Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин систем и сетей

Дисциплина: Основы алгоритмизации и программирования

УЧЕБНАЯ ПРАКТИКА
на тему
«ПРОГРАММА РЕАЛИЗАЦИИ ОПЕРАЦИИ * МЕТОДА МИНИМИЗАЦИИ БУЛЕВЫХ ФУНКЦИЙ РОТА»

Студент А.А. Грибовская

Руководитель Ю.А. Луцик

МИНСК 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Язык С является универсальный язык программирования. Он широко используется как в системном программировании, так и в разработке прикладного программного обеспечения. Язык С обладает простым и компактным синтаксисом, позволяющим разработчикам эффективно выражать свои идеи и реализовывать разнообразные задачи. Он обладает богатым набором функций и библиотек, предоставляющих мощные средства для работы с памятью, вводом-выводом, строками и другими основными конструкциями программ. Благодаря своей скорости и эффективности, язык С является популярным выбором для разработки системного и встроенного программного обеспечения, операционных систем, драйверов устройств и других приложений, где требуется высокая производительность и низкоуровневый доступ к ресурсам компьютера.

Язык С подходит для разработки программы реализации операции * метода минимизации булевых функций Рота. Алгоритм Рота (или алгоритм умножения минимизации) является одним из методов минимизации булевых функций. Он применяется для сокращения сложности булевых выражений путем комбинирования минтермов.

При перемножении двух минтермов происходит проверка соответствующих символов в строках. Если символы равны, они сохраняются в результате. Если символы отличаются, то, если хотя бы один из них равен 'x', в результате записывается символ, соответствующий непротиворечивому значению. Если оба символа отличны от 'x', в результате записывается символ 'y', обозначающий неопределенное значение.

После выполнения всех возможных комбинаций минтермов, происходит удаление повторяющихся и избыточных минтермов, что приводит к получению минимального набора минтермов, представляющих булевую функцию. В алгоритме Рота происходит последовательное перемножение пар минтермов, где каждый минтерм представляет булеву функцию в виде строки из символов '0', '1' и 'x'. Алгоритм Рота позволяет эффективно уменьшить количество переменных и логических операций в булевых выражениях, что упрощает их анализ и реализацию.

1 ОБЗОР ИСТОЧНИКОВ

1.1 Изучение задачи реализации операции * метода минимизации булевых функций Рота

Операция умножения кубов (*) в методе Рота используется для перемножения двух кубов, представленных в виде суммы квадратов. Она представляет собой формулу, которая позволяет свести перемножение суммы квадратов к произведению кубов.

Эта операция была придумана математиком Мишелем Ротом в 1930 году. Она является одним из основных примеров круговой алгебры и находит применение в решении различных задач, связанных с алгеброй и теорией чисел.

Например, операция умножения кубов может использоваться для быстрого вычисления квадратных корней и для решения некоторых диофантовых уравнений.

Благодаря своей эффективности в решении различных математических задач, операция умножения кубов занимает важное место в современной математике.

1.2 Постановка задачи

В программе будут реализованы функции:

- Умножения минтерм.
- Замены символа «y» на «x» в полученной минтерме.
- Удаление лишних минтерм.
- Вывод минимального набора минтерм.

Для реализации данного программного обеспечения используется язык программирования C.

2 ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ПЕРЕМЕННЫХ

В данном исходном коде используются следующие переменные:

- `num_vars` – количество переменных в каждой из строк;
- `amountRow1` – количество символа ‘x’ в первой строке;
- `amountRow2` – количество символа ‘x’ во второй строке;
- `amount` – количество несовпадающих символов в минтерме;
- `minterms2` – массив для хранения полученных минтерм;
- `minterms` – массив указателей на строки, содержащей изначальные минтермы.

Исходный код приведен в приложении А.

3 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Операция * применяется для нахождения наилучшего сочетания покрытий и оптимизации их числа, что позволяет снизить сложность цепей, реализующих логические функции

Сначала функция `enterAndCheckNumberString` запрашивает у пользователя количество минтерм и количество цифр в ней. Далее выпллняется проверка, содержит ли строка только цифры и имеет ли допустимую длину. В функциях ввода вызывется функция `setString`, которая отвечает за считывание символов из ввода до тех пор, пока не встретится новая строка или конец файла.

Далее вызывается функция `aToI`, которая служит для преобразования входных строк в целые числа (перевод цифровой последовательности типа string в число типа int) для дальнейшей работы в циклах.

Затем происходит ввод самих минтерм и вызов функции `enterAndCheckMinterms` для проверки, содержит ли минтерма только символы «0», «1» или «x» и имеет ли допустимую длину.

Далее вызывается функция `multiplicationInRoth` - это основная функция выполнения алгоритма умножения. В этой функции содержится цикл для выполнения операции умножения минтерм. В нем происходит проверка на пустые множества, которая использует функцию `checkY` (считает количество ‘y’ в минтерме). Затем происходит замена символа «y» на «x» в полученной минтерме. Далее переходим к циклу, который добавляет все оставшиеся минтермы (в том числе те, которые не образовали новых кубов) в окончательные минтермы. Затем происходит удаление лишних минтерм (повторяющихся и тех, которые образовали новые кубы). В конце выполняется цикл для вывода конечных результатов в консоль.

4 АЛГОРИТМ ПО ШАГАМ

1. Функция `multiplicationInRoth` принимает массив строк `minterms`, кол-во элементов `num_minterms` и кол-во переменных `num_vars` в функции.

2. Выделяется память для нового массива строк minterms2 с помощью функции `calloc`.

3. Создается переменная `size`.

4. Выводится строка "`Multiplication:`".

5. В цикле `for` перебираются все элементы массива `minterms`, начиная с первого.

6. Во вложенном цикле `for` перебираются все элементы массива `minterms`, начиная со следующего элемента после `i`.

7. Размер массива `minterms2` увеличивается на 1 с помощью функции `realloc`.

8. Вызывается функция `multiplication` с двумя аргументами - элементами массива `minterms` с индексами `i` и `j`.

9. Результат функции `multiplication` записывается в новый элемент массива `minterms2` с индексом `size - 1`.

10. Выводится строка, содержащая умножение двух элементов массива `minterms` и результат умножения.

11. Вызывается функция `checkY`, чтобы проверить, содержит ли результат умножения переменную `y`.

12. Если результат умножения не содержит переменную y, то элемент массива `minterms2` удаляется, размер массива уменьшается на 1.

13. Выводится строка "`=empty set`".

14. Если результат умножения содержит переменную `y`, то y заменяется на `x`, то есть переменную заменяем на 1.

15. Выводится строка со значением элемента массива `minterms2`.

16. Выводится символ перевода строки.

17. После завершения вложенного цикла `for` выводится символ перевода строки.

18. Выводится строка "`Final minterms: `".

19. В цикле `for` перебираются все элементы массива `minterms`.

20. Размер массива `minterms2` увеличивается на 1 с помощью функции `realloc`.

21. Текущий элемент массива `minterms` копируется в новый элемент массива `minterms2`.

22. Во следующем цикле `for` перебираются все элементы массива minterms2, начиная со следующего элемента после `i`.

23. Если результат функции `check` для текущего элемента и следующего элемента равен 1, то следующий элемент удаляется, размер массива уменьшается на 1.

24. Если результат функции `check` для текущего элемента и следующего элемента равен 2, то текущий элемент удаляется, размер массива уменьшается на 1.

25. После завершения вложенного цикла `for` выводится символ перевода строки.

26. В цикле `for` перебираются все элементы массива `minterms2`.

27. Во вложенном цикле `for` перебираются все переменные текущего элемента.

28. Выводится текущая переменная.

29. Если это последняя переменная в элементе, то выводится символ `","`.

30. После завершения вложенного цикла `for` выводится символ перевода строки.

31. Функция завершает свою работу.

32. Если результат умножения содержит переменную `y`, то её необходимо заменить на переменную `x`.

33. Функция `checkY` возвращает 1, если переменная `y` содержится в элементе.

34. Если переменная y содержится в элементе, то функция `checkY` возвращает 0.

35. Функция `check` возвращает 1, если элементы `a` и `b` различаются только в одной переменной.

36. Если элементы `a` и `b` различаются более чем в одной переменной, то функция `check` возвращает 0.

37. Если элементы `a` и `b` различаются только в одной переменной, то функция `check` возвращает 1.

38. Если элемент `a` является подмножеством элемента `b`, то функция `check` возвращает 2.

39. Если элемент `b` является подмножеством элемента `a`, то функция `check` возвращает 2.

40. Если элементы `a` и `b` не являются подмножествами друг друга, то функция `check` возвращает 0.

5 ОБЗОР РАБОТЫ ПРОГРАММЫ

На рисунке 5.1 представлены входные данные.

```
Enter minterms:
Enter 1-th minterm:
00000
Enter 2-th minterm:
00001
Enter 3-th minterm:
00010
Enter 4-th minterm:
00011
Enter 5-th minterm:
00100
```

Рисунок 5.1 – Входные данные

Результаты операции умножения представлены на рисунке 5.2.

```
Multiplication:
00000*00001=0000y=0000x
00000*00010=000y0=000x0
00000*00011=000yy=empty set
00000*00100=00y00=00x00
00001*00010=000yy=empty set
00001*00011=000y1=000x1
00001*00100=00y0y=empty set
00010*00011=0001y=0001x
00010*00100=00yy0=empty set
00011*00100=00yyy=empty set
```

Рисунок 5.2 – Результаты операции умножения

Выходные данные (минимальный набор минтерм, представляющих булевую функцию) представлены на рисунке 5.3.

```
Final minterms:
0000x,000x0,00x00,000x1,0001x
```

Рисунок 5.3 – Выходные данные

ЗАКЛЮЧЕНИЕ

В результате проделанной работы я разработала программу реализации операции * метода минимизации булевых функций Рота. Этот алгоритм и его реализация на языке С позволяют эффективно уменьшить сложность булевых выражений, что может привести к более оптимальной реализации логических схем и упрощению их анализа. Он может быть полезен в области схемотехники, автоматизации проектирования, оптимизации логических функций и других приложениях, требующих минимизации булевых выражений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Луцик, Ю. А. Учебное пособие по курсу «Арифметические и логические основы вычислительной техники» / Ю. А. Луцик, И. В. Лукьянова. - Минск: БГУИР, 2014.

[2] Искра, Н. А. Арифметические и логические основы вычислительной техники: пособие / Н. А. Искра, И. В. Лукьянова, Ю. А. Луцик. – Минск: БГУИР, 2016.

[3] Основы алгоритмизации и программирования: язык С: учеб. метод. пособие / Ю. А. Луцик, А. М. Ковальчук, Е. А. Сасин. – Минск: БГУИР, 2015. − 170с. : ил. 1

[4] Кнут Д. Э. Искусство программирования, том 1: Основные алгоритмы. - М.: Вильямс, 2006.

[5] Керниган Б., Ритчи Д. Язык программирования Си. - М.: Вильямс, 2011.

ПРИЛОЖЕНИЕ А

Листинг кода

1.Код roth.c

```
#include <stdio.h>
#include <stdlib.h>
#include "roth.h"

// Функция выполняет операцию * метода минимизации булевых функций Рота
char* multiplication(char* row1, char* row2, int num_vars)
{
    // Выделение памяти для полученной строки
    char* str = (char*)calloc(num_vars + 1, sizeof(char));

    // Выполнение умножения строк
    for (int i = 0; i < num_vars; i++)
    {
        if (row1[i] == row2[i])
            str[i] = row1[i];
        else {
            if (row1[i] == 'x' || row2[i] == 'x') {
                if (row1[i] == 'x')
                    str[i] = row2[i];
                else
                    str[i] = row1[i];
            }
            else
                str[i] = 'y';
        }
    }

    // Добавляем Null-terminate в конец строки
    str[num_vars] = '\0';

    return str;
}
// Функция считает количество 'y' в минтерме
int checkY(char* row, int num_vars)
{
    int count = 0;

    // Количество символов 'y' в строке
    for (int i = 0; i < num_vars; i++)
    {
        if (row[i] == 'y')
            count++;
    }

    // Если имеется более одного 'y', возвращается 0; в противном случае
возвращается 1
    if (count > 1)
        return 0;

    return 1;
}
// Функция проверяет, покрывает ли(содержит) одна минтерма другую
int check(char* row1, char* row2, int num_vars)
```

```
{
    int count = 0;

    // Количество совпадающих символов в строках
    for (int i = 0; i < num_vars; i++)
    {
        if (row1[i] == row2[i])
            count++;
    }

    //Если все символы совпадают, возвращается 1
    if (count == num_vars)
        return 1;
    else {
        int amount = num_vars - count; // Количество несовпадающих символов
        int amountRow1 = 0;            // Количество 'x' в первой минтерме
        int amountRow2 = 0;            // Количество 'x' во второй минтерме

        // Сравнение количества 'x' в каждой минтерме с количеством
несовпадающих переменных
        // Возвращаем 1, если первая минтерма покрывает вторую
        // Возвращаем 2, если вторая минтерма покрывает первую
        for (int i = 0; i < num_vars; i++)
        {
            if (row1[i] != row2[i]) {
                if (row1[i] == 'x')
                    amountRow1++;

                if (amountRow1 == amount)
                    return 1;

                if (row2[i] == 'x')
                    amountRow2++;

                if (amountRow2 == amount)
                    return 2;
            }
        }
    }

    return 0;
}
// Функция выполняет операцию *, выводит результат умножения, ищет новые
кубы, выводит конечный результат
void multiplicationInRoth(char** minterms, int num_minterms, int num_vars)
{
    // Массив для хранения полученных минтерм
    char** minterms2 = (char**)calloc(1, sizeof(char*));
    int size = 0; // Размер массива

    // Выполнение операции умножения минтерм
    printf("Multiplication:\n");
    for (int i = 0; i < num_minterms; i++)
    {
        for (int j = i + 1; j < num_minterms; j++)
```

```
        {
            size++;
            minterms2 = (char**)realloc(minterms2, size * sizeof(char*));

            minterms2[size - 1] = multiplication(minterms[i], minterms[j],
num_vars);
            printf("%s*%s=%s", minterms[i], minterms[j], minterms2[size -
1]);
            // Проверка на пустые множества
            if (checkY(minterms2[size - 1], num_vars) == 0) {
                size--;
                minterms2 = (char**)realloc(minterms2, size * sizeof(char*));
                printf("=empty set");
            }
            else {
                // Замена символа «y» на «x» в полученной минтерме
                for (int k = 0; k < num_vars; k++) {
                    if (minterms2[size - 1][k] == 'y') {
                        minterms2[size - 1][k] = 'x';
                    }
                }
                printf("=%s", minterms2[size-1]);           }

            printf("\n");
        }
    }

    printf("\n");

    printf("Final minterms: ");

    // Добавляем все оставшиеся минтермы (в том числе те, которые не
образовали новых кубов) в окончательные минтермы
    for (int i = 0; i < num_minterms; i++)
    {
        size++;
        minterms2 = (char**)realloc(minterms2, size * sizeof(char*));
        minterms2[size - 1] = minterms[i];
    }

    // Удаление лишних минтерм (повторяющихся и тех, которые образовали новые
кубы)
    for (int i = 0; i < size; i++) {
        for (int k = i + 1; k < size; k++) {
            if (check(minterms2[i], minterms2[k], num_vars) == 1)
            {
                for (int j = k; j < size - 1; j++)
                    minterms2[j] = minterms2[j + 1];
                size--;
                minterms2 = (char**)realloc(minterms2, size * sizeof(char*));
                k--;
            }
            else if (check(minterms2[i], minterms2[k], num_vars) == 2)
            {
                for (int j = i; j < size - 1; j++)
```

```
                    minterms2[j] = minterms2[j + 1];
                size--;
                minterms2 = (char**)realloc(minterms2, size * sizeof(char*));
                i--;
            }
        }
    }

    // Выводим в консоль конечный результат
    printf("\n");
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < num_vars; j++) {
            printf("%c", minterms2[i][j]);
        }
        if (i + 1 != size)
            printf(",");
    }
    printf("\n");
}
```

2. Код main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "roth.h"

// Функция ввода строки с консоли
char* setString() {
    char* str = (char*)malloc(1);
    str[0] = '\0';
    int i = 1, c;

    // Считывание символов из ввода до тех пор, пока не встретится новая
строка или конец файла
    while ((c = getchar()) != EOF && c != '\n') {
        i++;
        str = (char*)realloc(str, i * sizeof(char));
        if (str != NULL) {
            str[i - 2] = c;
            str[i - 1] = '\0';
        }
    }

    return str;
}
//Функция перевода цифровой последовательности типа string в число типа int
int aToI(char* str) {
    int result = 0;
    int i = 0;

    // Преобразование строки в целое число
    while (str[i] >= '0' && str[i] <= '9') {
        result = result * 10 + (str[i] - '0');
        i++;
    }

    return result;
```

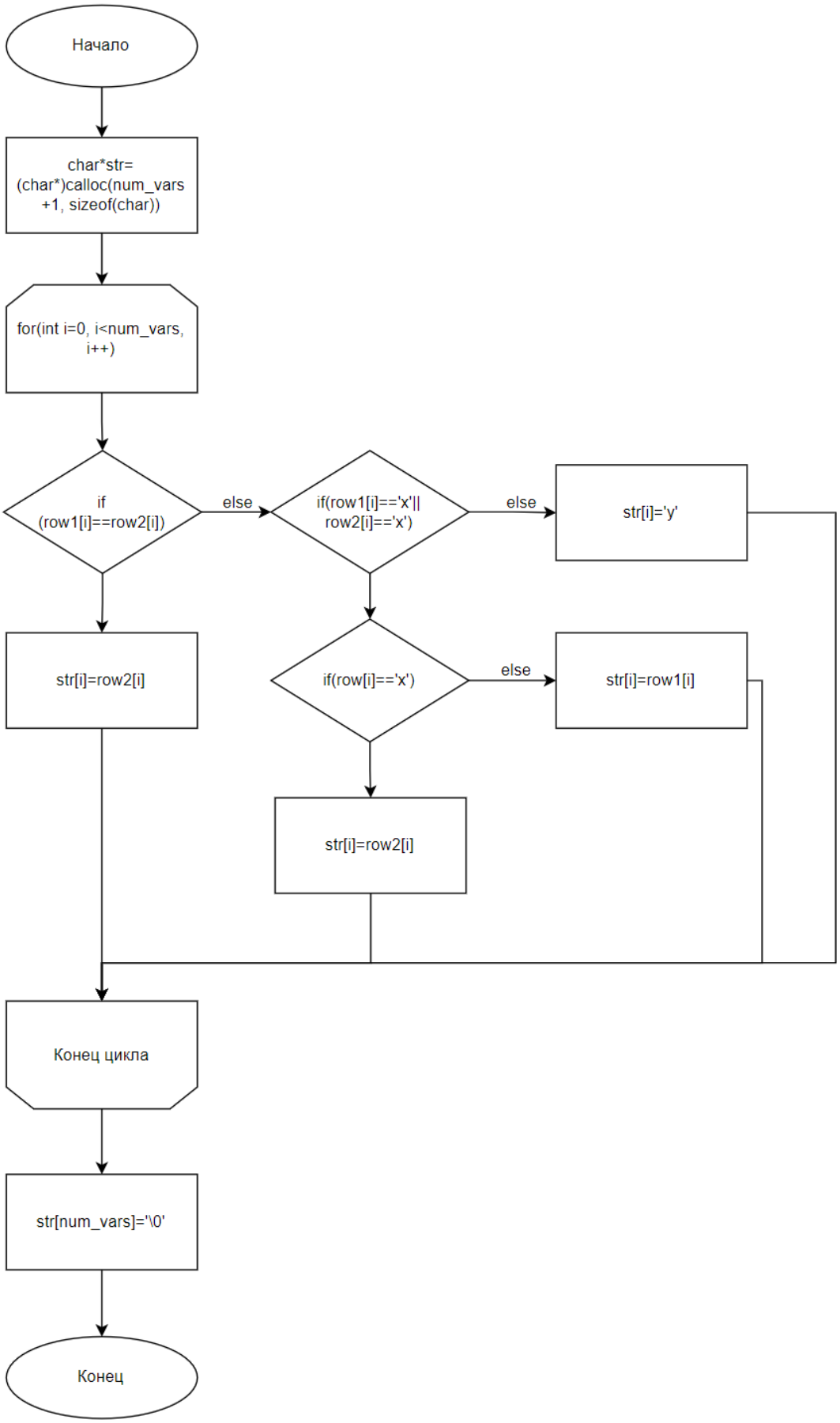
```
}
//Функция ввода строки и проверки на наличия только цифр
char* enterAndCheckNumberString(char* str)
{
    int flag = 0;

    while (!flag)
    {
        flag = 1;
        str = setString();
        int j = 0;

        // Проверка, содержит ли строка только цифры и имеет ли допустимую
длину
        while (str[j] != '\0') {
            if (str[j] < '0' || str[j] > '9' || j > 8)
                flag = 0;
            j++;
        }
       //Вывод сообщения о вводе некорректной информации
        if (flag == 0||atoi(str)<2) {
            printf("Error: incorrect data!\nEnter new data\n");
            flag=0;
        }
    }
    return str;
}
//Функция ввода минтерм и проверки на наличия только бинарных цифр и символа
'x'
char** enterAndCheckMinterms(int num_minterms, int num_vars)
{
    char** minterms = (char**)calloc(num_minterms, sizeof(char*));

    for (int i = 0; i < num_minterms; i++)
    {
        printf("Enter %d-th minterm: \n", i + 1);
        int flag = 0;

        while (!flag) {
            flag = 1;
            minterms[i] = setString();
            int j = 0, count = 0;

            // Проверка, содержит ли минтерма только символы «0», «1» или «x»
и имеет ли допустимую длину.
            while (minterms[i][j] != '\0') {
                count++;
                if ((minterms[i][j] < '0' || minterms[i][j] > '1') &&
(minterms[i][j] != 'x'))
                    flag = 0;
                j++;
            }

            if (count != num_vars)
                flag = 0;
```

```
            //Вывод сообщения о вводе некорректной информации
            if (flag == 0)
                printf("Error: incorrect data!\nEnter new data\n");
        }
    }

    return minterms;
}
int main() {
    char* num_vars;         // количество переменных в минтерме
    char* num_minterms;     // количество минтерм

    //Получение количество переменных с консоли
    printf("Enter number of variables: ");
    num_vars = enterAndCheckNumberString(num_vars);
    //Получение количество минтерм с консоли
    printf("Enter number of minterms: ");
    num_minterms = enterAndCheckNumberString(num_minterms);

    // Преобразование входных строк в целые числа
    int num_vars_int = aToI(num_vars);
    int num_minterms_int = aToI(num_minterms);

    // Получение минтерм с консоли
    printf("Enter minterms:\n");
    char** minterms = enterAndCheckMinterms(num_minterms_int, num_vars_int);

    // Выполнение умножения в алгоритме Рота
    multiplicationInRoth(minterms, num_minterms_int, num_vars_int);

    system("pause");
    return 0;
}
```

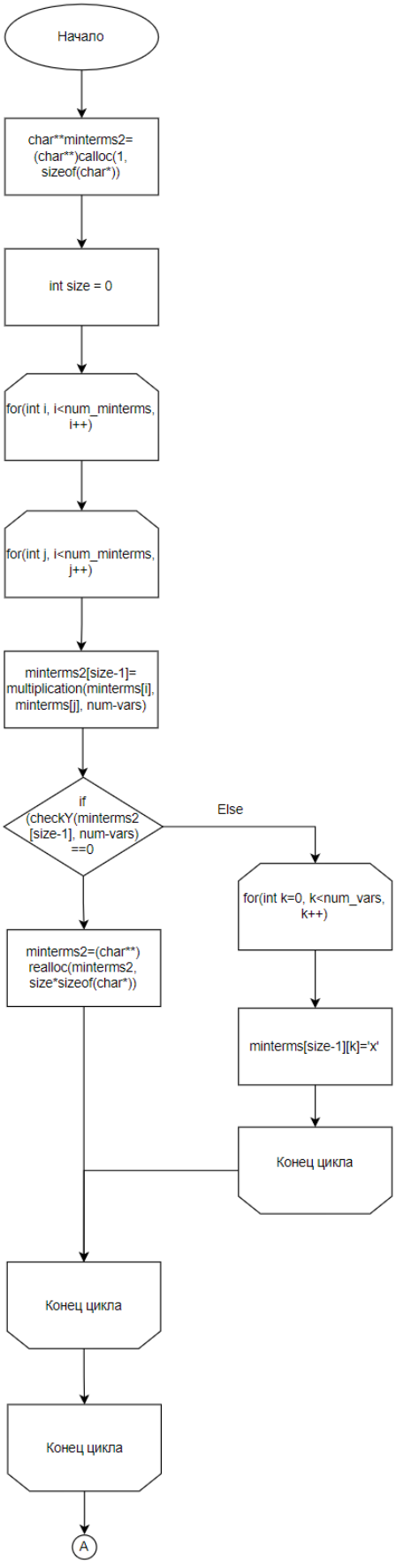
ПРИЛОЖЕНИЕ Б

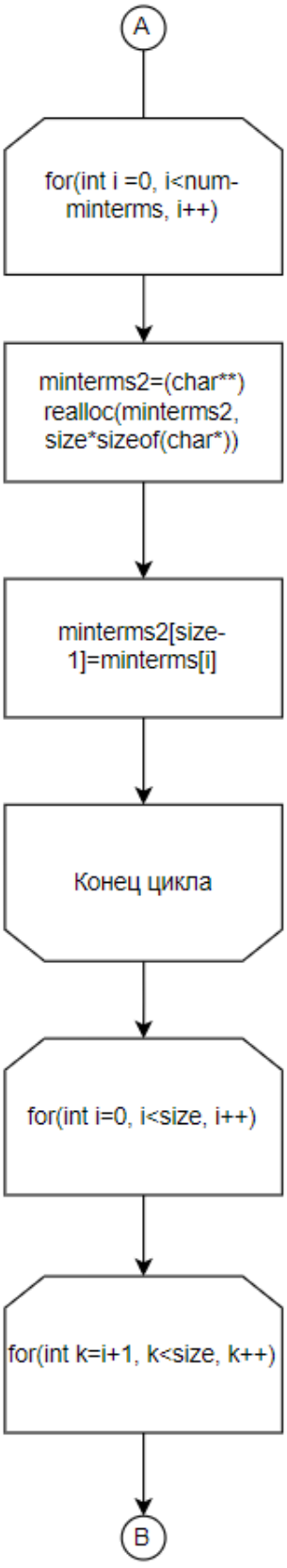
Блок-схема

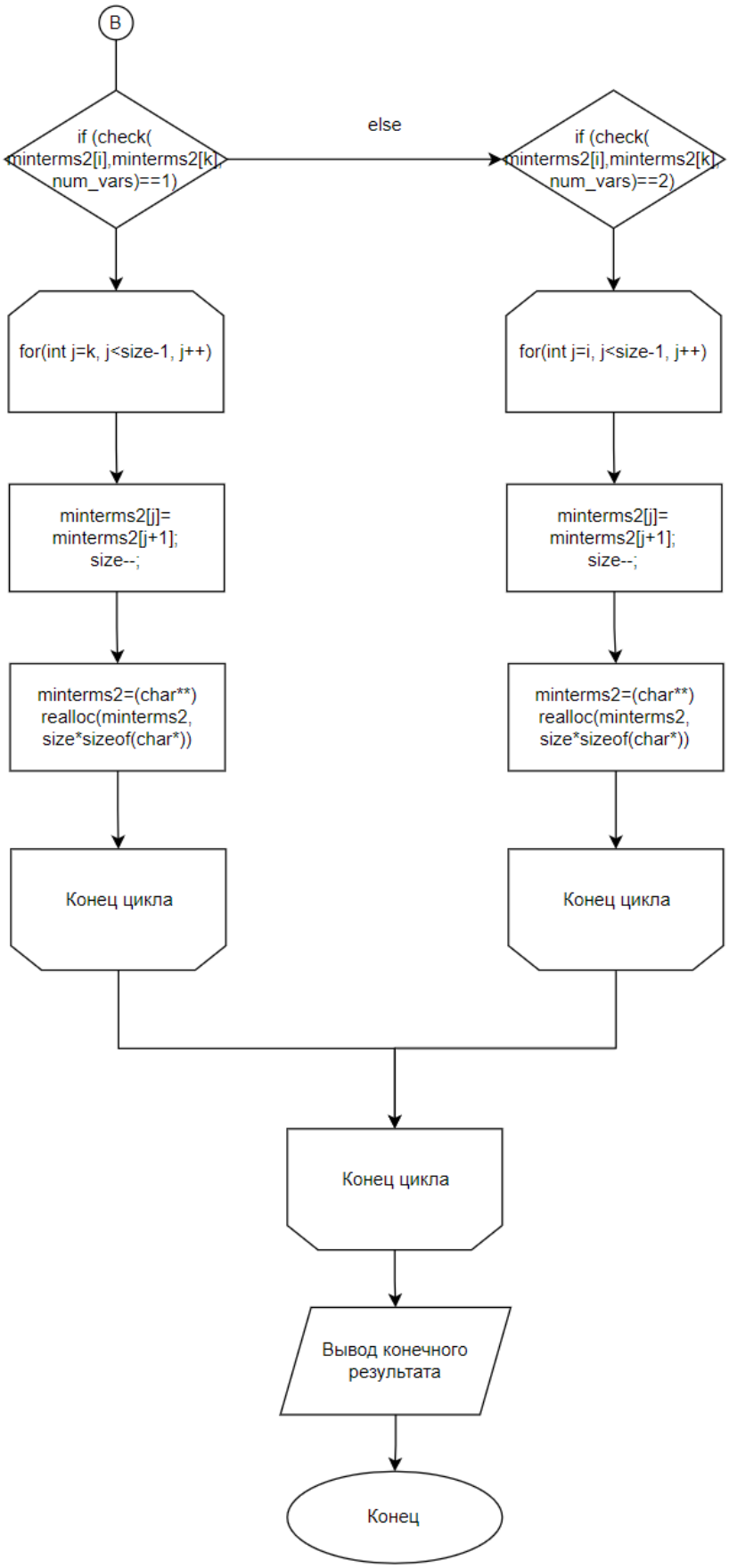
Операция * метода Рота(multiplication)



Операция *, поиск новых кубы, вывод конечного результата(multiplicationInRoth)




A
for(int i =0, i<num-
minterms, i++)
minterms2=(char**)
realloc(minterms2,
size*sizeof(char*))
minterms2[size-
1]=minterms[i]
Конец цикла
for(int i=0, i<size, i++)
for(int k=i+1, k<size, k++)
B


B
if (check(
minterms2[i],minterms2[k],
num_vars)==1)
else
if (check(
minterms2[i],minterms2[k],
num_vars)==2)
for(int j=k, j<size-1, j++)
for(int j=i, j<size-1, j++)
minterms2[j]=
minterms2[j+1];
size--;
minterms2[j]=
minterms2[j+1];
size--;
minterms2=(char**)
realloc(minterms2,
size*sizeof(char*))
minterms2=(char**)
realloc(minterms2,
size*sizeof(char*))
Конец цикла
Конец цикла
Конец цикла
Вывод конечного
результата
Конец