Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №4**

Дисциплина: «Дискретная математика и математическая логика»

Тема: Полнота системы функций

Выполнил работу

студент группы ИВТ-22-2б

Мельников Г. В.

Проверила

Доцент кафедры ИТАС

Рустамханова Г. И.

Пермь, 2023

**Постановка задачи**

1. Программа должен иметь возможность задавать систему функций состоящую минимум из трех функции длиной 8 символов и меньше

2. Программа должна выполнять следующие функции:

a. Создание и вывод на экран таблицы истинности функций.

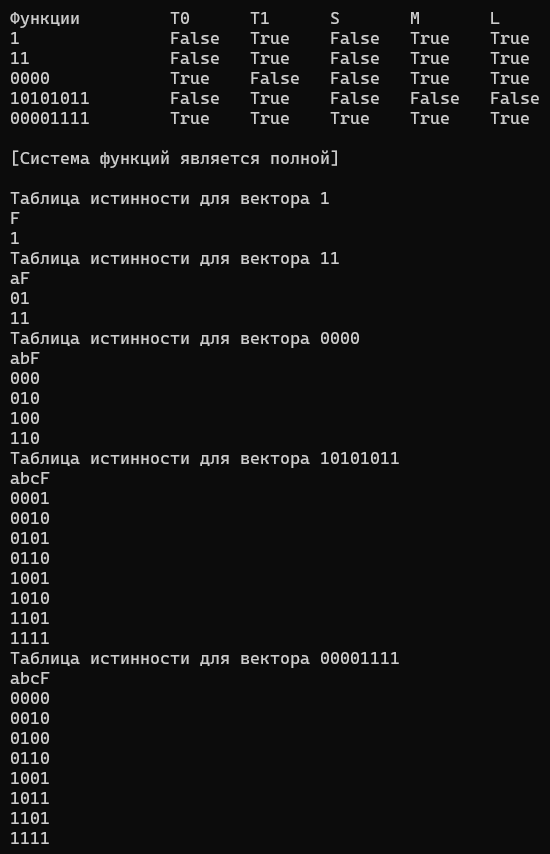
b. Показать к каким классам Поста относятся функции системы

c. Проверять систему на полноту

**Методы решения задачи**

1. Main(string[] args) – главный метод, вызывает остальные, частично выводит данные на консоль
2. PrintTrueTable(string functionVector) – выводит таблицу истинности для функции по её вектору
3. ReadFromFile(string filePath) – читает вектора функций из файла
4. CheckT0(string functionVector) – проверяет, принадлежит ли функция классу содержащий константу 0
5. CheckT1(string functionVector) - проверяет, принадлежит ли функция классу содержащий константу 1
6. CheckM(string functionVector) – проверяет, является ли функция монотонной
7. CheckS(string functionVector) – определяет является ли функция самодвойственной
8. CheckL(string functionVector) – проверяет функцию на линейность
9. CheckFullness(params string[] functionVectors) – проверяет, является ли система функций полной
10. WritePostsTable(params string[] functionVectors) – выводит таблицу значений классов Поста функций

**Результаты**

****

**Код программы**

namespace PostClasses

{

class Program

{

private static void Main(string[] args)

{

string rootProject = Directory.GetParent(Directory.GetCurrentDirectory())!

.Parent!.Parent!.FullName;

string filePath = rootProject + "\\1.txt";

string[] functionVectors = ReadFromFile(filePath);

WritePostsTable(functionVectors);

Console.WriteLine();

if (CheckFullness(functionVectors))

{

Console.WriteLine("[Система функций является полной]");

}

else

{

Console.WriteLine("[Система функций не является полной]");

}

Console.WriteLine();

foreach (var i in functionVectors)

{

Console.WriteLine($"Таблица истинности для вектора {i}");

PrintTrueTable(i);

}

}

public static void PrintTrueTable(string functionVector)

{

if (functionVector.Length == 1)

{

Console.WriteLine($"F\n{functionVector}");

}

else

{

int countVariables = (int)Math.Log2(functionVector.Length);

string stringVariables = "";

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

{

stringVariables += (char)('a' + i);

}

Console.WriteLine($"{stringVariables}F");

for (var i = 0; i < functionVector.Length; i++)

{

Console.WriteLine($"{Convert.ToString(i, 2).PadLeft(countVariables, '0')}{functionVector[i]}");

}

}

}

public static string[] ReadFromFile(string filePath)

{

if (!File.Exists(filePath))// Проверка наличия файла

{

throw new FileNotFoundException("Файл не найден!");

}

StreamReader reader = new(filePath);// Чтение строк из файла

int vectorsCount = int.Parse(reader.ReadLine()!);// Чтение количества строк из файла

string[] functionVectors = new string[vectorsCount];

for (int i = 0; i < vectorsCount; i++)// Чтение строк и добавление их в список

{

functionVectors[i] = reader.ReadLine()!;

}

return functionVectors;

}

private static bool CheckT0(string functionVector)

{

return functionVector[0] == '0';

}

private static bool CheckT1(string functionVector)

{

return functionVector[^1] == '1';

}

private static bool CheckM(string functionVector)

{

if (functionVector.Length == 1)

{

return true;

}

bool isCheck = true;

int len = functionVector.Length;

if (len == 2)

{

return functionVector[0] <= functionVector[1];

}

else if (len == 4)

{

for (int i = 0; i < len && isCheck; i += 2)

{

isCheck = functionVector[i] <= functionVector[i + 1];

}

isCheck = isCheck && functionVector[..(len / 2)].CompareTo(functionVector.Substring(len / 2)) != 1;

return isCheck;

}

else if (len == 8)

{

for (int i = 0; i < len && isCheck; i += 2)

{

isCheck = functionVector[i] <= functionVector[i + 1];

}

for (int i = 0; i < len && isCheck; i += 4)

{

isCheck = functionVector.Substring(i, len / 4).CompareTo(functionVector.Substring(i + len / 4, len / 4)) != 1;

}

isCheck = isCheck && functionVector[..(len / 2)].CompareTo(functionVector.Substring(len / 2, len / 2)) != 1;

return isCheck;

}

else

{

return false;

}

}//Сырая версия метода

private static bool CheckS(string functionVector)

{

if (functionVector.Length == 1) return false;

bool isS = true;

for (int i = 0; i <= functionVector.Length / 2 - 1 && isS; i++)

{

isS = functionVector[i] != functionVector[functionVector.Length - 1 - i];

}

return isS;

}

private static bool CheckL(string functionVector)

{

if (functionVector.Length < 4) return true;

int[] cValues = new int[functionVector.Length];

if (functionVector.Length == 4)

{

cValues[0] = int.Parse(functionVector[0].ToString());

cValues[1] = int.Parse(functionVector[2].ToString()) == 0 ?

cValues[0] : cValues[0] == 0 ?

1 : 0;

cValues[2] = int.Parse(functionVector[1].ToString()) == 0 ?

cValues[0] : cValues[0] == 0 ?

1 : 0;

cValues[3] = int.Parse(functionVector[3].ToString()) == 0 ?

(cValues[0] + cValues[1] + cValues[2]) % 2 == 0 ? 0 : 1 :

(cValues[0] + cValues[1] + cValues[2]) % 2 == 1 ? 0 : 1;

return cValues[3] != 1;

}

else if (functionVector.Length == 8)

{

cValues[0] = int.Parse(functionVector[0].ToString());

cValues[1] = int.Parse(functionVector[4].ToString()) == 0 ?

cValues[0] : cValues[0] == 0 ? 1 : 0;

cValues[2] = int.Parse(functionVector[2].ToString()) == 0 ?

cValues[0] : cValues[0] == 0 ? 1 : 0;

cValues[3] = int.Parse(functionVector[1].ToString()) == 0 ?

cValues[0] : cValues[0] == 0 ? 1 : 0;

cValues[4] = int.Parse(functionVector[6].ToString()) == 0 ?

(cValues[0] + cValues[1] + cValues[2]) % 2 == 0 ? 0 : 1 : (cValues[0] + cValues[1] + cValues[2]) % 2 == 1 ? 0 : 1;

if (cValues[4] == 1)

{

return false;

}

cValues[5] = int.Parse(functionVector[5].ToString()) == 0 ?

(cValues[0] + cValues[1] + cValues[3]) % 2 == 0 ? 0 : 1 : (cValues[0] + cValues[1] + cValues[3]) % 2 == 1 ? 0 : 1;

if (cValues[5] == 1)

{

return false;

}

cValues[6] = int.Parse(functionVector[3].ToString()) == 0 ?

(cValues[0] + cValues[2] + cValues[3]) % 2 == 0 ? 0 : 1 : (cValues[0] + cValues[2] + cValues[3]) % 2 == 1 ? 0 : 1;

if (cValues[6] == 1) return false;

cValues[7] = int.Parse(functionVector[7].ToString()) == 0 ?

(cValues[0] + cValues[1] + cValues[2] + cValues[3] + cValues[4] + cValues[5] + cValues[6] + cValues[7]) % 2 == 0 ?

0 : 1 : (cValues[0] + cValues[1] + cValues[2] + cValues[3] + cValues[4] + cValues[5] + cValues[6] + cValues[7]) % 2 == 1 ? 0 : 1;

return cValues[7] != 1;

}

else

{

return false;

}

}//Сырая версия метода

private static bool CheckFullness(params string[] functionVectors)

{

Func<string, bool>[] functions = new Func<string, bool>[5] { CheckT0, CheckT1, CheckS, CheckM, CheckL };

for (int i = 0; i < functions.Length; i++)

{

bool isFullness = false;

for (int j = 0; j < functionVectors.Length && !isFullness; j++)

{

isFullness = !functions[i](functionVectors[j]);

}

if (!isFullness)

{

return false;

}

}

return true;

}

private static void WritePostsTable(params string[] functionVectors)

{

Console.WriteLine($"Функции \tT0\tT1\tS\tM\tL");

foreach (var i in functionVectors)

{

Console.WriteLine($"{i}{new string(' ', 8 - i.Length)}\t{CheckT0(i)}\t{CheckT1(i)}\t" +

$"{CheckS(i)}\t{CheckM(i)}\t{CheckL(i)}");

}

}

}

}