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 систем функций»

Выполнила работу

студентка группы РИС-22-1Б

Вышенская Е.И.

Проверила

Старший преподаватель кафедры ИТАС

Рустамханова Г. И

Пермь, 2023

**Постановка задачи**

Необходимо реализовать программу, принимающую на вход до трёх функций в виде векторов на 2, 4 или 8 значений). На выходе функция должна определить принадлежность каждой функции одному из следующих классов: моннотонная, самодвойственная, линейная, сохраняющая ноль, сохраняющая единицу) и определяющая полноту системы.

**Анализ задачи**

Для определения классов функций создан класс FunctionClasses, который обрабатывает массив из 1, 2, 4 и 8 значений. У класса определены следующие поля:

* *int[] function:* хранит вектор значений,
* *int values:* количество значений, определено для удобства чтения кода,
* *int vars:* количество переменных, участвующих в функции,
* *int[,] truthTable:* заполненная таблица истинности.

Для определения классов функции определены следующие методы класса:

* *isPreserving0:* проверяет принадлежит ли функция классу сохраняющих 0. Булева функция сохраняет константу 0 (принадлежит классу T0), если на наборе из всех нулей функция принимает значение ноль. Необходимо проверить первое значение вектора на равенство нулю, поскольку это значение соответствует нулевым переменным.
* *isPreserving1:* проверяет принадлежит ли функция классу сохраняющих 1.  Булева функция сохраняет константу 1 (принадлежит классу T1), если на наборе из всех единиц функция принимает значение единица. Необходимо проверить последнее значение вектора на равенство единице, поскольку это значение соответствует единичным переменным.
* *isMonotous:* проверяет принадлежит ли функция классу монотонных. Булева функция f(x1, …, xn) называется монотонной (принадлежит классу M), если для любой пары наборов α и β таких, что αIMG_256β, выполняется условие f(α)≤ f(β).
* *isSelfDual:* проверяет принадлежит ли функция классу самодвойственных. Булева функция f(x1, …, xn) самодвойственна (принадлежит классу S), если она равна двойственной себе функции, то есть

f(x1, …, xn) = f\*(x1, …, xn)= f (x 1, …, x n).

* *isLinear:* проверяет принадлежит ли функция классу линейных.Булева функция называется линейной (принадлежит классу L), если ее полином Жегалкина линеен. Определение линейности полинома Жегалкина происходит с помощью метода Паскаля (треугольника).

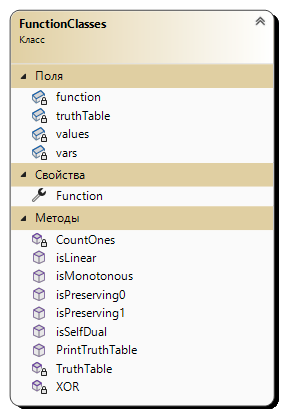
*Метод треугольника*

1. Метод треугольника (часто называемый методом треугольника Паскаля) позволяет преобразовать таблицу истинности в полином Жегалкина путём построения вспомогательной треугольной таблицы в соответствии со следующими правилами:
2. Строится полная таблица истинности, в которой строки идут в порядке возрастания двоичных кодов от 000…00 до 111…11.
3. Строится вспомогательная треугольная таблица, в которой первый столбец совпадает со столбцом значений функции в таблице истинности.
4. Ячейка в каждом последующем столбце получается путём суммирования по модулю 2 двух ячеек предыдущего столбца — стоящей в той же строке и строкой ниже.
5. Столбцы вспомогательной таблицы нумеруются двоичными кодами в том же порядке, что и строки таблицы истинности.
6. Каждому двоичному коду ставится в соответствие один из членов полинома Жегалкина в зависимости от позиций кода, в которых стоят единицы. Например, ячейке 111 соответствует член ABC, ячейке 101 — член AC, ячейке 010 — член B, ячейке 000 — член 1 и т. д.
7. Если в верхней строке какого-либо столбца стоит единица, то соответствующий член присутствует в полиноме Жегалкина.

Также определены вспомогательные функции:

* XOR: сложение по модулю 2.
* CountOnes: подсчёт количества единиц в строке таблице истинности.
* TruthTable: заполнение таблицы истинности.
* PrintTruthTable: печать таблицы истинности в зависимости от количества переменных.

**UML класса**



**Тесты**

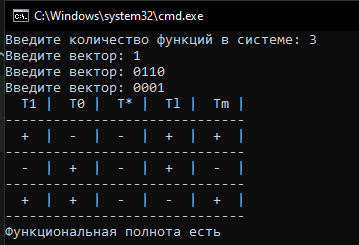


Рисунок 1 - Пример полной системы из трёх функций

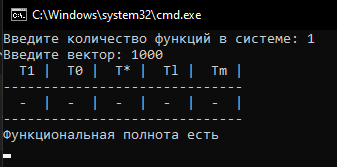


Рисунок 2 - Пример полной системы из одной функции

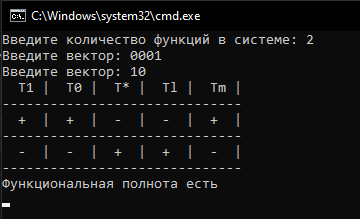


Рисунок 3 - Пример полной системы из двух функций

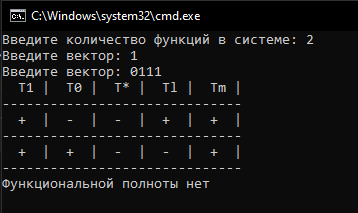


Рисунок 4 - Пример неполной системы из двух функций

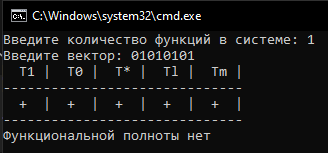


Рисунок 5 = Пример неполной системы из одной функции

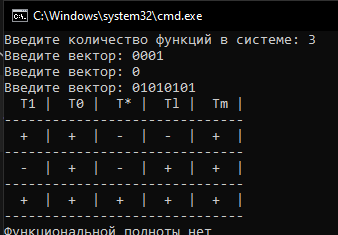


Рисунок 6 - Пример неполной системы из трёх функций

**Код программы**

*FunctionClasses.cs*

using System;

namespace FunctionClasses

{

internal class FunctionClasses

{

private int[] function;

private int values;

private int vars;

private int[,] truthTable;

public int[] Function

{

get => function;

set

{

function = value;

if (function.Length == 1)

{

int[] temp = new int[2];

temp[0] = function[0];

temp[1] = function[0];

function = temp;

}

values = function.Length;

vars = (int)Math.Log(values, 2);

truthTable = new int[values, vars];

TruthTable();

}

}

private void TruthTable()

{

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

{

string res = Convert.ToString(i, 2);

for (int j = 0; j < (vars - res.Length); ++j)

{

truthTable[i, j] = 0;

}

int counter = 0;

for (int j = (vars - res.Length); j < vars; j++)

{

truthTable[i, j] = int.Parse(res[counter].ToString());

++counter;

}

}

}

public void PrintTruthTable()

{

Console.WriteLine("\n-> Таблица истинности");

if (vars == 2)

{

Console.WriteLine(" x | y | f");

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

}

else if (vars == 3)

{

Console.WriteLine(" x | y | z | f");

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

}

else

{

Console.WriteLine(" x | y | z | w | f");

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

}

string result = "";

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

{

result = " ";

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

{

result += truthTable[i, j] + " ";

}

result = result.Substring(0, result.Length - 2);

result += "| " + Function[i];

Console.WriteLine(result);

}

Console.WriteLine();

}

public bool isPreserving0()

{

if (Function[0] == 0)

return true;

return false;

}

public bool isPreserving1()

{

if (Function[values-1] == 1)

return true;

return false;

}

public bool isMonotonous()

{

int previous;

int current;

string ones;

if (vars == 1)

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

for (int var = 2; var <= 4; var++)

{

previous = 0;

ones = "";

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

{

ones += Function[i];

if ((i + 1) % (values / var) == 0)

{

current = Convert.ToInt32(ones, 2);

if (previous > current)

return false;

previous = current;

ones = "";

}

}

}

return true;

}

public bool isSelfDual()

{

for (int i = 0; i < values / 2; ++i)

{

if (Function[values - i - 1] == Function[i])

return false;

}

return true;

}

private int CountOnes(int i)

{

int count = 0;

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

{

if (truthTable[i, j] == 1)

count++;

}

return count;

}

public bool isLinear()

{

int A, B;

int[,] matrix = new int[values, values];

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

{

matrix[0, i] = Function[i];

}

int count;

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

{

count = 0;

for (int j = 0; count < values-1; ++j)

{

A = matrix[i - 1, count];

count++;

B = matrix[i - 1, count];

matrix[i, j] = XOR(A, B);

}

}

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

{

if (matrix[i, 0] == 1)

{

if (CountOnes(i) > 1)

return false;

}

}

return true;

}

private int XOR(int a, int b)

{

if (a == b)

return 0;

return 1;

}

}

}

*Program.cs*

using System;

namespace FunctionClasses

{

internal class Program

{

static void Main(string[] args)

{

while (true)

{

Console.Clear();

Menu();

Console.ReadKey(true);

}

}

static void Menu()

{

int choice = InputInt("Введите количество функций в системе: ");

CheckBoundaries(ref choice, 1, 3);

switch (choice)

{

case 1:

{

FunctionClasses func1 = GetFunction();

PrintTable();

PrintClasses(func1);

OneFunctionPlenitude(func1);

break;

}

case 2:

{

FunctionClasses func1 = GetFunction();

FunctionClasses func2 = GetFunction();

PrintTable();

PrintClasses(func1);

PrintClasses(func2);

TwoFunctionPlenitude(func1, func2);

break;

}

case 3:

{

FunctionClasses func1 = GetFunction();

FunctionClasses func2 = GetFunction();

FunctionClasses func3 = GetFunction();

PrintTable();

PrintClasses(func1);

PrintClasses(func2);

PrintClasses(func3);

ThreeFunctionPlenitude(func1, func2, func3);

break;

}

}

}

static void ThreeFunctionPlenitude(FunctionClasses func1, FunctionClasses func2, FunctionClasses func3)

{

bool res = (!func1.isPreserving0() || !func2.isPreserving0() || !func3.isPreserving0())

&& (!func1.isPreserving1() || !func2.isPreserving1() || !func3.isPreserving1())

&& (!func1.isLinear() || !func2.isLinear() || !func3.isLinear())

&& (!func1.isSelfDual() || !func2.isSelfDual() || !func3.isSelfDual())

&& (!func1.isMonotonous() || !func2.isMonotonous() || !func3.isMonotonous())

;

if (res)

Console.WriteLine("Функциональная полнота есть");

else

Console.WriteLine("Функциональной полноты нет");

}

static void TwoFunctionPlenitude(FunctionClasses func1, FunctionClasses func2)

{

bool res = (!func1.isPreserving0() || !func2.isPreserving0())

&& (!func1.isPreserving1() || !func2.isPreserving1())

&& (!func1.isLinear() || !func2.isLinear())

&& (!func1.isSelfDual() || !func2.isSelfDual())

&& (!func1.isMonotonous()|| !func2.isMonotonous())

;

if (res)

Console.WriteLine("Функциональная полнота есть");

else

Console.WriteLine("Функциональной полноты нет");

}

static void OneFunctionPlenitude(FunctionClasses func)

{

bool res = !func.isPreserving0() && !func.isPreserving1()

&& !func.isLinear() && !func.isSelfDual()

&& !func.isMonotonous();

if (res)

Console.WriteLine("Функциональная полнота есть");

else

Console.WriteLine("Функциональной полноты нет");

}

static void PrintTable()

{

Console.Write("{0, -5}|", " T1");

Console.Write("{0, -5}|", " T0");

Console.Write("{0, -5}|", " T\*");

Console.Write("{0, -5}|", " Tl");

Console.Write("{0, -5}|", " Tm");

Console.Write("\n------------------------------\n");

}

static void PrintClasses(FunctionClasses func)

{

if (!func.isPreserving1())

Console.Write("{0, -5}|", " -");

else

Console.Write("{0, -5}|", " +");

if (!func.isPreserving0())

Console.Write("{0, -5}|", " -");

else

Console.Write("{0, -5}|", " +");

if (!func.isSelfDual())

Console.Write("{0, -5}|", " -");

else

Console.Write("{0, -5}|", " +");

if (!func.isLinear())

Console.Write("{0, -5}|", " -");

else

Console.Write("{0, -5}|", " +");

if (!func.isMonotonous())

Console.Write("{0, -5}|", " -");

else

Console.Write("{0, -5}|", " +");

Console.Write("\n------------------------------\n");

}

static FunctionClasses GetFunction()

{

Console.Write("Введите вектор: ");

string input = Console.ReadLine();

FunctionClasses func = new FunctionClasses();

func.Function = ReadVector(input);

return func;

}

static public int InputInt(string msg = "")

{

bool isNumber;

string inputLine;

int res;

Console.Write(msg);

do

{

inputLine = Console.ReadLine();

isNumber = int.TryParse(inputLine, out res);

if (!isNumber)

Console.Write("Ошибка! Введите число\n" + msg);

} while (!isNumber);

return res;

}

static int[] ReadVector(string input)

{

int[] vec = new int[input.Length];

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

{

vec[i] = int.Parse(input[i].ToString());

}

return vec;

}

static public void CheckBoundaries(ref int value, int left = 0, int right = int.MaxValue)

{

if (left > right)

(left, right) = (right, left);

while ((value < left) || (value > right))

value = InputInt($"Число не входит в границы[{left}, {right}]. Попробуйте снова: ");

}

}

}