



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

```
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.

#### 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 eliminar de cada materia que excede 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      while mat.numCupos < 0:
66          // Eliminar al estudiante con más créditos aprobados.
67
68  for sol in listaPrioridad:
69      mat = sol.materia
70      est = sol.estudiante
71      if mat.cupos > 0:
72          mat.AGREGAR-ESTUDIANTE(est)
```

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

## GENERAR-ARCHIVO-DEFINITIVO

```
73  archivoDef = Abrir archivo definitivo a escribir
74  for mat in listaMat:
75      archivoDef.ESCRIBIR('<mat.codigo> ')
76      archivoDef.ESCRIBIR("<mat.nombre> ")
77      archivoDef.ESCRIBIR('<mat.numCupos> \n')
78      for est in mat.estudiantes:
79          archivoTen.ESCRIBIR('<est.carnet> ')
80          archivoTen.ESCRIBIR('<est.nombre> ')
81          archivoTen.ESCRIBIR('<est.op> \n')
```

## 2. Estructuras utilizadas

En la sección 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:
  - $\text{CREAR}(\text{primero}, \text{segundo})$
- LISTA:
  - Atributos:
    - Cabeza.
    - Tamaño.
  - Operaciones:
    - $\text{CREAR}()$
    - $\text{INSERTAR}(\text{elemento})$
    - $\text{INSERTAR-ORDENADO}(\text{elemento}, f)$
- TABLAHASH:
  - Atributos:
    - Tamaño.
    - Tabla.
  - Operaciones:
    - $\text{CREAR}(\text{tam})$
    - $\text{INSERTAR}(\text{clave}, \text{valor})$
    - $\text{OBTENER-VALOR}(\text{clave})$
- ESTUDIANTE:
  - Atributos:
    - Carnet.
    - Nombre.
    - Índice.
    - Créditos aprobados.
  - Operaciones:
    - $\text{CREAR}(\text{carnet}, \text{nombre}, \text{indice}, \text{creditosAprob})$
- 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<sup>1</sup>, 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.

---

<sup>1</sup><https://cseweb.ucsd.edu/~kube/cls/100/Lectures/lec16/lec16-15.html>