Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Курсовая работа**

**Дисциплина**: Прикладное программирование

**Тема:** Интерпретатор BASIC

Выполнил студент гр. 23531/3 К.В. Шляжко

(подпись)

Преподаватель Т.Л. Сидорина

(подпись)

« » \_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Оглавление

[**1.** **Формулировка задачи** 3](#_Toc518265487)

[**2.** **Описание работы** 3](#_Toc518265488)

[**3.** **Результат работы** 4](#_Toc518265489)

[**4.** **Выводы** 5](#_Toc518265490)

[**5.** **Список используемой литературы** 5](#_Toc518265491)**.**

# **Формулировка задачи**

Реализовать интерпретатор BASIC

Минимальные требования:

* целочисленные переменные;
* инфиксные выражения c “+”, “-“, “\*”, “/”, “%”, скобками
* операторы сравнения “<”, “=”, “>”, “<=”, “<>”, “>=”
* операторы INPUT, PRINT, GOTO, GOSUB/RETURN, IF

Интерпретатор BASIC должен уметь анализировать, обрабатывать и выполнять исходный код программы или запроса.

# **Описание работы**

* 1. **Алгоритм**
     + Интерпретатор принимает на вход имя файла с исходным кодом программы
     + Анализ синтаксиса с записыванием в память информации о переменных и метках
     + В память посимвольно считывается весь код программы при удачном анализе
     + Читается лексема и определяется ее тип
     + Определяются и выполняются соответствующие действия
     + Если не достигнуто условие завершения программы, читается следующая лексема и повторяется пункт 5
  2. **Подробное описание алгоритма**

Принимается файл, посимвольно анализируется и сканируется на наличие меток и инициализирует переменные. При удачном анализе файл записывается в память, при неудачном - выводит ошибку с описанием проблемы. Далее в основном теле цикла интерпретатора, который выполняется пока не достигнут конец полученной на вход программы, читается лексема и определяется ее тип (DELIMITER, VARIABLE, NUMBER, COMMAND, QUOTE). Если DELIMITER, то читается следующая лексема (этот тип пригодится, для работы других функций). Если VARIABLE, то выполняется присваивание. Константа просто присваивается; выражение вычисляется с использованием метода рекурсивного спуска, а его результат записывается в переменную, далее читается следующая лексема. COMMAND – действия зависят от тип команды:

* INPUT – определяется сколько параметров подано на вход. Если подан один параметр, то он трактуется как число и записывается в соответствующую переменную, иначе выдается ошибка. Если подано два параметра, то первый считается строкой, которая выводится в консоль, а второй так же трактуется как переменная, иначе выдается ошибка. Все параметры записываются через запятую.
* PRINT – может получать любое количество параметров (константы, переменные, алгебраические выражения, строки), записанных через запятую в одну строку. Выражения вычисляются так же, как и в пункте выше. Если при считывании очередной лексемы возвращается тип QUOTE, то ищется закрывающая кавычка и строка выводится в консоль.
* IF – имеет вид IF... THEN..ENDFI, или IF .. THEN .. ELSE .. ENDFI
* GOTO – считывается метка и проверяется, есть ли такая метка в памяти, если нет – выдается ошибка, иначе интерпретатор переходит к указанному месту.
* GOSUB/RETURN – считывается метка и проверяется, есть ли такая метка в памяти, если нет – выдается ошибка, иначе исходное место запоминается в стек переходов интерпретатор продолжает работу с указанного места. Как только встречается RETURN из стека переходов достается последнее запомненное место и выполнение продолжается с этой позиции.

Как только встречается символ конца программы, интерпретатор прекращает свою работу.

При выполнении курсовой работы опирался на статью «Интерпретаторы языка», написанную по учебнику Шилдта «C для профессиональных программистов». Из статьи позаимствовал идею определения типа и внутреннего представления лексем (отражено в constants.h, а также частично в executor.c), метод рекурсивного спуска также был позаимствован и изменен мной (в файле executor.c), логика чтения файла и загрузка файла была тоже позаимствована (в файле main.c).

# **Результат работы**

* 1. **Исходный код**

Можно посмотреть на GitHub: https://github.com/Kir86russ/BasicInterpretator

* 1. **Примеры работы**

|  |  |
| --- | --- |
| **Программа на BASIC** | **Результат работы интерпретатора** |
| a = 10 INPUT "Enter var b = ", b IF a < b + a THEN PRINT "Hello"  ELSE PRINT "no hello" ENDFI  GOTO 100  a = 999 b = 666  100 PRINT a , " ", b END | Hello  10 54 |

|  |  |
| --- | --- |
| **Программа на BASIC с синтаксической ошибкой** | **Результат работы интерпретатора** |
| a = 10 INPUT "Enter var b = ", b IF a < b + a PRINT "Hello" ELSE PRINT "no hello" ENDFI GOTO 100 a = 999 b = 666 100 PRINT a , " ", b END | Syntax error! No way end of line, while not THEN Line: 3 |

# **Выводы**

В ходе работы был реализован интерпретатор языка BASIC. Получены бесценные навыки написания программ на языке Си.

# **Список используемой литературы**

* + - Шилдт Г. «C для профессиональных программистов»