**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Дисциплина: Теория цифровых автоматов**

**Тема: Диагностика неисправностей комбинационных схем**

**с одним выходом**

Выполнил: ст. группы ВТ-31

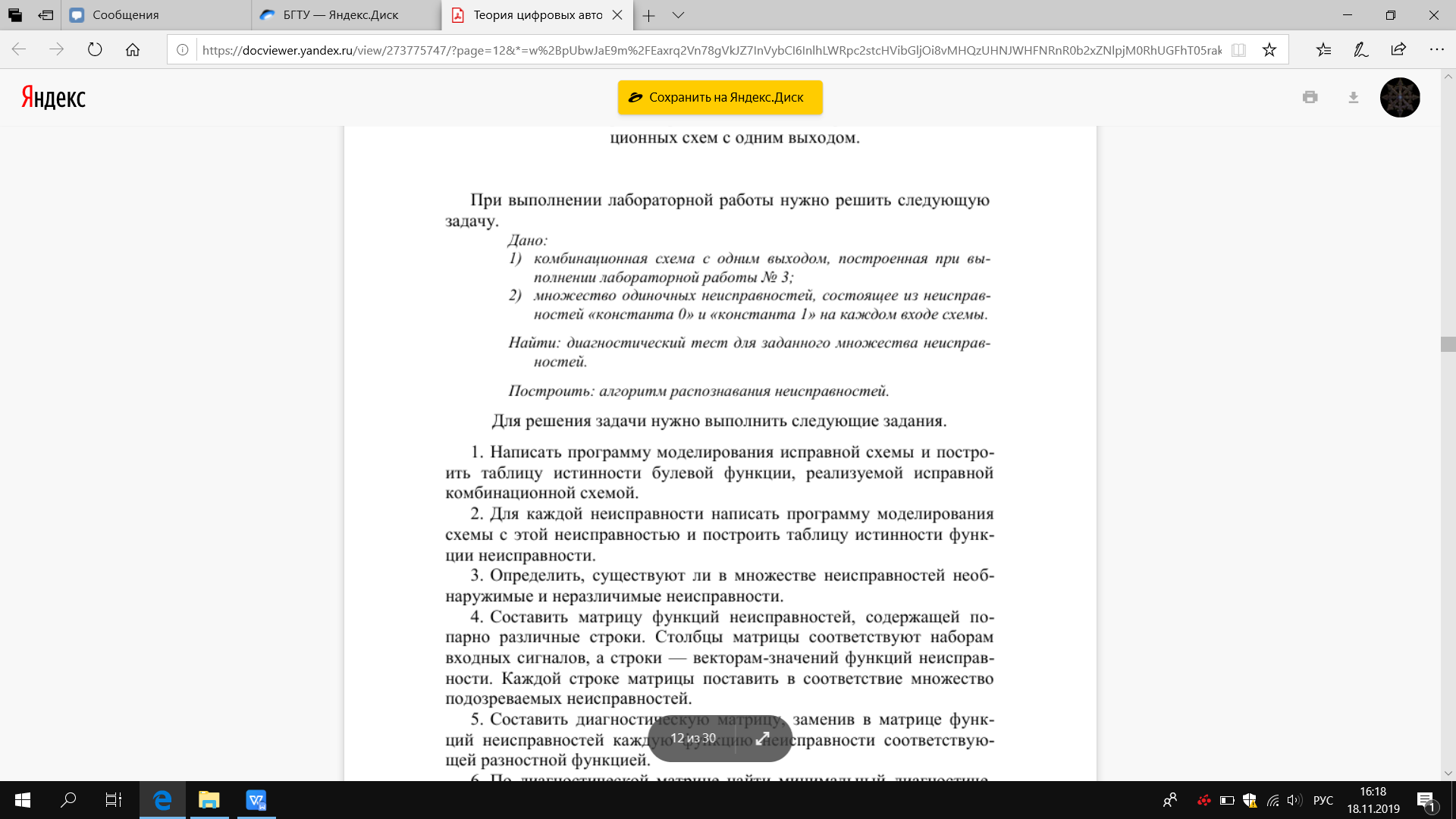
Подкопаев Антон Валерьевич

Проверил: Рязанов Юрий Дмитриевич

**Белгород 2019**

**Цель работы:** научиться строить диагностические тесты и алгоритмы распознавания неисправностей комбинационных схем с одним выходом.

**Вариант 13**

****

**Ход выполнения работы**

1. Написать программу моделирования исправной схемы и построить таблицу истинности булевой функции, реализуемой исправной комбинационной схемой.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 | x12 | x13 | x14 | x15 | x16 | x17 | x18 | x19 | x20 | x21 | x22 | x23 | x24 | x25 | x26 | x27 | x28 | x29 | x30 | x31 | x32 |
| x5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| x4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| x3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| x2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*см. Приложение*

1. Для каждой неисправности написать программу моделирования схемы с этой неисправностью и построить таблицу истинности функции неисправности.

*см. Приложение*

1. Определить, существуют ли в множестве неисправностей необнаружимые и неразличимые неисправности.

Необнаружимыми и неразличимыми являются неисправности в f, f1 и f2, так как x1 не влияет на набор результатов конечной функции.

1. Составить матрицу функций неисправностей, содержащей попарно различные строки. Столбцы матрицы соответствуют наборам входных сигналов, а строки — векторам-значений функций неисправности. Каждой строке матрицы поставить в соответствие множество подозреваемых неисправностей.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 | x12 | x13 | x14 | x15 | x16 | x17 | x18 | x19 | x20 | x21 | x22 | x23 | x24 | x25 | x26 | x27 | x28 | x29 | x30 | x31 | x32 |
| f | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| f(x1,0) 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| f(x1,1) 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| f(x2,0) 3 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| f(x2,1) 4 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| f(x3,0) 5 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| f(x3,1) 6 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| f(x4,0) 7 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| f(x4,1) 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| f(x5,0) 9 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| f(x5,1) 10 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

1. Составить диагностическую матрицу, заменив в матрице функций неисправностей каждую функцию неисправности соответствующей разностной функцией.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 | x12 | x13 | x14 | x15 | x16 | x17 | x18 | x19 | x20 | x21 | x22 | x23 | x24 | x25 | x26 | x27 | x28 | x29 | x30 | x31 | x32 |
| R1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | **1** | 0 |
| R4 | 0 | **1** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| R6 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R7 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | **1** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| R8 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | **1** | 0 | 0 |
| R9 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R10 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 |

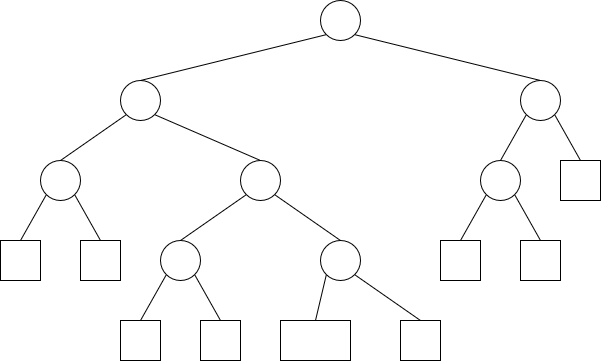
1. По диагностической матрице найти минимальный диагностический тест.

В диагностический тест войдут x2, x19, x30, x31, а также x6, x7, x11 и x26

1. В матрице функций неисправностей оставить только столбцы, соответствующие наборам входных сигналов, принадлежащим диагностическому тесту.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x2 | x19 | x30 | x31 | x6 | x7 | x11 | x26 |
| f | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| f(x1,0) 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| f(x1,1) 2 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| f(x2,0) 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| f(x2,1) 4 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| f(x3,0) 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| f(x3,1) 6 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| f(x4,0) 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| f(x4,1) 8 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| f(x5,0) 9 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| f(x5,1) 10 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

1. По полученной в матрице построить алгоритм распознавания неисправностей в виде диагностического дерева.



1. Используя программу моделирования комбинационной схемы с неисправностью и алгоритм распознавания неисправностей написать программу для проведения диагностического эксперимента.

*Приложение*

#include <iostream>

#include <vector>

using namespace std;

bool NextSet(int\* a, int n, int m) {

int j = m - 1;

while (j >= 0 && a[j] == n) j--;

if (j < 0) return false;

if (a[j] >= n)

j--;

a[j]++;

if (j == m - 1) return true;

for (int k = j + 1; k < m; k++)

a[k] = 0;

return true;

}

void Print(int\* a, int n) {

static int num = 1;

cout.width(3);

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

cout << a[i] << " ";

}

void vectPrint(bool\* vect, int n) {

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

printf("%d ", vect[i]);

}

void vectXOR(bool\* vect, bool\* vect2, int n) {

bool vectRes[32];

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

vectRes[i] = vect[i] xor vect2[i];

vectPrint(vectRes, n);

}

void printMatr1 (int matr[11][32]) {

for (int i = 0; i < 11; i++) {

for (int j = 0; j < 32; j++)

printf("%i ", matr[i][j]);

printf("\n");

}

}

void printMatr2(int matr[10][32]) {

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 32; j++)

printf("%i ", matr[i][j]);

printf("\n");

}

}

void XORMatr(int matr[11][32], int xorm[10][32]) {

for (int i = 1; i < 11; i++) {

for (int j = 0; j < 32; j++)

xorm[i-1][j] = matr[0][j] xor matr[i][j];

}

}

int treeCheck(int x[32]) {

if (x[1] == 1)

if (x[6] == 1)

return 4;

else

if (x[10] == 0)

return 9;

else

return 6;

else

if (x[25] == 0)

if (x[29] == 0)

return 8;

else

return 3;

else

if (x[30] == 0)

if (x[5] == 0)

return 5;

else

return 10;

else

if (x[18] == 1)

return 7;

else

return 0;

}

bool gun(int\* a, int n, int matr[11][32], int count, int x[32]) {

Print(a, n);

bool x1 = a[0], x2 = a[1], x3 = a[2], x4 = a[3], x5 = a[4];

bool z1, z2, u1, u2, u3, u4, u5, u6, v1, v2, v3, v4, f;

bool not\_x1, not\_x2, not\_x3, not\_x4, not\_x5, not\_z1, not\_z2, not\_u1, not\_u2, not\_v1, not\_v2, not\_v3, not\_v4, not\_f;

bool zero = 0, one = 1;

bool save;

// исходная модель часично определенной будевой функцииции определенная через ДНФ

u2 = x2 and !x4;

u3 = x3 and !x4;

u4 = !x4 and !x5;

z1 = x2 and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[0][count] = f;

printf(f ? " 1 " : " 0 ");

x[count] = f;

// ДНФ x1 = 0

u2 = x2 and !x4;

u3 = x3 and !x4;

u4 = !x4 and !x5;

z1 = x2 and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[1][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x1 = 1

u2 = x2 and !x4;

u3 = x3 and !x4;

u4 = !x4 and !x5;

z1 = x2 and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[2][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x2 = 0

u2 = zero and !x4;

u3 = x3 and !x4;

u4 = !x4 and !x5;

z1 = zero and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[3][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x2 = 1

u2 = one and !x4;

u3 = x3 and !x4;

u4 = !x4 and !x5;

z1 = one and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[4][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x3 = 0

u2 = x2 and !x4;

u3 = zero and !x4;

u4 = !x4 and !x5;

z1 = x2 and !x5;

u1 = zero and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[5][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x3 = 1

u2 = x2 and !x4;

u3 = one and !x4;

u4 = !x4 and !x5;

z1 = x2 and !x5;

u1 = one and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[6][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x4 = 0

u2 = x2 and !zero;

u3 = x3 and !zero;

u4 = !zero and !x5;

z1 = x2 and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[7][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x4 = 1

u2 = x2 and !one;

u3 = x3 and !one;

u4 = !one and !x5;

z1 = x2 and !x5;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[8][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x5 = 0

u2 = x2 and !x4;

u3 = x3 and !x4;

u4 = !x4 and !zero;

z1 = x2 and !zero;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[9][count] = f;

// printf(f ? " 1 " : " 0 ");

// ДНФ x5 = 1

u2 = x2 and !x4;

u3 = x3 and !x4;

u4 = !x4 and !one;

z1 = x2 and !one;

u1 = x3 and z1;

v1 = u2 or u3;

v2 = u1 or u4;

f = v1 or v2;

matr[10][count] = f;

// printf(f ? " 1 " : " 0 ");

printf("\n");

return (f);

}

int main() {

vector<vector<int>\*>\* list = new vector<vector<int>\*>();

int\* tmp;

int matr[11][32];

int xorm[10][32];

int count = 0;

int n = 1, m = 5, \* a = new int[m];

for (int i = 0; i < m; i++) a[i] = 0;

int x[32];

int p;

int i = 0;

gun(a, m, matr, count, x);

while (NextSet(a, n, m)) {

count++;

gun(a, m, matr, count, x);

}

printf("\n");

printMatr1(matr);

printf("\n");

XORMatr(matr, xorm);

printMatr2(xorm);

p = treeCheck(x);

printf("\n");

if (p == 0)

printf("no errors found");

else

printf("error number %i", p);

return 0;

}