Práctica 3: Estructuras de datos lineales.

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

DECSAI

Estructuras de Datos

Grado en Ingeniería Informática C

Índice de contenido

1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

1. Practicar con T.D.A lineales.
2. Establecer, en base a la eficiencia, la mejor representación para estos tipos de datos.
3. Seguir asimilando los conceptos de documentación de un tipo de dato abstracto (T.D.A)

Los requisitos para poder realizar esta práctica son:

1. Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
2. Haber estudiado el Tema 2: Abstracción de datos y Plantillas
3. Haber estudiado el Tema 3: T.D.A. Lineales

2. Tipos de datos abstractos lineales.

Los tipos de datos abstractos lineales se componen de una secuencia de elementos $a_0,a_1,...,a_{n-1}$ dispuestos en un dimensión. Las estructuras de datos lineales más relevantes son:

1. Vectores 1-d
2. Listas
3. Pilas (LIFO)
4. Colas (FIFO)

El nombre LIFO y FIFO hace referencia a como se acceden a los datos. Así tenemos LIFO (last input first output) el último en entrar es el primero en salir. O la política FIFO (first input first output) el primero en entrar es el primer en salir.

2.1. Pilas

Cuando usamos política LIFO se indica que todas las inserciones, consultas y borrados se hace sobre el **tope**. Esta política de acceso es la que implementan las PILAS.

Así si tenemos la secuencia de elementos 1,2,3 y vamos a almacenarlos en una Pila la forma de hacer las inserciones siempre es por el tope, como se ve en la figura 1.

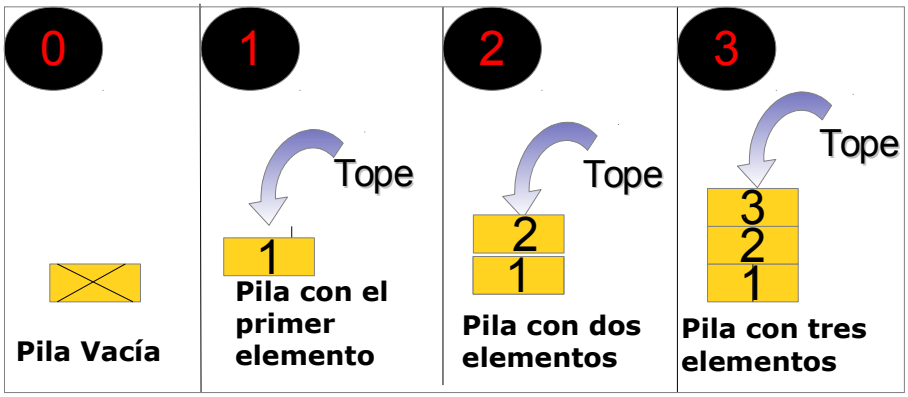


Figura 1: Proceso de inserción en un Pila

Cuando se consulta un dato en un pila siempre lo hacemos por el tope. **Nunca se va a poder acceder a datos que están debajo del tope sin previamente borrar lo que hubiese en el tope.**

Así si consultamos el tope de Pila creada en al figura 1 nos devolvería 3. Para poder acceder al 2 debemos borrar el tope y similarmente para poder acceder al uno deberemos a continuación borrar el 2.

3. Ejercicio

Crear el **T.D.A Pila_maxmin**. Esta Pila contiene un conjunto de números y además almacena para cada entrada el mayor y menor número de los almacenados en la pila. En la figura 2 se muestra un ejemplo de este tipo de Pila a la que se le ha insertado los datos 2, 7, 1 y 9.

En la primera inserción se pone el dato 2. Como la pila estaba vacía 2 será el máximo y mínimo

En el paso 2 se inserta el dato 7. Ahora para establecer cual es el máximo que hay que poner junto con 7, se compara 2 con el máximo previo, que es 2. En este caso como 7 es mayor este valor es el que se pone como máximo con el dato 7. Por otro lado el mínimo que tenemos es 2 y como es menor que 7 se mantiene.

En el paso 3 se inserta 1 y se vuelve a repetir el proceso de comparación. Por lo tanto se compara 1 con el máximo previo que es 7, como 7 es mayor se pone 7 como máximo del dato 1. Y como mínimo se pone 1 ya que es menor que 2.

Finalmente en el paso 4 se inserta el valor 9, y se compara 9 con 7, como 9 es mayor este se coloca como máximo. El mínimo se mantiene a 1.

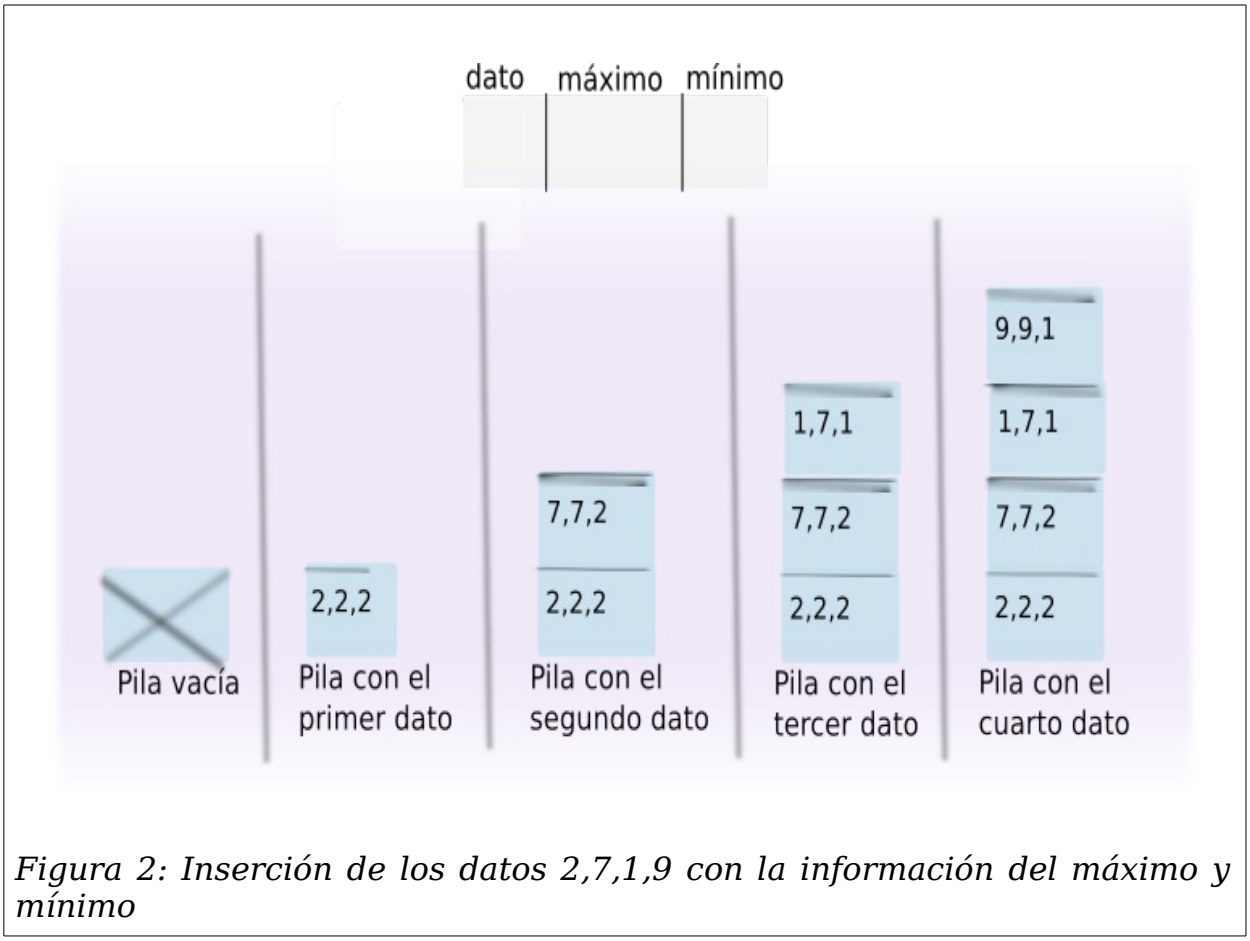


Figura 2: Inserción de los datos 2,7,1,9 con la información del máximo y mínimo

Por lo tanto si queremos consultar cual es el máximo o mínimo de los datos almacenados en la Pila simplemente tenemos que consultar el TOPE y acceder al campo máximo o mínimo. Suponer que ahora eliminamos el TOPE (ver paso 3 de la figura 2). Ahora nuestro máximo sería 7 del conjunto de datos {1,7,2} y como mínimo seria 1.

3.1. Tareas a realizar

El alumno debe llevar a cabo las siguientes tareas:

1. Dar una especificación del **T.D.A. Pila_maxmin**
2. Definir el conjunto de operaciones con sus especificaciones
3. Usar tres estructuras de datos para implementar el T.D.A. Pila_maxmin
    - Como una Lista (usando Celdas enlazadas con cabecera). Esta implementación será el módulo Pila_maxmin_List (Pila_maxmin_List.h y Pila_maxmin_List.cpp). La clase Lista deberá ser una clase plantilla (o clase template).
    - Como un Vector Dinámico. Esta implementación será el módulo Pila_maxmin_VD (Pila_maxmin_VD.h y Pila_maxmin_VD.cpp). La clase Vector Dinámico deberá ser una clase plantilla (o clase template).
    - Como un Cola. Esta implementación se desarrollará en el módulo Pila_maxmin_Cola (Pila_maxmin_Cola.h y Pila_maxmin_Cola.cpp). También será una clase plantilla (o clase template).
4. Probar los módulos con programas test.

IMPORTANTE: Para llevar a cabo la implementación de los diferentes módulos no se podrá usar la STL.

A continuación se pueden ver el fichero *h de las tres pilas que debemos implementar. Como se puede observar en todas la clase se llama Pila.

```
//Pila_maxmin_VD.h
#include "VD.h"
1.  class Pila{
2.  private:
3.      VD<elemento> datos
4.  ….
5.  public:
6.  elemento tope()const;
7.  bool vacia()const;
8.  void poner (const T &
    v);
9.  void quitar();
10. int size()const;
11.
12. };
13.
```

```
//Pila_maxmin_List.h
#include "Lista.h"
14. class Pila{
15. private:
16.     Lista<elemento> datos
17. ….
18. public:
19. elemento tope()const;
20. bool vacia()const;
21. void poner (const T &
    v);
22. void quitar();
23. int size()const;
24.
25. };
```

```
//Pila_maxmin_Cola.h
26. #include "Cola.h"
27. class Pila{
28. private:
29.   Cola<elemento> datos
30. ….
31. public:
32. elemento tope()const;
33. bool vacia()const;
34. void poner (const T &
    v);
35. void quitar();
36. int size()const;
37.
38. };
```

Como se puede ver en los tres módulos se instancia a ***elemento*** VD, Lista y Cola. Elemento será un struct que contenga tres enteros: el valor, el máximo hasta el momento y el mínimo.

3.2. Módulos a desarrollar.

Para un buen diseño crearemos tres módulos: VD (vector dinámico), Lista y Cola (para éste último se puede usar los fuentes dados). Estos tres módulos se implementarán con **templates.** Una vez probado el buen funcionamiento de estos tres módulos desarrollaremos las tres variantes del problema propuesto:

- Pila_maxmin_List: módulo que implementa una pila (con máximo y mínimo) basándose en un lista con cabecera.
- Pila_maxmin_VD: módulo que implementa una pila (con máximo y mínimo) basándose en un vector dinámico.
- Pila_maxmin_Cola:módulo que implementa una pila (con máximo y mínimo) basándose en una cola.

```
//Pila_maxmin.h
#define CUAL_COMPILA 3
#if CUAL_COMPILA==1
#include <pila_maxmin_vd.h>
#elif CUAL_COMPILA==2
#include <pila_maxmin_list.h>
#else
 #include <pila_maxmin_cola.h>
#endif
```

```
//Pila_maxmin.cpp
#include "Pila_maxmin.h"

#if CUAL_COMPILA==1
#include <pila_maxmin_vd.cpp>
#elif CUAL_COMPILA==2
#include <pila_maxmin_list.cpp>
#else
#include <pila_maxmin_cola.cpp>
#endif
```

Además crearemos el fichero Pila_maxmin.h y el correspondiente Pila_maxmin.cpp con el código que se muestra a la izquierda

De esta forma eligiendo un valor para CUAL_COMPILA elegimos que representación usar para nuestra pila_maxmin. Por lo tanto ***cuando compilemos nuestros proyectos a la hora de compilar Pila solamente tenemos que compilar Pila_maxmin.cpp.*** **No se compila ni Pila_maxmin_VD.cpp, ni Pila_maxmin_Lista.cpp ni Pila_maxmin_Cola.cpp.**

De esta forma nuestro fichero usopilas_maxmin.cpp, que testea el **T.D.A Pila_maxmin** podría tener las siguientes líneas:

```
1.  #include <iostream>
2.  #include "pila_maxmin.h"
3.  using namespace std;
4.  int main(){
5.    Pila_maxmin p;
6.    int i;
7.    for ( i=10; i>=0 ; i--)
8.      p.poner(i);
9.    while (!p.vacia() ){
10.     elemento x = p.tope();
11.     cout << x<<endl;
12.     p.quitar();
13.   }
14.   return 0;
15. }
```

Si nos fijamos en la línea 2 este código incluye el fichero pila_maxmin.h. En el proceso de precompilación pila_maxmin.h se transforma en pila_maxmin_vd.h o en pila_maxmin_list.h o en pila_maxmin_cola.h dependiendo de cual sea el valor de *CUAL_COMPILA.*

Con respecto al tipo ***elemento,*** que aparece en la línea 10 del código, este se define como:

```
struct elemento{
 int ele; ///<elemento a almacenar
 int maximo; ///<el máximo
 int miniimo; ///< el mínimo
};
```

Para que el código usopilas_maxmin.cpp funcione, se deberá sobrecargar para el tipo ***elemento*** el operador <<.

3.3. Programa Orden Conjuntos datos.

Crear un programa denominado ***orden*** que dados dos conjuntos de datos de tipo entero nos diga:

1. Si un conjunto es menor (mayor) que otro. ***Un conjunto es menor que otro si todos sus elementos son menores a otro.***
2. Si un rango de valores de un conjunto esta contenido en el rango de valores de otro. ***El rango de valores de un conjunto está contenido en otro si el mínimo y máximo esta contenido en el mínimo y máximo del otro conjunto***

Por ejemplo dados los dos conjuntos siguientes:

- {7,9,2,1,0} es mayor que {-1,-9,-13} ya que el mínimo del primero 0 es mayor que el máximo del segundo -1.
- {-1,2,5,10} tiene un rango de valores [-1,10] que contiene al rango de {3,4,7} siendo

este [3,7].

Así desde la línea de órdenes se ejecutará

```
%prompt>orden conjunto1.txt conjunto2.txt
```

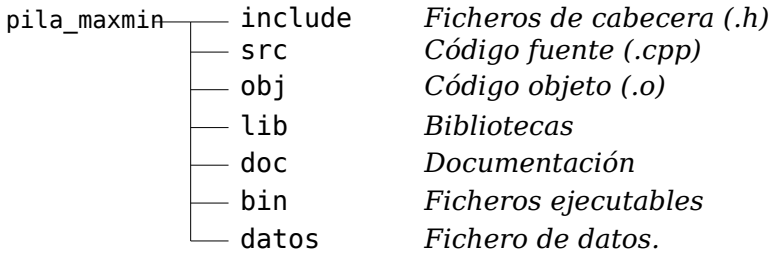
3.4. Tarea Extraordinaria.

La siguiente tarea que se propone es de carácter voluntario. El alumno tendrá que construir el ***T.D.A Cola_maxmin.*** Este T.D.A contiene un conjunto de enteros que se almacenan en una Cola (política FIFO) junto con el máximo y mínimo del conjunto. Para construir el ***T.D.A Cola_maxmin*** solamente se podrá usar el ***T.D.A Pila***

4. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre "pila_maxmin.*tgz*" y entregarlo antes de la fecha que se publicará en la página web de la asignatura. Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables, ni la carpeta datos. Es recomendable que haga una "limpieza" para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

El alumno debe incluir el archivo *Makefile* para realizar la compilación. Tenga en cuenta que los archivos deben estar distribuidos en directorios*:*



Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior (en el mismo nivel de la carpeta "*pila_maxmin*") para ejecutar:

```
prompt% tar zcv pila_maxmin.tgz pila_maxmin
```

tras lo cual, dispondrá de un nuevo archivo pila_maxmin.tgz que contiene la carpeta pila_maxmin así como todas las carpetas y archivos que cuelgan de ella.