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

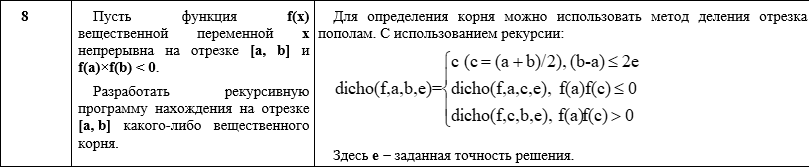
Мандрик Алексей Иванович

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 8**



**Код программы:**

#include <iostream>

#include <cmath>

using namespace std;

// Функция, для которой ищем корень

double f(double x) {

// Здесь нужно определить вашу функцию

// Например, для квадратного уравнения: return x\*x - 4; для удобства проверки

return x \* x - 4;

}

// Рекурсивная функция для нахождения корня

double findRoot(double a, double b, double eps) {

double c = (a + b) / 2; // Находим середину отрезка

if (abs(f(c)) < eps) // Проверяем условие останова: если значение функции на середине отрезка близко к нулю

return c; // Возвращаем найденный корень

if (f(a) \* f(c) < 0) // Если функция меняет знак между a и c

return findRoot(a, c, eps); // Рекурсивно ищем корень на отрезке [a, c]

else // Иначе

return findRoot(c, b, eps); // Рекурсивно ищем корень на отрезке [c, b]

}

int main() {

setlocale(LC\_ALL, "Rus");

double a, b, eps;

cout << "Введите начало отрезка a: ";

cin >> a;

cout << "Введите конец отрезка b: ";

cin >> b;

cout << "Введите точность eps: ";

cin >> eps;

// Проверяем условие f(a) \* f(b) < 0

if (f(a) \* f(b) >= 0) {

cout << "На отрезке [" << a << ", " << b << "] нет корня" << endl;

return 0;

}

// Вызываем функцию для нахождения корня

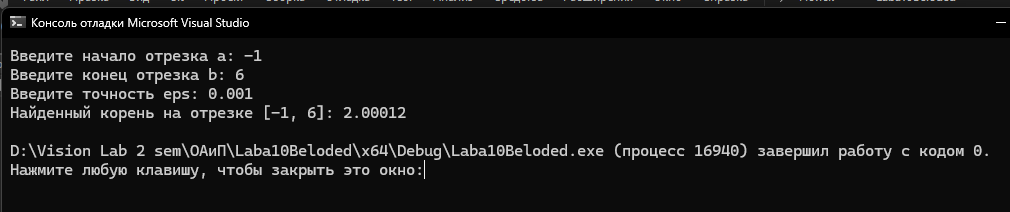
double root = findRoot(a, b, eps);

cout << "Найденный корень на отрезке [" << a << ", " << b << "]: " << root << endl;

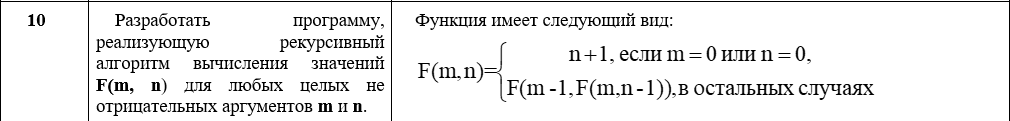
return 0;

}

}**Результат выполнения:**



Вариант 10



Код программы:

#include <iostream>

// Функция для вычисления значения F(m, n) по заданному условию

int F(int m, int n) {

if (m == 0 || n == 0) {

// Базовый случай: если m равно 0 или n равно 0, возвращаем n + 1

return n + 1;

}

else {

// Рекурсивный случай: вычисляем значение F(m - 1, F(m, n - 1))

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

}

}

int main() {

setlocale(LC\_ALL, "Rus");

int m, n;

std::cout << "Введите значения m и n: ";

std::cin >> m >> n;

// Проверяем, что введенные значения неотрицательны

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

std::cout << "Ошибка: Значения m и n должны быть неотрицательными." << std::endl;

return 1;

}

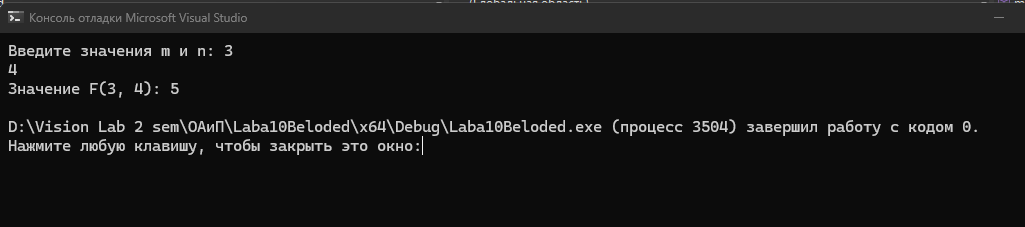
// Вызываем функцию F и выводим результат

std::cout << "Значение F(" << m << ", " << n << "): " << F(m, n) << std::endl;

return 0;

}

Результат выполнения:



**Дополнительные задания:**

**Доп 1:**

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.

**Код программы:**

#include <iostream>

#include <cmath>

#include <string>

using namespace std;

const int number[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }; // Массив цифр от 0 до 9

// Функция для вывода массива

void out(int\* arr, int a) {

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

cout << arr[k];

cout << "\n";

}

// Функция для проверки, является ли массив максимальным числом

bool isMax(int\* arr, int a) {

for (int t = 0; t < a; t++) {

if (arr[t] != a) {

return false;

}

}

return true;

}

// Рекурсивная функция для генерации чисел

void loop(int\*& arr, int left, int a) {

while (left < a) {

if (left > 0)

loop(arr, left - 1, a);

if (arr[left] == a) {

if (isMax(arr, a)) {

out(arr, a); // Выводим число

exit(1);

}

arr[left] = 0;

arr[left + 1]++;

if (arr[left + 1] > a) {

arr[left + 1] = a;

break;

}

}

else {

out(arr, a); // Выводим число

arr[left]++;

}

}

}

int main()

{

cin.tie(NULL); // Отключение синхронизации с потоками ввода-вывода C и C++

cout.tie(NULL);

ios\_base::sync\_with\_stdio(false); // Отключение синхронизации потоков ввода-вывода C и C++

setlocale(LC\_ALL, "Rus");

int a;

cin >> a; // Ввод числа

if (a > 9 || a < 1) { // Проверка, является ли число от 1 до 9

cout << "A - должно быть цифрой! " << endl;

return 1;

}

int\* arr{ new int[a] }; // Создание массива для хранения чисел

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

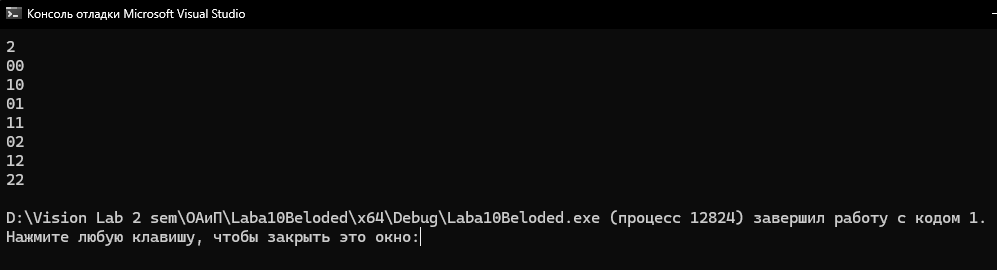
arr[i] = 0; // Инициализация массива нулями

loop(arr, a - 1, a); // Вызов рекурсивной функции для генерации чисел

return 0;

}

**Результат выполнения:**

****

**Доп 2:**

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.

**Код программы:**

#include <iostream>

using namespace std;

// Функция для получения символа по его порядковому номеру

char getSymbol(int n) {

if (n == 1) { // Базовый случай: если n равно 1, возвращаем символ 'A'

return 'A';

}

else { // Рекурсивный случай: вычисляем символ для n - 1 и прибавляем к нему 1

return (getSymbol(n - 1) - 'A' + 1) % 26 + 'A'; // Используем остаток от деления на 26 для цикличности алфавита

}

}

int main() {

setlocale(LC\_ALL, "Rus"); // Установка локали для вывода на русском языке

cout << "Введите n: "; // Запрос на ввод числа n

int n;

cin >> n; // Ввод числа n

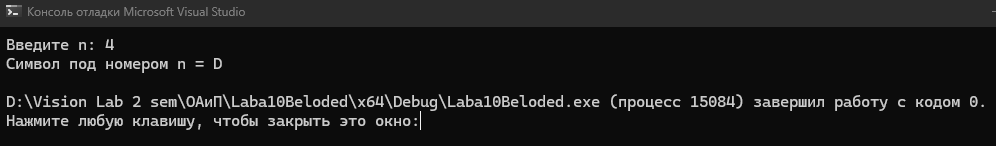
char symbol = getSymbol(n); // Получение символа для заданного n

cout << "Символ под номером n = " << symbol << endl; // Вывод символа

return 0;

}

**Результат выполнения:**



**Доп 3:**

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

**Код программы:**

#include <iostream>

using namespace std;

const int n = 5; // Количество элементов в перестановке

int a[n]; // Массив для хранения перестановки

void perm(int k) {

int i, j, t, p;

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

if (k < n - 2)

perm(k + 1); // Рекурсивный вызов для получения следующих перестановок

else { // Вывод готовой перестановки

for (p = 0; p < n; p++) cout << " " << a[p]; // Вывод элементов перестановки

cout << endl;

}

// Циклический сдвиг

t = a[k]; // Запоминаем текущий элемент

for (j = k + 1; j < n; j++) a[j - 1] = a[j]; // Сдвигаем все элементы на одну позицию влево

a[j - 1] = t; // Помещаем запомненный элемент в конец перестановки

}

}

int main(void)

{

int i;

// Заполнение массива значениями от 1 до n

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

perm(0); // Начинаем генерацию перестановок с первого элемента

return 0;

}

**Результат выполнения:**

