UNTREF

Algoritmos y programación 3

Trabajo práctico Final: Colecciones

1do cuatrimestre, 2017

(Trabajo Grupal)

(**Versión: 1.1**)

Alumnos:

Nro	Nombre	Legajo	Mail
1			
2			
3			

Fecha de entrega final: 24/06/2017

Nota Final:

Introducción

Objetivo del trabajo

Aplicar los conceptos enseñados en la materia a la resolución de un problema así como también la aplicación de buenas prácticas, trabajando en forma grupal y utilizando un lenguaje de tipado estático de bajo nivel (Lenguaje C)

Consigna general

Implementar un conjunto de primitivas para operar listas de valores (vectores) evitando duplicar o replicar código. Las primitivas a implementar serán:

1. Agregar elemento a la collección (`collection_add`).
2. Quitar elemento de la collección (`collection_remove`).
3. Inicializar la colección con un conjunto de valores (`collection_init`).
4. Iterar la colección (`collection_iterate`).
5. Encontrar un valor dentro de la colección (`void* collection_find( Collection* this, void* value,...)`).
6. Seleccionar un conjunto de elementos de la colección (`collection_select( Collection* this, Collection* dst, ...)`). Referencia: Array.filter.
7. Colectar elementos de una colección (`void collection_collect( Collection* this, Collection* dst, ...)`). Referencia: Array.map.
8. Filtar elementos de una colección (`void collection_filter( Collection* this, ...)`). Referencia: Igual que select pero se aplica en la colección c.
9. Reducir a derecha una colección (**definir**). Referencia: Array.reduceRight.
10. Reducir a izquierda una colección (**definir**). Referencia: Array.reduce.
11. Unir dos colecciones.
12. Intersección de colecciones.

Descripción de la aplicación a desarrollar

Consignas para el programa principal:

Cada una de las primitivas que se implementen deberá ser utilizada en forma aislada dentro de un bloque de código para poder mostrar su comportamiento. Un bloque de código se define utilizando {}.

Por ejemplo:

```
int main(int argc, char** argv) {
    { // Un bloque
    Collection c;
    collection_init(&c, bunch, sizeof(bunch), sizeof(int));
    collection_release(&c);
    }

    { // otro bloque
    Collection c;
    collection_init_empty(&c, sizeof(int));
```

```
    collection_release(&c);
    }

    return 0;
}
```

Pautas de calidad al momento de resolver el TP

- Los atributos del TDA deberán accederse mediante funciones provistas por la API **exclusivamente** (Application Programming Interface).
- Dentro de las funciones que operan sobre el TDA/s evitar los llamados a funciones que impriman en las salidas estándar de errores o en la salida estándar (printf, fprintf, etc.)
- Cada uno de los TDAs definidos deberán estar apropiadamente documentas, tanto sus campos como todas sus primitivas.
- Elegir un standard de codificación y seguirlo a lo largo del proyecto.
- Toda la memoria reservada dentro del TDA será responsabilidad de la API, es decir, si el TDA reserva memoria deberá proveer las herramientas necesarias para liberarla apropiadamente.
- Si necesita utilizar números mágicos, utilice #define y documéntelos apropiadamente.
- Los arrays utilizados dentro del TP deberán poder ser de una longitud indefinida (sin restricciones en su cantidad de elementos).
- La estructura del proyecto deberá ser:
    - Archivo main.c con el ejemplo de uso de el/los TDA/s
    - Archivo <declaración del tda>.h
    - Archivo <definición del tda>.c
- El proyecto deberá ser implementado utilizando la interfase Eclipse.

Formas de entrega

Habrá 4 entregas formales. Las mismas tendrán una calificación de APROBADO o NO APROBADO en el momento de la entrega.

1er Entrega: *Mínimamente* (se aconseja avanzar todo lo posible):

- Estructura completa del proyecto.
- Creación del main.c e implementación de las siguientes primitivas:
  - Agregar elemento a la collección (`collection_add`).
  - Quitar elemento de la collección (`collection_remove`).
  - Inicializar la colección con un conjunto de valores (`collection_init`).
  - Iterar la colección (`collection_iterate`).

2da Entrega: Tener la posibilidad de:

1. Encontrar un valor dentro de la colección (`void* collection_find( Collection* c, void* value,...)`).
2. Seleccionar un conjunto de elementos de la colección (`collection_select( Collection* c, Collection* dst, ...)`).
3. Colectar elementos de una colección (`void collection_collect( Collection* c, Collection* dst, ...)`)

3er Entrega: Tener la posibilidad de:

1. Colectar elementos de una colección (`void collection_collect( Collection* c, Collection* dst, ...)`).
2. Reducir a derecha una colección (**definir**).
3. Reducir a izquierda una colección (**definir**).

4ta y última Entrega: Trabajo Práctico completo, funcionando y cumpliendo todas las normas de calidad.

IMPORTANTE:
Si el TP no se puede ejecutar por cualquier tipo de error (acceso inválido de memoria, goteo de memoria, etc) el TP estará **desaprobado**.

Fechas de entrega programadas

1er Entrega: 06/05/2017
2da Entrega: 20/05/2017
3er Entrega: 03/06/2017
4ta y última Entrega: Ver portada del tp.

Informe

Supuestos

[Documentar todos los supuestos hechos sobre el enunciado. Asegurarse de validar con los docentes]

Modelo de dominio

[Explicar los elementos más relevantes del diseño. Es decir: qué TDAs se han creado, qué responsabilidades tienen asignadas, cómo se relacionan, etc]

Detalles de implementación

*[Deben **detallar/explicar** qué estrategias utilizaron para resolver los puntos más conflictivos del trabajo práctico.]*

Checklist de corrección

Esta sección es para uso exclusivo de los docentes, por favor no modificar.

Carpeta

Generalidades

- ¿Son correctos los supuestos y extensiones?
- ¿Es prolija la presentación? (hojas del mismo tamaño, numeradas y con tipografía uniforme)

Modelo

- ¿Está completo? ¿Contempla la totalidad del problema?
- ¿Respeta encapsulamiento?
- ¿Pierde memoria?
- ¿Cumple con las buenas prácticas?

Código

Generalidades

- ¿Respeta estándares de codificación?
- ¿Está correctamente documentado?
- ¿Define magic numbers? ¿Están documentados?
- ¿Respeta estructura del proyecto?