Лабораторная работа 10

Цель работы

Изучение функций для работы со строками.

Задание

Разработать программу и подпрограмму (подпрограммы), выполняющую действия согласно варианту задания

№	Задание
1	Выравнивание текста по правой границе. Правая граница определяется как сумма длины максимальной строки и некоторого числа Δ>0. Каждый символ табуляции во входном файле заменяется на n пробелов (1<=n<=8).
2	Создать выходной зашифрованный файл на основе входного. Шифрование файла выполняется по следующему правилу: в каждом символе строки биты в байте меняются в следующем порядке: бит 0 – с битом 7, бит 1 – с битом 6, бит 2 – с битом 5, бит 3 – с битом 4. Выполнить дешифровку созданного файла.
3	Выравнивание текста по центру при заданной длине строки. Длина строки для центрирования определяется как сумма длины максимальной строки и некоторого числа Δ>0.
4	Формирование строк текста одинаковой длины (в два раза меньше максимальной в исходном тексте) путем разбиения или объединения строк. Каждый символ табуляции во входном файле заменяется на 3 пробела.
5	Форматирование текста по правилам: после каждого символа, не являющегося буквой или цифрой, вставляется один пробел; после точки – пробел, а следующее слово начинается с большой буквы.
6	Выравнивание строк текста до заданной (не меньше максимальной) путем добавления пробелов между словами.

Теоретическая часть

Строки в языке Си

Для символьных данных в Си введен тип char. Для представления символьной информации используются символы, символьные переменные и текстовые константы.

Примеры:

```
const char c='c';    /* символ – занимает один байт, его значение не меняется */
char a,b;            /* символьные переменные, занимают по одному байту, значения меняются */
const char *s="Пример строки\n" ;      /* текстовая константа */
```

Строка описывается как массив символов. Число элементов массива равно числу элементов в строке плюс символ конца строки (\0). Поскольку строка – это массив, то для работы со строками очень удобно использовать указатели.

Пример. Записать введенную строку символов в обратном порядке.

```
#include<stdio.h>
void main()
{
int top,bot;
char string[10],temp; /*описание строки как массива символов*/
     scanf("%s",string);
/* при вводе строк символ & не используется, так как имя массива является указателем
на его начало */
     for(top=0,bot=10;top<bot:top++,bot--)
     {
          temp=string[top];
          string[top]=string[bot];
          string[bot]=temp;
     }
     printf("%s\n",string);
```

```
}
```

Присвоить значение строке с помощью оператора присваивания нельзя. Поместить строку в массив можно либо при вводе, либо с помощью инициализации.

Пример:

```
void main()
{
char s1[10] = "string1";
int k = sizeof(s1);
char s2[] = "string2";
     k=sizeof(s2);
char s3[] = {'s','t','r','i','n','g','3'}
     k=sizeof(s3);
char *s4 = "string4";        /* указатель на строку, ее нельзя изменить */
     k=sizeof(s4);
}
```

Результаты:

s1 – выделено 10 байт
s2 – выделено 8 байт (7+1 байт под \0)
s3 – выделено 8 байт (7+1 байт под \0)
s4 – размер указателя sizeof(char *)

Операции со строками

Для ввода и вывода символьных данных в библиотеке stdio.h определены следующие функции:

- **int getchar(void)** - осуществляет вод одного символа их входного потока, при этом она возвращает один байт информации (символ) в виде значения типа int. Это сделано для распознавания ситуации, когда при чтении будет достигнут конец файла.
- **int putchar (int c)** – помещает в стандартный выходной поток символ c.
- **char* gets(char*s)** – считывает строку s из стандартного потока до появления символа '\n', сам символ '\n' в строку не заносится.
- **int puts(const char* s)** записывает строку в стандартный поток, добавляя в конецс троки символ '\n', в случае удачного завершения возвращает значение, больше или равное 0, и отрицательное значение (EOF=-1) в случае ошибки.

Описание функции работы со строками содержится в стандартной библиотеке string.h. Ниже приведены прототипы этих функций и их описания.

Прототип функции	**Краткое описание**
char *strcpy(char *s ,const char *ct)	копирует строку ct в строку s, включая '\0'; возвращает s
char *strncpy(char *s ,const char *ct, size_t n)	копирует не более n символов строки ct в s; возвращает s. Дополняет результат символами '\0', если символов в ct меньше n
char *strcat(char *s ,const char *ct)	приписывает ct к s; возвращает s
char *strncat(char *s ,const char *ct, size_t n)	приписывает не более n символов ct к s, завершая s символом '\0'; возвращает s
int strcmp(const char *cs, char *st)	сравнивает cs и ct; возвращает <0, если cs<ct; 0, если cs==ct; и >0, если cs>ct
int strncmp(const char *s ,const char *ct)	сравнивает не более n символов cs и ct; возвращает <0, если cs<ct, 0, если cs==ct, и >0, если cs>ct
char *strchr(const char *cs, int c)	возвращает указатель на первое вхождение c в cs или, если такового не оказалось, NULL
char *strrchr(const char *cs, int c)	возвращает указатель на последнее вхождение c в cs или, если такового не оказалось, NULL
size_t strspn(const char *cs ,const char *ct)	возвращает длину начального сегмента cs, состоящего из символов, входящих в строку ct
size_t strcspn(const char *cs ,const char *ct)	возвращает длину начального сегмента cs, состоящего из символов, не входящих в строку ct

Прототип функции	Краткое описание
char *strpbrk(const char *cs ,const char *ct)	возвращает указатель в cs на первый символ, который совпал с одним из символов, входящих в ct, или, если такового не оказалось, NULL
char *strstr(const char *cs, const char *ct)	возвращает указатель на первое вхождение ct в cs или, если такового не оказалось, NULL
size_t strlen(const char *cs)	возвращает длину cs
char * strerror(int n)	возвращает указатель на зависящую от реализации строку, соответствующую номеру ошибки n
char * strtok(char *s, const char *ct)	strtok ищет в s лексему, ограниченную символами из ct;

Функция strtok предназначена для разбиения строки на лексемы – лексические единицы, разделенные так символами-разделителями. При первом вызове функции указатель s не должен быть равен NULL, а должен указывать на строку, подлежащую разбиению. Функция ищет в строке s первую лексему, состоящую из символов, не входящих в ct; а на место первого символа, совпадающего с одним из символов, заданных в ct, записывается символ '\0', и функция возвращает указатель на лексему.

Каждый последующий вызов функции strtok, в котором первый аргумент равен NULL, возвращает указатель на следующую лексему, которую функция будет искать сразу за концом предыдущей.

Функция strtok возвращает NULL, если больше никакой лексемы не обнаружено. Символы-разделители, задаваемые параметром ct, могут изменяться от вызова к вызову.

Пример. Необходимо разбить строку на лексемы (например, слова). В качестве символов-разделителей используются символы табуляции, пробела, точка и запятая. Пример программы, решающей эту задачу, приведен ниже.

```
#include<stdio.h>
#include<string.h>
void main()
{
char string[] = "Строка, содержащая несколько\tлексем.";
char delim[] = "\t ,.";      /* символы-разделители */
char *ptr;
     ptr = strtok(string,delim);         /* выделим первую лексему */
     while(ptr != NULL)                  /* если лексема найдена, продолжим поиск */
     {
          printf("Очередная лексема: %s\n",ptr);
          ptr = strtok(NULL, delim);
     }
}
```

В результате выполнения приведенной программы на экран будет выведено следующее:

```
Очередная лексема: Строка
Очередная лексема: содержащая
Очередная лексема: несколько
Очередная лексема: лексем
```

ВНИМАНИЕ!!! Поскольку функция strtok(char *s, const char *ct) записывает символ '\0' на место очередного разделителя в строке s, то после работы функции строка s оказывается "испорченной". Так, в приведенном выше примере после завершения цикла while длина строки string вместо первоначальных 36 символов станет равна 6, потому что на место символа 'запятая' будет записан символ '\0'.

Функции, начинающиеся с **mem** и описанные в библиотеке string.h, предназначены для работы с объектами, рассматриваемыми как массивы байт, а не символов.

Прототип функции	Краткое описание
void *memcpy(void *s, const void *ct, size_t n)	копирует n символов из ct в s и возвращает s
void *memmove(void *s, const void *ct, size_t n)	делает то же самое, что и memcpy, но работает и в случае "перекрывающихся" объектов.
int memcmp(const void *cs, const void *ct, size_t n)	сравнивает первые n символов cs и ct; выдает тот же результат, что и функция strcmp
void *memchr(cs, int c, size_t n)	возвращает указатель на первое вхождение символа c в cs или, если среди первых n символов c не встретилось, NULL

Прототип функции	Краткое описание
void *memset(void *s, int c, size_t n)	размещает символ c в первых n позициях строки s и возвращает s

ПРИМЕЧАНИЕ. Если в функциях копирования используются объекты, перекрывающиеся в памяти, то, за исключением функции **memmove**, поведение функций не определено. Функции сравнения рассматривают аргументы как массивы элементов типа *unsigned char*.

Содержание отчета

Отчет по лабораторной работе должен содержать:

- задание лабораторной работы, соответствующее варианту
- структурную схему алгоритма программы и подпрограммы (подпрограмм)
- текст программы
- результаты работы программы