**Comentando Listas Template Simplemente Enlazada**
**José Miguel Huamán Cruz, C.U.I.:20103389, josemiki24@gmail.com**
**Escuela Profesional de Ciencia de la Computación**
**Facultad de Producción y Servicios**
**Universidad Nacional de San Agustín**

## 1. Comentando Codigo "ListaT"

```
#include <iostream>
#include <string>
using namespace std;

//Traductor Bilingue
class Palabra//Diccionario
{
  public:
         string castellano;//almacena palabras en castellano
         string ingles;//almacena palabras en ingles
  public:
          Palabra(){};
          Palabra(string c, string i)
          {
                  castellano = c;//guarda string c en castellano
                  ingles = i;//guarda string i en castellano
          }

          string traducir(bool t =0)//solo retorna 0 cuando la
                                    //palabra esta en ingles
          {
                  if (t ==0) return ingles;
                  return castellano;

          }

          bool operator==(Palabra & p) //compara contenido catellano de
                                 //de la palabra
          {
                  return castellano == p.castellano;

          }

          bool operator>(Palabra & p)
          {
                  return castellano > p.castellano;

          }

          friend ostream & operator<<(ostream &os, Palabra & p)
                        //sobrecarga de operador binario << para
                        //retornar castellano ingles
          {
                  os<<p.castellano<<"--"<<p.ingles<<endl;
                  return os;
          }
```

```
47 };
48
49 template<class T>//Clase lista template de tipo T
50 class Lista;//clase vacia poque se llama en clase Nodo
51
52 template<class T>
53 class Nodo//Clase Nodo T
54 {
55    friend class Lista<T>;
56    typedef Nodo<T> * pNodo;//definimos a un Nodo<T>* como pNodo
57    private:
58        T       m_Dato;//miembro dato de tipo T
59        pNodo   m_Sig;// Puntero a el siguiente nodo
60    public:
61       Nodo(T d):m_Dato(d),m_Sig(0){};//inicializamos al nodo en 0
62                          // miembro dato como puntero al siguiente nodo
63       void matate()  //funcion que se llama a si recursivamente
64       {
65          if(m_Sig) m_Sig->Matate();//si hay un pustero a nodo matalo primero
66          delete this; //muere el ultimo
67       }
68       void print(ostream &os)//fumcion para imprimir de atras hacia adelante
69       {
70          if(m_Sig) m_Sig->print(os);//se llama recursivamente
71          cout<<m_Dato<<" "; //impreme el ultimo
72       }
73 };
74
75
76 template<class T>//clase template tipo T
77 class Lista
78 {
79      private:
80        Nodo<T> *  m_pHead;// puntero a la cabeza
81      public:
82        Lista()
83        {
84           m_pHead =0;//asigna 0 a la cabeaza lista vacia
85        };
86
87        bool find(T d, Nodo<T> ** & p)//busqueda que guarda la ultima ubicacion
88                                      //donde se quedo buscando
89        {
90             p = &m_pHead;//el doble puntero p apunta a la cabeza
91             while(*p)//mientras exista contenido en p
92             {
93                if ((*p)->m_Dato == d) return true;//revise si el m_Dato es d
94                if ((*p)->m_Dato > d)  return false;//revise si el
95                                                    //m_Dato es > que d
96                p = &((*p)->m_Sig);//p es un puntero doble que guarda direccion
97                                   //de m_Sig que guarda direccion de un nodo
98             }
99             return false;// si no lo encuentra retorna false
100       }
101       bool  Add(T d)//funcion agregar con base en funcion find
```

```
102         {
103             Nodo<T> ** q;//se crea un doble puntero a nodo q
104             if (find(d,q)) return  false;//hace una busqueda que si
105                                 // lo encuentra retorna false
106                         //y no lo agrega pero si no lo encuentra loa gregara
107             Nodo<T> * nuevo = new Nodo<T>(d);//crea un nodo nuevo con dato d
108             nuevo->m_Sig = *q;//lo agrega en la posicion que se quedo q
109             *q = nuevo; //y lo agrega a la lista
110             return true;
111 
112         }
113 
114         Lista<T> & operator<<(T d)//agrega datos a  la lista
115         {
116             Add(d);//llama la funcion Add
117             return *this;//retorna true o false depende de Add
118 
119         }
120 
121     friend ostream & operator<<(ostream & os, Lista<T> & l)//sobrecarga
122                         //el operador << para imprimir la lista
123     {
124           l.m_pHead->print(os);//la lista ingresa a la cabeza y llama
125                         //recursivamente a la impresion de nodos
126           os<<endl;//imprime el valor del nodo
127           return os;//retorna os
128 
129     }
130 };
131 
132 
133 int main()
134 {
135 
136 return 1;
137 }
```