

Tarea 4

Implementación de Tipos Abstractos de Datos

Curso 2020

Índice

1. Introducción y objetivos	2
2. Materiales	2
3. ¿Qué se pide?	2
4. Descripción de los módulos y algunas funciones	2
4.1. Módulos de tareas anteriores	2
4.2. Módulos nuevos	2
4.2.1. Módulo <i>pila</i>	2
4.2.2. Módulo <i>colaAvls</i>	3
4.2.3. Módulo <i>avl</i>	3
4.2.4. Módulo <i>conjunto</i>	4
4.2.5. Módulo <i>colaDePrioridad</i>	4
4.2.6. Módulo <i>mapping</i>	4
5. Entrega	4
5.1. Plazos de entrega	5
5.2. Identificación de los archivos de las entregas	5
5.3. Individualidad	5

1. Introducción y objetivos

El objetivo de esta tarea es la implementación de **Tipos Abstractos de Datos (TADs)** y de nuevas estructuras cumpliendo requerimientos de tiempo de ejecución. Mantenemos los módulos de las tareas anteriores. Los detalles sobre los módulos de tareas anteriores se presentan en la Sección 4.1.

También trabajaremos sobre módulos nuevos, que implementan pilas, árboles AVL, colas de árboles AVL, conjuntos, colas de prioridad y mapping. Los detalles de los módulos nuevos se presentan en la Sección 4.2.

Como corresponde al concepto de TAD no se puede usar la representación de un tipo fuera de su propio módulo.

El correcto uso de la memoria y el cumplimiento de las restricciones de tiempo será parte de la evaluación.

El resto del presente documento se organiza de la siguiente forma. En la Sección 2 se presenta una descripción de los materiales disponibles para realizar la presente tarea, y en la Sección 3 se detalla el trabajo a realizar. Luego, en la Sección 4 se explica mediante ejemplos el comportamiento esperado de algunas de las funciones a implementar. Por último, la Sección 5 describe el formato y mecanismo de entrega, así como los plazos para realizar la misma.

2. Materiales

Los materiales para realizar esta tarea se extraen de *MaterialesTarea4.tar.gz*. Ver el procedimiento en la Sección **Materiales** de [Funcionamiento y Reglamento del Laboratorio](#).

En esta tarea los archivos en el directorio `include` son los de la tarea anterior a los que se agregan `pila.h`, `avl.h`, `colaAvls.h`, `conjunto.h`, `colaDePrioridad.h` y `mapping.h`.

En el directorio `src` se incluyen ya implementados `utils.cpp` e `info.cpp`. Además está el archivo `ejemploAvl.png` cuyo contenido puede copiarse en `avl.cpp`.

3. ¿Qué se pide?

Ver la Sección **Desarrollo** de [Funcionamiento y Reglamento del Laboratorio](#).

En esta tarea se deben implementar los mismos archivos que en la tarea anterior y además los nuevos.

Se debe tener en cuenta que en `usoTads.h` se agregaron nuevas funciones.

En algunas operaciones se piden requerimientos de tiempo de ejecución. Ese tiempo es el del peor caso, a menos que se especifique otra cosa de manera explícita.

Se sugiere implementar y probar los tipos y las funciones siguiendo el orden de los ejemplos que se encuentran en el directorio `test`.

4. Descripción de los módulos y algunas funciones

4.1. Módulos de tareas anteriores

Se mantienen los mismos módulos de la Tarea 3: *utils*, *info*, *cadena*, *usoTads*, *iterador*, y *binario*.

En el módulo *usoTads* se agregó la operación *interseccionDeConjuntos*.

4.2. Módulos nuevos

4.2.1. Módulo *pila*

En este módulo se debe implementar pilas (de tipo `TPila`) de elementos de tipo `nat`.

4.2.2. Módulo *colaAvls*



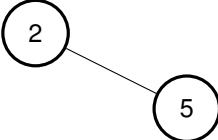
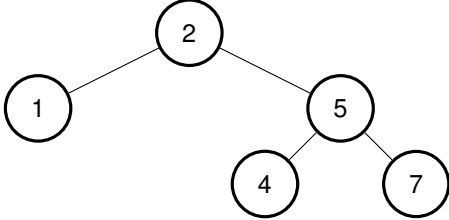
En este módulo se debe implementar colas (de tipo *TColaAvls*) de elementos de tipo árboles de tipo *TAvl*. Es posible que un árbol sea subárbol de otro, o sea, los árboles pueden compartir memoria. Para poder hacer pruebas de este módulo hace falta hacer una implementación mínima de *avl*: el *struct*, las funciones *crearAvl*, *estaVacioAvl*, *raizAvl* y *liberarAvl* y el caso en que se inserta en un árbol vacío de la función *insertarEnAvl*. Todo esto es similar a lo que ya se hizo en el módulo *binario*.

4.2.3. Módulo *avl*

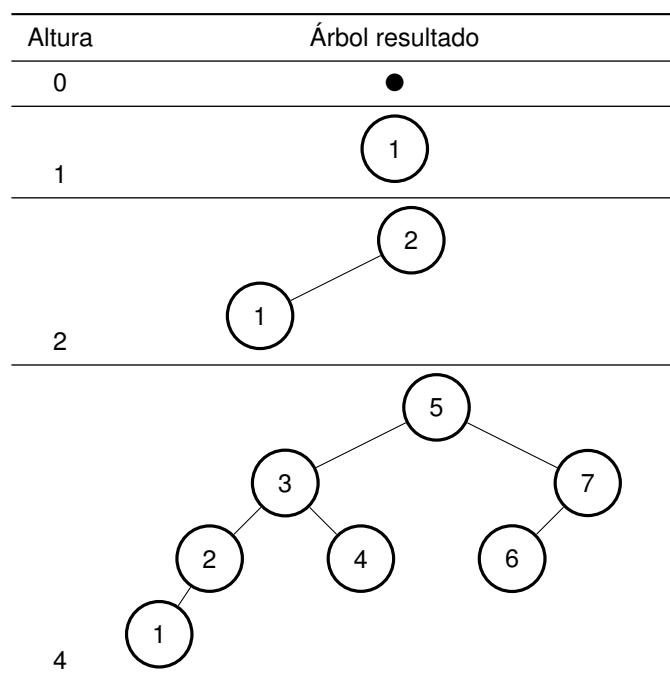
En este módulo se debe implementar árboles binario de búsqueda (de tipo *TAvl*) de elementos de tipo *nat* con la propiedad de estructura *avl*.

Se muestran ejemplos de lo que se espera de la ejecución de algunas funciones.

imprimirAvl Imprime los datos numéricos de los elementos del árbol por niveles, un nivel por línea desde el más profundo hasta la raíz.

avl	Salida esperada
	
	2
	5 2
	4 7 1 5 2

minAvl Devuelve el *avl* que tiene la mínima cantidad de nodos para la altura pasada por parámetro. Los valores deben ser consecutivos y empezar en 1. En ningún nodo puede ocurrir que el subárbol derecho tenga más nodos que el subárbol izquierdo.



4.2.4. Módulo *conjunto*

En este módulo se debe implementar conjuntos (de tipo `TConjunto`) de elementos de tipo `nat`.

4.2.5. Módulo *colaDePrioridad*

En este módulo se debe implementar colas de prioridad acotadas (de tipo `TColaDeprioridad`) de elementos de tipo `nat` y prioridad de tipo `double`.

4.2.6. Módulo *mapping*

En este módulo se debe implementar mappings (funciones parciales) (de tipo `TMapping`) de asociaciones de `nat` a `double`.

5. Entrega

Se mantienen las consideraciones reglamentarias y de procedimiento de la tarea anterior.

La tarea otorga hasta 6 puntos. La adjudicación de los puntos de las tareas de laboratorio queda condicionada a la respuesta correcta de una pregunta sobre el laboratorio en el segundo parcial.

Se debe entregar el siguiente archivo, que contiene los módulos implementados `avl.cpp`, `binario.cpp`, `cadena.cpp`, `colaAvls.cpp`, `colaDePrioridad.cpp`, `conjunto.cpp`, `iterador.cpp`, `mapping.cpp`, `pila.cpp`, `usoTads.cpp`:

■ Entrega4.tar.gz

Este archivo se obtiene al ejecutar la regla entrega del archivo *Makefile*:

```
$ make entrega
tar zcvf Entrega4.tar.gz -C src cadena.cpp usoTads.cpp binario.cpp iterador.cpp pila.cpp colaAvls.cpp
cadena.cpp
usoTads.cpp
binario.cpp
iterador.cpp
pila.cpp
colaAvls.cpp
```

```
avl.cpp  
conjunto.cpp  
colaDePrioridad.cpp  
mapping.cpp
```

Con esto se empaquetan los módulos implementados y se los comprime.

Nota: En la estructura del archivo de entrega los módulos implementados deben quedar en la raíz, NO en el directorio `src` (y por ese motivo se usa la opción `-C src` en el comando `tar`).

5.1. Plazos de entrega

El plazo para la entrega es el **miércoles 10 de junio a las 14 horas**.

5.2. Identificación de los archivos de las entregas

Cada uno de los archivos a entregar debe contener, en la primera línea del archivo, un comentario con el número de cédula del estudiante, **sin el guión y sin dígito de verificación**. Ejemplo:

```
/* 1234567 */
```

5.3. Individualidad

Ver la Sección **Individualidad** de [Funcionamiento y Reglamento del Laboratorio](#).