УДК 004.424.5

Студ. О.А. Милешко
Науч. рук. ст. преподаватель А.С. Наркевич
(Кафедра программной инженерии, БГТУ)

РЕАЛИЗАЦИЯ ПРОСТОГО КОМПИЛЯТОРА

Задача транслятора сделать программу, написанную на некотором языке программирования, понятной компьютеру. Этого можно добиться одним из двух способов: компиляцией или интерпретацией.

Целью этой работы являлась реализация синтаксического анализа и трансляция математических выражений. В результате мы хотели видеть серию команд на языке подобном ассемблеру, выполняющую необходимые действия.

Компилятор переводит исходное выражение в эквивалентную программу на языке подобном ассемблеру. В случае неверного выражения сообщает об ошибке.

Процесс создания компилятора можно свести к решению нескольких задач, которые принято называть фазами компиляции. Наш компилятор состоит из следующих фаз:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;
- генерация кода.

Сформулируем основные цели каждой из фаз компиляции.

Лексический анализ. На этом этапе происходит проверка на разрешенные символы.

Синтаксический анализ. Получает на вход результат работы лексического анализатора и выполняет выделение синтаксических конструкций в тексте исходного выражения.

Семантический анализ. Заключается в проверке деления на 0, правильности типов данных, используемых в программе.

Генерация кода. Из промежуточного представления формируется код на языке подобном ассемблеру.

Для реализации вышеперечисленных этапов мы написали программу на языке C++, содержащую функции относящиеся к каждой из фаз.

Например, лексический анализ:

- IsAlpha(char c) – Распознаёт букву;
- IsDigit(char c) – Распознаёт число;
- IsAddop(char c) - Распознаёт знаки «+», «-»;

– GetNum() - Получить число.

Синтаксический анализ:

– Factor() – Распознаёт скобки и меняет приоритетность операций;
– Term() – Распознаёт термы и сохраняет результат;
– Expression() - Анализ и перевод выражения.

Семантический анализ

– Expected(strings) – Выдаёт ошибку и сообщает о том, что ожидалось;
– IsZero() – выполняет проверку деления на 0.

К генерации кода относятся функции

– Multiply() – Распознаёт и переводит операцию умножения;
– Divide() – Распознаёт и переводит операцию деления;
– Add() - Распознаёт и переводит операцию сложения;
– Subtract() - Распознаёт и переводит операцию вычитания.

В итоге у нас получилась программа, которая распознаёт математические действия и выводит команды на языке похожем на ассемблер.

Пример работы программы:

5*(4+6)/10+5 – входное выражение.

Результат работы программы:

```
MOVE #5,D0
MOVE D0,-(SP)
MOVE #4,D0
MOVE D0,-(SP)
MOVE #6,D0
ADD (SP),D0
MULS (SP)+,D0
MOVE D0,-(SP)
MOVE #10,D0
MOVE (SP)+,D1
DIVS D1,D0
MOVE D0,-(SP)
MOVE #5,D0
ADD (SP),D0
```

12/(3+2)*5-10 – входное выражение.

Результат работы программы:

```
MOVE #12,D0
MOVE D0,-(SP)
MOVE #3,D0
MOVE D0,-(SP)
MOVE #2,D0
ADD (SP),D0
MOVE (SP)+,D1
DIVS D1,D0
MOVE D0,-(SP)
MOVE #5,D0
MULS (SP)+,D0
MOVE D0,-(SP)
MOVE #10,D0
SUB (SP),D0
```

Здесь команда MOVE означает запись в регистр или стек, ADD выполняет сложение, SUB – вычитание, MULS – умножение, DIVS – деление.

В отличие от компилятора интерпретатор не создает никакой новой программы. Интерпретатор, так же, как и компилятор, анализирует входное выражение, а затем выполняет операции, содержащиеся в тексте этого выражения.

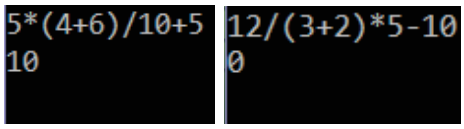
Этапы работы интерпретатора

- лексический анализ
- синтаксический анализ
- семантический анализ
- исполнение

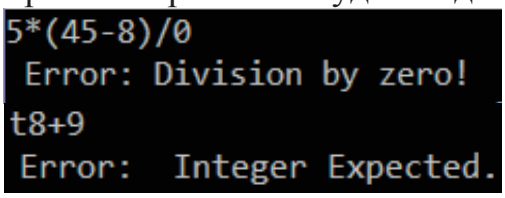
Этапы работы интерпретатора схожи с фазами компиляции за исключением последнего, здесь вместо генерации кода происходит исполнение входного выражения.

Преобразовав функции, написанные для компилятора мы получили интерпретатор, который будет распознавать и выполнять, а не переводить данные команды.

Примеры:



При вводе неверного выражения будет выдаваться ошибка:



Развивая эту программу далее (добавить переменные, функции, управляющие конструкции, булевы выражения, и т.п.), можно получить достаточно многофункциональный компилятор.

ЛИТЕРАТУРА

1. Д. Креншоу Пишем компилятор [Электронный ресурс] – Режим доступа: http://alexei-s1.narod.ru/books/pishem_compilator.pdf. – Дата доступа 20.03.2018.