Compiler for HHPS

💻

Course project for *CS327-Compilers* course at IITGN.

Contributors :
Harsh Patel - 18110062
Harshit Kumar - 18110063
Pushkar Mujumdar - 18110132
Shivam Sahni - 18110159

This is a compiler - a toy of sorts - written using `flex`, `bison` and `C` for our custom defined language `HHPS`.

Contents

📄

- About the Language
- Usage instructions
- Feature Checklist
- Basic Structure
- Lexicon and Syntax
    - Keywords
    - Special Characters
    - Arithmetic Operators
    - Logical Operators
    - Reserved names
- Grammar
- Directory Structure
- Sample Programs
- References

About the Language

ℹ

`HHPS` is a language based on the 2 most widely used programming languages : `C` and `Python` . `HHPS` is an attempt to bring out the best of both these languages and combine them into a single, easy to use language.

It is a language with simple and intuitive syntax and grammar. It can be used by coders of any level, be it beginner, intermediate or experienced.

Usage instructions

For building the project and getting the executable `a.exe` (`a.out` in case of Linux)

```
make
```

For compiling the code from `test/factorial.hhps` (could be any path)

```
a.exe < test/factorial.hhps
```

The output `MIPS` code will be generated in `asmb.asm`

For deleting the build:

- Linux

```
make clean
```

- Windows

```
make clean_win
```

Feature checklist:

- [x]
  - [x] Integer data type
  - [x] Relational operators
  - [x] Arithmetic operators
  - [x] If-else conditionals
  - [x] Nested conditionals
  - [x] While loop
  - [x] Nested while loops
  - [x] For loop (similar to python)
  - [x] Nested For loop
  - [x] Arrays
  - [x] Function calls
  - [x] Return statements
  - [x] Verbose error reporting (with line number) ❗
  - [x] Multi-line comments
  - [x] Input feature
  - [x] Print statement for output
  - [x] Print statement with newline

Basic structure

🛠

Any program written in `HHPS` has 2 major blocks :

```
** Function Declaration Block**
Any user-defined functions needed in the program should be declared here.
If none, this block can be skipped.
```

```
** Main Function **
The main function is similar to the one in C.
The main function is the first piece of code that is executed by the operating
system when it runs the program.
```

Lexicon and Syntax

❓

For the Lexical Analyser, refer tok.l

Identifiers : Combination of lower and upper case alphabets and _ (except the keywords).

Multi-line comments : Anything between `/*` and `*/`

Keywords

- `if` Conditional statement
- `else` Alternate Condition
- `for` Iterative Loop
- `while` Conditional Loop
- `range` Range
- `array` Array
- `int` Integer
- `decl` Declare function
- `main` Main function
- `return` Function Return
- `print` Print to console
- `println` Print with newline

Reserved names

- `fun_` Should be the prefix of the function names

Any function should have the above stated prefix in the function name, and any lexeme with the prefix will be treated as a function name.

Special Characters

- `\n` Newline Character (error reporting)
- `;` End of Statement

Arithmetic Operators

- `+` Addition
- `-` Subtraction / Unary Minus
- `*` Multiplication
- `/` Division
- `%` Modulo
- `&` Bitwise AND
- `|` Bitwise OR

Logical Operators

- `<` Less than
- `>` Greater than
- `<=` Less than or Equal to
- `>=` Greater than or Equal to
- `!=` Not equal to
- `==` Equal to
- `&&` Logical AND
- `||` Logical OR

Grammar

For the complete Grammar rules, refer the Parser Generator hhps.y

A statement can be :

- Variable Declaration `int b = 5;`
- Variable Assignment `a = b + 1;`
- Print Statement `print(c);`
- Return Statement `return 0;`
- Conditional Statement (If-Else) `if(x < 10){`
- For Loop `for i range(0,10){`
- While Loop `while(i < j){`
- Array Declaration `array (int, 10) a;`
- Function Declaration `decl int fun_square(){`
- Function call with parameters `x` and `y`
  `fun_test(x=5, y=4);`

Directory Structure:

```
├──hhps.y
├──hhps.h
├──main.c
├──Makefile
├──README.md
├──README.txt
├──tok.l
└──test
    ├──array_print_expressions.hhps
    ├──array_with_while.hhps
    ├──array_with_while_input.hhps
    ├──bubblesort.hhps
    ├──conditionals.hhps
    ├──error.hhps
    ├──factorial.hhps
    ├──for_loop.hhps
    ├──func_test.hhps
    ├──if-else.hhps
    ├──if-else_input.hhps
    ├──nested_for.hhps
    ├──nested_while.hhps
    ├──remainder.hhps
    ├──return_test.hhps
    └──while_loop.hhps
```

Sample Programs:

Sample code for **Bubblesort** written in HHPS

```
int main(){
    array (int, 10) a;
    for i range(0, 10){
        a[i] = 10-i;
    }
    for i range(0, 10){
        int t = 9-i;
        for j range(0, t){
            if (a[j] > a[j+1]){
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
```

```
        }
    }

    for i range(0, 10){
        println(a[i]);
    }

    return 0;
}
```

For sample programs covering all features, refer the test folder.

References:

- Code used in Compilers Labs {9,10,11}
- https://www.gnu.org/software/bison/manual/html_node/Mfcalc-Symbol-Table.html
- MIPS_Instruction_Set.pdf (unive.it)
- Compiler Construction using Flex and Bison, Anthony A. Aaby