Universidad Simón Bolívar

Informe de Proyecto I

Estructuras de Datos en MIPS

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

Profesor:
Guillermo Palma

Algoritmos y Estructuras III (CI2693)

11 de enero de 2022

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

```
33 for sol in listaSol:
34       est = sol.estudiante
35       mat = sol.materia
36       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; nótese que para estas inclusiones no se toma en cuenta el número de créditos aprobados del estudiante ni el mínimo de créditos requeridos por la materia, ya que la modalidad indica que en la primera etapa se aceptar **todas** las solicitudes.

GENERAR-ARCHIVO-TENTATIVO

```
37  archivoTen = Abrir archivo tentativo a escribir
38  for mat in listaMat:
39      archivoTen.ESCRIBIR('<mat.codigo> ')
40      archivoTen.ESCRIBIR('"<mat.nombre>" ')
41      archivoTen.ESCRIBIR('<mat.numCupos> \n')
42      for est in mat.estudiantes:
43          archivoTen.ESCRIBIR(' <est.carnet> ')
44          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. Para procesar dicha lista, se comienza aceptando todas las eliminaciones y paralelamente creando una lista de solicitudes de inclusión que estará ordenada de acuerdo a la prioridad dada en la modalidad M1.5: se le da más prioridad a los estudiantes con menor número de créditos aprobados.

Una vez aceptadas todas las eliminaciones, se procede a liberar cupos de las materias (líneas 64-70), para ello, primero se eliminan de cada materia a los estudiantes que no cuenten con los requisitos necesarios para cursarlas (número mínimo de créditos aprobados, en este caso), y luego se elimina de las materias que exceden el número de cupos a los estudiantes con menor prioridad. Por último, se aceptan inclusiones siguiendo la lista de prioridad siempre y cuando no se exceda el número de cupos.

EXTRAER-DATOS-CORRECCION

```
45  listaSolCor = new LISTA()
46  archivoCor = Leer archivo de solicitudes de corrección
47  for linea in archivoCor:
48      carnet = Guardar carnet del estudiante
49      est = tablaHashEst.OBTENER-VALOR(carnet)
50      codigo = Guardar codigo de la materia
51      mat = tablaHashMat.OBTENER-VALOR(codigo)
52      op = Guardar operación de la solicitud
53      sol = new SOLICITUD(est, mat, op)
54      listaCor.INSERTAR(sol)
```

PROCESAR-SOLICITUDES-CORRECCION

```
55  listaPrioridad = new LISTA()
56  for sol in listaCor:
57      est = sol.estudiante
58      mat = sol.materia
59      if sol.op == 'E':
60          mat.ELIMINAR-ESTUDIANTE(est)
```

```
61        else :
62              listaPrioridad.Insertar-Ordenado(sol, g)
63
64  for mat in listaMat:
65        for est in mat.estudiantes:
66              if est.creditosAprob < mat.minCreditos:
67                    mat.Eliminar-Estudiante(est)
68  for mat in listaMat:
69        while mat.numCupos < 0:
70              // Eliminar al estudiante con más créditos aprobados.
71
72  for sol in listaPrioridad:
73        mat = sol.materia
74        est = sol.estudiante
75        if mat.cupos > 0:
76              if est.creditosAprob ≥ mat.minCreditos:
77                    mat.Agregar-Estudiante(est)
```

Finalmente, se escribe en el archivo de salida cada materia con sus estudiantes inscritos y eliminados. Si el estudiante fue inscrito en el proceso de inscripción, no se denota la operación. En cambio, si estaba inscrito y fue eliminado, se denota la operación de eliminación, o si se inscribió en corrección, se denota la operación de inscripción. No estarán en en archivo definitivo los estudiantes cuya inclusión se haya negado en la corrección, bien sea por falta de créditos o porque no habían cupos disponibles.

Generar-Archivo-Definitivo

```
78  archivoDef = Abrir archivo definitivo a escribir
79  for mat in listaMat:
80        archivoDef.Escribir('<mat.codigo> ')
81        archivoDef.Escribir("'<mat.nombre>" ')
82        archivoDef.Escribir('<mat.numCupos> \n')
83        for est in mat.estudiantes:
84              archivoTen.Escribir('<est.carnet> ')
85              archivoTen.Escribir('<est.nombre> ')
86              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 enlistan cada una de ellas, junto con sus atributos y operaciones. En algunos casos, se implementaron más operaciones de las aquí mencionadas, que se dejaron en el código fuente.

- 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], la cual es sencilla de calcular a la vez que da buenos resultados en la práctica.

Por otro lado, se implementaron distintas funciones de comparación (a las que refiere el pseudocódigo en la sección 1 como f y g en las líneas 22 y 62). 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 realizan 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.

Para la implementación y almacenamiento en MIPS de algunos atributos de las estructuras de datos (como Estudiante y Materia), se realizaron funciones itoa y atoi, las cuales convierten un entero a una cadena de caracteres y viceversa, lo cual facilita las comparaciones numéricas cuando es necesario, y la impresión del archivo de texto al final de cada etapa.

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