UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Práctica 3: TDA lineales

Pila con máximo

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

Estructuras de Datos

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

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.

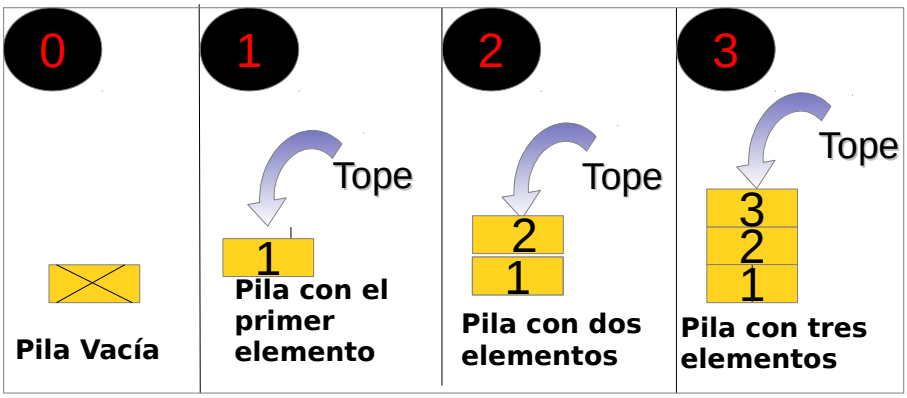


Figura 1.- Proceso de insercción en una Pila

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 de forma similar, para poder acceder al uno deberemos a continuación borrar el 2.

3.- Ejercicio

Crear el **T.D.A Pila_max (**Pila con máximo). Esta Pila contiene un conjunto de enteros pero además se almacena el valor máximo que existe 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, 6 y 9.

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

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.

En el paso 3 se inserta 6 y se vuelve a repetir el proceso de comparación. Por lo tanto se compara 6 con el máximo previo que es 7, como 7 es mayor se pone 7 como máximo del dato 6.

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

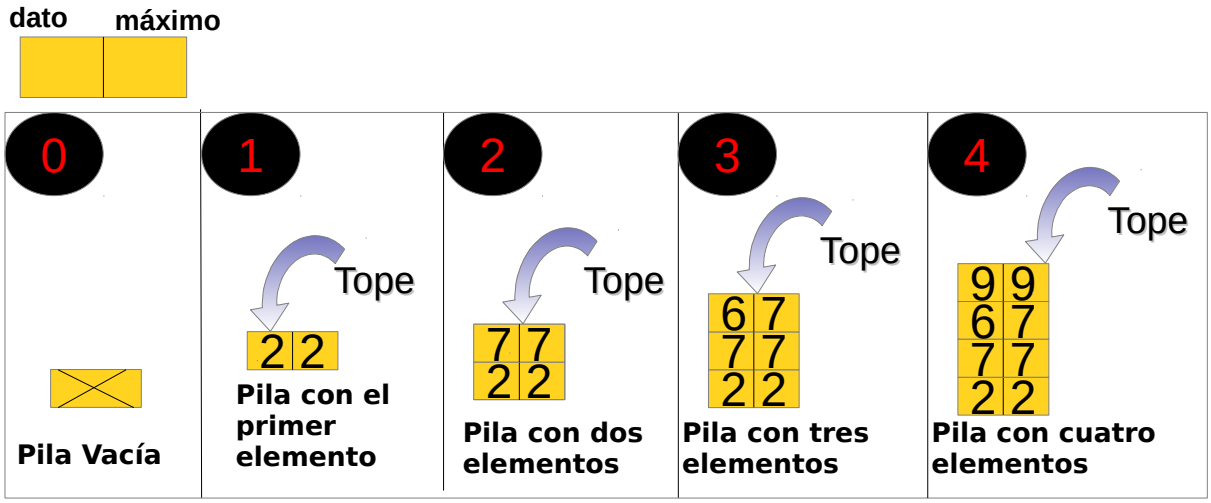


Figura 2: Inserción de los datos 2,7,6,9 con la información del máximo

Por lo tanto si queremos consultar cual es el máximo de los datos almacenados en la Pila simplemente tenemos que consultar el TOPE y acceder al campo máximo. Suponer que ahora eliminamos el TOPE (ver paso 3 de la figura 2. Ahora nuestro máximo sería 7 del conjunto de datos {6,7,2}.

3.1 Tareas a realizar

El estudiante debe llevar a cabo las siguientes tareas:

1. Dar una especificación del **T.D.A. Pila_max**
2. Definir el conjunto de operaciones con sus especificaciones
3. Usar dos estructuras de datos para implementar el T.D.A. Pila_max
    - Como un Cola. Esta implementación se desarrollará en el módulo Pila_max_Cola (Pila_max_Cola.h y Pila_max_Cola.cpp). **En este apartado, no se puede usar la STL**
    - Como un Vector Dinámico. Esta implementación será el módulo Pila_max_VD (Pila_max_VD.h y Pila_max_VD.cpp). **En este apartado si se quiere, puede usarse la clase vector<T> de la STL**).
4. Probar los módulos con programas test.

3.2 Módulos a desarrollar.

Para un buen diseño crearemos dos módulos: VD (vector dinámico), y Cola (para éste último se pueden usar los fuentes dados). Una vez probado el buen funcionamiento de estos dos módulos desarrollaremos las variantes del problema propuesto:

- **Pila_max_VD:** módulo que implementa una pila (con máximo) basándose en un TDA Vector dinámico (sin templates) que se suministra o usar la clase <vector> de la STL.
- **Pila_max_Cola:**módulo que implementa una pila (con máximo) basándose en una cola con templates que se suministra.

  Puede si se quiere crear el fichero **Pila_max.h** con el siguiente código:

```
1. #define CUAL_COMPILA 2
2. #if CUAL_COMPILA==1
3. #include <pila_max_vd.h>
4. #elif CUAL_COMPILA==2
5. #include <pila_max_cola.h>
6. #endif
```

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

```
1.  #include <iostream>
2.  #include "pila_max.h"
3.  using namespace std;
4.  int main(){
5.    Pila_max 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_max.h. En el proceso de precompilación pila_max.h se transforma en pila_max_vd.h o en pila_max_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
};
```

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

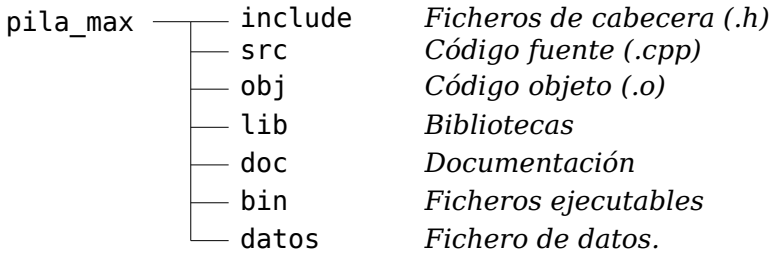
3.3 Tarea voluntaria.

Construir el ***T.D.A Cola_max.*** Este T.D.A contiene un conjunto de enteros que se almacenan en una Cola (política FIFO) junto con el máximo del conjunto. Para construir el ***T.D.A Cola_max*** solamente se podrá usar el ***T.D.A Pila. Se puede usar la clase stack<T> de la STL***

4.- Práctica a entregar

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

El estudiante debe incluir el archivo *Makefile* para realizar la compilación. Tener 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_max*") para ejecutar:

```
prompt% tar zcv pila_max.tgz pila_max
```

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

La práctica ***puede hacerse por parejas*** *y su puntuación máxima es de* ***0.5 puntos***

5.- Referencias

- Rosa Rodriguez-Sánchez, J. Fdez-Valdivia, J.A. García, Estructuras de Datos en C++. Un enfoque práctico.. 2020
- [GAR06b] Garrido, A. Fdez-Valdivia, J. "*Abstracción y estructuras de datos en C++*". Delta publicaciones, 2006.