МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 05

**Программирование с использованием одномерных массивов**

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.Е. Хлыстов

Проверил: к.т.н,, доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Г. Скурыдин

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2018

**Задача №1**

1. **Формулировка задачи**

В массиве C(n) подсчитать количество отрицательных и сумму положительных элементов.

1. **Постановка задачи**

На вход поступает положительное вещественное число n, которое является количеством элементов в линейном массиве. Далее на вход поступает последовательность из n вещественных чисел, образующих элементы массива. На выход поступает целое положительное число, определяющее количество отрицательных элементов массива и вещественное число, определяющее сумму его положительных элементов.

1. **Математическая модель**

Рассмотрим математическую модель на примере. Вводится массив: a[1]=10, a[2]=-1, a[3]=4, a[4]=-19, a[5]=5. Проверяется знак элементов: a[1]=10>0, к значению суммы положительных прибавляется значение данного элемента – sum+=a[1]. a[2]=-1<0, к значению счетчика отрицательных прибавляется единица – sch++, и так далее. Выходит что есть отрицательные: a[2]=-1, a[4]=-19 – и положительные: a[1]=10, a[3]=4, a[5]=5. Счетчик отрицательных выдаст ответ 2, так как отрицательных элемента 2. Сумма положительных равна 19.

1. **Описание алгоритма**

**Начало алгоритма основной программы**

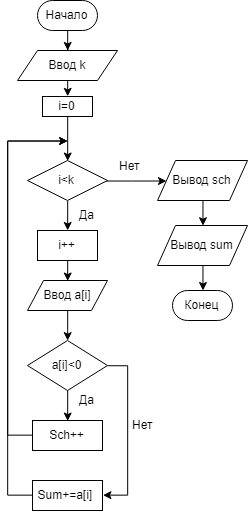
1. Ввод k //количество элементов массива
2. i=0
3. Проверка условия i<k. Если неверно, то переходит к пункту 8
4. i++
5. Ввод a[i] //значение элемента
6. Проверка условия a[i]<0. Если неверно, то переходит к пункту 7
7. Sch++ // подсчет количества отрицательных значений, переход к пункту 2
8. Sum+=a[i] //сумма положительных, переход к пункту 2
9. Вывод количества отрицательных
10. Вывод суммы положительных

**Конец алгоритма основной программы**

1. **Проект программы с определением замкнутых программных единиц и необходимых структур данных**

В данном случае программа состоит из трех частей – 2 блока ввода, 2 блока сравнения и блока вывода

1. **Опорный граф (блок-схема) алгоритма**



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <conio.h>

int main(int argc, char const \*argv[]) {

int k, sum, sch=0;

std::cout << "vvtdite k" << '\n';

std::cin >> k;

double a[k];

for (int i=0; i<k; i++)

{

std::cout << "a[" << i <<"]=";

std::cin >> a[i];

if (a[i]<0)

sch++;

else

sum+=a[i];

}

std::cout << "kollichestvo otrecatelnih = " <<sch<< '\n';

std::cout << "summa polojitelnih = " <<sum<< '\n';

return 0;

}

**на языке Pascal**

program zzz;

const k=10;

var

sch, i: integer;

sum:real;

a : array [1..k] of real;

begin

sch:=0;

sum:=0;

// writeln ('vvtdite k ');

// read (k);

for i:=1 to k do begin

write ('a[', i, ']= ');

read (a[i]);

if a[i]<0 then

sch:=sch+1

else

sum+=a[i];

end;

writeln ('kollichestvo otrecatelnih =', sch);

writeln ('summa polojitelnih = ', sum);

readln;

end.

1. **Проверка работоспособности (тестирование) программы**

Ввод массива a[1]=1, a[2]=-2, a[3]=3. Программа выдает значение количества отрицательных равное 1 и сумму положительный 4, что является правильным ответом.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.

**Задача №2**

1. **Формулировка задачи**

Даны действительные числа a1, a2, …, a2n. Получить a2n, a1, a2,…,a2n-1.

1. **Постановка задачи**

На вход поступает положительное целое число k, определяющее количество элементов в массиве. Далее на вход идут 2\*k вещественных элементов массива. Программа переносит значение последнего элемента на место первого, и смещает массив на один элемент вправо. На выход идет массив из 2\*k целых чисел.

1. **Математическая модель**

Разберем математическую модель на примере. Вводится значение k равное 3, определяющее количество элементов в массиве. Вводится массив a[2\*k], таким образом, что – a[1]=1, a[2]=2, a[3]=3, a[4]=4, a[5]=5, a[6]=6. Вводится второй массив b[2\*k], таким образом, что значение первого элемента массива b[1] становиться равно a[6]=6. Далее элементы массива b[2], b[3], b[4], b[5], b[6] принимают значение переменных a[1], a[2], a[3], a[4], a[5] соответственно, то есть b[2]= a[1]=1, b[3]= a[2]=2, b[4]= a[3]=3, b[5]= a[4]=4, b[6]= a[5]=5.

1. **Описание алгоритма**

**Начало алгоритма основной программы**

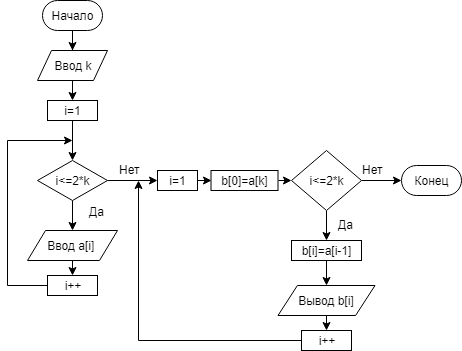
1. Ввод k // количество элементов массива
2. i=1.
3. Проверка условия i<=2\*k. Если неверно, то переходит к пункту 6
4. Ввод a[i]
5. i++ // Значение элемента, переход к пункту 3
6. i=1.
7. b[1]=a[2\*k]
8. Проверка условия i<=2\*k. Если неверно, то завершает работу программы
9. b[i]=a[i-1]
10. Вывод b[i]
11. i++, переход к пункту 8

**Конец алгоритма основной программы**

1. **Проект программы с определением замкнутых программных единиц и необходимых структур данных**

В данном случае программа состоит из трех частей – 2 блока ввода, 2 блока сравнения и блока вывода

1. **Опорный граф (блок-схема) алгоритма**

****

1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <conio.h>

int main(int argc, char const \*argv[]) {

int n, k;

std::cout << "vvtdite k" << '\n';

std::cin >> k;

double a[2\*k], b[2\*k];

for (int i=1; i<=2\*k; i++)

{

std::cout << "a[" << i <<"]=";

std::cin >> a[i];

}

b[1]=a[2\*k];

for (int i=1; i<=2\*k; i++)

{

b[i]=a[i-1];

std::cout << "a[" << i <<"]=" << b[i] << '\n';

}

return 0;

} **на языке Pascal**

program zzz;

const k=3;

var

i: integer;

a : array [1..2\*k] of integer;

b : array [1..2\*k] of integer;

begin

for i:=1 to 2\*k do begin

write ('a[', i, ']=');

read (a[i]);

end;

b[1]:=a[2\*k];

for i:=1 to 2\*k do begin

b[i]:=a[i-1];

writeln ('a[', i, ']=', b[i]);

end;

readln(i);

end.

1. **Проверка работоспособности (тестирование) программы**

Для проверки программы введем массив a[1]=1, a[2]=2, a[3]=3. Программа выдаст значения 3, 1, 2, что является верным решением.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.