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 6 группы

Кучерук Николай Петрович

Преподаватель: асс. Андронова М.В.

2023, Минск

**Задание**

|  |  |
| --- | --- |
| **№ варианта** | **Условие задачи** |
| 9 | Пусть для целых неотрицательных чисел **n**, **m** разрешены операции нахождения последующего числа **(n + 1)** и предыдущего числа **n - 1** **(n > 0)**.  С помощью рекурсивных функций определить операции нахождения суммы **(n + m)**, разности **(n − m)**,умножения **(n × m)**, возведения в степень **n ^ m (n > 0)**. |

|  |
| --- |
| **Код программы** |
| #include <iostream>  int sum(int n, int m) {// сумма  if (m == 0)  return n;  else  return sum(n + 1, m - 1);  }  int diff(int n, int m) {// разность  if (m == 0)  return n;  else  return diff(n - 1, m - 1);  }  int mult(int n, int m) {// умножение  if (m == 0)  return 0;  else  return n + mult(n, m - 1);  }  int power(int n, int m) {// возведение в степень  if (m == 0)  return 1;  else  return n \* power(n, m - 1);  }  int main(void)  {  system("color 70");  int n=2,m=3;  std::cout << sum(n, m)<<std::endl;  std::cout << diff(n, m) << std::endl;  std::cout << mult(n, m) << std::endl;  std::cout << power(n, m) << std::endl;    return 0;  } |
| **Результат программы** |
|  |

|  |  |
| --- | --- |
| **№ варианта** | **Функция f(x)** |
| 11 | Разработать программу, реализующую рекурсивный алгоритм для вычисления значений полиномов **Sn(x)**. |

|  |
| --- |
| **Код программы** |
| #include <iostream>  int polinom(int n,int x) {// функция нахождения значений полиномов  if (n == 0)  return n;  if (n == 1)  return 2\*x;  if (n > 1) {  return 2 \* n / (n - 1) \* polinom(n - 1, x) + ((n - 1) / (2 \* x))\*polinom(n-2,x);  }  }  int main(void)  {  system("color 70");  int n=3,x=15;  std::cout << polinom(n, x)<<std::endl;    return 0;  } |
| **Результат программы** |
|  |

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.

|  |
| --- |
| **Код программы** |
| #include <iostream>  #include <fstream>  using namespace std;  void filewr(int array[],int size, ofstream& file) {//функция для записи чисел в файл  for (int i = 0; i<size; i++) {  file << array[i];  }  file << "\n";  return;  }  // функция для генерации подходящих чисел в файл  void generate(int digit,int count, int array[],ofstream& file) {  if (count == digit) {  filewr(array,count, file);  return;  }  for (int i = 0; i <= digit; ++i) {  array[count] = i; // добавляем текущую цифру в текущее число  generate(digit, count + 1, array,file); //вызов рекурсивной функции  }    }  int main() {  system("color 70");  setlocale(LC\_CTYPE, "Russian");  int A;  cout << "Введите цифру A: ";  cin >> A;  // открываем файл для записи  ofstream file("numbers.txt");  if (!file) {  cerr << "Не удалось открыть файл для записи." << endl;  return 1;  }  int\* array = new int[A];//  // генерация и запись чисел в файл  generate(A, 0, array,file);  file.close();  cout << "Числа записаны в файл numbers.txt." << endl;  return 0;  }  кк |
| **Результат программы** |
|  |

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>  #include <vector>  #include <Windows.h>  using namespace std;  // Функция для определения двудольности графа с использованием рекурсии  // Параметры:  // - adjacencyMatrix: матрица смежности графа  // - current: текущая вершина, которую мы проверяем  // - colors: вектор цветов вершин (0 или 1, -1 означает неокрашенную вершину)  // - color: цвет текущей вершины  // Возвращает true, если граф двудольный, и false в противном случае  bool isBipartiteRecursive(const std::vector<std::vector<int>>& adjacencyMatrix, int current, std::vector<int>& colors, int color) {  colors[current] = color; // Окрашиваем текущую вершину  // Перебираем соседние вершины  for (int neighbor = 0; neighbor < adjacencyMatrix.size(); ++neighbor) {  if (adjacencyMatrix[current][neighbor] == 1) {  // Если соседняя вершина неокрашена, вызываем рекурсивно с другим цветом  if (colors[neighbor] == -1 && !isBipartiteRecursive(adjacencyMatrix, neighbor, colors, 1 - color)) {  return false; // Граф не двудольный  }  // Если соседняя вершина окрашена в тот же цвет, граф не двудольный  else if (colors[neighbor] == color) {  return false;  }  }  }  return true; // Все вершины правильно окрашены, граф двудольный  }  // Функция для определения двудольности графа с использованием рекурсии  // Параметры:  // - adjacencyMatrix: матрица смежности графа  // - start: вершина, с которой начинается проверка  // - colors: вектор цветов вершин (0 или 1, -1 означает неокрашенную вершину)  // Возвращает true, если граф двудольный, и false в противном случае  bool isBipartite(const std::vector<std::vector<int>>& adjacencyMatrix, int start, std::vector<int>& colors) {  return isBipartiteRecursive(adjacencyMatrix, start, colors, 0);  }  // Функция для вывода стран в группу  // Параметры:  // - colors: вектор цветов вершин  // - group: номер группы  void printGroup(const vector<int>& colors, int group) {  cout << "Группа " << group << ": ";  for (int i = 0; i < colors.size(); ++i) {  if (colors[i] == group) {  cout << i + 1 << " "; // Страны нумеруются с 1  }  }  cout << std::endl;  }  int main() {  SetConsoleOutputCP(1251);  SetConsoleCP(1251);  // Пример матрицы смежности для стран  vector<vector<int>> adjacencyMatrix = {  {0, 1, 1, 0},  {1, 0, 0, 1},  {1, 0, 0, 1},  {0, 1, 1, 0}  };  int numCountries = adjacencyMatrix.size();  vector<int> colors(numCountries, -1); // -1 означает, что вершина еще неокрашена  // Проверяем двудольность графа, начиная с каждой вершины  for (int i = 0; i < numCountries; ++i) {  if (colors[i] == -1 && !isBipartite(adjacencyMatrix, i, colors)) {  cout << "Границу нельзя разделить на две группы с минимальным количеством смежных пар" << endl;  return 1;  }  }  cout << "Границу можно разделить на две группы с минимальным количеством смежных пар" << endl;  // Выводим страны в соответствующие группы  printGroup(colors, 0);  printGroup(colors, 1);  return 0;  } |
| **Результат программы** |
|  |

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

|  |
| --- |
| **Код программы** |
| #include <iostream>  using namespace std;  // функция для перестановки значений  void swap(int& a, int& b) {  int temp = a;  a = b;  b = temp;  }  //функция для генерации всех перестановок  void generate(int nums[], int start, int n) {  if (start == n - 1) {  // если достигнут конец массива, печатаем текущую перестановку  for (int i = 0; i < n; ++i) {  cout << nums[i] << " ";  }  cout << endl;  }  else {  //с помощью позиции старт отделяем подмассив,для которго генерируем перестановки  for (int i = start; i < n; ++i) {  swap(nums[start], nums[i]);//меняем элементы  generate(nums, start + 1, n);// рекурсивно вызываем для оставшейся части массива  swap(nums[start], nums[i]);// возвращаем исходное состояние массива  }  }  }  int main() {  system("color 70");  setlocale(LC\_ALL, "RUS");  const int n = 5;  int numbers[n] = { 1, 2, 3, 4, 5 };  cout << "Перестановки:\n";  generate(numbers, 0, n);  return 0;  } |
| **Результат программы** |
|  |