Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина Методы трансляции

|  |  |
| --- | --- |
|  |  |

ОТЧЕТ

к лабораторной работе № 4

на тему

**СЕМАНТИЧЕСКИЙ АНАЛИЗ**

Выполнил             В. П. Бычко

Проверил                          Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc157960226)

[2 Краткие теоритические сведения 4](#_Toc157960235)

[3 Результаты выполнения лабораторной работы 5](#_Toc157960235)

[Выводы](#_Toc157960236) 9

[Список использованных источников](#_Toc157960237) 10

[Приложение А (обязательное) Листинг програмного кода](#_Toc157960238) 11

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью данной лабораторной работы является создание семантического анализа дерева выражений, полученного с помощью синтаксического анализатора из лабораторной работы 3. Целью также является обнаружение некорректных ситуаций, возникающих при обработке неверной ветки синтаксического дерева.

# 2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Семантический анализ — выявление несоответствий типов и структур переменных, функций и процедур. Семантический анализ — это проверка смысловой правильности конструкции. Например, если в выражении используется переменная, то она должна быть определена ранее по тексту программы, а из этого определения может быть получен ее тип. Исходя из типа переменной, можно говорить о допустимости операции с данной переменной. Семантические ошибки возникают при недопустимом использовании операций, массивов, функций, операторов и прочее.

Семантический анализатор выполняет следующие основные действия: проверку соблюдения во входной программе семантических соглашений входного языка, дополнение внутреннего представления программы в компиляторе операторами и действиями, неявно предусмотренными семантикой входного языка, проверку элементарных семантических (смысловых) норм языков программирования, напрямую не связанных со входным языком.

Семантический анализатор использует синтаксическое дерево и информацию из таблицы идентификаторов для проверки входной программы на семантическую согласованность с определением языка программирования. Он также собирает информацию о типах и сохраняет её в синтаксическом дереве или в таблице идентификаторов для последующего использования в процессе генерации промежуточного кода. [1]

После синтаксического и семантического анализа исходной программы компиляторы генерируют низкоуровневое промежуточное представление входной программы, которое можно рассматривать как программу для абстрактной вычислительной машины. Такое промежуточное представление должно обладать двумя важными свойствами: оно должно легко генерироваться и легко транслироваться в целевой машинный язык. В результате семантических анализатор выдаст абстрактное синтаксическое дерево. Абстрактное синтаксическое дерево – это конечное, помеченное, ориентированное дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья с соответствующими операндами. АСД отличается от дерева разбора тем, что сохраняет только существенную информацию о программе, тогда как дерево разбора, напротив, – много избыточной информации. [2]

# 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Во время выполнения лабораторной работы был реализовал семантический анализатор языка С. На рисунке 3.1 изображен код корректной программы

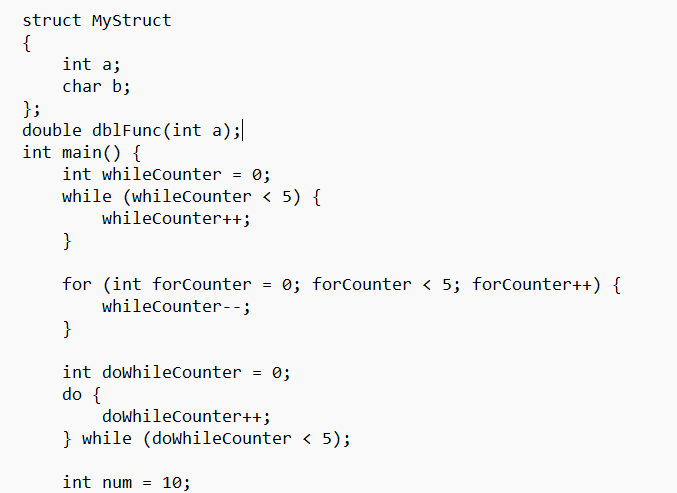


Рисунок 3.1 – Корректный код программы

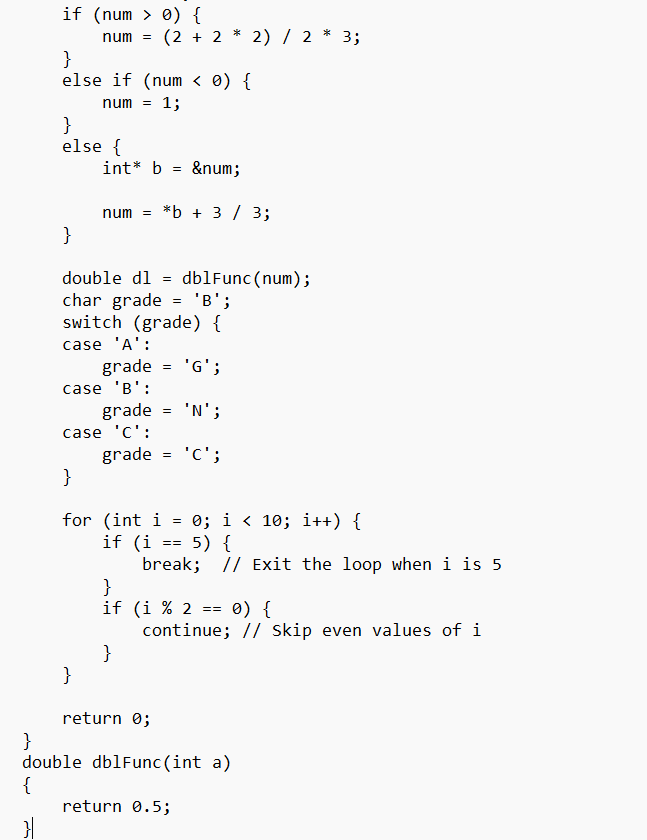


Рисунок 3.1 – Лист 2

На рисунке 3.2 изображен вывод программы, на котором выведено в консоль абстрактное синтаксическое дерево.

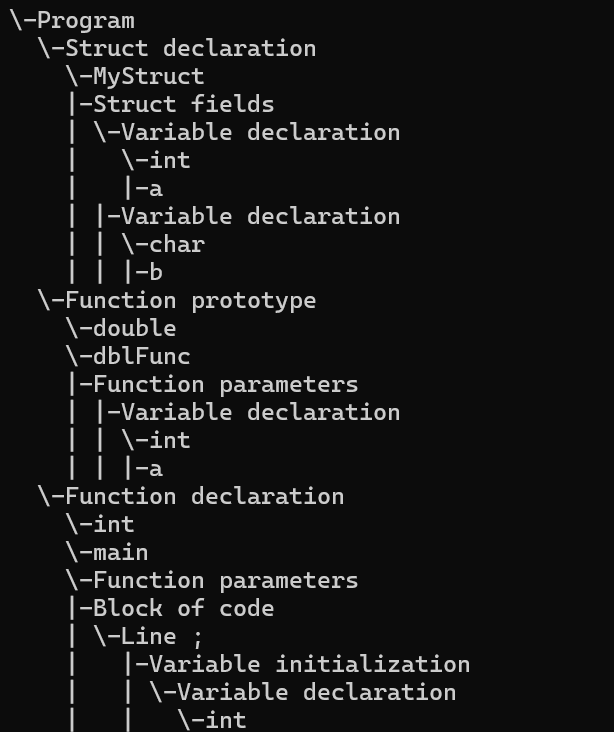


Рисунок 3.2 – Корректный результат программы

На рисунке 3.3 изображен блок кода с семантической ошибкой.

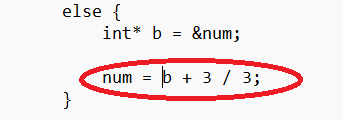


Рисунок 3.3 – Код с ошибкой

На рисунке 3.4 продемонстрировано, что программа выведет ошибку.



Рисунок 3.4 – Результат программы с ошибкой

## ВЫВОДЫ

В ходе выполнения лабораторной работы по написанию семантического анализатора был реализован семантический анализатор, способный проверять правильность построенное синтаксическое дерево на соответствие семантике языка Си.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Cемантический анализ [Электронный ресурс]. – Режим доступа:

https://myfilology.ru/165/yazyki-programmirovaniya-i-ix-ispolzovanie-v-informaczionnyx-sistemax/leksicheskij-sintaksicheskij-semanticheskij-analiz/– Дата доступа: 10.04.2024

1. АСТ дерево [Электронный ресурс]. – Режим доступа: https://pvs-studio.ru/ru/blog/terms/0004/ – Дата доступа: 10.04.2024

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг программного кода**

using MTran;

using static MTran.Compiler;

public class Program

{

public static void Main()

{

var filePath = "example.c";

string path = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory())!.Parent!.Parent!.FullName, filePath);

var code = File.ReadAllText(path);

try

{

var root = Compiler.SemanticAnalysis(Compiler.SyntaxAnalysis(new List<Compiler.Token>(Compiler.LexicalAnalysis(code))));

WriteTree(root, "", true);

}

catch (SemanticException ex)

{

Console.WriteLine(ex.Message);

}

}

public static void WriteTree(Node? node, string indent, bool last)

{

Console.Write(indent);

if (last)

{

Console.Write("\\-");

indent += " ";

}

else

{

Console.Write("|-");

indent += "| ";

}

if (node is EmptyNode)

{

return;

}

if (node is ValueNode tokenNode)

{

Console.WriteLine(tokenNode.Token.Value);

}

else if (node is OperatorNode operatorNode)

{

Console.WriteLine(operatorNode.Operator);

var enumerator = operatorNode.Children.GetEnumerator();

enumerator.MoveNext();

for (int i = 0; i < operatorNode.Children.Count(); i++)

{

WriteTree(enumerator.Current, indent, enumerator.MoveNext());

}

}

else if (node is TypesNode typesNode)

{

Console.WriteLine(string.Join(' ', typesNode.Types.Select(t => t.Value)));

}

}

}