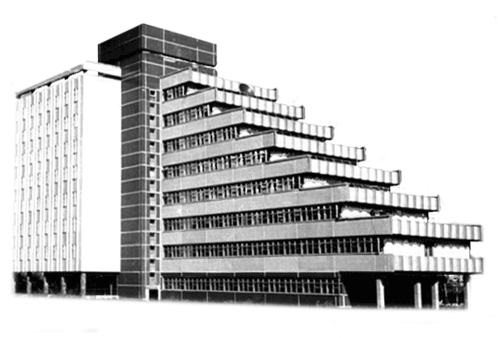
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное агентство по образованию

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Учебный Центр Информационных Технологий «Информатика»



Лабораторная работа № 2

по дисциплине «Информатика и программирование (I часть)»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники

и автоматизированных систем»

Выполнил слушатель: Коренский А.А.

Группа: 1

Вариант: 9

Дата сдачи: .03.2015

Преподаватель: Прытков Д.В.

Новосибирск, 2015г.

# Задание

Задан массив, определить значение k, при котором сумма |A(1)+A(2)+…+A(k)-A(k+1)+…+A(N)| минимальна (то есть минимален модуль разности сумм элементов в правой и левой части, на которые массив делится

этим k).

# Теоретический материал

Операторы создают то, что в обыденном сознании ассоциируется с понятием алгоритм – описание последовательности действий, выполняемых  программой, или логика ее работы. В Си реализован общий для большинства языков программирования «джентльменский набор» управляющих конструкций:

* линейная последовательность действий;
* условная конструкция (если-то-иначе);
* конструкция повторения (цикл);
* переход (и его разновидности).

Управляющая конструкция  - линейная последовательность действий является основной в языках программирования.

В соответствии с принципом вложенности элементами последовательности могут быть не только элементарные действия (простые операторы), но и вложенные синтаксические конструкции, которые на текущем уровне также выполняются последовательно друг за другом.

В Си используется такой принцип: если составной частью управляющей конструкции является единственный оператор, то он никак синтаксически не выделяется. Если же составной частью является последовательность операторов, то она заключается в фигурные скобки ({}) и образует блок.

Блок, или составной оператор выступает как единая синтаксическая единица, вложенная в конструкцию верхнего уровня. В начале его могут быть определены собственные переменные блока, действие которых не распространяется за его пределы, а время существования совпадает с временем его выполнения. Операторы, составляющие блок, выполняются последовательно друг за другом.

Единственный условный оператор имеет две разновидности: с else и без него.

В качестве условия выступает выражение, которое может иметь любой целый результат и интерпретируется в соответствии с принятыми в Си соглашениями о логических значениях: 0 –«ложь», не 0 – «истина». Действует он как и во всех языках программирования: если значение выражения есть «истина», то выполняется первый оператор, если «ложь» - второй (после else). Конструкция является структурированной, обе ветви – прямая и альтернативная – «сливаются» в одну

 В Си имеется три вида циклических конструкций. Общее у них одно: все условия в них являются условиями продолжения, то есть циклы продолжаются, пока значение этих условных выражений – «истина». Операторы цикла состоят из заголовка, в котором определяется характер циклического процесса и оператора - тела цикла. Скобки в заголовке цикла являются неотъемлемым элементом синтаксиса языка.

Метка default обозначает последовательность, которая выполняется «по умолчанию».

Массив – совокупность элементов одного типа.

тип имя\_массива[количество\_элементов]

тип – задает тип элемента массива и может быть как стандартный так и определенный пользователем.

имя\_массива – задается пользователем и используется для обращения к массиву, может состоять из символов английского алфавита, цифр и символов подчеркивания но недолжно начинаться с цифры и недолжно совпадать с зарезервированными словами среды разработки.

количество\_элементов – целочисленная константа определяющая сколько элементов будет в массиве.

Доступ к элементу массива осуществляется с помощью индекса, например A[0] – первый элемент, A[1] – второй элемент, A[n] – n-й элемент массива. Нумерация элементов начинается с нуля. В памяти элементы массива хранится последовательно т.е A[0], A[1], A[2]…A[n]. Имя массива это адрес ячейки памяти по которому находится первый элемент.

# Описание алгоритма

Цель данной программы – нахождение значения K, при котором сумма |A(1)+A(2)+…+A(k)-A(k+1)+…+A(N)| минимальна. Алгоритм нахождения искомого значения представлен двумя способами.

1. С помощью функции *FirstLab1():*

Printf(“Sum”);

Sum+=M[h];

Rmin+=abs(M[h]);

For

(h=0;h<N;h++)

printf("M[i]”);

For

(i=0;i<N;i++)

K=N, h, R, SumP, Sum=0, SumL=0, Rmin=0, i, M[N]

SumL+=M[i];

SumP=Sum-SumL;

R=abs(SumL-SumP);

if (R<=Rmin) {

Rmin=R;

K=i;}

printf("SumL , SumP, R , K”);

For

(i=0;i<N;i++)

false

true

Printf(“K не существует”);

Printf(“K”);

If

(K!=N)

1. С помощью функции *FirstLab2()*:

Функция *FirstLab2()* использует почти такой же набор циклов и операторов, что и *FirstLab1()*. Первые два цикла *For* не изменились (для вывода массива и определения суммы значений массива по модулю и без него), а третий цикл построен иначе. Он повторяет четвертый цикл *For* для нахождения суммы значений правой стороны массива на каждом из своих этапов. Затем он находит сумму левой стороны массива, разность сумм двух сторон. Далее все построено аналогично функции *FirstLab1()*.

# 4. Описание реализации

Подключаемые библиотеки:

#include "stdio.h"

#include "conio.h"

#include "stdlib.h"

Используемые функции:

int main() – вызывает функцию *FirstLab1()* и в случае необходимости *FirstLab2()*;

printf() – вывод форматированного текста на экран;

system("cls") – для очистки экрана;

scanf() – для ввода информации с клавиатуры;

getch() – ожидание ввода символа с клавитуры.

Используемые конструкции:

if – условный оператор, для создания ветвления ;

for – оператор цикла;

#define – препроцессорная директива (подстановка значения по всему коду программы).

# 5. Пример работы программы

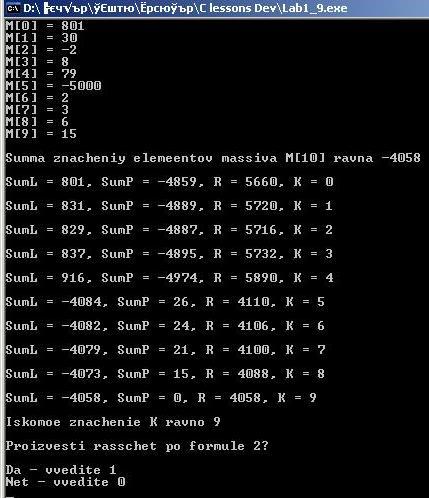


Рисунок 1. Результат работы функции *FirstLab1()*

На рисунке 1 показан результат работы функции *FirstLab1()*. Вначале она выводит на экран заданный массив, затем поочередные значения правой, левой сумм и их разность, а также значение K. По окончании перебора всех элементов массива выводится на экран искомое значение К и предлагается вызвать функцию *FirstLab2()*. Результат работы функции *FirstLab2()* показан на рисунке 2.

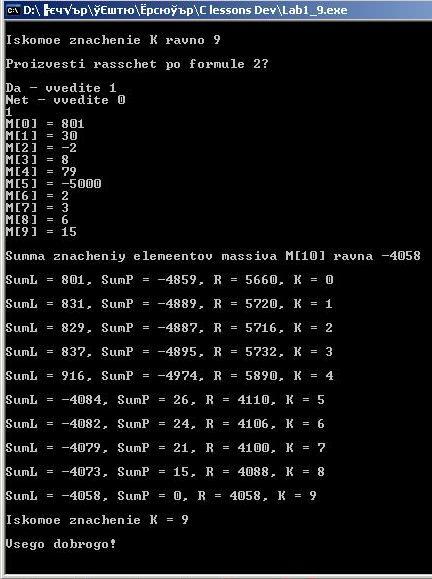


Рисунок 2. Результат работы функции *FirstLab2()*

**Выводы**

Целью этой лабораторной работы было знакомство со средой программирования, а также изучение операторов, управляющих структур, знакомство с целочисленными массивами на практике.

Перед выполнением этой работы, был изучен теоретический материал по данным темам. Затем необходимо было составить блок схему будущей программы, для общего представления ее конструкции. После чего, я приступил к написанию исходного кода.

После написания первой части программы, реализованной с помощью функции FirstLab1(), встал вопрос о ее правильности. Отсюда возникла дублирующая функция FirstLab2(). Основное их различие состоит в том, что в первой функции значение сумм правой и левой сторон находится с помощью одного цикла начиная с первого элемента массива, а во второй функции используется для этой цели два цикла: первый – для левой стороны, второй - для правой.

После было проведено контрольное тестирование программы, которое показало правильность ее работы.

**Приложение. Текст программы с комментариями**

#include "stdio.h"

#include "conio.h"

#include "stdlib.h"

#define N 10

//Лабораторная работа №2 Вариант 9

//Задание: задан массив, определить значение К, при котором сумма

//|A(1)+A(2)+...A(K)-A(K+1)+...+A(N)| минимальна (т.е. минимален модуль разности

//сумм элементов в правой и левой части, на которые массив делится этим К)

int main() {

int O, FirstLab1(), FirstLab2(); //объявление переменной и функции 1, 2

system("cls");

FirstLab1(); //вызов функции 1

printf("Proizvesti rasschet po formule 2?\n\nDa - vvedite 1\nNet - vvedite 0\n"); //предлагает вызвать функцию 2 с другим методом расcчета

scanf("%d", &O);

if (O==true)

FirstLab2(); //вызов функции 2

else

printf("Vsego dobrogo!\n");

getch();

}

int FirstLab1() { //функция определения искомого значения K

int K=N, h, R, SumP, Sum=0, SumL=0, Rmin=0, i, M[N] = {801, 30, -2, 8, 79, -5000, 2, 3, 6, 15}; //объявление переменных и определение массива

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

printf("M[%d] = %d\n", i, M[i]);

for(h=0;h<N;h++) { //цикл определяющий сумму значений элементов //массива и сумму по модулю

Sum+=M[h];

Rmin+=abs(M[h]); //сумма значений массива по модулю - для

//сравнения с разностью в первом приближении

}

printf("\nSumma znacheniy elemeentov massiva M[%d] ravna %d\n\n", N, Sum);

for(i=0;i<N;i++) { //цикл определяющий сумму правой и левой сторон

//массива и их разницу по модулю

SumL+=M[i];

SumP=Sum-SumL;

R=abs(SumL-SumP);

if (R<=Rmin) {

Rmin=R;

K=i;

}

printf("SumL = %d, SumP = %d, R = %d, K = %d\n\n", SumL, SumP, R, i);

}

if (K!=N)

printf("Iskomoe znachenie K ravno %d\n\n", K);

else

printf("Iskomoe znachenie dlya zadannogo massiva M[%d] ne suschestvuet\n\n", N);

}

int FirstLab2() { //функция предлагающая другой метод нахождения

//правой и левой сумм элементов массива

int R, Z=0, z=0, SumL=0, SumP=0, i, j, h, Ki, K=N, M[N] = {801, 30, -2, 8, 79, -5000, 2, 3, 6, 15}; //объявление переменных и определение массива

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

printf("M[%d] = %d\n", i, M[i]);

for(h=0;h<N;h++) { //цикл определяющий сумму значений элементов

//массива:

Z+=abs(M[h]); //по модулю (Z) - для сравнения с разностями

z+=M[h]; //сумма без модуля (z) - для информативности

}

printf("\nSumma znacheniy elemeentov massiva M[%d] ravna %d\n\n", N, z);

for(i=0;i<N;i++) { //цикл для нахождения суммы значений левой

//стороны массива

for(j=N-1;j>=i+1;j--) { //цикл для нахождения суммы значений

//правой стороны массива

SumP+=M[j];

}

SumL+=M[i];

R=abs(SumL-SumP); //нахождение разности сумм значений элементов

//массива с левой и правой стороны

Ki=i;

printf("SumL = %d, SumP = %d, R = %d, K = %d\n\n", SumL, SumP, R, Ki);

SumP=0; //обнуление суммы значений элементов с правой

//стороны массива для следующей итерации

if (R<=Z) { //проверка на минимальную разность сумм начений

//элементов массива с правой и левой сторон

Z=R;

K=i;

}

}

if (K!=N)

printf("Iskomoe znachenie K = %d\n\nVsego dobrogo!", K);

else

printf("Iskomoe znachenie dlya zadannogo massiva M[%d] ne suschestvuet\n\nVsego dobrogo!", N);

}

Защита:

1)

int F4(int n)

{ int k,m;

for (k=0, m=1; m <= n; k++, m = m \* 2);

return k-1; }

Поиск степени двойки, при которой число m будет максимально приближено к заданному числу n.

2)

int F6(int c[], int n)

{ int i,j,k1,k2;

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

{

for (j=k1=k2=0; j<n; j++)

if (c[i] != c[j])

{ if (c[i] < c[j]) k1++; else k2++; }

if (k1 == k2) return i;

}

return -1; }

Поиск середины массива по значению элементов, как только нашли, возвращаем его индекс.