Programming Language Documentation

Data Types

There are six data types in the language:

- Character (Keyword: “car”)
- Boolean (Keyword: “bool”)
- Signed Integer (Keyword: “ent”)
- Double (Keyword: “real”)
- String (Keyword: “string”)
- Void (Keyword: “vacio”)

These data types are used when declaring variables or functions.

Variable declaration example:

```
car c = 'c';
bool b = cierto;
ent e = 0;
real r = 3.14;
cad s = "Hola, Mundo.";
vacio v;
```

Function declaration example:

```
func foo() -> ent {
    ...
}
```

Keywords

In addition to the keywords above, the other keywords include

- “Cierto”: used to assign the “true” value to a Boolean variable and to use in a Boolean expression.
- “False”: used to assign the “false” value to a Boolean variable and to use in a Boolean expression.

```
bool true = cierto;
bool false = falso;
```

- “Si”: beginning of an if statement.
- “Sino”: beginning of an else statement.
    - The “si” and “sino” keywords can be combined to do an else-if statement.

```
si n < 0 {
    ...
} sino si n == 0 {
    ...
} sino {
    ...
}
```

- “Por”: used to signify a for loop.
    - The keyword “en” is used with the for loop for the iteration to occur.
    - Then, the range is specified by the two values separated by the ellipsis. In the below example, the iterator i starts from 0 and ends at 10.
    - A variable can also be used for the value to the right of the ellipsis.

Example of iterating between two integers inclusively:

```
ent arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

por i en 0...9 {
```

```
    imprimir(arr[i]);
}
```

Example of iterating between an integer and a variable inclusively:

```
ent arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
ent n = 9;

por i en 0...n {
    imprimir(arr[i]);
}
```

- “Mientras”: used to signify a while loop.

Example of a hanging while loop:

```
mientras cierto {
    ...
}
```

- “Parar”: stops a loop from continuing; equivalent to the “break” keyword.
- “Continuar”: keyword that automatically iterates a loop without doing anything; equivalent to “continue.”

```
por i en 0...10 {
    si i == 5 {
        parar;
    } sino {
        continuar;
    }
}

ent i = 0;
mientras cierto {
    si i == 100 {
        parar;
```

```
    }

    i += 1;
}
```

- “Regresar”: returns a value from a function.

```
func principal() -> ent { regresar 0; }
```

Operators

Mathematical

The standard mathematical operations addition (+), subtraction (-), multiplication (*), and division (/).

```
ent c = a + b;

ent c = a - b;

ent c = a * b;

real c = a / b;
```

In addition, there are operators that allow a direct assignment after doing a mathematical operation which looks like one of the operators listed above followed by an equal sign (+=, -=, *=, /=).

```
i += 1;

i -= 1;

i *= 1;

i /= 1;
```

Boolean

There are two Boolean operators that can be used in logical expressions: “&&”, “||”, and “!”, which are the Boolean AND, OR, and NOT operators respectively.

```
si a && b {
    ...
}

si a || b {
    ...
}

si !a {
    ...
}
```

Equality

There is the “==” operator to check if two things are equal and the “!=” to check if two things are not equal.

```
si a == b {
    ...
}

si a != b {
    ...
}
```

Relational

The “<” operator checks if one thing is less than another thing (or “<=” if something is less than or equal to another thing), and the “>” operator checks if one thing is greater than another thing (or “>=” if something is greater than or equal to another thing).

```
si a < b {
    ...
}

si a > b {
    ...
}

si a <= b {
    ...
}

si a >= b {
    ...
}
```

Information on Writing Programs

Functions need to be in a specific order when writing them. Specifically, the variable declarations always need to be at the beginning of the function, the statements need to occur after the declarations, and the return statement always needs to be at the end as this is how the parser defines the syntax of a function. This issue will be attempted to be fixed. Functions are made by using the “func” keyword at the beginning then giving the name of the function, the arguments of the function, and specifying the return type using the arrow syntax. The body of the function will be encased in curly braces.

If statements can’t have return statements as the parser doesn’t recognize this to be legal. Parentheses are not needed around the condition(s) of the if statement. Including parentheses will lead to a syntax error. For example, doing something like “si (n % 2 == 0)” will result in a syntax error. The body of the if statement will be encased in curly braces as well.

There is no functionality for global variables, but that can be fixed by modifying the parser to allow it. A production needs to be made that can occur alongside function creation.

There are several built-in functions in the programming language for printing something to the screen, getting user input, and acting as the starting point to the program (like the main function in C/C++). The names for these functions are “imprimir”, “leer”, and “principal” respectively. The imprimir and leer functions can take several arguments separated by commas to either print multiple things or to read in multiple things.

Example of the built-in functions:

```
func principal() -> ent {
    ent n;

    imprimir("Introduzca un número: ");
    leer(n);

    si n % 2 == 0 {
        imprimir(n, " es par.\n");
    } sino {
        imprimir(n, " es impar.\n");
    }

    regresar 0;
}
```