Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

**Кафедра «Компьютерные системы и программные технологии»**

**КУРСОВОЙ ПРОЕКТ**

**Интерпретатор SmallBasic**

по дисциплине «Прикладное программирование»

Выполнил

студент гр.23531/1 Д.И.Графов

Руководитель М.А.Петров

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Санкт-Петербургский государственный политехнический университет

**ЗАДАНИЕ**

**НА ВЫПолнение курсовоГО ПРОЕКТА**

студенту группы 23531/1 Графову Денису Ивановичу

***1. Тема проекта (работы):*** Интерпретатор SmallBasic

***2. Срок сдачи законченного проекта (работы)*** 31.05.18 г.

***3. Исходные данные к проекту (работе)***:

Реализовать интерпретатор выбранного диалекта Basic. Выполнить минимальные требования:

­ целочисленные переменные;

­ инфиксные выражения с “+”, “­“, “\*”, “/”, скобками;

­ операторы сравнения “<”, “<=”, “=”, “<>”, “>=”, “>”;

­ операторы TextWindow.Write(), TextWindow.Read(), If, GoTo, Sub.

Разработка кода и документации должна вестись в репозитории git.

Разработанное ПО должно собираться с помощью команды make компилятором gcc (mingw) с опциями -std=c11 -pedantic -Wall -Wextra без предупреждений компилятора.

***4. Содержание пояснительной записки***: введение, основная часть (раскрывается структура основной части), заключение, список использованных источников, приложения.

***Дата получения задания***: «5» апреля 2018 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А.Петров

*(подпись)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.И.Графов

*(подпись студента)*

05.04.2018г.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . …………….. | 4 |
| 1. Алгоритм работы…….……………………………………………............................ | 5 |
| 2. Структура интерпретатора………………..……………………................................ | 6 |
| 3. Код программы…………..………………..……………………................................ | 7 |
| 3.1. main.c ………………………………………………………………………. | 7 |
| 3.2. Основной цикл программы из файла analyzer.c ……………..…….......... | 8 |
| 3.3. Makefile…….…………………………...………………………….............. | 9 |
| 4. Методика испытаний……………………………………………………….. | 10 |
| Заключение ...................................................................................................................... | 11 |
| Список использованных источников…......................................................................... | 12 |

**ВВЕДЕНИЕ**

В данном курсовом проекте мною будет написана программа, являющаяся интерпретатором SmallBasic, что соответствует моему индивидуальному заданию по прикладному программированию.

В ней будут реализованы основные возможности интерпретатора, такие чтение команд из входных файлов, хранение и использование целочисленных переменных, обработка инфиксных выражений, операторов сравнения, а также выполнение команд из входных файлов.

1. **Алгоритм работы интерпретатора**

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

Алгоритм работы:

1) прочитать инструкцию;

2) проанализировать инструкцию и определить соответствующие действия;

3) выполнить соответствующие действия;

4) если не достигнуто условие завершения программы, прочитать следующую инструкцию и перейти к пункту 2.

Для выполнения пункта 2 создан синтаксический анализатор выражений, который разбивает исходное выражение на составные части, компоненты (неделимые элементы выражения – лексемы).

Функция, разбивающая выражение на составные части, решает несколько задач:

1) игнорировать пробелы и символы табуляции;

2) извлекать каждую лексему из текста;

3) определять тип лексемы.

Анализатор использует шесть типов лексем: DELIMITER, VARIABLE, NUMBER, COMMAND, STRING, MARK (разделитель, переменная, число, команда, строка, метка).

Все ключевые команды имеют внешнее представление (строковое) и внутреннее (число).

Также при создании интерпретатора были учтены особенности и грамматика языка. Например, все числа – целые (integers: -32767...32767).

**Формат входных данных:**

Входным файлом является текстовый файл с программой на языке SmallBasic.

Имя данного файла передается в программу (интерпретатор) как аргумент командной строки.

Пример: **HelloWorld.sb**

|  |
| --- |
| TextWindow.Write("Hello, ") TextWindow.WriteLine("World!") |

**Результат работы интерпретатора:**

Выполнение команд входной программы

Пример: Результат выполнения **HelloWorld.sb**

Cmd



1. **Структура интерпретатора**

Проект состоит из директории **\main**, где хранятся файлы, реализующие Интерпретатор. В **\test** содержаться тестовые файлы для проверки работоспособности программы. **\result** – место сборки программы с помощью **Makefile** (SmallBasic\_Interpreter.exe). Также в проекте находится CMakeLists.txt и упомянутый ранее Makefile.

Программа состоит из 4 файлов (main.c, main.h, lexemes.h, analyzer.c).

В главном файле **main.c** происходит считывание аргументов командной строки и их обработка. Допустимый вариант аргументов: <executable>.exe <fileName>.sb. При некорректных входных аргументах программа выдаст ошибку и выведет пример допустимых аргументов.

Далее в файле main.c происходит обработка входных аргументов, выделение памяти, инициализация состояния интерпретатора. Функция loadProgram(char \*) считывает входной файл, записывает в память и при необходимости выделяет еще памяти.

Далее вызывается функция start(), которая содержится в файле **analyzer.c**. В данной функции находится главный цикл программы. Также в Analyzer.c содержится рекурсивный синтаксический анализатор, который состоит из функций level1, level2, level3, level4, level5 и вспомогательных value(для определения типа переменная/число), unary(для установки знака), arithmetic(для выполнения математических операций). При их реализации частично использовался код из источника [7].

Функция getExp для входа в синтаксический анализатор. Также в analyzer.c содержится главная функция анализатора, которая определяет тип лексемы, устанавливает ее внутреннее представление и сохраняет ее в динамически расширяемую память для дальнейшей обработки лексемы. В этом файле находятся основные структуры, необходимые для работы программы, настройки интерпретатора, а также функции инициализации, отчистки памяти и различные вспомогательные функции (в том числе для обработки ошибок).

Виды ошибок:

* Syntax error
* Brackets requires
* Assignment needed
* Then required
* Undefined label

Все ошибки указывают на причину сбоя и завершают работу интерпретатора.

1. **Код программы**
   1. **main.c**

|  |
| --- |
| #include **<stdio.h>** #include **<stdlib.h>** #include **<setjmp.h>** #include **<mem.h>** #include **<ctype.h>** #include **"main.h"** #define **PROG\_SIZE** 16  *//Объявление переменных* **char** \*program;  *//Объявление функций* **int** loadProgram(**char** \*); *//Считывает программу* **int** main(**int** argc, **char** \*argv[]) {  **char** \*file\_name = argv[1]; *//Имя файла программы* **if** (argc != 2) {  printf(**"Use format: <executable file>.exe <program file>.sb"**);  exit(1);  }   *//Выделение памяти для программы* **if** (!(program = (**char** \*) malloc(**PROG\_SIZE**))) {  printf(**"Error allocating memory"**);  exit(1);  }   *//Загрузка программы* **if** (!loadProgram(file\_name)) exit(1);   start(program);  **return** 0; }  **int** loadProgram(**char** \*fname) {  FILE \*file;  **if** (!(file = fopen(fname, **"r"**)))  **return** 0;  **char** \*point = program;  **int** i = 0, k = 1;   **do** {  \*point = (**char**) getc(file);  point++;  i++;  **if** (i == k \* **PROG\_SIZE**) {  k++;  program = (**char** \*) realloc(program, (size\_t) (k \* **PROG\_SIZE**));  point = program;  point += i;  }  } **while** (!**feof**(file));  \*(point - 1) = 0;  fclose(file);  **return** 1; } |

* 1. **Основной цикл программы из файла analyzer.c**

|  |
| --- |
| **void** start(**char** \*p) {  program = p;  scanLabels();   **do** {  getToken();  *//Проверка на присваивание* **if** (token.type == **VARIABLE**) {  **if** (findSub(token.name) != **NULL**) {  *//Проверка на вызов функции* **if** (\*program == **'('**) {  program++;  **if** (\*program == **')'**) {  program++;  size\_t len = strlen(findSub(token.name)) + strlen(program);  **char** \*cpy = malloc(**sizeof**(**char**) \* (len + 1));  strcpy(cpy, findSub(token.name));  cpy[len] = **'\0'**;  program = strcat(cpy, program);  *//puts(program);* } **else** printError(**"Unpaired brackets when calling a function"**);  } **else** printError(**"Syntax error: undefined variable or function"**);  } **else** {  putBack(); *//откатиться на 1 лексему* setAssignment();  }  }   *//Проверка на команду* **if** (token.type == **COMMAND**) {  **switch** (token.id) {  **case WriteLine**:  printLine();  **break**;  **case Write**:  print();  **break**;  **case If**:  sbIf();  **break**;  **case Else**:  skipElse();  **break**;  **case Goto**:  sbGoto();  **break**;  **case Sub**:  setSub();  **break**;  **default**:  **break**;  }  }  } **while** (token.id != **FINISHED**); } |

**3.3. Makefile**

|  |
| --- |
| all: SmallBasic\_Interpreter SmallBasic\_Interpreter : result/main.o result/analyzer.o  gcc -std=c11 -pedantic -Wall -Wextra -o result/SmallBasic\_Interpreter result/main.o result/analyzer.o  result/main.o : main/main.c  gcc -std=c11 -pedantic -Wall -Wextra -c -o result/main.o main/main.c  result/analyzer.o : main/analyzer.c  gcc -std=c11 -pedantic -Wall -Wextra -c -o result/analyzer.o main/analyzer.c |

**4. Методика испытаний**

Работоспособность интерпретатора, а также его соответствие минимальным требованиям ТЗ были проверены с помощью набора тестовых файлов: HelloWorld.sb, Goto.sb, factorial.sb, test.sb.

Пример с программой “Hello, World!” был приведен в п.1.

|  |  |  |
| --- | --- | --- |
| **Файл на языке SmallBasic** | **Исходный код** | **Результат работы** |
| Goto.sb | Hello: x = 1 Start: TextWindow.WriteLine(x) x = x + 1 If x <= 5 Then  Goto Start EndIf Goto Finish a = TextWindow.Read() b = 10 TextWindow.WriteLine(a - b/2) Finish: TextWindow.WriteLine("Xex") | 1  2  3  4  5  Xex |
| factorial.sb  (введено число 5) | a = TextWindow.Read() f = 1 Start: If a > 1 Then  f = f \* a  a = a - 1  Goto Start EndIf TextWindow.Write("Result: ") TextWindow.WriteLine(f) | Result: 120 |
| test.sb | Goto HiMark Continue: a = 10 c = a Sub hello  TextWindow.WriteLine("I did not hit her!")  b = c + 10 EndSub Sub what  TextWindow.WriteLine("What?")  hello() EndSub hello() what() TextWindow.WriteLine("It's not true!!!") TextWindow.Write("a + b = ") TextWindow.WriteLine(a + b) TextWindow.Write("c = ") TextWindow.WriteLine(c) Goto Finish HiMark: TextWindow.WriteLine("Oh, Hi Mark!") Goto Continue Finish: | Oh, Hi Mark!  I did not hit her!  What?  I did not hit her!  It's not true!!!  a + b = 30  c = 10 |

## ЗАКЛЮЧЕНИЕ

В данном курсовом проекте мною была написана программа, реализующая интерпретатор SmallBasic.

Работоспособность интерпретатора была проверена набором тестовых файлов.

Таким образом, программа полностью соответствует заданным в индивидуальном задании требованиям.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Документация по SmallBasic

(<http://smallbasic-publicwebsite-dev.azurewebsites.net/Pages/DocumentReference.aspx>)

1. Информация по make ([http://pubs.opengroup.org/onlinepubs/9699919799/utilities/make.html#tag20761304](http://pubs.opengroup.org/onlinepubs/9699919799/utilities/make.html%23tag20761304))
2. Применение make

(<https://habr.com/post/211751/)>

1. Общая информация об интерпретаторах

(<http://cpp.com.ru/shildt_spr_po_c/01/0105.html>)

1. M. Уэйт, С. Прата, Д. Мартин - Язык Си руководство для начинающих [Электронный ресурс] 1988г. – Режим доступа: <http://storage.ded32.net.ru/Lib/TX/CPrimerPlus.pdf>
2. Стандарт языка C[11]

(<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf>)

1. Самоучитель с – Интерпретатор small basic

(<http://tigrazone.narod.ru/smallbasic.html>)