ICON PROGRAMMING LANGUAGE

Madison Ridley
madisonridley@email.arizona.edu

Sam Bryant
sbryant1@email.arizona.edu

April 7, 2020

ABSTRACT

1 Introduction

2 History

Icon was developed by Ralph Griswold and David Hanson with the first iteration being released in 1977. Since then, there have been nine versions of the language. The goal of the language according to Griswold is to create entirely new possible mechanisms rather than redefining already implemented ones. Icon's development was heavily inflenced by SNOBOL and SNOBOL5. One of the main functions of Icon was to redefine some of SNOBOL's mechanisms in a more general manner. The main function of the language is to deal with general tasks involving strings and data structures.

Currently, the ninth version of Icon is run on many operating systems. There is also now a language, Jcon which is a Java-based implementation of the Icon language.

3 Control Structures

Icon uses expressions for control structures, which makes it different from many other programming languages. As opposed to using booleans to drive control structures, Icon uses the success or failure of an expression to do so. Braces can be used in Icon programs, however they are not a necessary component for the code to run. In control structures, braces can be used to contain multiple expressions.

3.1 Loops

Included in Icon's control structures, there are several types of loops which evaluate expressions multiple times. Two of these are "while" and "until". Both have a similar structure, as shown in the example code below.

EXAMPLE:

```
procedure main()
i := 0
while i < 10 do
write(i)
i := i + 1
end
```

```
procedure main()
i := 0
until (i == 10) do
write(i)
i +:= 1
end
```

While both examples include the keyword "do", these structures don't require its use; it is optional to include. "While" will execute as long as the included expression is successful, and "until" will halt execution once the expression is successful. This is the main, notable difference between the two options. There are several additional control structures which are often used in tandem with the loops. One example is "not". This will succeed if the inside expression fails. Another helpful addition is "next". This can be used in an instance where a program may want to skip the current value when looping. Using the "every" control structure provides another way for programs to execute code multiple times. A variable will iterate until it reaches the end condition, in which case the execution of the loop will stop.

EXAMPLE:
```
procedure main()
every i := 1 to 10 do
write(i)
end
```

The last control structure that is used for loops is "repeat". This is the only method which continues to execute with no regard to whether the expression is successful or not. The only way to stop the repetition is with the use of "break".

EXAMPLE:
```
procedure main()
i := 0
repeat {
write(i)
if (i > 10) then break
else i +:= 2
}
end
```

3.2 Selections

Other than loops, there are several other control structures that can be used throughout icon programs. A key structure is "if-then-else". Much like other languages, this control structure is used to conditionally execute expressions. The "if" statement is evaluated, and should the expression be successful, the program moves on to execute the code included in the "then" statement that follows. Otherwise, the code included with the "else" statement will be executed. It is not a requirement to include "else", so in those cases, no code will execute should the initial expression fail.

EXAMPLE:
```
procedure main()
i := 1
write("i is 1")
if i < 0 then write("i is less than 0")
else write("i is greater than 0")
end
```

Lastly, Icon can use "case" to execute selection based on a specified value. It includes an expression which, when evaluated, is compared to the various possibilities provided.

EXAMPLE:
```
procedure main()
x := 117
case x of {
8 : write("x is 8")
28 : write("x is 28")
42 : write("x is 42")
117 : write("x is 117")
default : write("No number found")
}
end
```

4 Data Types

5 Subprograms

6 Summary

References

[1] Bob Alexander. Icon programming language reference. `https://www2.cs.arizona.edu/icon/refernce/ref.htm`.

[2] Thomas W. Christopher. *Icon Programming Language Handbook*. Dr. Thomas W. Christopher, Tools of Computing LLC, 1996.

[3] Todd A. Proebsting Gregg M. Townsend. Jcon: A java-based implementation of icon, 1999. `http://www.cs.arizona.edu/icon/docs/ipd286.htm`.

[4] Mariya Mykhailova. Programming language icon, 2012. `http://progopedia.com/language/icon/`.

[5] NA. The icon programming language. `https://www2.cs.arizona.edu/icon/`.

[6] Madge T. Griswold Ralph E. Griswold. *The Icon Programming Language*. Peer-to-Peer Communications, 3 edition, 2002.

[7] Carl Sturtivant. Experimental native distribution of icon for microsoft windows, 2015. `https://www2.cs.arizona.edu/icon/v95w.htm`.

[1] [2] [3] [5] [4] [6] [7]