всю или **n** символов)

```
char * strcpy(char * dest, const char * source);
char * strncpy(char * dest, const char * source, int n);
```

- Сравнивает строки (целиком или первые **n** символов)

```
int strcmp(const char * s1, const char * s2);
int strncmp(const char * s1, const char * s2, int n);
```

- Ищет подстроку в строке

```
char * strstr(const char * s1, const char * s2);
```

# Определение длины строки

```
char *myString = "Hello, world!";
printf("%d\n", strlen(myString) );
```

13

# Сложение двух строк (конкатенация)

```
char destination[25]={0};
char *blank = " ", *c = "C++", *vis = "Visual";
strcat(destination, vis);
strcat(destination, blank);
strcat(destination, c);
puts(destination);
```

Visual C++

Ответственность за то, что в массиве **destination** хватит памяти, лежит на программисте, вызывающем функцию **strcat**

# Добавление к строке указанного количества символов

```
char myString[80] = "This is the initial string!";
char suffix[] = " extra text to add to the string...";
printf("Before: %s\n", myString);
strncat( myString, suffix, 19 );
printf("After: %s\n", myString);
```

*Before: This is the initial string!*
*After:  This is the initial string! extra text to add*

# Копирование строки в строку

```
char destination[25];
char *blank = " ", *c = "C++", *vis = "Visual";
strcpy(destination,vis);
strcpy(destination,blank);
strcpy(destination,c);
puts(destination);
```

C++

Ответственность за то, что в массиве **destination** хватит памяти, лежит на программисте, вызывающем функцию **strcpy**

# Копирование части строки в строку

```
char destination[25];
char *blank = " ", *c = "C++", *vis = "Visual";
strcpy(destination, vis);
strncpy(destination, c, 1);
puts(destination);
```

C

# Сравнение строк

```
char *buf1 = "aaa", *buf2 = "aaa";
if(buf1 == buf2)
   puts("buf2 == buf1");
else
   puts("buf2 != buf1");
buf2 = buf1;
if(buf1 == buf2)
   puts("buf2 == buf1");
else
   puts("buf2 != buf1");
```

В этой программе сравниваются адреса первых элементов двух строк, а не их содержимое

```
buf2 != buf1
```

```
buf2 == buf1
```

# Сравнение строк

- Функция **`strcmp`** сравнивает строки посимвольно
- Символы хранятся в виде целых чисел. Одни их самых популярных кодировок – ASCII и EBCDIC
- Код каждого символа одной строки сравнивается с кодом каждого символа другой строки
- Латинские буквы упорядочены по алфавиту (к кириллице это не относится), поэтому имеет смысл сравнивать строки, состоящие их латинских букв
- Цифры также упорядочены по возрастанию, от 0 до 9

# Сравнение строк

```
char *buf1 = "aaa", *buf2 = "bbb", *buf3 = "ccc";
int ptr;
ptr = strcmp(buf2, buf1);
if(ptr > 0)
   puts("buf2 > buf1");
else
   puts("buf2 <= buf1");
ptr = strcmp(buf2, buf3);
if(ptr > 0)
   puts("buf2 > buf3");
else
   puts("buf2 <= buf3");
```

strcmp возвращает 1, если первая строка больше второй, -1, если первая строка меньше второй, и 0, если строки эквивалентны

# Сравнение первых n символов

```
char *buf1 = "aaa", *buf2 = "bbb", *buf3 = "ccc";
int ptr;
ptr = strncmp(buf2, buf1, 2);
if(ptr > 0)
   puts("buf2 > buf1");
else
   puts("buf2 <= buf1");
ptr = strncmp(buf2, buf3, 2);
if(ptr > 0)
   puts("buf2 > buf3");
else
   puts("buf2 <= buf3");
```

# Поиск подстроки в строке

```
char myString[]= "Visual C++";

char *c = "C++";

char *res = 0;

res = strstr(myString, c);

if(res)

    printf("C++ found at %d position\n" ,
           (res-myString+1) );
```

Функция **strstr()** ищет первое вхождение указанной подстроки в строке. Если вхождение найдено, то она возвращает указатель на первый символ найденной подстроки. Если вхождение найдено не было, то возвращается нулевой указатель

# Работа с файлами в Си (<stdio.h>)

- Открытие/закрытие файла

```
FILE * fopen(char * filename, char * type);
void fclose(FILE* stream);
```

- Чтение из файла

```
fscanf(поток, шаблон, адреса)
fgetc(поток)
fgets(адрес, размер, поток)
```

- Запись в файл

```
fprintf(поток, шаблон, данные)
fputc(символ, поток)
fputs(строка, поток)
```

- Смещение внутри файла

```
int fseek(FILE * stream, long offset, int fromwhere);
```

- Расстояние от начала файла до текущей позиции

```
unsigned long ftell(FILE* stream);
```

# Открытие файла

```
FILE * fopen(char * filename, char * type);
```

Строка **type** может принимать следующие значения:

- **r** - открытие файла только для чтения;
- **w** - создание файла для записи;
- **a** - присоединение; открытие для записи в конец файла или создание для записи, если файл не существует;
- **r+** - открытие существующего файла для обновления (чтения и записи);
- **w+** - создание нового файла для изменения;
- **a+** - открытие для присоединения; открытие (или создание, если файл не существует) для обновления в конец файла

# Режим открытия файла

- Если данный файл открывается или создается в текстовом режиме, то можно приписать символ **t** к значению параметра **type** (**rt**, **w+t**, и т.д.)
- Для открытия в бинарном режиме можно к значению параметра **type** добавить символ **b** (**wb**, **a+b**, и т.д.)
- Если в параметре **type** отсутствуют символы **t** или **b**, режим будет определяться глобальной переменной **_fmode**. Если переменная **_fmode** имеет значение **O_BINARY**, файлы будут открываться в бинарном режиме, иначе, если **_fmode** имеет значение **O_TEXT**, файлы открываются в текстовом режиме. Данные константы определены в файле **fcntl.h**

# Смещение внутри файла

```
int fseek(FILE * stream, long offset, int fromwhere);
```

- Функция **fseek()** устанавливает адресный указатель файла, соответствующий потоку **stream**, в новую позицию, которая расположена по смещению **offset** относительно места в файле, определяемого параметром **fromwhere**
- Параметр **fromwhere** может иметь одно из трех значений 0, 1 или 2, которые представлены тремя символическими константами (определенными в файле **stdio.h**), следующим образом:

| Параметр | Размещение в файле frowmwhere |
|---|---|
| SEEK_SET (0) | начало файла |
| SEEK_CUR (1) | позиция текущего указателя файла |
| SEEK_END (2) | конец файла (EOF) |

# Пример работы с файлом

```
FILE *stream;
char mstring[] = "Тестовый пример";
char msg[100];
// создать файл для его изменения
stream = fopen("proba.txt", "w+");
// записать в файл данные
fputs(mstring, stream);
// перейти в начало файла
fseek(stream, 0, SEEK_SET);
// вывести строку из файла
fgets(msg, 100, stream);
// напечатать строку
puts(msg);
fclose(stream);
```

# Конец