Programación III

5to. Semestre

Universidad Mariano Gálvez de Guatemala

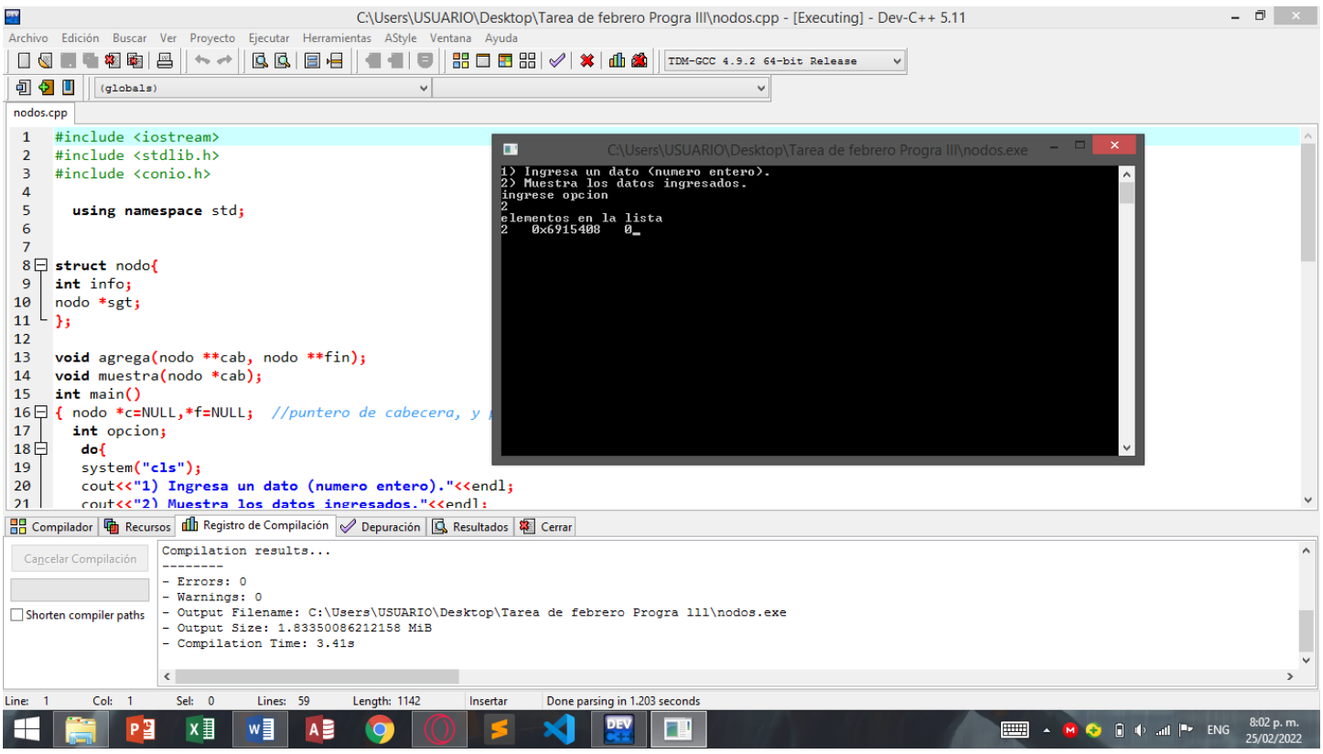
Ingeniería en Sistemas de Información y Ciencias de la Computación.

Ing. Pedro Donis

Programación lll

7490-20-26594
Anthony Efrain Estrada Barrera
aestradab1@miumg.edu

- **Ejecutando el programa nodos.cpp**

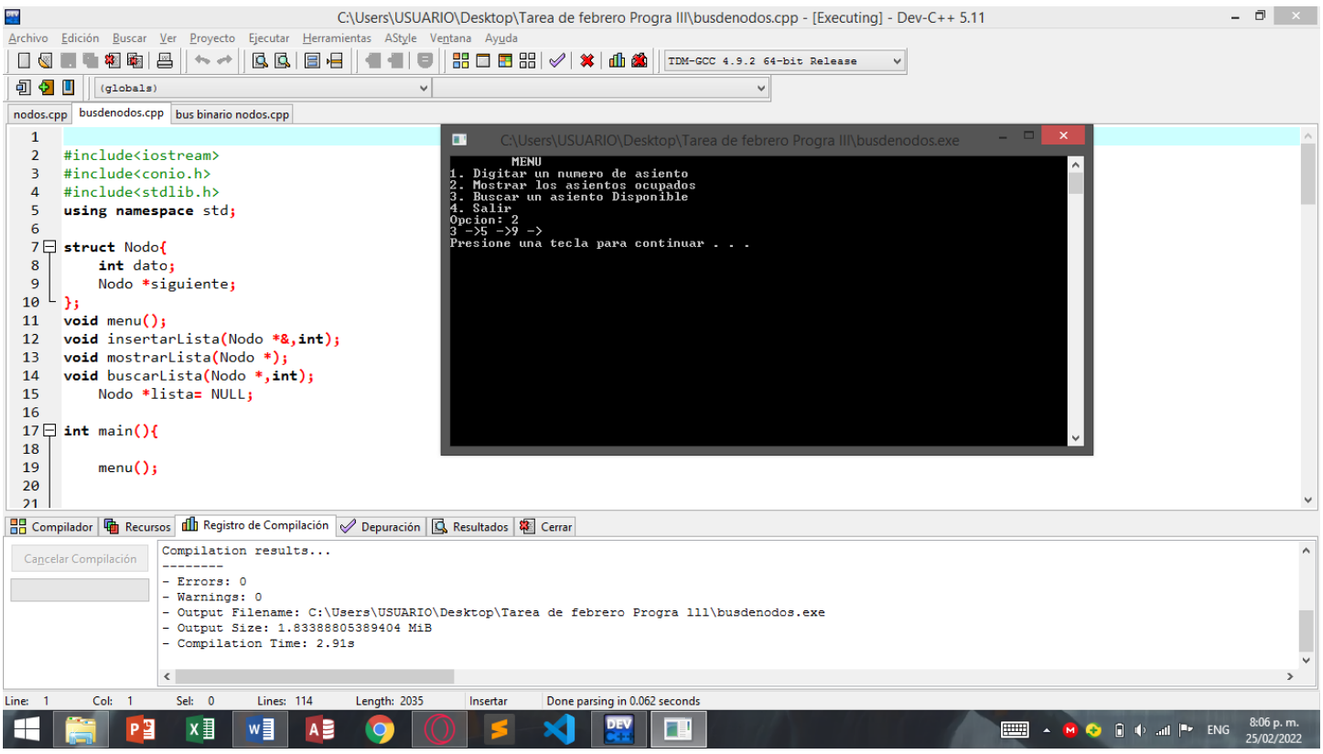


Código en C++

```
#include <iostream>

#include <stdlib.h>

#include <conio.h>


  using namespace std;



struct nodo{

int info;

nodo *sgt;

};


void agrega(nodo **cab, nodo **fin);

void muestra(nodo *cab);

int main()

{ nodo *c=NULL,*f=NULL;  //puntero de cabecera, y puntero de fin de lista

  int opcion;

   do{

   system("cls");

   cout<<"1) Ingresa un dato (numero entero)."<<endl;

   cout<<"2) Muestra los datos ingresados."<<endl;
```

```
  cout<<"ingrese opcion"<<endl;
  cin>>opcion;
  switch(opcion){
  case 0: exit(0);break;
  case 1: agrega(&c, &f);break;
  case 2: muestra(c);break;
  }
  }
  while(opcion!=0);
     system("PAUSE");
     return 0;
}
void agrega(nodo **cab, nodo **fin){
int num;
cout<<"ingrese informacion"<<endl;
cin>>num;
if((*cab)==NULL){
*cab = new nodo;
(*cab)->info =num;
(*cab)->sgt=NULL;
(*fin)=(*cab);
}else{
(*fin)->sgt=new nodo;
(*fin)->sgt->info=num;
(*fin)=(*fin)->sgt;
(*fin)->sgt=NULL;
}
}
void muestra(nodo *cab){
cout<<"elementos en la lista"<<endl;
nodo* temp;
temp=cab;
while ( temp != NULL){
cout<<temp->info<<"   "<<temp->sgt;
temp=temp->sgt;
}
getche();
}
```

➢ Ejecutando el programa de bus en C++



Código en C++

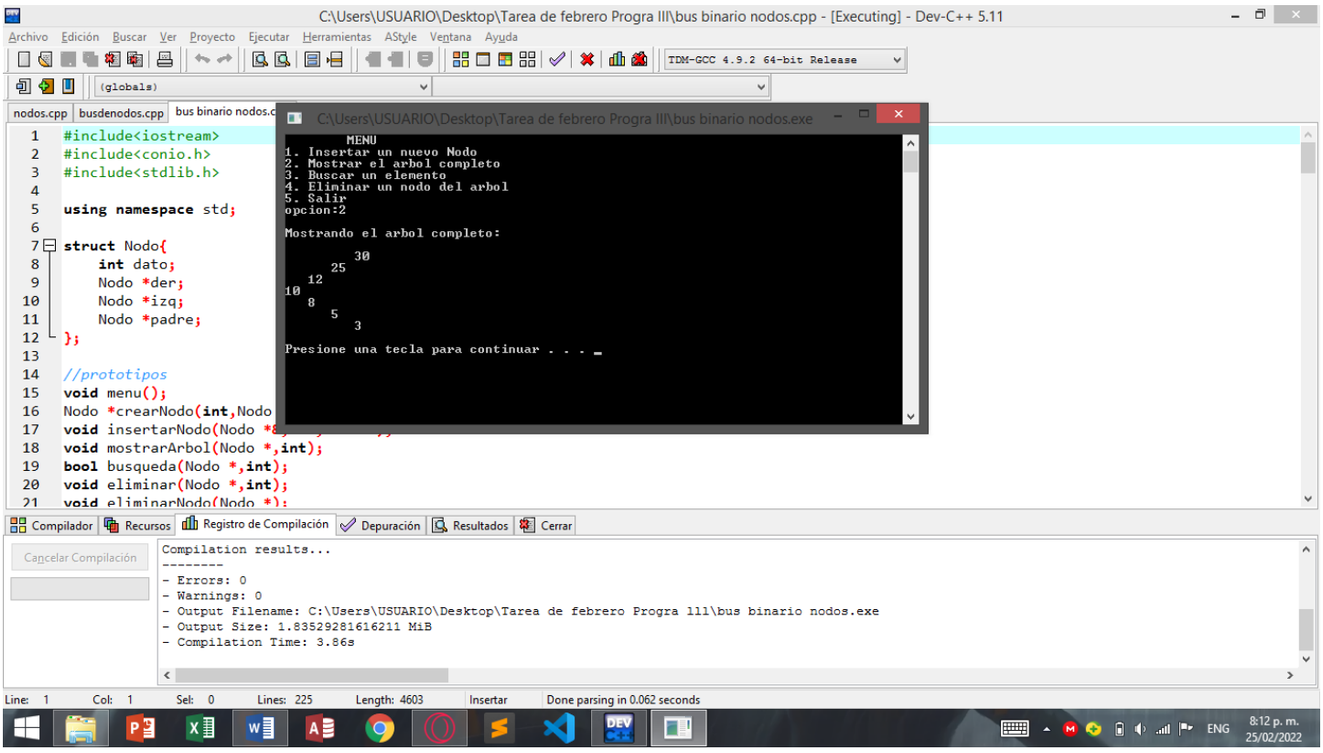
```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;


struct Nodo{
	int dato;
	Nodo *siguiente;
};
void menu();
void insertarLista(Nodo *&,int);
void mostrarLista(Nodo *);
void buscarLista(Nodo *,int);
	Nodo *lista= NULL;


int main(){

	menu();



	getch();
```

```
	return 0;
}

void menu(){
	int opcion,dato;

	do{
		cout<<"\tMENU\n";
		cout<<"1. Digitar un numero de asiento\n";
		cout<<"2. Mostrar los asientos ocupados\n";
		cout<<"3. Buscar un asiento Disponible\n";
		cout<<"4. Salir\n";
		cout<<"Opcion: ";
		cin>>opcion;

		switch(opcion){
			case 1: cout<<"Digite un numero:";
		    cin>>dato;
		insertarLista(lista,dato);
			cout<<"\n";
			system("pause");
			break;

			case 2: mostrarLista(lista);
			cout<<"\n";
			system("pause");
			 break;

			 case 3: cout<<"\nDigite un numero a buscar: ";
				cin>>dato;
				buscarLista(lista,dato);
				cout<<"\n";
				system("pause");
			  break;

		}
		system("cls");
	}while(opcion != 4);
```

```
}

void insertarLista(Nodo *&lista, int n){
        Nodo *nuevo_nodo = new Nodo();
        nuevo_nodo->dato = n;

        Nodo *aux1 = lista;
        Nodo *aux2;

        while((aux1 != NULL) && (aux1->dato < n)){
                aux2 = aux1;
                aux1 =aux1->siguiente;
        }

        if(lista == aux1){
                lista = nuevo_nodo;
        }
        else{
                aux2->siguiente = nuevo_nodo;
        }
      nuevo_nodo->siguiente = aux1;



}

void mostrarLista(Nodo *lista){
        Nodo *actual = new Nodo();
        actual = lista;

        while(actual != NULL){
                cout<<actual->dato<<" ->";
                actual = actual->siguiente;
        }
}

void buscarLista(Nodo *lista,int n){
   bool band =false;
```

```
        Nodo *actual=new Nodo();
        actual=lista;

        while((actual !=NULL)&&(actual->dato <=n)){
                if(actual->dato==n){
                        band = true;
                }
                actual = actual->siguiente;
        }
        if(band == true){
                cout<<"El asiento"<<n<<" esta ocupado\n";
        }
        else{
                        cout<<"El asiento "<<n<<" esta disponible\n";
        }
}
```

➢ Ejecutando el programa de ejemplo de nodos C++



Código en C++

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>

using namespace std;

struct Nodo{
	int dato;
	Nodo *der;
	Nodo *izq;
	Nodo *padre;
};

//prototipos
void menu();
Nodo *crearNodo(int,Nodo *);
void insertarNodo(Nodo *&,int,Nodo *);
void mostrarArbol(Nodo *,int);
bool busqueda(Nodo *,int);
void eliminar(Nodo *,int);
void eliminarNodo(Nodo *);
Nodo *minimo(Nodo *);
```

```
void reemplazar(Nodo *,Nodo *);
void destruirNodo(Nodo *);

Nodo *arbol = NULL;
int main(){
	menu();



	getch();
	return 0;
}

void menu(){
	int dato,opcion,contador=0;
	do{
		cout<<"\tMENU"<<endl;
		cout<<"1. Insertar un nuevo Nodo"<<endl;
		cout<<"2. Mostrar el arbol completo"<<endl;
		cout<<"3. Buscar un elemento"<<endl;
		cout<<"4. Eliminar un nodo del arbol"<<endl;
		cout<<"5. Salir"<<endl;
		cout<<"opcion:";
		cin>>opcion;

		switch(opcion){
			case 1:
			    cout<<"\nDigite un numero: ";
			    cin>>dato;
			    insertarNodo(arbol,dato,NULL);
			    cout<<"\n";
			    system("pause");
			    break;

			case 2:
			    cout<<"\nMostrando el arbol completo:\n\n";
			    mostrarArbol(arbol,contador);
			   cout<<"\n";
			    system("pause");
```

```
                    break;


                case 3:
                    cout<<"\nDigite el elemento a buscar: ";
                    cin>>dato;
                    if(busqueda(arbol,dato)== true){
                        cout<<"\nElemento "<<dato<<" a sido encontrado en el arbol\n";
                        }
                        else{
                            cout<<"\nElemento no encontrado\n";
                        }
                    cout<<"\n";
                    system("pause");
                    break;

                case 4: cout<<"\nDigite el numero a eliminar: ";
                    cin>>dato;
                        eliminar(arbol,dato);
                        cout<<"\n";
                        system("pause");
                         break;
            }
            system("cls");
        }while(opcion !=5);
}
//Funcion para crear un nuevo Nodo
Nodo *crearNodo(int n, Nodo *padre){
        Nodo *nuevo_nodo= new Nodo();

        nuevo_nodo-> dato= n;
        nuevo_nodo-> der= NULL;
        nuevo_nodo-> izq= NULL;
        nuevo_nodo-> padre;

        return nuevo_nodo;
}

//Funcion para insertar nodos en el arbol
```

```
void insertarNodo(Nodo *&arbol,int n, Nodo *padre){
	if(arbol==NULL){
		Nodo *nuevo_nodo = crearNodo (n,padre);
		arbol = nuevo_nodo;
	}
    else{
	int valorRaiz = arbol->dato;
	if(n < valorRaiz){
		insertarNodo(arbol->izq,n,arbol);
		 }

		else{
			insertarNodo(arbol->der,n,arbol);
		}
	  }
	}

//Fundion para mostrar el arbol completo
void mostrarArbol(Nodo *arbol, int cont){
	if(arbol == NULL){
		return;
	}
	else{
		mostrarArbol(arbol->der,cont+1);
		for(int i=0; i<cont;i++){
			cout<<"   ";
		}
		cout<<arbol->dato<<endl;
		mostrarArbol(arbol->izq,cont+1);
	}
}

//funsion para buscar un elemento en el arbol
bool busqueda(Nodo *arbol, int n){
	if(arbol==NULL){
		return false;
	}
	else if(arbol->dato == n){
```

```
                return true;
        }
        else if(n < arbol->dato){
                return busqueda(arbol->izq,n);
        }
        else{
                return busqueda(arbol->der,n);
        }
}

//Eliminar un nodo en el arbol
void eliminar(Nodo *arbol,int n){
        if(arbol == NULL){
                return;
        }
        else if(n< arbol->dato){
                eliminar(arbol->izq,n);
        }
        else if(n> arbol->dato){
                eliminar(arbol->der,n);
                }
        else{
                eliminarNodo(arbol);
        }

}

//funcion para determinar el nodo mas izq posible
Nodo *minimo(Nodo *arbol){
        if(arbol == NULL){
                return NULL;
        }
        if(arbol->izq){
                return minimo(arbol->izq);
        }
        else{
                return arbol;
        }
```

```
}

//funsion para reemplazar dos nodos
void reemplazar(Nodo *arbol,Nodo *nuevoNodo){
	if(arbol->padre){
		//arbol->padre hay que asignarle su nuevo hijo
	if(arbol->dato == arbol->padre->izq->dato){
		arbol->padre->izq=nuevoNodo;
	  }
	  else if(arbol->dato == arbol->padre->der->dato){
		arbol->padre->der=nuevoNodo;
	  }
	}
	if(nuevoNodo){
		//procedemos a asignarle su nuevo padre
		nuevoNodo->padre= arbol->padre;
	}
}

//funcion para destruir un nodo
void destruirNodo(Nodo *nodo){
	nodo->izq=NULL;
	nodo->der=NULL;

	delete nodo;
}




//funcion eliminar el nodo encontrado
void eliminarNodo(Nodo *nodoEliminar){
	if(nodoEliminar->izq && nodoEliminar->der){
		Nodo *menor=minimo(nodoEliminar->der);
		nodoEliminar->dato=menor->dato;
		eliminarNodo(menor);
	}
	else if(nodoEliminar->izq){
```

```
            reemplazar(nodoEliminar, nodoEliminar->izq);
            destruirNodo(nodoEliminar);
        }
        else if(nodoEliminar->der){
            reemplazar(nodoEliminar, nodoEliminar->der);
            destruirNodo(nodoEliminar);
        }
        else{//no tiene hijos
            reemplazar(nodoEliminar,NULL);
            destruirNodo(nodoEliminar);

        }
}
```