Learning Language (c--)

Stephen Robinson (sar2120)
Joseanibal Colon Ramos (jc2373)
George Liao (gkl2104)
Huabiao Xu(hx2104)

Purpose

Learning Language is a programming language designed to be accessible to students prior to entering high school. Students at this age traditionally have no exposure to computer programming despite the fact that many of them have the maturity and education to write procedural logic. For these students, existing programming languages can be overwhelming due to their abstract syntax and the difficulty of implementing I/O operations. To that end, the language is designed with the needs and abilities of 10-14 year old children in mind.

Language Features

Learning Language is designed to be accessible to students without any prior knowledge or experience with computer programming. It has natural syntax, default GUI and is case insensitive yet maintains enough similarities with advanced programming languages to ease the transition as students become more familiar with computer science concepts.

Natural Syntax: Learning language is designed to prevent many of the common errors made by beginner programmers. To do so, it gives the programmer the option of using short descriptive keywords in place of punctuation wherever possible. This allows for an easier understanding of code without sacrificing speed. Additionally, Learning Language requires no header above the main code, allowing users to write programs easily with no overhead.

Default GUI: By default, the compiler will create a simple java GUI for the program without the programmer being required to add any code. Furthermore, small additions to the GUI will be made easily available to the user through a simple API. This feature allows users the opportunity to see the results of their work on the screen to promote greater satisfaction for the young programmer. A compiler flag is available to disable this feature where a console program is desired.

Case insensitive: For younger users, keeping track of the capitalization of keywords and variables can be difficult and frustrating. Learning Language identifies the word that is being used and not the sequence of characters. Thus if a user names a variable "someNumber", when using this variable, the programmer can spell it "Somenumber" or "sOmEnUmBeR" without fault.

Syntax

If statements:

```
If (expression) then

ifnot

Endif
```

Loops:

```
Repeat (integer) times
    //loop count is held in implicitly defined variable “nth”
    Display nth
Endloop


Repeat until (expression)

Endloop
```

Expressions:

```
//assignment
A <- B

//comparison operators
A = B
A > B
A < B
A >= B
A <= B
A < B < C < D …

//operators
//order of operations is standard
A <- B + (C – D) * E / F ^ G % H
StringA <- StringB + StringC
numA <- |stringA| //magnitude operator, also works for lists
```

Data Types:

```
String called (name)
Number called (name)
Fraction called (name)
List of (type) called (name)
```

Sample Code

```
//Hello World
disPlay “Hello World!”

//counts to user specified number
Number called num
num <- input
repeat num times
   display nTh
endloop

//prints every other element in a list
List of string called strA <- {“1”, “cat”, “house”}
strA[2] <- “zzhShkgv”
Repeat |strA| times
   If nth % 2 = 0 then
      Displayline STRA[nth]
   endIf
Endloop

//prints the following output:
zzhShkgv
```