Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 10

По дисциплине «Основы алгоритмизации и программирования»

На тему «Рекурсивные алгоритмы»

Выполнил:

Студент 1 курса 10 группы

Жамойдо Артём Игоревич

Преподаватель: асс. Андронова М.В.

Минск, 2024

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 4**

Задан прямоугольник со сторонами **а** и **b** (**a, b** − натуральные числа). Разбить его на части с помощью квадратов и определить, сколько квадратов получится, если каждый раз выбирается самый большой квадрат.

#include <iostream>

#include <cmath>

using namespace std;

// Функция для нахождения количества квадратов при разбиении прямоугольника

int numsq(int a, int b)

{

if (b == 0)

{

return 0;

}

else

{

return a / b + numsq(b, a % b);

}

}

int main()

{

setlocale(0, "ru");

int a, b;

cout << "Введите стороны прямоугольника (a и b через пробел): ";

cin >> a >> b;

// Проверка на натуральные числа

if (a <= 0 || b <= 0)

{

cout << "Строны прямоугольника должны быть натуральными числами." << endl;

return 1;

}

// Обмен значений, чтобы a всегда было больше b

if (a < b)

swap(a, b);

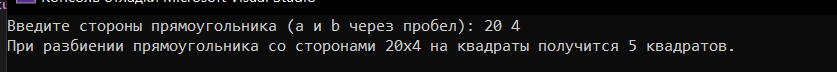
int result = numsq(a, b);

cout << "При разбиении прямоугольника со сторонами " << a << "x" << b

<< " на квадраты получится " << result << " квадратов." << endl;

return 0;

}



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

**Вариант 6**

Разработать программу, реализующую рекурсивный алгоритм вычисления **A(m, n)** для любых целых не отрицательных **m** и **n**.

#include <iostream>

using namespace std;

// Рекурсивная функция для вычисления A(m, n)

int function(int m, int n)

{

if (m == 0)

{

return n + 1;

}

else if (n == 0)

{

return function(m - 1, 1);

}

else

{

return function(m - 1, function(m, n - 1));

}

}

int main()

{

setlocale(0, "ru");

int m, n;

cout << "Введите целые неотрицательные числа m и n через пробел: ";

cin >> m >> n;

// Проверка на неотрицательные числа

if (m < 0 || n < 0)

{

cout << "Числа должны быть неотрицательными." << endl;

return 1;

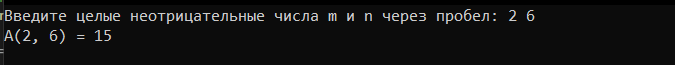
}

int result = function(m, n);

cout << "A(" << m << ", " << n << ") = " << result << endl;

return 0;

}



Доп. задачи

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;

// Рекурсивная функция для генерации и записи чисел в файл

void num(int n, int a[], int index, ofstream& outFile)

{

if (index == n)

{

// Записываем сгенерированное число в файл и переходим к следующей перестановке

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

{

outFile << a[i];

}

outFile << endl;

return;

}

// Генерируем все возможные перестановки чисел

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

{

a[index] = i;

num(n, a, index + 1, outFile);

}

}

int main()

{

setlocale(0, "ru");

int n;

cout << "Введите число: ";

cin >> n;

// Открываем файл для записи

ofstream outFile("num.txt");

// Создаем массив для хранения перестановок

int\* a = new int[n];

num(n, a, 0, outFile);

// Закрываем файл

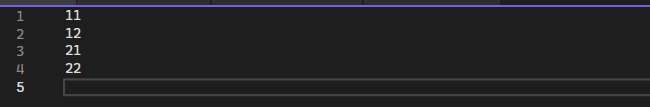
outFile.close();

cout << "Числа записаны в файл num.txt" << endl;

return 0;

}





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>

#include <vector>

#include <cmath>

using namespace std;

// Функция для проверки, можно ли добавить страну к определенной группе

bool isSafe(int country, int group, const vector<vector<int>>& A, const vector<int>& groups)

{

for (int i = 0; i < A.size(); i++)

{

// Если смежная страна уже в той же группе, то нельзя добавить текущую страну

if (A[country][i] && groups[i] == group)

{

return false;

}

}

return true;

}

// Рекурсивная функция для разделения стран на две группы

void divideCountries(int country, int totalCountries, vector<vector<int>>& A, vector<int>& groups, int& minPairs)

{

if (country == totalCountries)

{

// Подсчитываем количество смежных стран из разных групп

int pairs = 0;

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

{

for (int j = 0; j < totalCountries; j++)

{

if (A[i][j] && groups[i] != groups[j])

{

pairs++;

}

}

}

// Обновляем минимальное количество смежных стран из разных групп, если нашли лучшее разбиение

if (pairs < minPairs)

{

minPairs = pairs;

}

return;

}

// Для текущей страны пробуем добавить ее в обе группы и рекурсивно вызываем функцию для следующей страны

for (int group = 0; group < 2; group++)

{

if (isSafe(country, group, A, groups))

{

groups[country] = group;

divideCountries(country + 1, totalCountries, A, groups, minPairs);

groups[country] = -1; // Сбрасываем группу после возврата из рекурсии

}

}

}

int main()

{

setlocale(0, "ru");

int totalCountries;

cout << "Введите общее количество стран на карте: ";

cin >> totalCountries;

vector<vector<int>> A(totalCountries, vector<int>(totalCountries));

cout << "Введите матрицу смежности стран (1 - смежные, 0 - нет):\n";

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

{

for (int j = 0; j < totalCountries; ++j)

{

cin >> A[i][j];

}

}

vector<int> groups(totalCountries, -1); // Инициализируем все страны без группы

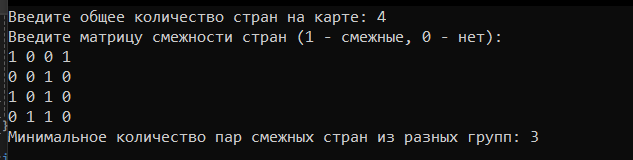
int minPairs = INT\_MAX;

divideCountries(0, totalCountries, A, groups, minPairs);

cout << "Минимальное количество пар смежных стран из разных групп: " << minPairs << endl;

return 0;

}



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

#include <iostream>

#include <fstream>

using namespace std;

// Рекурсивная функция для генерации всех перестановок чисел

void num(int n, int a[], int index, ofstream& outFile)

{

if (index == n)

{

// Записываем перестановку в файл и переходим к следующей перестановке

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

{

outFile << a[i] << " ";

}

outFile << endl;

return;

}

// Генерируем все возможные перестановки чисел

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

{

bool found = false;

// Проверяем, было ли число i уже использовано в текущей перестановке

for (int j = 0; j < index; j++)

{

if (a[j] == i)

{

found = true;

break;

}

}

// Если число i не было использовано, добавляем его к перестановке

if (!found)

{

a[index] = i;

num(n, a, index + 1, outFile);

}

}

}

int main()

{

setlocale(0, "ru");

int n = 5;

int\* a = new int[n]; // Создаем массив для хранения перестановок

ofstream outFile("num.txt"); // Открываем файл для записи перестановок

num(n, a, 0, outFile);

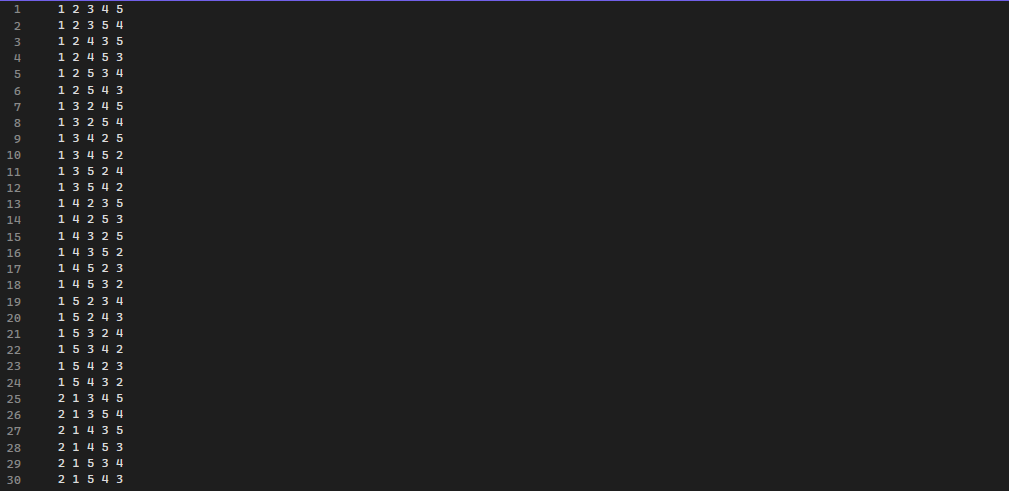
outFile.close(); // Закрываем файл

cout << "Перестановки чисел записаны в файл num.txt" << endl;

return 0;

}

****

****