|  |  |
| --- | --- |
| Estructuras | Funciones asociadas |
| 1º Arreglo de listas simplemente entrelazadas de arboles.  typedef struct { int comision; nodoMateria\* listaMateria; }Celda;  Contiene:  1-Numero de comisión.  2-Lista de materias. | 1-altaArreglo: Es la función principal que llama las demás funciones y permite la carga de la información del archivo de forma ordenada.  2-buscarPosComision:Busca la comisión ingresada por parámetro para usar como posición del arreglo.  3-agregarComision: Crea la posición y comisión correspondiente.  4-altaMaterias:Ubicada la posición del arreglo, ingresa las materias correspondientes y notas de los alumnos que irán a la estructura del árbol.  5-mostrarComisiones: Muestra en pantalla las comisiones del arreglo.  6-eliminarAlumno: Elimina un alumno de todas las materias de una comisión según su legajo.  7-existeLegajoEnComision: Verifica si existe un legajo en la comisión ingresada por parámetro.  8-copiarRegistro : Recorre el archivo y acomoda la información en las estructuras de forma correspondiente.  9-actualizarArchivo : Si se ha modificado información del programa, se sobrescribe el archivo con la nueva información, o la elimina de ser el caso |
| 2º Lista simplemente entrelazada de arboles  typedef struct nodoMateria {  stMateria dato; arbolNotas\*arbolNota; struct nodoMateria\*siguiente; }nodoMateria;  Contiene:  1-Materias.  2-Arbol de notas. | 1-mostrarMateria:Muestra en pantalla la información de la materia.  2-inicListaMateria: Inicia la lista de materias en NULL.  3-crearNodoMateria: Crea un nodo tipo materia con la información suministrada.  4-agregarPpioMateria: Ingresa el nuevo nodo creado al inicio del a lista.  5-mostrarNodoMateria: Muestra la información contenida en el nodo materia suministrado.  6-mostrarListaMateria: Recorre la lista ingresada y muestra su información.  7-existeMateria: Busca una materia por su id e indica si existe o no en la lista.  8-buscarPosMateria: Busca una materia por su numero de identificación y devuelve el nodo correspondiente a esa posición. |
| 3º Árbol de alumnos  Contiene:  typedef struct arbolNotas{  stNota nota;  stAlumno alumno; struct arbolNotas\* izq; struct arbolNotas\* der; }arbolNotas;  1-Alumno.  2-Nota. | 1-inicArbolNotas: Inicializa un árbol en NULL.  2-mostrarNota: Muestra en pantalla la información de la estructura nota.  3-mostrarAlumno: Muestra en pantalla la información de la estructura alumno.  4-mostrarNodoNotas: Muestra la información del nodo.  5-crearArbolNotas: Crea el árbol con sus respectivos datos.  6-insertarNotas: Ingresa nuevos nodos al árbol en orden por valor de la nota.  7-buscarNota: Recorre el árbol y devuelve el SubArbol con la nota pedida según el legajo y numero de examen pasado por parametro.  8-inOrderNotas: Recorre el árbol y muestra su información ordenando las notas de menor a mayor.  9-nmd :Busca y retorna el nodo mas a la derecha del arbol pasado por parámetro.  10-nmi: Busca y retorna el nodo mas a la izquierda del arbol pasado por parámetro.  11-existeNota: Verifica si existe una nota según legajo y numero de examen.  12-modificarNota: Busca y modifica una nota de un alumno.  13-cantNotasXLegajo: Retorna la cantidad de notas asociadas a un legajo.  14-borrarLegajo: Elimina todos los nodos vinculados a un legajo.  15-existeLegajo: Verifica si un legajo existe en el árbol.  16-mostrarNotaXLegajo: Recorre y muestra las notas según el legajo suministrado por parámetro. |