Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 10

По дисциплине «Основы алгоритмизации и программирования»

На тему «Рекурсивные алгоритмы»

Выполнил:

Кулешов Артём

Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

4. В соответствии со своим вариантом выполнить задания из таблицы, представленной ниже. В некоторых заданиях имеются ошибки: не выполняется условие завершения рекурсии. Изменить условие такой задачи с тем, чтобы рекурсия выполнялась.

Вариант №5

Бином Ньютона определяется как: **Сnm = n! / (m!(n – m)!)**

Рекурсивно описать функцию  **C(m, n)**, где **0 <= m <= n** для биноминального коэффициента **Сn**.

Формулы имеют следующий вид:

**Сno = Сnn = 1;**

**Сnm = Сn-1m + Сn-1m-1**

#include <iostream>

using namespace std;

//рекурсивная функция для вычисления биномиального коэффициента//

long long binom(int n, int m) {

if (m == 0 || m == n) { //базовые случаи//

return 1;

}

else {

return binom(n - 1, m - 1) + binom(n - 1, m); //формула бинома ньютона для коэфф.//

}

}

int main() {

int n, m;

cout << "Enter n and m (0 <= m <= n): ";

cin >> n >> m;

if (n < 0 || m < 0 || m > n) {

cout << "Error." << endl;

return 1;

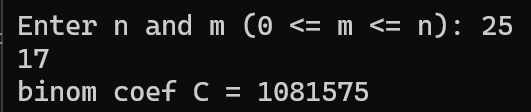
}

long long result = binom(n, m); //вычисление и вывод//

cout << "binom coef C = " << result << endl;

return 0;

}



5. К номеру своего варианта прибавить число 2 и написать программу для новых исходных данных (для вариантов 15, 16 перейти к вариантам 1, 2).

Вычислить выражение в правой части, используя рекурсию.



#include <iostream>

#include <cmath>

using namespace std;

//рукурсивная функция дл вычисления суммы синусов//

double sin(int n, double x)

{

if (n == 1) //базовый случай//

{

return sin(x);

}

else {

return sin(n \* x) + sin(n - 1, x); //сумма синусов//

}

}

int main()

{

int n;

double x;

cout << "Enter n: ";

cin >> n;

cout << "Enter x (rad): ";

cin >> x;

if (n <= 0)

{

cout << "Error." << endl;

return 1;

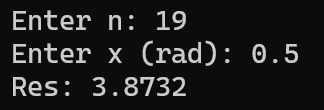
}

double result = sin(n, x); //вычисление и вывод//

cout << "Res: " << result << endl;

return 0;

}



6. Дополнительные задания.

1. Ввести цифру **А**, записать в файл все возможные числа, состоящие из цифр, не превышающих или равных **A**. Количество цифр в числах должно быть равно **А**.

Примечание: использовать дополнительный массив.

#include <iostream>

#include <fstream> /

using namespace std;

int i, c, b = 0;

//функция заполняет массив array значениями от 1 до x и возвращает количество элементов//

int filling(int x, int max, int\* array)

{

//рекурсивная функция, заполняющая массив значениями от 1 до x//

if (i <= x) {

array[c] = i;

++i;

c++; //увеличиваем счетчик элементов//

filling(x, max, array); //вызов функции для след. элемента//

}

else {

return c;

}

}

int main()

{

int a;

cout << "Enter value: ";

cin >> a;

int\* arr = new int[a]; //выделение памяти под массив arr размером a//

ofstream out("file.txt");

int counter = 1; //начальное значение счетчика для функции//

int count = filling(a, counter, arr);

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

cout << arr[i] << endl;

}

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

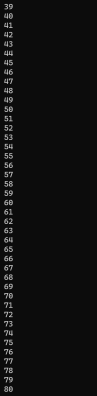
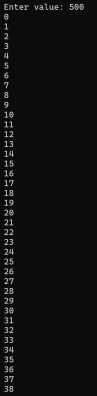
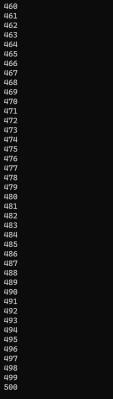
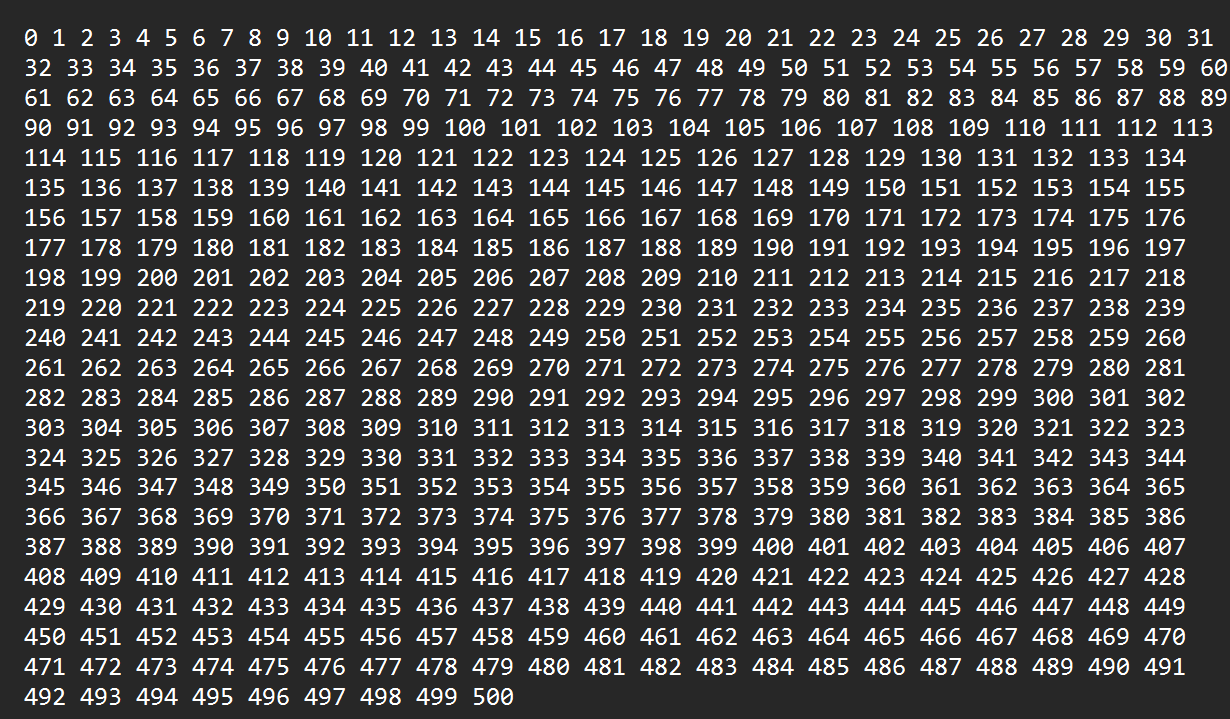
out << arr[i] << " ";

}

delete[] arr;

return 0;

}

 и тд… 

2. Задача проведения границы на карте («создание военных блоков»). Страны на карте заданы матрицей смежности. Если страны **i**, **j** имеют на карте общую границу, то элемент матрицы **A[i, j]** равен 1, иначе 0.

Необходимо разбить страны на две группы так, чтобы количество пар смежных стран из противоположных групп было минимальным.

#include <iostream>

using namespace std;

const int MAX\_COUNTRIES = 4; //кол-во стран//

int adjacencyMatrix[MAX\_COUNTRIES][MAX\_COUNTRIES] = { //матрица смежности//

{0, 1, 0, 1},

{1, 0, 1, 0},

{0, 1, 0, 1},

{1, 0, 1, 0}

};

int groups[MAX\_COUNTRIES];

//рекурсивная функция//

void dfs(int node, int group) {

groups[node] = group;

for (int i = 0; i < MAX\_COUNTRIES; ++i) {

if (adjacencyMatrix[node][i] == 1 && groups[i] == -1) {

dfs(i, 1 - group);

}

}

}

int main() {

for (int i = 0; i < MAX\_COUNTRIES; ++i) { //инициализация массива//

groups[i] = -1;

}

for (int i = 0; i < MAX\_COUNTRIES; ++i) { //разделение стран на группы//

if (groups[i] == -1) {

dfs(i, 0);

}

}

cout << "Groups of countries: " << endl;

for (int i = 0; i < MAX\_COUNTRIES; ++i) {

if (groups[i] == 0) {

cout << "Country " << i + 1 << ": Group 1" << endl;

}

else {

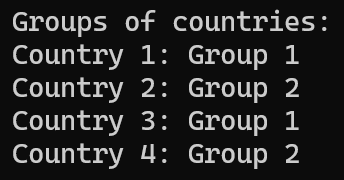
cout << "Country " << i + 1 << ": Group 2" << endl;

}

}

return 0;

}



3. Дано **n** различных натуральных чисел (**n = 5**). Напечатать все перестановки этих чисел.

#include <iostream>

using namespace std;

const int n = 5;

int a[n];

//функция для перестановки элементов массива//

void perestanovka(int k) {

int i, j, t, p;

for (i = 0; i < n - k; i++) { //перебор всех возможных перестановок//

if (k < n - 2)

perestanovka(k + 1); //рекурсивный вызов функции для след. элемента//

else {

for (p = 0; p < n; p++) cout << " " << a[p];

cout << endl;

}

t = a[k];

for (j = k + 1; j < n; j++) a[j - 1] = a[j]; //cдвиг элементов массива влево//

a[j - 1] = t; //перемещаем сохраненный элемент на правый край массива//

}

}

int main(void)

{

int i;

for (i = 0; i < n; i++) a[i] = i + 1;

perestanovka(0);

return 0;

}

