Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 17

По дисциплине «Основы алгоритмизации и программирования»

На тему «Указатели и ссылки при работе с функциями»

 Выполнила:

Студентка1 курса 6 группы

Альшевская Алина Михайловна

Преподаватель: асс. Андронова М.В.

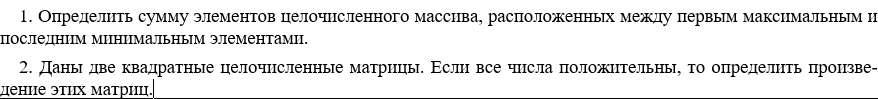
2023, Минск

5. В соответствии со своим вариантом написать главную функцию, в которой имеются вызовы ***функций пользователя***, реализующих задачи из таблицы ниже. Ввод исходных данных и вывод результатов осуществить в главной функции, при этом использовать***динамические* *массивы***.

Для передачи параметров в функции пользователя и возвращения результатов применить ***указатели*** и ***ссылки***.

Вариант 1

Задание 5



#include <iostream>

#include <Windows.h>

using namespace std;

int& sumBetweenMinMax(int\* ptr, int& size, int& sum) {

int max = \*ptr, indexMax = 0, min = \*ptr, indexMin = 0;

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

//находим и сохраняем индекс максимального элемента

if (\*(ptr + i) > max) {

max = \*(ptr+i);

indexMax = i;

}

//находим и сохраняем индекс минимального элемента

if (\*(ptr + i) < min) {

min = \*(ptr + i);

indexMin = i;

}

}

//Поскольку индекс минимального может быть больше индекса максимального и наоборот, использую стандартные функции для выбора минимального и максимального из них для выбора индекса начала и конца цикла соответственно

int start = min(indexMax, indexMin);

int end = max(indexMax, indexMin);

//суммируем все эти элементы (++start поскольку первый элемент по индексу начала цикла не нужен)

for (int i = ++start; i < end; i++) {

sum += \*(ptr + i);

}

//возвращаем ссылку на sum

return sum;

}

int main() {

SetConsoleOutputCP(1251);

int size = 15, sum = 0;

int rmn = 1, rmx = 10;

//создаем массив и заполняем его случайными числами

int\* ptr = new int[size];

srand((unsigned)time(NULL));

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

ptr[i] = rand() % (rmx - rmn) + rmn;

cout << ptr[i] << ' ';

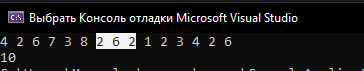
}

cout << '\n';

sumBetweenMinMax(ptr, size, sum);

cout << sum;

}



#include <iostream>

#include <ctime>

using namespace std;

//Функция для создания и заполнения матрицы случайными числами

int\*\* createAndFillMatrix(int size, int rmn, int rmx) {

//Создаем матрицу размерности size

int\*\* matrix = new int\* [size];

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

matrix[i] = new int[size];

}

//Заполняем матрицу случайными положительными числами

srand((unsigned)time(NULL));

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

for (int j = 0; j < size; j++) {

matrix[i][j] = rand() % (rmx - rmn) + rmn;

cout << matrix[i][j] << ' ';

}

cout << '\n';

}

return matrix;

}

//Функция для перемножения двух матриц

int\*\* multiplyMatrices(int\*\* matrix1, int\*\* matrix2, int size) {

//Создаем матрицу размерности size, которая будет хранить результат умножения матриц

int\*\* resultMatrix = new int\* [size];

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

resultMatrix[i] = new int[size];

}

//перемножение матриц

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

for (int j = 0; j < size; j++) {

resultMatrix[i][j] = 0;

for (int k = 0; k < size; k++) {

resultMatrix[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

return resultMatrix;

}

int main() {

int size = 3; //размер квадратных матриц

int rmn = 1, rmx = 10; //минимальное и максимальное значения для рандома

//Создаем и заполняем две матрицы

int\*\* matrix1 = createAndFillMatrix(size, rmn, rmx);

int\*\* matrix2 = createAndFillMatrix(size, rmn, rmx);

//результат перемножения матриц

int\*\* resultMatrix = multiplyMatrices(matrix1, matrix2, size);

cout << "\nРезультат перемножения матриц:\n";

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

for (int j = 0; j < size; ++j) {

cout << resultMatrix[i][j] << ' ';

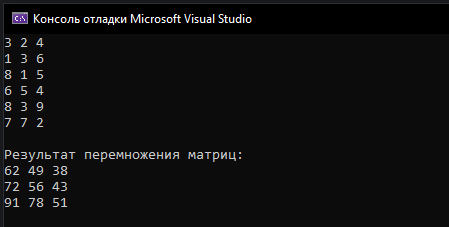
}

cout << '\n';

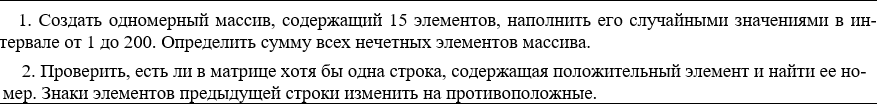
}

return 0;

}



Вариант 4



#include <iostream>

#include <ctime>

using namespace std;

int& sumBetweenZeros(int\* arr, int size, int& sum) { // Функция, возвращающая ссылку на сумму

sum = 0;

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

if (arr[i] % 2 != 0) { // Если элемент нечетный

sum += arr[i];

}

}

// Возвращаем ссылку на сумму

return sum;

}

int main() {

int size = 15, rmn = 1, rmx = 200;

int\* ptr = new int[size];

// Заполняем массив случайными числами и сразу выводим

srand((unsigned)time(NULL));

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

ptr[i] = rand() % (rmx - rmn) + rmn;

cout << ptr[i] << ' ';

}

cout << '\n';

int sum = 0; // Переменная, которая будет хранить сумму чисел

sum = sumBetweenZeros(ptr, size, sum);

cout << "Сумма: " << sum;

return 0;

}



#include <iostream>

#include <Windows.h>

using namespace std;

bool findPositiveRowAndInvertPrevious(int\*\* matrix, int rows, int cols, int& rowIndex) {

bool found = false;

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

for (int j = 0; j < cols; ++j) {

if (matrix[i][j] > 0) { // Если число положительное

found = true; // Устанавливаем флаг, что было найдено положительное число

rowIndex = i; // Сохраняем индекс строки

break; // Выходим из цикла для элементов строки

}

}

if (found) { // Если было найдено положительное число

if (rowIndex > 0) { // Меняем предыдущую строку, если это не первая строка. Иначе не меняем

for (int j = 0; j < cols; ++j) {

matrix[rowIndex - 1][j] \*= -1;

}

}

break; // Выходим из цикла

}

}

// Возвращаем флаг, который означает, было ли найдено положительное число

return found;

}

int main() {

SetConsoleOutputCP(1251);

int n = 10, m = 3, rmn = -10, rmx = 10;

// Создаем массив n на m с использованием new

int\*\* ptr = new int\* [n];

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

ptr[i] = new int[m];

srand((unsigned)time(NULL));

// Создаем массив из случайных чисел и сразу выводим его

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

for (int j = 0; j < m; j++) {

ptr[i][j] = rand() % (rmx - rmn) + rmn;

cout << ptr[i][j] << " ";

}

cout << '\n';

}

int rowIndex = 0; // Переменная, которая будет хранить индекс строки, содержащей положительное число

bool wasFoundPositiveNumber;

// Вызываем функцию и сохраняем ее результат в переменную

wasFoundPositiveNumber = findPositiveRowAndInvertPrevious(ptr, n, m, rowIndex);

if (wasFoundPositiveNumber) { // Если было найдено положительное число

cout << "Было найдено положительное число в строке под номером: " << rowIndex + 1 << endl;

cout << "\nНовая матрица: " << endl;

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

for (int j = 0; j < m; j++) {

cout << ptr[i][j] << ' ';

}

cout << '\n';

}

}

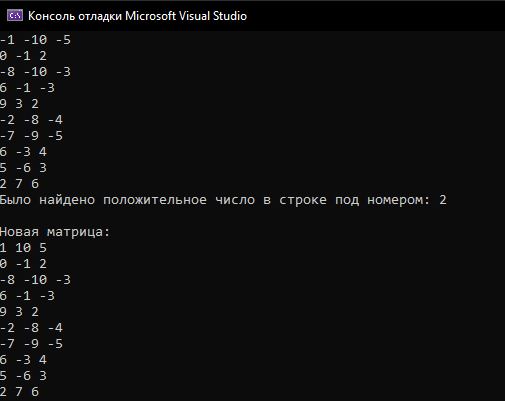
else {

cout << "Все числа не положительные";

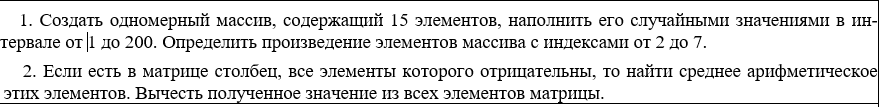
}

return 0;

}



Вариант 7



#include <iostream>

using namespace std;

// Функция, возвращающая ссылку на произведение элементов с индексами от 2 до 7

long long& productBetweenIndices(int\* arr, int size, long long& product) {

product = 1;

for (int i = 2; i <= 7; i++) {

product \*= arr[i];

}

// Возвращаем ссылку на произведение

return product;

}

int main() {

int size = 15, rmn = 1, rmx = 200;

int\* ptr = new int[size];

// Заполняем массив случайными числами и сразу выводим

srand((unsigned)time(NULL));

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

ptr[i] = rand() % (rmx - rmn) + rmn;

cout << ptr[i] << ' ';

}

cout << '\n';

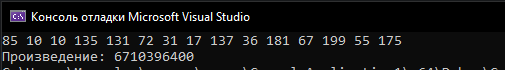
long long product = 1;

product = productBetweenIndices(ptr, size, product);

cout << "Произведение: " << product;

return 0;

}



#include <iostream>

#include <Windows.h>

using namespace std;

// Функция, возвращающая ссылку на среднее арифметическое

double& averageAndModifyMatrix(double\*\* arr, int n, int m, double& average) {

bool flag = false;

average = 0;

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

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

if (arr[j][i] >= 0) {

flag = true;

break;

}

}

if (!flag) {

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

average += arr[j][i];

}

break;

}

flag = false;

}

average = average / n;

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

for (int j = 0; j < m; j++) {

arr[i][j] = arr[i][j] - average;

}

}

return average;

}

int main() {

SetConsoleOutputCP(1251);

int n = 3, m = 4, rmn = -10, rmx = 10;

double\*\* matrix = new double\* [n];

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

matrix[i] = new double[m];

srand((unsigned)time(NULL));

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

for (int j = 0; j < m; j++) {

matrix[i][j] = rand() % (rmx - rmn) + rmn;

cout << matrix[i][j] << " ";

}

cout << '\n';

}

double average = 0;

average = averageAndModifyMatrix(matrix, n, m, average);

if (average == 0) {

cout << "Нету" << endl;

}

else {

cout << "Среднее арифметическое: " << average << endl;

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

for (int j = 0; j < m; j++) {

cout << matrix[i][j] << ' ';

}

cout << '\n';

}

}

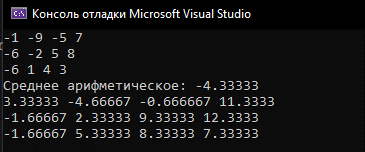
for (int i = 0; i < n; ++i)

delete[] matrix[i];

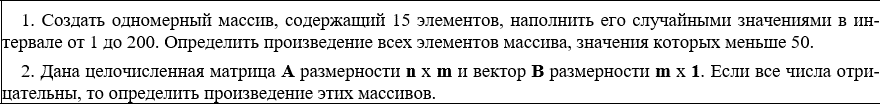
delete[] matrix;

return 0;

}



Вариант 9



#include <iostream>

using namespace std;

// Функция, возвращающая ссылку на произведение элементов меньше 50

long long& productBetweenIndices(int\* arr, int size, long long& product) {

product = 1;

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

if (arr[i] < 50) {

product \*= arr[i];

}

}

// Возвращаем ссылку на произведение

return product;

}

int main() {

int size = 15, rmn = 1, rmx = 200;

int\* ptr = new int[size];

// Заполняем массив случайными числами и сразу выводим

srand((unsigned)time(NULL));

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

ptr[i] = rand() % (rmx - rmn) + rmn;

cout << ptr[i] << ' ';

}

cout << '\n';

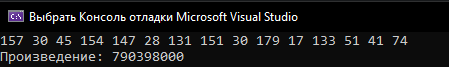
long long product = 1;

product = productBetweenIndices(ptr, size, product);

cout << "Произведение: " << product;

return 0;

}



#include <iostream>

#include <ctime>

using namespace std;

#include <Windows.h>

// Функция для проверки, все ли числа в матрице отрицательны

// Тут можно было бы извратиться и вернуть ссылку, но это нецелесообразно, учитывая что надо вернуть булевое значение

bool allNegative(int\*\* matrix, int &n, int &m) {

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

for (int j = 0; j < m; j++) {

if (matrix[i][j] >= 0) {

return false; // Если найдено неотрицательное число

}

}

}

return true; // Если все числа отрицательны

}

// Функция для вычисления произведения матрицы на вектор

int\* multiplyMatrixVector(int\*\* matrix, int\* vector, int &n, int &m) {

int\* result = new int[n];

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

result[i] = 0;

for (int j = 0; j < m; j++) {

result[i] += matrix[i][j] \* vector[j];

}

}

return result;

}

int main() {

SetConsoleOutputCP(1251);

int size = 15, rmn = -30, rmx = 5;

int n = 2;

int m = 3;

//создаем матрицу n на m

int\*\* matrix = new int\* [n];

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

matrix[i] = new int[m];

srand((unsigned)time(NULL));

// Заполненяем матрицу случайными числами и сразу выводим

cout << "Матрица:" << endl;

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

for (int j = 0; j < m; j++) {

matrix[i][j] = rand() % (rmx - rmn) + rmn;

cout << matrix[i][j] << ' ';

}

cout << endl;

}

// Заполняем вектор случайными числами и сразу выводим его

int\* vector = new int[m];

cout << "\nВектор:" << endl;

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

vector[i] = rand() % (rmx - rmn) + rmn;

cout << vector[i] << endl;

}

// Проверка, все ли числа в матрице отрицательны

if (allNegative(matrix, n, m)) {

// Вычисление произведения матрицы на вектор

int\* result = multiplyMatrixVector(matrix, vector, n, m);

// Вывод результата

cout << "\n Результат:" << endl;

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

cout << result[i] << endl;

}

}

else {

cout << "\nне все числа отрицательны" << endl;

}

return 0;

}

