Basic BASIC Language Specification

Maddy Wu Amir Estejab

April 29, 2024

Introduction

As student's that took intro to CS at Williams, we learned how to program with Python. There were, of course, many benefits to learning how to code this way. For example, Python syntax, at the level that we were learning it, was not difficult to learn. However, there were also many challenges – especially for individuals that have never coded before or have only had experience coding with languages like Scratch. One of the main challenges that we came across as students in 134, was trying to gain a solid foundational understanding of the specifications in a language. Python is obviously a large language with many specifications which is daunting to people that have never programmed before.

BASIC is a programming language that was originally designed to remedy this problem. Designed by John G. Kemeny and Thomas E. Kurtz at Dartmouth College in 1963 as a way to make it easy for non-STEM students to learn how to code. For our project, we are looking to create an even simpler version of the language that will hopefully further lower the barriers of entry to learning how to the program and really reinforce what we believe to be the foundations of programming.

Design Principles

Because Basic BASIC is meant to be accessible even to most non-technical of users, both our primitive types and our combining forms will try to resemble plain English as much as possible. Furthermore, Basic BASIC will have even fewer specifications than BASIC. Having a small set of primitive types and limited data structures, allows the user to gain a stronger understanding of "foundational" concepts which will lend itself nicely if the user chooses to continue learning how to program.

Examples

Note: The file extensions are a work in progress

1. test1.txt:

```
PRINT "Hello World"
```

   Run "dotnet run test1.txt" in the project file. The output should be the following:

```
"Hello World"
```

2. test2.txt:

```
PRINT "gibberish"
```

   Run "dotnet run test2.txt" in the project file. The output should be the following:

```
"gibberish"
```

3. example-3.bb:

```
"Programming is cool"
```

Run ”dotnet run test3.txt” in the project file. The output should be the following:

```
"Programming is cool"
```

Language Concepts

For now, the user needs only to understand strings and print statements. Strings are a primitive data type in this language meaning they cannot be broken down into constiuent parts. The print statment, on the other hand, is a combining form that acts as an 'operation' which takes a string and does something with it (in this case, prints it on screen).

As we continue with our language, we will incorporate other elements of the language like GO TO or INPUT. Once again, the goal of Basic BASIC is to be easy to learn and program with. All types be it primitives or combining forms should resemble plain English as much as possible.

Formal Syntax

```
<Expr>      := <Command>_<String>
            | <String>
            | \epsilon
<Command>   := PRINT
<String>    := ""
```

Semantics

The 1.0 specification of Basic BASIC is very limited in its scope. It gives users one primitive type (string) and one combining form (PRINT) that allows for printing of different strings. With the current minimally working version, all valid expressions under the BNF prints out the AST. The language can read in one line from a file (i.e.: ”PRINT ”hello world”") as input and will output the AST with the prettyprint function as a result on the terminal.
As we continue to build the language we will likely also incorporate numbers or integers and characters. The combining forms would be PRINT, INPUT, IF ELSE, and maybe GO TO. We will also want a way to read in multiple lines from a file because a program written in Basic BASIC should have multiple lines. Given that we eventually do want to implement an INPUT function, our language will have the ability to read input.

The ultimate goal of Basic BASIC is to be able to operate the way Python might in 134. We want to be able to create very elementary programs that do whatever the user may want them to do. One example migh tbe if a user wants to create a mini video game using Basic BASIC. That would certainly be a possibility. Another possibilty, though we might not be able to implement it quite yet, is the ability to create graphics using Basic BASIC while still using the same few primitives and combining forms that Basic BASIC has.