Работа со строками и файлами

Лекция 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";
```

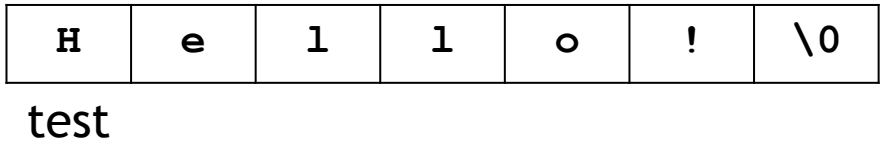
Объявление строк в формате Си

Объявления r и s одинаковы, но s - короче

Разница в объявлении

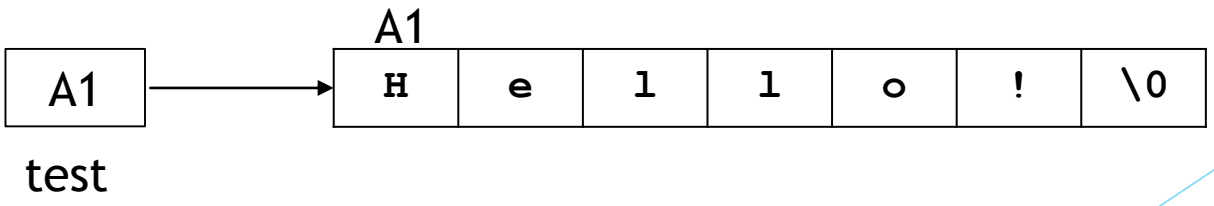
- Объявляем символьный массив

```
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!!!
std::cout << test << std::endl;
```

hello!

- Объявляем указатель

```
const char *test = "Hello!";
test = "world";         Можно!!!
test[0] = 'W';          Нельзя!!!
std::cout << test << std::endl;
```

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!";
std::cout << strlen(myString) << std::endl;
```

13

Сложение двух строк (конкатенация)

```
char destination[25]={0};
char *blank = " ", *c = "C++", *vis = "Visual";
strcat(destination, vis);
strcat(destination, blank);
strcat(destination, c);
std::cout << destination << std::endl;
```

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...";
std::cout<< "Before: " << myString << std::endl;
strncat( myString, suffix, 19 );
std::cout<< "After:  " << myString << std::endl;
```

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);
std::cout << destination << std::endl;
```

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);
std::cout << destination << std::endl;
```

C

Сравнение строк

```
char *buf1 = "aaa", *buf2 = "aaa";
if(buf1 == buf2)
   std::cout << "buf2 == buf1" << std::endl;
else
   std::cout << "buf2 != buf1" << std::endl;
buf2 = buf1;
if(buf1 == buf2)
   std::cout << "buf2 == buf1" << std::endl;
else
   std::cout << "buf2 != buf1" << std::endl;
```

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)
   std::cout << "buf2 > buf1" << std::endl;
else
   std::cout << "buf2 < buf1" << std::endl;
ptr = strcmp(buf2, buf3);
if(ptr > 0)
   std::cout << "buf2 > buf3" << std::endl;
else
   std::cout << "buf2 < buf3" << std::endl;
```

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)
   std::cout << "buf2 > buf1" << std::endl;
else
   std::cout << "buf2 < buf1" << std::endl;
ptr = strncmp(buf2, buf3, 2);
if(ptr > 0)
   std::cout << "buf2 > buf3" << std::endl;
else
   std::cout << "buf2 < buf3" << std::endl;
```

Поиск подстроки в строке

```
char myString[]= "Visual C++";
char *c = "C++";
char *res = 0;
res = strstr(myString, c);
if(res)
    std::cout << "C++ found at "  <<
                 (res-myString+1) << " position" <<
                  std::endl;
```

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 со строками в C++

- Для работы со строками разработан класс **`string`**. Чтобы его использовать, нужно подключить библиотеку **`<string>`**
- Класс находится в стандартном пространстве имен **`std`**

```
std::string s0 = "abcde";
std::string s1 = "fg";
```

Объявление двух строк

Определение длины строки

- Для определения длины строки в классе string предусмотрена функция length

```
std::string s0 = "abcde";
std::string s1 = "fg";
std::cout << s0.length() << std::endl;
std::cout << s1.length() << std::endl;
```

```
5
2
```

Сложение двух строк (конкатенация)

- Складывать две строки можно как обычные переменные, используя оператор +

```
std::string s0 = "abcde";
std::string s1 = "fg";
std::string s = s0 + s1;
std::cout << s << std::endl;
```

```
abcdefg
```

Копирование строки в строку

- Для копирования содержимого одной строки в другую строку достаточно выполнить операцию присваивания

```
std::string s0 = "abcde";
std::string s1 = "fg";
s1 = s0;
std::cout << s0 << std::endl;
std::cout << s1 << std::endl;
```

```
abcde
abcde
```

Сравнение строк

- Строки можно сравнивать как обычные переменные с помощью операций сравнения

```
std::string s0 = "abcde";
std::string s1 = "fg";
if(s0 == s1)
    std::cout << "s0 == s1" << std::endl;
else
    std::cout << "s0 != s1" << std::endl;
s1 = s0;
if(s0 == s1)
    std::cout << "s0 == s1" << std::endl;
else
    std::cout << "s0 != s1" << std::endl;
```

s0 != s1

s0 == s1

Поиск подстроки в строке

- Поиск подстроки осуществляется функцией **`find()`** класса **`String`**
- Второй параметр функции **`find()`** - индекс начала поиска.
- Функция возвращает индекс первого вхождения подстроки

```
std::string s0 = "abcde";
std::string s1 = "fg";
int a = s0.find("cd", 0);
std::cout << a << std::endl;
a = s1.find("cd", 0);
std::cout << a << std::endl;
```

```
2
-1
```

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);
```

Работа с файлами в C++ (<fstream>)

- Работа с файлами в C++ осуществляется так же, как с потоками ввода/вывода
- Для чтения из файла существует поток **ifstream**
- Для записи в файл используется поток **ofstream**
- Существует также поток **fstream**, который можно использовать и для ввода, и для вывода

Запись в файл двух строк

```
std::string s0 = "abcde";
std::string s1 = "fg";
//открытие файла для записи
std::ofstream stream("test.txt");
//запись двух строк
stream << s0 << std::endl;
stream << s1 << std::endl;
//закрытие файла
stream.close();
```

Чтение строк из файла

```
std::string s;
//открытие файла для чтения
std::ifstream stream("test.txt");
//смещение на 2 позиции вправо от начала файла
stream.seekg(2, stream.beg);
//чтение строки
stream >> s;
//вывод на экран
std::cout << s << std::endl;
//чтение следующей строки и вывод ее на экран
stream >> s;
std::cout << s << std::endl;
```

```
cde
```

```
fg
```

Запись в файл переменных разных типов

```
std::ofstream stream("proba.txt");

int a = 10;

float pi = 3.14;

char* mstr = "Hello, world!";

stream << a    << std::endl <<
          pi   << std::endl <<
          mstr << std::endl;

stream.close();
```

Чтение из файла переменных разных типов

```
ifstream stream("proba.txt");
int a = 0;
float pi = 0;
char mstr[100] = {0};
stream >> a >> pi;
stream.getline(mstr, 100);
std::cout << a    << std::endl <<
             pi   << std::endl <<
             mstr << std::endl;
stream.close();
```

```
10
3.14
Hello, world!
```

Конец