Obligatorio 1
Programación 3
Curso 2011
Versión 1.0

1 Objetivos

- Conocer las estructuras básicas del lenguaje C.
- Diseñar e implementar un tipo abstracto de datos de manera modular.
- Utilizar librerías y funciones de entrada/salida.
- Manejo de memoria dinámica y punteros.

2 Conocimientos previos

- Metodología de programación estructurada
- Tipos Abstractos de Datos (TAD).

3 Descripción del problema

Se desea implementar un sistema formado por un conjunto de estructuras capaces de manipular números enteros. Las estructuras que se desea implementar se describen en la sección 3.2.
Además se desea disponer de un intérprete de comandos que permita administrar dichas estructuras (tanto para agregar o quitar elementos, como para guardarlas o cargarlas de disco).

3.1 Comandos

Al iniciar el sistema se debe desplegar en pantalla:
```
Obligatorio 1 – 2011 – Programacion 3
```

El sistema deberá soportar los siguientes comandos:

- pilaAgregar *n*

Esta opción agrega el elemento *n* en la Pila.
Ejemplo:
```
>pilaAgregar 5
El elemento 5 se ha agregado en la Pila.
```

- pilaObtener

Esta opción remueve el tope de la Pila y lo despliega en pantalla.
Precondición: La Pila contiene al menos un elemento.
Ejemplo:
```
>pilaObtener
El tope de la pila es 5.
```

- colaAgregar *n*

Esta opción agrega el elemento *n* en la Cola.
Ejemplo:
```
>colaAgregar 6
El elemento 6 se ha agregado en la Cola.
```

- colaObtener

Esta opción remueve el primer elemento de la Cola y lo despliega en pantalla.
Precondición: La Cola contiene al menos un elemento.
Ejemplo:
```
>colaObtener
El primer elemento de la Cola es 6.
```

- abbAgregar *n*

Esta opción agrega el elemento *n* en el ABB.
Ejemplo:
```
>abbAgregar 8
El elemento 8 se ha agregado en el ABB.
```

- abbMinimo

Esta opción remueve el mínimo elemento del ABB y lo despliega en pantalla.
Precondición: El ABB contiene al menos un elemento.
Ejemplo:
```
>abbMinimo
El mínimo elemento del ABB es 8.
```

- abbMaximo

Esta opción remueve el máximo elemento del ABB y lo despliega en pantalla.
Precondición: El ABB contiene al menos un elemento.
Ejemplo:

```
>abbMaximo
El máximo elemento del ABB es 14.
```

- cargar *nombre*

Las estructuras Pila, Cola y ABB son cargadas desde los archivo *nombre*_pila.txt, *nombre*_cola.txt y *nombre*_abb.txt respectivamente, *nombre* es un string identificador.
Las estructuras existentes en el sistema son remplazadas por las nuevas.
Al cargar las estructuras desde los archivos, recordar que cada archivo debe ser abierto una única vez.
Precondición: Los archivos existen y tienen el formato que se describe en la sección 3.3.
Ejemplo:

```
>cargar prueba
Estructuras cargadas con exito.
```

- guardar *nombre*

Las estructuras Pila, Cola y ABB son guardadas en los archivo *nombre*_pila.txt, *nombre*_cola.txt y *nombre*_abb.txt respectivamente descartando la información previamente almacenada, *nombre* es un string identificador. En caso de que no existan dichos archivos, estos son creados.
Ejemplo:

```
> guardar prueba
Estructuras guardadas con exito.
```

- salir

Termina la ejecución del programa.
Ejemplo:

```
> salir
```

Importante:

- Los comandos mencionados anteriormente son los únicos reconocidos por el sistema, en caso de ingresar un comando distinto se debe desplegar en pantalla el mensaje: “Comando no reconocido.”

```
> comando desconocido
Comando no reconocido.
```

- En caso de que el comando ingresado sea correcto, los parámetros también lo serán.
  Ejemplo:

```
> pilaAgregar
```

Ésta **No** se considerará como entrada válida.

3.2 Estructuras

3.2.1 Cola

Se desea implementar la estructura de Cola, con las operaciones que se especifican a continuación. Estas operaciones están definidas en el archivo `Cola.h`:

```
#ifndef _COLA_H
#define _COLA_H

struct Cola;

Cola* crearCola();
//devuelve la cola vacia

bool esVaciaCola(Cola* c);
//devuelve true si c es vacia

void encolar(Cola* &c, int i);
//agrega el elemento i en c

int desencolar(Cola* &c);
//devuelve el primer elemento de c
//Precondicion: !esVaciaCola(c)

void destruirCola(Cola* &c);
// libera toda la memoria ocupada por c

void cargarColaDeArchivo(Cola* &c, char* nomArchivo);
//carga los datos guardados en el archivo nomArchivo en la cola c
//Precondicion: Existe un archivo con el nombre nomArchivo y tiene el formato
correcto.

void guardarColaAArchivo(Cola* c, char* nomArchivo);
//guarda la cola c en el archivo de nombre nomArchivo.
//Si existe el archivo nomArchivo lo sobrescribe.

#endif /* _COLA_H */
```

3.2.2 Pila

Se desea implementar la estructura de Pila, con las operaciones que se especifican a continuación. Estas operaciones están definidas en el archivo `Pila.h`:

```
#ifndef _PILA_H
#define _PILA_H

struct Pila;

Pila* crearPila();
//devuelve la pila vacia

bool esVaciaPila(Pila* p);
//devuelve true si s es vacio

void apilar(Pila* &p, int i);
//inserta el elemento i en la pila p

int desapilar(Pila* &p);
//devuelve el primer elemento de la pila p
//Precondicion: !esVaciaPila(p)

void destruirPila(Pila* &p);
// libera toda la memoria ocupada por p

void cargarPilaDeArchivo(Pila* &p, char* nomArchivo);
//carga los datos guardados en el archivo nomArchivo en la pila p
//Precondicion: Existe un archivo con el nombre nomArchivo y tiene el formato
correcto.

void guardarPilaAArchivo(Pila* p, char* nomArchivo);
//guarda la pila p en el archivo de nombre nomArchivo.
//Si existe el archivo nomArchivo lo sobrescribe.

#endif /* _PILA_H */
```

3.2.3 Árbol Binario de Búsqueda

Se desea implementar la estructura de Árbol Binario de Búsqueda, con las operaciones que se especifican a continuación. Estas operaciones están definidas en el archivo `ABB.h`:

```
#ifndef _ABB_H
#define _ABB_H

struct ABB;

ABB* crearABB();
//devuelve el arbol vacio

void agregarABB(ABB* &abb, int i);
//agrega el elemento i en abb

bool esVacioABB(ABB* abb);
//devuelve true si abb es vacio

int valorABB(ABB* abb);
//devuelve el valor de la raiz de abb
//Precondicion: !esVacioABB(abb)

ABB* arbolIzquierdo(ABB* abb);
//devuelve el subarbol izquierdo de abb
//Precondicion: !esVacioABB(abb)

ABB* arbolDerecho(ABB* abb);
//devuelve el subarbol derecho de abb
//Precondicion: !esVacioABB(abb)

int menorABB(ABB* &abb);
//devuelve el minimo valor de abb
//Precondicion: !esVacioABB(abb)

int mayorABB(ABB* &abb);
//devuelve el maximo valor de abb
//Precondicion: !esVacioABB(abb)

void destruirABB(ABB* &abb);
// libera toda la memoria ocupada por abb

void cargarABBDeArchivo(ABB* &abb, char* nomArchivo);
//carga los datos guardados en el archivo nomArchivo en el ABB abb
//Precondicion: Existe un archivo con el nombre nomArchivo y tiene el formato
correcto.

void guardarABBAArchivo(ABB* abb, char* nomArchivo);
//guarda el ABB abb en el archivo de nombre nomArchivo.
//Si existe el archivo nomArchivo lo sobrescribe.

#endif /* _ABB_H */
```

3.3 Formato de los archivos

Los archivos contendrán por cada línea un elemento de la estructura.
A continuación se especifica el orden en que los elementos deben ser almacenados para cada estructura.

- <nombre>_pila.txt

Los elementos de la pila deben ser almacenados en orden inverso a como fueron agregados en la estructura.

- <nombre>_cola.txt

Los elementos de la cola deben ser almacenados en el mismo orden en que fueron agregados en la estructura.

- <nombre>_abb.txt

En esta estructura para cada elemento se almacena el mismo, luego el subárbol con los elementos menores, y por último el subárbol con los elementos mayores, comenzando por la raíz del árbol. Para representar un árbol vacío se utiliza la palabra NULL.

En los tres casos la palabra FIN indica el final del archivo.

3.4 Ejemplo de ejecución

La salida en pantalla del programa debe ser:

```
Obligatorio 1 - 2011 - Programacion 3

>abbAgregar 5
El elemento 5 se ha agregado en el abb.
>abbAgregar 2
El elemento 2 se ha agregado en el abb.
>abbAgregar 8
El elemento 8 se ha agregado en el abb.
>pilaAgregar 5
El elemento 5 se ha agregado en la pila.
>pilaAgregar 2
El elemento 2 se ha agregado en la pila.
>pilaAgregar 8
El elemento 8 se ha agregado en la pila.
>colaAgregar 5
El elemento 5 se ha agregado en la cola.
>colaAgregar 2
El elemento 2 se ha agregado en la cola.
>colaAgregar 8
El elemento 8 se ha agregado en la cola.
>guardar ejemplo
Estructuras guardadas con exito.
>abbMaximo
El maximo elemento del abb es 8.
>pilaObtener
El tope de la pila es 8.
>colaObtener
El primer elemento de la cola es 5.
>abbMaximo
El maximo elemento del abb es 5.
>pilaObtener
El tope de la pila es 2.
>colaObtener
El primer elemento de la cola es 2.
>abbMaximo
El maximo elemento del abb es 2.
>pilaObtener
El tope de la pila es 5.
>colaObtener
El primer elemento de la cola es 8.
>salir
```

La ejecución anterior debe generar los siguientes archivos:

- ejemplo_abb.txt

```
5
2
NULL
NULL
8
NULL
NULL
FIN
```

- ejemplo_cola.txt

```
5
2
8
FIN
```

- ejemplo_pila.txt

```
8
2
5
FIN
```

4 Lenguaje a utilizar

El lenguaje a utilizar en este trabajo será `C` con las siguientes extensiones:

- Operadores `new` y `delete`.
- Pasaje de parámetros por referencia (uso de `&`).
- Declaración de tipos como en `C++` para registros y enumerados.
- Sobrecarga de funciones.
- Uso de `cin` y `cout`.
- Uso del tipo `bool` predefinido en `C++`.

5 Qué se espera

Para cada módulo de cabecera (.h) con los prototipos de las operaciones solicitadas, debe entregarse un módulo (.cpp) con la implementación de dichas operaciones. Debe respetarse estrictamente los prototipos especificados, esto es: nombre de la operación, tipo, orden y forma de pasaje de los parámetros y tipo de retorno. Los módulos de cabecera pueden bajarse de la página web del curso (http://www.fing.edu.uy/inco/cursos/prog3). Estos módulos no forman parte de la entrega, y por lo tanto, no deben ser modificados.

Los módulos deben funcionar en el ambiente MinGW instalado en facultad. Se espera que todos los módulos compilen y linkediten sin errores y sin warnings (utilizando la flag "`-Wall`" y "`-Werror`" siempre), se ejecuten sin colgarse y den los resultados correctos.

6 Forma de la entrega

Se deberá entregar únicamente 4 archivos (respetando las mayúsculas en los nombres):

- `Pila.cpp`
- `Cola.cpp`
- `ABB.cpp`
- `principal.cpp`

correspondientes a la implementación de las operaciones especificadas en cada uno de los archivos de cabecera (.h). No se podrá entregar otra cosa que no sea estos archivos.

La primera línea de cada uno de estos archivos debe contener un comentario (`/* ... */`) con la cédula del autor, sin puntos ni dígito de verificación. Por ejemplo, si la cédula es `1.234.567-8`, la primera línea de cada archivo deberá ser exactamente:

```
/* 1234567 */
```

7 Advertencia sobre el manejo de la memoria

Cuando un programa contiene errores en el manejo de la memoria, su comportamiento puede ser inestable. Esto implica que algunas veces funciona correctamente y otras no. En ciertos casos esto puede inducir a creer (erróneamente) que ciertos programas, que en realidad son incorrectos, funcionan correctamente. Este aspecto es influenciado, entre otras cosas, por el *sistema operativo* en el que se ejecuten los programas. Recomendamos tener sumo cuidado con este punto y testear los módulos en sistemas operativos Windows NT, Windows 2000 o Windows XP. CON LA VERSION DE MINGW INSTALADA EN LAS SALAS DE INFORMÁTICA DE LA FACULTAD.

8 Sobre la individualidad del trabajo

El laboratorio es INDIVIDUAL. Los estudiantes pueden estudiar en grupo y consultarse mutuamente, pero NO pueden realizar en grupo las tareas de codificación, escritura, compilación y depuración del programa.

Los trabajos de laboratorio que a juicio de los docentes no sean individuales serán eliminados, con la consiguiente pérdida del curso, para todos los involucrados. Además todos los casos serán enviados a los órganos competentes de la Facultad, lo cual puede acarrear sanciones de otro carácter y gravedad para los estudiantes involucrados.

No existirán instancias en el curso para reclamos frente a la detección por parte de los docentes de trabajos de laboratorio no individuales, independientemente de las causas que pudiesen originar la no individualidad. A modo de ejemplo y sin ser exhaustivos: utilizar código realizado en cursos anteriores u otros cursos, perder el código, olvidarse del código en lugares accesibles a otros estudiantes, prestar el código o dejar que el mismo sea copiado por otros estudiantes, dejar la terminal con el usuario abierto al retirarse, enviarse código por mail, etc. Asimismo, se prohíbe el envío de código al grupo de noticias del curso, dado que el mismo será considerado como una forma de compartir código y será sancionada de la manera más severa posible.

Es decir que se considera a cada estudiante RESPONSABLE DE SU TRABAJO DE LABORATORIO Y DE QUE EL MISMO SEA INDIVIDUAL.

9 Fecha de entrega

El trabajo debe entregarse el día **lunes 29 de agosto de 2011** antes de las **22:00** horas.

La entrega se realizará mediante un formulario que se habilitará oportunamente en la página web del curso.