Отчёт по первой

практической работе.

Группа: ИС-33;

ФИО Студента: Юрасов Никита Дмитриевич;

ФИО преподавателя: Холодов Дмитрий Сергеевич;

Программа 1:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main(){  
double x = 14.26;  
double y = -1.22;  
double z = 3.5\*pow(10, -2);  
  
double c = 2.\*cos(x-2./3.);  
double b = 1./2.+pow(sin(y), 2);  
double a = 1.+(pow(z, 2)/(3.-pow(z, 2)/5.));  
cout<< c/b\*a;  
}

Программа 2:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main(){  
double x = -4.5;  
double y = 0.000075;  
double z = -84.5;  
double c = (cbrt(9.+pow((x-y), 2))) / (pow(x, 2) + pow(y, 2) + 2.);  
double b = (exp(abs(x-y))) \* pow(tan(z), 3);  
cout<< c-b  
}

Программа 3:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main(){  
double x = 0.0374;  
double y = -0.825;  
double z = 16;  
double c = ((1+pow(sin(x+y), 2)) / (abs(x-((2\*y)/(1+pow(x, 2)\*pow(y, 2)))))) \* pow(x, abs(y));  
double b = pow(cos(atan(1/z)), 2);  
cout<<c+b  
}

Программа 4:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main(){  
double x = 0.4\*pow(10, 4);  
double y = -0.875;  
double z = -0.475\*pow(10, -3);  
double c = pow(abs(cos(x)-cos(y)), (1.+(2\*pow(sin(y), 2))));  
double b = 1.+z+(pow(z, 2)/2.)+(pow(z, 3)/3.)+(pow(z, 4)/4.);  
cout<< c\*b;  
}

Программа 5:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main(){  
double x = -15.246;  
double y = 4.642\*pow(10, -2);  
double z = 21;  
double c = log(pow(y, (-sqrt(abs(x)))));  
double b = x-y/2;  
double a = pow(sin(atan(z)), 2);  
 cout << c\*b+a;  
}

Практическая работа №2:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main()  
{  
 double x;  
 double y;  
 double s;  
 cout << "Ввeдите число" << endl;  
 cin >> x;  
 cout << "Ввeдите второе число" << endl;  
 cin >> y;  
 if(x > y){  
 s = cbrt(abs(x-y))+tan(x);  
 cout << s;  
 }  
 else if(x < y){  
 s = pow((y-x), 3)+ cos(x);  
 cout << s;  
 }  
 else{  
 s = (pow((y+x), 2)) + pow(x, 3);  
 cout << s;  
 }  
}

Практическая работа №3:

#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
int main()  
{  
 double x;  
 double y;  
 cout << "Ввeдите число" << endl;  
 cin >> x;  
 for(int n = 1; n<=20; n++){  
 y+=((pow(n, 2)+1)/n)\*(pow((x/2), n));  
 }  
 cout << y <<endl;  
}

Основные практические работы

Практическая работа №1

Задание 1

Дан текст. Сколько раз в нём встречается символ @.

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

char str[] = "@egnognei@egbeki@";

int counting = 0;

for(int i = 0; i < strlen(str); i++){

if(str[i] == '@'){

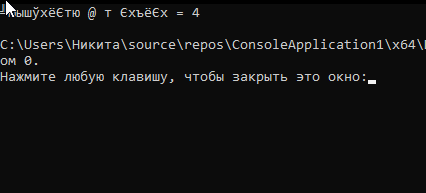
counting++;

}

}

cout << "Количество @ в тексте = " << counting << endl;

}



Задача 2

Даны 3 слова – ваши Имя, Отчество, Фамилия в 3-х разных переменных. Образовать новую символьную переменную, хранящую только ваши инициалы (через точку и пробел). (использовать склейку +).

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

string name = "Nikita";

string surname = "Yurasov";

string patronymic = "Dmitrievich";

string initials = initials + name[0];

initials = initials + ". ";

initials = initials + surname[0];

initials = initials + ". ";

initials = initials + patronymic[0];

initials = initials + ".";

cout << "Инициалы: " << initials;

}

Задание 3

Имеется некоторые последовательность символов. Образовать новую последовательность, включив в неё символы исходной, кроме запятых.

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

char str[] = "evbuevbqueb,qekvu,evybq,qeu,eg,,,,,gqg";

string str1;

for(int i = 0; i<strlen(str); i++){

if(str[i] != ','){

str1 = str1+str[i];

}

}

cout << str1;

}

Задача 4

Задан текст, содержащий пару фигурных скобок. Создать новый текст, включив в него текст заключённый в фигурные скобки.

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

char str[] = "Образовать новую символьную переменную, хранящую только ваши инициалы (через точку и пробел)";

string str2;

for(int i = 0; i<strlen(str); i++){

if(str[i] == '('){

int j = i+1;

while(str[j] != ')'){

str2 = str2 + str[j];

j++;

}

}

}

cout << str2;

}



Практическая работа №2

Задание 1

Задан список из десяти городов (массив [.] string). Поменять

местами названия двух городов, названия которых

оканчиваются сочетанием букв «ск» или «sk».

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

//1

int main() {

vector<string> cities = {"Moscow", "New York", "Warsaw", "Minsk", "Kyiv", "Omsk", "Tomsk", "Kiev", "Toronto", "Bratislava"};

auto it1 = find\_if(cities.begin(), cities.end(), [](const string& city) {

return (city.size() >= 2 && (city.substr(city.size() - 2) == "sk" || city.substr(city.size() - 2) == "ск"));

});

auto it2 = find\_if(it1 + 1, cities.end(), [](const string& city) {

return (city.size() >= 2 && (city.substr(city.size() - 2) == "sk" || city.substr(city.size() - 2) == "ск"));

});

if (it1 != cities.end() && it2 != cities.end()) {

swap(\*it1, \*it2);

cout << "Измененный список городов:\n";

for (const auto& city : cities) {

cout << city << endl;

}

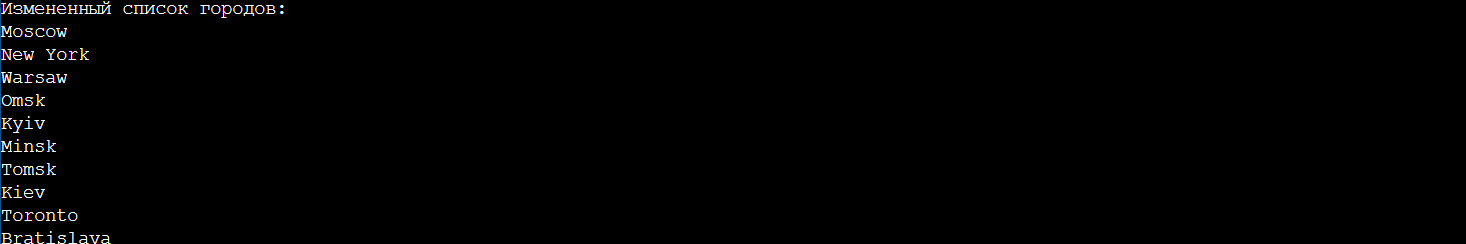
} else {

cout << "Не найдено двух городов с окончанием \"ск\" или \"sk\".\n";

}

return 0;

}



Задание 2

Задан список из 20 названий горных вершин (массив

[.] string). Найти количество букв в самом коротком названии.

//2

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

vector<string> mountainPeaks = {"Everest", "K2", "Kangchenjunga", "Lhotse", "Makalu", "Cho Oyu", "Dhaulagiri", "Manaslu", "Nanga Parbat", "Annapurna", "Gasherbrum I", "Broad Peak", "Gasherbrum II", "Shishapangma", "Mount McKinley", "Mount Kilimanjaro", "Mount Elbrus", "Mount Vinson", "Puncak Jaya", "Aconcagua"};

auto shortestPeak = min\_element(mountainPeaks.begin(), mountainPeaks.end(), [](const string& a, const string& b) {

return a.length() < b.length();

});

if (shortestPeak != mountainPeaks.end()) {

cout << "Количество букв в самом коротком названии: " << shortestPeak->length() << endl;

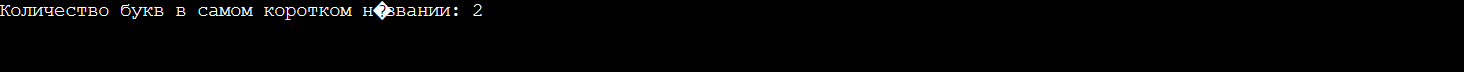
} else {

cout << "Список горных вершин пуст." << endl;

}

return 0;

}



Задание 3

Задан список из 20 названий горных вершин (массив

[.] string). Найти название с максимальным количеством

букв <k>.

//3

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

vector<string> mountainPeaks = {"Everest", "K2", "Kkangchenjunga", "Lhotse", "Makalu", "Cho Oyu", "Dhaulagiri", "Manaslu", "Nanga Parbat", "Annapurna", "Gasherbrum I", "Broad Peak", "Gasherbrum II", "Shishapangma", "Mount McKinley", "Mount Kilimanjaro", "Mount Elbrus", "Mount Vinson", "Puncak Jaya", "Aconcagua"};

auto maxKLengthPeak = max\_element(mountainPeaks.begin(), mountainPeaks.end(), [](const string& a, const string& b) {

int countA = count(a.begin(), a.end(), 'k') + count(a.begin(), a.end(), 'K');

int countB = count(b.begin(), b.end(), 'k') + count(b.begin(), b.end(), 'K');

return countA < countB;

});

if (maxKLengthPeak != mountainPeaks.end()) {

cout << "Название горной вершины с максимальным количеством букв \"k\": " << \*maxKLengthPeak << endl;

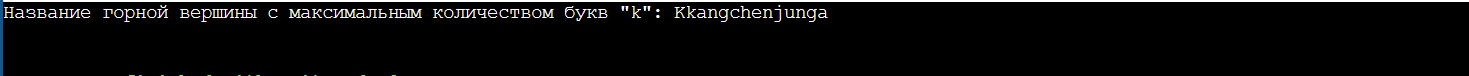
} else {

cout << "Не найдено названия горной вершины с буквой \"k\"." << endl;

}

return 0;

}



Задание 4

Дан список фамилий сотрудников (массив

[.] string). Переписать в другой список только те фамилии,

которые заканчиваются на ‘ова’ или ‘ova’. Затем упорядочить

по алфавиту второй список.

//4

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

vector<string> employeeNames = {"Ivanov", "Petrov", "Sidorova", "Smith", "Johnson", "Ivanova", "Kuznetsova", "Brown", "Ovchinnikova", "Novikova"};

vector<string> filteredNames;

for (const auto& name : employeeNames) {

if (name.size() >= 3) {

string suffix = name.substr(name.size() - 3);

if (suffix == "ova" || suffix == "ова") {

filteredNames.push\_back(name);

}

}

}

sort(filteredNames.begin(), filteredNames.end());

cout << "Фамилии сотрудников, заканчивающиеся на 'ova' или 'ова', в алфавитном порядке:\n";

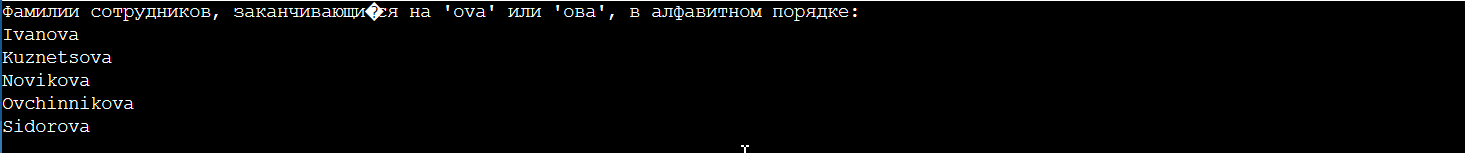
for (const auto& name : filteredNames) {

cout << name << endl;

}

return 0;

}



Практическая работа №3

Задание 1

Создать динамические массивы, используя указатели. Дан

двумерный массив a, размером (n⋅m). Заполнить одномерный

массив, найдя количество отрицательных элементов в каждом

столбце матрицы.

//1

#include <iostream>

using namespace std;

int main() {

int n, m;

cout<<"Введите количество строк >>> ";

cin >> n;// Количество строк

cout<<"Введите количество столбцов >>> ";

cin >> m;// Количество столбцов

// Создаем динамический двумерный массив

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

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

A[i] = new int[m];

}

// Заполняем матрицу случайными значениями

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

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

A[i][j] = rand() % 21 - 10; // Случайные значения от -10 до 10

}

}

// Выводим матрицу для наглядности

cout << "Исходная матрица A:\n";

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

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

cout << A[i][j] << "\t";

}

cout << endl;

}

// Создаем динамический массив для хранения количества отрицательных элементов в каждом столбце

int\* negativeCount = new int[m];

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

negativeCount[j] = 0; // Инициализируем счетчик для каждого столбца

}

// Находим количество отрицательных элементов в каждом столбце

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

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

if (A[i][j] < 0) {

negativeCount[j]++;

}

}

}

// Выводим результат

cout << "\nКоличество отрицательных элементов в каждом столбце:\n";

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

cout << "Столбец " << j + 1 << ": " << negativeCount[j] << " отрицательных элементов\n";

}

// Освобождаем выделенную память

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

delete[] A[i];

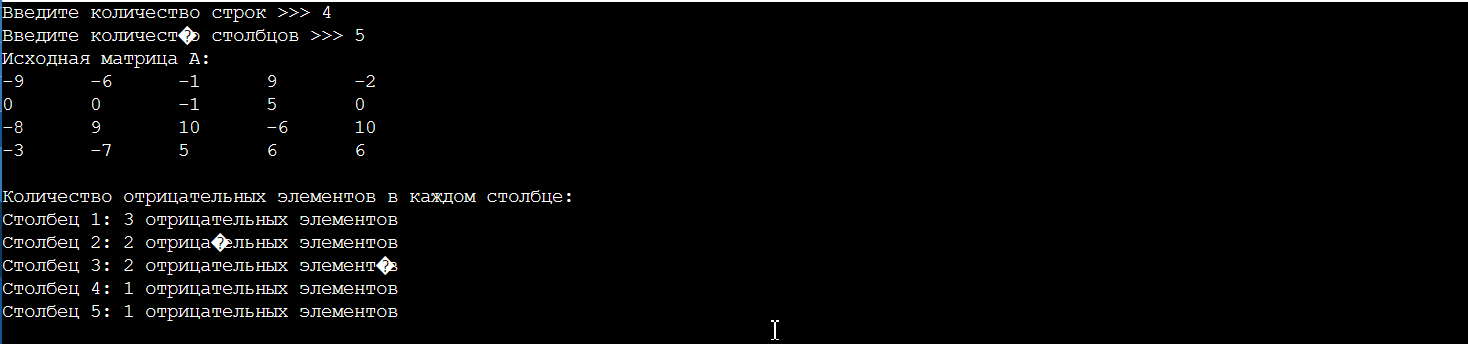
}

delete[] A;

delete[] negativeCount;

return 0;

}



Задание 2

Создать динамические массивы, используя указатели. Дан

двумерный массив a, размером (n⋅n) (или квадратная

матрица a). Найти среднее арифметическое

положительных элементов, параллели главной диагонали

расположенной выше над диагональю.

//2

#include <iostream>

using namespace std;

int main() {

srand(time(0));

int n;

cout<<"Ведите размер матрицы >>> ";

cin>> n; // Размер квадратной матрицы

// Создаем динамический двумерный массив (квадратную матрицу)

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

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

A[i] = new int[n];

}

// Заполняем матрицу случайными значениями

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

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

A[i][j] = rand() % 21 - 10; // Случайные значения от -10 до 10

}

}

// Выводим исходную матрицу для наглядности

cout << "Исходная матрица A:\n";

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

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

cout << A[i][j] << "\t";

}

cout << endl;

}

// Вычисляем среднее арифметическое положительных элементов выше главной диагонали

int count = 0;

int sum = 0;

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

for (int j = i + 1; j < n; ++j) {

if (A[i][j] > 0 && j == i+1) {

sum += A[i][j];

count++;

}

}

}

double average = 0.0;

if (count != 0) {

average = static\_cast<double>(sum) / count;

}

// Выводим результат

cout << "\nСреднее арифметическое положительных элементов выше главной диагонали: " << average << endl;

// Освобождаем выделенную память

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

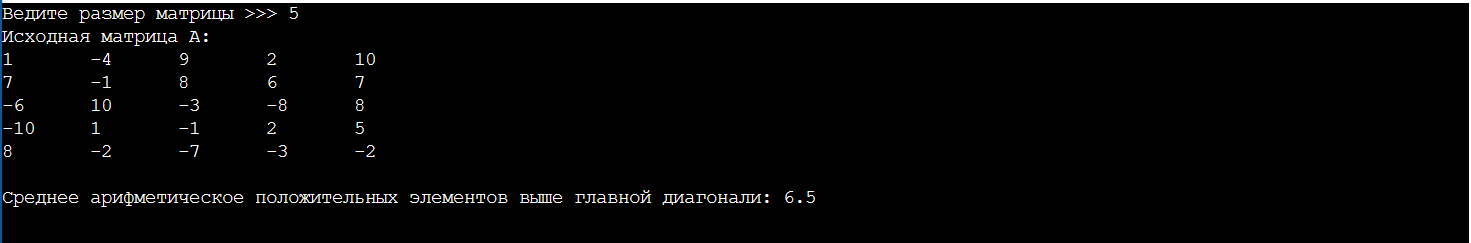
delete[] A[i];

}

delete[] A;

return 0;

}



Задание 3

Создать динамические массивы, используя указатели. Дан

двумерный массив a, размером (n⋅m). Переставить третий и

пятый столбец. Затем упорядочить по убыванию первый

столбец матрицы.

//3

#include <iostream>

#include <algorithm>

using namespace std;

void SortMatrix(int\*\* Matrix, int string, int column)

{

int tmp;

for (int col = 0; col < column; col++) // Цикл по столбцам матрицы

{

for (int row = 0; row < string - 1; row++) // Сортировка пузырьком

{

for (int i = 0; i < string - row - 1; i++)

{

if (Matrix[i + 1][col] > Matrix[i][col] && col == 0)

{

tmp = Matrix[i][col];

Matrix[i][col] = Matrix[i + 1][col];

Matrix[i + 1][col] = tmp;

}

}

}

}

}

int main() {

srand(time(0));

int n = 4; // Количество строк

int m = 5; // Количество столбцов

// Создаем динамический двумерный массив

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

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

A[i] = new int[m];

}

// Заполняем матрицу значениями от 1 до n\*m для наглядности

int value = 1;

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

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

A[i][j] = rand() % (30 - 10 + 1) + 10;

}

}

// Выводим исходную матрицу для наглядности

cout << "Исходная матрица A:\n";

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

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

std::cout << A[i][j] << "\t";

}

cout << endl;

}

// Переставляем третий и пятый столбцы

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

swap(A[i][2], A[i][4]);

}

// Упорядочиваем по убыванию первый столбец матрицы

SortMatrix(A, n, m);

// Выводим результат

cout << "\nМатрица после перестановки столбцов и упорядочивания первого столбца по убыванию:\n";

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

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

cout << A[i][j] << "\t";

}

cout << endl;

}

// Освобождаем выделенную память

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

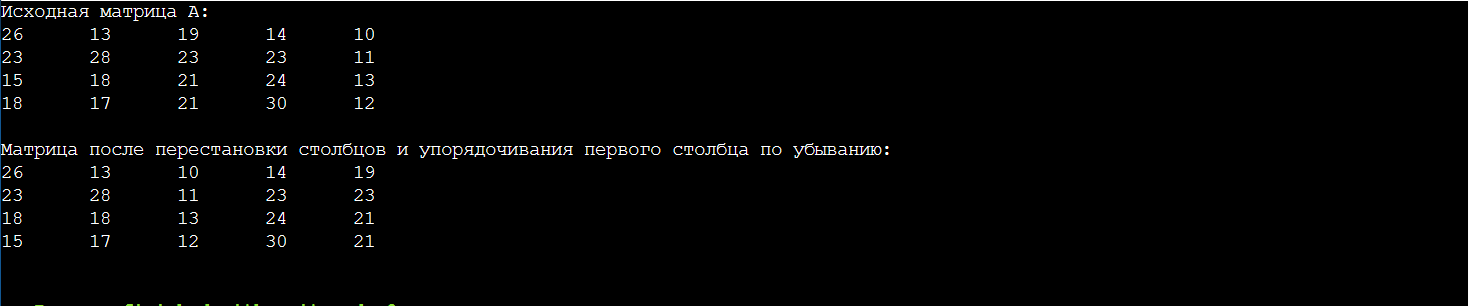
delete[] A[i];

}

delete[] A;

return 0;

}



Задание 4

Создать динамические массивы, используя указатели. Дан

двумерный массив a, размером (n⋅n). Упорядочить по

возрастанию побочную диагональ.

//4

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

srand(time(0));

int n = 4; // Размер квадратной матрицы

// Создаем динамический двумерный массив

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

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

A[i] = new int[n];

}

// Заполняем матрицу случайными значениями

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

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

A[i][j] = rand() % 21 - 10; // Случайные значения от -10 до 10

}

}

// Выводим исходную матрицу для наглядности

cout << "Исходная матрица A:\n";

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

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

cout << A[i][j] << "\t";

}

cout << endl;

}

// Упорядочиваем по возрастанию побочную диагональ

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

int index\_max = i;

for(int j = (i+1); j < n; j++){

if(A[j][n - j - 1]>A[index\_max][n - index\_max - 1]){

index\_max = j;

}

}

if(index\_max != i){

int swap = A[i][n - i - 1];

A[i][n - i - 1] = A[index\_max][n - index\_max - 1];

A[index\_max][n - index\_max - 1] = swap;

}

}

// Выводим результат

cout << "\nМатрица после упорядочивания по возрастанию побочной диагонали:\n";

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

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

cout << A[i][j] << "\t";

}

cout << endl;

}

// Освобождаем выделенную память

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

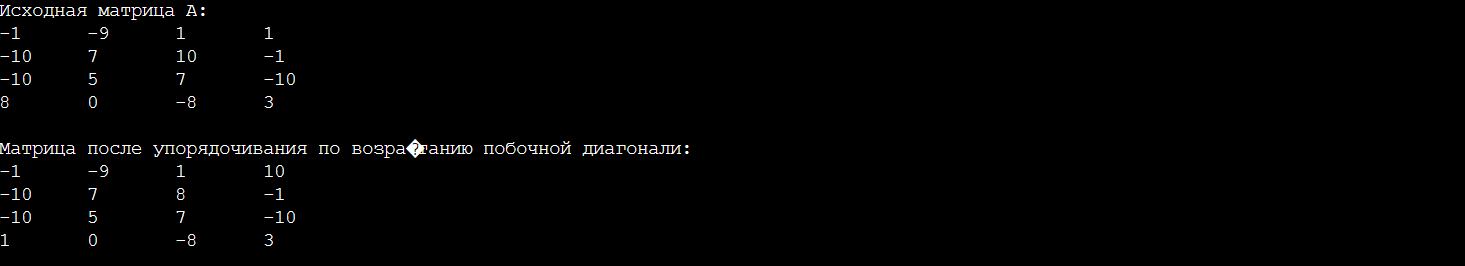
delete[] A[i];

}

delete[] A;

return 0;

}



Практическая работа №4

Задание 1

Ввести значение 2-х вещественных переменных a и b.

Направить два указателя на эти переменные. С помощью

указателя увеличить значение переменной a в 3 раза,

а b уменьшить в 3 раза.

//1

#include <iostream>

using namespace std;

int main() {

// Ввод значений переменных a и b

double a, b;

cout << "Введите значение переменной a: ";

cin >> a;

cout << "Введите значение переменной b: ";

cin >> b;

// Создаем указатели на переменные a и b

double \*ptrA = &a;

double \*ptrB = &b;

// Увеличиваем значение переменной a в 3 раза с помощью указателя

\*ptrA \*= 3;

// Уменьшаем значение переменной b в 3 раза с помощью указателя

\*ptrB /= 3;

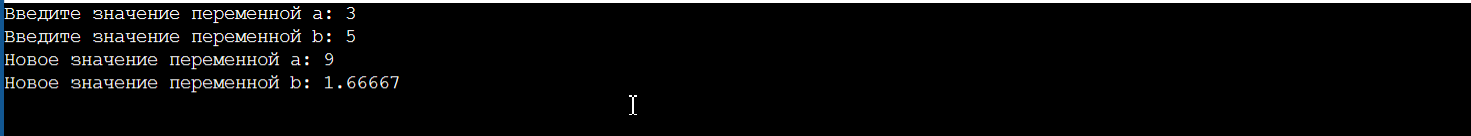
// Выводим результат

cout << "Новое значение переменной a: " << a << endl;

cout << "Новое значение переменной b: " << b << endl;

return 0;

}



Задание 2

Описать 1 указатель на целый тип. Выделить для него

динамическую память. Ввести значения в выделенную память

с клавиатуры. Затем увеличить ее на 2

//2

#include <iostream>

using namespace std;

int main() {

// Описываем указатель на целый тип

int\* dynamicPtr;

// Выделяем динамическую память для указателя

dynamicPtr = new int;

// Вводим значение в выделенную память с клавиатуры

cout << "Введите значение: ";

cin >> \*dynamicPtr;

// Увеличиваем значение в выделенной памяти на 2

\*dynamicPtr += 2;

// Выводим результат

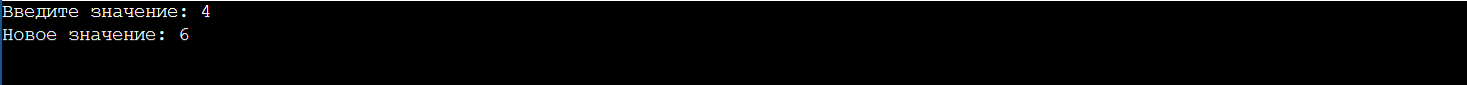
cout << "Новое значение: " << \*dynamicPtr << endl;

// Освобождаем выделенную память

delete dynamicPtr;

return 0;

}



Задание 3

Создать динамические массивы, используя указатели. В каком

из двух данных массивов p(n) и q(n) больше элементов,

равных 1?

//3

#include <iostream>

#include <cstdlib>

using namespace std;

int countOnes(int\* arr, int size) {

int count = 0;

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

if (arr[i] == 1) {

count++;

}

}

return count;

}

int main() {

srand(time(0));

int n;

cout << "Введите размер массивов: ";

cin >> n;

// Создаем два динамических массива

int\* p = new int[n];

int\* q = new int[n];

// Заполняем массивы случайными значениями (0 или 1)

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

p[i] = rand() % 2;

q[i] = rand() % 2;

}

cout<<endl;

cout << "Массив p"<<endl;

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

cout<<p[i]<<" ";

}

cout<<endl;

cout<< "Массив q"<<endl;

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

cout<<q[i]<<" ";

}

cout<<endl;

// Подсчитываем количество элементов, равных 1, в каждом массиве

int countP = countOnes(p, n);

int countQ = countOnes(q, n);

// Сравниваем результаты и выводим информацию

if (countP > countQ) {

cout << "Массив p содержит больше элементов, равных 1." << endl;

} else if (countQ > countP) {

cout << "Массив q содержит больше элементов, равных 1." << endl;

} else {

cout << "Количество элементов, равных 1, в массивах p и q, одинаково." << endl;

}

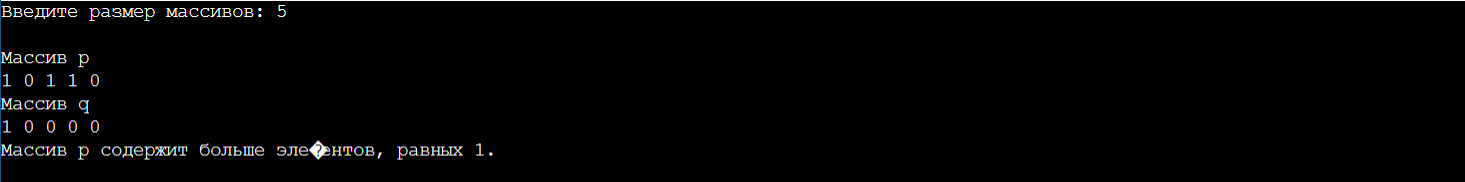
// Освобождаем выделенную память

delete[] p;

delete[] q;

return 0;

}



Задание 4

Создать динамические массивы, используя указатели. Дан массив x(n)

Переписать в массив y(n) элементы массива x, большие 3 (со сжатием,

без пустых элементов внутри). Затем упорядочить методом «выбора и

перестановки» по возрастанию новый массив.

//4

#include <iostream>

using namespace std;

void compressArray(int\* arr, int& size) {

int newSize = 0;

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

if (arr[i] > 0) {

arr[newSize++] = arr[i];

}

}

size = newSize;

}

void selectionSort(int\* arr, int size) {

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

int minIndex = i;

for (int j = i + 1; j < size; ++j) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

swap(arr[i], arr[minIndex]);

}

}

}

int main() {

int n;

cout << "Введите размер массива X: ";

cin >> n;

// Создаем динамический массив X

int\* X = new int[n];

// Вводим значения в массив X

cout << "Введите элементы массива X:\n";

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

cin >> X[i];

}

// Создаем динамический массив Y

int\* Y = new int[n];

// Переписываем в Y элементы X, большие 3, и сжимаем массив Y

int newYSize = 0;

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

if (X[i] > 3) {

Y[newYSize++] = X[i];

}

}

// Сжимаем массив Y

compressArray(Y, newYSize);

// Выводим массив Y до сортировки

cout << "\nМассив Y до сортировки:\n";

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

cout << Y[i] << " ";

}

cout << "\n";

// Сортируем массив Y методом "выбора и перестановки"

selectionSort(Y, newYSize);

// Выводим массив Y после сортировки

cout << "\nМассив Y после сортировки:\n";

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

cout << Y[i] << " ";

}

cout << "\n";

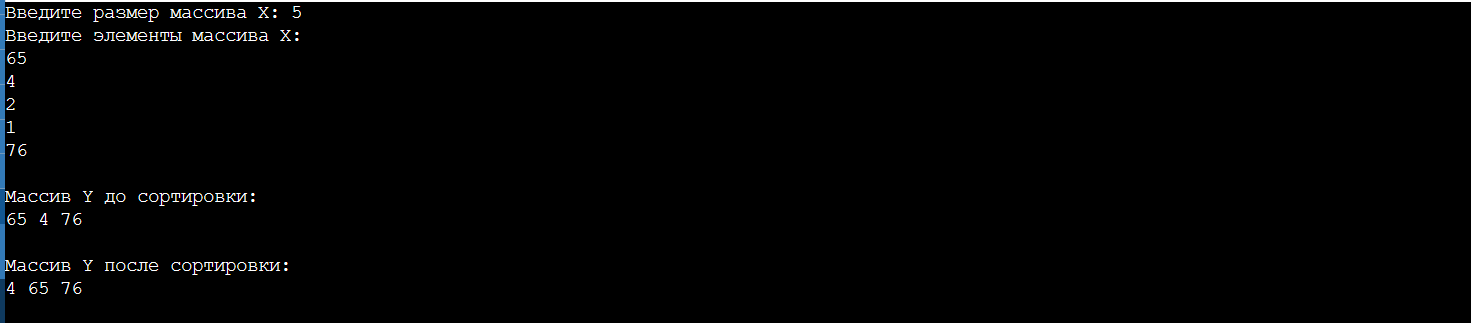
// Освобождаем выделенную память

delete[] X;

delete[] Y;

return 0;

}



Практическая работа №5

Задание 1

Составить программу, выводящую на экран ведомость начисленной заработной платы (Ф.И.О., должность, год и дата рождения, заработная плата). Вывести дату рождения сотрудника с самой маленькой зарплатой.

//1

#include <iostream>

#include <string>

using namespace std;

struct Employee {

string name;

string position;

int birth\_year;

int birth\_month;

int birth\_day;

double salary;

};

int main() {

const int NUM\_EMPLOYEES = 3;

Employee employees[NUM\_EMPLOYEES] = {

{"John Smith", "Manager", 1980, 1, 1, 5000},

{"Jane Doe", "Salesperson", 1985, 2, 3, 3000},

{"Bob Johnson", "Programmer", 1990, 3, 5, 4000}

};

cout << "Name\tPosition\tBirth Date\tSalary\n";

for (int i = 0; i < NUM\_EMPLOYEES; i++) {

cout << employees[i].name << "\t" << employees[i].position << "\t" << employees[i].birth\_year << "/"

<< employees[i].birth\_month << "/" << employees[i].birth\_day << "\t" << employees[i].salary << "\n";

}

double min\_salary = employees[0].salary;

int min\_salary\_index = 0;

for (int i = 1; i < NUM\_EMPLOYEES; i++) {

if (employees[i].salary < min\_salary) {

min\_salary = employees[i].salary;

min\_salary\_index = i;

}

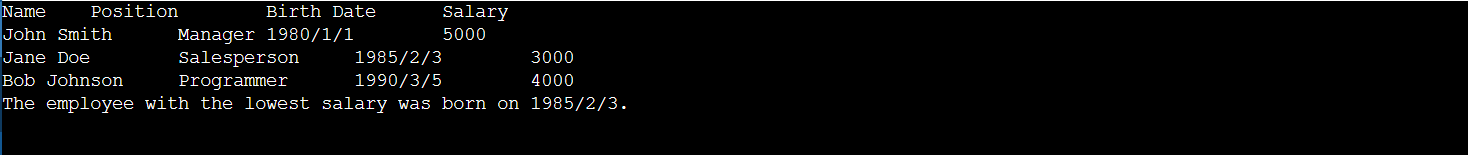
}

cout << "The employee with the lowest salary was born on " << employees[min\_salary\_index].birth\_year << "/"

<< employees[min\_salary\_index].birth\_month << "/" << employees[min\_salary\_index].birth\_day << ".\n";

return 0;

}



Задание 2

Определить комбинированный (структурный) тип для

представления информации о кости домино, состоящей из

левой половинки и правой половинки. Поля «левая» и

«правая» половинки хранят информацию о количестве

точек на половинках. Описать массив из 28 элементов

(кости домино). Заполнить массив случайными числами или

ввести с клавиатуры. Определить, правильно ли

выставлены кости в данном массиве (равна ли правая

цифра очередной кости левой цифре следующей кости).

//2

#include <iostream>

#include <time.h>

using namespace std;

struct Domino {

int left;

int right;

};

void fillArray(Domino arr[], int size) {

srand(time(0));

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

arr[i].left = rand() % 7;

arr[i].right = rand() % 7;

}

}

void checkArray(Domino arr[], int size) {

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

if (arr[i].right != arr[i + 1].left) {

cout << "Kosts are not properly placed at index " << i << " and " << i + 1 << endl;

}

}

cout << "All kosts are properly placed" << endl;

}

int main() {

const int size = 28;

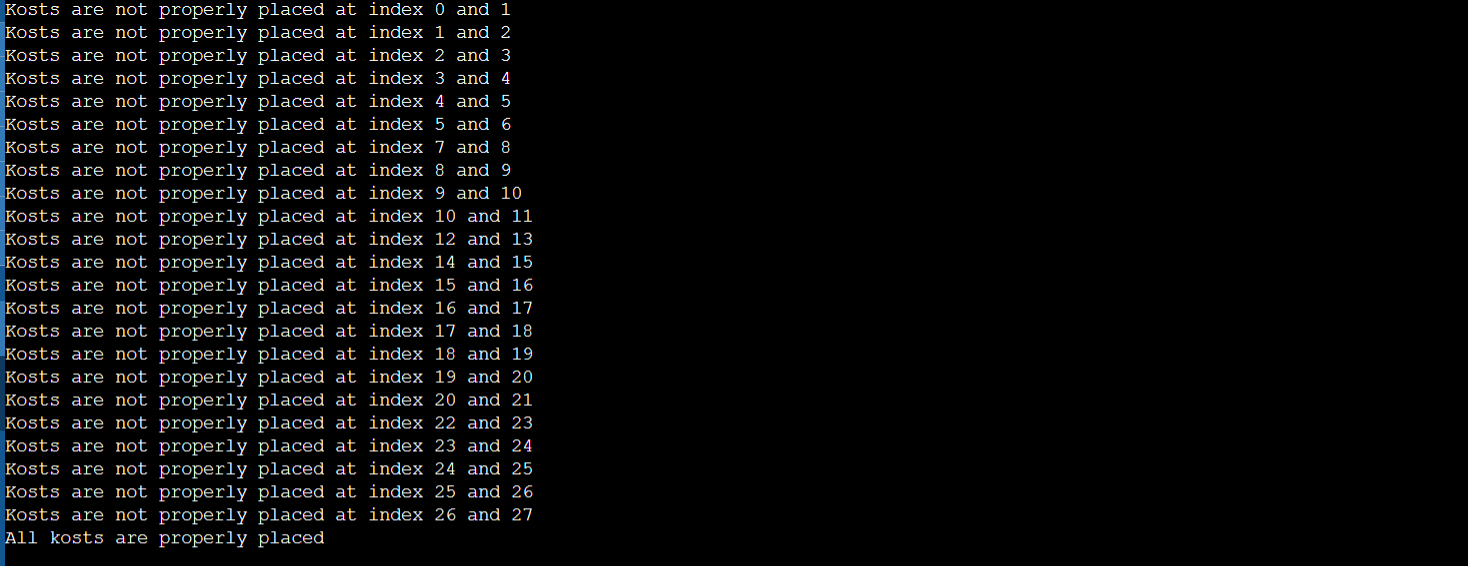
Domino dominoArray[size];

fillArray(dominoArray, size);

checkArray(dominoArray, size);

return 0;

}



Задание 3

Определить комбинированный (структурный) тип,

описывающий окружность и состоящий из двух полей:

«радиус» и «центр». Поле «центр» в свою очередь состоит

еще из двух полей: «координата Х» и «координата Y».

Ввести информацию по 10 окружностям. Переписать из

исходного массива в другой массив, информацию только о

тех окружностях, центр которых лежит ниже оси OX.

//3

#include <iostream>

using namespace std;

struct Point {

int x;

int y;

};

struct Circle {

int radius;

Point center;

};

int main() {

Circle circles[10];

// Ввод информации по 10 окружностям

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

cout << "Enter radius for circle " << i + 1 << ": ";

cin >> circles[i].radius;

cout << "Enter X coordinate for circle " << i + 1 << ": ";

cin >> circles[i].center.x;

cout << "Enter Y coordinate for circle " << i + 1 << ": ";

cin >> circles[i].center.y;

}

// Переписать информацию в другой массив о кругах, центр которых лежит ниже оси OX

Circle belowOX[10];

int countBelowOX = 0;

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

if (circles[i].center.y < 0) {

belowOX[countBelowOX] = circles[i];

countBelowOX++;

}

}

// Вывод информации о кругах, центр которых лежит ниже оси OX

cout << "Circles with centers below OX:" << endl;

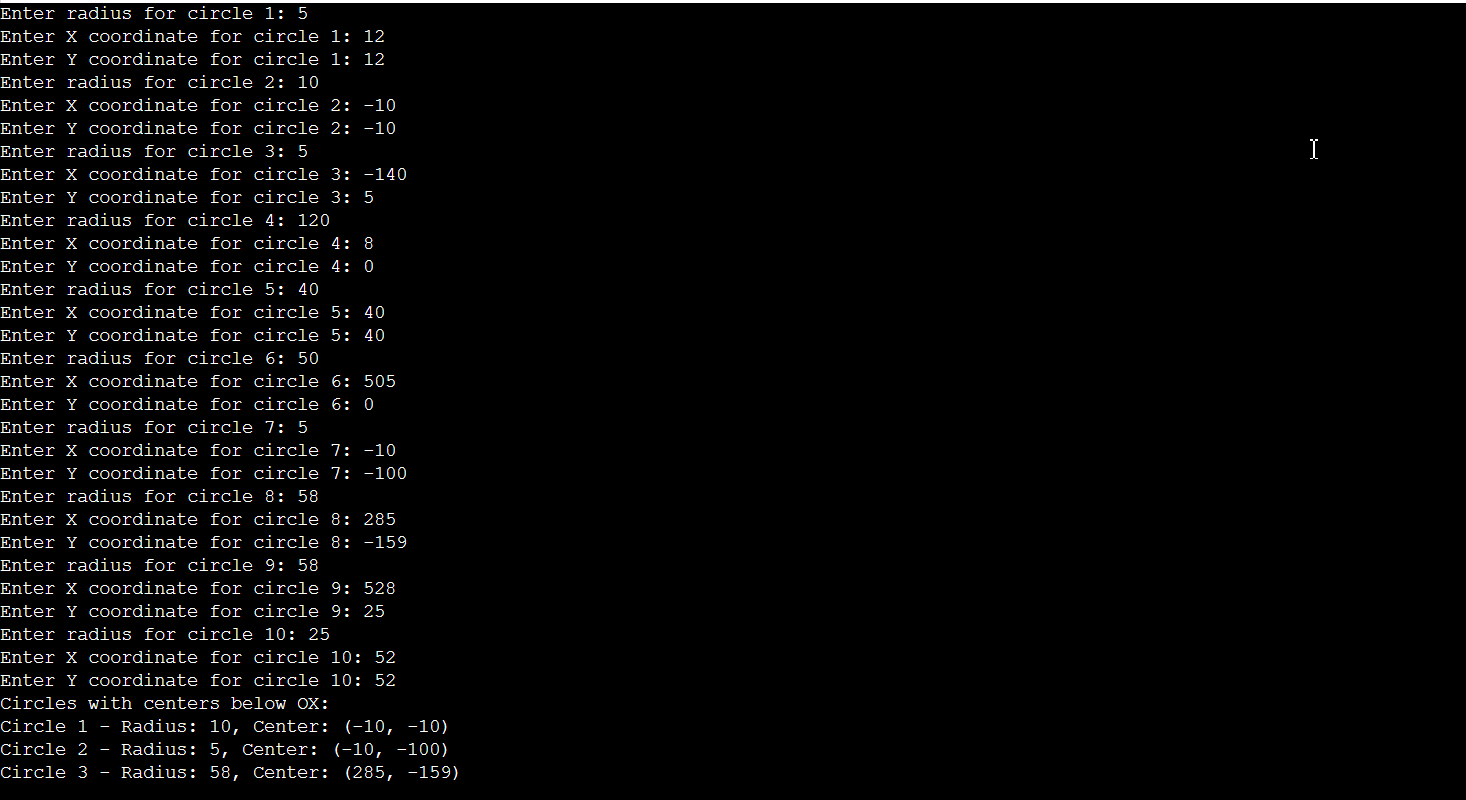
for (int i = 0; i < countBelowOX; i++) {

cout << "Circle " << i + 1 << " - Radius: " << belowOX[i].radius << ", Center: (" << belowOX[i].center.x << ", " << belowOX[i].center.y << ")" << endl;

}

return 0;

}



Задание 4

Определить структрый тип, описывающий расписание полетов

самолетов (пункт назначения, время отправления, время

прибытия, время полета, стоимость билета). Заполнить

структурный массив 10-ю записями. Переписать из исходного

массива в другой массив, информацию только о тех рейсах,

пункт назначения которых оканчивается сочетанием «град».

Затем новый массив отсортировать по времени полета.

//4

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

struct FlightSchedule {

string destination;

string departureTime;

string arrivalTime;

int flightDuration;

float ticketCost;

};

int main() {

FlightSchedule flights[10] = {

{"Петроград", "08:00", "12:00", 4, 300.50},

{"London", "09:00", "14:00", 5, 400.25},

{"Paris", "10:00", "15:00", 5, 350.75},

{"Tokyo", "11:00", "19:00", 8, 600.00},

{"Berlin", "12:00", "16:00", 4, 320.75},

{"Moscow", "13:00", "18:00", 5, 380.00},

{"Sydney", "14:00", "22:00", 8, 550.50},

{"Los Angeles", "15:00", "20:00", 5, 420.25},

{"Rome", "16:00", "20:00", 4, 330.75},

{"Madrid", "17:00", "21:00", 4, 340.00}

};

FlightSchedule destinationWithGrad[10];

int destinationWithGradCount = 0;

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

if (flights[i].destination.find("град") != string::npos) {

destinationWithGrad[destinationWithGradCount] = flights[i];

destinationWithGradCount++;

}

}

sort(destinationWithGrad, destinationWithGrad + destinationWithGradCount,

[](const FlightSchedule &a, const FlightSchedule &b) {

return a.flightDuration < b.flightDuration;

});

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

cout << "Destination: " << destinationWithGrad[i].destination << " | Departure time: " << destinationWithGrad[i].departureTime << " | Arrival time: " << destinationWithGrad[i].arrivalTime << " | Flight duration: " << destinationWithGrad[i].flightDuration << " hours | Ticket cost: $" << destinationWithGrad[i].ticketCost << endl;

}

return 0;

}



Задание 5

Определить комбинированный (структурный) тип, описывающий

окружность и состоящий из двух полей: «радиус» и «центр».

Поле «центр» в свою очередь состоит еще из двух полей:

«координата Х» и «координата Y». Ввести информацию

по 10 окружностям. Вывести радиус окружности, чей центр

самый удаленный от оси OX (оси абсцисс). Затем вывести

информацию, отсортированную по возрастанию радиуса

окружности.

//5

#include <iostream>

#include <cmath>

#include <vector>

#include <algorithm>

using namespace std;

// Структура для представления координат точки

struct Point {

double x, y;

};

// Структура для представления центра окружности

struct Center {

Point coordinates;

};

// Структура для представления окружности

struct Circle {

double radius;

Center center;

};

// Функция для ввода информации о окружности

Circle inputCircle() {

Circle circle;

cout << "Введите радиус окружности: ";

cin >> circle.radius;

cout << "Введите координаты центра окружности (X Y): ";

cin >> circle.center.coordinates.x >> circle.center.coordinates.y;

return circle;

}

// Функция для вычисления расстояния от центра окружности до оси OX

double distanceToOX(const Circle& circle) {

return abs(circle.center.coordinates.y);

}

// Функция для сравнения окружностей по радиусу

bool compareByRadius(const Circle& circle1, const Circle& circle2) {

return circle1.radius < circle2.radius;

}

int main() {

const int numCircles = 10;

vector<Circle> circles;

// Ввод информации о 10 окружностях

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

cout << "Окружность " << i + 1 << ":" << endl;

circles.push\_back(inputCircle());

}

// Находим окружность с самым удаленным центром от оси OX

auto maxDistanceIt = max\_element(circles.begin(), circles.end(),

[](const Circle& circle1, const Circle& circle2) {

return distanceToOX(circle1) < distanceToOX(circle2);

});

cout << "Окружность с самым удаленным центром от оси OX:" << endl;

cout << "Радиус: " << maxDistanceIt->radius << endl;

// Сортируем окружности по возрастанию радиуса

sort(circles.begin(), circles.end(), compareByRadius);

// Выводим информацию, отсортированную по возрастанию радиуса

cout << "\nИнформация, отсортированная по возрастанию радиуса:" << endl;

for (const Circle& circle : circles) {

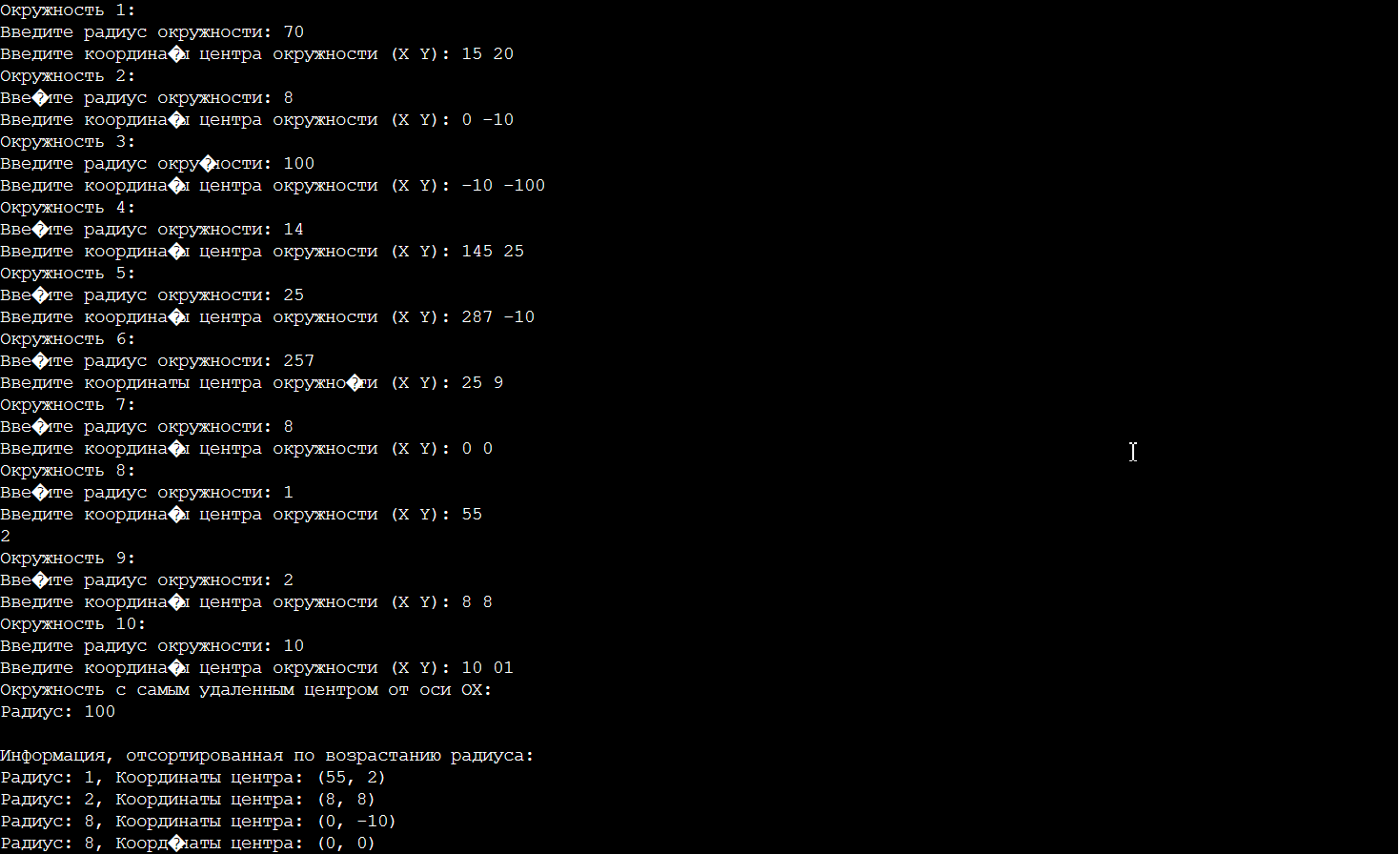
cout << "Радиус: " << circle.radius << ", Координаты центра: (" << circle.center.coordinates.x

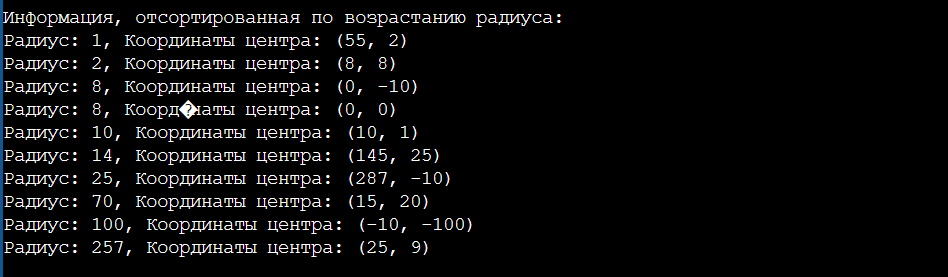
<< ", " << circle.center.coordinates.y << ")" << endl;

}

return 0;

}





Практическая работа №6

Задание 1

Пользователь вводит текст. Вывести исходный текст, заменив в нем

слово <дублирование> на <копирование>. Вычислить

количество всех слов.

//1

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

int main() {

std::string inputText;

// Ввод текста от пользователя

cout << "Введите текст: ";

getline(cin, inputText);

// Замена слова "дублирование" на "копирование"

size\_t pos = inputText.find("дублирование");

while (pos != string::npos) {

inputText.replace(pos, 21, "копирование"); // Длина слова "дублирование" равна 13

pos = inputText.find("дублирование", pos);

}

// Вывод исходного текста с заменой

cout << "Исходный текст с заменой: " << inputText << endl;

// Подсчет количества слов

istringstream iss(inputText);

int wordCount = 0;

while (iss >> inputText) {

wordCount++;

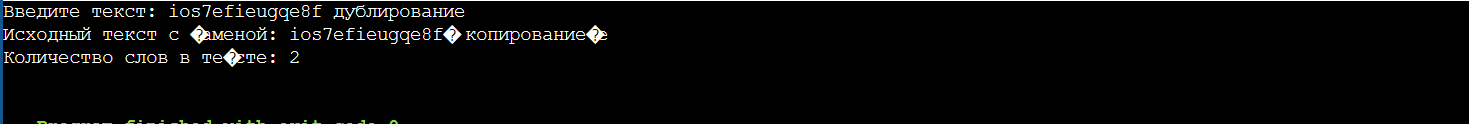
}

// Вывод количества слов

cout << "Количество слов в тексте: " << wordCount << endl;

return 0;

}



Задание 2

Исходный текст набран с ошибками: иногда отсутствуют

пробелы после точек. Вставить 1 пробел после каждой точки,

если он отсутствует перед следующим предложением. а

также вычислить количество предложений. А также удалить

квадратные скобки.

//2

#include <iostream>

#include <string>

using namespace std;

int main() {

string inputText;

// Ввод текста от пользователя

cout << "Введите текст: ";

getline(std::cin, inputText);

// Вставка пробелов после точек перед следующим предложением

size\_t pos = 0;

while ((pos = inputText.find('.', pos)) != std::string::npos) {

if (pos + 1 < inputText.length() && !isspace(inputText[pos + 1])) {

inputText.insert(pos + 1, " ");

}

pos += 2; // Переходим к следующему символу после вставленного пробела

}

// Удаление квадратных скобок

size\_t bracketPos;

while ((bracketPos = inputText.find('[')) != std::string::npos) {

size\_t endBracketPos = inputText.find(']', bracketPos);

if (endBracketPos != std::string::npos) {

inputText.erase(bracketPos, endBracketPos - bracketPos + 1);

}

}

// Вывод исходного текста с вставленными пробелами и удаленными скобками

cout << "Исходный текст: " << inputText << endl;

// Вычисление количества предложений

size\_t sentenceCount = 0;

pos = 0;

while ((pos = inputText.find('.', pos)) != string::npos) {

sentenceCount++;

pos++;

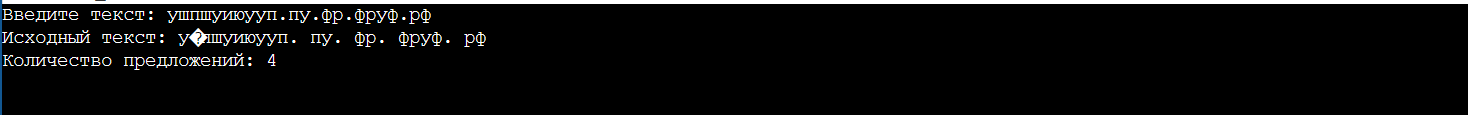
}

// Вывод количества предложений

cout << "Количество предложений: " << sentenceCount << endl;

return 0;

}



Задание 3

Дана строка текста. Отсортировать первые 7 символов в

строке по алфавиту (или по таблице ASCII).

//3

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

string inputText;

// Ввод текста от пользователя

cout << "Введите строку текста: ";

getline(cin, inputText);

// Проверка, что строка содержит хотя бы 7 символов

if (inputText.length() < 7) {

cerr << "Строка должна содержать хотя бы 7 символов." << endl;

return 1;

}

// Сортировка первых 7 символов по алфавиту (или ASCII)

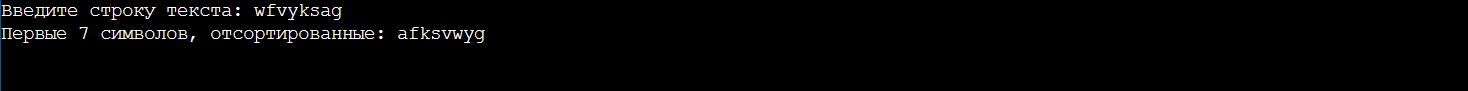
sort(inputText.begin(), inputText.begin() + 7);

// Вывод отсортированной строки

cout << "Первые 7 символов, отсортированные: " << inputText << endl;

return 0;

}



Задание 4

Даны две строки текста. Определить встречается ли хотя бы один раз каждый из символов первой строки во второй строке. Например: Пусть исходная строка Str1 : “xyz”; Str2: “x a d c “х” – встречается 3 раза “y”— встречается 1 раз, “z”— встречается раз. Не каждый символ встречается , например “z” не встречается ни разу. Далее заменить все слова «компьютер» на слова «ПК» .

//4

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main() {

string str1, str2;

// Ввод двух строк от пользователя

cout << "Введите первую строку: ";

getline(cin, str1);

cout << "Введите вторую строку: ";

getline(cin, str2);

// Проверка встречаемости каждого символа из str1 в str2

cout << "Проверка встречаемости символов первой строки во второй:" << endl;

for (char ch : str1) {

int count = std::count(str2.begin(), str2.end(), ch);

cout << "\"" << ch << "\" - встречается " << count << " раз(a)" << endl;

}

// Замена слов "компьютер" на "ПК"

size\_t pos = str2.find("computer");

while (pos != string::npos) {

str2.replace(pos, 8, "PC"); // Длина слова "компьютер" равна 9

pos = str2.find("computer", pos);

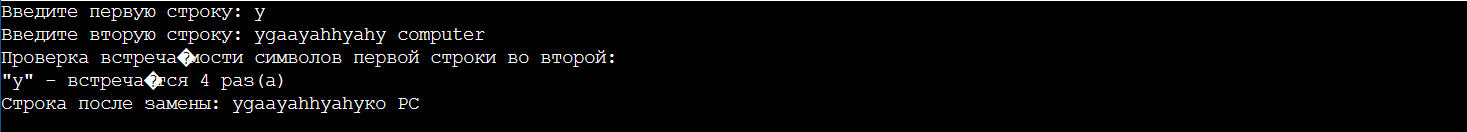
}

// Вывод измененной второй строки

cout << "Строка после замены: " << str2 << endl;

return 0;

}



Практическая работа №7

Задание 1.1

Main.cpp

//1.1

#include <iostream>

#include <stack>

#include <fstream>

#include <string>

using namespace std;

bool areBracketsBalanced(const string& code) {

stack<char> bracketStack;

for (char ch : code) {

if (ch == '(' || ch == '{' || ch == '[') {

bracketStack.push(ch);

} else if (ch == ')' || ch == '}' || ch == ']') {

if (bracketStack.empty()) {

return false; // Закрывающая скобка без соответствующей открывающей

}

char openBracket = bracketStack.top();

bracketStack.pop();

if ((openBracket == '(' && ch != ')') ||

(openBracket == '{' && ch != '}') ||

(openBracket == '[' && ch != ']')) {

return false; // Несоответствие типов скобок

}

}

}

return bracketStack.empty(); // В конце должны быть закрыты все открытые скобки

}

int main() {

ifstream file("test.txt"); // Замените "your\_file\_name.txt" на имя вашего файла

if (!file.is\_open()) {

cerr << "Unable to open the file" << endl;

return 1;

}

string code((istreambuf\_iterator<char>(file)), istreambuf\_iterator<char>());

file.close();

if (areBracketsBalanced(code)) {

cout << "Скобки сбалансированы." << endl;

} else {

cout << "Скобки не сбалансированы." << endl;

}

return 0;

}

Text.txt

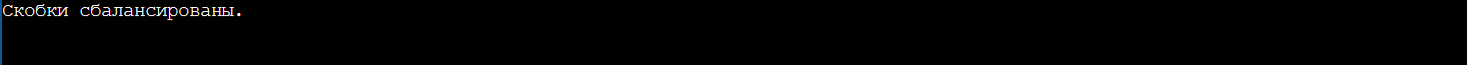
{

(Юрасов Никита Дмитриевич 2005 10 20000)

(Зулумханов Гаджи Андреевич 1963 15 37890.71)

(Скляров Владимир Дмитриевич 2005 8 10000)

}



Задание 1.2

Сформировать массив на диске, содержащий сведения о клиентах

предприятия. Класс содержит поля: фамилия, имя, отчество, дата рождения, число

покупок, потраченная сумма.

Написать программу, которая выбирает необходимую информацию с диска и

выводит на экран:

- список клиентов у которых покупок более чем на Х;

- средний размер покупки для клиентов пенсионного возраста (старше 60).

Main.cpp

//1.2

#include <iostream>

#include <fstream>

#include <vector>

#include <ctime>

#include <iomanip>

using namespace std;

struct Client {

string lastName;

string firstName;

string middleName;

int birthDate;

int purchases;

double spentAmount;

};

// Функция для чтения данных о клиентах из файла

vector<Client> readClientsFromFile(const string& test) {

vector<Client> clients;

ifstream file(test);

if (!file.is\_open()) {

cerr << "Unable to open the file" << endl;

return clients;

}

while (!file.eof()) {

Client client;

file >> client.lastName >> client.firstName >> client.middleName;

file >> client.birthDate;

file >> client.purchases >> client.spentAmount;

clients.push\_back(client);

}

file.close();

return clients;

}

// Функция для вывода информации о клиентах с покупками более чем на X

void displayHighPurchasesList(const vector<Client>& clients, int threshold) {

cout << "Список клиентов с покупками более чем " << threshold << ":" << endl;

for (const Client& client : clients) {

if (client.purchases > threshold) {

cout << client.lastName << " " << client.firstName << " " << client.middleName << endl;

}

}

cout << endl;

}

// Функция для вывода среднего размера покупки для клиентов пенсионного возраста

void displayAveragePurchaseForSeniors(const vector<Client>& clients, int retirementAge) {

int totalPurchases = 0;

int seniorClientsCount = 0;

for (const Client& client : clients) {

int age = client.birthDate;

if (age >= retirementAge) {

totalPurchases += client.purchases;

seniorClientsCount++;

}

}

if (seniorClientsCount > 0) {

double averagePurchase = static\_cast<double>(totalPurchases) / seniorClientsCount;

cout << "Средний размер покупки для клиентов пенсионного возраста: " << averagePurchase << endl;

} else {

cout << "Нет клиентов пенсионного возраста." << endl;

}

}

int main() {

vector<Client> clients = readClientsFromFile("test.txt");

if (!clients.empty()) {

displayHighPurchasesList(clients, 10);

displayAveragePurchaseForSeniors(clients, 60);

}

return 0;

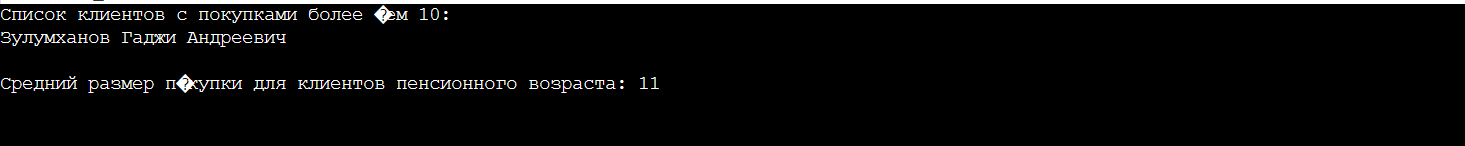
}

Text.txt

Юрасов Никита Дмитриевич 2005 10 20000

Зулумханов Гаджи Андреевич 1963 15 37890.71

Скляров Владимир Дмитриевич 2005 8 10000



Практическая работа №8

Задание 1

Определить  класс  «вектор  координат  в  пространстве».  В  класс  включить  два  конструктора  для

определения   вектора   по   умолчанию   и   путем   копирования   другого   вектора.   При   задании   вектора   по

умолчанию   предусмотреть   его   заполнение   случайными   числами.   Предусмотреть   функции   нахождения

длины вектора, умножения вектора на вектор, скалярного произведения векторов.

#include <iostream>

#include <cmath>

#include <cstdlib>

#include <ctime>

using namespace std;

class Vector3D {

private:

double x, y, z;

public:

// Конструктор по умолчанию

Vector3D() {

// Заполняем вектор случайными числами

srand(static\_cast<unsigned>(time(nullptr)));

x = rand() % 100;

y = rand() % 100;

z = rand() % 100;

}

// Конструктор копирования

Vector3D(const Vector3D& other) {

x = other.x;

y = other.y;

z = other.z;

}

// Функция для нахождения длины вектора

double length() const {

return sqrt(x \* x + y \* y + z \* z);

}

// Функция для умножения вектора на вектор

Vector3D multiply(const Vector3D& other) const {

Vector3D result;

result.x = x \* other.x;

result.y = y \* other.y;

result.z = z \* other.z;

return result;

}

// Функция для скалярного произведения векторов

double dotProduct(const Vector3D& other) const {

return x \* other.x + y \* other.y + z \* other.z;

}

// Функция для вывода координат вектора

void print() const {

cout << "(" << x << ", " << y << ", " << z << ")";

}

};

int main() {

// Использование класса

Vector3D defaultVector; // Создание вектора по умолчанию

Vector3D copiedVector(defaultVector); // Создание вектора путем копирования другого вектора

cout << "Длина вектора по умолчанию: " << defaultVector.length() << endl;

cout << "Умножение вектора по умолчанию на скопированный вектор: ";

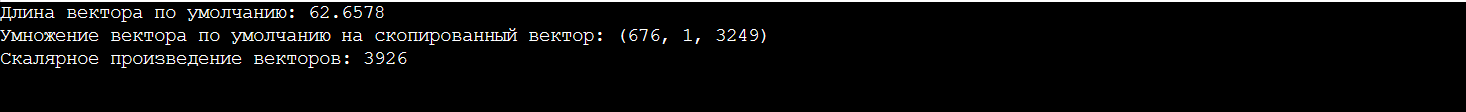
copiedVector.multiply(defaultVector).print();

cout << std::endl;

cout << "Скалярное произведение векторов: " << defaultVector.dotProduct(copiedVector) << endl;

return 0;

}



Практическая работа №9

Задание 1

Программа должна содержать:

базовый класс XX, включающий два элемента x1x1, x2x2 типа int;

конструктор с параметрами для создания объектов в динамической области

памяти;

деструктор;

виртуальные методы просмотра текущего состояния и переустановки

объектов базового класса в новое состояние.

Производный класс YY, включающий один элемент yy типа int:

конструктор с параметрами и списком инициализаторов, передающий

данные конструктору базового класса;

переопределенные методы просмотра текущего состояния объектов и их

переустановки в новое состояние.

Индивидуальные задания

Создать в производном классе метод Run, определяющий:

Значение x1/y-x2.

#include <iostream>

using namespace std;

class XX {

protected:

int x1, x2;

public:

// Конструктор с параметрами для создания объектов в динамической области памяти

XX(int x1\_value, int x2\_value) : x1(x1\_value), x2(x2\_value) {

cout << "Конструктор XX" << endl;

}

// Деструктор

virtual ~XX() {

cout << "Деструктор XX" << endl;

}

// Виртуальный метод просмотра текущего состояния

virtual void viewState() const {

cout << "Текущее состояние XX: x1 = " << x1 << ", x2 = " << x2 << endl;

}

// Виртуальный метод переустановки объектов в новое состояние

virtual void resetState(int new\_x1, int new\_x2) {

x1 = new\_x1;

x2 = new\_x2;

cout << "Объект XX переустановлен в новое состояние" << endl;

}

};

class YY : public XX {

private:

int yy;

public:

// Конструктор с параметрами и списком инициализаторов

YY(int x1\_value, int x2\_value, int yy\_value)

: XX(x1\_value, x2\_value), yy(yy\_value) {

cout << "Конструктор YY" << endl;

}

// Переопределенный метод просмотра текущего состояния объектов

void viewState() const override {

cout << "Текущее состояние YY: x1 = " << x1 << ", x2 = " << x2 << ", yy = " << yy << endl;

}

// Переопределенный метод переустановки объектов в новое состояние

void resetState(int new\_x1, int new\_x2, int new\_yy){

XX::resetState(new\_x1, new\_x2); // Вызываем метод базового класса

yy = new\_yy;

cout << "Объект YY переустановлен в новое состояние" << endl;

}

// Метод Run

void Run() {

if (x2 != 0) {

double result = static\_cast<double>(x1) / yy - x2;

cout << "Результат выражения x1 / yy - x2: " << result << endl;

} else {

cerr << "Ошибка: деление на ноль!" << endl;

}

}

};

int main() {

// Создаем объект YY на динамической области памяти

YY\* objYY = new YY(10, 5, 3);

// Выводим текущее состояние объекта YY

objYY->viewState();

// Переустанавливаем состояние объекта YY

objYY->resetState(20, 4, 5);

// Выводим текущее состояние объекта YY после переустановки

objYY->viewState();

// Вызываем метод Run

objYY->Run();

// Освобождаем память

delete objYY;

return 0;

}

