Universidad Simón Bolívar
Depto. de Computación y T.I.
Sistemas de Operación (CI-3815)
Enero-Marzo 2020

# Proyecto # 1 (12%)

## Objetivo

Familiarizarse con la representación de datos compuestos y programación estructurada en el ambiente de programación SPIM y con el lenguaje ensamblador MIPS.

## Actividades

### Actividad 0: TAD_Elemento

Esta primera actividad supone la definición del tipo generico Elemento. Este tipo tiene definida las siguientes operaciones:

- HashCode: Elemento → Entero

  Genera un valor entero, no necesariamente diferente, para cada objeto.

- Compara: Elemento x Elemento → {-1, 0, 1}

  Compara dos elementos de la siguiente forma:

  - compara(a, b) = -1, si a < b
  - compara(a, b) = 0, si a == b
  - compara(a, b) = 1, si a > b

- Imprimir: Elemento → void

  Imprime el elemento.

## Actividad 1: Tad_Lista

Corresponde con una lista genérica, es decir, debe funcionar para cualquier tipo de elementos. Ver Actividad 0.

1. **list_crear(IN address funCompara; IN funImprime: entero ; OUT address: entero; OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras de la representación del TAD se encuentran inicializadas. |
| Post: | Las estructuras se actualizan reflejando que se creó una lista en la dirección address. Se inicializaron las funciones de comparación e impresión a funCompara y funImprime respectivamente. |
| Parámetros: | |
| funCompara: | Dirección de la función de comparación de los elementos. |
| FunImprime: | Dirección de la función de impresión de los elementos |
| Retorno: | |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |
| address: | La dirección de la lista |

2. **list_insertar(IN address: entero; IN element: entero; OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | El elemento element fue insertado respetando el orden (creciente) dado por la función de comparación |
| Parámetros: | |
| address: | Dirección de la Lista |
| element | El elemento a insertar |
| Retorno: | |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

3. **list_longitud(IN address: entero; OUT len: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | La estrutura no sufre cambios |
| Parámetros: | |
| address: | Dirección de la Lista |
| Retorno: | |

| | |
|---|---|
| len: | Número de elementos en la lista. |

4. **list_obtener(IN address: entero; IN posicion: entero; OUT ele: entero; OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas y 1<= Posicion <= Longitud |
| Post: | La estrutura no sufre cambios |
| Parámetros: | |
| address: | Dirección de la Lista |
| posicion | Posición del elemento a retornar |
| Retorno: | |
| ele: | La dirección del elemento en la posición indicada. |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

5. **list_imprimir(IN address: entero;  OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | La estrutura no sufre cambios |
| Parámetros: | |
| address: | Dirección de la Lista |
| Retorno: | |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

6. **list_destruir(IN address: entero  OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | La estrutura y todos sus datos son destruidos (liberados) |
| Parámetros: | |
| address: | Dirección de la Lista |
| Retorno: | |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

## Actividad 2: TAD_TablaHash

Corresponde con una tabla de hash con manejo de colisiones. Asuma que no puede haber dos elementos con la misma clave.

1. **tab_crear(IN numClas: entero; IN funHash: entero, IN funCompara: entero;  OUT address: entero; OUT code: entero)**

| | |
|---|---|
| Pre: | NumClass > 1 y funCompara es una función válida |
| Post: | Las estructuras se actualizan reflejando la creación de una tabla de hash con numClas particiones. Se inicializan la función funComprar. |
| Parámetros: | |
| numClas | Número de particiones/clases |
| funCompara: | Función de comparación de los elementos a almacenar |
| FunHash: | Función que calcula el hash de un elemento |
| Retorno: | |
| address | Dirección donde se creó la tabla de hash |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

2. **tab_insertar(IN address: entero; IN clave: entero; IN ele: entero; OUT inserto: entero; OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | El elemento ele fue insertado con la clave *clave*. |
| Parámetros: | |
| address | Dirección donde se creó la tabla de hash |
| clave | Clave del elemento |
| Ele: | elemento |
| Retorno: | |
| inserto | 1 si el elemento fue insertado, 0 en caso contrario. |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

3. **tab_buscar(IN address: entero; IN clave: entero; OUT ele: entero; OUT code: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | La estructura no sufre cambios |
| Parámetros: | |

| | |
|---|---|
| address | Dirección dónde se creó la tabla de hash |
| clave | Clave del elemento |
| Retorno: | |
| ele | Elemento correspondiete a clave |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

4. **tab_rehash(IN address: entero; IN numClas: entero; OUT code: entero);**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | La estructura se reorganiza de manera que los elementos sean reubicados en función del nuevo número de clases. |
| Parámetros: | |
| numClas | Nuevo número de clases |
| Retorno: | |
| code: | Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente. |

5. **tab_destruir(IN address: entero)**

| | |
|---|---|
| Pre: | Las estructuras se encuentran inicializadas. |
| Post: | Se libera toda la memoria para representar esta instancia de tabla de hash. |
| Parámetros: | |
| address: | Dirección de la tabla de hash |

Operaciones adicionales

**perror(IN code:entero; OUT void)**

| | |
|---|---|
| Pre: | true |
| Post: | true |
| Parámetros: | |
| Code | valor negativo que identifica un error ocurrido. Debe imprimir el mensaje asociado al valor *code. Si* el valor es positivo esta función no debe imprimir nada. |
| Retorno: | void |

**Nota:**

Uds. noo deben ofrecer un archivo principal ni un TAD elemento, las pruebas se realizarán con un programa principal y elemento desarrollado por su profesor. Por esto es importante que no realice

cambios a la especificación.

### Recomendaciones

1. Comience con tiempo.
2. Estructure bien su código.
3. Trabaje en forma ordenada e incremental
4. Pruebe que cada una de sus funciones funciona correctamente
5. **Respete** la especificación dada en el presente enunciado: No cambie el nombre de las rutinas NI las definiciones aquí establecidas.
6. Tenga presente que es mejor tener más funciones pequeñas que menos funciones largas.

## Entrega

El proyecto debe ser entregado hasta las 9:00 am del 21 de febrero. Debe entregar:

1. Un informe **impreso** (de no más de 6 páginas) explicando el diseño de su implementación. Las estructuras de datos que UD haya definido, explique sus ventajas y desventajas con respecto a las otras estructuras consideradas.
2. Entregar el código **impreso** y **apropiadamente** documentado de su implementación de las actividades definidas.
3. Enviar la versión electrónica al correo de su profesor con asunto “Proyecto 1 – Carnet1-Carnet2.

4. Note que las versiones impresa y digital deben corresponder.