Операции с текстовыми строками

- Строка представляет собой массив символов типа **char**, ограниченный нулем. Операции обработки строк по своей сути очень просты и, поэтому, удобны для:
- изучения основ алгоритмизации и программирования. Рассмотрим в качестве примера функцию, которая определяет длину строки (завершающий **ноль не** учитывается):

Операции с текстовыми строками

```
int Len(char *string)
{
int j=0;

while(string[j] != 0) j++;

return j;
}
```

строками

- Вданном примере индекс массива инкрементируется в цикле до тех пор, пока не найден символ завершения строки. При выходе функция возвращает индекс последнего найденного ненулевого элемента массива. Так же, как стандартная функция **strlen**, функция **Len** при определении длины строки не учитывает завершающий символ.

Операции с текстовыми строками

- Если для выполнения некоторой операции над массивом используется подпрограмма, то в качестве аргумента в нее передается не сам массив, а только указатель на его местоположение в памяти, В приведенном примере передается указатель на текстовую строку **string**.

Операции с текстовыми строками

- При выполнении операции вставки символа в строку необходимо освободить место для символа, подвинув на одну позицию вправо все символы строки,
- расположенные за позицией вставки (включая завершающий 0).
- Перестановка символов начинается с конца строки, поэтому должен использоваться цикл с декрементом счетчика.

Операции с текстовыми строками

- Если требуется вставить в строку слово длинной **k** символов, то перед началом копирования слова необходимо освободить для него место, подвинув на **k** позиций вправо все символы строки, расположенные за позицией вставки.
- Если требуется добавить символ в конец строки, то проводить перестановку элементов строки не нужно: новый символ записывается за последним символом строки, а в следующую позицию заносится нулевой символ окончания строки **('\0')**.

Операции с текстовыми строками

- При выполнении операции удаления символа из строки все элементы (включая символ окончания строки), расположенные после удаляемого элемента, сдвигаются на одну позицию влево.

Операции с текстовыми строками

- **Текст** в простейшем случае представляет собой двумерный массив символов, содержащий строки.
- **Недостатки такого примитивного способа хранения данных**: неэффективное использование памяти и необходимость перемещения большого объема информации при добавлении и удалении строк.

Стандартные функции для обработки строк

- Язык **Си** долгое время использовался для создания текстовых редакторов, поэтому он располагает широким набором средств для обработки текстовой информации.
- языке **Си** имеется набор стандартных функций для работы с текстовыми строками. Описания строковых функций находятся в заголовочном файле **string.h**, который необходимо подключить директивой **#include** в начале программы..
- Рассмотрим функции, которые используются наиболее часто.

Стандартные функции для обработки строк

- Функция **strcat** добавляет одну строку к другой. Функция объявлена следующим образом:
- **char *strcat(char *dest, const char *src);**
- Функция добавляет копию строки **src** в конец **dest** и возвращает указатель на результирующую строку.

Стандартные функции для обработки строк

- Функция **strchr** ищет в строке первое вхождение заданного символа. Функция объявлена следующим образом:
- **char *strchr(const char *s, int c);**
- Функция ищет первое вхождение символа **c** в строку **s.** Она возвращает указатель на первое вхождение символа **c** в **s**; если **c** не обнаружен в **s,** то **strchr** возвращает
- **NULL.**

Стандартные функции для обработки строк

- Функция **strcmp** сравнивает одну строку с другой. Функция объявлена следующим образом:
- **int strcmp(const char *s1, const char *s2);**
- Функция **strcmp** осуществляет сравнение строк **s1** и **s2**, начиная с первого символа каждой строки, до тех пор, пока очередные соответствующие символы в строках не будут различны или пока не будут достигнуты концы строк. Функция возвращает отрицательное значение, если **s1** меньше чем **s2**, ноль, если **s1** равна **s2**, и положительное значение, если **s1**

Стандартные функции для обработки строк

- Функция **strcpy** копирует одну строку в другую. Функция объявлена следующим образом:
- **char *strcpy(char *dest, const char *src);**
- Функция копирует строку src в dest и возвращает **dest.**
- Функция **strlen** вычисляет длину строки. Функция объявлена следующим образом:
- **size_t strlen(const char *s);**
- Функция вычисляет длину строки **s** и возвращает полученное значение (символ конца строки не учитывается).

Стандартные функции для обработки строк

- Функция **strstr** ищет в строке вхождение заданной подстроки.
- Функция объявлена следующим образом:
- **char *strstr(const char *s1, const char *s2);**

  Функция осуществляет поиск в **s2** первого вхождения подстроки **s1**. Функция возвращает указатель на элемент в строке **s1**, с которого начинается **s2**. Если **s2** не обнаружена в **s1**, функция возвращает **NULL**.
-

Перестановки, размещения, сочетания, факториал

- Пусть ***Q*** – некоторое конечное множество, состоящее из **n** элементов:
- $Q = \{q_1, q_2, \ldots, q_n\}$.
- Будем образовывать из элементов множества ***Q*** упорядоченные множества. В качестве первого возьмем множество, в котором элементы расположены в порядке возрастания их номеров, второе образуем, поменяв местами первый и второй элементы и т.д.