МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности
Высшая школа технологий искусственного интеллекта
Направление 02.03.01 Математика и компьютерные науки

Отчет по лабораторной работе №2

Построение значений булевой функции по бинарной диаграмме решений. Построение совершенной дизъюнктивной нормальной формы, соршенной конъюнктивной нормальной формы и полинома Жегалкина по таблице истинности.

Обучающийся: _____ Гладков И.А.

Руководитель: _____ Востров А.В.

«_____»_____20____г.

Санкт-Петербург, 2024

Содержание

Введение

Номер варианта – 20. Исходный вектор значений – (0011110011000011). В данной лабораторной работе необходимо реализовать хранение бинарной диаграммы решений (БДР) функции четырех переменных, заданной вектором значений, в соответствии с вариантом. Реализовать генерацию по таблице истинности СДНФ, СКНФ и полинома Жегалкина. Такжже нужно , чтобы была возможноть вычисления значения по пользовательскому вводу через БДР, СДНФ и полиномом Жегалкгина.

1 Математическое описание

В этом разделе рассмотрены понятия и алгоритмы, которые лежат в основе реализации лабораторной работы.

1.1 Булева функция

Булева функция (или логическая функция, или функция алгебры логики) от n аргументов - в дискретной математике - отображение $B^n \to B$, где $B = \{0, 1\}$ булево множество.

Каждая булева функция арности n полностью определяется заданием своих значений на своей области определения, то есть на всех булевых векторах длины n. Число таких векторов равно 2^n.

В данной работе $n = 4$, а значит количество векторов булевой функции будет равно $2^n = 16$. Ниже, в таблице 1 представлена таблица истинности, построенная по вектору-столбцу (0011110011000011).

x	y	z	e	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0

x	y	z	e	f
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Таблица 1. Таблица истинности булевой функции.

Как можно заметить по таблице истинности, от изменения перемнной x=0 на x=1, значения функции меняются, следовательно переменная x – существенная.

1.2 Бинарная диаграмма решений

Бинарная диаграмма решений (БДР) или "программа с ветвлением"является формой представления булевой функции $f(x_1, x_2, \ldots, x_n)$ от n переменных в виде направленного ациклического графа, состоящего из внутренних узлов решений (помеченных x_i), каждый из которых имеет по два потомка, и двух терминальных узлов (помеченных 0 и 1), каждый из которых соответствует одному из двух значений булевой функции.

На рисунке 1 изображено бинарное дерево принятия решений (без применения правил сокращения), соответствующее приведенной на этом же рисунке таблице истинности для булевой функции $f(x_1, x_2, x_3)$. Для заданных входных значений x_1, x_2, x_3 можно определить значение булевой функции, двигаясь по дереву от корневого узла дерева к терминальным узлам, выбирая направле-

ние перехода из узла x_i в зависимости от входного значения x_i. Пунктирными линиями на рисунке изображены переходы к младшему потомку, а непрерывными линиями изображены переходы к старшему потомку. Например, если заданы входные значения ($x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0$), то из корневого узла x_1 необходимо перейти по пунктирной линии влево (так как значение x_1 равно 0), после этого необходимо перейти по непрерывным линиям вправо два раза (так как значения x_2 и x_3 равны 1) и после перейти влево по пунктирной линии (так как значение x_4 равно 0). В результате мы окажемся в 0-терминальном узле, то есть значение $f(x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0)$ равно 0.

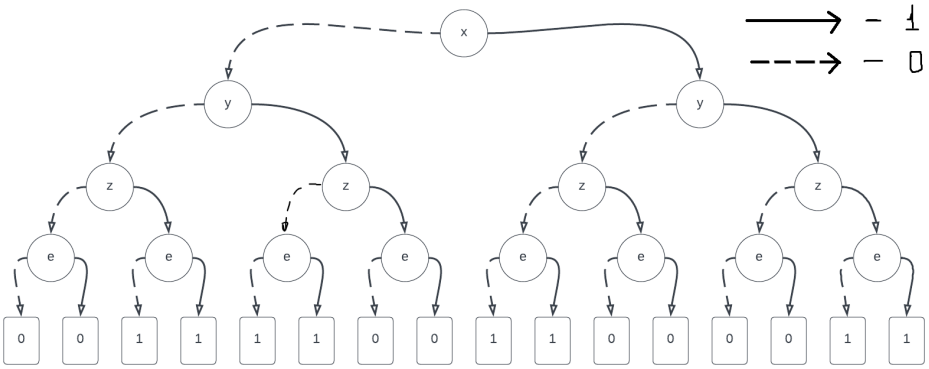

Рис. 1. Семантическое дерево.

Семантическое дерево на рисунке 1 можно преобразовать в бинарную диаграмму решений путём применения двух правил сокращения:

1. Слияние любых изоморфных подграфов.
2. Удаление всех узлов с изоморфными потомками.

Получившаяся БДР изображена на рисунке 2.

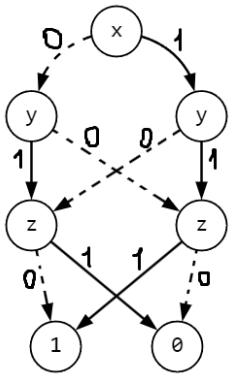

Рис. 2. Бинарная диаграмма решений.

1.3 СКНФ

Совершенная конъюнктивная нормальная форма (СКНФ) - одна из форм представления функции алгебры логики (булевой функции) в виде логического выражения. Представляет собой частный случай КНФ, удовлетворяющий следующим трём условиям:

1. в ней нет одинаковых множителей (элементарных дизъюнкций);
2. в каждом множителе нет повторяющихся переменных;
3. каждый множитель содержит все переменные, от которых зависит булева функция (каждая переменная может входить в множитель либо в прямой, либо в инверсной форме).

Любая булева формула, не являющаяся тождественно истинной, может быть приведена к СКНФ. Также всякая булева функция имеет единственную совершенную конъюктивную нормальную форму (СКНФ).

$$f(x_1, ..., x_n) = \bigwedge_{\{\sigma_1,...,\sigma_n\}|f^*\{\sigma_1,...,\sigma_n\}\}} x_1^{\sigma_1} \wedge ... \wedge x_n^{\sigma_n}$$

СКНФ для функции 4-х переменных заданной вектором значений (0011110011000011) будет следующей: СКНФf $= (x \vee y \vee z \vee e) \wedge (x \vee y \vee z \vee \overline{e}) \wedge (x \vee \overline{y} \vee \overline{z} \vee e) \wedge (x \vee \overline{y} \vee \overline{z} \vee \overline{e}) \wedge (\overline{x} \vee y \vee \overline{z} \vee e) \wedge (\overline{x} \vee y \vee \overline{z} \vee \overline{e}) \wedge (\overline{x} \vee \overline{y} \vee z \vee e) \wedge (\overline{x} \vee \overline{y} \vee z \vee \overline{e})$

1.4 СДНФ

Совершенная дизъюнктивная нормальная форма (СДНФ) - одна из форм представления функции алгебры логики (булевой функции) в виде логического выражения. Представляет собой частный случай ДНФ, удовлетворяющий следующим трём условиям:

1. в ней нет одинаковых слагаемых (элементарных конъюнкций);
2. в каждом слагаемом нет повторяющихся переменных;
3. к каждое слагаемое содержит все переменные, от которых зависит булева функция (каждая переменная может входить в слагаемое либо в прямой, либо в инверсной форме).

Любая булева формула, не являющаяся тождественно ложной, может быть приведена к СДНФ, причём единственным образом, то есть для любой выполнимой функции алгебры логики существует своя СДНФ, причём единственная.

$$f(x_1, ..., x_n) = \bigvee_{\{\sigma_1,...,\sigma_n\}|f\{\sigma_1,...,\sigma_n\}\}} x_1^{\sigma_1} \vee ... \vee x_n^{\sigma_n}$$

СДНФ для функции 4-х переменных заданной вектором значений (0011110011000011) будет следующей:

СДНФf$= \bar{x}y\bar{z}e \vee \bar{x}y\bar{z}\bar{e} \vee \bar{x}\bar{y}ze \vee \bar{z}\bar{y}z\bar{e} \vee x\bar{y}\bar{z}\bar{e} \vee x\bar{y}\bar{z}e \vee xyz\bar{e} \vee xyze$

1.5 Полином Жегалкина

Полином Жегалкина - многочлен над полем Z_2, то есть полином с коэффициентами вида 0 и 1 , где в качестве произведения берётся конъюнкция, а в качестве сложения - исключающее или. Полином был предложен в 1927 году Иваном Жегалкиным в качестве удобного средства для представления функций булевой логики.

Ормально полином Жегалкина можно представить в следующем виде:

$$P(x_1, ..., x_n) = \alpha_0 a_1 \oplus \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus ... \oplus \alpha_{2^n-1} x_1 x_2 ... x_n$$

Полином Жегалкина для функции 4-х переменных заданной вектором значений (0011110011000011) будет следующим:

$$\mathrm{Pf} = x \oplus y \oplus z$$

1.6 Дерево решений

На рисунке 3 изображено дерево решений для функции 4 -х переменных заданной вектором значений (0011110011000011), с порядком переменных xyze.

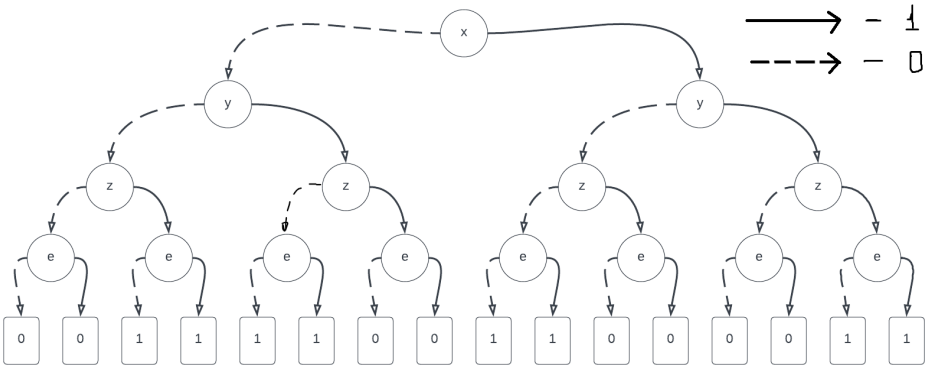

Рис. 3. Бинарная диаграмма решений.

1.7 Бинарная диаграмма решений

На рисунке 4 изображена бинарная диаграмма решений для функции 4 -х переменных заданной вектором значений (0011110011000011), с порядком переменных xyze.

1.8 Синтаксическое дерево

На рисунке 5 изображено синтаксическое дерево, построенное по минимизированной формуле $xyz \vee x\bar{y}\bar{z} \vee \bar{x}y\bar{z} \vee \bar{x}\bar{y}z$, полученной путём сокращения СДНФ для функции 4 -х переменных заданной вектором значений (0011110011000011).

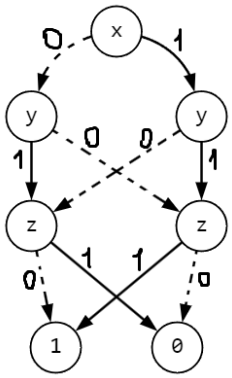

Рис. 4. Бинарная диаграмма решений (сплошная линия - для 1 , пунктирная для 0).

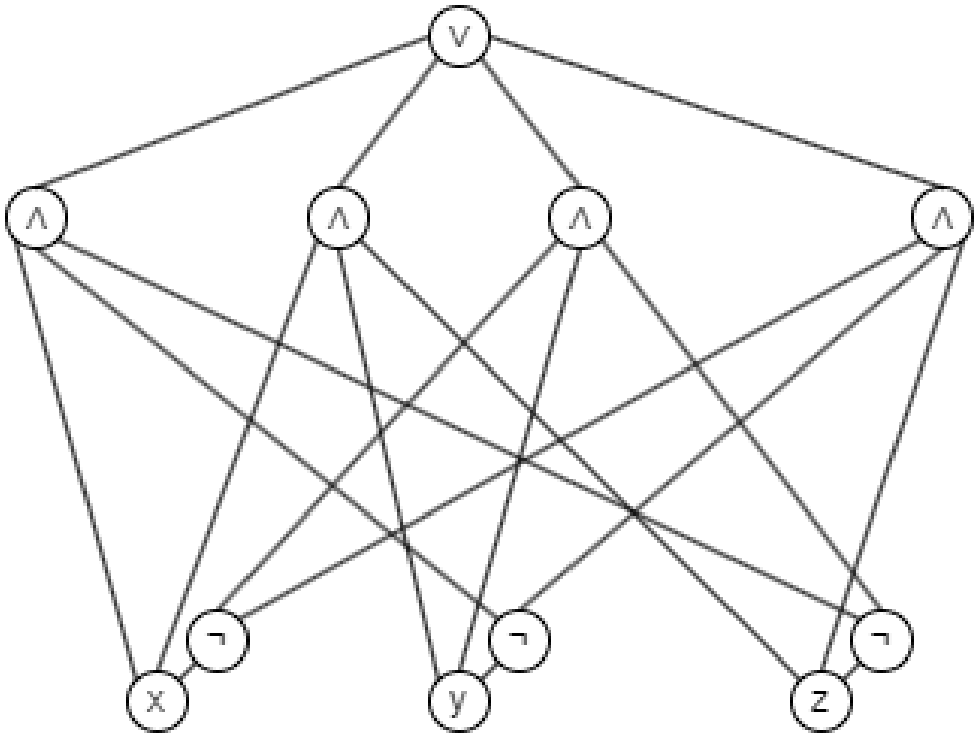

Рис. 5. Синтаксическое дерево.

2 Особенности реализации

2.1 Структура data

Структура создана для помощи хранения Бинарной диаграммы решений в виде бинарного дерева.

Она содержит следующие переменные:

1. char name – переменная для хранения названия
2. data* False – указатель на структуру: отвлетление – 0.
3. data* True; указатель на структуру: отвлетление – 1.

2.2 Класс Tabl

Класс Tabl – основной класс в данной лабораторной работе. Он отвечает за хранение и доступ к таблице истинности, СДНФ, СКНФ, БДР и полиному Жегалкина.

В классе присутствуют переменные:

1. vector<bool> x – массив отвечающий за столбец x в таблице истинности.
2. vector<bool> y – массив отвечающий за столбец y в таблице истинности.
3. vector<bool> z – массив отвечающий за столбец z в таблице истинности.
4. vector<bool> e – массив отвечающий за столбец e в таблице истинности.
5. vector<bool> f – массив отвечающий за столбец f в таблице истинности.
6. data* BDR – указатель на объект структуры, необходим, для хранения БДР: верхушку бинарного дерева.
7. string SDNF – строка содержащая СДНФ.
8. string SKNF – строка содержащая СКНФ.
9. vector <vector<bool» Zhegalkin – двумерный массив, хранящий полином жегалкина в виде таблицы.
10. vector <int> Alpha_index – массив хранящий индексы α_i в определенном порядке.
11. string Zhegalkin_str –строка содержащая полином жигалкина в виде выражения.

2.2.1 Конструктор Tabl

Вход: создание таблицы истинности, СДНФ, СКНФ, БДР и полинома Жегалкина

Выход: сформированные таблица истинности, СДНФ, СКНФ, БДР и полином Жегалкина.

Вначале составляется таблица истинности с помощью цикла for. В столбец x, то есть в массив x, ставится "0"если i<8, иначе "1". В массив y ставится "1 если i кратна либо 4, либо 5, либо 6, либо 7, иначе "0". В столбец z ставится "1 когда i кратна либо 2, либо 3, в другох случаях "0". В e ставится "1 если i– нечетное, иначе "0".

Далее создаются вершины дерева БДР, и соединяются друг с другом, с помощью указалей в стрктуре data, в соответсвии с БДР начерченной на листке.

После с помощью цикла for заполняются строки СДНФ и СКНФ, опираясь на данные таблицы истинности.

Следующим шагом заполняется двумерный массив полинома жигалкина в виде таблицы, тоже с помощью цикла for. Также заполняется массив коэффициэнтов α_i в специальном порядке, которые соответствует формуле: $\mathrm{Pf} = \alpha_0 \oplus \alpha_1 x \oplus \alpha_2 y \oplus \alpha_3 z \oplus \alpha_4 e \oplus \alpha_5 xy \oplus \alpha_6 xz \oplus \alpha_7 xe \oplus \alpha_8 yz \oplus \alpha_9 ye \oplus \alpha_{10} ze \oplus \alpha_{11} xyz \oplus \alpha_{12} xye \oplus \alpha_{13} xze \oplus \alpha_{14} yze \oplus \alpha_1 5 xyze$. После по данной таблице строится полином жегалкина в виде выражения.

Код контруктора ниже, в листинге 1.

```
1  Tabl::Tabl() {
2    x.resize(16, 0);
3    y.resize(16, 0);
```

```
    z.resize(16, 0);
    e.resize(16, 0);
    f.resize(16, 0);

    int kx = 0, ky = 0, kz = 0, ke = 0;
    for (int i = 0; i < 16; i++) {

      x[i] = i < 8 ? 0 : 1;

      y[i] = (i & 4) == 4  (i & 5) == 5  (i & 6) == 6  (i & 7) == 7 ? 1 : 0;

      z[i] = (i & 2) == 2  (i & 3) == 3 ? 1 : 0;

      e[i] = i & 1 ? 1 : 0;

      f[i] = i == 2  i == 3  i == 4  i == 5  i == 8  i == 9  i == 14  i == 15 ? 1 : 0;
    }

    BDR = new data('x');

    data* BDR_y0 = new data('y');
    data* BDR_y1 = new data('y');
    BDR->AddFalse(BDR_y0);
    BDR->AddTrue(BDR_y1);

    data* BDR_z0 = new data('z');
    data* BDR_z1 = new data('z');

    BDR_y0->AddFalse(BDR_z1);
    BDR_y0->AddTrue(BDR_z0);

    BDR_y1->AddFalse(BDR_z0);
    BDR_y1->AddTrue(BDR_z1);

    data* BDR_0 = new data('0');
    data* BDR_1 = new data('1');

    BDR_z0->AddFalse(BDR_1);
    BDR_z0->AddTrue(BDR_0);

    BDR_z1->AddFalse(BDR_0);
    BDR_z1->AddTrue(BDR_1);

    SDNF = "СДНФ = ";
    SKNF = "СКНФ = ";
    for (int i = 0, k1=0,k2=0; i < 16; i++) {
      if (f[i] == 1) {
        SDNF += x[i] == 1 ? "x " : "!x ";
```

```
        SDNF += y[i] == 1 ? "y " : "!y ";
        SDNF += z[i] == 1 ? "z " : "!z ";
        SDNF += e[i] == 1 ? "e " : "!e ";
        SDNF += k1 < 7 ? " V  " : "\0";
        k1++;
      }
      else {
        SKNF += "(";
        SKNF += x[i] == 0 ? "x" : "!x";
        SKNF += " V ";
        SKNF += y[i] == 0 ? "y" : "!y";
        SKNF += " V ";
        SKNF += z[i] == 0 ? "z" : "!z";
        SKNF += " V ";
        SKNF += e[i] == 0 ? "e" : "!e";
        SKNF += ")";
        SDNF += k2 < 7 ? "" : "\0";
        k2++;
      }
    }

    for (int i = 16; i >0; i--) {
      std::vector <bool> buf(i);
      Zhegalkin.push_back(buf);
    }
    for (int i = 0; i < 16; i++) {
      Zhegalkin[0][i] = f[i];
    }
    for (int i = 1; i < 16; i++) {
      for (int j = 0; j < Zhegalkin[i].size(); j++) {
        Zhegalkin[i][j] = Zhegalkin[i - 1][j] == Zhegalkin[i - 1][j + 1] ? 0 : 1;
      }
    }

    Alpha_index.push_back(0);
    Alpha_index.push_back(4);
    Alpha_index.push_back(3);
    Alpha_index.push_back(10);
    Alpha_index.push_back(2);
    Alpha_index.push_back(9);
    Alpha_index.push_back(8);
    Alpha_index.push_back(14);
    Alpha_index.push_back(1);
    Alpha_index.push_back(7);
    Alpha_index.push_back(6);
    Alpha_index.push_back(13);
    Alpha_index.push_back(5);
    Alpha_index.push_back(12);
```

```
100     Alpha_index.push_back(11);
101     Alpha_index.push_back(15);
102 
103     Zhegalkin_str = "F = ";
104     for (int i = 0; i < 16; i++) {
105       if (Zhegalkin[i][0] == 1) {
106         Zhegalkin_str += "a_";
107         Zhegalkin_str += std::to_string(Alpha_index[i]);
108         Zhegalkin_str += " ";
109         Zhegalkin_str += x[i] == 1 ? "x" : "";
110         Zhegalkin_str += y[i] == 1 ? "y" : "";
111         Zhegalkin_str += z[i] == 1 ? "z" : "";
112         Zhegalkin_str += e[i] == 1 ? "e" : "";
113         Zhegalkin_str += i < 15 ? " + " : "\0";
114       }
115     }
116   }
```

Листинг 1. Конструктор Tabl

2.2.2 Метод Print

Вход: намерение отобразить таблицу истинности на консоль.

Выход: отображение таблицы истинности на консоли.

Метод Print выводит на экран таблицу истинности.

```
1  void Tabl::Print() {
2    std::cout << "  " << "x y z e \textbar f" << std::endl;
3    std::cout << "  -----------\n";
4 
5    for (int i = 0; i < 16; i++) {
6      std::cout << "  " << x[i] << " " << y[i] << " " << z[i] << " " << e[i] <<" \textbar "
       << f[i] << std::endl;
7    }
8  }
```

Листинг 2. Метод Print

2.2.3 Метод Print_SKNF

Вход: намерение отобразить СКНФ и на консоль.

Выход: отображение СКНФ на консоли.

Метод Print_SKNF выводит на экран СКНФ.

```
1     void Tabl::Print_SKNF() {
2       std::cout << std::endl<<SKNF << std::endl;
3     }
```

Листинг 3. Метод Print_SKNF

2.2.4 Метод Print_SDNF

Вход: намерение отобразить СДНФ и на консоль.

Выход: отображение СДНФ на консоли.

Метод Print_SDNF выводит на экран СДНФ.

```
void Tabl::Print_SDNF() {
  std::cout << std::endl<<SDNF << std::endl;
}
```

Листинг 4. Метод Print_SDNF

2.2.5 Метод Print_Zhegalkin

Вход: намерение отобразить полином Жегалкина и на консоль.

Выход: отображение полинома Жегалкина на консоли.

Метод Print_Zhegalkin выводит на экран полином Жегалкина, то есть строит таблицу, и ниже пишет сам полином.

```
void Tabl::Print_Zhegalkin() {
  std::string s = " ";
  std::printf(" Полином Жегалкина:\n");
  std::printf(" -----------------\n");

  for (int i = 0; i < 16; i++) {
    std::cout << "a_" << Alpha_index[i] <<'\t' << s;
    s += " ";
    for (int j = 0; j < Zhegalkin[i].size(); j++) {
      std::cout << Zhegalkin[i][j] << " ";
    }
    std::printf("\n");
  }
  std::cout <<std::endl<< Zhegalkin_str<<std::endl;
}
```

Листинг 5. Метод Print_Zhegalkin

2.2.6 Метод Search_SDNF

Вход: булева функция, которую вводит пользователь.

Выход: вывод на консоль значения булевой функции.

В методе Search_SDNF происходит подставление элементов булевой функции в строчку СДНФ, затем подсчет выражение, и получение результата.

```
void Tabl::Search_SDNF(std::string s) {
  std::string s_out = "";

  for (int i = 0; i < 4; i++) {
    s_out += s[i] == '0' ? "1" : "0";
  }

  std::string buf="";

  for (int i = 0; SDNF[i] != '\0'; i++) {
    buf += SDNF[i] == 'x' ? SDNF[i - 1] == '!' ? std::string(1, s_out[0]) : std::string
  (1, s[0]) : "";
    buf += SDNF[i] == 'y' ? SDNF[i - 1] == '!' ? std::string(1, s_out[1]) : std::string
  (1, s[1]) : "";
    buf += SDNF[i] == 'e' ? SDNF[i - 1] == '!' ? std::string(1, s_out[3]) : std::string
  (1, s[3]) : "";
    buf += SDNF[i] == 'V' ? "+" : "";
    buf += SDNF[i + 1] == '\0' ? "+!\0" : "";
  }

  std::vector<int> result;

  for (int i = 0, k = 0; buf[i] != '!'; i += 5) {
    result.push_back(1);
    result[k] &= buf[i] == '1' ? 1 : 0;
    result[k] &= buf[i + 1] == '1' ? 1 : 0;
    result[k] &= buf[i + 2] == '1' ? 1 : 0;
    result[k] &= buf[i + 3] == '1' ? 1 : 0;
    k++;
  }
  int Result=0;
  for (int i = 0; i < result.size(); i++) {
    Result \textbar=  result[i];
  }
  std::cout << "\tОтвет: " << Result << std::endl;
}

```

Листинг 6. Метод Search_SDNF

2.2.7 Метод Search_Zhegalkin

;

Вход: булева функция, которую вводит пользователь.

Выход: вывод на консоль значения булевой функции.

В методе Search_Zhegalkin происходит подставление элементов булевой функции в строчку

полинома Жегалкина, затем подсчет выражение, и получение результата.

```
  void Tabl::Search_Zhegalkin(std::string s) {
    std::string buf = "";

    for (int i = 0; Zhegalkin_str[i] != '\0'; i++) {
      buf += Zhegalkin_str[i] == 'x' ? std::string(1,s[0]) : "";
      buf += Zhegalkin_str[i] == 'y' ? std::string(1,s[1]) : "";
      buf += Zhegalkin_str[i] == 'z' ? std::string(1,s[2]) : "";
      buf += Zhegalkin_str[i] == 'e' ? std::string(1,s[3]) : "";
      buf += Zhegalkin_str[i] == '+' ? "+" : "";
    }

    bool result=buf[0]=='0'?0:1;
    bool buffer= buf[2] == '0' ? 0 : 1;
    bool buffer1= buf[4] == '0' ? 0 : 1;
    result = result != buffer ? 1 : 0;
    result = result != buffer1 ? 1 : 0;

    std::cout << "\tОтвет: " << result << std::endl;
  }
```

Листинг 7. Метод Search_Zhegalkin

2.2.8 Метод Search

Вход: булева функция, которую вводит пользователь.

Выход: вывод на консоль значения булевой функции.

В методе Search происходит поиск в БДР решения по булевой функции, затем получнение результата, и вывод на консоль.

```
void Tabl::Search(std::string s) {

  data* buf = this->BDR;

  for (int i = 0; i < 3; i++) {
    if (s[i] == '0')
    buf = buf->False;
    if (s[i] == '1')
    buf = buf->True;
  }
  std::printf("\tОтвет: %c\n", buf->name);
}
```

Листинг 8. Метод Search

2.2.9 Метод Check

Вход: булева функция, которую вводит пользователь.

Выход: вывод булевой функции по таблице истинности.

В методе Check происходит поиск в таблице истинности решения по булевой функции, после происходит вывод данного результата. Это необходимо для автоматической проверки с таблицей истинности в других методах, где происходит поиск результата по БДР, СДНФ и полиному Жегалкина.

```
bool Tabl::Check(std::string s) {
  bool result = 0;
  bool s_0 = s[0] == '0' ? 0 : 1;
  bool s_1 = s[1] == '0' ? 0 : 1;
  bool s_2 = s[2] == '0' ? 0 : 1;
  bool s_3 = s[3] == '0' ? 0 : 1;

  for (int i = 0; i < 16; i++)
  if (x[i] == s_0)
  if (y[i] == s_1)
  if (z[i] == s_2)
  if (e[i] == s_3) {
    result = f[i];
    return result;;
  }
}
```

Листинг 9. Метод Check

2.2.10 Метод CheckInput_Menu

Вход: ожидание правильного ввода пользователя для работы в дальнейшем, а именно выбора действия в меню.

Выход: вывод возможный номер действия из меню действий.

В методе CheckInput_Menu происходит проверка на корректность пользовательского ввода. Данный метод при помощи цикла while заставляет пользователя вводить значение заново, пока оно не будет корректным. Для проверки используется диапазон от 0 до 7 вкючительно и try catch,если пользователь введет вместо цифру любой другой символ.

```
std::string Tabl::CheckInput_Menu() {
  std::string s;

  while (true) {
    std::cin >> s;

    try {
      if (stoi(s) >= 0 && stoi(s) <= 7) {
```

```
          break;
        }
        else {
          std::cout << "Ошибка ввода! Ведите число от 0 до 7\n";
        }
      }
      catch (const std::invalid_argument& e) {
        std::cout << "Ошибка ввода! Неверный формат числа\n";
      }
    }
    return s;
  }
```

Листинг 10. Метод CheckInput_Menu

2.2.11 Метод CheckInput

Вход: ожидание правильного ввода пользователя для работы в дальнейшем, а именно булевой функции.

Выход: булевая функция пригодная для использования.

В методе CheckInput происходит проверка на корректность пользовательского ввода. Данный метод при помощи цикла while заставляет пользователя вводить значение заново, пока оно не будет корректным. Для проверки используется регулярное выражение в котором необходимо вводить либо 0, либо 1, и всего цифр должно быть 4, иначе пользователю необходимо вводить булеву функции заново.

```
  std::string Tabl::CheckInput() {
    std::regex BDR("^[0-1]{4}$");
    std::string s;
    std::printf("\nВведите значения переменных x, y, z, e в однустрочку без пробелов (при
    мер: 1010)\n");
    while (true) {
      std::cin >> s;
      if (regex_match(s, BDR))
        break;
      else
        std::printf("Ошибка в вводе, попробуйте снова\n");
    }
    return s;
  }
```

Листинг 11. Метод CheckInput

2.2.12 Menu

Вход: ожидание вывести возможные действия работы программы пользователю.

Выход: выводится меню действий на консоль, позьзователь выбирает нужное для него действие.

В методе Menu происходит вывод на экран всех возможных действий. Пользователь выбирает нужное ему действие, при этом вызывается метод для проверки пользовательсктго ввода, дальше вызывается метот соответствующий выбору пользователя.

```
void Tabl::Menu() {
  while (true) {
    printf("\nВыберите что вы хотите:\n");
    printf(" [1]\tВывести на экран таблицу истинности\n");
    printf(" [2]\tПолином Жегалкина\n");
    printf(" [3]\tСДНФ\n");
    printf(" [4]\tСКНФ\n");
    printf(" [5]\tНайти значение по БДР\n");
    printf(" [6]\tНайти значение по СДНФ\n");
    printf(" [7]\tНайти значение по полиному Жегалкина\n");
    printf(" [0]\tВыход\n");

    std::string s;

    s = CheckInput_Menu();

    bool Out = 0;

    switch (stoi(s)) {
      case 0:
        Out = 1;
        break;

      case 1:
        Print();
        break;

      case 2:
        Print_Zhegalkin();
        break;

      case 3:
        Print_SDNF();
        break;

      case 4:
        Print_SKNF();
        break;

      case 5:
        s = CheckInput();
```

```
42            Search(s);
43            break;
44
45          case 6:
46            s = CheckInput();
47            Search_SDNF(s);
48            break;
49
50          case 7:
51            s = CheckInput();
52            Search_Zhegalkin(s);
53            break;
54        }
55        if (Out)
56          break;
57        system("pause");
58      }
59    }
```

Листинг 12. Метод Menu

3 Результаты работы программы

При запуске программы на консоль выводится главное меню с перечнем возможных действий, которые может выбрать пользователь (рисунок 6).

```
Выберите что вы хотите:
 [1]    Вывести на экран таблицу истинности
 [2]    Полином Жегалкина
 [3]    СДНФ
 [4]    СКНФ
 [5]    Найти значение по БДР
 [6]    Найти значение по СДНФ
 [7]    Найти значение по полиному Жегалкина
 [0]    Выход
```

Рис. 6. Главное меню программы

После нажатия клавиши «1», либо «2», либо «3», либо «4», на консоль выводится таблица истинности, полином Жегалкина, СДНФ, СКНФ соответственно (рисунки 7-10).

```
  x y z e | f
  ----------
  0 0 0 0 | 0
  0 0 0 1 | 0
  0 0 1 0 | 1
  0 0 1 1 | 1
  0 1 0 0 | 1
  0 1 0 1 | 1
  0 1 1 0 | 0
  0 1 1 1 | 0
  1 0 0 0 | 1
  1 0 0 1 | 1
  1 0 1 0 | 0
  1 0 1 1 | 0
  1 1 0 0 | 0
  1 1 0 1 | 0
  1 1 1 0 | 1
  1 1 1 1 | 1
Для продолжения нажмите любую клавишу . . .
```

Рис. 7. Таблица истинности.

```
 Полином Жегалкина:
 ------------------
a_0      0 0 1 1 1 1 0 0 1 1 0 0 0 0 1 1
a_4       0 1 0 0 0 1 0 1 0 1 0 0 0 1 0
a_3        1 1 0 0 1 1 1 1 1 1 0 0 1 1
a_10        0 1 0 1 0 0 0 0 0 1 0 1 0
a_2          1 1 1 1 0 0 0 0 1 1 1 1
a_9           0 0 0 1 0 0 0 1 0 0 0
a_8            0 0 1 1 0 0 1 1 0 0
a_14            0 1 0 1 0 1 0 1 0
a_1              1 1 1 1 1 1 1 1
a_7               0 0 0 0 0 0 0
a_6                0 0 0 0 0 0
a_13                0 0 0 0 0
a_5                  0 0 0 0
a_12                  0 0 0
a_11                   0 0
a_15                    0

F = a_3 z + a_2 y + a_1 x +
```

Рис. 8. Полином Жегалкина.

```
СДНФ = !x !y z !e  V  !x !y z e  V  !x y !z !e  V  !x y !z e  V
x !y !z !e  V  x !y !z e  V  x y z !e  V  x y z e
Для продолжения нажмите любую клавишу . . .
```

Рис. 9. СДНФ

```
СКНФ = (x V y V z V e)(x V y V z V !e)(x V !y V !z V e)(x V !y V
!z V !e)(!x V y V !z V e)(!x V y V !z V !e)(!x V !y V z V e)(!x V
 !y V z V !e)
Для продолжения нажмите любую клавишу . . .
```

Рис. 10. СКНФ

При нажатии клавиши «5», либо «6», либо «7», у пользователя запрашивается булева функция, после происходит вывод результата который производился поиском по БДР, СДНФ и полиному Жегалкина соответственно, а также автоматическая проверка значения по таблице истинности (рисунки 11-13).

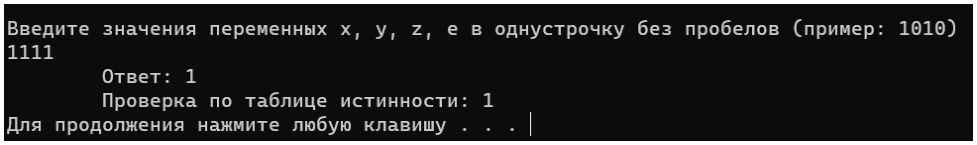

Рис. 11. Вывод значения булевой функции 1111 по БДР с проверкой по таблице истинности.

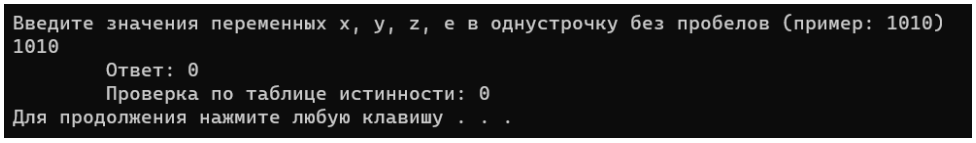

Рис. 12. Вывод значения булевой функции 1010 по СДНФ с проверкой по таблице истинности.

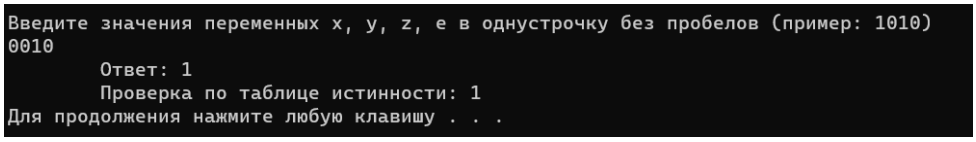

Рис. 13. Вывод значения булевой функции 0010 по полиному Жегалкина с проверкой по таблице истинности.

На рисунках 14-15 изображены возможные ошибки при некорректном пользовательском вводе.

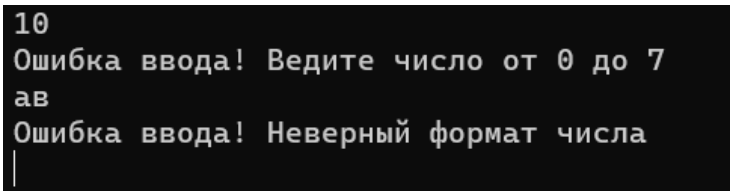

Рис. 14. Ввод номера действия, которого нет в меню и ввод не цифры, а любого другого символа.

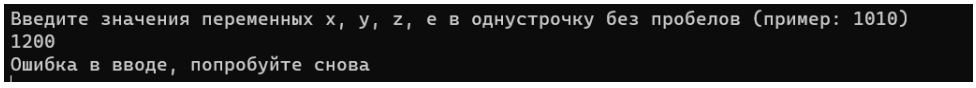

Рис. 15. Ввод некоректного булевой функции.

Заключение

В ходе выполнения лабораторной работы №2 была реализована программа, вычисляющая по заданному вектору-столбцу значений функции 4-х переменных СДНФ, СКНФ и полином Жегалкина. Программа также хранит бинарную диаграмму решений, вычисляет по СДНФ, БДР и полиному Жегалкина значение функции и сверяет его с таблицей истинности. В программе реализована защита от некорректного пользовательского ввода. Также отчёте представлены, построенные вручную, дерево решений, бинарная диаграмма решений и синтаксическое дерево.

Достоинством реализованной программы является объектно-ориентированный подход. Класс Tabl и структура data независимы от внешних параметров, их можно использовать в других программах и совмещать с другими классами.

Из недостатков, я считаю то, что эта реализация расчитана на функцию четырех переменных, при изменении количества переменных, придется достаточно много изменить в коде уже реализованной программы. можно считать, что это узконаправленная реализация. Также недостатком является то, что при вводе тождественно истинной или тождественно ложной функции, программа не будет генерировать строковое представление СКНФ. В случае ввода тождественно ложной функции не будет генерироваться строковое представление СДНФ.

Данную реализацию возможно расширять, без изменении основного функционала существующего кода. Например, можно реализовать построении СДНФ,СКНФ, полинома Жегалкина по введеному пользователем вектору-столбцу значений функции. Также возможно построения базиса Шеффера или базиса Пирса.Все это показывает о способности к расширении данной программы.

Лабораторная работа реализована на языке C++, в среде разработки Visual Studio 2022.

Источники

1. Дискретная математика для программистов. 3-е издание. Новиков Ф.А. СПБ:.Питер,2009.