Basic BASIC Language Specification

Maddy Wu Amir Estejab

May 13, 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

1. test1.bbas:

```
10 PRINT 2^3
```

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

```
8
```

2. test-2.bbas:

```
10 INPUT "Enter the first number: ", num1
20 INPUT "Enter the second number: ", num2
30 PRINT "The sum is: "; num1 + num2
```

Run ”dotnet run test2.bbas” in the project file. The user will then be prompted twice for a number. Once the user inputs both numbers, say the user inputs 10 and 20, the output should be:

```
"The sum is: 30"
```

3. STM.bbas:

```
10  REM TI-STEM
20  CALL CLEAR
30  RANDOMIZE
40  B=0
50  R$=" "
60  PRINT "Do you want to play with"
70  PRINT "1:letters?"
80  PRINT "2:numbers?"
90  PRINT "3:0 an 1 only?"
100 PRINT "4:or do you want to stop?"
110 CALL KEY (5,K,S)
120 IF S = 0 THEN 110
130 IF K > 57 THEN 110
140 IF K < 48 THEN 110
150 K=K-48
160 ON K GOSUB 420,450,480,510
170 INPUT "Difficulty? (1-6)":DIF
180 REM  DISPLAY***********
190 B=B+1
200 R$=" "
210 CALL CLEAR
220 FOR CC1=1 TO B
230 A=INT(RND*NR)+W
240 CALL HCHAR(12,16,A)
250 R$=R$&CHR$(A)
260 FOR CC2=1 TO 100/DIF
270 NEXT CC2
280 CALL HCHAR(12,16,31)
290 FOR CC2=1 TO 100/DIF
300 NEXT CC2
310 NEXT CC1
320 REM INPUT ANSWER *******
330 INPUT "ANSWER: ":AN$
340 IF AN$=R$ THEN 190
350 PRINT "WRONG"
360 PRINT "YOUR SCORE IS:";B-1
370 PRINT "THE ANSWER IS:";R$
380 PRINT "PRESS ANY KEY"
390 CALL KEY (5,K,S)
400 IF S=0 THEN 390
410 GOTO 20
420 NR=26
430 W=65
440 RETURN
450 NR=10
```

```
460 W=48
470 RETURN
480 NR=2
490 W=48
500 RETURN
510 PRINT "End of this game"
520 END
```

Run ”dotnet run STM.bbas” in the project file. The output should be the following:

```
DO YOU WANT TO PLAY WITH
1. LETTERS?
2. NUMBERS?
3. 0 OR 1 ONLY?
4. OR DO YOU WANT TO STOP?
```

How STM works:

Now you make your choice by entering 1, 2, 3, or 4 and also choose a level of difficulty. Say you entered 1 to play with letters. A letter will now appear on the screen but only for a very short time. You have to enter that letter. The computer will then show you two letters which you have to enter, then three, and so on. Obviously, as the number of letters increases, remembering them all becomes more difficult. What is the longest string of characters you can remember?

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> | <Num> |
              (<Expr> + <Expr>) | (<Expr> - <Expr>) |
              (<Expr> * <Expr>) | (<Expr> / <Expr>) |
              (<Expr> ^ <Expr>) | \epsilon | <Paren>
<Command>  := PRINT
<String>   := ""
<Num>      := n\in\Z
```

Semantics

Semantics				
Syntax	Abstract Syntax	Type	Prec./Assoc.	Meaning
”Hello World”	Bstring of string	string	n/a	A sequence of characters enclosed in double quotes (”). It is a primitive
n	Num of int	int	n/a	N is any positive or negative integer. It is a primitive.
+	Plus of Expr * Expr	char	left-associative	Adds the value of the left expression to the value of the right expression.

1. What are the primitive kinds of values in your system? For example, a primitive might be a number, a string, a shape, or a sound. Every primitive should be an idea that a user can explicitly state in a program written in your language.

   (a) The primitives that we currently have are strings and numbers.

2. What are the combining forms in your language? In other words, how are values combined in a program? For example, your system might combine primitive “numbers” using an operation like “plus.” Or perhaps a user can arrange primitive “notes” within a “sequence.”

   (a) The combining forms we have are your typical arithmetic operators like addition, subtratction, * multiplication, division, and exponentiation.

3. How is your program evaluated? In particular

   (a) Do programs in your language read any input?

      i. Our language does take in user input. Our language is a way to simplify programming, so users will be programming using our language.

   (b) What is the effect (output) of evaluating a program? Does the language produce a file or print something to the screen? Use one of your example programs to illustrate what you expect as output.

      i. Depending on the program that the user writes, Basic BASIC should be able to interpret any program written in the language and output what the user is hoping to output. The most complex example we have provided is a game called STM which should generate a playable game.