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)**

24 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, op)
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 supone 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
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 procesa 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
 2  listaSolCor = new Lista()
 3  archivoCor = Leer archivo de solicitudes de corrección
 4  for linea in archivoCor:
 5      carnet = Guardar carnet del estudiante
 6      est = tablaHashEst.Obtener-Valor(carnet)
 7      codigo = Guardar codigo de la materia
 8      mat = tablaHashMat.Obtener-Valor(codigo)
 9      op = Guardar operación de la solicitud
10      sol = new Solicitud(est, mat, op)
11      listaCor.Insertar(sol)
```

```
Procesar-Solicitudes-Correccion
33  for sol in listaCor:
34      est = sol.estudiante
35      mat = sol.materia
36      if sol.op == 'E':
37          mat.Eliminar-Estudiante(est)
38          mat.cupos++
39      else :
40          mat.Agregar-Estudiante(est)
41          mat.cupos- -
```

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

```
GENERAR-ARCHIVO-DEFINITIVO
37  archivoDef = Abrir archivo definitivo a escribir
38  for mat in listaMat:
39      archivoDef.ESCRIBIR('<mat.codigo> ')
40      archivoDef.ESCRIBIR('"<mat.nombre>" ')
41      archivoDef.ESCRIBIR('<mat.numCupos> \n')
42      for est in mat.estudiantes:
43          archivoTen.ESCRIBIR('<est.carnet> ')
44          archivoTen.ESCRIBIR('<est.nombre> ')
45          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.
  - 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*)