Отчёт по дискретной математике

"Булева алгебра"

Оглавление

[Задание 3](#_Toc437964567)

[1 Определение булевой функции 4](#_Toc437964568)

[2 Построение СДНФ 4](#_Toc437964569)

[3 Вычисление значений булевой функции 7](#_Toc437964570)

[Заключение 8](#_Toc437964572)

Задание…………………………………………………………………………………………………………………………………………...…….9

[Приложение 1. Диаграммы решений 10](#_Toc437964574)

[Приложение 2. Построение СДНФ 1](#_Toc437964575)1

# Задание

В результате выполнения данной работы были выполнены следующие задачи:

1. Построение по таблице истинности дерева решений и минимальной бинарной диаграммы решений.
2. Реализация программного вычисления значений булевой функции по бинарной диаграмме решений.
3. Программное построение СДНФ по таблице истинности, а также вычисление с ее помощью значений булевой функции.

# 1 Определение булевой функции

Булевой функцией от *n* переменных называется любая функция, в которой аргументы и функция могут принимать значения *0* либо *1*.

Исходная булева функция, заданная с помощью таблицы истинности:

|  |  |
| --- | --- |
| f | x1x2x3x4 |
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 1 | 0 0 1 0 |
| 1 | 0 0 1 1 |
| 0 | 0 1 0 0 |
| 0 | 0 1 0 1 |
| 0 | 0 1 1 0 |
| 1 | 0 1 1 1 |
| 1 | 1 0 0 0 |
| 0 | 1 0 0 1 |
| 0 | 1 0 1 0 |
| 1 | 1 0 1 1 |
| 1 | 1 1 0 0 |
| 1 | 1 1 0 1 |
| 0 | 1 1 1 0 |
| 1 | 1 1 1 1 |

Дерево решений данной булевой функции, а также упрощенная диаграмма решений представлены в Приложении 1 (Рис.2 и Рис.3)

# 2 Построение СДНФ

Для любой функции алгебры логики существует своя СДНФ(совершенная дизъюнктивная нормальная форма), причём единственная.

Для того чтобы получить СДНФ функции, требуется составить её таблицу истинности. Это реализуется следующим способом:

cout << "Enter the values of your function - choose value with up/down arrows, invert it with right arrow, press ENTER when ready" << endl;

for(int i = 0; i < 16; i++){

short int temp = i;

for(int j = 0; j < 4; j++){

table[i][j] = (i >> 3-j) & 1;

cout << table[i][j] << " ";

}

cout << "- " << table[i][4] << endl;

}

bool correct = 0;

char comm;

int selection = 0;

gotoxy(10,selection+2);

cout << "|" << table[selection][4] << "|";

int maxSelection = 15;

while(!correct){

comm = getch();

switch(comm){

case UP\_ARROW:

if(selection > 0){

gotoxy(10,selection+2);

cout << " " << table[selection][4] << " ";

selection--;

gotoxy(10,selection+2);

cout << "|" << table[selection][4] << "|";

}

break;

case DOWN\_ARROW:

if(selection < maxSelection){

gotoxy(10,selection+2);

cout << " " << table[selection][4] << " ";

selection++;

gotoxy(10,selection+2);

cout << "|" << table[selection][4] << "|";

}

break;

case RIGHT\_ARROW:

table[selection][4] = !table[selection][4];

gotoxy(11,selection+2);

cout << table[selection][4];

break;

case ENTER:

correct = 1;

break;

}

}

Пользователь может с помощью стрелок вверх/вниз выбирать нужную строку, с помощью правой и левой стрелки можно инвертировать значение функции. После заполнения значений пользователь должен нажать ENTER для завершения редактирования функции (Рис.1).

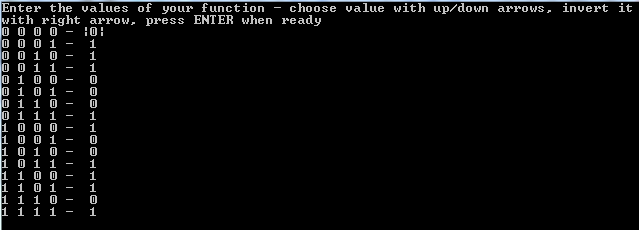


Рис.1 Ввод значений булевой функции

Построение СДНФ происходит по следующему правилу: в цикле, проходящем по значениям функции, выбираются лишь те комбинации, которые дают единицу по таблице истинности. Далее рассматриваются значения переменных, при которых функция равна *1.* Если значение переменной равно *0,* то она записывается с инверсией (в данном случае с "-"). Если значение переменной равно *1*, то без инверсии.

cout << endl << "SDNF" << endl;

int count = 0;

for(int i = 0; i < 16; i++){

if(table[i][4] != 0){

count++;

cout << "(";

for(int j = 0; j < 4; j++){

cout << (table[i][j] == 1?"x":"-x") << j+1 << (j == 3?"":"^");

}

cout << ")";

if(i != 15){

if(count % 4 == 0){

cout << " v " << endl;

}

cout << " v ";

}

}

}

Если все значения функции равны нулю, то автоматически выводится сообщение о том, что СДНФ равна 0.

Значения заданной булевой функции, а также построение требуемой СДНФ приведены в Приложении 2.

# 3 Вычисление значений булевой функции

Для того чтобы вычислить значение функции следует ввести четырехразрядное бинарное число, способ ввода аналогичен заполнению значений функции.

cout << "Chose variable with left/right arrows, invert with up/down, ENTER to calculate, q to exit." << endl;

cout << "|0| 0 0 0 ";

bool n[4] = {0};

bool res = 0;

while(!correct){

comm = getch();

switch(comm){

case LEFT\_ARROW:

if(selection > 0){

gotoxy(selection\*4,24+newLine);

cout << " " << n[selection] << " ";

selection--;

gotoxy(selection\*4,24+newLine);

cout << "|" << n[selection] << "|";

}

break;

case RIGHT\_ARROW:

if(selection < maxSelection){

gotoxy(selection\*4,24+newLine);

cout << " " << n[selection] << " ";

selection++;

gotoxy(selection\*4,24+newLine);

cout << "|" << n[selection] << "|";

}

break;

case UP\_ARROW:

n[selection] = !n[selection];

gotoxy(selection\*4 + 1,24+newLine);

cout << n[selection];

break;

case ENTER:

res = CalcSDNF(n[0],n[1],n[2],n[3],table);

gotoxy(0,25+newLine);

cout << "Result: " << res;

break;

case 'q':

correct = 1;

break;

}

}

После ввода введённые значения подставляются в СДНФ, далее происходит вычисление результата и вывод его в консоль.

bool CalcSDNF(bool a, bool b, bool c, bool d, bool table[16][5]){

bool SDNFrez = 0;

for(int i = 0; i < 16; i++){

bool elemRez = 0;

if(table[i][4] != 0){

elemRez = (table[i][0] == 1?a:!a) && (table[i][1] == 1?b:!b) && (table[i][2] == 1?c:!c) && (table[i][3] == 1?d:!d);

SDNFrez = SDNFrez || elemRez;

}

}

return SDNFrez;

}

# Заключение

В результате выполнения работы было построено дерево решений, минимальная бинарная диаграмма решений и СДНФ по таблице истинности для функции от четырех аргументов. Программа способна вычислять значения функции по СДНФ, подставляя в неё 4 переменные, введённые пользователем.

Возможные усовершенствования:

* использование динамического массива для записи значений булевой функции, чтобы предоставить пользователю возможность работать с любым кол-вом аргументов.
* Реализовать программное построение дерева решений для визуального представления функции. Так же можно реализовать программное построение сокращённой диаграммы решений, однако, эта задача не является тривиальной, потребуется научить программу интерпретировать различные сокращения для корректного построения диаграммы.

# Список литературы

1. Новиков
2. C++

# Приложение 1. Диаграммы решений

Рис.2 Дерево решений

Рис.3 Компактная бинарная диаграмма решений

# Приложение 2. Построение СДНФ

В данном приложении показан пример программного построения СДНФ по таблице истинности

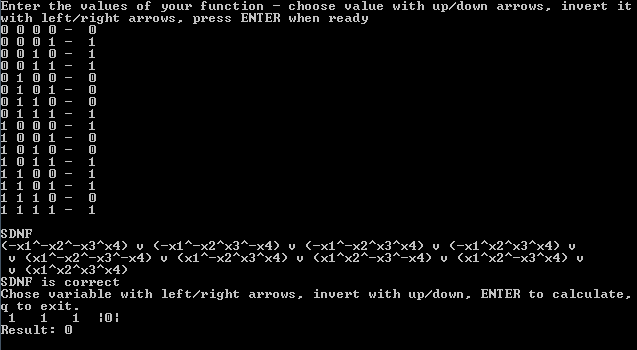


Рис. 4 Таблица истинности вместе с построенной по ней СДНФ