Práctica 10

Pilas

1. Ejecute el programa ***Dev-C++***.
2. En el menú ***Archivo*** seleccione la opción ***Nuevo*** y a continuación ***Archivo Fuente***.
3. En el nuevo documento transcriba el código del siguiente programa, en el cual se utiliza un arreglo para modelar una pila, en la que se realizan las operaciones de pop y push, el inconveniente principal es que la pila no puede rebasar el tamaño máximo fijado:

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

const int MAX=6;

template<class T>
class Pila {
   private: T Arreglo[MAX];
            int cima;
   public:  Pila();
            void Push(const T Dato);
            int Pop(void);
            int PilaLlena(void);
            int PilaVacia(void);
            void Ver(void);
};

template<class T>
Pila<T>::Pila() {
   cima=-1;
}

template<class T>
int Pila<T>::PilaVacia(void) {
   return (cima==-1);
}

template<class T>
int Pila<T>::PilaLlena(void) {
   return (cima==(MAX-1));
}

template<class T>
```

```
void Pila<T>::Push(const T Dato) {
      if(PilaLlena()) cout<<"Pila llena\n";
      else Arreglo[++cima]=Dato;
}

template<class T>
int Pila<T>::Pop(void) {
  int Val;
  if(!PilaVacia()) {
    Val=Arreglo[cima--];
    return Val;
     }
}

template<class T>
void Pila<T>::Ver(void) {
  int i=0, j;
  j=cima;
  if(j==-1) cout<<"\nNo hay datos que mostrar";
  else {
    cout<<endl<<"Los datos contenidos en la pila son: ";
    while(j>=i) {
      cout<< setw(5) <<Arreglo[j];
      j--;
    }
  }
  cout<<endl;
}

int main(void) {
  Pila<int> pila;
  char opc;
  int dato;
  cout<<"Implementacion de una pila estatica"<<endl<<endl;
  do {
     cout<<"\nMenu de opciones:";
     cout<<"\n1.- Insertar\n2.- Eliminar\n3.- Ver\n4.- Salir\n";
     cout<<"\nElija una opcion:";
     cin>>opc;
     switch(opc) {
       case '1': if(!pila.PilaLlena()) {
                    cout<<"\nDame el dato a insertar: ";
                    cin>>dato;
                    pila.Push(dato);
                 }
                 else cout<<"\nLa Pila esta llena.";
```

```
                break;
        case '2': if(pila.PilaVacia())cout<<"\nLa Pila esta vacia\n";
                else  cout<<"El dato eliminado es: "<<pila.Pop();
                break;
        case '3': pila.Ver();
                break;
      }
  } while(opc!='4');
}
```

4. Ahora localice el icono de ***Compilar*** que está al inicio de la barra de herramientas y oprímalo, proporcione el nombre ***ejercicio 10A*** y elija el destino que guste para guardarlo.
5. Ahora ejecútelo y pruebe las operaciones de su pila.
6. En el menú ***Archivo*** seleccione la opción ***Nuevo*** y a continuación ***Archivo Fuente***.
7. En el nuevo documento transcriba el código del siguiente programa, en el cual se construye una pila de forma dinámica utilizando listas, el código es prácticamente el mismo de la lista simple, hemos renombrado al apuntador ***Ie***, que indicaba el inicio de la lista y ahora le llamamos ***Cima*** de la pila, hemos renombrado los métodos ***Ins_Ini*** y ***Eli_Ini*** como ***Push*** y ***Pop*** respectivamente y descartamos los métodos ***Ins_Fin*** y ***Eli_Fin*** que no se utilizan en las pilas.

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

template<class T>
class Nodo {
   public: T info;
           Nodo<T> *sig;
           Nodo( const T);
};

template<class T>
class Pila {
    private: Nodo<T> *Cima;

    public:  Pila();
             ~Pila();
             void Push(const T);
             T Pop();
             void Ver();
             int Vacia();
};

template<class T>
Nodo<T>::Nodo(const T v) {
```

```
    info = v;
    sig = NULL;
}

template<class T>
Pila<T>::Pila() {
     Cima = NULL;
}

template<class T>
Pila<T>::~Pila() {
     Nodo<T> *sale;
     T val;
     while(Cima) {
          sale=Cima;
          Cima=Cima->sig;
          val=sale->info;
          delete(sale);
          cout<<"Bloque de memoria Liberado: "<<val<<endl;
     }
     delete(Cima);
}

template<class T>
void Pila<T>::Push(const T v) {
    Nodo<T> *nvo = new Nodo<T>(v);
    nvo->sig=Cima;
    Cima=nvo;
}

template<class T>
T Pila<T>::Pop() {
    Nodo<T> *sale;
    T val;
    if(Vacia()) val=0;
    else {
          sale=Cima;
          Cima=sale->sig;
          val=sale->info;
          delete(sale);
    }
    return val;
}

template<class T>
int Pila<T>::Vacia() {
```

```
    return Cima==NULL ? 1 : 0;
}

template<class T>
void Pila<T>::Ver()  {
   Nodo<T> *tmp = Cima;
   if(Vacia()) cout<<"No existe nada en la pila.";
   else {
      cout <<endl<<"La pila contiene: ";
      while (tmp){
          cout << setw(5) << tmp->info;
          tmp = tmp->sig;
      }
   }
   cout <<endl;
}

int main(void) {
    Pila <int> pila;
    int valor;
    char opc;
    cout<<"\nImplementacion de una Pila mediante una Lista.\n\n";
    do {
      cout<<"\n1.- Insertar \n2.- Eliminar \n3.- Ver Pila\n4.- Salir\n\n";
      cout<<"Opcion: ";
      cin>>opc;
      switch(opc) {
         case '1': cout<<endl<<"Introduce un valor diferente de cero:  ";
                   cin>>valor;
                   pila.Push(valor);
                   break;
         case '2': valor=pila.Pop();
                   if(valor ==0) cout<<endl<<" La Pila esta vacia."<<endl;
                   else cout<<endl<<"Se elimino dato: "<< valor <<endl;
                   break;
         case '3': pila.Ver();
                   break;
      }
    } while(opc!='4');
}
```

8. Guárdelo con el nombre de ***ejercicio 10B***, compílelo, ejecútelo y pruebe las operaciones de su pila.
9. Fin de la Práctica.