ЛАБОРАТОРНАЯ РАБОТА №6
СИМВОЛЬНЫЕ СТРОКИ.
ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

***Цель работы**:* изучить особенности работы со строковыми объектами как одномерными и двумерными символьными массивами, научиться использовать динамическое распределение памяти.

Краткие теоретические сведения

СИМВОЛЬНЫЕ СТРОКИ

Символьная строка – это массив символов, заканчивающийся нулевым символом.

Нулевой символ – это символ с кодом 0, что записывается следующим образом ‘\0’.

Положение нулевого символа определяет фактическую длину строки. Количество элементов в символьной строке на 1 больше, чем изображение строки.

Спецификация для символьных строк – %s

Определение символьной строки

char ИмяСтроки [размер];

ИмяСтроки – это указатель на первый символ в строке, т.е. адрес первого символа.

размер-1 – это максимальное количество символов, которые могут быть записаны в строке.

Строковая константа – это набор символов, заключенных в двойные кавычки, например: “Лабораторная работа по строкам”. В конце строковой константы явно указывать символ ´\0´ не нужно.

При определении строки можно инициализировать:

char S1[10] = ”123456789”, S2[ ] = ”12345”;

в последнем случае размер строки будет установлен по фактическому количеству символов.

ФУНКЦИИ ВВОДА СИМВОЛЬНЫХ ДАННЫХ

1. Функция **scanf(“%s”, имяСтроки);**

Единственный случай, когда в функции **scanf** не записывается адрес перед вводимой переменной – это при вводе символьной строки, т.к. имя строки – это и есть адрес. Для функции **scanf** ввод завершается либо символом пробел, либо символом '\n ', причем сами эти символы в строку не заносятся. Следовательно, с помощью функции **scanf** можно вводить только слова.

Пример:

```
char str[80];
scanf("%s", str);
```

2. char* gets(char*s) – считывает только строку **s** из стандартного потока.

Для функции **gets** ввод завершается символом '\n ', сам символ '\n' в строку не заносится. Следовательно, с помощью функции **gets** можно вводить и предложения.

Пример:

```
char str[80];
gets(str);
```

ФУНКЦИЯ ВЫВОДА ТОЛЬКО СИМВОЛЬНЫХ ДАННЫХ

int puts(**const char*** s) – записывает строку в стандартный поток, добавляя в конец строки символ '\n‘.

В случае удачного завершения возвращает значение большее или равное 0 и отрицательное значение (EOF=-1) в случае ошибки.

Пример:

```
char str[80];
gets(str);
puts(str);
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ

Для работы со строками существуют специальные библиотечные функции, которые содержатся в заголовочном файле string.h.

1. *Функция присвоения* **strcpy**:

char* strcpy (**char***s1, **const char***s2);

копирует строку s2 в строку s1, причем размер строки s1 должен быть достаточно большим, чтобы хранить строку s2 и символ конец строки. Возвращает значение строки s1.

2.*Функция добавления* **strcat**:

char* strcat (**char***s1, **const char***s2);

добавляет к строке s1 строку s2, причем размер строки s1 должен быть достаточно большим, чтобы хранить строки s1 и s2, а также символ –конец строки. Возвращает значение строки s1.

3. *Функция сравнения* **strcmp**:

int strcmp (**const char***s1,**const char***s2);

Сравнивает посимвольно строки s1 и s2, причем регистр учитывается.

Возвращает: 0, если строки одинаковы; 1 если встречается хотя бы один символ в строке s1, код которого больше, чем код символа строки s2; -1, если встречается хотя бы один символ в строке s1, код которого меньше, чем код символа строки s2.

4.*Функция, определяющая фактический размер строки* **strlen**:

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

Возвращает количество символов, предшествующих символу конец строки.

МАССИВ СТРОК

Определение массива строк

char ИмяСтроки [размер1][размер2];

размер1– это целая положительная константа, т.е. количество указателей, которые содержат адреса строк.

размер2– размер каждой строки.

ИмяСтроки – это адрес адреса, т.е. указатель на указатель.

Пример ввода и вывода массива символьных строк

char string[5][20];

```
int i;
for( i = 0;i < 5; i++){
        printf("Строка %d\n",i+1);
        gets(string[i]);// ввод массива строк
}
for(i = 0; i < 5; i++)   puts(string[i]); // вывод массива строк
```

ОПЕРАЦИЯ sizeof

Операция sizeof позволяет определить размер типа данных в оперативной памяти.

Операция sizeof имеет следующий вид:

```
sizeof(имя)
```

где имя –это *идентификатор* или *имя типа*. Имя не может быть void.

Примеры:

1.
```
short int num;
printf("short int = %d\n",sizeof(short int)); // результат 2
printf("short int = %d\n",sizeof(num)); // результат 2
```

2.
```
int a1,a2;
double *p;
a1 = sizeof ( p ); // a1=4
a2 = sizeof ( *p );//a2 = 8
```

3.
```
int N1, N2;
double arr[10];
N1 = sizeof ( arr );// N1 = 80
N2 = sizeof ( arr ) / sizeof ( arr[0] );// N2= 10
```

ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

Ядром динамического выделения памяти являются функции, объявленные в стандартной библиотеке в заголовочном файле **stdlib.h**

1. ***Функция выделения памяти*** **malloc**

void * malloc(**unsigned** size);

выделяет область памяти размером *size* байт, в случае успеха, возвращает указатель на начало выделенного блока памяти, если для выделенной памяти не хватает места, возвращает NULL.

2. ***Функция выделения памяти*** **calloc**

void * calloc(**unsigned** num ,**unsigned** size);

выделяет область памяти размером num**size* байт, в случае успеха, возвращает указатель на начало выделенного блока памяти, выделяемая память автоматически инициализируется нулями, если для выделенной памяти не хватает места, возвращает NULL.

3.***Функция переопределения выделенной памяти*** **realloc**

void * realloc (**void** *ptr ,**unsigned** size);

изменяет размер *динамически выделенной памяти*, на которую указывает *ptr* на новый размер – *size* байтов. Т.е для того, чтобы можно было изменить размер памяти, его необходимо обязательно динамически выделить с помощь функции malloc или функции calloc;

4. ***Функция освобождения памяти*** **free**

void free(**void** *ptr);

Функция *освобождает* область памяти, ранее выделенной при помощи функции malloc, calloc или realloc на которую указывает ptr.

Для того, чтобы динамически выделить память, необходимо:

1.*Определить указатель того типа, для которого выделяется память*

ТипДанных *ptr;

2. *Выделить память либо с помощью функции malloc, либо с помощью calloc*

2.1. с помощью функции **malloc**

ptr = (**ТипДанных***) malloc (n * **sizeof** (**ТипДанных**));

2.2. с помощью функции **calloc**

ptr = (**ТипДанных***) calloc (n , **sizeof** (**ТипДанных**));

3.*Проверка на выделения памяти*

if (!ptr){ puts(“ Not enough memory ”); **return**; }

4. *После использования освободить*: free (ptr);

ПРИМЕРЫ РЕШЕНИЙ

1. Ввести предложение и слова для поиска в предложении.

```
#include <stdio.h>
#include <string.h>
void main(){
char x[100], y[50], *ptr;
printf("Введите строку: ");
gets(x);
printf("Введите слова для поиска в строке: ");
scanf("%s", y);
ptr = strstr(x,y);
if(!ptr)
       printf("нет такого слова в предложении\n");
else
      printf("есть такого слово в предложении\n");
}
```

2. Найти количество гласных букв в веденном предложении.

```
#include <stdio.h>
```

//получает строку (содержащую гласные буквы) и символ. Возвращает -1, если символ не //гласная буква и неотрицательное число, если буква гласная.

```
int find(char*, char);
void main(){
     char str[80];
     gets(str);
```

//строка, которая содержит гласные

```
     char gl[7] = "aouiey";
```

```
    for(int i=0,k=0; str[i] != '\0' ;i++)
        if(find(gl,str[i])>=0) k++;
    printf("\n%d",k);
}
```

//Функция поиска заданного символа в строке

```
int find(char* s, char c) {
    for (int i=0; s[i] ;i++)
        if(s[i] == c) return i;
    return -1;
}
```

3. **Ввести строку, которая содержит целые числа, и преобразовать ее в число.**

```
#include<stdio.h>
void main(){
    char m[100];
    int n = 0,i;
    printf("Ввести строку: ");
    scanf("%s",m);
    for(i=0; m[i] ;i++){
            n *= 10;
            n += m[i]-'0';
    }
    printf("\n%d\n",n);
}
```

Предположим в строке m записаны следующие символы ‘1’’2’’4’’5’. Если мы целочисленной переменной n присвоим первый символ, т.е. m[0], то в переменную n запишется код символа ‘1’, т.е. число 49. Если же мы хотим, чтобы в целочисленной переменной n было записано число 1, а не код символа ‘1’, то необходимо от самого символа отнять символ 0, т.е. n = m[0]-‘0’. И, наоборот, если число необходимо преобразовать в символ, то к числу прибавляется символ 0.

4. Ввести массив из пяти строк, рассортировать в алфавитном порядке

```
#include<stdio.h>
#include<string.h>
void main(){
     char string[5][20],buf[20];
     int i,j,k;
     // ввод массива строк
     . . . . . .
     for(i=0;i<4;i++){
          for(k=i,j=i+1;j<5;j++)
               if(strcmp(string[k],string[j])>0)
                    k=j;
          strcpy(buf,string[i]);
          strcpy(string[i],string[k]);
          strcpy(string[k],buf);
     }
     puts("");
     //вывод отсортированного массива строк
     . . . . . .
}
```

5. Программа, которая динамически выделяет массив целых чисел. Удаление числа из массива и добавление чисел в массив.

```
#include <stdio.h>
#include <stdlib.h>
void vvod_mas(double *, int, int);
void vivod_mas(double *, int);
int del_elm(double *, int);
void main(){
     double  *arr, *ptr;
```

```
    int n, k, n1,s;
    printf("Enter size of array: ");
    scanf("%d",&n);
    if(n <= 0) {
         puts("Errors");
         return;
    }
    arr=(double *) malloc ( n * sizeof(double));
    if( !arr ) {
         printf("Not enough memory \n");
         exit(1);
    }

    vvod_mas(arr, 0, n);

    printf("\nEnter\n1-to delete\n2- to add\n");
    scanf("%d",&s);
    if(s == 1){
    k = del_elm ( arr,  n);
  if(k == 0)
    printf("There are no such elements\n");
  else{
    if( n == k ) {
      puts("Array is empty");
      free (arr);
      return;
    }
ptr=(double*)realloc(arr,(n- k)*sizeof( double));
    if( ptr ) {
         arr = ptr;
         n  -=  k;
```

```
      }
   }
}
else
{
   printf("Enter the numbers to add: ");
   scanf("%d", &n1);
   if(n1 > 0){
    ptr=(double*)realloc(arr,(n+n1)*sizeof(double));
    if( ptr ){
      arr = ptr;
      n += n1;
      vvod_mas(arr, n - n1, n);
     }
   }
}

vivod_mas(arr, n);
free(arr);
}
int  del_elm( double *a,  int n1 ){
      int k = 0, i, j;
      double y;
      printf("Vvedite chislo: ");
      scanf("%lf", &y);
      for ( i = 0; i  <  n1 - k;  i++)
           if(a[i] == y){
                k++;
                for(j = i; j < n1 - k; j++)
                     a[j] = a[j+1];
                i--;
```

```
        }
        return k;
}
void vvod_mas( double *p, int k1, int k2 ){
    for ( int i=k1;i<k2;i++){
        printf("[%d]=",i);
        scanf("%lf", &p[i]);
    }
}
void vivod_mas( double *p, int k){
    for( int i=0; i < k; i++)
        printf("[%d]=%.2lf\n", i, *(p++) );
}
```

6. Двумерный массив. Дописать освобождение памяти.

```
#include <stdio.h>
#include <stdlib.h>
void main(){
    int**arr;
    int row, col, i, j;
    printf("Enter row: ");
    scanf("%d", &row);
    printf("Enter col: ");
    scanf("%d", &col);
    if(row <= 0 || col <= 0) {
        puts("Errors");
        return;
    }
// массив указателей
    arr=(int**)malloc(row*sizeof(int*));
```

```
//одномерный массив
    for(i = 0 ; i < row; i++)
        arr[i]=(int*)malloc(col* sizeof(int));
    if( !arr ) {
        printf("Not enough memory \n");
        exit(1);
    }
    for(i = 0; i < row; i++)
        for(j = 0; j < col; j++){
            printf("[%d][%d]=",i,j);
            scanf("%d", &arr[i][j]);
        }
    for(i = 0; i < row; i++){
        for(j = 0; j < col; j++)
            printf("%d\t",arr[i][j]);
        puts("");
    }
//добавить освобождение выделенной памяти
}
```

ЗАДАЧИ ДЛЯ ВЫПОЛНЕНИЯ:

1. Ввести строку и распечатать каждый ее символ с новой строки.
2. Ввести строку и переписать ее в обратном порядке в новую строку.
3. Ввести строку и заменить символ «а» на символ «!» в данной строке.
4. Посчитать, сколько раз в словах встречается буква «а» и поменять первый и последний символы.
5. Посчитать число слов в введенном предложении. Учесть, что предложение может начинаться и заканчиваться пробелами, а также между словами может быть больше, чем один пробел.
6. Написать функцию присвоения строк.

Прототип функции **char* prisv(char*,char*);** т.е. присвоить вторую строку первой и вернуть первую строку.

7. Написать функцию добавления строк.

Прототип функции **char* dobav(char*,char*);** т.е. добавить вторую строку первой и вернуть первую строку.

8. Написать функцию, которая считает количество введенных символов. Прототип функции **int dlina(char*);**получает строку и возвращает количество введенных символов.
9. Написать функцию, которая сравнивает строки и возвращает: **0**, если строки одинаковы; **1**, если в первой строке встретился символ код, которого больше чем, код символа во второй строке и **-1** в противном случае. Прототип функции **int srav(char*, char*);**
10. Имеем 4 строки, например **char x[20], y[20], z[20], t[80];** В первую строку вводим фамилию, во вторую имя, а в третью отчество и используя функции из библиотеки **<string.h>** записываем в строку **z** сначала фамилию, потом пробел, имя, пробел и отчество, пробел. Распечатываем только строку **z**.
11. Введите строку. Если длина строки а) больше 10, то удалить два последних символа; б) меньше 10, то удалить два первых символа; в) равно 10, то удалить символ посередине.
12. Введите предложение. Определите, сколько раз в данном предложении встречается введенное слово.
14. Ввести число и распечатать те цифры данного числа, которые делятся на 2.
15. Ввести число и посчитать произведение цифр данного числа.
16. Ввести число и распечатать цифры данного числа через два пробела. Например, если ввели 123456, то печатает 1 2 3 4 5 6.
17. Ввести дату в строку следующим образом: 12/05/1956 и программа распечатывает строку следующим образом: 12 мая, 1956 года (использовать оператор **switch**).

18. Ввести число, например 3451, программа должна напечатать три тысячи четыреста пятьдесят один.
19. Ввести число, посчитать сумму цифр введенного числа, используя **char ***.
20. За один просмотр исходного текста определить, сколько раз встречается каждый символ**.**

21. Ввести массив строк, в который записываются целые числа, преобразовать их в массив чисел и вывести массив чисел.

22. Дана действительная квадратная матрица порядка N (матрица выделяется динамически). Найти сумму и произведение элементов, расположенных ниже главной диагонали.

23. Задачу 12 из лабораторной 5 переделать следующим образом:

1. Массив выделить динамически.
2. Удаление заданного числа из массива (переопределить размер массива).
3. Добавление чисел в массив (переопределить размер массива).

24. Ввести строку символов. Определить, является ли данная строка палиндромом (т.е. справа налево и слева направо читается одинаково).