

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Факультет энергетический
Кафедра информатика и вычислительная техника и прикладная математика

КУРСОВАЯ РАБОТА

по дисциплине Теория языков программирования и метода трансляции

на тему: «Программная реализация транслятора»

Выполнил студент группы ИВТ-18 Забаровский Р.В.

Проверил доцент кафедры ИВТ и ПМ Коган Е.С.

Отлично
Е.С. Коган

Чита
2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Факультет энергетический
Кафедра информатика и вычислительная техника и прикладная математика

ЗАДАНИЕ

на курсовую работу

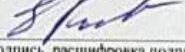
По дисциплине: Теория языков программирования и метода трансляции
Студенту Забаровскому Роману Вадимовичу

специальности 09.03.01 Информатика и вычислительная техника

1. Тема курсовой работы: Программная реализация транслятора
2. Срок подачи студентом законченной работы: 23 декабря 2021 г.
3. Исходные данные к работе: литературные источники, источники в сети интернет
4. Перечень подлежащих в курсовой работе вопросов:
 - а) Постановка и анализ задачи;
 - б) Анализ данных;
 - в) Программная реализация;
 - г) Техническое задание;
 - д) Руководство пользователя;

5. Перечень графического материала (если имеется): -

Дата выдачи задания «9» сентября 2021 г.

Руководитель курсовой работы  /Коган.Е.С./
(подпись, расшифровка подписи)

Задание принял к исполнению

«11» сентября 2021 г.

Подпись студента  /Забаровский Р.В./

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Факультет энергетический
Кафедра информатика и вычислительная техника и прикладная математика

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по 09.03.01 Информатика и вычислительная техника

на тему: «Программная реализация транслятора»

Выполнил студент группы ИВТ-18 Забаровский Р.В.

Руководитель работы: доцент кафедры ИВТ и ПМ Коган Евгения Семёновна

РЕФЕРАТ

Пояснительная записка – 20 страниц, 6 – рисунков.

ТРАНСЛЯТОР, С#, С++, БЛОК СХЕМА, ЛЕКСИЧЕСКИЙ АНАЛИЗАТОР, ГРАММАТИКА, ЛЕКСЕМА.

В работе описывается создание транслятора с языка С++ на язык Блок схем. Разработка транслятора в данной курсовой работе производится на высокоуровневом объектно-ориентированном языке С#. В работе рассматриваются понятие транслятора, принципы построения транслятора. Также курсовая работа содержит руководство пользователя.

СОДЕРЖАНИЕ

Реферат	4
Введение	6
1 Постановка и анализ задачи	7
1.1 Описание предметной области.....	7
1.2 Постановка задачи	9
1.3 Средства реализации	9
2 Анализ данных.....	10
2.1 Входные данные.....	10
2.2 Выходные данные.....	10
3 Программная реализация.....	11
3.1 Лексический анализатор.....	12
3.2 Графическое представление.....	14
4 Техническое задание	15
4.1 Введение.....	15
4.2 Требования к функциональным характеристикам	15
4.3 Требования к надёжности	15
4.4 Требования к программной документации	15
4.5 Требования к информационной и программной совместимости.....	15
5 Руководство пользователя.....	16
Заключение.....	19
Список использованных источников	20

ВВЕДЕНИЕ

В настоящее время искусственные языки, использующие для описания предметной области текстовое представление, широко применяются не только в программировании, но и в других областях. С их помощью описывается структура всевозможных документов, трехмерных виртуальных миров, графических интерфейсов пользователя и многих других объектов, используемых в моделях и в реальном мире. Для того, чтобы эти текстовые описания были корректно составлены, а затем правильно распознаны и интерпретированы, используется специальный вид программ, именуемый трансляторами. Разработка трансляторов на сегодняшний момент является одной из самых актуальных областей в разработке программного обеспечения.

В данной курсовой работе рассматривается процесс разработки транслятора с языка C++ в язык программирования блок схем.

1 Постановка и анализ задачи

1.1 Описание предметной области

Транслятор – это программа или техническое средство, выполняющее трансляцию программы. Трансляция программы – преобразование программы, представленной на одном из языков программирования, в программу на другом языке, также обычно выполняет диагностику ошибок и формирует словари идентификаторов и т.д.

Язык, на котором написана входная программа, называется исходным языком, а сама программа исходным кодом. Выходной язык называется целевым языком.

Основным в определении транслятора является эквивалентность исходной и результирующей программ. Эквивалентность этих двух программ означает совпадение их смысла с точки зрения семантики входного языка и семантики выходного языка. Без этого сам транслятор не имеет практического смысла.

Процесс отображение входной программы в эквивалентную ей выходную программу состоит из двух частей: анализ и синтез.

Анализ разбивает входную программу на составные части и накладывает на них грамматическую структуру. Затем использует эту структуру для создания промежуточного представления входной программы. Если анализ обнаруживает, что входная программа неверно составлена синтаксически или семантически, он должен выдать информативные сообщения об этом, чтобы пользователь мог исправить обнаруженные ошибки. Анализ также собирает информацию об исходной программе и сохраняет её в структуре данных, называемой таблицей идентификаторов, которая передаётся вместе с промежуточным представлением синтезу.

Синтез строит требуемую целевую программу на основе промежуточного представления и информации из таблицы идентификаторов.

Анализ часто называют начальной стадией, а синтез – заключительной.

Если посмотреть на процесс трансляции более детально, можно увидеть, что он представляет собой последовательность фаз, каждая из которых преобразует одно из представлений входной программы в другое. На практике некоторые фазы могут быть объединены, а межфазное промежуточное представление может не строиться явно. Таблица идентификаторов, в которой хранится информация обо всей входной программе, используется всеми фазами транслятора.

Первая фаза трансляции называется лексическим анализом или сканированием. Лексический анализатор считывает поток символов, составляющих входную программу, и группирует эти символы в значащие последовательности, называемые лексемами. Лексемы передаются последующей фазе транслятора, синтаксическому анализу.

Вторая фаза трансляции – синтаксический анализ или разбор. Во время выполнения этой фазы используются лексемы, полученные от лексического анализатора, для создания древовидного промежуточного представления программы, которое описывает грамматическую структуру потока лексем. Типичным промежуточным представлением является синтаксическое дерево, в котором каждый внутренний узел представляет операцию, а дочерние узлы – аргументы этой операции.

После синтаксического анализа исходной программы трансляторы с помощью полученного ранее синтаксического дерева генерируют код на целевом машинном языке, и трансляция завершается.

Таблица идентификаторов представляет собой структуру данных, содержащую записи для каждого имени переменной с полями для атрибутов имени. Структура данных должна быть разработана таким образом, чтобы позволять транслятору быстро находить запись для каждого имени, а также быстро сохранять данные в записи и получать их из неё.

1.2 Постановка задачи

Разрабатываемый транслятор будет иметь возможность выполнять трансляцию с языка C++ в блок схему.

При трансляции транслятор будет выполнять фазы лексического анализа, графического построения. Важно отметить, что на каждой из фаз транслятор должен осуществлять контроль ошибок в анализируемом им исходном коде и по мере их нахождения уведомлять об этом пользователя.

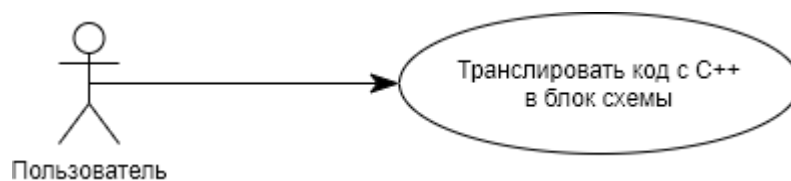


Рисунок 1 – Use-case диаграмма «Транслятор из C++ в блок схему»

1.3 Средства реализации

Для реализации разрабатываемого транслятора был выбран язык программирования высокого уровня C#. Это объектно-ориентированный язык программирования со статической типизацией. C# поддерживает универсальные методы и типы, которые повышают безопасность типов и производительность. Так как это ООП язык, значит он поддерживает инкапсуляцию, наследование и полиморфизм, что облегчает разработку программных компонентов и средств.

2 Анализ данных

Данные, с которыми работает транслятор можно разделить на несколько групп. Входные данные – данные поступающие от пользователя транслятору и называемые исходным кодом. Промежуточные данные – это данные, используемые транслятором во время работы. Выходные данные – данные поступающие от транслятора пользователю, целевой язык.

2.1 Входные данные

В качестве входных данных для разрабатываемого транслятора выступает исходный код программы на языке программирования C++.

2.2 Выходные данные

В качестве выходных данных для разрабатываемого транслятора выступают файлы с изображением блок схемы, которые он создает в качестве результата, выполняя трансляцию исходного кода, полученного от пользователя.

3 Программная реализация

В ходе разработки программы были созданы классы основных идентификаторов исходного языка:

1. Cout.cs – класс отвечающий за блок вывода.
2. Cin.cs – класс отвечающий за блок ввода.
3. If.cs – класс отвечающий за блок ветвления.
4. Cycle.cs – класс отвечающий за блок циклов.
5. Switch.cs – класс отвечающий за блок оператора switch.
6. Action.cs – класс отвечающий за блок действия (присваивания, умножения, деления и т.д.).
7. EntryBlock.cs – класс отвечающий за начало и конец блок схемы.

Для основных фигур блок схемы были созданы классы, в которых описаны функции необходимые для их построения и заполнения:

1. ActionBlock.cs
2. InputBlock.cs
3. OutputBlock.cs
4. CycleBlock.cs
5. CaseBlock.cs
6. ConditionalBlock.cs
7. EntryBlock.cs

Главной целью классов является добавление элементов, соединенных линиями на блок-схему (Рис.2).

Также были созданы классы:

1. CPPAnalyzer.cs – класс лексического анализатора.
2. Vizualizer.cs – класс строящий блок схему.
3. GraphicOperator.cs – класс строящий отдельные блоки.

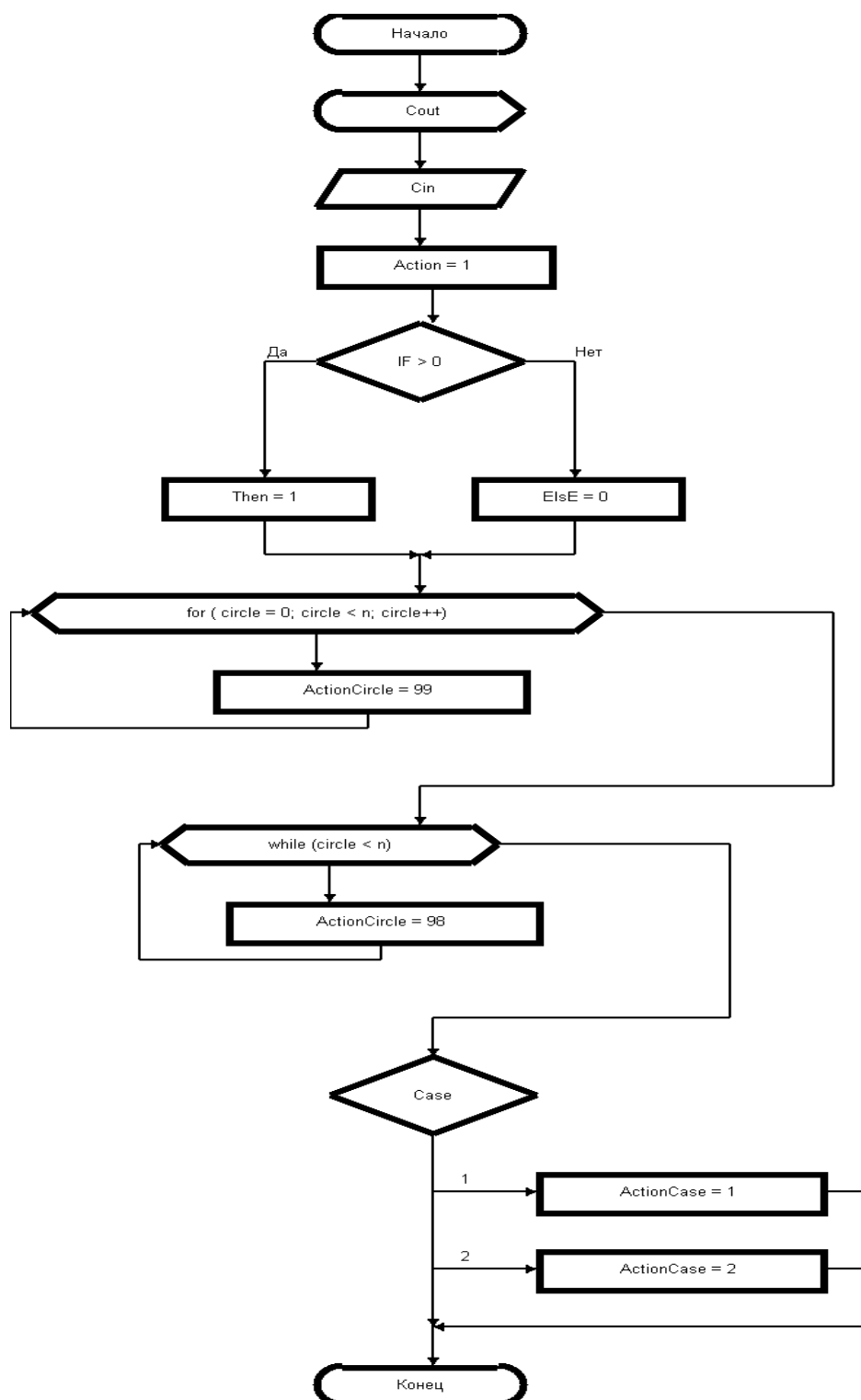


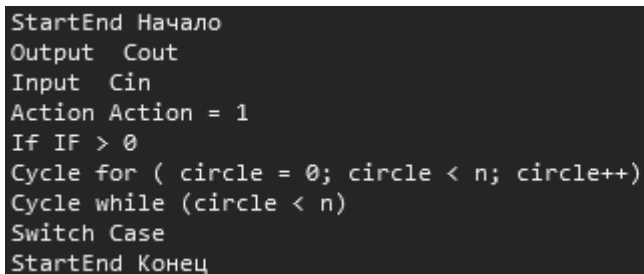
Рисунок 2 - Блок схема элементов

3.1 Лексический анализатор

В начале выполняется этап лексического анализа, процесс аналитического разбора входной последовательности символов на лексемы, с

целью получения на выходе идентифицированных последовательностей или же «токенов».

На вход лексического анализатора передается текст входной программы, где для каждой лексемы строится токен (Рис. 3). В таком виде первым отображается имя токена, а вторым его значение, которое указывается в таблице идентификаторов соответствующее этому токenu.



```
StartEnd Начало
Output Cout
Input Cin
Action Action = 1
If IF > 0
Cycle for ( circle = 0; circle < n; circle++)
Cycle while (circle < n)
Switch Case
StartEnd Конец
```

Рисунок 3 – Вывод токенов в консоль

Для транслятора был создан лексический анализатор языка C++. Для считывания символов и работы с идентификаторами были разработаны классы такие как: Cout.cs, Cin.cs, CPPAnalyzer.cs и т.д.

При получении исходного кода класс CPPAnalyzer начиная после «int main(){» разбивает весь текст на массив, убирая при этом все управляющие символы исходного языка. В начало добавляется токен StartEnd. После вызывается функция, в которую передается массив и в нем выделяются токены. Завершается работа добавлением токена StartEnd в конец массива.

В начало и конец добавляется один и тот же токен, но с разным значением, это делается для того чтобы класс, который будет строить блок схему мог определить где начало и конец исходной программы.

Если при проведении анализа возникнет ошибка, из-за которой становится невозможным дальнейший анализ, программа сообщит об этом, но если все прошло успешно, то работа переходит к классу визуализации Vizualizer.

3.2 Графическое представление

Построение графического изображения блок схемы в общем виде происходит по следующему порядку:

1. Строится блок «Начало».
2. Из предыдущего блока строится линия в следующий.
3. Из массива берется токен и в зависимости от токена строится определенный блок с необходимыми ему линиями (такими как для операнда ветвления if: линия для верного условия и линия для неверного условия).
4. Повторяется второе действие.
5. Повторяется третье действие.
6. Строится блок «Конец».

В класс Vizualizer передается массив токенов, где по описанному алгоритму строятся блоки схемы по порядку их выполнения в исходном коде.

После построения блок схема сохраняется на устройство пользователя в диалоговом окне, либо при ошибке построения выдается соответствующая ошибка.

4 Техническое задание

4.1 Введение

Приложение транслятор является программой, которая считывает текст программы, написанной на одном языке (исходном), и транслирует (переводит) его в эквивалентный текст на другом языке (целевом).

4.2 Требования к функциональным характеристикам

Программа должна выполнять трансляцию текста программы с языка программирования C++ в блок схему.

4.3 Требования к надёжности

В случае ошибок в работе приложения или некорректности данных предоставляемых пользователем программа должна вывести уведомление с информацией о действиях необходимых для исправления полученной ошибки.

4.4 Требования к программной документации

Программная документация должна содержать руководство пользователя.

4.5 Требования к информационной и программной совместимости

Для нормальной работы приложение должно функционировать на компьютере под управлением семейства Windows.

5 Руководство пользователя

Разрабатываемый в данной курсовой работе транслятор является программой с графическим интерфейсом.

Для запуска программы необходимо нажать на исполняемый файл CPlusFlowCharts.exe, после этого пользователь увидит главное окно программы (Рис. 4).

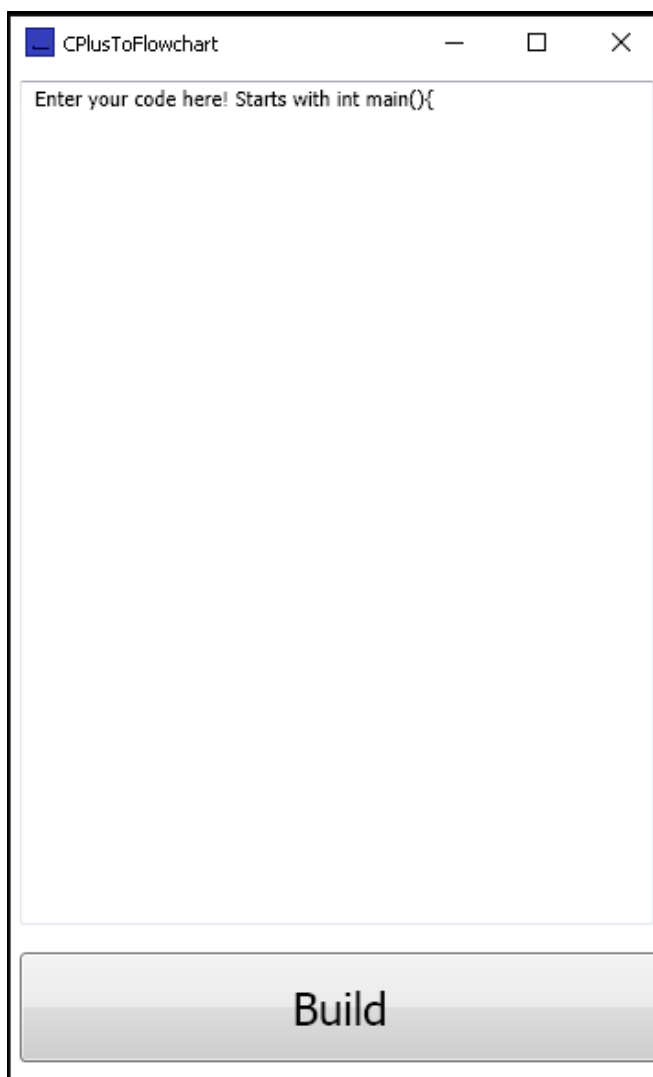


Рисунок 4 – Главное окно программы

Первым шагом перед работой транслятора, необходимо вставить текст исходной программы в поле ввода (рис. 5). Это может быть, как придуманный на ходу код, так и ранее разработанная программа.

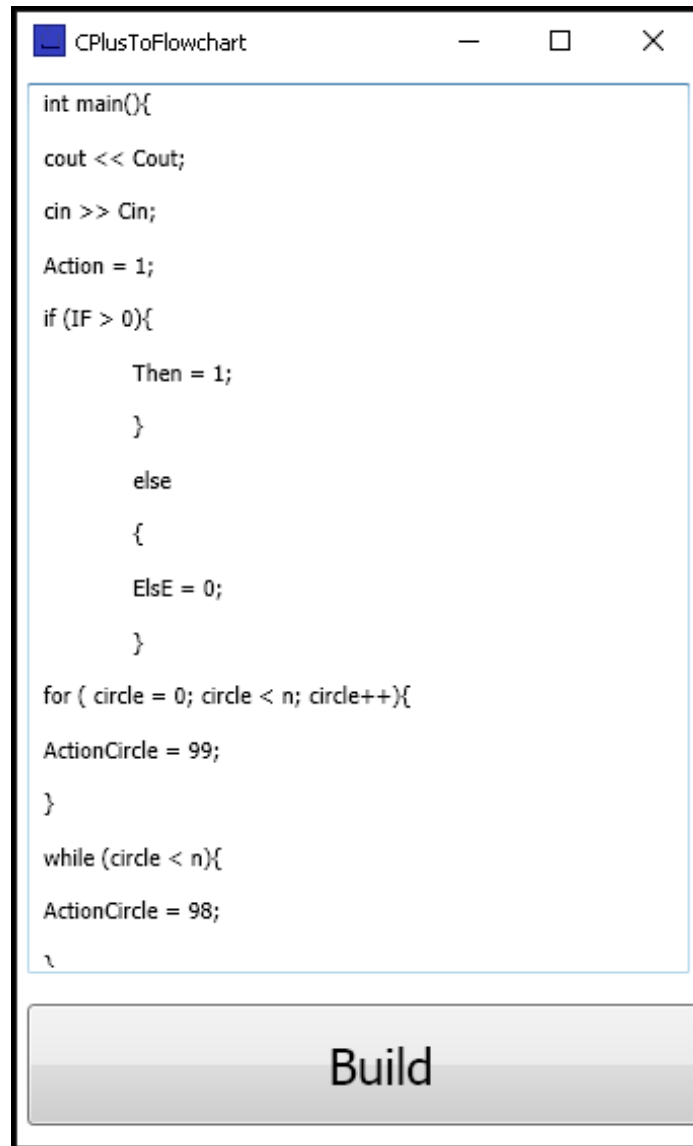


Рисунок 5 – Ввод исходной программы

Вторым шагом является нажатие на кнопку Build, после чего откроется диалоговое окно сохранения и пользователю нужно будет указать имя файла и его расположение для сохранения (Рис. 6).

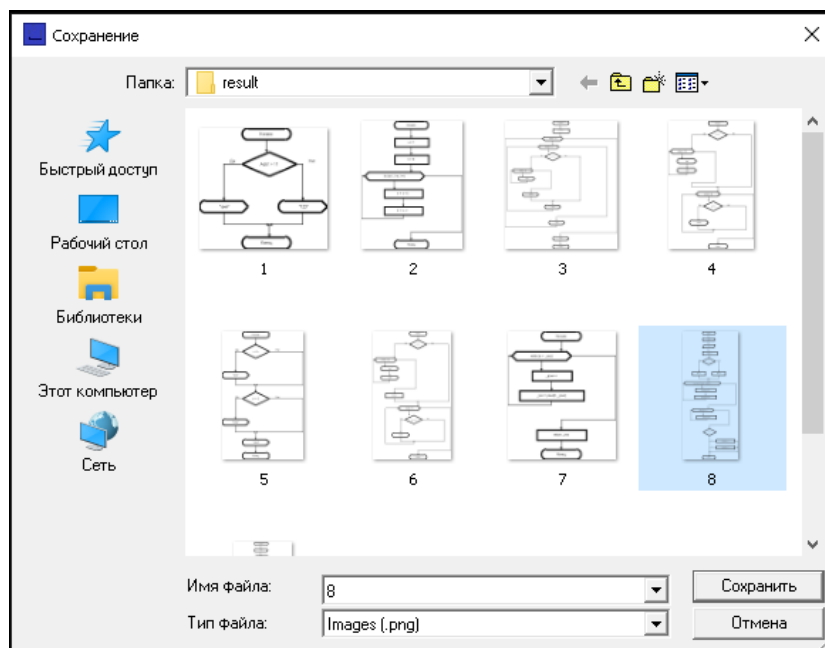


Рисунок 6 – Окно сохранения блок схемы

На этом работа пользователя и программы заканчивается. Далее пользователю нужно открыть расположение сохраненного png изображения чтобы проверить правильность работы его программы и построенной блок схемы на её основе.

ЗАКЛЮЧЕНИЕ

В результате курсовой работы по программной реализации транслятора с языка программирования C++ в целевой язык блок схем, с использованием объектно-ориентированного языка программирования C#, была разработана программа – транслятор. В процессе написания программы было выявлено, что при лексическом анализе выполняется разбиение последовательности входных символов на лексемы, что при графическом отображении в процессе построения изображения необходимо корректно обрабатывать последовательность лексем и их значений.

В ходе выполнения курсовой работы были получены навыки по проектированию программ, их отладке и документированию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wikipedia. Транслятор [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: [https://ru.wikipedia.org/wiki/Транслятор свободный](https://ru.wikipedia.org/wiki/Транслятор_свободный)
2. Wikipedia. Лексический анализ [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: [https://ru.wikipedia.org/wiki/Лексический_анализ свободный](https://ru.wikipedia.org/wiki/Лексический_анализ_свободный)
3. Молдованова О.В. Языки программирования и методы трансляции: учеб. пособие / О.В. Молдованова. – Новосибирск: ФГОБУ ВПО «СибГУТИ», 2012. – 39 с.
4. Молдованова О.В. Языки программирования и методы трансляции: учеб. пособие / О.В. Молдованова. – Новосибирск: ФГОБУ ВПО «СибГУТИ», 2012. – С. 61 - 65.
5. Молдованова О.В. Языки программирования и методы трансляции: учеб. пособие / О.В. Молдованова. – Новосибирск: ФГОБУ ВПО «СибГУТИ», 2012. – С. 79 - 97.