Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Электротехнический факультет   
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе №4**

**“Система функций”**

Дисциплина: «Дискретная математика и математическая логика»

Выполнил   
студент группы ИВТ-23-2б  
Злыгостев Д.Н.

Проверил   
доцент кафедры ИТАС  
Рустамханова Г. И.

Пермь 2024

Оглавление

[1. Введение 3](#_Toc183817122)

[2. Теоретическая часть 3](#_Toc183817123)

[3. Задачи 3](#_Toc183817124)

[4. Исходный код 4](#_Toc183817125)

[5. Теоретические сведения 8](#_Toc183817126)

[6. Описание программы 9](#_Toc183817127)

[7. Структура 10](#_Toc183817128)

[8. Git-Hub 10](#_Toc183817129)

[9. Пример работы программы 10](#_Toc183817130)

[10. Заключение 11](#_Toc183817131)

1. Введение

Программа предназначена для анализа логических функций, представленных в виде векторов значений. Она позволяет проверить различные свойства функций, такие как монотонность, линейность, самодвойственность и другие. Также программа выводит таблицу, в которой отображаются результаты проверки для нескольких функций и определяет, является ли система функций полной. Программа реализует анализ функций на основе их представлений в виде векторов и с использованием стандартных методов работы с логическими функциями.

2. Теоретическая часть

Целью данной программы является анализ заданных пользователем логических функций по следующим критериям:

* Сохранение нуля (T0)
* Сохранение единицы (T1)
* Самодвойственность (S)
* Монотонность (M)
* Линейность (L)

Программа выводит результаты проверки для каждой функции и определяет, является ли система функций полной.

3. Задачи

Для реализации программы необходимо решить несколько ключевых задач:

1. Запросить у пользователя количество функций и длину векторов.
2. Ввести векторы логических функций, заданных пользователем.
3. Реализовать функции для проверки свойств логических функций:
   * Сохранение нуля (T0).
   * Сохранение единицы (T1).
   * Самодвойственность (S).
   * Монотонность (M).
   * Линейность (L).
4. Для каждой функции вывести таблицу с результатами.
5. Определить, является ли система функций полной.

4. Исходный код

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <locale>

#include <iomanip> // Для форматирования вывода

using namespace std;

// Функция для подсчета количества единичных битов

int popcount(int x) {

int count = 0;

while (x) {

count += x & 1;

x >>= 1;

}

return count;

}

// Проверка на T0 (сохраняет 0)

bool isT0(const vector<int>& func) {

return func[0] == 0; // Первый элемент вектора

}

// Проверка на T1 (сохраняет 1)

bool isT1(const vector<int>& func) {

return func.back() == 1; // Последний элемент вектора

}

// Проверка на S (самодвойственность)

bool isS(const vector<int>& func) {

int n = func.size();

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

if (func[i] == func[n - 1 - i]) return false;

}

return true;

}

// Проверка на M (монотонность)

bool isM(const vector<int>& func) {

int n = func.size();

for (int i = 0; i < n; ++i) {

for (int j = i + 1; j < n; ++j) {

if ((i & j) == i && func[i] > func[j]) return false;

}

}

return true;

}

// Проверка на L (линейность)

bool isL(const vector<int>& func) {

int n = func.size();

vector<int> coeffs = func;

// Применяем преобразование Жегалкина

for (int i = 0; i < n; ++i) {

for (int j = 0; j < i; ++j) {

if ((i & j) == j) {

coeffs[i] ^= coeffs[j];

}

}

}

// Проверяем, что все старшие коэффициенты равны 0

for (int i = 1; i < n; ++i) {

if (popcount(i) > 1 && coeffs[i] != 0) {

return false;

}

}

return true;

}

// Функция для вывода таблицы

void printTable(const vector<vector<int>>& functions,

const vector<string>& results) {

cout << left << setw(10) << "Функция"

<< setw(5) << "T0"

<< setw(5) << "T1"

<< setw(5) << "S"

<< setw(5) << "M"

<< setw(5) << "L" << endl;

cout << string(35, '-') << endl;

for (size\_t i = 0; i < functions.size(); ++i) {

cout << left << setw(10) << ("f" + to\_string(i + 1));

for (char c : results[i]) {

cout << setw(5) << (c == '1' ? "+" : " ");

}

cout << endl;

}

}

int main() {

setlocale(LC\_ALL, "RU");

int n, vectorLength;

cout << "Введите количество функций: ";

cin >> n;

cout << "Введите длину вектора (4 или 8): ";

cin >> vectorLength;

if (vectorLength != 4 && vectorLength != 8) {

cout << "Некорректная длина вектора! Допустимы только 4 или 8." << endl;

return 1;

}

cin.ignore(); // Очищаем буфер ввода

vector<vector<int>> functions(n, vector<int>(vectorLength));

vector<string> results(n, "00000");

for (int i = 0; i < n; ++i) {

cout << "Введите вектор " << i + 1

<< " через пробел (" << vectorLength << " цифр): ";

string input;

getline(cin, input);

stringstream ss(input);

for (int j = 0; j < vectorLength; ++j) {

ss >> functions[i][j];

}

}

// Проверяем свойства функций

for (int i = 0; i < n; ++i) {

if (isT0(functions[i])) results[i][0] = '1';

if (isT1(functions[i])) results[i][1] = '1';

if (isS(functions[i])) results[i][2] = '1';

if (isM(functions[i])) results[i][3] = '1';

if (isL(functions[i])) results[i][4] = '1';

}

// Вывод таблицы

printTable(functions, results);

// Проверяем полноту системы

bool isIncomplete = false; // Флаг, указывающий на неполноту

for (int i = 0; i < 5; ++i) {

bool fullColumn = true;

for (const auto& result : results) {

if (result[i] == '0') {

fullColumn = false;

break;

}

}

if (fullColumn) {

isIncomplete = true;

break;

}

}

cout << "Система функций " << (isIncomplete ? "неполная" : "полная") << "." << endl;

return 0;

}

5. Теоретические сведения

Логическая функция может быть представлена в виде вектора значений, где каждый элемент вектора принимает значение 0 или 1, соответствующее значению функции для соответствующего входного вектора. В теории логических функций существуют различные свойства, которые можно проверить на основе такого представления. Рассмотрим несколько основных свойств:

**Сохранение нуля (T0)** – функция сохраняет ноль, если при входе 0 функция всегда возвращает 0.

**Сохранение единицы (T1)** – функция сохраняет единицу, если при входе 1 функция всегда возвращает 1.

**Самодвойственность (S)** – функция называется самодвойственной, если её значение при входе x противоположно значению при входе инвертированного x.

**Монотонность (M)** – функция монотонна, если для любых двух входов x и y, где x меньше y, значение функции для x не больше, чем для y.

**Линейность (L)** – функция линейна, если она может быть представлена как полином Жегалкина, где коэффициенты старших степеней равны нулю.

6. Описание программы

Программа реализует следующие этапы работы:

1. **Ввод данных**: Пользователь вводит количество функций и длину вектора (доступные длины — 4 и 8). Далее вводятся сами функции в виде последовательности битов.
2. **Проверка свойств**: Для каждой функции выполняются проверки на соблюдение свойств:
   * isT0: Проверяет, сохраняет ли функция 0.
   * isT1: Проверяет, сохраняет ли функция 1.
   * isS: Проверяет самодвойственность функции.
   * isM: Проверяет монотонность функции.
   * isL: Проверяет линейность функции.
3. **Вывод таблицы**: Результаты проверки для каждой функции выводятся в виде таблицы, где для каждой строки отображаются соответствующие свойства.
4. **Проверка полноты системы**: После проверки всех функций программа анализирует, являются ли все свойства (T0, T1, S, M, L) представленными в таблице хотя бы один раз. Если для какого-то свойства в системе не хватает хотя бы одной функции, система считается неполной.

* Временная сложность операций:
  + Пересечение: O(n + m), где n и m — размеры множеств.
  + Объединение: O(n + m).
  + Разность: O(n + m).
  + Симметрическая разность: O(n + m).
  + Дополнение: O(U), где U — размер универсума.

7. Структура

Программа состоит из нескольких основных частей:

1. **Функции проверки свойств**:
   * isT0: Проверка на сохранение 0.
   * isT1: Проверка на сохранение 1.
   * isS: Проверка на самодвойственность.
   * isM: Проверка на монотонность.
   * isL: Проверка на линейность.
2. **Функции вспомогательного характера**:
   * popcount: Функция для подсчета количества единичных битов в числе.
   * printTable: Функция для вывода таблицы с результатами анализа логических функций.
3. **Основная функция (**main**)**:
   * Запрашивает количество функций и длину векторов.
   * Вводит функции.
   * Выполняет проверки и выводит результаты.
   * Проверяет полноту системы.

8. Git-Hub

Ссылка на репозиторий:

[ZligostevDenis/Diskretka\_Math](https://github.com/ZligostevDenis/Diskretka_Math)

9. Пример работы программы

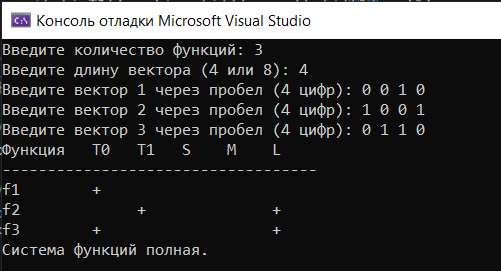


Рисунок – “Вывод”

10. Заключение

Программа эффективно решает задачу анализа логических функций на предмет их свойств и полноты системы. Это позволяет с помощью нескольких простых шагов проверить функции на важнейшие логические характеристики.