**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

СМК 24/05-21

Срок хранения 5 лет

ВЛГУ 10.05.04.09.05.00 ПЗ

**П О Я С Н И Т Е Л Ь Н А Я З А П И С К А К   
 К У Р С О В О Й Р А Б О Т Е**

«Компилятор подмножества процедурного языка в ассемблер»

Дисциплина: Прикладные алгоритмы

Направление: 10.05.04 – Информационно-аналитические системы безопасности

СПИСОК ИСПОЛНИТЕЛЕЙ

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н., доц. каф. ИЗИ Монахов Ю.М. |
|  |
| Исполнитель |  | студент гр. ИСБ-118 Михайлов В.С. |

# АННОТАЦИЯ

В данном программном документе проведен код компилятора подмножества процедурного языка. Код программы реализован на языке программирования Python в интегрированной среде разработки PyCharm.

Разработанная программа состоит из лексического анализатора, синтаксического анализатора и генератора объектного кода.

Лексический анализатор считывает текстовый файл, разбирает содержимое на токены и, затем, направляет их синтаксическому анализатору.

В свою очередь синтаксический анализатор сопоставляет поток токенов с описанной грамматикой.

Генератор объектного кода переделывает полученную программу в последовательность инструкций, которые в дальнейшем могут выполняться на машине.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 3](#_Toc73495709)

[1. ЭТАПЫ РАЗРАБОТКИ КОМПИЛЯТОРА 5](#_Toc73495710)

[1.1. Основные требования 5](#_Toc73495711)

[1.2. Лексер 5](#_Toc73495712)

[1.3. Синтаксический анализатор 7](#_Toc73495713)

[1.4. Таблица символов 8](#_Toc73495714)

[1.5. Построение генератора объектного кода 9](#_Toc73495715)

[2.ПРОВЕРКА НА СООТВЕТСВИЕ ОСНОВНЫМ ТРЕБОВАНИЯМ 11](#_Toc73495716)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ: 16](#_Toc73495717)

[**ПРИЛОЖЕНИЕ А -** Исходный код программы 17](#_Toc73495718)

# ЭТАПЫ РАЗРАБОТКИ КОМПИЛЯТОРА

## 1.1. Основные требования

Разработка будет производиться в соответствии со следующими требованиями:

Требования к входному языку:

1. Должны присутствовать операторные скобки;
2. Должна игнорироваться идентация программы;
3. Должны поддерживаться комментарии любой длины;
4. Входная программа должна представлять собой единый модуль, но также должна быть поддержка вызова функций.

Требования к операторам:

1. Оператор присваивания;
2. Арифметические, логические операторы;
3. Условный оператор (ЕСЛИ);
4. Операторы цикла (break, continue);
5. Базовый вывод;
6. Типы (целочисленный, вещественный).

Требования к выходному языку:

Ассемблер для архитектуры x86.

## 1.2. Лексер

На начальном этапе содержимое текстового файла разбивается на токены в соответствии с описанными правилами лексического анализатора. Результатом работы является последовательный список всех найденных лексем.

В текстовом файле имеются операторы присваивания, цикла, условные и логические, арифметика, типы и функции. Также присутствуют комментарии, которые игнорируются при разборе.

На рисунке 1 представлен небольшой пример правил разбора лексем из текстового файла.

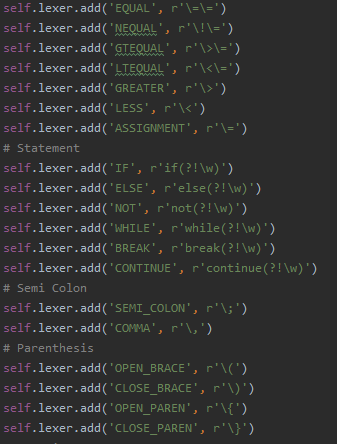


Рисунок 1 – Лексема

А на рисунке 2 представлен пример полученных токенов.

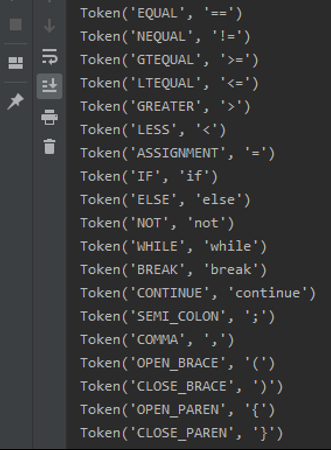


Рисунок 2 – токены

После этого полученный поток токенов направляется в синтаксический анализатор.

## 1.3. Синтаксический анализатор

Второй стадией компилятора является синтаксический анализ. На вход синтаксическому анализатору подаётся набор токенов из лексического анализатора.

Для каждого токена описаны определённые наборы правил, по котором синтаксический анализатор их может обработать. При помощи этих правил, токены преобразуются в синтаксическое дерево разбора. После завершения, итоговый результат отправляется транслятору для дальнейших преобразований.

Правила использования языка.

Функции

Функция включает в себя ключевое слово func, название функции и перечисление параметров в скобках (), после этого идет блок, содержащий выражение {} пример: func main(int a, int b){…}

Выражения

В грамматике определено несколько выражений: объявление переменной, присваивание, if/else конструкция, while конструкция, вызов функции, вывод print.

1. Объявление переменной состоит из объявления типа, названия переменной и значения (может отсутствовать), пример: int a; float b = 3.0;
2. Присваивание состоит из объявления типа, имени переменной и присваивания ей значения пример:int a = 3 + 3;
3. if/else конструкция состоит из объявления ключевого слова if, логического выражения внутри фигурных скобок (), выражений внутри блока {} и ключевого слова else и блока выражений {} пример: if (a > b){ ... } else { ... }
4. while конструкция состоит из объявления ключевого слова while, логического выражения внутри фигурных скобок (), выражений внутри блока {} пример: while (a > b) { ... }
5. Вызов функций осуществляется посредством указывания имени функции, и скобок с переменными, если они необходимы () пример: main(15);
6. Вызов встроенных функций осуществляется посредством указывания имени функции, скобок с переменными пример: sum(1, 3), sub(5, 2);
7. Вывод осуществляется посредством указывания ключевого слова print и скобок () внутри которых перечисление параметров через запятую, которые необходимо вывести пример: print(a, b, c);

## 1.4. Таблица символов

Таблицу символов создает генератор объектного кода.

Пример таблицы символов:

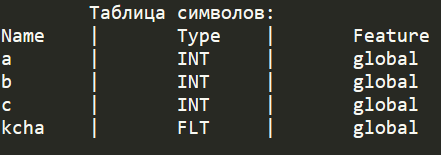


Рисунок 5 – Таблица символов

## 1.5. Построение генератора объектного кода

При синтаксическом анализе классы вершин синтаксического дерева добавляются к корневой вершине global. Данные синтаксического анализатора продолжают обработку дальше методами генерации при помощи библиотеки llvmlite.

Благодаря данной библиотеке он создает промежуточный код «LLVM IR» для его дальнейшего преобразования в assembler.

Пример промежуточного кода.



Рисунок 3 – Входная программа

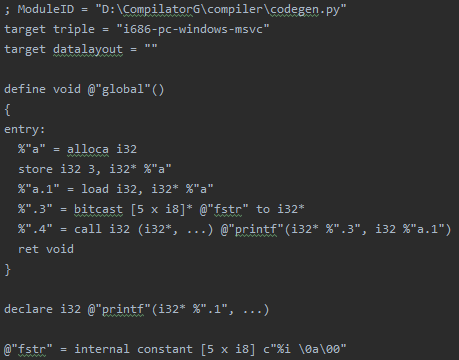


Рисунок 4 – Промежуточный код

# 2.ПРОВЕРКА НА СООТВЕТСВИЕ ОСНОВНЫМ ТРЕБОВАНИЯМ

При проектировании компилятора к основному языку были установлены следующие минимальные требования: наличие операторных скобок, игнорирование пробелов и идентации программы, поддержка многострочных комментариев и вызова функций. Наличие операторов присваивания, условных, цикла, арифметических, логических. Должны присутствовать два типа данных.

Далее приведено тестирование компилятора.

Вывод числа последовательности Фибоначчи. Данный пример демонстрирует:

* Операторы присваивания
* Арифметика
* Операторы цикла (while)
* Базовый вывод переменных и строковых литералов

Исходный код программы:

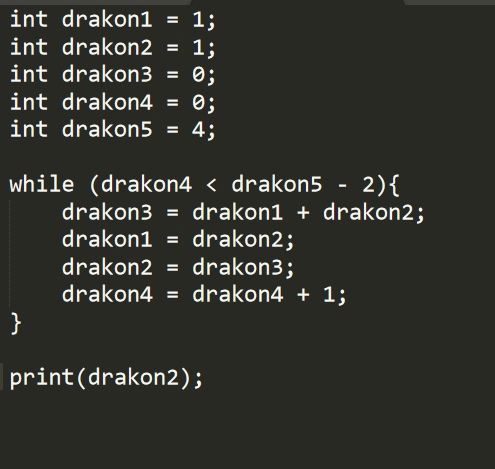


Рисунок 6 – Исходный код входящей программы

Список токенов:



Рисунок 7 – Список сгенерированных токенов

Таблица символов:

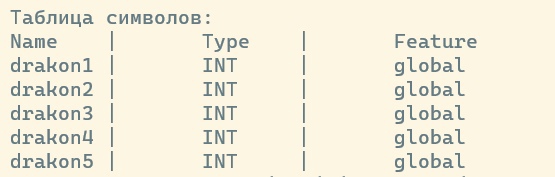


Рисунок 8 – Сгенерированная таблица символов

Сгенерированный код:

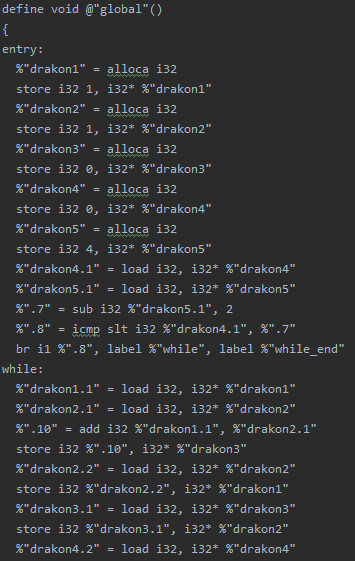


Рисунок 9 – Сгенерированный код программы, выходной

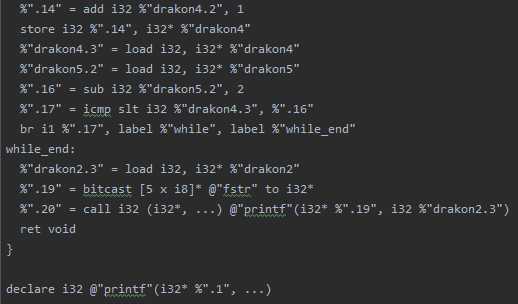


Рисунок 10 – Сгенерированный код программы, выходной

Результат работы программы:

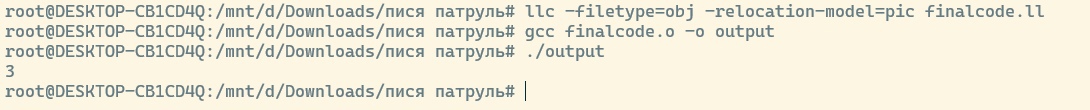


Рисунок 11 – Результат работы программы

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ:

1. Создаем свой язык программирования с блэкджеком и компилятором. Электронный ресурс. URL: https://proglib.io/p/your-own-programming-language/ (дата обращения: 13.03.2021);
2. Monica S. Lam, Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman Compilers: Principles, Techniques, and Tools // Scott Ullman of Strange Tonic Productions 2007 // ISBN 0-321-48681-1, 1184 p.;
3. Symbol table. Электронный ресурс. URL: https://www.tutorialspoint.com/compiler\_design/compiler\_design\_symbol\_table.htm (дата обращения: 24.03.2021);
4. Жуков А.В., Авдюхин А.А. Ассемблер. // Спб.: БХВ-Петербург, 2003. // ISBN 5-94157-133-Х // 448 с.

# **ПРИЛОЖЕНИЕ А -** Исходный код программы

Исходный код программы (ссылка на github репозиторий):

https://github.com/MilLer931/CompilerG