Universidad Simón Bolívar

Informe de Proyecto I

Estructuras de Datos en MIPS

Autor:
Christopher Gómez (18-10892)
Ka Fung (18-10492)

Profesor:
Eduardo Blanco

Organización del Computador (CI3815)

25 de noviembre de 2021

1. Pseudocódigo

Se presentan a continuación los pseudocódigos del programa que se desea implementar.

Primeramente, se necesita extraer los datos necesarios de cada archivo de entrada, para ello, se usan tablas de hash y listas.

EXTRAER-DATOS

```
1  archivoEst = Leer archivo de estudiantes
2  tablaHashEst = new TABLAHASH(101)
3  for linea in archivoEst:
4        carnet = Guardar carnet
5        nombre = Guardar nombre
6        indice = Guardar índice
7        creditosAprob = Guardar número de créditos aprobados
8        est = new ESTUDIANTE(carnet, nombre, indice, creditosAprob)
9        tablaHashEst.INSERTAR(carnet, est)
10
11 archivoMat = Leer archivo de materias
12 tablaHashMat = new TABLAHASH(101)
13 listaMat = new LISTA()
14 for linea in archivoMat:
15       codigo = Guardar codigo
16       nombre = Guardar nombre
17       creditos = Guardar creditos
18       numCupos = Guardar número de cupos
19       minCreditos = Guardar mínimo de créditos
20       mat = new MATERIA(codigo, nombre, creditos, numCupos, minCreditos)
21       tablaHashMat.INSERTAR(carnet, est)
22       listaMat.INSERTAR-ORDENADO(codigo, f)
23
24 listaSol = new LISTA()
25 archivoSol = Leer archivo de solicitudes
26 for linea in archivoSol:
27       carnet = Guardar carnet del estudiante
28       est = tablaHashEst.OBTENER-VALOR(carnet)
29       codigo = Guardar codigo de la materia
30       mat = tablaHashMat.OBTENER-VALOR(codigo)
31       sol = new SOLICITUD(est, mat, 'S')
32       listaSol.INSERTAR(sol)
```

Al terminar este pseudocódigo, se debe tener una lista de solicitudes, una lista de códigos de materias en orden lexicográfico, una tabla de estudiantes, y una tabla de materias, la idea ahora es procesar la lista de solicitudes para que cada materia tenga una lista de estudiantes inscritos.

PROCESAR-SOLICITUDES

```
1  for sol in listaSol:
2        est = sol.estudiante
3        mat = sol.materia
4        mat.AGREGAR-ESTUDIANTE(est)
```

Ahora, cada materia contiene una lista con los estudiantes inscritos. Se asume que la estructura se encarga de mantener actualizado el número de cupos y de agregar en orden a los estudiantes en su lista de estudiantes. Así, para finalizar esta primera etapa solamente resta escribir en el archivo de salida cada materia con sus estudiantes inscritos.

```
GENERAR-ARCHIVO-TENTATIVO
1  archivoTen = Abrir archivo tentativo a escribir
2  for mat in listaMat:
3      archivoTen.ESCRIBIR('<mat.codigo> ')
4      archivoTen.ESCRIBIR('"<mat.nombre>" ')
5      archivoTen.ESCRIBIR('<mat.numCupos> \n')
6      for est in mat.estudiantes:
7          archivoTen.ESCRIBIR(' <est.carnet> ')
8          archivoTen.ESCRIBIR('<est.nombre> \n')
```

Luego, se procesan las solicitudes de corrección. Al terminar el siguiente pseudocódigo, se debe tener una lista de solicitudes de corrección. Se procesa la lista de solicitudes para que cada materia tenga una lista de estudiantes inscritos y eliminados. Se ordena la lista de estudiantes de cada materia según la prioridad: M1.5. Se le da prioridad a los estudiantes con menor número de créditos aprobados.

```
EXTRAER-DATOS-CORRECCION
 1  listaSolCor = new LISTA()
 2  archivoCor = Leer archivo de solicitudes de corrección
 3  for linea in archivoCor:
 4      carnet = Guardar carnet del estudiante
 5      est = tablaHashEst.OBTENER-VALOR(carnet)
 6      codigo = Guardar codigo de la materia
 7      mat = tablaHashMat.OBTENER-VALOR(codigo)
 8      op = Guardar operación de la solicitud
 9      sol = new SOLICITUD(est, mat, op)
10      listaCor.INSERTAR(sol)
```

```
PROCESAR-SOLICITUDES-CORRECCION
 1  listaPrioridad = new LISTA()
 2  for sol in listaCor:
 3      est = sol.estudiante
 4      mat = sol.materia
 5      if sol.op == 'E':
 6          mat.ELIMINAR-ESTUDIANTE(est)
 7      else :
 8          listaPrioridad.INSERTAR-ORDENADO(sol, g)
 9
10  for sol in listaPrioridad:
11      mat = sol.materia
12      est = sol.estudiante
13      if mat.cupos > 0:
14          mat.AGREGAR-ESTUDIANTE(est)
```

Finalmente, se escribe en el archivo de salida cada materia con sus estudiantes inscritos y eliminados.

```
GENERAR-ARCHIVO-DEFINITIVO
1  archivoDef = Abrir archivo definitivo a escribir
2  for mat in listaMat:
3        archivoDef.ESCRIBIR('<mat.codigo> ')
4        archivoDef.ESCRIBIR('"<mat.nombre>" ')
5        archivoDef.ESCRIBIR('<mat.numCupos> \n')
6        for est in mat.estudiantes:
7              archivoTen.ESCRIBIR('<est.carnet> ')
8              archivoTen.ESCRIBIR('<est.nombre> ')
9              archivoTen.ESCRIBIR('<est.op> \n')
```

2. Estructuras utilizadas

En la seccion anterior se menciona el uso de distintas estructuras de datos utilizadas en el diseño del programa. En esta sección se describe cada una de ellas, junto con sus atributos y operaciones.

- PAR:
  - Atributos:
    - Primer elemento.
    - Segundo elemento.
  - Operaciones:
    - CREAR(*primero*, *segundo*)
- LISTA:
  - Atributos:
    - Cabeza.
    - Tamaño.
  - Operaciones:
    - CREAR()
    - INSERTAR(*elemento*)
    - INSERTAR-ORDENADO(*elemento*, f)
- TABLAHASH:
  - Atributos:
    - Tamaño.
    - Tabla.
  - Operaciones:
    - CREAR(*tam*)
    - INSERTAR(*clave*, *valor*)
    - OBTENER-VALOR(*clave*)

- Estudiante:
  - Atributos:
    - Carnet.
    - Nombre.
    - Índice.
    - Créditos aprobados.
  - Operaciones:
    - Crear(*carnet*, *nombre*, *indice*, *creditosAprob*)
- Materia:
  - Atributos:
    - Código.
    - Nombre.
    - Número de créditos.
    - Cupos.
    - Número mínimo de créditos aprobados.
    - Lista Estudiantes.
  - Operaciones:
    - Crear(*codigo*, *nombre*, *creditos*, *numCupos*, *minCreditos*)
    - Aumentar-Cupo()
    - Disminuir-Cupo()
    - Agregar-Estudiante(*Estudiante*)
    - Eliminar-Estudiante(*Estudiante*)
- Solicitud:
  - Atributos:
    - Estudiante.
    - Materia.
    - Operación.
  - Operaciones:
    - Crear(*est*, *mat*, *op*)

3. Consideraciones

Durante la implementación de las estructuras de datos, se decidió crear una tabla de hash para almacenar las materias y los estudiantes, ya que se realizaba numerosas búsquedas durante el programa. Ejemplo de ello se presenta al procesar cada solicitud de inscripción en Extraer-Datos y Extraer-Datos-Correccion, en la cual se tiene que buscar por cada solicitud, los datos del estudiante y la lista de estudiantes de cada Materia para insertar la inscripción o eliminación. Así, para evitar búsquedas lineales con estructuras como listas, con las tablas de hash implementadas se logra

obtener la información de cada materia/estudiante en tiempo amortizado constante. En este sentido, la función de Hash escogida para TablaHash fue basada en la función de Hash para Strings que utiliza el lenguaje de programación Java [1].

Por otro lado, se implementó distintas funciones de comparación. Para imprimir cada materia con sus respectivos estudiantes inscritos y eliminados, se toma en cuenta el orden ascendente según el código de la materia y el carnet del estudiante. Para ello, se realiza comparaciones por cada valor ASCII de un caracter del carnet al insertar el estudiante en la lista de estudiantes de la materia. En cambio, para tomar en cuenta la modalidad asignada en las solicitudes de corrección, se realiza una comparación entre el número de créditos aprobados de cada estudiante a inscribir.

Además, para

atoi itoa

[1] https://cseweb.ucsd.edu/ kube/cls/100/Lectures/lec16/lec16-15.html