МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра информатики и вычислительной техники (ИВТ)

Лабораторная работа №06

**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**

С помощью датчика случайных чисел сформировать двумерный массив n×n и упорядочить (переставить) строки матрицы по не возрастанию сумм элементов строк.

1. **Постановка задачи**

На вход программы поступает целое положительное число n определяющее размерность массива вида n×n. После этого программа находи суммы строк матрицы и сортирует строки по невозрастанию сумм. На выход поступает матрица вида n×n.

1. **Математическая модель**

Для построения математической модели рассмотрим пример. Инициализируется 9-элементный массив вида 3×3, с значениями вида arr(line, column) arr(1, 1)=10, arr(1, 2)=10, arr(1, 3)=10, arr(2, 1)=6, arr(2, 2)= 10, arr(2, 3)=24, arr(3, 1)=2, arr(3, 2)=4, arr(3, 3)=1. После этого считается сумма первой строки равная 30 и сумма второй строки равная 40. Выполняется проверка условия, если сумма первой меньше суммы второй строки, то элементам arr(1, 1), arr(1, 2), arr(1, 3) присваиваются значения arr(2, 1), arr(2, 2) и arr(2, 3), а элементам arr(2, 1) arr(2, 2), arr(2, 3) присваиваются значения arr(1, 1), arr(1, 2), arr(1, 3) соответственно. После этого считается сумма третьей строки равная 7. В данном случае сумма второй строки больше чем сумма третьей строки, значит строки местами не меняются. В конце выполнения алгоритма программа выводит матрицу  
6 10 24

10 10 10

2 4 1

1. **Описание алгоритма**

**Начало алгоритма**

**//**генерация массива

* 1. Ввод переменной n
  2. Присвоение переменной line значения 0
  3. Если line >= n переход к пункту 4.10
  4. Присвоение переменной line значения line++
  5. присвоение переменной column значения 0
  6. Если column >= n переход к пункту 4.3
  7. Присвоение переменной column значения column++
  8. Присвоение переменной arr(line, column) значения (rand() % 20)
  9. Вывод arr(line, column)и переход к пункту 4.6

//сортировка матрицы

* 1. Присвоение переменной sum1 значения 0
  2. Присвоение переменной sum2 значения 0
  3. Присвоение переменной line значения 0
  4. Присвоение переменной q значения 0
  5. Если q >= n переход к пункту 4.40
  6. Присвоение переменной q значения q++

//считаю сумму первой строки

* 1. Присвоение переменной column значения 0
  2. Если column >= n переход к пункту 4.20
  3. Присвоение переменной column значения column++
  4. Присвоение переменной sum1 значения sum1+=arr(line, column) и переход к пункту 4.17
  5. Вывод “sum1= “ sum1

//Считаю сумму второй строки

* 1. Присвоение переменной line значения 1
  2. Если line >= n переход к пункту 4.38
  3. Присвоение переменной line значения line++
  4. Присвоение переменной column значения 0
  5. Если column >= n переход к пункту 4.28
  6. Присвоение переменной column значения column++
  7. Присвоение переменной sum2 значения sum2+= arr(line, column) и переход к пункту 4.25
  8. Вывод “Sum2= “ sum2
  9. Если sum2 < sum1 переход к пункту 4.36
  10. Присвоение переменной a значения line
  11. Присвоение переменной b значения line – 1
  12. Присвоение переменной k значения 0
  13. Если k >= n переход к пункту 4.36
  14. Присвоение переменной k значения k++
  15. Выполнения функции swap(arr(a, k), arr(b, k)) и переход к пункту 4.33
  16. Присвоение переменной sum1 значения sum2
  17. Присвоение переменной sum2 значения 0 и переход к пункту 4.25
  18. Присвоение переменной column значения 0
  19. Присвоение переменной line значения 1 и переход к пункту 4.14

//вывод матрицы

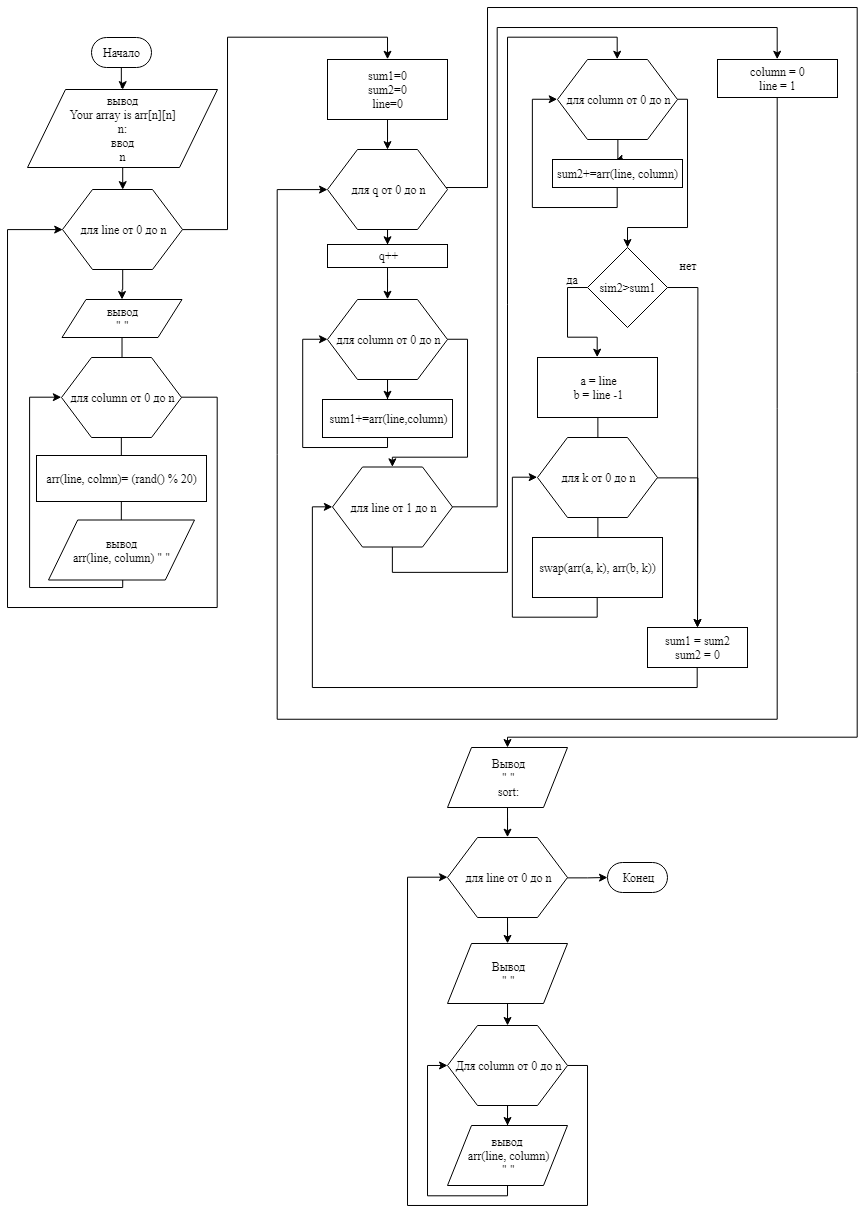
* 1. Вывод “sort:“
  2. Присвоение переменной line значения 0
  3. Если line >=n переход завершение работы программы
  4. Присвоение переменной line значения line++
  5. Вывод “ “
  6. Присвоение переменной column значения 0
  7. Если column >= n переход к пункту 4.42
  8. Присвоение переменной column значения column++
  9. Вывод arr(line, column) “ “ и переход к пункту 4.46

**Конец алгоритма**

1. **Проект программы с сортировкой строк матрицы n**×**n по не убыванию**

В данном случае программа должна состоять из 39 частей: 2 блока вывода, блок ввода, цикл с параметром(блок вывода, блок арифметических действий, цикл с параметром [2 блока арифметических действий, блок вывода]), три блока арифметических действий, цикл с параметром(3 блока арифметических действий, цикл с параметром[2 блока арифметических действий], цикл с параметром[4 блока арифметических действий, цикл с параметром(2 блока арифметических действий) блок проверки условия, цикл с параметром(одна функция)]), 2 блока вывода, цикл с параметром(блок арифметических действий, блок вывода, цикл с параметром[блок арифметических действий, блок вывода])

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



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <ctime>

#include <cstdlib>

#define arr(line,column) \*(arr+line\*n+column)

using namespace std;

int main()

{

srand(time(NULL));

int n, line, column, sum1, sum2, temp;

//ввод размерности

cout << "Your array is arr[n][n]" << endl;

cout << "n: ";

cin >> n;

//генерация массива

int \* arr = (int \*)malloc(n \* n \* sizeof(int));

for (int line=0; line < n; line++) {

cout << " " << endl;

for (int column =0; column < n; column++) {

arr(line, column)= (rand() % 20);

cout << arr(line, column) << " ";

}

}

//сортировка

sum1 = 0;

sum2 = 0;

line = 0;

for (int q = 0; q < n; q++)

{

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

sum1 += arr(line, column);//считаю сумму чисел первой строки

}

//cout << "sum1= " << sum1 << endl;

for (line = 1; line < n; line++)

{

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

{

sum2 += arr(line, column);//считаю сумму чисел следующей строки

}

//cout << "Sum2= " << sum2 << endl;

if (sum2 > sum1)

{

int a = line;

int b = line - 1;

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

{

swap(arr(a, k), arr(b, k));

}

}

/\*cout << "-----" << endl;

for (int line = 0; line < n; line++)

{

cout << " " << endl;

for (int column = 0; column < n; column++) {

cout << arr(line, column) << " ";

}

}

cout << "-----" << endl;\*/

//проверка массивов

sum1 = sum2;

sum2 = 0;

}

column = 0;

line = 1;

}

//вывод массива

cout << " " << endl;

cout << "sort:" << endl;

for (int line = 0; line < n; line++) {

cout << " " << endl;

for (int column = 0; column < n; column++) {

cout << arr(line, column) << " ";

}

}

return 0;

}

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**

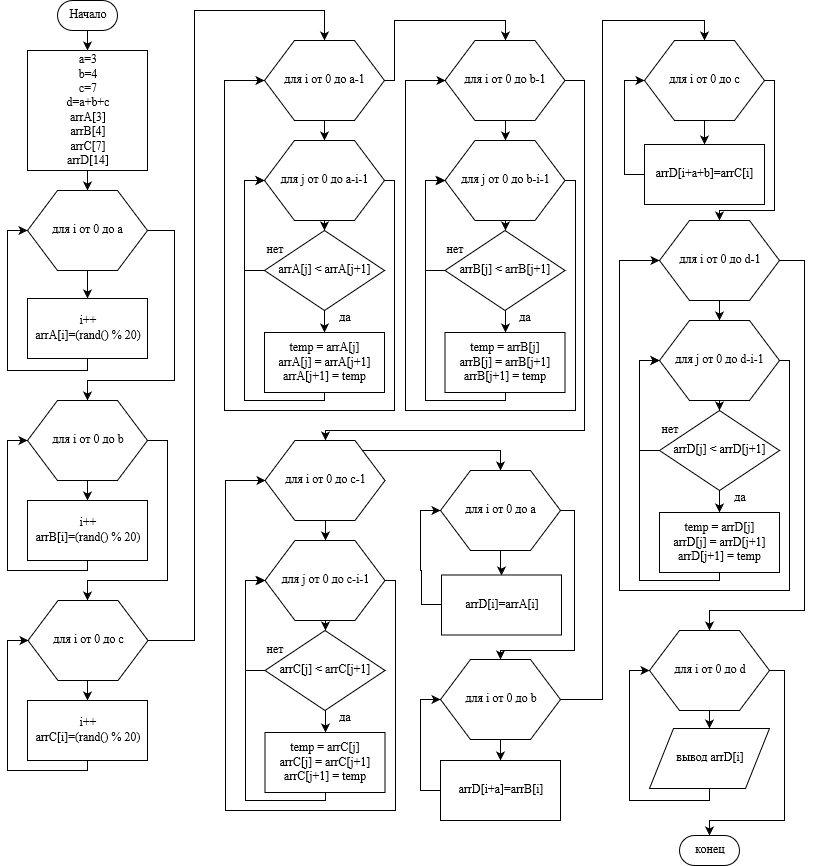
* 1. Присвоение переменной i значения 0
  2. Если i >= a-1 конец функции
  3. Присвоение переменной i значения i++
  4. Присвоение переменной j значения 0
  5. Если j >=a-i-1 переход к пункту 4.2
  6. Присвоение переменной j значения j++
  7. Если arr[j]>=arr[j+1] переход к пункту 4.5
  8. Присвоение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