Notación Alpha - Referencia

Disponible en http://goo.gl/cmUX93

Alpha es una notación para la construcción de algoritmos y estructuras de datos que es simple, eficaz y moderna, permitiendo una rápida conversión entre el pseudocódigo y un lenguaje de programación.

Tipos de Datos Simples y Declaraciones

```
Simples: Char, Integer, Boolean,  Real, Enum, Pointer
Char cMychar          //these are examples
Integer iValue, iResult;   Boolean bFlag;
Real * pMyArray
Enum eDays = [Monday, Wednesday]
Const Real PI = 3.14159
```

Tipo de Dato Compuesto: String

```
String sName = "Springfield"      //there are 11 letters
Integer iPos = 4
Char c = sName[iPos]              //equal to sName[4]
Print (c)                         //the output is 'i'
String sSubName = sName[2..5]
Print (sSubName)                  //the output is "prin"
```

Tipo de Dato Compuesto: Array

```
Array aMonths of Integer [1..6] = {1, 2, 3, 5, 8, 13}
Array aTable of Char[1..2][1..2] = {{'a', 'b'}, {'c', 'd'}}
Array aInput  of Real [1..3]
aInput[1] = 1.2;    aInput[2] = 3.4;    aInput[3] = 7.8;
```

Tipo de Dato Compuesto: Register

```
Register rgPerson
  String sName
  Integer iId
  String  sAddress
end
rgPerson = { "Homer Simpson" , "666" ,  "N/A" }
rgPerson. sAddress = "742 Evergreen Terrace"
```

Entrada/Salida Estándar

```
Read (<identificador-1>, ... , <identificador-k>[;]
Print ( <identificador-1 >, ... , <identificador-k>[;]
```

Tipo de Dato Definido

```
Type Array aWeek of Boolean [1..7]
aWeek tLaboral, tHolidays
```

Estructuras de Control

```
if iNumber == 0  then          // iNumber is an integer between [0,n]
  Print ( "The number is 0")
elseif  iNumber == 1   then
  Print  ( "The number is 1" )
else
  Print  ( "The number greater than 1" )
end
// using select statement
select
iNumber == 0: Print  ( "The number is 0" )
iNumber == 1:  Print  ( "The number is 1" )
iNumber > 1:  Print  ( "The number greater than 1" )
end
```

Acciones y Funciones

```
void Factorial (Integer iN)
  Integer iFactorial = 1
  while iN > 1 do
   iFactorial = iFactorial * iN
   iN = iN - 1
  end
  Print (iN +"!="+ iFactorial)
end
Integer iNumber = 8
Factorial (iNumber)
```

```
function Factorial (Integer iN) :Integer
  Integer iFactorial = 1
  while iN > 1 do
   iFactorial = iFactorial * iN
   iN = iN - 1
  end
  return iFactorial
end
Integer iN = 8
Print(iN + "!=" + Factorial(iN))
```

Los parámetros de las acciones y funciones por defecto son por valor. Para utilizar parámetros por referencia, se debe anteponer la palabra reservada ref a un identificador.

Estructuras Iterativas

```
Integer iFactorial = 1, iNumber = 8
//using the while and do-while statement
while iNumber > 1 do
  iFactorial = iFactorial * iNumber
  iNumber = iNumber - 1
end
Print (iNumber + "!=" + iFactorial)
iFactorial = 1, iNumber = 8
do
  iFactorial = iFactorial * iNumber
  iNumber = iNumber - 1
while iNumber > 1
Print (iNumber + "!=" + iFactorial)
//using the for and foreach statement
iFactorial = 1
for iNumber = 2 to 8 do
  iFactorial = iFactorial * iNumber
end
Print (iNumber + "!=" + iFactorial)
iFactorial = 1
Array aValues of Integer [] = {2,3,4,5,6,7,8}
foreach iNumber in aValues
  iFactorial = iFactorial * iNumber
end
Print (iNumber + "!=" + iFactorial)
```

Tipo de Dato Pointer

```
Type Register Point
  Real x,y
end
Point ptMyPoint
Point * pPointer
pPointer = ref ptMyPoint
*pPointer.x = 4
*( ref ptMyPoint).y = 3    //equal to ptMyPoint.y = 3
*(*( ref pPointer)).x = 1   //equal to ptMyPoint.x = 1
```

Notación Alpha - Referencia

Disponible en http://goo.gl/cmUX93

Alpha es una notación para la construcción de algoritmos y estructuras de datos que es simple, eficaz y moderna, permitiendo una rápida conversión entre el pseudocódigo y un lenguaje de programación.

Definición de una Clase

```
class Student
private:
 String m_sName
 Integer m_iId
 Boolean m_bIsActive
public:
 Constructor Student()
  m_sName = ""
  m_iId = -1
  m_bIsActive = false
 end
 Constructor Student(String sName, Integer iId, Boolean bIsActive)
  m_sName = sName
  m_iId = iId
  m_bIsActive = bIsActive
 end
 void PrintInfo()
  String sValue = "active"
  if not m_bIsActive then
   sValue = "not" + sValue
  end
  Print (m_sName + ", id: " + m_iId + " is " + sValue)
 end
protected:
 void ResetId(Integer iId)
  m_iId = iId
 end
end
```

Herencia Simple

```
class Bachelor inherited Student
public:
 void PrintInfo()
  Print ("Bachelor with the following info:")
  super.PrintInfo()
 end
end
```

Clase empleando *templates*

```
class Node <T>     //class Node using templates
public:
 T tInfo
 Node<T> * pNext
end
class Stack <T>    //class Stack using pointers
private:
 Node <T> * pTop
 Integer iN
public:
 Constructor Stack()
  iN = 0
  pTop = NIL
 end
 Destructor Stack()
  while pTop != NIL do
   Node<T> * pTemp = pTop
   pTop = *pTop.pNext
   delete pTemp
 end
 function IsEmpty() : Boolean
  return iN == 0
 end
 void Push(ref T x)
  Node<T> * pNew = new Node
  *pNew.x = *x
  *pNew.pNext = pTop
  pTop = pNew
  iN = iN + 1
 end
 void Pop()
  Node<T> * pTemp = pTop
  pTop = *pTop.pNext
  *pNew.pNext = pTop
  delete pTemp
  iN = iN - 1
 end
 function Size() : Integer
  return iN
 end
 function Top() : T
  return *pTop.tInfo
 end
end
//the class can be used directly after its declaration
Stack myStack <Integer> = new Stack()
myStack.Push(1); myStack.Push(3); myStack.Push(5);
 myStack.Push(8);
Print (myStack.Size()) //the output is 4
Print (myStack.Top()) //the output is 8
myStack.Pop()
myStack.Pop()
Print (myStack.Top()) //the output is 3
```

Conversiones Automáticas

Char	→	String
Integer	→	Real
Enum	→	Integer

Prioridad de Operadores

Aritmético	[ ] ( ) *(derreferenciado) ref -(unario) ^(potencia) * / div(división entera) mod(módulo) + -
Relacional	> < ≥ ≤ == !=
Lógico	not and or
Asignación	=

@esmittramirez
esmitt.ramirez@ciens.ucv.ve