**Ejemplo 01**

Escriba la especificación de los TAD’s, los algoritmos y su implementación en C++ para registrar los datos de los alumnos en una lista.

*Solución:*

*Especificación de los TAD’S y los algoritmos*

**Especificacion** ALUMNO

**variable**

codigo : cadena.

nombre : cadena.

edad : entero.

**métodos**

menu() : no retorna valor.

ingresarDatos() : no retorna valor.

mostrarDatos() : no retorna valor.

**significado**

*menu* muestra las opciones a escoger.

*ingresarDatos ingresa los datos de un alumno.*

*mostrarDatos visualiza los datos de un alumno.*

**Fin\_especificacion**

**Especificacion** LISTA

**usa** ALUMNO

**variable**

info : ALUMNO.

raiz : LISTA.

sgte : LISTA.

**métodos**

ingresarLista(raiz,x) :no retorna valor.

mostrarLista(raiz) :no retorna valor.

**significado**

*ingresaLista* registra en la lista con cabecera raiz del tipo LISTA un alumno *x* del tipo ALUMNO.

*mostrarLista* muestra todos los alumnos de la lista con cabecera raiz del tipo LISTA.

**Fin\_especificacion**

Procedimiento ingresarLista (raiz, x)

// definir variables

1. LISTA: q
2. crear (n)
3. Asignar ( n, x )
4. Si raiz = nulo entonces
5. raiz 🡨 n
6. Sino
7. q 🡨 raíz
8. Mientras (sgte(q) ≠ nulo) hacer
9. q 🡨 sgte(q)
10. Fin\_mientras
11. sgte(q) 🡨n
12. Fin\_si
13. sgte(n) 🡨 nulo

Fin\_procedimiento

Vamos a explicar el algoritmo para los siguientes valores:

**Condicion inicial:**

raiz 🡨 NULL

Datos

1 cesar 12

2 pedro 14

3 Javier 11

**Se hacen tres llamadas al método ingresarLista**

* Primera llamada

(2) crear(n)

n

(3) Asignar(n, x)

Info

sgte

1 cesar 12

n

(4) verdad (5) raiz 🡨n

Info

sgte

1 cesar 12

n

raiz

(13) sgte(n) 🡨 NULL

Info

sgte

1 cesar 12

n

raiz

nulo

* Segunda llamada

(2)crear(n)

n

(3)Asignar(n, x)

Info

sgte

2 Pedro 12

n

(4) falso (7) q🡨raiz (8) falso

Info

sgte

1 cesar 12

q

raiz

nulo

(11)sgte(q)🡨n

Info

sgte

1 cesar 12

q

raiz

Info

sgte

2 Pedro 14

n

(13) sgte(n) 🡨 NULL

Info

sgte

1 cesar 12

q

raiz

Info

sgte

2 Pedro 14

n

nulo

* Tercera llamada

(2)crear(n)

n

(3)Asignar(n, x)

Info

sgte

3 Javier 11

n

(4) falso (7) q🡨raiz (8) verdadero

Info

sgte

1 cesar 12

q

raiz

Info

sgte

2 Pedro 14

nulo

(9) q 🡨 sgte(q)

Info

sgte

1 cesar 12

q

raiz

Info

sgte

2 Pedro 14

nulo

(8) falso (11) sgte(q)🡨n

Info

sgte

1 cesar 12

q

raiz

Info

sgte

2 Pedro 14

n

Info

sgte

3 Javier 11

(13) sgte(n) 🡨 nulo

raiz

Info

sgte

Info

3 Javier 11

2 Pedro 14

1 cesar 12

n

q

sgte

Info

sgte

nulo

Procedimiento mostrarLista (raiz)

// Definir variables

LISTA: ptr

ptr 🡨 raiz

Mientras ( ptr ≠ nulo) Hacer

Leer (ptr, info)

info.mostrarDatos()

ptr 🡨 sgte(ptr)

Fin\_mientras

Fin\_procedimiento

*Implementación del TAD*

#include <iostream>

#include <cstdlib>

using namespace std;

class ALUMNO

{

char codigo[10];

char nombre[40];

int edad;

public :

void menu()

{

cout<< "\nMENU DE OPCIONES\n";

cout<< "----------------\n" ;

cout<<"<1> Ingresar Lista\n";

cout<<"<2> Mostrar Lista\n";

cout<<"<3> Salir \n";

}

ALUMNO(){ strcpy(codigo,"");strcpy(nombre,"");int nota=0; }

void ingresarDatos()

{

fflush(stdin);

cout<<"ingrese codigo : ";

gets(codigo);fflush(stdin);

cout<<"ingrese nombre : ";

gets(nombre ); fflush(stdin);

cout<<"ingrese nota : ";

cin>>edad;

}

void mostrarDatos()

{

cout<<"\n "<<codigo<<" "<<nombre<<" "<<edad;

}

};

class LISTA

{

ALUMNO info;

LISTA \*sig;

public:

void ingresaLista (LISTA \*&raiz,ALUMNO x)

{

LISTA \*q;

LISTA \*n =new LISTA;

n->info=x;

if(raiz==NULL)

raiz=n;

else

{

q=raiz;

while (q->sig!= NULL)

q=q->sig;

q->sig=n;

}

n->sig=NULL;

}

void mostrarLista (LISTA \*raiz)

{

LISTA \*ptr;

ptr= raiz;

while( ptr!= NULL)

{

ptr->info.mostrarDatos();

ptr=ptr->sig;

}

cout<<endl;

}

};

int main()

{

char opcion;

ALUMNO x;

LISTA lista;

LISTA \*raiz;

raiz=NULL;

do

{

x.menu();

cout<<"\ningrese opcion : ";

opcion=cin.get();

switch(opcion)

{

case '1':

x.ingresarDatos();

lista.ingresaLista(raiz,x);

break;

case '2':

lista.mostrarLista(raiz);

break;

}

cin.ignore();

}

while( opcion !='3');

system("PAUSE");

return 0;

}

**Ejemplo 02**

Escriba la especificación de los TAD’s, el algoritmo y su implementación en C++, que permita registrar en una tercera lista, la concatenación de dos listas dinámicas simple. Por ejemplo:

Entrada

Lista 1: 20, 3, 7, 9

Lista 2: 5, 6

Salida

Lista 3: 20, 3, 7, 9, 5, 6

*Solución:*

*Especificación de los TAD’s y los algoritmos*

**Especificación** ENTERO

**variable**

num : entero

**métodos**

menu() : no retorna valor.

ingresarEntero( ) : no retorna valor.

mostrarEntero( ) : no retornar valor.

**significado**

*menu* muestra las opciones a escoger.

*ingresarEntero ingresa un valor entero.*

*mostrarEntero visualiza un valor entero.*

**Fin\_especificacion**

**Especificación** LISTA

**Usa** ENTERO

**variable**

info : ENTERO

sgte : LISTA

**métodos**

ingresarLista(raiz, x) : no retorna valor.

concatenarLista(raiz1,raiz2, raiz3) : no retorna valor.

mostrarLista(raiz) : no retorna valor.

**significado**

*ingresaLista* registra en la lista con cabecera raiz de tipo LISTA un valor x de tipo ENTERO.

*concatenarLista* tienecomo precondición a la lista1 y a la lista2, y como postcondición a la lista3.

*mostrarLista* muestra los datos de la lista con cabecera raiz de tipo LISTA.

**Fin\_especificacion**

*Implementación del TAD*

#include <cstdlib>

#include <iostream>

using namespace std;

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std;

class ENTERO

{

int num;

public:

void menu()

{

cout<< "\nMENU DE OPCIONES\n";

cout<< "----------------\n" ;

cout<< "<1>. Ingresar Lista1\n";

cout<< "<2>. Ingresar Lista2\n";

cout<< "<3>. Concatenar Lista \n";

cout<< "<4>. Mostrar Lista\n";

cout<< "<5>. Salir\n";

}

void ingresarEntero()

{

cout<<"\n Ingresar entero: ";cin>>num;

}

void mostrarEntero()

{

cout<<num<<setw(4);

}

};

class LISTA

{

ENTERO info;

LISTA \*sgte;

public:

void ingresarLista(LISTA \*&raiz,ENTERO x)

{

LISTA \*q;

LISTA \*n =new LISTA;

n->info=x;

if(raiz==NULL)

raiz=n;

else

{

q=raiz;

while(q->sgte!= NULL)

q=q->sgte;

q->sgte=n;

}

n->sgte=NULL;

}

void concatenarLista(LISTA \*raiz1, LISTA \*raiz2, LISTA \*&raiz3)

{

LISTA \*p,\*q;

p=raiz1;

while(p!=NULL)

{

IngresarLista(raiz3,p->info);

p=p->sgte;

}

q=raiz2;

while(q!=NULL)

{

ingresarLista(raiz3,q->info);

q=q->sgte;

}

}

void mostrarLista(LISTA \*raiz)

{

LISTA \*p=raiz;

while(p!=NULL)

{

p->info.mostrarEntero();

p=p->sgte;

}

}

};

int main(int argc, char \*argv[])

{

char opcion;

ENTERO e;

LISTA list,\*raiz1=NULL,\*raiz2=NULL,\*raiz3=NULL;

do

{

e.menu();

cout<<"ingrese opcion : ";

opcion =cin.get();

switch (opcion)

{

case '1':

e.ingresarEntero();

list.ingresarLista(raiz1,e);

break;

case '2':

e.ingresarEntero();

list.ingresarLista(raiz2,e);

break;

case '3':

list.concatenarLista(raiz1,raiz2,raiz3);

cout<<"\n Se concateno exitosamente \n";

break;

case '4':

cout<<"\n Lista 1 \n";

list.mostrarLista(raiz1);

cout<<"\n Lista 2 \n";

list.mostrarLista(raiz2);

cout<<"\n Lista 3 \n";

list.mostrarLista(raiz3);

break;

}

cin.ignore();

}

while (opcion !='5');

system("PAUSE");

return EXIT\_SUCCESS;

}

Si ejecutamos el programa anterior, lo primero que veamos será el menú de opciones, listo para elegir el número de opción que elijamos, como vera se ha elegido para ingresar la opción 1 que ingresara el número 2 a la primera lista, osea la lista1:

MENU DE OPCIONES

----------------

<1>. Ingresar Lista1

<2>. Ingresar Lista2

<3>. Concatenar Lista

<4>. Mostrar Lista

<5>. Salir

Ingrese opcion : 1

Ingresar entero: 2

Una vez ingresado el número 2 a la primera lista, ingresamos el número 8 a la misma lista y los números 9, 7, y 3 a la segunda lista usando la opción 2, lo que nos llevara a tener dos listas con los siguientes datos:

**Lista2**

**Lista1**

3

7

**raiz2**

9

**raiz1**

8

2

**nulo**

**nulo**

Ahora veamos como el programa realiza la concatenación que se realiza al elegir la opción 3 del menú. Para empezar numeramos cada una de las instrucciones de la función concatenarLista para saber que instrucción es la que se esta usando en el proceso paso a paso de concatenar las dos listas anteriores:

void concatenarLista(LISTA \*raiz1, LISTA \*raiz2, LISTA \*&raiz3){

1. LISTA \*p,\*q;
2. p=raiz1;
3. while(p!=NULL){
4. ingresarLista(raiz3,p->info);
5. p=p->sgte;
6. }
7. q=raiz2;
8. while(q!=NULL){
9. ingresarLista(raiz3,q->info);
10. q=q->sgte;
11. }

}

(2) p=raiz1 (3) verdadero

2

**nulo**

**raiz1**

8

**p**

**raiz3**

**nulo**

(4) ingresarLista(raiz3,p->info) (5) p=p🡪sgte

2

**raiz3**

**nulo**

2

**nulo**

**raiz1**

8

**p**

(3) verdadero (4) ingresarLista(raiz3,p->info) (5) p=p->sgte

2

**raiz3**

2

**nulo**

**raiz1**

8

**p**

8

**nulo**

(3) falso (7) q=raiz2 (8) verdadero

9

**raiz2**

7

**q**

**nulo**

3

(9) ingresarLista(raiz3,q->info) (10) q=q🡪sgte

2

**raiz3**

8

**nulo**

9

9

**raiz2**

7

**q**

**nulo**

3

(8) verdadero (9) ingresarLista(raiz3,q->info) (10) q=q🡪sgte

2

**raiz3**

8

9

9

**raiz2**

7

**q**

**nulo**

3

**nulo**

7

(8) verdadero (9) ingresarLista(raiz3,q->info) (10) q=q🡪sgte

2

**raiz3**

8

9

9

**raiz2**

7

**q**

**nulo**

3

7

**nulo**

3

(8) falso y termina el proceso de concatenacion

Para verificar si el proceso se ha realizado correctamente solo elija la opción 4 y se mostraran las tres listas:

MENU DE OPCIONES

----------------

<1>. Ingresar Lista1

<2>. Ingresar Lista2

<3>. Concatenar Lista

<4>. Mostrar Lista

<5>. Salir

ingrese opcion : 4

Lista 1

2 8

Lista 2

9 7 3

Lista 3

2 8 9 7 3

Escriba la especificación, el algoritmo y la implementación en C++, para registrar en una lista lineal simple a los alumnos(cada uno con código y nombre) de una facultad. Se pide realizar la operación de eliminar un alumno según su ubicación en la lista.

**Observacion**

Suponiendo una lista como la que observamos abajo, el primer alumno(juan) en ingresar a la lista sería el alumno en una ubicación 0, el segundo(Julio) en una ubicación 1 y asi sucesivamente.

0950080

Juan

**nulo**

**raiz**

0950100

Julio

0951200

Manuel

Ubicación lógica: 0 1 2

Una vez ingresada la ubicación o posición del nodo a eliminar, se recomienda usar un puntero(r) adicional para facilitar el enlace del nodo anterior con el nodo siguiente del nodo que se quiere destruir.

0950080

Juan

**nulo**

**raiz**

0950100

Julio

0951200

Manuel

**r**

**q**

**sgte(q)**

posición = 1

Una vez encontrado el nodo a eliminar se procede de la siguiente manera:

0950080

Juan

**nulo**

**raiz**

0950100

Julio

0951200

Manuel

sgte(r)🡨sgte(q)

liberar(q)

**sgte(q)**

***x***

**q**

**r**

*Especificación de los TAD’s y los algoritmos*

**Especificación** ALUMNO

**variable**

codigo : entero.

nombre : cadena.

edad : entero.

**métodos**

menu() : no retorna valor.

ingresarDatos( ) : no retorna valor.

mostrarDatos( ) : no retornar valor.

**significado**

*menu* muestra las opciones a escoger.

*ingresarDatos* lee los datos de un alumno.

*mostrarDatos* visualiza los datos de un alumno.

**Fin\_especificación**

**Especificación** LISTA

**Usa** ALUMNO.

**variable**

info : ALUMNO.

sgte : LISTA.

**métodos**

ingresarLista(raiz, x) : no retorna valor.

eliminarPorPosicion(raiz, posicion) : retorna valor lógico.

mostrarLista(raiz) : no retorna valor.

**significado**

*ingresaLista* registra en la lista con cabecera raiz de tipo LISTA un valor x de tipo ALUMNO.

*eliminarPorPosicion* tiene como precondición a la lista1 y a su posición, y como postcondición, a la lista1.

*mostrarLista* muestra los datos de la lista con cabecera raiz del tipo LISTA.

**Fin\_especificación**

Funcion eliminarPorPosicion(raiz, posicion) : lógico

// Definir variables

LISTA: r, q=raiz

Logico: encontro 🡨 false

Entero: i 🡨0

Mientras (q != NULL and no encontro) Hacer

Si (posicion = i) entonces

Si (raiz=q) entoces

q 🡨 sgte(q )

liberar (raiz)

raiz🡨 q

Sino

Sgte( r) 🡨 sgte (q )

Liberar (q)

Fin\_si

encontro 🡨 true

Sino

r 🡨q

q 🡨 sgte(q)

i 🡨 i +1

Fin\_si

Fin\_mientras

retornar encontro

Fin\_funcion

*Implementación del TAD.*

#include <iostream>

#include <cstdlib>

using namespace std;

class ALUMNO

{

char codigo[10];

char nombre[40];

int edad;

public :

void menu()

{

cout<< "\nMENU DE OPCIONES\n";

cout<< "----------------\n" ;

cout<<"<1> Ingresar Lista\n";

cout<<"<2> Eliminar por posicion\n";

cout<<"<3> Mostrar Lista \n";

cout<<"<4> Salir \n";

}

void ingresarDatos()

{

fflush (stdin) ;

cout<<"\n ingrese codigo : "; gets(codigo);

cout<<"ingrese nombre : "; gets(nombre);

fflush (stdin) ;

cout<<"ingrese edad : "; cin>>edad;

}

void mostrarDatos()

{

cout<<"\n "<<codigo<<" "<<nombre<<" "<<edad;

}

};

class LISTA

{

ALUMNO info;

LISTA \*sgte;

public:

void ingresarLista(LISTA \*&raiz,ALUMNO x)

{

LISTA \*q;

LISTA \*n =new LISTA;

n->info=x;

if(raiz==NULL)

raiz=n;

else

{

q=raiz;

while(q->sgte!= NULL)

q=q->sgte;

q->sgte=n;

}

n->sgte=NULL;

}

bool eliminarPorPosicion(LISTA \*&raiz,int posicion){

LISTA \*r,\*q=raiz;

bool encontro =false;

int i=0;

while((q != NULL) && !encontro)

{

if(posicion == i){

if(raiz==q){

q=q->sgte;

delete raiz;

raiz=q;

}else{

r->sgte=q->sgte;

delete q;

}

encontro =true;

}else {

r=q;

q=q->sgte;

i++;

}

}

return encontro;

}

void mostrarLista(LISTA \*raiz)

{

LISTA \*q=raiz;

while(q!=NULL)

{

q->info.mostrarDatos();

q=q->sgte;

}

cout<<endl;

}

};

int main(int argc, char \*argv[])

{

char opcion;

int posicion;

ALUMNO a;

LISTA list,\*raiz=NULL;

do

{

a.menu();

cout<<"ingrese opcion : ";

opcion=cin.get();

switch (opcion)

{

case '1':

a.ingresarDatos();

list.ingresarLista(raiz,a);

break;

case '2':

cout<<"\n Ingrese posicion de alumno a eliminar: ";

cin>>posicion;

if(list.eliminarPorPosicion(raiz,posicion))

cout<<"\n Se elimino alumno";

else

cout<<"\n no se elimino alumno";

break;

case '3':

list.mostrarLista(raiz);

break;

}

cin.ignore( );

}

while (opcion !='4');

system("PAUSE");

return EXIT\_SUCCESS;

}