Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Издательское дело и полиграфия

Кафедра Информационные системы и технологии

Специальность 1-40 01 02-03

Специализация «Издательско-полиграфический комплекс»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА**

по дисциплине: «Специализированные информационные системы»

Тема: «Преобразование логических формул на язык FBD»

Исполнитель

Студентка 5 курса группы 10 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.С. Ракович

подпись, дата

Руководитель

доцент, канд. техн. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. И. Акунович

подпись, дата

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_  
Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. И. Акунович

подпись

Минск 2013

**Реферат**

Пояснительная записка содержит 15 страниц, 6 рисунков и 2 источника.

ПРОГРАММА ДЛЯ ФУНКЦИОНАЛЬНОГО ПРЕДСТАВЛЕНИЯ ФОРМУЛ, МЕТОДИКА ДЛЯ ПРЕОБРАЗОВАНИЯ ОПЕРАТОРНОГО ПРЕДСТАВЛЕНИЯ ФОРМУЛ В ФУНКЦИОНАЛЬНОЕ ПРЕДСТАВЛЕНИЕ, ГРАФИЧЕСКИЙ ЯЗЫК FBD, ЯЗЫК ПРОГРАММИРОВАНИЯ C#, ПРОГРАММНОЕ СРЕДСТВО ISAGRAF 6.1, ПРОГРАММНОЕ СРЕДСТВО VISUAL STUDIO 2010.

Курсовой проект состоит из пояснительной записки, проекта, разработанного на языке С#, исходного примера.

Целью выполнения курсового проекта является разработка программного средства, преобразующего логические формулы на язык FBD.

Во введении описывается графический язык FBD и его применение.

В первом разделе осуществляется постановка задачи курсового проекта.

Во втором разделе приводится реализация поставленной задачи.

В заключении – вывод о проделанной работе.

Оглавление

[Введение 5](#_Toc375843145)

[Постановка задачи 6](#_Toc375843146)

[Реализация поставленной задачи 7](#_Toc375843147)

[Заключение 12](#_Toc375843148)

[Список литературы 13](#_Toc375843149)

[Приложение 14](#_Toc375843150)

# Введение

Язык программирования FBD (Function Block Diagram) является составной частью стандарта IEC-61131 и так же входит в стандарт IEC-61499.

FBD является*графическим* языком и применяется для построения комплексных процедур, состоящих из различных функциональных библиотечных блоков: арифметических, тригонометрических, регуляторов, мультиплексоров и т.д. Он подходит также для управления непрерывными процессами и процессами регулирования. При этом осуществляется представление функций посредством блоков, связанных между собой*.* Соединения между выходами функциональных блоков в явном виде могут отсутствовать, а выход блока может соединяться с входами одного или нескольких блоков.

Основными объектами языка FBD являютсяэлементарные функции и элементарные функциональные блоки (ФБ). Они находятся в библиотеке, логика их работы (программа) написана на языке С и не может быть изменена в редакторе FBD (изменять можно только их параметры). Кроме них можно использовать функции и ФБ пользователя, которые конструируются пользователем из элементов языка FBD.

Разработка программы осуществляется с помощью *графического редактора* посредством формирования блок-схемы из перечисленных выше компонентов, которые объединяются друг с другом либо посредством внешних (фактических) параметров (переменные, соответствующие входам и выходам), либо непосредственно линиями связи – *графическими связями*.

FBD - программа описывает *функцию* между входными и выходными переменными. Эта функция представляется совокупностью элементарных ФБ. Тип каждой переменной должен быть тем же, что и тип соответствующего входа. Входом FBD - блока может быть константа, любая внутренняя, входная или выходная переменная.

Данный язык программирования, помимо прочего, может использоватся для описания шагов и переходов в языке SFC. Функциональные блоки инкапсулируют данные и методы, чем подобны объектно-ориентированным языкам программирования, но они не поддерживают наследование и полиморфизм.

# Постановка задачи

Целью выполнения курсового проекта является разработка программного средства, выполняющего преобразование логических формул на язык FBD.

Проблема заключается непосредственно в преобразовании логических операторов в функциональные. Для этого будет использоваться дерево синтаксического разбора.

Порядок выполнения преобразования:

1. Преобразование логических операторов в функциональные;
2. Запись функциональных формул в текстовый файл;
3. Преобразование данного текстового файла в xml-документ;
4. Распознавание документа средствами IsaGraf и преобразование его в изображение (функциональные блоки языка FBD).

# Реализация поставленной задачи

Основная задача данной работы заключается в том, чтобы логические формулы преобразовать в функциональные формулы. Предположим, что наша логическая формула имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  | | (1) |
| где | Знак «+» является логической операцией «И» |  |
|  | Знак «\*» является логической операцией «ИЛИ» |  |
|  | Знак «^» является логической операцией «НЕ» |  |

Значит, данную логическую формулу нужно преобразовать таким образом, чтобы функциональный вид этой же формулы был следующим:

|  |  |
| --- | --- |
|  | (2) |

Преобразовывать логическую формулу в функциональную формулу будем с помощью дерева, которое будет состоять узлов. Каждый узел будет содержать в себе левую ветку, правую ветку, и сами символы. Смотри «Приложение 1».

Для более понятного представления дерева ниже приведем часть программного кода:

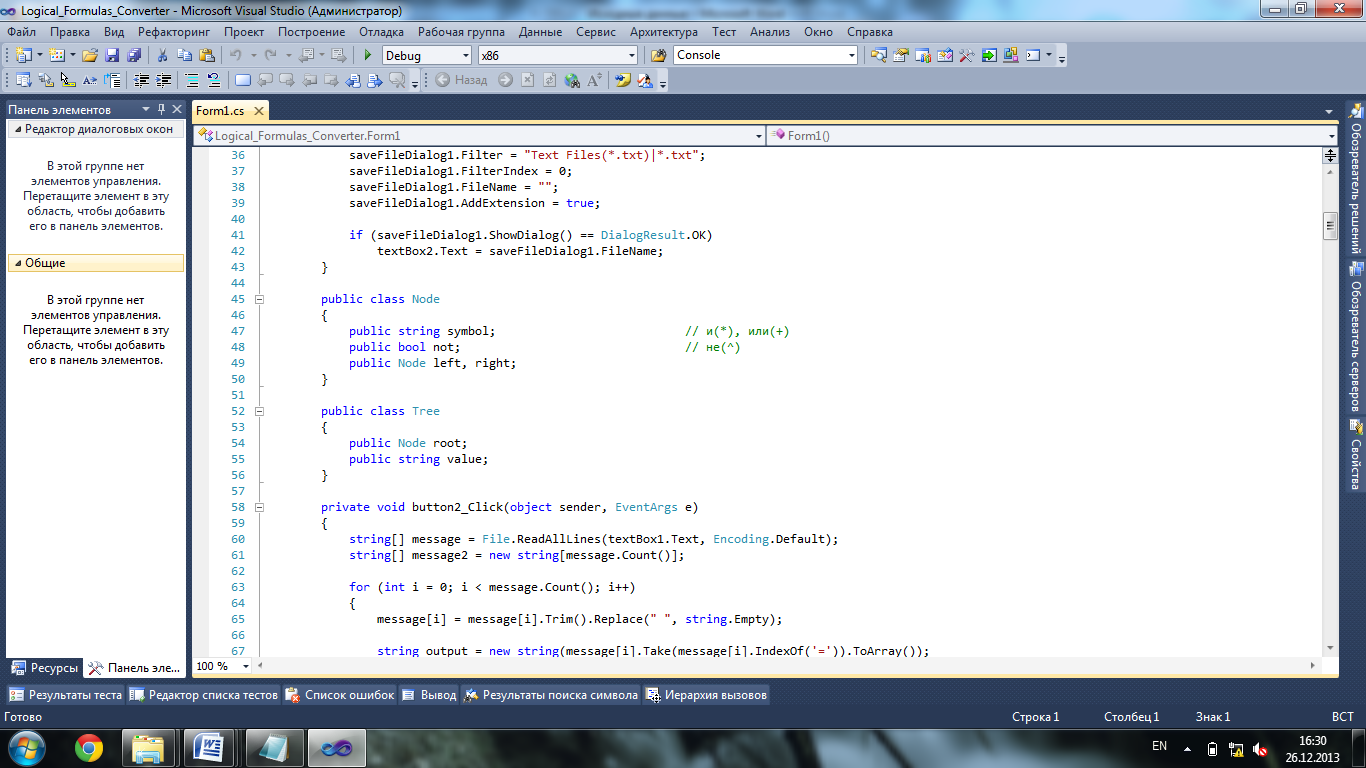


Рис. 1 – Класс дерева и узлов дерева

Разберем код подробнее. Выше написан класс «Дерево синтаксического разбора». У данного дерева будет один «Корень дерева». Корнем дерева будет другой класс, называемый «Узел дерева».

Класс «Узел дерева» будет содержать в себе элемент «Символ», в который будет записываться «+», либо «\*», либо «^», либо конечный символ «a», «b», либо «c».

У класса «Узел» также имеется левая ветка и правая ветка. В левую ветку и правую ветку записывается новый «Узел», который, в свою очередь, будет иметь свои «Узлы»: левый и правый узлы.

Класс «Узел» также имеет булевскую переменную, в которой будет указываться, имеется ли в данном «Узле» отрицание («НЕ»).

Проблема реализации данного алгоритма ещё состоит и в том, чтобы синтаксический разбор работал правильно и в случае, если логическая формула имеет вложенные скобки, чтобы отрицание ставилось для всего выражения в скобках.

Для этого создается следующая функция, которая подсчитывает количество вложенных скобок в исходном выражении, запоминает их позиции и сопоставляет позициям открывающихся скобок позиции закрывающихся скобок.

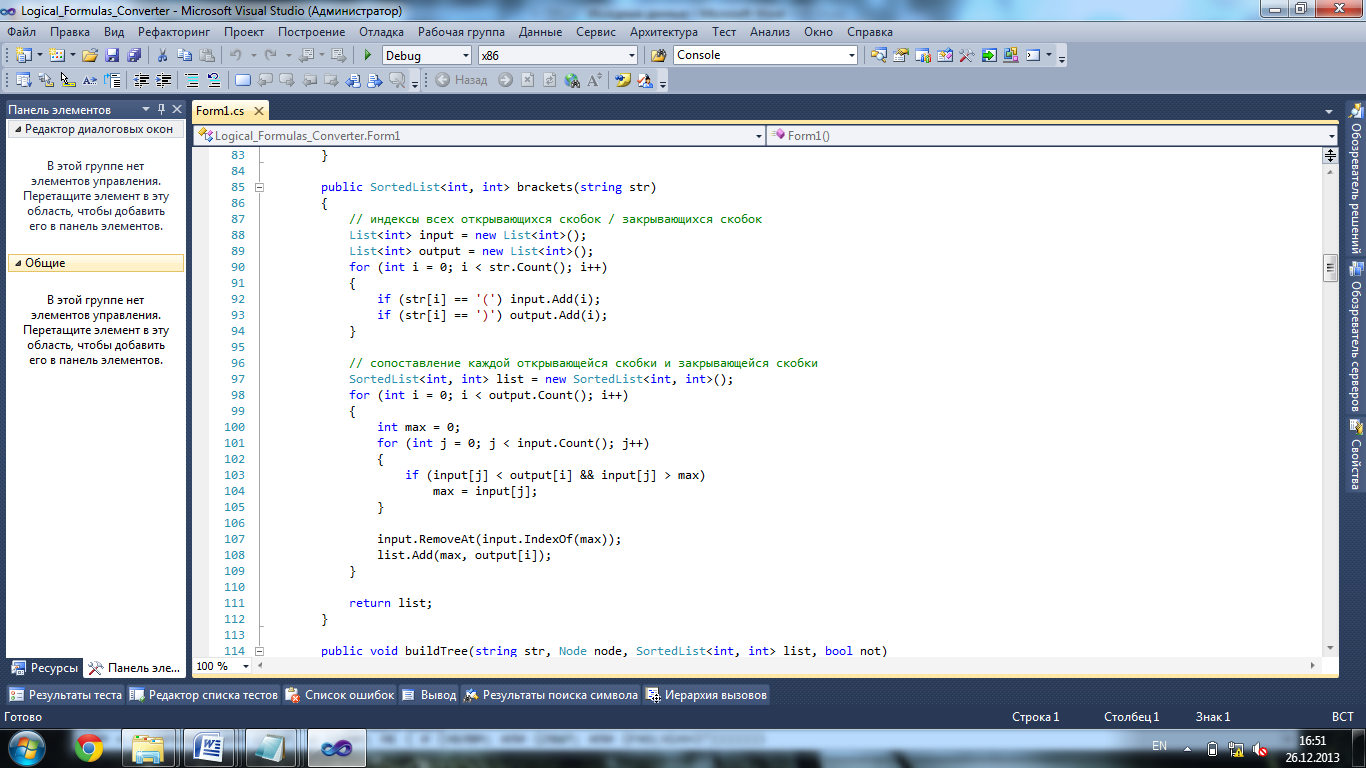


Рис. 2 – Функция сопоставления индексов открывающихся скобок  
закрывающимся скобкам

После нахождения позиций всех открывающихся и закрывающихся скобок, проходим по циклу всех закрывающихся скобок и ищем среди открывающихся скобок индексы, которые меньше индекса закрывающейся скобки, но наибольшая среди других индексов открывающихся скобок.

Сведения о сопоставлении позиций открывающихся и закрывающихся скобок содержатся в сортированном списке list.

Логически выше уже была описана функция построения дерева, поэтому описывать более подробно её не будем.

Функция вывода дерева похожа на функцию построения дерева, она выглядит следующим образом:

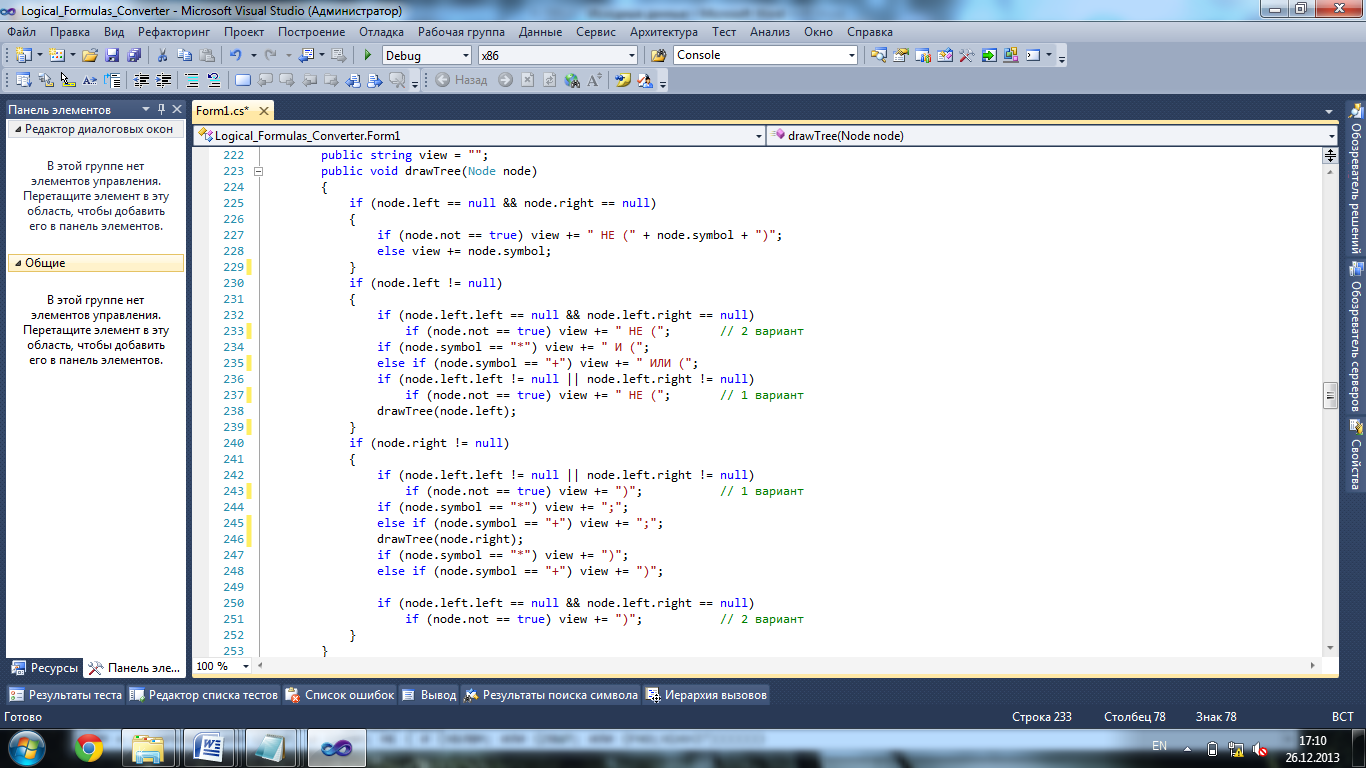


Рис. 3 – Функция вывода дерева

Прохождение дерева происходит рекурсивно.

Для начала выводится текущий символ, далее идет проверка, есть ли у узла левая ветка или правая ветка. Если есть, то функция вызывает сама себя рекурсивно, но входным параметром, то есть, текущим узлом дерева, задается уже левая или правая ветка дерева.

Результатом работы функции вывода дерева является следующий файл:

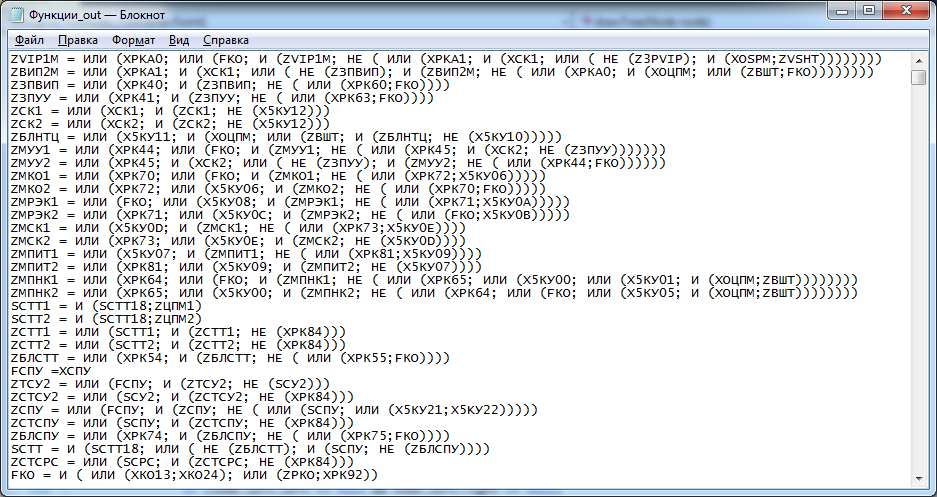


Рис. 4 – Преобразованный файл с функциональными формулами

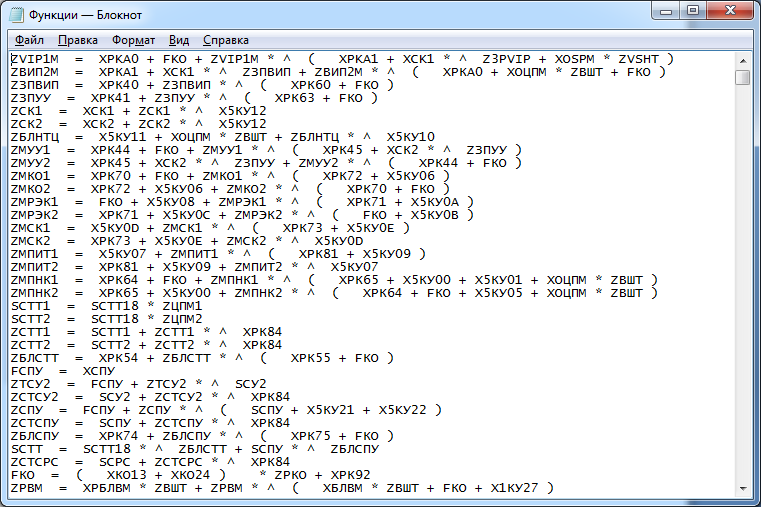


Рис. 5 – Исходный файл с логическими формулами

Теперь файл с функциональными формулами можно преобразовать в файл .isaxml, который программа ISaGRAF 6.1, сама преобразует в блок-схему.

Для этого необходимо отредактировать файл с расширением .isaxml, таким образом, как показано в листинге 1.

Функциональная формула, которая описана выше, имеет следующий внешний вид:

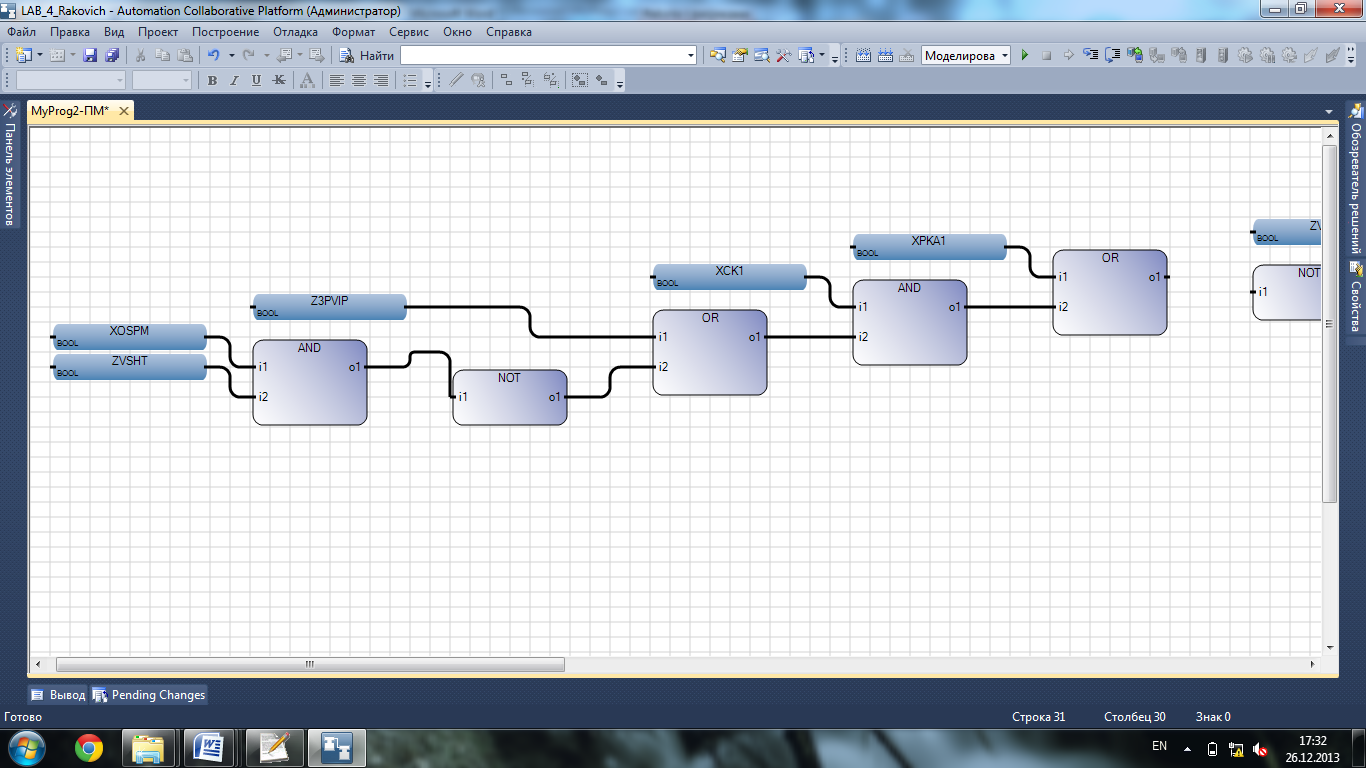


Рис. 6 – Внешний вид функциональной формулы

# Заключение

В результате выполнения курсового проекта было разработано программное средство, осуществляющее преобразование логических формул на язык FBD.

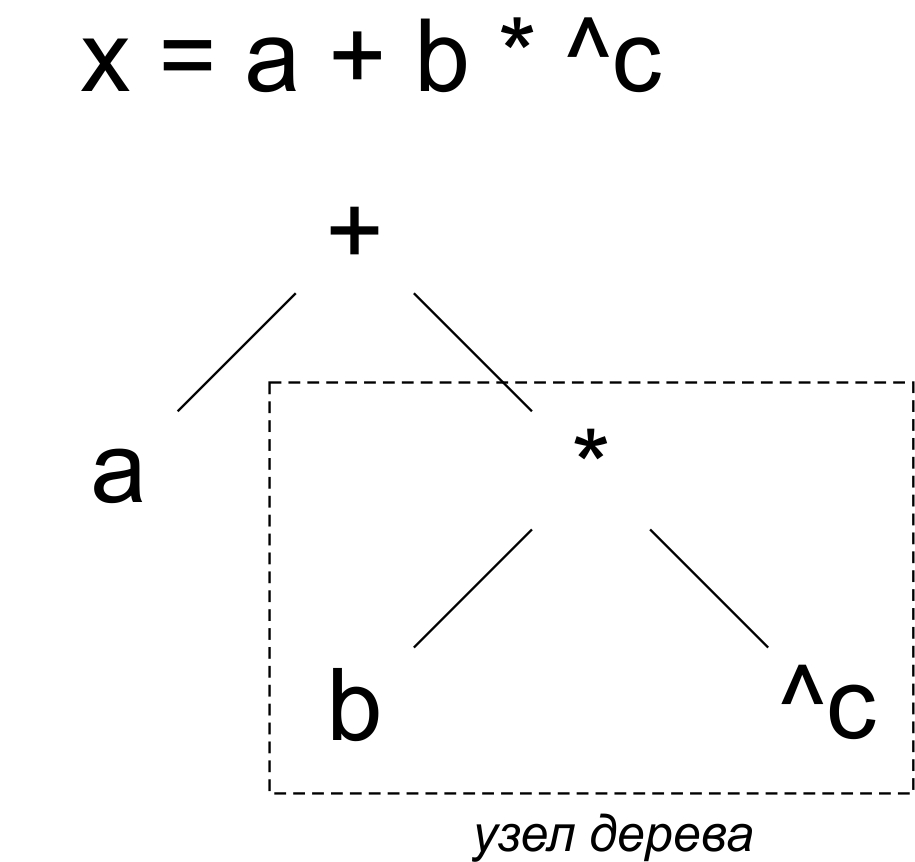
Все этапы преобразования выполняются успешно. Результатом работы программы является набор функциональных блоков, описывающих логические операции, представленные в исходном примере.

# Список литературы

1. Акунович, С. И. Серия «Авторское приложение». Приложение «ГИПЕРСИСТЕМА» // Описание применения – Мн.: БГТУ, 2013.
2. Акунович, С. И. Композиционное конструирование моделей систем дискретного управления в среде DELPHI// Методические указания – Мн.: БГТУ, 2009.

# Приложение

**Приложение 1**



\* - на данном рисунке изображена структурная схема дерева синтаксического разбора. Здесь выделен узел дерева.

**Листинг 1**

<?xml version="1.0" encoding="utf-8"?>

<Pou FileVersion="200.00.000" Name="MyProg2" ElementGuid="c5a3cad5-afcf-4220-8030-436a6124947c" Comment="" SingleTextLine="" Language="FBD" CustomFbd="0" Localization="Prj" GroupName="(Определяемый пользователем)" GenDebugInfo="false" UseResPassword="true" GenerateSymbMon="true" Is1499="false">

<Program />

<LocalVars>

<Variable Name="XPKA0" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="FKO" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="ZVIP1M" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarOutput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="XPKA1" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="XCK1" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="Z3PVIP" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="XOSPM" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

<Variable Name="ZVSHT" DataType="BOOL" InitialValue="" Comment="" Address="" Kind="VarInput" Alias="" AccessRights="ReadWrite" StringSize="0" />

</LocalVars>

<PouBody><![CDATA[PROGRAM MyProg2

#info= FBD

@@NBID=17

@BOX:2,P=(368,6),S=(24,12),C=(2,1),X=OR

T=0

@BOX:4,P=(328,10),S=(24,12),C=(2,1),X=OR

T=0

@BOX:6,P=(288,14),S=(24,12),C=(2,1),X=AND

T=0

@BOX:8,P=(248,20),S=(24,8),C=(1,1),X=NOT

T=0

@BOX:9,P=(208,18),S=(24,12),C=(2,1),X=OR

T=0

@BOX:11,P=(168,22),S=(24,12),C=(2,1),X=AND

T=0

@BOX:13,P=(128,26),S=(24,12),C=(2,1),X=OR

T=0

@BOX:14,P=(88,28),S=(24,8),C=(1,1),X=NOT

T=0

@BOX:16,P=(48,30),S=(24,12),C=(2,1),X=AND

T=0

@VAR:1,P=(400,4),S=(32,4),C=(1,1),X=ZVIP1M

@VAR:3,P=(328,4),S=(32,4),C=(1,1),X=XPKA0

@VAR:5,P=(288,8),S=(32,4),C=(1,1),X=FKO

@VAR:7,P=(248,12),S=(32,4),C=(1,1),X=ZVIP1M

@VAR:10,P=(168,16),S=(32,4),C=(1,1),X=XPKA1

@VAR:12,P=(128,20),S=(32,4),C=(1,1),X=XCK1

@VAR:15,P=(48,24),S=(32,4),C=(1,1),X=Z3PVIP

@VAR:17,P=(8,28),S=(32,4),C=(1,1),X=XOSPM

@VAR:18,P=(8,32),S=(32,4),C=(1,1),X=ZVSHT

#end\_info

#info= ID\_MAX

NextId=17

#end\_info

END\_PROGRAM]]></PouBody>

</Pou>