UNIVERSIDAD DE GRANADA

Estructuras de datos

2º Grado en Ingeniería Informática

Práctica 3. Estructuras de datos lineales

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

Índice de contenidos

1 Introducción

Los objetivos que se pretenden alcanzar con la realización de este guión de prácticas son los siguientes:

- Practicar con tipos de datos abstractos (TDA) lineales.
- Establecer la mejor representación para estos tipos de datos.
- Practicar con el uso de doxygen para generar la documentación del TDA.

Para realizar correctamente este guión de prácticas se deben haber estudiado los siguientes temas:

- Tema 1: Introducción a la eficiencia de los algoritmos.
- Tema 2: Abstracción de datos.
- Tema 3: Tipos de datos contenedores básicos.

2 Tipos de datos abstractos (TDA) lineales

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

- Vectores 1-d (una dimensión).
- Listas.
- Pilas.
- Colas.

A continuación, se explican en detalle dos de estas estructuras de datos lineales: las pilas y las colas.

2.1 Pilas

Una pila es una estructura de datos que permite almacenar y recuperar datos. Está optimizada para realizar inserciones, consultas y borrados por solo uno de los extremos (denominado tope). El modo de acceso a sus elementos sigue la política LIFO (Last Input First Output), es decir, el primero en entrar es el primero en salir. Entre las operaciones típicas para el manejo de las pilas tenemos las siguientes:

- **Tope**. Consulta el elemento del tope.
- **Vacía**. Devuelve verdadero si la pila no tiene ningún elemento.
- **Quitar**. Elimina el elemento en el tope.
- **Poner**. Inserta un nuevo elemento en el tope.

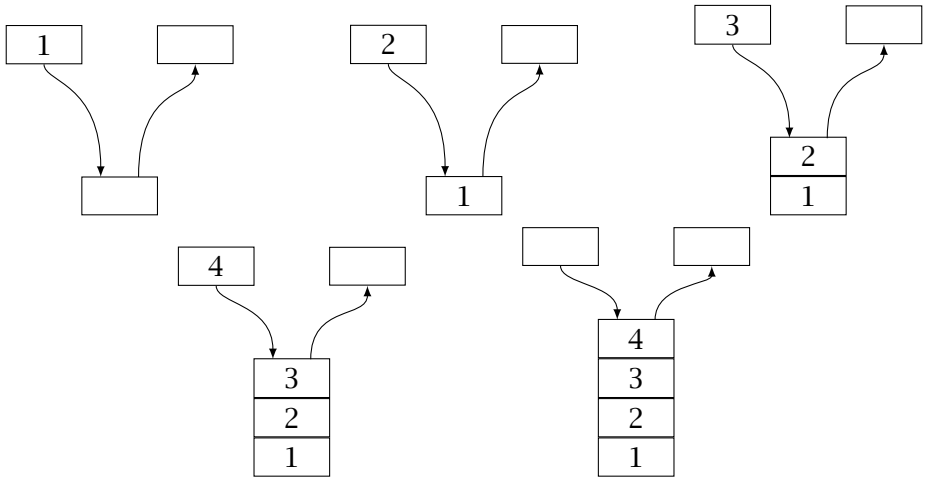


Figura 1. Proceso de inserción en una pila.

Si tenemos la secuencia de elementos 1, 2, 3 y 4, y vamos a almacenarlos en una pila, la forma de hacer las inserciones siempre es por el tope (ver Figura 1).

Cuando se consulta un dato en un pila siempre se hace por el tope. Nunca se va a poder acceder a datos que están debajo del tope sin previamente borrar lo que hubiese en él. Así, si consultamos el tope de la pila creada en la Figura 1 nos devolvería 4. Para poder acceder al 3 debemos borrar el tope y similarmente para poder acceder al 2 deberemos a continuación borrar el 3. Finalmente, para acceder al 1, habría que borrar previamente el 2.

2.2 Colas

Una cola es una estructura de datos caracterizada por ser una secuencia de elementos. El modo de acceso a sus elementos sigue la política FIFO (First Input First Output), en la que las inserciones se realizan por un extremo y los borrados y consultas por el otro. El extremo en el que está el primer elemento se llama frente y es por donde se hacen las consultas y borrados, mientras que el extremo en el que está el último valor se denomina última y es por donde se realizan las inserciones. Entre las operaciones típicas para el manejo de las colas tenemos las siguientes:

- **Frente**. Consulta el elemento del frente.
- **Vacía**. Devuelve verdadero si la cola está vacía.
- **Quitar**. Elimina el elemento que está en el frente.
- **Poner**. Añade un nuevo elemento por el final (por la posición última).

Si tenemos la secuencia de elementos 1, 2, 3, y 4, y vamos a almacenarlos en una cola, la forma de hacer las inserciones siempre es por el final (ver Figura 2). Por otro lado, cuando se consulta un elemento en una cola, siempre lo hacemos por el frente. Nunca se va a poder acceder a los elementos que están encima del frente sin previamente borrar lo que hubiese en él. Así, se consultamos el frente de la cola creada en la Figura 2, nos devolvería el elemento 1. Para poder acceder al elemento 2 debemos borrar el frente y similarmente para poder acceder al elemento 3 debemos a continuación borrar el elemento 2.

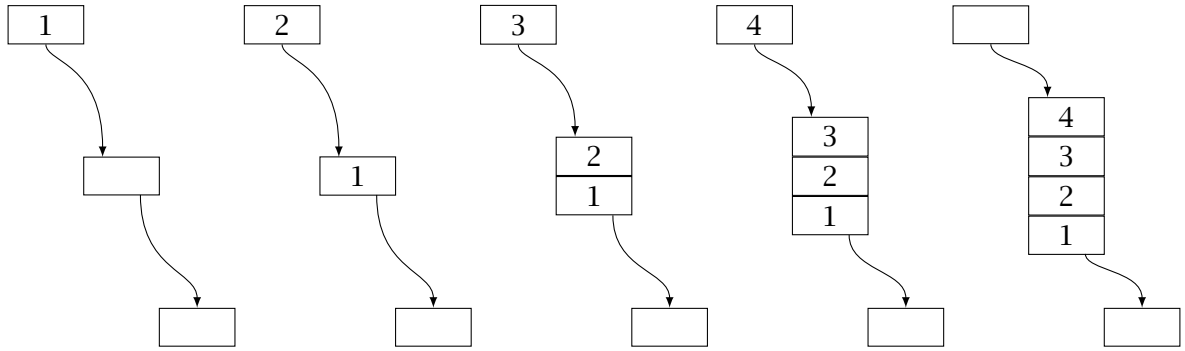


Figura 2. Proceso de inserción en una cola.

3 Ejercicio

Desarrollar el TDA `Cola_max` (Cola con máximo). Esta cola contiene un conjunto de elementos pero, además, almacena el valor máximo de los elementos que existen en la cola. En la Figura 3, se muestra un ejemplo de este tipo de cola a la que se le han insertado los valores enteros 2, 7, 9 y 6:

- En la primera inserción se pone el 2. Como la cola estaba vacía, este valor será el máximo.
- En el segundo paso, se inserta el 7 (tanto como valor del elemento como del máximo del mismo). Este valor se compara con el máximo del siguiente elemento, como es mayor (7 > 2), se pone como máximo en el siguiente elemento también.
- En el tercer paso, se inserta el 9 (tanto como valor del elemento como del máximo del mismo) y se vuelve a repetir el proceso de comparación de forma que se va insertando este valor en el máximo de los siguientes elementos mientras sea mayor que el máximo de los mismos.
- Finalmente, en el cuarto paso, se inserta el 6 (tanto como valor del elemento como del máximo del mismo). En este caso, como es menor que el máximo del siguiente elemento (9), no hay que seguir comparando.

De esta forma, si queremos consultar cuál es el máximo de los elementos almacenados en la cola, simplemente tenemos que consultar el frente y acceder al campo máximo.

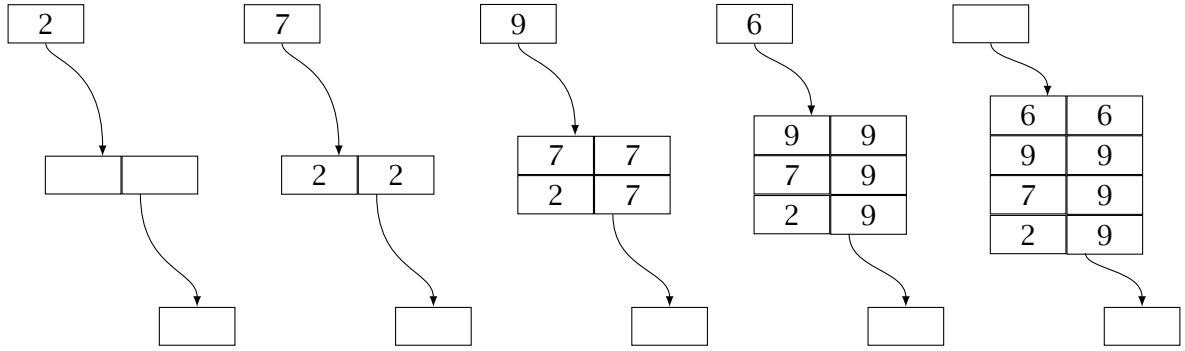


Figura 3. Inserción de 2, 7, 9 y 6, con la información del máximo.

Se pide desarrollar el TDA `Cola_max`. En particular, las tareas que hay que realizar son las siguientes:

- Dar una especificación del TDA `Cola_max`.
- Definir el conjunto de operaciones con sus especificaciones.
- Usar dos estructuras de datos para implementar el TDA `Cola_max`:
  - Vector dinámico. Esta implementación será el módulo `Cola_max_vd` (`Cola_max_vd.h` y `Cola_max_vd.cpp`).
  - Pilas. Esta implementación será el módulo `Cola_max_pila` (`Cola_max_pila.h` y `Cola_max_pila.cpp`).
- Documentar los TDA desarrollados.
- Haciendo uso de `pruebacola_max.cpp`, probar los TDA desarrollados.

3.1 Módulos a desarrollar

Se desarrollarán las dos variantes del ejercicio propuesto:

- `Cola_max_vd`. Este módulo implementa una cola (con máximo) basándose en un vector dinámico. Se valorará que la operación **Quitar** tenga la mayor eficiencia posible. Se tendrá que implementar como **template**.
- `Cola_max_pila`. Este módulo implementa una cola (con máximo) basándose en pilas. Una primera pila para inserciones y una segunda pila para consultas y borrados. Se tendrá que implementar como **template**.

Si fuera necesario para el desarrollo de los módulos anteriores, se pueden desarrollar más módulos.

3.2 Fichero de prueba

En el directorio **src** se encuentra el fichero `pruebacola_max.cpp` (ver Listado 3.1). Este código debe funcionar con los TDA desarrollados.

```
#include <iostream>
#include "Cola_max.h"

using namespace std;

int main(){

 Cola_max <int> p;
 p.poner(2);
 cout << "Se introduce el valor 2. El maximo es " << p.maximo() << endl;
 p.poner(5);
 cout << "Se introduce el valor 5. El maximo es " << p.maximo() << endl;
 p.poner(4);
 cout << "Se introduce el valor 4. El maximo es " << p.maximo() << endl;
 p.poner(9);
 cout << "Se introduce el valor 9. El maximo es " << p.maximo() << endl;
```

```
 p.poner(7);
 cout << "Se introduce el valor 7. El maximo es " << p.maximo() << endl;
 p.poner(8);
 cout << "Se introduce el valor 8. El maximo es " << p.maximo() << endl;

 cout << "La cola tiene " << p.num_elementos() << " elementos. ";
 cout << "El maximo es " << p.maximo() << endl;

 while(!p.vacia()){
   cout << "El frente contiene " << p.frente() << ". ";
   cout << "El maximo es " << p.maximo() << ". ";
   p.quitar();
   cout << "Se quita este valor" << endl;
 }

 Cola_max <float> q;
 q.poner(2.4);
 cout << "Se introduce el valor 2.4. El maximo es " << q.maximo() << endl;
 q.poner(5.5);
 cout << "Se introduce el valor 5.5. El maximo es " << q.maximo() << endl;
 q.poner(4.1);
 cout << "Se introduce el valor 4.1. El maximo es " << q.maximo() << endl;
 q.poner(9.6);
 cout << "Se introduce el valor 9.6. El maximo es " << q.maximo() << endl;
 q.poner(7.9);
 cout << "Se introduce el valor 7.9. El maximo es " << q.maximo() << endl;
 q.poner(8.3);
 cout << "Se introduce el valor 8.3. El maximo es " << q.maximo() << endl;

 cout << "La cola tiene " << q.num_elementos() << " elementos. ";
 cout << "El maximo es " << q.maximo() << endl;

 while(!q.vacia()){
   cout << "El frente contiene " << q.frente() << ". ";
   cout << "El maximo es " << q.maximo() << ". ";
   q.quitar();
   cout << "Se quita este valor" << endl;
 }

 return 0;
}
```

Listado 3.1. `pruebacola_max.cpp`.

Si nos fijamos en la segunda línea, se incluye el fichero `Cola_max.h`. En el proceso de precompilación, `Cola_max.h` se transforma en `Cola_max_vd.h` o en `Cola_max_pila.h` dependiendo del valor de `CUAL_COMPILA` (ver Listado 3.2).

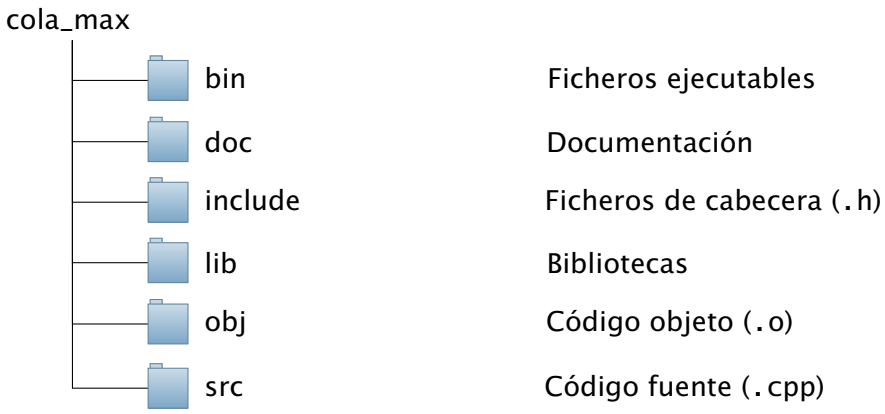
```
#define CUAL_COMPILA 1
#if CUAL_COMPILA == 1
 #include "Cola_max_vd.h"
#else
 #include "Cola_max_pila.h"
#endif
```

Listado 3.2. `Cola_max.h`.

4 Entrega

Se deberán empaquetar todos los ficheros relacionados con la práctica en un fichero con nombre `cola_max.tar`. No se incluirán ficheros objeto ni ficheros ejecutables. Por tanto, se deben eliminar los ficheros temporales y aquellos que se generen a partir de los fuentes.

Se deberá incluir el fichero `Makefile` para realizar la compilación. Los ficheros deben estar distribuidos en directorios:



Para realizar la entrega, se deben eliminar los ficheros que no se incluirán en ella. Estando en la carpeta superior (en el mismo nivel de la carpeta **cola_max**) se ejecutará la siguiente instrucción:

```
tar -cvf cola_max.tar cola_max
```

tras lo cual se dispondrá de un nuevo fichero llamado `cola_max.tar` que contendrá la carpeta **cola_max** y todas las carpetas y ficheros que cuelgan de ella.