Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Дискретная математика и математическая логика»

Тема: «Полнота функции»

Семестр 2

Выполнил работу

Студент группы ИВТ-23-2б

Меновщиков Глеб Николаевич

Проверил

старший преподаватель кафедры ИТАС

Рустамханова Гульшат Ильдаровна

**Цель программы**

Цель данной программы — анализировать логические функции и определять их принадлежность к различным классам полноты: T0, T1, S, M и L. Программа позволяет пользователю вводить логические функции, представленные в виде массивов значений, и выводит результаты анализа в виде таблицы, а также сообщает о полноте системы логических функций.

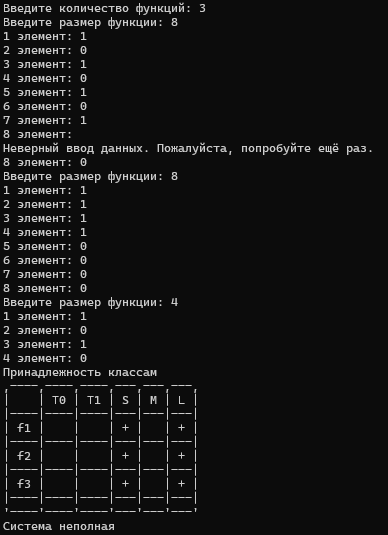
**Описание работы программы**

1. **Ввод количества функций**: Программа запрашивает у пользователя количество логических функций, которые он хочет ввести. Пользователь может ввести от 1 до 3 функций. Если введённое значение не соответствует этому диапазону, программа выводит сообщение об ошибке и запрашивает ввод снова.
2. **Ввод логических функций**: Для каждой логической функции программа запрашивает её размер, который должен быть равен 1, 2, 4 или 8 (то есть 2^i). Если введённое значение не соответствует этим требованиям, программа снова запрашивает ввод. После этого пользователь вводит значения для каждого элемента функции (0 или 1).
3. **Анализ логических функций**: Программа создает массив объектов **BooleanFunction**, каждый из которых представляет одну логическую функцию. Для каждой функции выполняются следующие проверки:
   * **IsT0**: Проверяет, является ли функция T0 (значение 0 для всех входов).
   * **IsT1**: Проверяет, является ли функция T1 (значение 1 для всех входов).
   * **IsS**: Проверяет, является ли функция S (симметричная).
   * **IsM**: Проверяет, является ли функция M (модулярная).
   * **IsL**: Проверяет, является ли функция L (линейная).
4. **Вывод результатов**: Программа формирует таблицу, в которой для каждой функции отображается её принадлежность к классам T0, T1, S, M и L. Для этого используется символ **+** для обозначения принадлежности и пробел для отсутствия принадлежности.
5. **Определение полноты системы**: Программа проверяет, принадлежат ли все функции к одному из классов. Если хотя бы одна функция принадлежит к классу T0, T1, S, M или L, программа выводит сообщение о том, что система неполная. В противном случае программа сообщает, что система полная.
6. **Обработка ошибок**: Программа включает обработку ошибок ввода, чтобы гарантировать, что пользователь вводит корректные значения.

**Структура программы**

* **Класс BooleanFunction**:
  + **Свойство Values**: Массив, хранящий значения логической функции.
  + **Методы**:
    - **SetValue(int index, int value)**: Устанавливает значение в массиве.
    - **IsT0()**: Проверяет, является ли функция T0.
    - **IsT1()**: Проверяет, является ли функция T1.
    - **IsS()**: Проверяет, является ли функция S.
    - **IsM()**: Проверяет, является ли функция M.
    - **IsL()**: Проверяет, является ли функция L.
    - **Operation(int i, int j)**: Выполняет логическую операцию (исключающее ИЛИ).
* **Класс Program**:
  + **Методы**:
    - **GetNumberInt(string str)**: Запрашивает у пользователя ввод целого числа.
    - **GetNumberIntVector(string str)**: Запрашивает у пользователя ввод значения для логической функции (0 или 1).
    - **ShowTable(BooleanFunction[] functions)**: Выводит таблицу принадлежности функций к классам.
    - **GetAnswer(BooleanFunction[] functions)**: Определяет полноту системы логических функций.
    - **ConvertSymbol(bool f)**: Преобразует логическое значение в символ для вывода.
* **Основной метод**: **Main(string[] args)** — точка входа в программу, где происходит последовательный вызов всех вышеописанных методов для выполнения анализа логических функций.

**Скриншоты работы программы**:



**Программа:**

[**https://github.com/Gpushak/CsharpUnProj/tree/7d357b4e79d29a5e293faec5bbd90852e5c73cee/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/4labaDiskMat/tochno3labaDisk**](https://github.com/Gpushak/CsharpUnProj/tree/7d357b4e79d29a5e293faec5bbd90852e5c73cee/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/4labaDiskMat/tochno3labaDisk)

using System;

namespace Completeness

{

public class BooleanFunction

{

public int[] Values { get; private set; }

public BooleanFunction(int size)

{

Values = new int[size];

}

public void SetValue(int index, int value)

{

Values[index] = value;

}

public bool IsT0() => Values == null || Values[0] == 0;

public bool IsT1() => Values == null || Values[Values.Length - 1] == 1;

public bool IsS()

{

if (Values == null) return true;

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

for (int i = 0; i < Values.Length / 2; i++)

{

if (Values[i] == Values[Values.Length - 1 - i])

{

return false;

}

}

return true;

}

public bool IsM()

{

if (Values == null || Values.Length == 1) return true;

for (int i = 0; i < Values.Length / 2; i++)

{

if (Values[i] == 1 && Values[i] > Values[i + Values.Length / 2]) return false;

}

int[] func1 = new int[Values.Length / 2];

int[] func2 = new int[Values.Length / 2];

for (int i = 0; i < Values.Length / 2; i++)

{

func1[i] = Values[i];

func2[i] = Values[func2.Length + i];

}

return new BooleanFunction(func1.Length) { Values = func1 }.IsM() && new BooleanFunction(func2.Length) { Values = func2 }.IsM();

}

public bool IsL()

{

if (Values.Length == 1 || Values.Length == 2) return true;

int[] c = new int[Values.Length];

c[0] = Values[0];

c[1] = Operation(c[0], Values[1]);

c[2] = Operation(c[0], Values[2]);

if (Values.Length == 4)

{

c[3] = Operation(Operation(Operation(c[0], c[1]), c[2]), Values[3]);

return c[3] == 0;

}

else

{

c[3] = Operation(c[0], Values[4]);

c[4] = Operation(Operation(Operation(c[0], c[2]), c[1]), Values[3]);

c[5] = Operation(Operation(Operation(c[0], c[3]), c[1]), Values[5]);

c[6] = Operation(Operation(Operation(c[0], c[3]), c[2]), Values[6]);

c[7] = Operation(Operation(Operation(Operation(Operation(Operation(Operation(c[0], c[3]), c[2]), c[1]), c[6]), c[5]), c[4]), Values[7]);

return c[4] == 0 && c[5] == 0 && c[6] == 0 && c[7] == 0;

}

}

private int Operation(int i, int j) => i == j ? 0 : 1;

}

internal class Program

{

static int GetNumberInt(string str)

{

int number;

string inputNumber;

bool isRightType;

do

{

Console.Write(str);

inputNumber = Console.ReadLine();

isRightType = Int32.TryParse(inputNumber, out number);

if (!isRightType)

Console.Write("Неверный ввод данных. Пожалуйста, попробуйте ещё раз.\n");

}

while (!isRightType);

return number;

}

static int GetNumberIntVector(string str)

{

int number;

string inputNumber;

bool isRightType;

do

{

Console.Write(str);

inputNumber = Console.ReadLine();

isRightType = Int32.TryParse(inputNumber, out number);

if (!isRightType || number < 0 || number > 1)

Console.Write("Неверный ввод данных. Пожалуйста, попробуйте ещё раз.\n");

}

while (!isRightType || number < 0 || number > 1);

return number;

}

static void ShowTable(BooleanFunction[] functions)

{

Console.WriteLine("Принадлежность классам");

Console.WriteLine(",----,----,----,---,---,---,");

Console.WriteLine("| | T0 | T1 | S | M | L |");

Console.WriteLine("|----|----|----|---|---|---|");

foreach (var func in functions)

{

Console.WriteLine($"| f{Array.IndexOf(functions, func) + 1} | {ConvertSymbol(func.IsT0())} | {ConvertSymbol(func.IsT1())} | " +

$"{ConvertSymbol(func.IsS())} | {ConvertSymbol(func.IsM())} | {ConvertSymbol(func.IsL())} |");

Console.WriteLine("|----|----|----|---|---|---|");

}

Console.WriteLine("'----'----'----'---'---'---'");

GetAnswer(functions);

}

static void GetAnswer(BooleanFunction[] functions)

{

bool t0 = functions.All(f => f.IsT0());

bool t1 = functions.All(f => f.IsT1());

bool s = functions.All(f => f.IsS());

bool m = functions.All(f => f.IsM());

bool l = functions.All(f => f.IsL());

if (t0 || t1 || s || m || l) Console.WriteLine("Система неполная");

else Console.WriteLine("Система полная");

}

static char ConvertSymbol(bool f) => f ? '+' : ' ';

static void Main(string[] args)

{

int amountFunctions;

do

{

amountFunctions = GetNumberInt("Введите количество функций: ");

if (amountFunctions < 1 || amountFunctions > 3)

{

Console.WriteLine("Функций может быть не больше трёх и не меньше 1!");

}

}

while (amountFunctions < 1 || amountFunctions > 3);

BooleanFunction[] functions = new BooleanFunction[amountFunctions];

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

{

int size;

do

{

size = GetNumberInt("Введите размер функции: ");

if (size != 1 && size != 2 && size != 4 && size != 8)

{

Console.WriteLine("Размер функции должен быть равен 2^i!");

}

}

while (size != 1 && size != 2 && size != 4 && size != 8);

functions[i] = new BooleanFunction(size);

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

{

functions[i].SetValue(j, GetNumberIntVector($"{j + 1} элемент: "));

}

}

ShowTable(functions);

}

}

}