#include <cstdlib>

#include <iostream>

#include<math.h>

#include "Listas.cpp"

using namespace std;

struct Paquete

{

char origen[16];

char destino[16];

char protocolo[20];

char dato[980];

int suma;

};

struct Protocolo

{

char nombre[20];

Lista<Paquete> \*listaPaquetes;

};

void insertarPaquete(Lista<Protocolo \*> \*protocolos, Paquete p)

{

if( p.suma == strlen(p.dato))

{

bool inserte = false;

for( int i = 0; i < tamano(protocolos); i++)

{

Protocolo \* temp = obtenerDato(protocolos, i);

if( strcmp(p.protocolo, temp->nombre) == 0)

{

insertar(temp->listaPaquetes, p);

inserte = true;

break;

}

}

if( !inserte)

{

Protocolo \*nuevo = new Protocolo;

strcpy(nuevo->nombre, p.protocolo);

nuevo->listaPaquetes = crearLista<Paquete>();

insertar(nuevo->listaPaquetes, p);

insertar(protocolos, nuevo);

}

}

}

Lista<Paquete> \*buscarPaquetes (Lista<Protocolo \*> \*protocolos, char \*ip, char \*protocolo)

{

Lista<Paquete> \*listaResultado = crearLista<Paquete>();

for( int i = 0; i < tamano(protocolos); i++)

{

Protocolo \*temp = obtenerDato(protocolos, i);

if( strcmp(temp->nombre, protocolo) == 0)

{

for( int j = 0; j < tamano(temp->listaPaquetes); j++)

{

Paquete aux = obtenerDato(temp->listaPaquetes, j);

{

if( strcmp(ip, aux.origen) == 0 || strcmp(ip, aux.destino)== 0)

{

insertar(listaResultado, aux);

}

}

}

}

}

return listaResultado;

}

void cambiarMensajes( Lista<Protocolo \*> \*protocolos)

{

for( int i = 0; i < tamano(protocolos); i++)

{

Protocolo \*temp = obtenerDato(protocolos, i);

if( strcmp(temp->nombre, "SECRET MESSAGE")== 0)

{

Nodo<Paquete> \*aux = temp->listaPaquetes->cab;

while( aux != NULL)

{

int tam = strlen(aux->info.dato);

char \*cadena = aux->info.dato;

for( int j = 0; j < tam; j++)

{

if( cadena[j] == '1')

{

cadena[j] = 'a';

}

if( cadena[j] == '2')

{

cadena[j] = 'e';

}

if( cadena[j] == '3')

{

cadena[j] = 'i';

}

if( cadena[j] == '4')

{

cadena[j] = 'o';

}

if( cadena[j] == '5')

{

cadena[j] = 'u';

}

}

aux = aux->sig;

}

}

}

}

void quemarDatos(Lista<Protocolo \*> \*protocolos)

{

Paquete p;

strcpy(p.origen, "10.8.1.255");

strcpy(p.destino, "10.2.36.25");

strcpy(p.protocolo, "DNS");

strcpy(p.dato, "34348 www[Sync] Seq=0 Len=0");

p.suma = 27;

insertarPaquete(protocolos, p);

strcpy(p.origen, "192.175.3.21");

strcpy(p.destino, "19.8.1.111");

strcpy(p.protocolo, "DNS");

strcpy(p.dato, "Response Mac Address");

p.suma = 20;

insertarPaquete(protocolos, p);

strcpy(p.origen, "192.168.0.1");

strcpy(p.destino, "10.2.36.25");

strcpy(p.protocolo, "TCP");

strcpy(p.dato, "Estandar query response.");

p.suma = 24;

insertarPaquete(protocolos, p);

strcpy(p.origen, "10.6.3.236");

strcpy(p.destino, "19.8.1.111");

strcpy(p.protocolo, "SECRET MESSAGE");

strcpy(p.dato, "T4m1 1n34s c4nstr53r l1 c4nf31nz1 y s4l4 m3n5t4s p1r1 d2str53rl1");

p.suma = strlen(p.dato);

insertarPaquete(protocolos, p);

strcpy(p.origen, "10.6.3.236");

strcpy(p.destino, "19.8.1.111");

strcpy(p.protocolo, "SECRET MESSAGE");

strcpy(p.dato, "2st2 2s 5n m2ns1j2 d2 pr52b1");

p.suma = strlen(p.dato);

insertarPaquete(protocolos, p);

}

void imprimirPaquetes(Lista<Protocolo \*> \*protocolos)

{

for( int i =0; i< tamano(protocolos); i++)

{

Protocolo \* temp = obtenerDato(protocolos, i);

cout<<"Protocolo: "<<temp->nombre<<endl;

for(int j = 0; j < tamano(temp->listaPaquetes); j++)

{

Paquete p = obtenerDato(temp->listaPaquetes, j);

cout<<"\t"<<p.origen<<endl;

cout<<"\t"<<p.destino<<endl;

cout<<"\t"<<p.protocolo<<endl;

cout<<"\t"<<p.dato<<endl;

cout<<"\t"<<p.suma<<endl;

cout<<endl;

}

cout<<"------------------------------"<<endl;

cout<<endl;

}

}

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

{

Lista<Protocolo \*> \*protocolos = crearLista<Protocolo \*>();

quemarDatos(protocolos);

imprimirPaquetes(protocolos);

cambiarMensajes(protocolos);

cout<<"Mensajes desencriptados:"<<endl<<endl<<endl;

Lista<Paquete> \* mensajes = buscarPaquetes(protocolos, "10.6.3.236", "SECRET MESSAGE");

for( int i = 0; i < tamano(mensajes); i++)

{

Paquete p = obtenerDato(mensajes, i);

cout<<"\t"<<p.origen<<endl;

cout<<"\t"<<p.destino<<endl;

cout<<"\t"<<p.protocolo<<endl;

cout<<"\t"<<p.dato<<endl;

cout<<"\t"<<p.suma<<endl;

cout<<endl;

}

system("pause");

return EXIT\_SUCCESS;

}

#include <cstdlib>

#include <iostream>

#include<math.h>

#include "listas.cpp"

using namespace std;

struct sInfo{

int numero;

Lista<int> \*listaValores;

};

sInfo crearInfo(int valor){

sInfo info;

info.numero=valor;

info.listaValores=crearLista<int>();

for (int i=valor; i <=valor +8 ;i++){

insertar<int>(info.listaValores,i);

}

return info;

}

void imprimirLista(Lista<sInfo>\* lista){

Nodo<sInfo> \* aux=lista->cab;

while(aux !=NULL){

cout << (aux->info).numero << "\n";

Nodo<int> \* auxValores=((aux->info).listaValores)->cab;

while(auxValores!=NULL){

cout << '\t' << auxValores->info<<endl;

auxValores=auxValores->sig;

}

aux=aux->sig;

}

}

void insertarValorLista(Lista<sInfo>\* lista,int valor){

Nodo<sInfo> \* aux=lista->cab;

bool inserto=false;

while(aux !=NULL){

cout << (aux->info).numero << "\n";

if ((valor >= (aux->info).numero) && (valor < (aux->sig->info).numero) ){ //Encuentra sublista en donde insertar

Nodo<int> \* auxValores=((aux->info).listaValores)->cab;

if ((aux->info).listaValores->tam <10){ // Hay espacio en la lista correspondiente

while(auxValores!=NULL && !inserto){

if ( auxValores->info >= valor){

//cout << "este es el sitio " << valor << endl;

Nodo<int> \* nodo = crearNodo<int>(valor);

nodo->sig=auxValores->sig;

auxValores->sig=nodo;

inserto=true;

(aux->info).listaValores->tam++;

}

cout << '\t' << auxValores->info<<endl;

auxValores=auxValores->sig;

}

if (!inserto){//el nodo a insertar es el ultimo

insertar<int>((aux->info).listaValores,valor);

}

}else{// No hay espacio en la sublista corespondiente es mayor a 10, hay que crear un nueva sublista

//Obtiene el dato del ˙ltimo nodo en la sublista donde no hay espacio y crea un nuevo nodo en la lista principal

sInfo info;

info.numero= obtenerDato((aux->info).listaValores, 10-3);

info.listaValores=crearLista<int>(); // Crea la nueva sublista

Nodo<sInfo> \* nodoNuevo = crearNodo<sInfo>(info);

// Crea e inserta el nuevo Nodo en la ubicaciÛn correspondiente para mantenere la lista ordenada

nodoNuevo->sig=aux->sig;

aux->sig=nodoNuevo;

//Elimina los tres ultimos nodos de la sublista en donde no hay espacio y los inserta en la nueva sublista

for (int i=3; i>0; i--){

insertar<int>(info.listaValores,obtenerDato((aux->info).listaValores, (aux->info).listaValores->tam-i));

eliminar<int>((aux->info).listaValores, (aux->info).listaValores->tam-i);

}

insertarValorLista(lista,valor); // LLAMADO RECURSIVO PARA INSERTAR DE NUEVO EL VALOR EN LA NUEVA SUBLISTA

}

}

aux=aux->sig;

}

}

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

{

Lista<sInfo> \* lista=crearLista<sInfo>();

for (int i=10; i < 50 ; i+=10){

sInfo info =crearInfo(i);

insertar<sInfo>(lista,info);

}

/\*

insertarValorLista(lista,38);

insertarValorLista(lista,32);

insertarValorLista(lista,36);

insertarValorLista(lista,32);

insertarValorLista(lista,33);

\*/

//insertarValorLista(lista,39);

//insertarValorLista(lista,37);

//insertarValorLista(lista,37);

//insertarValorLista(lista,36);

// insertarValorLista(lista,36);

// insertarValorLista(lista,36);

imprimirLista(lista);

system("PAUSE");

return EXIT\_SUCCESS;

}

#include <iostream>

#include "listas.h"

#include <string.h>

using namespace std;

sInfo crearInfo(int valor)

{

sInfo info;

info.materia = valor;

info.ListaEstudiantes= crearLista<Estudiante>();

return info;

}

Estudiante crear(char \*N, char \*C, long I)

{

Estudiante info;

info.nombre = N;

info.carrera = C;

info.ID = I;

return info;

}

void imprimirLista(Lista<sInfo>\* lista)

{

Nodo<sInfo> \* aux=lista->cab;

while(aux !=NULL)

{

cout << "-Materia "<<(aux->info).materia<<": "<<endl;

if(tamano((aux->info).ListaEstudiantes)>0)

{

Nodo<Estudiante> \*auxValores =((aux->info).ListaEstudiantes)->cab;

while(auxValores!=NULL)

{

cout << '\t' <<"øNombre: "<<(auxValores->info).nombre<<endl;

cout << '\t' <<"Carrera: "<<(auxValores->info).carrera<<endl;

cout << '\t' <<"ID: "<<(auxValores->info).ID<<endl;

auxValores=auxValores->sig;

}

cout<<endl;

}

aux=aux->sig;

}

}

void insertarValorLista(Lista<sInfo> \*lista,int asig,char \*nom,char \*car,long id)

{

bool inserto = false;

int Dondevoy = 0;

Lista <Estudiante> \*Materia;

Nodo<Estudiante> \*Aux2;

Nodo<sInfo> \*Aux = lista->cab;

Estudiante Nuevo = crear(nom,car,id);

for(int i=1; i<asig; i++)

Aux = Aux->sig; //Me ubico en el nodo de la materia

Materia = (Aux->info).ListaEstudiantes; //Lista de estudiantes de la mateira

Aux2 = Materia->cab; //Me ubico en la cabeza de la sublista de la materia

if(tamano(Materia) == 0)

{

insertar<Estudiante>(Materia,Nuevo);

inserto = 1;

Aux2 = Materia->cab;

}

else

{

while(Aux2 != NULL && inserto != 1)

{

if(tamano(Materia) == 1)

{

if (strcmp(nom,((Aux2->info).nombre)) >=0)

insertar<Estudiante>(Materia,Nuevo);

else

insertarAlInicio<Estudiante>(Materia,Nuevo);

inserto = 1;

Aux2 = Materia->cab;

}

else if ((strcmp(nom,((Aux2->info).nombre))<=0) && Dondevoy == 0)

{//CASO ESPECIAL DONDE LA CABEZA SEA MAYOR QUE EL NOMBRE QUE INGRESE

insertarAlInicio<Estudiante>(Materia,Nuevo);

inserto = 1;

Aux2 = Materia->cab;

}

else if((strcmp(nom,((Aux2->info).nombre))>=0)&& Dondevoy == tamano(Materia)-1)

{

insertar<Estudiante>(Materia,Nuevo);

inserto = 1;

Aux2 = Materia->cab;

}

else if ((strcmp(nom,((Aux2->info).nombre))>=0) && (strcmp(nom,((Aux2->sig)->info).nombre))<=0)

{

insertarEnUnaPosicion<Estudiante>(Materia,Nuevo,Dondevoy+2);

inserto = 1;

Aux2 = Materia->cab;

}

Dondevoy = Dondevoy + 1;

Aux2 = Aux2->sig;

}

}

}

void BuscarEstudiante(Lista<sInfo> \*lista,int asig,long id, int hacer)

{

Nodo<sInfo> \*Aux = lista->cab;

Nodo<Estudiante> \*Aux2;

int Aux3 = -1;

if(hacer == 1)

{

for(int i=1; i<asig; i++)

Aux = Aux->sig; //Me ubico en el nodo de la materia

Aux2 = ((Aux->info).ListaEstudiantes)->cab;

if(tamano((Aux->info).ListaEstudiantes)>0)

{

if( Buscar(Aux2,id,&Aux3) == 1)

cout<<"La persona con Id "<<id<< " esta en la asignatura "<<asig<<endl;

else

cout<<"La persona no se encuentra inscrita en la asignatura"<<asig<<endl;

}

else

cout<<"La lista no tiene informacion"<<endl;

}

else

{

cout<<"La persona con Id "<<id<<" esta en las siguientes materias: "<<endl;

int contador= 0;

while(Aux != NULL)

{

Aux2 = ((Aux->info).ListaEstudiantes)->cab;

if( Buscar(Aux2,id,&Aux3) == 1)

{

cout<<contador+1<<". "<<"Materia "<<(Aux->info).materia<<endl;

contador = contador + 1;

}

Aux = Aux->sig;

}

}

}

int Buscar(Nodo<Estudiante> \*Aux,long id, int \*Aux2)

{

if(Aux == NULL)

return 0;

else

{

if(id == (Aux->info).ID)

\*Aux2 = 1;

else

\*Aux2 = Buscar(Aux->sig,id,Aux2);

}

if(\*Aux2 == 0)

return 0;

else

return 1;

}

template <typename T>

Nodo<T> \* crearNodo(T dato)

{

Nodo<T> \*nodo = new Nodo<T>;

nodo->info=dato;

nodo->sig=NULL;

return nodo;

}

template<typename T>

void destruirLista(Lista<T> \*lista)

{

Nodo<T> \*aux = lista->cab;

while( !vacia(lista) )

{

eliminar(lista, 0);

}

delete lista;

}

template<typename T>

void imprimir(Lista<T> \* lista)

{

if(vacia(lista))

{

cout<<"La Lista no tiene Informacion"<<endl;

}

else

{

cout<<"La informacion de la Lista es:"<<endl;

Nodo<T> \* aux;

aux = lista->cab;

int pos = 1;

while(aux != NULL)

{

cout<<"Info "<<pos<<": "<<aux->info<<endl;

aux = aux->sig;

pos++;

}

}

}

//Este metodo cambia segun el tipo de dato de la informacion de la lista

//Inserta la informacion en la ultima posicion de la lista

template<typename T>

void insertar(Lista<T> \* lista, T infoNueva)

{

Nodo<T> \* nuevo;

nuevo = crearNodo<T>(infoNueva);

if(lista->tam == 0)

{

lista->cab = nuevo;

lista->tam++;

}

else

{

Nodo<T> \* aux;

aux = lista->cab;

while(aux->sig != NULL)

{

aux = aux->sig;

}

aux->sig = nuevo;

lista->tam++;

}

}

//Inserta la informacion en la primera posicion de la lista

template<typename T>

void insertarAlInicio(Lista<T> \* lista, T infoNueva)

{

Nodo<T> \* nuevo;

nuevo = crearNodo<T>(infoNueva);

nuevo->sig = lista->cab;

lista->cab = nuevo;

lista->tam++;

}

//Inserta la informacion en la primera posicion de la lista

template<typename T>

void insertarEnUnaPosicion (Lista<T> \* lista, T infoNueva, int pos)

{

Nodo<T> \* nuevo;

nuevo = crearNodo<T>(infoNueva);

Nodo<T> \* aux;

aux = lista->cab;

int p = 1;

while(aux != NULL)

{

if(pos-1 == p)

{

nuevo->sig=aux->sig;

aux->sig=nuevo;

break;

}

aux = aux->sig;

p++;

}

}