МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра информатики и вычислительной техники (ИВТ)

Лабораторная работа №05

**Программирование с использованием одномерных массивов**

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.М. Губченко

Проверил: к.т.н,, доцент каф. ВТиЭ

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

Лабораторная работа защищена

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

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

Барнаул 2019

1. **Формулировка задачи**

Выполнить арифметический сдвиг элементов 10-элементного массива на три позиции влево.

1. **Постановка задачи**

Арифметический сдвиг элементов массива на три позиции влево- это процесс, при котором первый элемент массива остаётся неизменным, а все последующие значения смещаются на 3 позиции влево. Последние три элемента массива заполняются нулями.

Инициализируется 10 элементный массив и программа выполняет арифметический сдвиг элементов на три позиции влево. На выход поступают 10 действительных чисел.

1. **Математическая модель**

Для построения математической модели рассмотрим пример. Инициализируется 10-элементный массив, с значениями вида a[i] a[1]=1, a[2]=2, a[3]=3, a[4]=4, a[5]=5, a[6]=6, a[7]=7, a[8]=8, a[9]=9, a[10]=10. После этого выполняется проверка условия, если i меньше 10 и не равно единице, то, если i меньше 8, то этому элементу массива присваивается значение i+3-го элемента массива, иначе присваивается значение равное нулю. В итоге элементу a[2] присваивается значение a[5], a[3] присваивается значение a[6] и так далее. Элементы a[8], a[9] и a[10] становятся равны нулю. Программа выводит конечный массив 1 5 6 7 8 9 10 0 0 0.

1. **Описание алгоритма**

**Начало алгоритма**

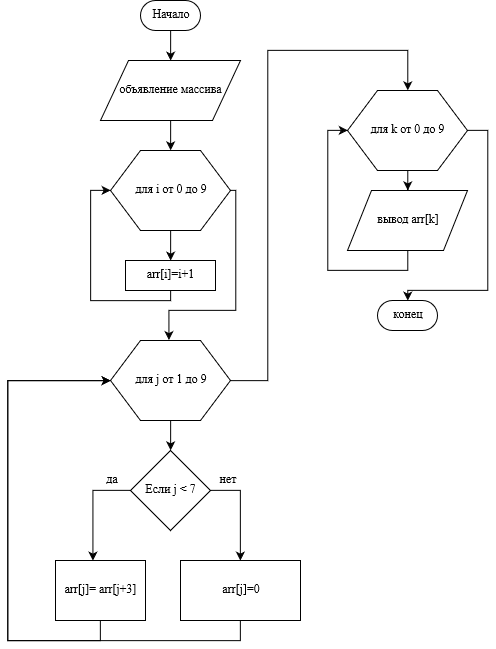
* 1. Инициализируется массив размерностью 10
  2. Присвоение переменной i значения 0
  3. Если i > 10, переход к пункту 4.6
  4. Присвоение i-ому элементу массива значения i+1
  5. Присвоение переменной i значения i+1, переход к пункту 4.3
  6. Присвоение переменной j значения 1
  7. Если j>9 переход к пункту 4.11
  8. Если j >=7 переход к пункту 4.10
  9. Присвоение j-ому элементу массива значения j+3-го элемента**,** переход к пункту 4.8
  10. Присвоение j-ому элементу значения 0, переход к пункту 4.7
  11. Присвоение переменной k значения 0
  12. Если k>=9, конец выполнения программы
  13. Вывод значения k-го элемента массива
  14. Присвоение переменной k значения k+1, переход к пункту 4.12

**Конец алгоритма**

1. **Проект программы с выполнением арифметического сдвига 10-элементного массива**

Проект программы с выполнением арифметического сдвига 10-элементного массива. В данном случае программа должна состоять из двенадцати частей: цикл с параметром (содержит блок определения переменной i, 1 блок арифметических действий и 1 блок вывода), цикл с параметром(содержит блок определения переменной j, 1 блок проверки условия, 2 блока арифметических действий), цикл с параметром(содержит блок определения переменной k, блок вывода).

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



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int arr[9];

cout << "Input array" << '\n';

for (int i = 0; i < 10; i++) {

arr[i] = i+1;

cout << arr[i] << " ";

}

for (int j = 1; j<=9; j++)

{

if (j < 7) {

arr[j] = arr[j+3];

} else {

arr[j] = 0;

}

}

cout << " " << '\n';

for (int k = 0; k < 9; k++)

{

cout << arr[k] << " ";

}

cout << "0\n";

return 0;

}

**на языке Pascal**

program lab\_05;

var

i, j, k: integer;

arr: array [1..10] of integer;

begin

WriteLn('Input array');

for i := 1 to 10 do

begin

arr[i] := i;

Write(arr[i], ' ');

end;

for j := 2 to 10 do

begin

if j < 8 THEN

begin

arr[j] := arr[j + 3];

end ELSE

begin

if j >= 8 THEN begin

arr[j] := 0;

end;

end;

end;

WriteLn();

for k := 1 to 9 do

begin

Write(arr[k], ' ');

end;

WriteLn(0);

end.

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

Для проверки работоспособности программы необходимо просто посмотреть на то, что выведет программа в конце работы.

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

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

1. **Формулировка задачи**

Даны три целочисленных массива A, B, C размерности NA, NB, NC соответственно. Элементы всех массивов упорядочены по убыванию. Объединить эти массивы таким образом, чтобы результирующий массив D оставался упорядоченным по убыванию.

1. **Постановка задачи**

На вход программы поступают три целочисленных массива A, B, C размерности NA, NB, NC соответственно,где NA, NB, и NC целые, положительные числа. После этого они сортируются по убыванию. После этого массивы объединяются в массив D размерностью NA+NB+NC, который сортируется по убыванию. На выход поступают элементы массива D.

1. **Математическая модель**

Для построения математической модели рассмотрим пример. Инициализируются три массива A, B, C размерностью 3, 4, 7 соответственно. Элементы массива принимают значения a[1]=5, a[2]=3, a[3]=20. После сортировки массив принимает вид a[1]=20, a[2]=5, a[3]=3. Аналогично получаем массив B с элементами b[1]=10, b[2]=7, b[3]=3, b[4]=1 и массив C с элементами c[1]=24, c[2]=14, c[3]=7, c[4]=6, c[5]=4, c[6]=2, c[7]=0. После объединения массивов A, B и С в массив D и последующей сортировки получаем d[1]=24, d[2]=20, d[3]=20, d[4]=14, d[5]=10, d[5]=7, d[6]=7, d[7]=6, d[8]=5, d[9]=4, d[10]=3, d[11]=3, d[12]=2, d[13]=1, d[14]=0. На выход поступает один целочисленный массив D.

1. **Описание алгоритма**

**Начало алгоритма**

* 1. Присвоение переменной a значения 3
  2. Присвоение переменной b значения 4
  3. Присвоение переменной c значения 7
  4. Присвоение переменной d значения a+b+c
  5. Инициализация массива arrA размерностью 3
  6. Инициализация массива arrB размерностью 4
  7. Инициализация массива arrC размерностью 7
  8. Инициализация массива arrD размерностью 14
  9. Присвоение переменной i значения 0
  10. Если i>=a переход к пункту 4.13
  11. Присвоение переменной i значения i++
  12. Присвоение элементу arrA[i] значения (rand() % 20), переход к пункту 4.10
  13. Присвоение переменной i значения 0
  14. Если i>=b переход к пункту 4.17
  15. Присвоение переменной i значения i++
  16. Присвоение элементу arrB[i] значения (rand() % 20), переход к пункту 4.14
  17. Присвоение переменной i значения 0
  18. Если i>=c переход к пункту 4.21
  19. Присвоение переменной i значения i++
  20. Присвоение элементу arrC[i] значения (rand() % 20), переход к пункту 4.18
  21. Вызов функции(сортировка, arrA, a)
  22. Вызов функции(сортировка, arrB, b)
  23. Вызов функции(сортировка, arrC, c)
  24. Присвоение переменной i значения 0
  25. Если i>=a переход к пункту 4.28
  26. Присвоение переменной i значения i++
  27. Присвоение элементу arrD[i] значения arrA[i], переход к пункту 4.25
  28. Присвоение переменной i значения 0
  29. Если i>=b переход к пункту 4.32
  30. Присвоение переменной i значения i++
  31. Присвоение элементу arrD[i + a] значения arrB[i], переход к пункту 4.29
  32. Присвоение переменной i значения 0
  33. Если i>=c переход к пункту 4.36
  34. Присвоение переменной i значения i++
  35. Присвоение элементу arrD[i+a+b] значения arrC[i], переход к пункту 4.33
  36. Вызов функции(сортировка, arrD, d)
  37. Вывод массива D

**Конец алгоритма**

**Начало сортировка**

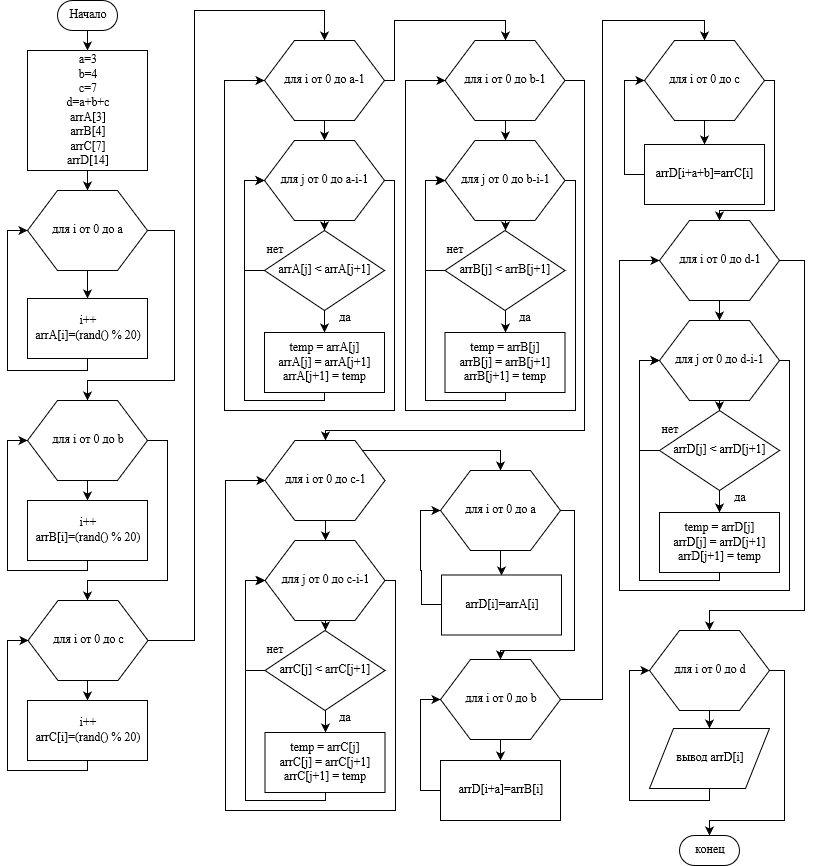
* 1. Присвоение переменной i значения 0
  2. Если i >= a-1 конец функции
  3. Присвоение переменной i значения i++
  4. Присвоение переменной j значения 0
  5. Если j >=a-i-1 переход к пункту 4.2
  6. Присвоение переменной j значения j++
  7. Если arr[j]>=arr[j+1] переход к пункту 4.5
  8. Присвоение переменной temp значения arr[j]
  9. Присвоение переменной arr[j] значения arr[j+1]
  10. Присвоение переменной arr[j+1] значения temp

**Конец сортировка**

1. **Проект программы с объединением трёх упорядоченных по убыванию массивов в четвёртый упорядоченный по убыванию массив**

Проект программы с объединением трёх упорядоченных по убыванию массивов в четвёртый упорядоченный по убыванию массив замкнутых. В данном случае программа должна состоять из 30 частей: 5 блоков определения переменной, 4 блока определения массивов, 3 цикла с параметром (содержат 2 блока арифметических действий), 4 цикла с параметром(содержат цикл с параметром[содержит блок проверки условия и 3 блока арифметических действий]), 4 цикла с параметром(содержат блок арифметических действий и блок вывода), 3 цикла с параметром(содержат 2 блока арифметических действий). В качестве структуры данных могут быть введены переменные для хранения переменной определяющей точность вычислений, промежуточных переменных и результата.

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



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

int main() {

srand(time(NULL));

int a = 3, b = 4, c = 7, d, temp;

int arrA[3];

int arrB[4];

int arrC[7];

int arrD[14];

d = a + b + c;

for (int i = 0; i < a; i++) {

arrA[i] = (rand() % 20);

}

for (int i = 0; i < b; i++) {

arrB[i] = (rand() % 20);

}

for (int i = 0; i < c; i++) {

arrC[i] = (rand() % 20);

}

//сортировка массивов

cout << "Array A: ";

for (int i = 0; i < a-1; i++) {

for (int j = 0; j < a-i-1; j++) {

if (arrA[j] < arrA[j+1]){

temp = arrA[j];

arrA[j] = arrA[j+1];

arrA[j+1] = temp;

}

}

}

for (int i = 0; i < a; i++) {

std::cout << arrA[i] << " ";

}

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

cout << "Array B: ";

for (int i = 0; i < b - 1; i++) {

for (int j = 0; j < b - i - 1; j++) {

if (arrB[j] < arrB[j + 1]) {

temp = arrB[j];

arrB[j] = arrB[j + 1];

arrB[j + 1] = temp;

}

}

}

for (int i = 0; i < b; i++) {

std::cout << arrB[i] << " ";

}

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

cout << "Array C: ";

for (int i = 0; i < c - 1; i++) {

for (int j = 0; j < c - i - 1; j++) {

if (arrC[j] < arrC[j + 1]) {

temp = arrC[j];

arrC[j] = arrC[j + 1];

arrC[j + 1] = temp;

}

}

}

for (int i = 0; i < c; i++) {

std::cout << arrC[i] << " ";

}

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

//сортировка окончена

//Объединение массивов

for (int i = 0; i < a; i++) {

arrD[i] = arrA[i];

}

for (int i = 0; i < b; i++) {

arrD[i + a] = arrB[i];

}

for (int i = 0; i < c; i++) {

arrD[i + b + a] = arrC[i];

}

for (int i = 0; i < d - 1; i++) {

for (int j = 0; j < d - i - 1; j++) {

if (arrD[j] < arrD[j + 1]) {

temp = arrD[j];

arrD[j] = arrD[j + 1];

arrD[j + 1] = temp;

}

}

}

cout << "Array D: ";

for (int i = 0; i < d; i++) {

cout << arrD[i] << " ";

}

return 0;

}

**На языке Pascal**

**program** lab\_052;

**var** i, j, a, b, c, d, temp:integer;

arrA: **array** [1..3] **of** integer;

arrB: **array** [1..4] **of** integer;

arrC: **array** [1..7] **of** integer;

arrD: **array** [1..14] **of** integer;

**BEGIN**

a:=3;

b:=4;

c:=7;

d:=14;

arrA[1]:=20;

arrA[2]:=4;

arrA[3]:=10;

arrB[1]:=16;

arrB[2]:=42;

arrB[3]:=58;

arrB[4]:=65;

arrC[1]:=99;

arrC[2]:=103;

arrC[3]:=71;

arrC[4]:=32;

arrC[5]:=6;

arrC[6]:=6;

arrC[7]:=6;

**for** i := 1 **to** a-1 **do**

**begin**

**for** j := 1 **to** a-i-1 **do**

**begin**

**if** arrA[j] < arrA[j+1] **THEN**

**begin**

temp:= arrA[j];

arrA[j]:= arrA[j+1];

arrA[j+1]:= temp;

**end**

**end**

**end**;

**for** i :=1 **to** a **do**

**begin**

Write(arrA[i], ' ');

**end**;

WriteLn(' ');

**for** i := 1 **to** b-1 **do**

**begin**

**for** j := 1 **to** b-i-1 **do**

**begin**

**if** arrB[j] < arrB[j+1] **THEN**

**begin**

temp:= arrB[j];

arrB[j]:= arrB[j+1];

arrB[j+1]:= temp;

**end**

**end**

**end**;

**for** i :=1 **to** b **do**

**begin**

Write(arrB[i], ' ');

**end**;

WriteLn(' ');

**for** i := 1 **to** c-1 **do**

**begin**

**for** j := 1 **to** c-i-1 **do**

**begin**

**if** arrC[j] < arrC[j+1] **THEN**

**begin**

temp:= arrC[j];

arrC[j]:= arrC[j+1];

arrC[j+1]:= temp;

**end**

**end**

**end**;

**for** i :=1 **to** c **do**

**begin**

Write(arrC[i], ' ');

**end**;

WriteLn(' ');

**for** i := 1 **to** a **do**

**begin**

arrD[i]:=arrA[i];

**end**;

**for** i := 1 **to** b **do**

**begin**

arrD[i+a]:=arrA[i];

**end**;

**for** i := 1 **to** c **do**

**begin**

arrD[i+a+b]:=arrA[i];

**end**;

**for** i := 1 **to** d-1 **do**

**begin**

**for** j := 1 **to** c-i-1 **do**

**begin**

**if** arrD[j] < arrD[j+1] **THEN**

**begin**

temp:= arrD[j];

arrD[j]:= arrD[j+1];

arrD[j+1]:= temp;

**end**

**end**

**end**;

**for** i :=1 **to** d **do**

**begin**

Write(arrD[i], ' ');

**end**

**END**.

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

Для проверки работоспособности программы необходимо посмотреть на то, что выведет программа.

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

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