Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы трансляции

ОТЧЁТ

по лабораторной работе

на тему

Семантический анализатор.

Выполнил

Студент гр. 053501

Волковский О.А.

Проверил

Ассистент кафедры информатики

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1. Цель работы 3](#_Toc128743288)

[2. Краткие теоретические сведения 4](#_Toc128743289)

[3. Примеры работы парсера 5](#_Toc128743290)

[4. Выводы 6](#_Toc128743291)

1. **ЦЕЛЬ РАБОТЫ**

Создать семантический анализатор для реализации возможности интерпретации программы на выбранном языке. Необходимо показать скриншоты нахождения 2-х семантических ошибок.

1. **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Семантический анализ является одним из основных этапов теории трансляции. Он представляет собой процесс проверки исходного кода на наличие семантических ошибок, которые не могут быть обнаружены на уровне лексического и синтаксического анализа.

Фаза контроля типов проверяет, удовлетворяет ли программа контекстным условиям. Главной составляющей контекстных условия является правильное использование программой типов данных, предоставляемых входным языком, т.е. корректность выражений, встречающихся в программе, с точки зрения использования типов.

Идентификация идентификаторов – одна из задач, решение которой необходимо для проверки правильности использования типов. Понятно, что мы не можем убедиться в правильности использования типов в какой-нибудь конструкции до тех пор, пока не определим типы всех ее составных частей. Например, для того, чтобы выяснить правильность оператора присваивания мы должны знать типы его получателя (левой части) и источника (правой части). Для того, чтобы выяснить, каков тип идентификатора, являющегося, например, получателем присваивания, мы должны понять, каким образом этот идентификатор был объявлен в программе.

Каждое вхождение идентификатора в программу является либо определяющим, либо использующим. Под определяющим вхождением идентификатора понимается его вхождение в описание, например, int i. Все остальные вхождения являются использующими, например, i = 5 или i+13.

Цель идентификации идентификаторов – определить тип использующего вхождения идентификатора.

1. **ПРИМЕРЫ РАБОТЫ ПАРСЕРА**

Рассмотрим следующую программу, для которой было построено синтаксическое дерево в предыдущей лабораторной (см. рисунок 1).

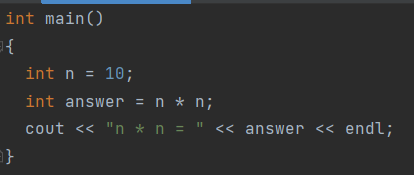


Рисунок 1 – Пример программы

Добавим, например, в выражение переменную, которая не определена (int answer = l \* n), в этом случае парсер отловит ошибку (см. рисунок 2).



Рисунок 2 – Пример вывода ошибки c неопределенной переменной

Добавим, например, определение переменной, но имя переменной будет является константным значением, в этом случае парсер отловит ошибку (см. рисунок 3).



Рисунок 3 – Пример ошибки с некорректным левым значением выражения

Если добавим определение новой переменной m(string m = “1”) и попробуем изменить логику answer(int answer = n \* m;), то парсер отловит ошибку несовместимости типов (см. рисунок 4).

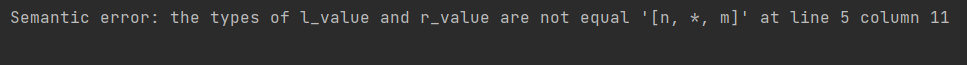


Рисунок 4 – Пример несовместимости типов

1. **ВЫВОДЫ**

Таким образом, реализация семантического анализатора из теории трансляции позволяет производить проверку исходного кода на наличие семантических ошибок. Это важный шаг в процессе компиляции, который помогает обнаружить ошибки, которые могут привести к неправильной работе программы или ее аварийному завершению.

Семантический анализатор выполняет несколько задач, включая проверку соответствия типов данных в операциях, обнаружение необъявленных переменных и функций в исходном коде и проверку правильного их использования.

В процессе работы анализатор может выявлять различные ошибки, такие как неправильное использование операторов и функций, приведение несовместимых типов переменных. Использование семантического анализатора позволяет повысить качество и надежность программного обеспечения, ускорить процесс разработки и снизить затраты на отладку и исправление ошибок.