MANUAL DE USUARIO LOOPY

DISEÑO DE COMPILADORES

NOVIEMBRE 23, 2016

ANDRÉS ÁVILA MARTÍN DEL CAMPO / JESÚS NAVARRO MARÍN

Índice

INTRODUCCIÓN

En el presente documento se describe la manera en la que se debe programar en el lenguaje Loopy. Es un manual que requiere conocimientos básicos de programación para su entendimiento por lo que no es muy detallada la descripción de sus operaciones. A continuación se da la información necesaria para poder utilizar el lenguaje Loopy.

TIPOS DE VARIABLES

En en lenguaje Loopy acepta los tipos enteros: int, flotantes: float, caractéres: char, strings: string. A continuación se muestra un ejemplo de constantes de los diferentes tipos aceptables:

```
int - 2
float - 3.4
char - 'a'
string - 'var'
bool - _true
```

OPERACIONES EN LOOPY

- Declaración de variables
- Declaración de variables dimensionadas
- Asignación de variables
- Asignación de variables dimensionadas
- Operador de condición “if” / “else”
- Operador de ciclo “while”

- Operadores aritméticos suma, resta, multiplicación y división
- Operadores relacionales “<”, “>”, “==”, “<=”, “>=”, “!=”.
- Operadores lógicos “&&”, “||”
- Declaración de Funciones
- Llamada de Funciones
- Escritura en pantalla

DECLARACIÓN DE VARIABLES

La declaración de variables en Loopy se describe en la siguiente imagen:

```
var x: int;
```

- El token “var” quiere decir que está reservada en el lenguaje y quiere decir que es una esa línea es una declaración de variable.
- El token “x” es el nombre con el cual se identificará la variable.
- El token “:” es necesario para separar al nombre de la variable con su tipo.
- El token “int” es el tipo de variable.

También es posible declarar más de una variable del mismo tipo en una misma línea como se muestra a continuación:

```
var y,x,j,i: int;
```

DECLARACIÓN DE VARIABLES DIMENSIONADAS

La declaración de variables dimensionadas se muestra a continuación:

```
var a: float [10];
```

- El token “var” quiere decir que está reservada en el lenguaje y quiere decir que es una esa línea es una declaración de variable.
- El token “a” es el nombre con el cual se identificará el arreglo.
- El token “:” es necesario para separar al nombre de la variable con su tipo.
- El token “float” es el tipo de variable.
- Los tokens “[10]” definen al tamaño del arreglo.

A diferencia de las variables no dimensionadas, únicamente se puede declarar un arreglo por línea.

ASIGNACIÓN DE VARIABLES

La asignación de variables se muestra a continuación:

```
a = 5;
```

- “a” es una variables que tuvo que haber sido declarada previamente.
- “=” es el operador el cual le asignará un valor a “a”.

- El “5” es un valor que debe coincidir con el tipo de “a”, en este caso podemos intuir que a es de tipo “int”.

ASIGNACIÓN DE VARIABLES DIMENSIONADAS

```
arr[4] = 4.7;
```

- “arr” es el nombre del arreglo.
- “[4]” es el índice del arreglo en el cual se le asignará el valor 4.7. El índice puede estar en forma de una expresión siempre y cuando el resultado de tipo entero.
- El tipo de “arr” debe coincidir con el tipo al que pertenece 4.7 que en este caso es “float”.

OPERADOR DE CONDICIÓN IF / ELSE

A continuación se muestra la sintaxis de “if” / “else” :

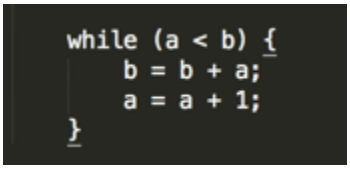
```
if (a > b) {
    a = 5;
}
else {
    b = 5;
}
```

- Después de la palabra “if” debe seguir un paréntesis y dentro de este una o más operaciones lógicas.
- Después del paréntesis deben ir llaves (“{“, “}”) y dentro de estas cualquier operación.
- El else únicamente puede ir después de un “if” y es opcional. No lleva paréntesis con una operación lógica ya que esta está implícita.

- Al igual que en el “if” después siguen llaves (“{“, “}”) y dentro de estas cualquier operación.

OPERADOR DE CICLO WHILE

A continuación se muestra la sintaxis del “while”:

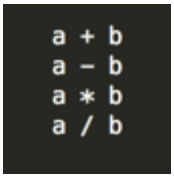


```
while (a < b) {
    b = b + a;
    a = a + 1;
}
```

- Similar a la sintaxis del “if”
- Es importante recordar que la variable de control en este caso “a” debe incrementar / decrementar dependiendo el caso para que no se cicle el programa.

Operadores aritméticos suma, resta, multiplicación y división

Las sintaxis de las operaciones aritméticas sigue el estándar común y se muestra a continuación.



```
a + b
a - b
a * b
a / b
```

- La prioridad de operaciones es la siguiente: multiplicación / división y después suma / resta.
- Nuestro lenguaje tiene asociatividad izquierda en estas operaciones por lo que se realizarán las que se encuentran más a la izquierda primero.

OPERADORES RELACIONALES <, >, ==, <=, >=, !=.

Estos operadores únicamente se pueden realizar en operaciones condicionales y cíclicas como se muestra a continuación:

```
if (a < b) {
}
if (a > b) {
}
if (a <= b) {
}
while (a >= b) {
}
while (a == b) {
}
```

- Estos operadores tienen una menor jerarquía que los operadores aritméticos por lo que se realizarán después.

OPERADORES LÓGICOS &&, ||

A continuación se muestra cómo se deben utilizar los operadores lógicos:

```
if (a < b || c == f) {
}
while (a >= b && c == f) {
}
```

- Estos operadores únicamente pueden separar a operaciones relacionales y son de una menor jerarquía por lo que se ejecutan después.

DECLARACIÓN DE FUNCIONES

Todas las funciones en Loopy tienen un valor de retorno el cual debe coincidir con el tipo del cual se declaró la función. Las funciones pueden tener parámetros de cualquier tipo de dato y son utilizados localmente únicamente. A continuación se muestra cómo debe ser declarada una función:

```
function int fact(int n) {
    var result: int;
    if(n == 1) {
        return 1;
    }

    result = call:fact(n - 1) * n;
    return result;
}
```

- Se pueden declarar variables dentro de la función las cuales únicamente pueden ser accesadas localmente y estas deben estar al principio del bloque, es muy importante esto o de lo contrario el programa fallará.
- El final de la función debe contener un “return” con una expresión.

LLAMADA DE FUNCIONES

Para llamar a una función es necesario asignarle a una variable previamente declarada el valor de retorno de dicha función cuyo tipo debe coincidir con el tipo de la variable. A continuación se muestra cómo se llama a la función “fact” de la imagen anterior:

```
var a: int;
a = call:fact(4);
```

- Es necesaria la palabra “call” seguida de “:” seguida del nombre de la función y sus parámetros.

ESCRITURA EN PANTALLA

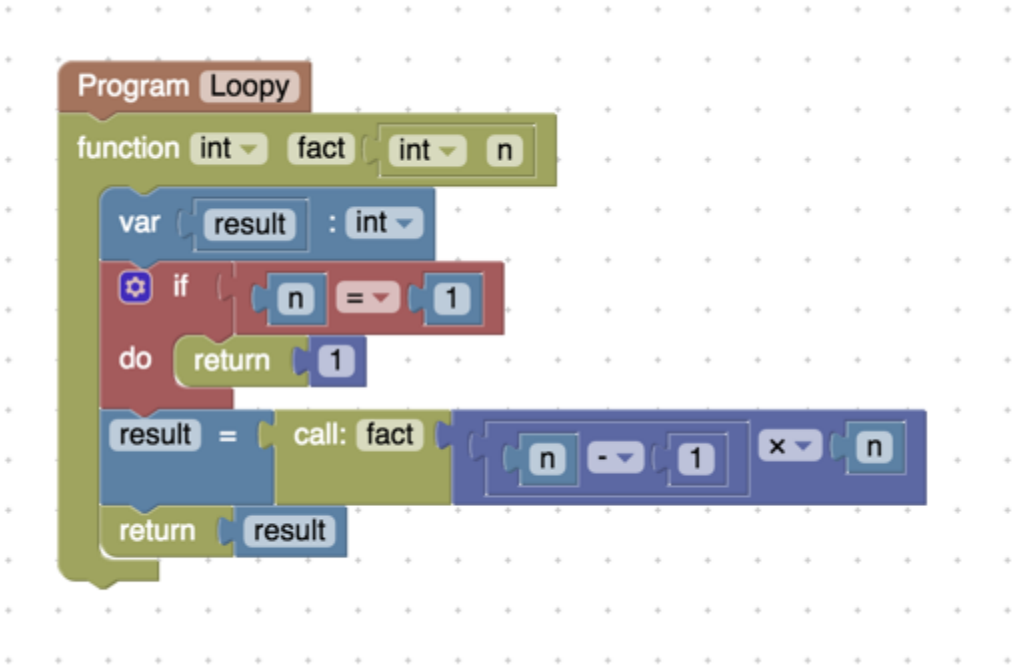
Esta función despliega en pantalla el valor de alguna variable. A continuación se muestra cómo se debe escribir.

```
print(a);
```

- Debe tener la palabra “print” al inicio y seguida de paréntesis con diferentes valores o expresiones separadas por comas.

BLOCKLY

Blockly es la herramienta gráfica utilizada para generar código Loopy. El objetivo de utilizar Blockly es que cualquier programador sin conocer la sintaxis de Loopy pueda desarrollar código en este únicamente teniendo conocimientos básicos de programación. A continuación se muestran pantallas de cómo construir los programas por medio de bloques y el código que generan estos:



El código generado es el siguiente:

```
program Loopy;
function int fact(int n) {
  var result : int;
  if (n == 1) {
    return 1;
  }
  result = call: fact(n - 1) * n;
  return result;
}
```

Como podemos observar este código generado es del lenguaje Loopy. Una vez que obtienes este código puedes compilarlo y obtener resultados.