Taller de estructura de datos.

Estructuras de datos dinámicas (Lista enlazadas).

Andrés Felipe Galván.

Javier Enrique Guerra.

Santiago Andrés David.

Fernando Javier Mercado.

Estructura de datos - Grupo 01

Docente: Braulio Barrios.

Universidad Popular Del Cesar

Facultad de Ingenierías y Tecnologías

Valledupar, Cesar

2021 - I

Punto 1.

```
DEFINIR Nodo1
   entero dato

   Nodo1 *siguiente
   Nodo1 *anterior
FIN Nodo1


REGISTRO ListaEnlazada1
   Nodo1 *cabeza
   Nodo1 *ultimo
FIN REGISTRO


DEFINIR Nodo2
   entero dato

   Nodo2 *siguiente
FIN Nodo


REGISTRO ListaEnlazada2
   Nodo2 *cabeza
   Nodo2 *ultimo
FIN REGISTRO


FUNCION pedirDatos1(Nodo1 nodo)
   ESCRIBIR "Introduzca un numero: "
   LEER(nodo^.dato)
FIN pedirDatos


FUNCION insertarDescendentemente(ListaEnlazada1 lista1, Nodo1 *nodo)
   SI (nodo^.dato > lista.cabeza^.dato) ENTONCES
       nodo^.siguiente <- lista.cabeza
       lista.cabeza^.antes <- nodo
```

```
        lista.cabeza <- nodo
    SINO
        SI (nodo^.dato < lista.ultimo^.dato) ENTONCES
            lista.ultimo^.siguiente <- nodo
            nodo^.anterior <- lista.ultimo
            lista.ultimo <- nodo
        SINO
            Nodo *cursor <- lista.cabeza

            MIENTRAS (cursor^.siguiente != NULL) AND (nodo^.dato <= cursor^.dato) HACER
                cursor <- cursor^.siguiente
            FIN_MIENTRAS

            nodo^.anterior <- cursor^.anterior
            nodo^.anterior^.siguiente <- nodo
            nodo^.siguiente <- cursor
            cursor^.anterior <- nodo
        FIN_SI
    FIN_SI
FIN insertarDescendentemente


FUNCION contiene(ListaEnlazada1 lista, entero dato)
    Nodo1 *cursor <- lista.cabeza
    repetido <- false

    MIENTRAS (cursor^.siguiente <> NULL) AND (repetido = false) ENTONCES
        SI (dato = cursor^.dato) ENTONCES
            repetido <- true
        FIN_SI
    FIN_MIENTRAS

    RETORNAR repetido
FIN contiene


FUNCION agregarUnico(ListaEnlazada1 lista1, Nodo1 *nodo)
    SI (contiene(lista1, nodo^.dato) = false) ENTONCES
        insertarDescendentemente(lista1, nodo)
    FIN_SI
```

```
FIN agregarUnico


FUNCION llenarDatos1(ListaEnlazada1 lista)
   ESCRIBIR "LLENADO DE DATOS"
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo1 *nodo
       NEW(nodo)
       SI (nodo = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatos1(nodo)
           agregarUnico(ListaEnlazada1, nodo)
       FIN_SI
       ESCRIBIR "Desea agregar otro nodo a la lista? [S/N]: "
       LEER(continuar)
   FIN_MIENTRAS
FIN llenarDatos


FUNCION pedirDatos2(Nodo2 nodo)
   ESCRIBIR "Introduzca un numero: "
   LEER(nodo^.dato)
FIN pedirDatos


FUNCION agregarNodo2(ListaEnlazada2 lista, Nodo *nodo)
   nodo^.siguiente <- NULL

   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- nodo
   SINO
       lista.ultimo^.siguiente <- nodo
   FIN_SI
   lista.ultimo <- nodo
FIN agregarNodo2
```

```
FUNCION llenarDatos2(ListaEnlazada1 lista)
   ESCRIBIR "LLENADO DE DATOS"
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo2 *nodo
       NEW(nodo)
       SI (nodo = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatos2(nodo)
           agregarNodo2(ListaEnlazada1, nodo)
       FIN_SI

       ESCRIBIR "Desea agregar otro nodo a la lista? [S/N]: "
       LEER(continuar)
   FIN_MIENTRAS
FIN llenarDatos


FUNCION borrarCabeza(ListaEnlazada2 lista)
   Nodo2 *nodoBorrar <- lista.cabeza
   lista.cabeza <- lista.cabeza^.siguiente
   lista.cabeza^.anterior <- NULL
   DELETE(nodoBorrar)
FIN borrarCabeza


FUNCION borrarUltimo(ListaEnlazada2 lista)
   Nodo2 *nodoBorrar <- lista.ultimo
   lista.ultimo <- lista.ultimo^.anterior
   lista.ultimo^.siguiente <- NULL
   DELETE(nodoBorrar)
FIN borrarUltimo



FUNCION borrarNodo(ListaEnlazada2 lista, Nodo2 *nodoBorrar)
```

```
   nodoBorrar^.anterior^.siguiente <- nodoBorrar^.siguiente
   nodoBorrar^.siguiente^.anterior <- nodoBorrar^.anterior
   DELETE(nodoBorrar)
FIN borrarNodo


FUNCION borrar(ListaEnlazada2 lista, Nodo2 *nodoBorrar)
   SI (nodoBorrar = lista.cabeza) ENTONCES
       borrarCabeza(lista)
   SINO
       SI (nodoBorrar = lista.ultimo) ENTONCES
           borrarUltimo(lista)
       SINO
           borrarNodo(lista, nodoBorrar)
       FIN_SI
   FIN_SI
FIN borrar


FUNCION copiarUnicos(ListaEnlazada1 lista1, ListaEnlazada2 lista2)
   Nodo2 *cursor <- lista2.cabeza

   MIENTRAS (cursor <> NULL) HACER
       SI (contiene(lista2, cursor^.dato) = true)
           borrar(lista2, cursor^.dato)
       SINO
           Nodo1 *nuevoNodo
           NEW(nuevoNodo)
           SI (nuevoNodo = NULL)
               ESCRIBIR "Not memory"
           SINO
               nuevoNodo^dato <- cursor^.dato
               agregarUnico(lista1, nuevoNodo)
           FIN_SI
       FIN_SI
   FIN_MIENTRAS
FIN copiarUnicos


FUNCION mostrarLista1(ListaEnlazada1 lista)
```

```
    Nodo1 *cursor
    cursor <- lista.cabeza
    ESCRIBIR "NULL <- "
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR cursor^.dato, " <-> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR " -> NULL"
FIN mostrarLista1


FUNCION mostrarLista2(ListaEnlazada2 lista)
    Nodo2 *cursor
    cursor <- lista.cabeza
    ESCRIBIR "NULL <- "
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR cursor^.dato, " <-> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR " -> NULL"
FIN mostrarLista2


FUNCION mostrarResultados(ListaEnlazada1 lista1, ListaEnlazada2 lista2)
    ESCRIBIR "RESULTADO"

    ESCRIBIR "Lista doblemente enlazada original"
    mostrarLista1(lista1)
    ESCRIBIR "Lista simplemente enlazada original"
    mostrarLista2(lista2)

    copiarUnicos(lista1, lista2)

    ESCRIBIR "====================================="

    ESCRIBIR "Lista doblemente enlazada modificada"
    mostrarLista1(lista1)
    ESCRIBIR "Lista simplemente enlazada modificada"
```

```
   mostrarLista2(lista2)
FIN mostrarResultados


PROGRAMA_PRINCIPAL
   ListaEnlazada1 lista1
   ListaEnlazada2 lista2
   ESCRIBIR "Llenar listas doblemente enlazada"
   llenarDatos1(lista1)
   ESCRIBIR "---------------------------------"
   ESCRIBIR "Llenar listas doblemente enlazada"
   llenarDatos2(lista2)

   mostrarResultados(lista1, lista2)
FIN PROGRAMA_PRINCIPAL
```

Punto 2.

```
// Estructuras de datos utilizadas
DEFINIR Nodo
   entero corto dato

   Nodo *siguiente
   Nodo *anterior
FIN Nodo

// Crear una estructura personalizada para representar las listas enlazadas
REGISTRO ListaEnlazada
   Nodo *cabeza
   Nodo *ultimo
FIN REGISTRO


FUNCION insercionOrdenada(ListaEnlazada lista, Nodo *nodo)
   SI (nodo^.dato < lista.cabeza^.dato) ENTONCES
       nodo^.siguiente <- lista.cabeza
       lista.cabeza^.antes <- nodo
       lista.cabeza <- nodo
   SINO
       SI (nodo^.dato > lista.ultimo^.dato) ENTONCES
           lista.ultimo^.siguiente <- nodo
           nodo^.anterior <- lista.ultimo
           lista.ultimo <- nodo
       SINO
           Nodo *cursor <- lista.cabeza

           MIENTRAS (cursor^.siguiente != NULL) AND (nodo^.dato >= cursor^.dato) HACER
               cursor <- cursor^.siguiente
           FIN_MIENTRAS

           nodo^.anterior <- cursor^.anterior
           nodo^.anterior^.siguiente <- nodo
           nodo^.siguiente <- cursor
           cursor^.anterior <- nodo
       FIN_SI
```

```
   FIN_SI
FIN insercionOrdenada


FUNCION contiene(ListaEnlazada lista, entero dato)
   Nodo1 *cursor <- lista.cabeza
   repetido <- false

   MIENTRAS (cursor^.siguiente <> NULL) AND (repetido = false) ENTONCES
       SI (dato = cursor^.dato) ENTONCES
           repetido <- true
       FIN_SI
   FIN_MIENTRAS

   RETORNAR repetido
FIN contiene


FUNCION insercionUnica(ListaEnlazada lista, Nodo *nodo)
   SI (contiene(lista, nodo^.dato) = false) ENTONCES
       insercionOrdenada(lista, nodo)
   FIN_SI
FIN insercionUnica


FUNCION pedirDatos(Nodo *nodo)
   ESCRIBIR "Ingrese un numero: "
   LEER(nodo^.dato)
FIN pedirDatos


FUNCION llenarDatos(ListaEnlazada lista)
   ESCRIBIR  "LLENADO DE DATOS"
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo *nodo
       NEW(nodo)
       SI (nodo = NULL) ENTONCES
```

```
            ESCRIBIR "No hay memoria disponible en el sistema"
        SINO
            pedirDatos(nodo)
            insercionOrdenada(lista, nodo)
            ESCRIBIR "Desea agregar otro nodo a la lista? [S/N]: "
            LEER(continuar)
        FIN_SI
    FIN_MIENTRAS
FIN llenarDatos


FUNCION moverCabeza(ListaEnlazada base, ListaEnlazada objetivo)
    Nodo *nodoMover <- base.cabeza
    base.cabeza <- base.cabeza^.siguiente
    base.cabeza^.anterior <- NULL
    insercionUnica(objetivo, nodoMover)
FIN moverCabeza


FUNCION moverUltimo(ListaEnlazada base, ListaEnlazada objetivo)
    Nodo *nodoMover <- base.ultimo
    base.ultimo <- base.ultimo^.anterior
    base.ultimo^.siguiente <- NULL
    insercionUnica(objetivo, nodoMover)
FIN moverUltimo


FUNCION moverNodo(Nodo *nodoMover, ListaEnlazada base, ListaEnlazada objetivo)
    nodoMover^.anterior^.siguiente <- nodoMover^.siguiente
    nodoMover^.siguiente^.anterior <- nodoMover^.anterior
    insercionUnica(objetivo, nodoMover)
FIN moverNodo


FUNCION mover(Nodo *nodoMover, ListaEnlazada base, ListaEnlazada objetivo)
    SI (nodoMover = base.cabeza) ENTONCES
        moverCabeza(base, objetivo)
    SINO
        SI (nodoMover = base.ultimo) ENTONCES
            moverUltimo(base, objetivo)
```

```
        SINO
            moverNodo(nodoMover, base, objetivo)
        FIN_SI
    FIN_SI
FIN mover


FUNCION moverDatosPares(ListaEnlazada lista, ListaEnlazada objetivo)
    Nodo *cursor <- lista.cabeza

    MIENTRAS (cursor <> NULL) HACER
        Nodo *siguienteAux <- cursor^.siguiente
        SI (cursor^.dato MOD 2 = 0) ENTONCES
            mover(cursor, lista, objetivo)
        FIN_SI
        cursor <- siguienteAux
    FIN_MIENTRAS
FIN moverDatosPares


FUNCION mostrarLista(ListaEnlazada lista)
    Nodo *cursor
    cursor <- lista.cabeza
    ESCRIBIR "NULL <- "
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR cursor^.dato, " <-> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR " -> NULL"
FIN mostrarLista


PROGRAMA_PRINCIPAL
    ListaEnlazada listaOriginal
    ListaEnlazada listaPares

    ESCRIBIR  "LLENADO DE DATOS LISTA 1"
    llenarDatos(listaOriginal)
```

```
    ESCRIBIR  "-------------------------"
    ESCRIBIR  "Lista original"
    mostrarLista(listaOriginal)
    ESCRIBIR  "-------------------------"
    moverDatosPares(listaOriginal, listaPares)
    ESCRIBIR  "Lista original sin valores pares"
    mostrarLista(listaOriginal)
    ESCRIBIR  "Lista con los valores pares"
    mostrarLista(listaPares)
FIN PROGRAMA_PRINCIPAL
```

Punto 3.

```
// Estructuras de datos utilizadas
DEFINIR Nodo
   char[40] nombre

   Nodo *siguiente
   Nodo *anterior
FIN Nodo

// Crear una estructura nombrelizada para representar las listas enlazadas
REGISTRO ListaEnlazada
   Nodo *cabeza
   Nodo *ultimo
FIN REGISTRO


// Llenar los campos de un nodo dado
FUNCION pedirDatos(Nodo nombre)
   ESCRIBIR "Introduzca el nombre: "
   LEER(nombre^.nombre)
FIN pedirDatos


// Append nodo O(1)
FUNCION agregarNodo(ListaEnlazada lista, Nodo *nombre)
   nombre^.anterior <- NULL
   nombre^.siguiente <- NULL

   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- nombre
   SINO
       lista.ultimo^.siguiente <- nombre
       nombre^.anterior <- lista^.ultimo
   FIN_SI
   lista.ultimo <- nombre
FIN agregarNodo
```

```
FUNCION registrarNombre(ListaEnlazada lista)
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo *nombre
       NEW(nombre)
       SI (nombre = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatos(nombre)
           agregarNodo(lista, nombre)
           ESCRIBIR "Desea agregar otro registro de nombre a la lista? [S/N]: "
           LEER(continuar)
       FIN_SI
   FIN_MIENTRAS
FIN registrarNombre


FUNCION buscarNodoPorNombre(ListaEnlazada lista, char[] nombre)
   encontrado <- false
   Nodo *cursor <- lista.cabeza
   Nodo *nombreEncontrada # NULL

   MIENTRAS (cursor <> NULL) AND (encontrado = false) HACER
       SI (cursor^.nombre = nombre) ENTONCES
           nombreEncontrada <- cursor
           encontrado <- true
       SINO
           cursor <- cursor^.siguiente
       FIN_SI
   FIN_MIENTRAS

   RETORNAR nombreEncontrada
FIN buscarNodoPorNombre


FUNCION moverCabeza(ListaEnlazada base, ListaEnlazada objetivo)
   Nodo *nodoMover <- base.cabeza
   base.cabeza <- base.cabeza^.siguiente
```

```
   agregarNodo(objetivo, nodoMover)
FIN moverCabeza


FUNCION moverUltimo(ListaEnlazada base, ListaEnlazada objetivo)
   Nodo *nodoMover <- base.ultimo
   base.ultimo <- base.ultimo^.anterior
   agregarNodo(objetivo, nodoMover)
FIN moverUltimo


FUNCION moverNodo(Nodo *nodoMover, ListaEnlazada base, ListaEnlazada objetivo)
   nodoMover^.anterior^.siguiente <- nodoMover^.siguiente
   nodoMover^.siguiente^.anterior <- nodoMover^.siguiente
   agregarNodo(objetivo, nodoMover)
FIN moverNodo


FUNCION mover(Nodo *nodoMover, ListaEnlazada base, ListaEnlazada objetivo)
   SI (nodoMover = base.cabeza) ENTONCES
       moverCabeza(base, objetivo)
   SINO
       SI (nodoMover = base.ultimo) ENTONCES
           moverUltimo(base, objetivo)
       SINO
           moverNodo(nodoMover, base, objetivo)
       FIN_SI
   FIN_SI
FIN mover


FUNCION trasladarNombre(char[] accion, ListaEnlazada lista, ListaEnlazada objetivo)
   continuar <- "s"

   SI (lista.cabeza = NULL) ENTONCES
       ESCRIBIR "La lista está vacía."
       RETORNAR
   FIN_SI

   MIENTRAS (continuar = "s") HACER
```

```
        ESCRIBIR "Nombre que desea ", accion, ": "
        LEER(nombre)

        Nodo *nodoMover <- buscarNodoPorNombre(lista, nombre)

        SI (nodoMover = NULL) ENTONCES
            ESCRIBIR "Nombre no encontrado en la lista."
        SINO
            ESCRIBIR "Seguro que desea ", accion, " el nombre [S/N]: "
            LEER(seguro)
            SI (seguro = "s") ENTONCES
                mover(nodoMover, lista, objetivo)
            SINO
                ESCRIBIR "Acción de ", accion, " cancelada."
            FIN_SI
        FIN_SI

        ESCRIBIR "Desea ", accion, " otro nombre? [S/N]: "
        LEER(continuar)
    FIN_MIENTRAS
FIN trasladarNombre


FUNCION mostrarLista(ListaEnlazada lista)
    Nodo *cursor
    cursor <- lista.cabeza
    ESCRIBIR "NULL <- "
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR cursor^.dato, " <-> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR " -> NULL"
FIN mostrarLista


PROGRAMA_PRINCIPAL
    ListaEnlazada listaNombres
    // Crear una lista para guardar los nombres borradas
    ListaEnlazada papelera
```

```
    ESCRIBIR  "Llenar lista de nombres"
    registrarNombre(listaNombres)

    ESCRIBIR  "-------------------------"
    ESCRIBIR  "Lista de nombres"
    mostrarLista(listaNombres)

    ESCRIBIR  "-------------------------"
    ESCRIBIR  "Borrar nombres"
    trasladarNombre("borrar", listaNombres, papelera)
    ESCRIBIR  "Lista de nombres"
    mostrarLista(listaNombres)
    ESCRIBIR  "Papelera"
    mostrarLista(papelera)

    ESCRIBIR  "-------------------------"
    ESCRIBIR  "Recuperar nombres"
    trasladarNombre("recuperar", papelera, listaNombres)
    ESCRIBIR  "Lista de nombres"
    mostrarLista(listaNombres)
    ESCRIBIR  "Papelera"
    mostrarLista(papelera)
FIN PROGRAMA_PRINCIPAL
```

Punto 4.

```
DEFINIR Persona
   char[40] id
   char[40] nombre

   Persona *siguiente
FIN Persona

DEFINIR Notas
   char[40] id
   double[3] notas

   Notas *siguiente
FIN Notas

DEFINIR Estudiante
   char[40] id
   char[40] nombre
   double[3] notas
   double promedio

   Estudiante *siguiente
FIN Estudiante

REGISTRO ListaPersonas
   Persona *cabeza
   Persona *ultimo
FIN REGISTRO

REGISTRO ListaNotas
   Notas *cabeza
   Notas *ultimo
FIN REGISTRO

REGISTRO ListaEstudiantes
   Estudiante *cabeza
   Estudiante *ultimo
FIN REGISTRO
```

```
FUNCION agregarPersona(ListaPersonas lista, Persona *persona)
   persona^.anterior <- NULL
   persona^.siguiente <- NULL

   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- persona
   SINO
       lista.ultimo^.siguiente <- persona
   FIN_SI
   lista.ultimo <- persona
FIN agregarPersona



FUNCION pedirDatosPersona(ListaPersonas lista, Persona *persona)
   repetido <- false

   ESCRIBIR "REGISTRAR PERSONA"
   ESCRIBIR "Ingrese el id: "
   LEER(persona^.id)

   Persona *cursor <- lista.cabeza
   MIENTRAS (cursor <> NULL) AND (repetido = false) HACER
       SI (persona^.id = cursor^.id) ENTONCES
           ESCRIBIR "Persona ya registrada"
           repetido <- true
       FIN_SI
       persona <- persona^.siguiente
   FIN_MIENTRAS

   SI (repetido = false) ENTONCES
       ESCRIBIR "Ingrese el nombre: "
       LEER(persona^.nombre)
       agregarPersona(lista, persona)
   FIN_SI
FIN pedirDatosPersona


FUNCION registrarPersona(ListaPersonas lista)
```

```
    continuar <- "s"

    MIENTRAS (continuar = "s") HACER
        // Crear nuevo nodo
        Persona *persona
        NEW(persona)
        SI (persona = NULL) ENTONCES
            ESCRIBIR "No hay memoria disponible en el sistema"
        SINO
            pedirDatosPersona(persona)
        FIN_SI
        ESCRIBIR "Desea agregar otra persona a la lista? [S/N]: "
        LEER(continuar)
    FIN_MIENTRAS
FIN registrarPersona


FUNCION pedirDatosNotas(Notas *notas)
    ESCRIBIR "REGISTRAR NOTAS"
    ESCRIBIR "Ingrese el id: "
    LEER(notas^.id)

    ESCRIBIR "Notas de los cortes."
    PARA i <- 1 hasta 3 con paso 1 HACER
        ESCRIBIR "Corte ", i, ": "
        LEER(notas^.notas[i])
    FIN_PARA
FIN pedirDatosNotas

FUNCION agregarNotas(ListaNotas lista, Notas *notas)
    notas^.anterior <- NULL
    notas^.siguiente <- NULL

    SI (lista.cabeza = NULL) ENTONCES
        lista.cabeza <- notas
    SINO
        lista.ultimo^.siguiente <- notas
    FIN_SI
    lista.ultimo <- notas
FIN agregarNotas
```

```
FUNCION registrarNotas(ListaNotas lista)
   ESCRIBIR "LLENADO DE DATOS"
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       Notas *notas
       NEW(notas)
       SI (notas = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatosNotas(notas)
           agregarNotas(lista, notas)
       FIN_SI
       ESCRIBIR "Desea agregar otra nota a la lista? [S/N]: "
       LEER(continuar)
   FIN_MIENTRAS
FIN registrarNotas


FUNCION agregarEstudiante(ListaEstudiantes lista, Estudiante *estudiante)
   estudiante^.anterior <- NULL
   estudiante^.siguiente <- NULL

   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- estudiante
   SINO
       lista.ultimo^.siguiente <- estudiante
   FIN_SI
   lista.ultimo <- estudiante
FIN agregarEstudiante

FUNCION calcularPromedio(Estudiante estudiante)
   double promedio <- estudiante^.notas[0] * 0.3
   promedio <- promedio + estudiante^.notas[1] * 0.3
   promedio <- promedio + estudiante^.notas[2] * 0.4

   RETORNAR promedio
FIN calcularPromedio
```

```
FUNCION generarEstudiante(Persona *persona, Notas *notas)
   Estudiante *estudiante
   NEW(estudiante)

   estudiante^.id <- persona^.id
   estudiante^.nombre <- persona^.nombre
   estudiante^.notas <- notas^.notas
   estudiante^.promedio <- calcularPromedio(estudiante)

   RETORNAR estudiante
FIN generarEstudiante

FUNCION cruzarListas(ListaPersonas listaPersonas, ListaNotas listaNotas)
   ListaEstudiantes listaEstudiantes
   Persona *cursorP
   Notas *cursorN

   cursorP <- listaPersonas.cabeza

   MIENTRAS (cursorP <> NULL) HACER
       cursorN <- listaNotas.cabeza
       MIENTRAS(cursorN <> NULL) HACER
           SI (cursorN^.id = cursorP^.id) ENTONCES
               Estudiante *estudiante <- generarEstudiante(cursorP, cursorN)
               agregarEstudiante(listaEstudiantes, estudiante)
           FIN_SI
           cursorN <- cursorN.siguiente
       FIN_MIENTRAS
       cursorP <- cursorP^.siguiente
   FIN_MIENTRAS
   RETORNAR listaEstudiantes
FIN cruzarListas


FUNCION mostrarEstudiante(Estudiante estudiante)
   ESCRIBIR "----------------------"
   ESCRIBIR "ESTUDIANTE"
   ESCRIBIR "Id: ", estudiante^.id
```

```
   ESCRIBIR "Nombre: ", estudiante^.nombre
   ESCRIBIR "Notas: ", estudiante^.notas
   ESCRIBIR "Promedio: ", estudiante^.promedio
FIN mostrarEstudiante


FUNCION mostrarEstudiantes(ListaEnlazada lista)
   Nodo *cursor
   cursor <- lista.cabeza
   MIENTRAS (cursor <> NULL) HACER
       mostrarEstudiante(cursor)
       cursor <- cursor^.siguiente
   FIN_MIENTRAS
FIN mostrarEstudiantes


PROGRAMA_PRINCIPAL
   ListaPersonas listaPersonas
   ListaNotas listaNotas

   registrarPersona(listaPersonas)
   ESCRIBIR "----------------------"
   registrarNotas(listaNotas)

   ListaEstudiantes listaEstudiantes <- cruzarListas(listaPersonas, listaNotas)
   mostrarEstudiantes(listaEstudiantes)
FIN PROGRAMA_PRINCIPAL
```

Punto5.

```
// Estructuras de datos utilizadas
DEFINIR Nodo
   entero dato

   Nodo *siguiente
FIN Nodo


// Crear una estructura personalizada para representar las listas enlazadas
REGISTRO ListaEnlazada
   Nodo *cabeza
   Nodo *ultimo
FIN REGISTRO


FUNCION pedirDatos(Nodo *nodo)
   ESCRIBIR "Ingrese un numero: "
   LEER(nodo^.dato)
FIN pedirDatos


FUNCION agregarNodo(ListaEnlazada lista, Nodo *nodo)
   // Balancear nodo
   nodo^.siguiente <- NULL
   // Verificar que la lista no esté vacía
   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- nodo
   SINO
       lista.ultimo^.siguiente <- nodo
   FIN_SI
   lista.ultimo <- nodo
FIN agregarNodo


FUNCION llenarDatos(ListaEnlazada lista)
   ESCRIBIR  "LLENADO DE DATOS"
   continuar <- "s"
```

```
   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo *nodo
       NEW(nodo)
       SI (nodo = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatos(nodo)
           agregarNodo(lista, nodo)
       FIN_SI
       ESCRIBIR "Desea agregar otro nodo a la lista? [S/N]: "
       LEER(continuar)
   FIN_MIENTRAS
FIN llenarDatos


FUNCION pop(ListaEnlazada lista)
   Nodo *nodoBorrar <- lista.cabeza
   lista.cabeza <- lista.cabeza^.siguiente
   DELETE(nodoBorrar)
FIN pop


FUNCION detach(ListaEnlazada lista)
   Nodo *cursor <- lista.cabeza

   SI (lista.cabeza <> NULL ) ENTONCES

       MIENTRAS(cursor^.siguiente^.siguiente <> NULL) HACER
           cursor <- cursor^.siguiente
       FIN_MIENTRAS

       Nodo *nodoBorrar
       nodoBorrar <- lista.ultimo
       lista.ultimo <- cursor
       lista.ultimo^.siguiente <- NULL
       nodoBorrar^.siguiente <- NULL
       DELETE(nodoBorrar)
   FIN_SI
FIN detach
```

```
FUNCION borrarNodoMitad(ListaEnlazada lista, Nodo *nodoBorrar)
   Nodo *cursor
   cursor <- lista.cabeza

   MIENTRAS (cursor^.siguiente^.siguiente <> NULL) AND (cursor^.siguiente <> nodoBorrar)
HACER
       cursor <- cursor^.siguiente
   FIN_MIENTRAS

   SI (cursor^.siguiente = nodoBorrar) HACER
       nodoBorrar <- cursor^.siguiente
       cursor^.siguiente <- nodoBorrar^.siguiente
       nodoBorrar^.siguiente <- NULL
       DELETE(nodoBorrar)
   FIN_SI
FIN borrarNodoMitad


FUNCION borrarNodo(ListaEnlazada lista, Nodo *nodoBorrar)
   SI (nodoBorrar = lista.cabeza) ENTONCES
       pop(lista)
   SINO
       SI (nodoBorrar = lista.ultimo) ENTONCES
           detach(lista)
       SINO
           borrarNodoMitad(lista, nodoBorrar)
       FIN_SI
   FIN_SI
FIN borrarNodo


FUNCION diferenciarListas(ListaEnlazada listaA, ListaEnlazada listaB)
   Nodo *cursorA, *cursorB

   cursorB <- listaB.cabeza

   MIENTRAS (cursorB <> NULL) HACER
       cursorA <- listaA.cabeza
```

```
        MIENTRAS(cursorA <> NULL) HACER
            siguienteAux <- cursorA.siguiente
            SI (cursorA^.dato = cursorB^.dato) ENTONCES
                borrarNodo(listaA, cursorA)
            FIN_SI
            cursorA <- siguienteAux
        FIN_MIENTRAS
        cursorB <- cursorB^.siguiente
    FIN_MIENTRAS
FIN diferenciarListas


FUNCION mostrarLista(ListaEnlazada lista)
    Nodo *cursor
    cursor <- lista.cabeza
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR cursor^.dato, " -> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR "NULL"
FIN mostrarLista


PROGRAMA_PRINCIPAL
    ListaEnlazada lista1
    ListaEnlazada lista2

    ESCRIBIR  "LLENADO DE DATOS LISTA 1"
    llenarDatos(lista1)
    ESCRIBIR  "-------------------------"
    ESCRIBIR  "LLENADO DE DATOS LISTA 2"
    llenarDatos(lista2)

    ESCRIBIR "Lista 1 original"
    mostrarLista(lista1)
    ESCRIBIR "Lista 2"
    mostrarLista(lista2)

    diferenciarListas(lista1, lista2)
```

```
   ESCRIBIR "Lista 1 retirando elementos iguales."
   mostrarLista(lista1)
FIN PROGRAMA_PRINCIPAL
```

Punto 6.

```
DEFINIR Nodo1
   entero codigo
   Nodo1 *siguiente
FIN Nodo

DEFINIR ListaEnlazada1
   Nodo1 *cabeza
   Nodo1 *ultimo
FIN ListaEnlazada

DEFINIR Nodo2
   entero codigo
   char accion
   Nodo2 *siguiente
FIN Nodo2

DEFINIR ListaEnlazada2
   Nodo2 *cabeza
   Nodo2 *ultimo
FIN ListaEnlazada2



// Append nodo O(1)
FUNCION agregarNodo1(ListaEnlazada1 lista, Nodo1 *nodo)
   nodo^.siguiente <- NULL
   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- nodo
   SINO
       lista.ultimo^.siguiente <- nodo
   FIN_SI
   lista.ultimo <- nodo
FIN agregarNodo1



FUNCION pedirDatos1(Nodo1 *nodo)
   ESCRIBIR "Ingrese un numero: "
   LEER(nodo^.dato)
FIN pedirDatos1
```

```
FUNCION llenarDatos1(ListaEnlazada1 lista1)
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo1 *nodo
       NEW(nodo)
       SI (nodo = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatos1(nodo)
           agregarNodo1(lista, nodo)
           ESCRIBIR "Desea agregar otro nodo a la lista? [S/N]: "
           LEER(continuar)
       FIN_SI
   FIN_MIENTRAS
FIN llenarDatos1


// Append nodo O(1)
FUNCION agregarNodo2(ListaEnlazada2 lista, Nodo2 *nodo)
   nodo^.siguiente <- NULL

   SI (lista.cabeza = NULL) ENTONCES
       lista.cabeza <- nodo
   SINO
       lista.ultimo^.siguiente <- nodo
   FIN_SI
   lista.ultimo <- nodo
FIN agregarNodo2


FUNCION pedirDatos2(Nodo2 *nodo)
   ESCRIBIR "Ingrese un numero: "
   LEER(nodo^.dato)
   ESCRIBIR "Ingrese la accion: "
   LEER(nodo^.accion)
FIN pedirDatos2
```

```
FUNCION llenarDatos2(ListaEnlazada2 lista2)
   ESCRIBIR  "LLENADO DE DATOS"
   continuar <- "s"

   MIENTRAS (continuar = "s") HACER
       // Crear nuevo nodo
       Nodo2 *nodo
       NEW(nodo)
       SI (nodo = NULL) ENTONCES
           ESCRIBIR "No hay memoria disponible en el sistema"
       SINO
           pedirDatos2(nodo)
           agregarNodo2(lista, nodo)
           ESCRIBIR "Desea agregar otro nodo a la lista? [S/N]: "
           LEER(continuar)
       FIN_SI
   FIN_MIENTRAS
FIN llenarDatos2


FUNCION buscarNodoPorCodigo(ListaEnlazada1 lista1, entero codigo)
   encontrado <- false
   Nodo1 *cursor <- lista1.cabeza
   Nodo1 *nodoEncontrado # NULL

   MIENTRAS (cursor <> NULL) AND (encontrado = false) HACER
       SI (cursor^.codigo = codigo) ENTONCES
           nodoEncontrado <- cursor
           encontrado <- true
       SINO
           cursor <- cursor^.siguiente
       FIN_SI
   FIN_MIENTRAS

   RETORNAR nodoEncontrado
FIN buscarNodoPorCodigo
```

```
FUNCION pop(ListaEnlazada1 lista1)
   Nodo *nodoBorrar <- lista1.cabeza
   lista1.cabeza <- lista1.cabeza^.siguiente
   DELETE(nodoBorrar)
FIN pop


FUNCION detach(ListaEnlazada1 lista1)
   Nodo1 *cursor
   cursor <- lista1.cabeza

   MIENTRAS(cursor^.siguiente^.siguiente <> null) HACER
       cursor <- cursor^.siguiente
   FIN_MIENTRAS

   Nodo *nodoBorrar
   nodoBorrar <- lista1.ultimo
   lista1.ultimo <- cursor
   lista1.ultimo^.siguiente <- null
   nodoBorrar^.siguiente <- null
   DELETE(nodoBorrar)
FIN detach


FUNCION borrarNodoMitad(ListaEnlazada1 lista1, Nodo1 *nodoBorrar)
   Nodo1 *cursor
   cursor <- lista1.cabeza^.siguiente

  MIENTRAS (cursor^.siguiente^.siguiente <> NULL) AND (cursor^.siguiente <> nodoBorrar) HACER
       cursor <- cursor^.siguiente
  FIN_MIENTRAS

   SI (cursor^.siguiente = nodoBorrar) HACER
       cursor^.siguiente <- nodoBorrar^.siguiente
       nodoBorrar^.siguiente <- NULL
       DELETE(nodoBorrar)
   FIN_SI
FIN borrarNodoMitad
```

```
FUNCION borrarNodo(ListaEnlazada1 lista1, Nodo1 *nodoBorrar)
   SI (nodoBorrar = lista1.cabeza) ENTONCES
       pop(lista1)
   SINO
       SI (nodoBorrar = lista1.ultimo) ENTONCES
           detach(lista1)
       SINO
           borrarNodoMitad(lista1, nodoBorrar)
       FIN_SI
   FIN_SI
FIN borrarNodo


FUNCION borrar(ListaEnlazada1 lista1, entero codigo)
   Nodo1 *nodoBorrar

   REPETIR
       nodoBorrar <- buscarNodoPorCodigo(lista1, codigo)
       SI (nodoBorrar <> NULL) ENTONCES
           borrarNodo(lista1, nodoBorrar)
       FIN_SI
   HASTA (nodoBorrar = NULL)
FIN borrar


FUNCION ejecutarAccion(ListaEnlazada1 lista1, ListaEnlazada2 lista2)
   Nodo2 *cursor2 <- lista2^.cabeza
   MIENTRAS (cursor2 <> NULL) HACER
       SI (cursor2^.accion = 'A') ENTONCES
           Nodo1 *nuevoNodo
           NEW(nuevoNodo)
           SI (nuevoNodo = NULL)
               ESCRIBIR "Not memory"
           SINO
               nuevoNodo^.codigo <- cursor2^.codigo
               agregarNodo1(lista1, nuevoNodo)
           FIN_SI
       SINO
           SI (cursor2^.accion = 'R') ENTONCES
               borrar(lista1, cursor2^.codigo)
```

```
            FIN_SI
        FIN_SI
        cursor2 <- cursor2^.siguiente
    FIN_MIENTRAS
FIN ejecutarAccion


FUNCION mostrarLista1(ListaEnlazada1 lista1)
    Nodo1 *cursor
    cursor <- lista1.cabeza
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR "{ dato: ", cursor^.dato, "} -> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR "NULL"
FIN mostrarLista1


FUNCION mostrarLista2(ListaEnlazada2 lista2)
    Nodo2 *cursor
    cursor <- lista2.cabeza
    MIENTRAS (cursor <> NULL) HACER
        ESCRIBIR "{ dato: ", cursor^.dato, "accion: ", cursor^.accion, "} -> "
        cursor <- cursor^.siguiente
    FIN_MIENTRAS

    ESCRIBIR "NULL"
FIN mostrarLista2


PROGRAMA_PRINCIPAL
    ListaEnlazada1 lista1
    ListaEnlazada2 lista2

    ESCRIBIR  "LLENADO DE DATOS LISTA 1"
    llenarDatos1(lista1)
    ESCRIBIR  "-------------------------"
    ESCRIBIR  "LLENADO DE DATOS LISTA 2"
    llenarDatos2(lista2)
```

```
    ESCRIBIR "Lista 1 original"
    mostrarLista1(lista1)
    ESCRIBIR "Lista 2"
    mostrarLista2(lista2)

    ejecutarAccion(lista1, lista2)
    ESCRIBIR "Resultado"
    ESCRIBIR "Lista 1"
    mostrarLista1(lista1)
    ESCRIBIR "Lista 2"
    mostrarLista2(lista2)
FIN PROGRAMA_PRINCIPAL
```