Задачи

Задачи оформляются в виде функций с передачей всех входных и выходных данных через формальные параметры и результат. Задачи, работающие со строками, по заданию преподавателя реализовать либо с использованием массивов char[], либо с использованием указателей char\* и операций вида \*p++. Для  функции привести пример вызова из main со статическими  входными параметрами.

1. **Найти наименьшее общее кратное для всех элементов массива - минимальное число, которое делится на все элементы массива без остатка.**

int f( intA [])

{

constint n = 10;

inti, nok = 1, j;

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

{

j = A[i];

if (j<nok)

j = nok;

for (; j <= nok\*A[i]; j++)

{

if (j%A[i] == 0 &&j%nok == 0)

{

nok = j;

break;

}

}

}

returnnok;

}

**2.      Написать функцию проверки, является ли заданное число простым. С ее помощью написать программу поиска простых чисел в диапазоне 1000-2000, две любые части которого - также простые (например, 1997, 1-997,19-97,199-7)**

bool simple(intx)

{

int n = 0;

for (n = 2; n <x; n++)

{

if (x%n == 0) break;// перебор всех значени n до первого деления без остатка

}

if (n == x) returntrue;

elsereturnfalse;

}

int f(intA[])

{

int c = 0;

for (int i = 1000; i < 2000; i++)

{

int temp = i,j=1,k=0; bool good = false;

if (simple(temp) == true)

good = true;

else good = false;

if (simple(temp / 1000) == true&& simple(temp % 1000))

good = true;

else good = false;

if (simple(temp / 100) == true&& simple(temp % 100))

good = true;

else good = false;

if (simple(temp / 10) == true&& simple(temp % 10))

good = true;

else good = false;

if (good = true)

{

A[c] = i;

c++;

}

}

return c;

}

3.      Сформировать массив простых чисел в диапазоне от 2 до заданного. Очередное простое число определяется попыткой деления нацело числа на все уже накопленные простые числа.

**4.      Любая целочисленная денежная сумма n>7 может быть выдана без сдачи “трешками” и “пятерками”. Программа находит эти числа для заданного n.**

5.      **На заданном интервале найти все числа, цифры которых находятся в строго возрастающем порядке (например, 532).**

int f(inta, intb,intB[])

{

int c = 0;

for (inti = a; i<b; i++)

{

int temp = i, n = 0,s = 0;

int A[30];

for (int t = temp; t!=0; n++,t/=10);

for (int j = 0, ch = temp; j < n; j++)

{

A[j] = ch % 10;

ch /= 10;

}

for (int j = 0; j < n-1; j++)

{

if (A[j + 1]>A[j])

s = 0;

else s = 1;

}

if (s == 0)

{

B[c] = i;

c++;

}

}

return c;

}

6**.      Число Армстронга – такое число из k цифр, для которого сумма k-х степеней его цифр равна самому этому числу, например 153=13+53+33. Найти все трехзначные числа Армстронга.**

int f(intB[])

{

int c = 0;

for (inti = 100; i< 999; i++)

{

int temp = i, n = 0,s = 0;

int A[30];

for (int t = temp; t!=0; n++,t/=10);

for (int j = 0, ch = temp; j < n; j++)

{

A[j] = ch % 10;

ch /= 10;

}

if (temp == (pow(A[0], n) + pow(A[1], n) + pow(A[2], n)))

s = 0;

else s = 1;

if (s == 0)

{

B[c] = i;

c++;

}

}

return c;

}

7**.      Найти все двухзначные (трехзначные) числа, которые совпадают с последними цифрами своих квадратов, например, 252=625, 762=5676.**

int f(intB[])

{

int c = 0;

for (inti = 25; i< 999; i++)

{

int temp = i, n1 = 0, n2 = 0, s = 0;

int A[30],D[100];

for (int t = temp; t!=0; n1++,t/=10);

for (int j = 0, ch = temp; j < n1; j++)

{

A[j] = ch % 10;

ch /= 10;

}

for (int t = pow(temp, 2); t != 0; n2++, t /= 10);

for (int j = 0, ch = pow(temp,2); j < n2; j++)

{

D[j] = ch % 10;

ch /= 10;

}

if (n1 == 2)

{

if ((A[0] == D[0]) && (A[1] == D[1]))

s = 0;

else s = 1;

}

elseif (n1 = 3)

{

if ((A[0] == D[0]) && (A[1] == D[1]) && (A[2] == D[2]))

s = 0;

else s = 1;

}

if (s == 0)

{

B[c] = temp;

c++;

}

}

return c;

}

**8.      Число называется совершенным, если оно равно сумме всех своих делителей, например, 6=1+2+3, 28=1+2+4+7+14. Найти все совершенные числа в заданном интервале.**

void f(inta, intb,intB[])

{

inti=0, j=0, sdel, s = 1,k=0;

for (i = a; i<= b; i++)

{

sdel = 0;//summa delitelei

for (j = 1; j<i; j++) //poiskvsehdelitelei

if (i%j == 0)//delits`abezostatka

sdel += j;

if (i == sdel)

{

B[k] = i;

k++;

s++;

}

}

B[k] = 0;

}

9**.      Найти все числа в заданном диапазоне, которые делятся на любую из своих цифр.**

int f(inta,intb,intB[])

{

int c = 0;

for (inti = a; i<b; i++)

{

int temp = i, n = 0, s = 0;

int A[30];

for (int t = temp; t!=0; n++,t/=10);

for (int j = 0, ch = temp; j < n; j++)

{

A[j] = ch % 10;

ch /= 10;

}

for (int j = 0; j < n && s == 0 && A[j]!=0; j++)

{

if (temp%A[j] == 0)

s = 0;

else s = 1;

}

if (s == 0)

{

B[c] = temp;

c++;

}

}

return c;

}

**10.  Найти числа в диапазоне 100-10000, для которых куб суммы цифр равен значению самого числа (например, 512 - 5+1+2=8).**

int f(inta,intb,intB[])

{

int c = 0;

for (inti = a; i<b; i++)

{

int temp = i, n = 0, s = 0, sum = 0;;

int A[30],D[100];

for (int t = temp; t!=0; n++,t/=10);

for (int j = 0, ch = temp; j < n; j++)

{

A[j] = ch % 10;

ch /= 10;

sum += A[j];

}

if (temp == pow(sum, 3))

s = 0;

else s = 1;

if (s == 0)

{

B[c] = temp;

c++;

}

}

return c;

}

**11.  Разложить число на простые множители (например 36=3\*3\*2\*2). Результат – последовательность множителей в массиве, ограниченная 0.**

void f(intt,intB[])

{

inti = 2,j=0;

while (i<= t)

{

if (t%i == 0)

{

t = t / i;

B[j] = i;

j++;

}

else

i += 1;

}

B[i + 1] = 0;

}

**12.  Определить в массиве максимальную длину последовательности расположенных подряд возрастающих значений и возвратить индекс ее начала.ЗАМЕТКА – функция возвращает длину последовательности а переменная по ссылке ее индекс**

int f(intB[],intn,int&d)

{

int c = 0, l = 0; int A[256],t = 0;

bool up = false;

for (inti = 0; i<n; i++)

{

if (B[i + 1] >B[i])

l++;

else

{

A[i-l] = l+1;

l = 0;

}

}

int max = A[0];

for (inti = 0; A[i] != 0;i++)

if (A[i] > max)

{

max = A[i];

d = i;

}

returnmax;

}

**13.  Подсчитать количество слов в строке.**

int f(charB[])

{

intcuri = 0, curn = 0, count = 0;

while (B[curi - 1] != 0){

if (isalnum(B[curi]))

curn++;

elseif (curn> 0)

{

count++;

curn = 0;

}

curi++;

}

returncount;

}

**14.  Найти в строке слово минимальной длины и возвратить индекс его начала.**

int f(charB [])

{

intcuri = 0, curn = 0, count = 0; int A[128];

while (B[curi - 1] != 0){

if (isalnum(B[curi]))

curn++;

elseif (curn> 0)

{

A[curi - curn] = curn;

count++;

curn = 0;

}

curi++;

}

int min = A[0];

for (inti = 0; i<curi; i++)

if (A[i] < min&&A[i]>0)

min = A[i]-1;

returnmin;

}

**15.  Оставить в строке по одному пробелу между словами.**

void f(charB [])

{

intcuri = 0, curn = 0, count = 0; int A[128];

while (B[curi - 1] != 0){

if (B[curi]==32)

curn++;

elseif (curn> 1)

{

A[curi - curn] = curn;

count++;

inti = curi-curn;

while (B[i])

{

B[i] = B[i + curn - 1];

i++;

}

curn = 0;

curi -= curn;

}

curi++;

}

}

16.  Возвратить индекс начала фрагмента, симметричного относительно центрального символа (например, "abcba"), и имеющего максимальную длину. Длину фрагмента возвратить по ссылке.

17.  "Перевернуть" в строке все слова. (Например: "Жили были дед и баба" - "илиЖилиб дед и абаб").

void f(charB[])

{

char temp[256];

intcuri = 0, curn = 0, count = 0;

while (B[curi - 1] != 0){

if (isalnum(B[curi]))

curn++;

elseif (curn> 0)

{

for (int j = 0, t = curi-1; j <curn; j++,t--)

temp[j] = B[t];

for (int j = curi-curn,t=0; j<curi; j++,t++)

B[j] = temp[t];

count++;

curn = 0;

}

curi++;

}

}

18.  Преобразовать целое из внутренней формы во внешнюю в шестнадцатеричной СС.

void f(charB[], intx)

{

int A[10], temp = 0, n = 0;

char tem [15];

temp = x;

for (int t = temp; t != 0; n++, t /= 10);

for (int j = n-1, ch = temp; j+1 > 0; j--)

{

A[j] = ch % 10;

ch /= 10;

}

inti = 0;

for (; i< n; i++)

{

if (A[i] <= 9)

{

B[i] = A[i] + '0';

}

elseB[i] = A[i] - 10 + 'A';

}

}

19.  Преобразовать число в шестнадцатеричной СС из внешней формы во внутреннюю.

void f(charB[],intC[],intn)

{

int A[10];

inti = 0;

for (; i<n; i++)

{

if (B[i] >= '0'&&B[i] <= '9')

C[i] = B[i] - '0';

elseif (B[i] >= 'A'&&B[i] <= 'F')

{

C[i] = B[i] - 'A' + 10;

}

}

}

20.  Преобразовать дробную часть переменной типа double во внешнюю форму представления (строку символов).

21.  В строке находится символ “точка” и символы-цифры дробной части числа. Преобразовать во внутреннюю форму представления (переменную типа double).

22.  Найти в строке два одинаковых фрагмента, не содержащих пробелы и имеющих максимальную длину и возвратить индекс начала первого из них.

23.  В строке,  содержащей   абзац текста, найти концы  предложений,  обозначенный  символом  "точка". В  следующих за ними  словах  первую строчную  букву  заменить   на прописную. Между словами количество пробелов может быть любым.

24.  Заменить  в  строке  все восьмеричные целые константы на символы с соответствующими кодами, (например,  101 на A).

25.  Найти слово, начинающееся с самой младшей латинской буквы и возвратить индекс его начала.

**26.  Удалить из строки комментарии вида "/\* ... \*/". Игнорировать вложенные комментарии.**

void f(charc [])

{

int k = 0;

for (inti = 0; i<strlen(c); i++)

{

if (c[i] == '/'&&c[i + 1] == '\*'&&k==0)

{

k+=1;

c[i] = ' ';

c[i + 1] = ' ';

}

if (c[i] == '/'&&c[i - 1] == '\*'&&k == 1)

{

c[i] = ' ';

c[i - 1] = ' ';

k = 0;

}

}

}

27.  Заменить  в  строке все большие латинские буквы на соответствующие им шестнадцатеричные коды (например,  А на 0x41, в константе использовать 2 цифры для представления байта).

28.  Заменить в строке все целые константы из 1-2 цифр соответствующим повторением следующего за ними символа (например "abc5xacb15y" - " abcxxxxxacbyyyyyyyyyyyyyyy ").

int f(charB[])

{

int s = 0, curn = 0, c = 0, maxcurn = 0; int A[128];

for (inti = 0; i<strlen(B); i++)

{

if (B[i] == B[i + 1])

curn++;

elseif (curn>1)

{

A[i - curn] = curn;

if (curn>maxcurn)

{

maxcurn = curn;

c = i - maxcurn;

}

int temp = curn;

if (temp / 10 >= 1)

{

B[c] = temp % 10 + '0';

temp /= 10;

B[c-1] = temp % 10 + '0';

for (inti = c + 3; i<strlen(B) + maxcurn; i++)

B[i] = B[i + maxcurn];

}

elseif (temp / 10 == 0)

{

B[c+1] = temp + '0';

for (inti = c+2; i<strlen(B) + maxcurn; i++)

B[i] = B[i + maxcurn+1];

}

curn = 0;

}

}

returnstrlen(B)-2;

}

29.  Найти в строке и удалить из нее последовательность повторяющихся символов максимальной длины (например, "abcxxxxxacbyyyyyyyyyyyyyyyz" - "abcxxxxxacbz").

int f(charB[])

{

int s = 0, curn = 0, c = 0, maxcurn = 0; int A[128];

for (inti = 0; i<strlen(B); i++)

{

if (B[i] == B[i + 1])

curn++;

elseif (curn>1)

{

A[i - curn] = curn;

if (curn>maxcurn)

{

maxcurn = curn;

c = i - maxcurn;

}

curn = 0;

}

}

for (inti = c; i<strlen(B)+maxcurn; i++)

B[i] = B[i + maxcurn+1];

returnstrlen(B);

}

30.  Найти в строке наиболее часто встречающийся символ и заменить его на пробел.

31.  Найти все вхождения подстроки в строке. Строка и подстрока заданы в массивах символов. Результат – массив, заполненный индексами начала подстроки в строке, последовательность ограниченна -1.

32.  Найти в строке самую внутреннюю пару скобок и возвратить индекс открывающейся, например “a(bb(c(d)e(f   **(**ggg)h)))(d)”.

33.  Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы до конца массива сдвигаются на один влево. Сам элемент заносится на освободившуюся последнюю позицию.

|  |
| --- |
| void F33(int A[], int n){ bool kp; int n2 = n - 1; int i, j, minmin = INT\_MAX, pos, curmin = 0; for (i = 0; i < n; i++){ for (j = 0; j < n; j++){ if (A[j] <= A[j+1]){ if (A[j] < minmin && A[j] > curmin){ minmin = A[j]; pos = j; kp = true; } } else{ if (A[j] < minmin && A[j] > curmin){ minmin = A[j]; pos = j; kp = true; } } } if (kp){ for (pos; pos < n2; pos++){ A[pos] = A[pos + 1]; } curmin = minmin; A[pos] = curmin; } kp = false; minmin = INT\_MAX; } } |

34.  Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы массива сдвигаются на один влево. Сам элемент переносится в новый массив.

35.  Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем на его место записывается «очень большое число», например, максимальное +1. Сам элемент переносится в новый массив.

36.  Сортировка выбором. Выбирается максимальный элемент из оставшихся и меняется с последним из них  (обратить внимание на диапазон неупорядоченной части).

void F36(int A[], int n){

bool op;

int i, j, max = 0, pos, curmax = INT\_MAX;

for (i = 0; i < n; i++){

for (j = 0; j < n; j++){

if (A[j] > A[i]){

if (A[j] > max && A[j] < curmax){

max = A[j];

pos = j;

op = true;

}

}

else{

if (A[j] > max && A[j] < curmax){

max = A[j];

pos = j;

op = true;

}

}

}

curmax = max;

max = 0;

if (op){

A[pos] = A[n - 1 - i];

A[n - 1 - i] = curmax;

op = false;

}

}

}

37.  Сортировка выбором. Выбирается минимальный элемент из оставшихся и меняется с первым из них  (обратить внимание на диапазон неупорядоченной части).

**38.  Сортировка подсчетом. Выходной массив заполняется значениями “-1”. Затем для каждого элемента определяется его место в выходном массиве путем подсчета количества элементов строго меньших данного. Естественно, что все одинаковые элементы попадают на одну позицию, за которой следует ряд значений “-1”. После чего оставшиеся в выходном массиве позиции со значением “-1” заполняются копией предыдущего значения.**

void f(intA[],intB[],intn)

{

inti, k, j;

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

B[i] = -1; // Выходной массив заполняется значениями “-1”.

for (i = 0; i <n; i++) // перебирает элементы массива а

{

k = 0;

for (j = 0; j <n; j++) // перебирает элементы массива а

{

if (A[j] <A[i]) // для каждого элемента определяется его место в

k = k++; // выходном массиве путем подсчета количества

// элементов строго меньших данного

}

B[k] = A[i]; // присваивает значение элементу массива b

}

for (i = 1; i <n + 1; i++) // оставшиеся в выходном массиве позиции со

{

if (B[i] == -1) // значением «-1» заполняется копией предыдущего

B[i] = B[i - 1]; // значения

}

}

39.  Сортировка вставками. Берется очередной элемент и извлекается из массива. Затем от начала массива ищется первый элемент, больший данного. Все элементы, от найденного до очередного сдвигаются на один вправо и на освободившееся место помещается очередной элемент. (Поиск места включения от начала упорядоченной части).

40.  Сортировка вставками («всплытием»). Берется предпоследний элемент и меняется со следующим, пока не закончится массив и пока следующий будет меньше текущего. Затем берется предыдущий и т.д..

41.  Сортировка вставками. Берется очередной элемент массива. Затем от начала выходного массива ищется первый элемент, больший данного. Все элементы, от найденного до последнего сдвигаются на один вправо и на освободившееся место помещается очередной элемент.

42.  Сортировка выбором. Выбирается минимальный элемент в массиве и переносится в новый массив. Затем на его место записывается последний элемент исходного массива.