По индивидуальному заданию создать функцию для обработки символьных строк. За образец брать библиотечные функции обработки строк языка C, но не применять их в своей функции. Предусмотреть обработку ошибок в задании параметров и особые случаи. Разработать два варианта заданной функции — используя традиционную обработку массивов и используя адресную арифметику.

**Функция Findwords(s,s1)**

Назначение: поиск вхождения в строку s заданной фразы (последовательности слов) s1.

**Пример решения задачи**

**Функция substr(s,n,l)**

Назначение: выделение из строки s подстроки, начиная с позиции n, длиной l.

**Описание метода решения**

Символьная строка в языке C представляется в памяти как массив символов, в конце которого находится байт с кодом 0 — признак конца строки. Строку, как и любой другой массив можно обрабатывать либо традиционным методом — как массив, с использованием операции индексации, либо через указатели, с использованием операций адресной арифметики. При работе со строкой как с массивом нужно иметь в виду, что длина строки заранее неизвестна, так что циклы должны быть организованы не со счетчиком, а до появления признака конца строки.

Функция должна реализовывать поставленную задачу — и ничего более. Это означает, что функцию можно будет, например, перенести без изменений в любую другую программу, если спецификации функции удовлетворяют условиям задачи. Это также означает, что при ошибочном задании параметров или при каких-то особых случаях в их значениях функция не должна аварийно завершать программу или выводить какие-то сообщения на экран, но должна возвращать какое-то прогнозируемое значение, по которому та функция, которая вызвала нашу, может сделать вывод об ошибке или об особом случае.

Определим состав параметров функции:

int substr (src, dest, num, len);

где

 **src** — строка, с которой выбираются символы;

 **dest** — строка, в которую записываются символы;

 **num** — номер первого символа в строке src, с которого начинается подстрока (нумерация символов ведется с 0);

 **len** — длина выходной строки.

Возможные возвращаемые значения функции установим: 1 (задание параметров правильное) и 0 (задание не правильное). Эти значения при обращениях к функции можно будет интерпретировать как «истина» или «ложь».

Обозначим через **Lsrc** длину строки **src**. Тогда возможны такие варианты при задании параметров:

 **num+len <= Lsrc** — полностью правильное задание;

 **num+len > Lsrc; num < Lsrc** — правильное задание, но длина выходной строки будет меньше, чем **len**;

 **num >= Lsrc** — неправильное задание, выходная строка будет пустой;

 **num < 0** или **len <= 0** — неправильное задание, выходная строка будет пустой.

Заметим, что интерпретация конфигурации параметров как правильная/неправильная и выбор реакции на неправильное задание — дело исполнителя. Но исполнитель должен строго выполнять принятые правила. Возможен также случай, когда выходная строка выйдет большей длины, чем для нее отведено места в памяти. Однако, поскольку нашей функции неизвестен размер памяти, отведенный для строки, функция не может распознать и обработать этот случай — так же ведут себя и библиотечные функции языка C.

**Описание логической структуры**

Программа состоит из одного программного модуля — файл **LAB1.C**. В состав модуля входят три функции — **main**, **substr\_mas** и **subs\_ptr**. Общих переменных в программе нет. Макроконстантой **N** определена максимальная длина строки — 80.

Функция **main** является главной функцией программы, она предназначена для ввода исходных данных, вызова других функций и вывода результатов. В функции определены переменные:

 **ss** и **dd** — входная и выходная строки соответственно;

 **n** — номер символа, с которого должна начинаться выходная строка;

 l — длина выходной строки.

Функция запрашивает и вводит значение входной строки, номера символа и длины. Далее функция вызывает функцию **substr\_mas**, передавая ей как параметры введенные значения. Если функция **substr\_mas** возвращает 1, выводится на экран входная и выходная строки, если 0 — выводится сообщение об ошибке и входная строка. Потом входная строка делается пустой и то же самое выполняется для функции **substr\_ptr**.

Функция **substr\_mas** выполняет поставленное задание методом массивов. Ее параметры: — **src** и **dest** — входная и выходная строки соответственно, представленные в виде массивов неопределенного размера; **num** и **len**. Внутренние переменные **i** и **j** используются как индексы в массивах.

Функция проверяет значения параметров в соответствии со случаем 4, если условия этого случая обнаружены, в первый элемент массива **dest** записывается признак конца строки и функция возвращает 0.

Если случай 4 не выявлен, функция просматривает num первых символов входной строки. Если при этом будет найден признак конца строки, это — случай 3, при этом в первый элемент массива dest записывается признак конца строки и функция возвращает 0.

Если признак конца в первых num символах не найден, выполняется цикл, в котором индекс входного массива начинает меняться от 1, а индекс выходного — от 0. В каждой итерации этого цикла один элемент входного массива пересылается в выходной. Если пересланный элемент является признаком конца строки (случай 2), то функция немедленно заканчивается, возвращая 1. Если в цикле не встретится конец строки, цикл завершится после len итераций. В этом случае в конец выходной строки записывается признак конца и Функция возвращает 1.

Функция **substr\_ptr** выполняет поставленное задание методом указателей. Ее параметры: — **src** и **dest** — входная и выходная строки соответственно, представленные в виде указателей на начала строк; **num** и **len**.

Функция проверяет значения параметров в соответствии со случаем 4, если условия этого случая выявлены, по адресу, который задает **dest**, записывается признак конца строки и функция возвращает 0, эти действия выполняются одним оператором.

Если случай 4 не обнаружен, функция пропускает num первых символов входной строки. Это сделано циклом while, условием выхода из которого является уменьшение счетчика num до 0 или появление признака конца входной строки. Важно четко представлять порядок операций, которые выполняются в этом цикле:

 выбирается счетчик **num**;

 счетчик **num** уменьшается на 1;

 если выбранное значение счетчика было 0 — цикл завершается;

 если выбранное значение было не 0 — выбирается символ, на который указывает указатель **src**;

 указатель **src** увеличивается на 1;

 если выбранное значение символа было 0, то есть, признак конца строки, цикл завершается, иначе — повторяется.

После выхода из цикла проверяется значение счетчика **num**: если оно не 0, это означает, что выход из цикла произошел по признаку конца строки (случай 3), по адресу, который задает **dest**, записывается признак конца строки и функция возвращает 0.

Если признак конца не найден, выполняется цикл, подобный первому циклу **while**, но по счетчику **len**. В каждой итерации этого цикла символ, на который показывает **src** переписывается по адресу, задаваемому **dest**, после чего оба указателя увеличиваются на 1. Цикл закончится, когда будет переписано **len** символов или встретится признак конца строки. В любом варианте завершения цикла по текущему адресу, который содержится в указателе **dest**, записывается признак конца строки и функция завершается, возвращая 1.

**Данные для тестирования**

Тестирование должно обеспечить проверку работоспособности функций для всех вариантов входных данных. Входные данные, на которых должно проводиться тестирование, сведены в таблицу:

вариант src num len dest

1 012345 2 2 23

012345 0 1 0

012345 0 6 012345

2 012345 5 3 5

012345 2 6 2345

012345 0 7 012345

3 012345 8 2 пусто

4 012345 -1 2 пусто

012345 5 0 пусто

012345 5 -1 пусто

**Текст программы**

include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <iostream>

#define N 80

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Функция выделения подстроки (массивы) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int substr\_mas(char src[N],char dest[N],int num,int len){

 int i, j;

 /\* проверка случая 4 \*/

 if ( (num<0)||(len<=0) ) {

 dest[0]=0; return 0;

 }

 /\* выход на num-ый символ \*/

 for (i=0; i<=num; i++)

 /\* проверка случая 3 \*/

 if ( src[i]=='\0') {

 dest[0]=0; return 0;

 }

 /\* перезапись символов \*/

 for (i--, j=0; j<len; j++, i++) {

 dest[j]=src[i];

 /\* проверка случая 2 \*/

 if ( dest[j]=='\0') return 1;

 }

 /\* запись признака конца в выходную строку \*/

 dest[j]='\0';

 return 1;

 }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Функция выделение подстроки \*/

/\* (адресная арифметика) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int substr\_ptr(char \*src, char \*dest, int num, int len) {

 /\* проверка случая 4 \*/

 if ( (num<0)||(len<=0) ) return dest[0]=0;

 /\* выход на num-ый символ или на конец строки \*/

 while ( num-- && \*src++ );

 /\* проверка случая 3 \*/

 if ( !num ) return dest[0]=0;

 /\* перезапись символов \*/

 while ( len-- && \*src ) \*dest++=\*src++;

 /\* запись признака конца в выходную строку \*/

 \*dest=0;

 return 1;

 }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int \_tmain(int argc, \_TCHAR\* argv[])

 {

 char ss[N], dd[N];

 int n, l;

 system("cls");

 printf("input str\n");

 gets(ss);

 printf("begin=");

 scanf("%d",&n);

 printf("size=");

 scanf("%d",&l);

 printf("array:\n");

 if (substr\_mas(ss,dd,n,l)) printf(">>%s<<\n>>%s<<\n",ss,dd);

 else printf("error! >>%s<<\n",dd);

 dd[0]='\0';

 printf("adress arifmetika:\n");

 if (substr\_ptr(ss,dd,n,l)) printf(">>%s<<\n>>%s<<\n",ss,dd);

 else printf("error! >>%s<<\n",dd);

 getch();

return 0;

}