UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
EDD
Practica 1
SECCIÓN: “A”
Ing. Luis Espino
AUX. Robinson Pérez

Manual Técnico

Nombre: Brayan Mauricio Aroche Boror
Carné: 201503918

Lista Simplemente Enlazada

Struct NodoListaSimple:

```
#ifndef NODOLISTASIMPLE
#define NODOLISTASIMPLE
struct NodoListaSimple
{
   int identificador;
   bool estado;
   int turnos;
   int lugares;
   NodoListaSimple *siguiente;
};
#endif // NODOLISTASIMPLE
```

Struct ListaSimple:

```
struct ListaSimple
{
   NodoListaSimple *ini;
   NodoListaSimple *fin;
   int tam;
   void inicializarListaSimple()
   {
      ini = NULL;
      fin = NULL;
      tam = 0;
   }
```

```
int insertarListaSimple(int id, bool estado, int turnos,int lugar)
{
   NodoListaSimple *nuevoNodo = NULL;
   NodoListaSimple *nodoAux;
   nuevoNodo = new NodoListaSimple();
   nodoAux = new NodoListaSimple();
   nuevoNodo->identificador = id;
   nuevoNodo->estado = estado;
   nuevoNodo->turnos = turnos;
   nuevoNodo->lugares = lugar;

   if(ini == NULL)
   {
      nuevoNodo->siguiente = ini;
      ini = nuevoNodo;
      fin = nuevoNodo;
      tam++;
   }
   else
   {
      nodoAux = fin;
      nodoAux->siguiente = nuevoNodo;
      nuevoNodo->siguiente = NULL;
      fin = nuevoNodo;
      tam++;
```

```
    }
    return 0;
}

void modificarListaSimple(int id, bool estado, int turnos)
{
    NodoListaSimple *nuevoNodo;
    NodoListaSimple *nodoAux;
    nuevoNodo = new NodoListaSimple();
    nuevoNodo->identificador = id;
    nuevoNodo->estado = estado;
    nuevoNodo->turnos = turnos;

    nodoAux = ini;
    while(nodoAux->estado!=false)
    {
        nodoAux = nodoAux->siguiente;
    }
    nodoAux->estado = nuevoNodo->estado;
    nodoAux->identificador = nuevoNodo->identificador;
    nodoAux->turnos = nuevoNodo->turnos;
}

bool turnosFaltantes()
{
    if(ini->turnos>0)
```

```
    {
        for(int i = 0; i<tam; ++i)
        {
            ini->turnos--;
            ini = ini->siguiente;
        }
        return true;
    }
    else
    {
        modificarListaSimple(0,false,0);
        return false;
    }
    //while(nodoActual->turnos)
}

bool recorrido()
{
    NodoListaSimple *nodoActual;
    nodoActual = ini;
    while(nodoActual->estado != false)
    {
        nodoActual = nodoActual->siguiente;
        //return false;
    }
    return true;
```

```
    }


    void recorridoListaSimple()
    {
        NodoListaSimple *nodoActual;
        nodoActual = ini;
        while(nodoActual!= NULL)
        {
            //printf("Persona: %d",nodoActual->identificador,"%s \n");
            //qDebug() << "Persona: " << nodoActual->identificador << "\n";
            qDebug() << "Avion en lista s->"<<nodoActual->identificador << "Estado: "<<nodoActual->estado;
            nodoActual = nodoActual->siguiente;
        }
        //return true;
    }
};
#endif // LISTASIMPLE


Struct Nodo Lista Doblemente Enlazada
struct NodoListaDoble
{
    char letra;
    ColaEscriorio *cola = new Cola();
    Pila *pila = new Pila();
    NodoListaDoble *siguiente;
```

```
    NodoListaDoble *anterior;
};
```

Metodos y funciones:

```
#ifndef LISTADOBLE
#define LISTADOBLE
#include "nodolistadoble.h"

struct ListaDoble
{
    NodoListaDoble *ini;
    NodoListaDoble *fin;
    int tam;

    void inicializarlistadoble()
    {
        ini = NULL;
        fin = NULL;
        tam = 0;
    }

    int insertarListaDoble()
    {

    }

    void ordenamiento(ListaDoble *lista, int izq, int der)
```

```
{
   int i, j, x , aux;
   i = izq;
   j = der;
   x = lista[(izq + der)/2];
      do{
         while( (lista->tam < x) && (j <= der) )
         {
            i++;
         }

         while( (x < lista->tam) && (j > izq) )
         {
            j--;
         }

         if( i <= j )
         {
            aux = lista->tam; lista->tam = lista->tam[j]; lista[j] = aux;
            i++;  j--;
         }

      }while( i <= j );

      if( izq < j )
         ordenamiento(lista,izq,j);
```

```
        if( i < der )
            ordenamiento(lista,i,der);
    }
};


#endif // LISTADOBLE
```

Nota: para la cola y la lista simple se usaron los mismo criterios para realizar sus nodos y sus métodos de inserción y eliminación, al igual para la lista doblemente enlazada con la cola doble y para las listas circulares solo se apuntaron al valor de inicio el nodo siguiente del fin y al final del nodo anterior de inicio.

Pila

```
#ifndef NODOPILA
#define NODOPILA


struct NodoPila
{
    int doccumento;
    NodoPila *siguiente;
};
#endif // NODOPILA



#ifndef PILA
#define PILA
#include "nodopila.h"
#include <QDebug>
```

```
struct Pila
{
   int tam;
   NodoPila *ini;

   void inicialiizarPila()
   {
      tam = 0;
      ini = NULL;
   }

   int push(int documento)
   {
      NodoPila *nuevoNodo = NULL;
      nuevoNodo = new NodoPila();
      nuevoNodo->doccumento = documento;
      if(ini == NULL)
      {
         nuevoNodo->siguiente = ini;
         ini = nuevoNodo;
         tam++;
      }
      else
      {
         nuevoNodo->siguiente = ini;
```

```
        ini = nuevoNodo;
        tam++;
    }
}

void mostrarPila()
{
    NodoPila *nodoAux;
    int i;
    nodoAux = ini;
    for(i=0;i<tam;++i)
    {
        //printf("%d",nodoAux->doccumento,"\n");
        qDebug()<<nodoAux->doccumento<<"\n";
        nodoAux = nodoAux->siguiente;
    }
}

int pop()
{
    NodoPila *nodoDelete;
    if(tam==0)
        return -1;
    nodoDelete = ini;
    ini = ini->siguiente;
    free(nodoDelete);
```

```
        tam--;
        return 0;
    }
};
#endif // PILA
```