ЛАБОРАТОРНАЯ РАБОТА № 21

УКАЗАТЕЛИ И СТРОКИ

Цель работы: приобрести практические навыки использования техники указателей при работе со строками.

Краткие теоретические сведения

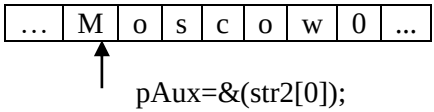
Указатели и символьные массивы (строки). Для решения задач, сводящихся к задаче о выделении подстроки, начало которой не совпадает с началом содержащей ее строки, нужно воспользоваться указателями. Пример: выделить из текста N символов, начиная с позиции m-го символа.

Для решения задачи определим указатель pAux на тип данных char (т.е. на любые переменные типа char), которому с помощью операции взятия адреса присвоим адрес первого элемента строки str2.

```
char  *pAux;
pAux=&(str2[0]);
```

Эта ситуация иллюстрируется на рисунке:

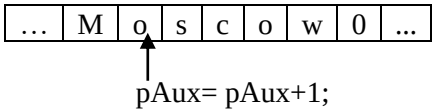
char str2[]=”Moscow”;



pAux=&(str2[0]);

После того как мы увеличим исходное значение указателя pAux на единицу (pAux= pAux+1), он будет показывать на следующий байт памяти, т.е. на следующий элемент массива str2 с элементами типа char, каждый из которых занимает в памяти компьютера всего один байт:

char str2[]=”Moscow”;



pAux= pAux+1;

Ситуация, показанная на этом рисунке, является исходной для выделения из строки str2 подстроки, начинающейся со второй позиции. Например, следующий вызов функции strncpy():

strncpy(str1, pAux, 3);

приведет к тому, что в буфер str1 будет скопирована подстрока ”osc”, т.е. три символа из строки str2, начиная с позиции символа, на который показывает указатель pAux. Убедиться в этом на практике можно с помощью следующей программы.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{ char str1[128];
  char str2[]=”Moscow”;
  char *pAux;
  pAux=&(str2[0]);
  pAux=pAux+1;
  memset(str1, 0, sizeof(str1));
  strncpy(str1, pAux, 3);
  printf(“str1→%s\n”, str1);
  getch();
}
```

Задача о выделении из строки концевого отрезка заданной длины (последних N символов), сводится к только что рассмотренной, так как можно вычислить полную длину строки, отнять длину отрезка выделения и получится начальная позиция внутри строки, с которой и можно производить функцией strncpy() копирование символов в буфер.

Очень распространена задача о поиске внутри строки заданной подстроки с тем, чтобы изъять эту подстроку или заменить на другую и т.д.

Для решения этой задачи следует применить библиотечную функцию strstr():

pPos= strstr(text, fragment);

которая ищет местоположение начала подстроки fragment внутри строки text и возвращает указатель типа char* на соответствую-

щий символ строки text. Если фрагмент внутри текстового отрезка не находится, то возвращается значение нуль.

Например, для текстовых строк

char text[]=”Time”; char frag[]=”me”;

поиск вхождения второй строки в первую функцией strstr():

pPos= strstr(text, frag);

дает в качестве результата указатель pPos, показывающий на символ ‘m’ в первой строке. Так что если воспользоваться этим указателем в функции printf():

printf(“%s”, pPos);

то на дисплее увидим строку ”me”. Номер (индекс) N этого символа можно получить с помощью следующих вычислений:

N=pPos–&(text[0]);

То есть нужно из найденного адреса вычесть адрес самого первого элемента строки text. В примере целая переменная N получит значение 2.

Пример: найти **все** вхождения подстроки в одну и ту же строку.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{ char text[64]=”When you say yes, I say yes too”;
  char aux[64];
  char frag[]=”yes”;
  char *pPos;  char  *pAux;  int N;
  memset(aux, 0, sizeof(aux));
  pPos= strstr(text, frag);
  if (pPos!=0)
    { N = pPos–&( text[0]);
      strncpy(aux, text, N);
      strcat(aux, “no”);
      pAux = pPos + strlen(frag);
    }
  while (pPos != 0)
    { pPos = strstr(pAux, frag);
      if (pPos !=0)          { N = pPos – pAux;
            strcat(aux, pAux, N);
```

```
            strcat(aux, “no”);
            pAux = pAux + N +strlen(frag);          }
      else  strcat(aux, pAux);
     }
    printf(“Original text: %s\n”, text);
    printf(“Processed text: %s\n”, aux);
    getch();
}
```

Программа призвана найти **все** вхождения слова “yes” в исходном тексте.

Рассмотрим первый вызов функции strstr().

```
  pPos= strstr(text, frag);
  if (pPos!=0)      { N = pPos–&( text[0]);
      strncpy(aux, text, N);
      strcat(aux, “no”);
      pAux = pPos + strlen(frag);      }
```

Если искомый фрагмент вообще находится в тексте, то после отработки функции strstr() переменная pPos не равна нулю и выполняются операторы, копирующие начальный кусок исходного текста во вспомогательный буфер aux, туда же добавляется новое слово no, после чего значение указателя pAux устанавливается на позицию в исходном тексте, следующую за первым вхождением слова yes.

Теперь pAux «смотрит» на ту часть исходного текста, которая расположена сразу за первым вхождением искомого фрагмента. В результате последующий вызов функции strstr()

```
pPos = strstr(pAux, frag);
```

может найти последующее вхождение подстроки yes.

Функция strstr() вызывается в цикле до тех пор, пока она вернет 0, означающий, что больше не удается найти подстроку yes ввиду того, что в оставшемся хвостовом куске текста его уже нет. Тогда в else-части оператора if-else нужно эту хвостовую часть просто перекопировать во вспомогательный буфер aux, в котором собирается новая фраза.

Кроме функции поиска strstr() в стандартной библиотеке языка Си есть еще функция поиска одиночных символов strchr().

```
char str[]=”Moscow”;  int N;
char * pPos;  char  *pBeg = &(str[0]);
```

pPos = strchr(str, ‘s’);

N = pPos – pBeg;

В данном фрагменте целая переменная N примет значение, равное 2, так как символ ‘s’ входит в строку ”Moscow” на позиции с индексом 2.

Порядок выполнения работы

1. Изучить теоретические сведения.
2. Выполнить задание.

Задания для выполнения

1. Ввести строку. Преобразовать ее, удалив все двоеточия (**:**), встречающиеся среди первых n/2 символов (n – длина введенной строки). Вывести преобразованную строку.

2. Если в заданной строке есть хотя бы один символ “*”, то продублировать все цифры, встречающихся до первого символа “*”, иначе - вывести соответствующее сообщение.

3. Если в заданной строке есть хотя бы один символ “*”, то удалить все точки, встречающихся после первого символа “*”, иначе - вывести соответствующее сообщение.

4. Ввести набор слов, разделенных одним пробелом. Удалить в нем все слова, заканчивающиеся заданной буквой (ввести с клавиатуры), и подсчитать их количество.

5. Ввести строку, в которой слова разделены пробелами. Если в ней есть хотя бы один символ “:”, то удалить все слова, начинающиеся с большой буквы, расположенные после первого символа “:” или вывести сообщение об отсутствии указанного символа.

6. Ввести три строки, сцепить их. В результирующей строке подсчитать количество заглавных букв. Продублировать все запятые. Результат вывести на экран.

7. Ввести строку из английских слов, разделенных пробелами. Удалить слова, начинающиеся с гласных букв.

8. Ввести строку, в которой слова разделены одним пробелом. Удалить слова, состоящие из не более чем четырех букв.

9. Ввести строку, в которой слова разделены одним пробелом. Удалить слова, начинающиеся и заканчивающиеся одинаковой буквой.

10. Ввести строку, в которой слова разделены одним пробелом. Заменить одно заданное слово другим заданным словом.

11. Ввести набор слов, разделенных одним пробелом. Удалить слова, начинающиеся с заданной буквы (ввести с клавиатуры).

12. Ввести строку, в которой слова разделены одним пробелом. Удалить слова, состоящих из пяти букв.

13. Если в заданной строке есть хотя бы один символ “*”, то удалить заданное слово после первого символа “*”.

14. Ввести строку, содержащую скобки. Удалить все цифры между первой и последней скобками.

15. Ввести строку, в которой слова разделены одним пробелом. Удалить слова, состоящих из M (ввести с клавиатуры) букв, и подсчитать их количество.