|  |
| --- |
| UDE |
| Taller 1 Licenciatura en informática |
| Facundo Francia  Mathias Pouso  Nicolás Martino |

Contenido

[Elección de estructuras de datos 2](#_Toc1403900)

[Justificación 2](#_Toc1403901)

[Escritura en C++ y cabezales 3](#_Toc1403902)

[Diagrama de inclusiones 6](#_Toc1403903)

[Pseudocódigo 7](#_Toc1403904)

[Comandos del programa 7](#_Toc1403905)

[Funciones y Procedimiento auxiliares 14](#_Toc1403906)

[Planificación – Gantt 18](#_Toc1403907)

# Elección de estructuras de datos

## Justificación

**EnumMensajes:**

Se cuenta con un enumerado de mensajes el cual contendrá valores para cada tipo de mensajes, teniendo una función que, dependiendo del tipo de mensaje, despliegue cierto mensaje, esto nos ayuda a el mantenimiento de los errores ya que, si se necesita agregar, editar o eliminar un error, se hace desde un solo lugar.

**String:**

Se elige la estructura de string dinámico para trabajar con los textos debido a que es más óptimo en el uso de memoria, ya que calcula el tamaño exacto para los textos.

**ListaParametros:**

Elegimos una lista dinámica ya que no sabemos cuántos parámetros ingresara el usuario, así que esta puede ser muy grande.

**Termino:**

Armamos una estructura para representar el término. El mismo está conformado por 3 componentes: Signo, Base y Exponente.

**ListaTermino:**

Esta lista conformada por nodos de tipo Termino representa todos los términos que poseerá el polinomio. Se arma de tipo dinámica ya que no conocemos la cantidad de términos que tendrá el polinomio.

**Polinomio:**

Elegimos el tipo struct para representar el polinomio formado por un String para el nombre, y una lista de términos.

**ABBPolinomio:**

Elegimos una estructura de tipo Árbol Binario de Búsqueda, ya que la letra pide que los polinomios se ordenen en base a su nombre. Y como dice el nombre del tipo de dato, este tipo de estructura tiene mejores posibilidades a la hora de buscar.

**EnumComandos:**

Enumerado para mapear los comandos que se ingresan. Elegimos un enumerado para así, en caso de tener que agregar valores, no sería necesario volver a escribir los valores en cada lugar que se utilizan, sino solo modificar el enumerado.

## Escritura en C++ y cabezales

**Boolean.h**

typedef enum {FALSE, TRUE} Boolean;

**EnumMensajes.h**

Typedef enum {OK, NOMBRE\_INVALIDO, NUMERO\_INVALIDO, COMANDO\_INVALIDO,

YA\_EXISTE\_POLINOMIO, NO\_EXISTE\_POLINOMIO, NO\_INGRESO\_NOMBRE, NO\_INGRESO\_TERMINO, PARAMETROS\_INVALIDOS, LISTA\_VACIA, COEF\_IGUAL\_CERO } TipoMensaje;

void MostrarMensaje(TipoMensaje tipo);

**String.h**

#include "Boolean.h"

#include "EnumMensajes.h"

const int MAX = 80;

typedef char \* String;

int ConvertirCharANumero(String s);

void ConvertirTerminoAString(String s1, String &s2);

Boolean EsValidoNombre(String s);

Boolean EsValidoNumero(String s);

Boolean ValidarNombreArchivo(String s);

**ListaParametros.h**

#include "String.h"

typedef struct nodo\_string{String info; nodo\_string \* sig;} Nodo;

typedef Nodo \* ListaParam;

int CantParametros(ListaParam l);

void Parsear(String s, ListaParam &lista);

**Termino.h**

typedef struct {char signo; int base; int exponente;} Termino;

char DarSigno(Termino term);

int DarExponente(Termino term);

int DarBase(Termino term);

Termino ConvertirStringATermino (String s);

**ListaTerminos.h**

#include "Termino.h"

typedef struct nodo\_termino{Termino info; nodo\_termino \* sig;} Nodo;

typedef Nodo \* ListaTerm;

int CantTerminos(ListaTerm lista);

void InsTermBack(ListaTerm &lista, Termino term);

**Polinomio.h**

#include "ListaTerminos.h"

#include "ListaParametros.h"

typedef struct nodo\_int{String nombre; ListaTerm Listaterminos} Polinomio;

void DarNombre(Polinomio p);

void DarListaTerm(Polinomio p, ListaTerm &lista);

void Crear(Polinomio &p, ListaParam lista);

void Sumar(Polinomio a, Polinomio b, Polinomio &resu);

void Multiplicar(Polinomio a, Polinomio b, Polinomio &resu);

int Evaluar(Polinomio p, int x);

Boolean EsRaiz(Polinomio p, int x);

void GuardarPolinomio(Polinomio p, String archivo);

void RecuperarPolinomio(Polinomio &p, String poli, String archivo);

**ABBPolinomio.h**

#include "Polinomio.h"

typedef struct nodoN {Polinomio info; nodoN \*hizq; nodoN \*hder;} nodo;

typedef nodo \*Arbol;

void InicializarArbol(Arbol &ABB);

void MostrarABBPoli(Arbol ABBPoli);

void InsPoliABBPoli(Arbol &ABBPoli, Polinomio p);

Boolean ExistePolinomio(Arbol &ABBPoli, String nombre\_poli);

void DarPolinomio(Arbol &ABBPoli, String nombre\_poli);

void BorrarABB(Arbol &ABBPoli);

void Salir(Arbol &ABBPoli, ListaParametros &l);

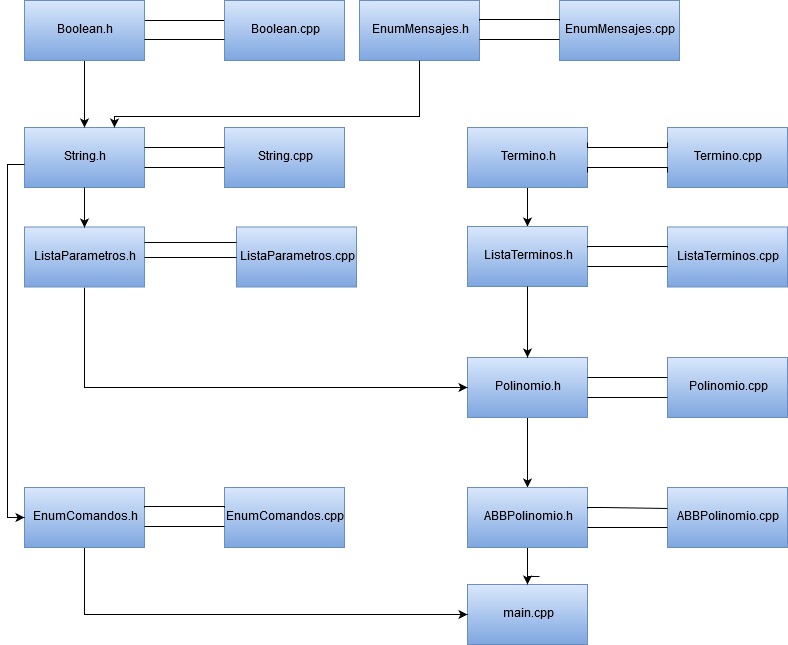
**EnumComandos.h**

#include "String.h"

typedef enum{CREAR, SUMAR, MULTIPLICAR, EVALUAR, ES\_RAIZ, MOSTRAR, GUARDAR, RECUPERAR, SALIR} EnumComandos;

Boolean ValidarComando(String s);

# Diagrama de inclusiones



# Pseudocódigo

## Comandos del programa

|  |  |
| --- | --- |
| CrearPolinomio | |
| Entrada: | Lista de String con parámetros cargados |
| Salida: | Nuevo Polinomio almacenado en el ABBPolinomio |
| Método: | Creo nueva Lista de terminos  Defino variable signo de tipo char  Defino variable base de tipo int  Defino variable cant\_terminos con CantParametros y le resto 2  Tomo segundo elemento de la lista de parámetros  Si elemento no es NULL  Si EsValidoNombre es OK  Tomo el tercer elemento de la lista  Si el elemento no es NULL  Si tercer termino es diferente a 0 y cant\_terminos es mayor a 2  Mientras haya elementos tomo elemento  Recorro String  Mientras no se llegue a carácter fin de linea  Tomo carácter del string  Si carácter es igual a ‘-‘  Guardo en variable signo ‘-‘  Tomo resto del string y lo guardo en una nueva variable temp  Valido variable temp con EsValidoNumero  Si EsValidoNumero es OK  Convierto el string temp a numero y lo guardo en variable base  Sino  MuestroMensaje de TERMINO\_INVALIDO  FinSi  Sino  Guardo en variable signo ‘+’  Tomo resto del string y lo guardo en una nueva variable temp  Valido variable temp con EsValidoNumero  Si EsValidoNumero es OK  Convierto el string temp a numero y lo guardo en variable base  Sino  MuestroMensaje de numero invalido  FinSino  Creo Termino con variables base, signo y cant\_terminos como exponente  Agrego termino al fila nueva lista de terminos  Resto 1 a cant\_terminos  Tomo siguiente string de la lista  FinMientras  FinMientras  Creo polinomio con nombre y la nueva lista de términos  Agrego polinomio a ABB  Sino  MuestroMensaje COEF\_IGUAL\_CERO  Sino  MuestroMensaje NO\_INGRESO\_TERMINO  Sino  MuestroMensaje de NOMBRE\_INVALIDO  Sino  MuestroMensaje de NO\_INGRESO\_NOMBRE |

|  |  |
| --- | --- |
| Sumar | |
| Entrada: | Lista de string cargada con parámetros |
| Salida: | Nuevo polinomio cargado en el ABBPolinomio |
| Método: | Obtengo cantidad de elementos de la lista de string.  Si Cantidad de elemento es igual a 4  . Tomo segundo elemento  . Si EsValidoNombre de NombreNuevo es OK  . . Si ExistePolinomio de NombreNuevo es NO\_EXISTE\_POLINOMIO  . . . Tomo tercer y cuarto elemento  . . . Si ExistePolinomio es OK  . . . . Si lista1 es más grande o son iguales  . . . . . Recorro primer lista  . . . . . Mientras lista1 no sea vacía  . . . . . . Obtengo exponente de término de primer lista  . . . . . . Si signo es igual ‘-’  . . . . . . Tomo base y multiplico por -1  . . . . . . FinSi  . . . . . .  . . . . . . Recorro segunda lista desde j  . . . . . . Mientras lista2 no sea vacía y variable Booleana sea FALSE  . . . . . . . Obtengo exponente de término de segunda lista  . . . . . . . Si ambos exponentes son iguales  . . . . . . . . Seteo variable en TRUE  . . . . . . . . Si signo es igual a ‘-’  . . . . . . . . Tomo base y multiplico por -1  . . . . . . . . FinSi  . . . . . . . FinSi  . . . . . . . Creo nuevo termino con base igual a suma de ambas bases y exponente  . . . . . . . Agrego término a lista de términos  . . . . . . . Guardo última posición recorrida en j  . . . . . . . Sino  . . . . . . . Creo nuevo termino con base igual a base y exponente  . . . . . . . Agrego término a lista de términos  . . . . . . . FinSi  . . . . . .FinMientras  . . . . . FinMientras  . . . . Sino  . . . . Realizo el mismo procedimiento recorriendo primero lista2 y luego lista1..  . . . . FinSi  . . . .  . . . . Agrego lista de términos a Polinomio vacío  . . . . Agrego polinomio al ABBPolinomio  . . . Sino  . . . MuestroMensaje POLINOMIO\_NO\_EXISTE  . . Sino  . . MuestroMensaje POLINOMIO\_YA\_EXISTE  . Sino  . MuestroMensaje NOMBRE\_INVALIDO  Sino  MuestroMensaje PARAMETROS\_INVALIDOS |

|  |  |
| --- | --- |
| Multiplicar | |
| Entrada: | Lista de string con parámetros cargados |
| Salida: | Polinomio cargado en ABBPolinomio |
| Método: | Si Cantidad de elemento es igual a 4  . Tomo segundo elemento  . Si EsValidoNombre de NombreNuevo es OK  . . Si ExistePolinomio de NombreNuevo es NO\_EXISTE\_POLINOMIO  . . . Tomo tercer y cuarto elemento  . . . Si ExistePolinomio es OK  . . . . Recorro primer lista  . . . . Mientras lista1 no sea vacía obtengo termino  . . . . . Recorro segunda lista  . . . . . Mientras lista2 no sea vacía obtengo termino  . . . . . . Si exponentes1 es mayor o igual a exponente2  . . . . . . . Si base es distinta de 0  . . . . . . . . Sumo ambos exponentes  . . . . . . . . Obtengo bases de ambos términos y multiplico  . . . . . . . . Obtengo signos de ambos términos  . . . . . . . . Si ambos signos son ‘-’ o ambos signos son ‘+’  . . . . . . . . Asigno ‘+’ a variable Signo  . . . . . . . . Sino  . . . . . . . . Asigno ‘-‘ a variable Signo  . . . . . . . . FinSi  . . . . . . . . Creo termino con dichos datos  . . . . . . . . Inserto termino en lista términos1  . . . . . . . FinSi  . . . . . . FinSi  . . . . . FinMientras  . . . . FinMientras  . . . .  . . . . Recorro segunda lista  . . . . Mientras lista2 no sea vacía obtengo termino  . . . . . Recorro primer lista  . . . . . Mientras lista1 no sea vacía obtengo termino  . . . . . . Si exponentes2 es mayor a exponente1  . . . . . . . Si base es distinta de 0  . . . . . . . . Sumo ambos exponentes  . . . . . . . . Obtengo bases de ambos términos y multiplico  . . . . . . . . Obtengo signos de ambos términos  . . . . . . . . Si ambos signos son ‘-’ o ambos signos son ‘+’  . . . . . . . . Asigno ‘+’ a variable Signo  . . . . . . . . Sino  . . . . . . . . Asigno ‘-‘ a variable Signo  . . . . . . . . FinSi  . . . . . . . . Creo termino con dichos datos  . . . . . . . . Inserto termino en lista términos2  . . . . . . . FinSi  . . . . . . FinSi  . . . . . FinMientras  . . . . FinMientras  . . . Sino  . . . MuestroMensaje POLINOMIO\_NO\_EXISTE  . . Sino  . . MuestroMensaje POLINOMIO\_YA\_EXISTE  . Sino  . MuestroMensaje NOMBRE\_INVALIDO  Sino  MuestroMensaje PARAMETROS\_INVALIDOS  Creo polinomio temp1 con lista1 y lo guardo en ABBPolinomio  Creo polinomio temp2 con lista2 y lo guardo en ABBPolinomio  Defino nuevo string igual a “sumar string\_de\_entrada temp1 temp2”  Llamo a procedimiento Sumar pasando el nuevo string  Elimino del ABBPolinomio polinomios temp1 y temp2 |

|  |  |
| --- | --- |
| Evaluar | |
| Entrada: | Lista de String con parámetros cargados |
| Salida: | Valor del polinomio dado el entero de entrada |
| Método: | Creo variable total = 0  Creo variable temp  Obtengo cantidad de elementos la lista de entrada  Si Cantidad de elemento es igual a 3  . Tomo segundo elemento  . Si EsValidoNombre de segundo elemento es OK  . . Si ExistePolinomio de elemento es NO\_EXISTE\_POLINOMIO  . . . Tomo tercer elemento  . . . Si EsValidoNumero es OK  . . . . Tomo lista de terminos  . . . . Obtengo termino  . . . . Mientras que la lista no sea nula  . . . . Si base es diferente a 0  . . . . . Obtengo exponente  . . . . . Defino temp = entrada  . . . . . Si exponente es mayor a 1  . . . . . . Defino variable i igual a exponente  . . . . . . Mientras i sea mayor a 1  . . . . . . . Temp = Temp \* entrada  . . . . . . . Resto 1 a variable i  . . . . . . FinMientras  . . . . . . Obtengo signo  . . . . . . Temp = temp \* base  . . . . . FinSi  . . . . . Sino  . . . . . . Si exponente es igual a 1  . . . . . . Multiplico entrada por base y lo guardo en Temp  . . . . . . FinSI  . . . . . . Sino  . . . . . . . Si exponente = 0  . . . . . . . Temp = a base.  . . . . . . . FinSI  . . . . . . FinSino  . . . . . FinSino  . . . . FinSI  . . . . Si signo = ‘-‘  . . . . Temp = Temp por -1  . . . . Total = Total + Temp  . . . . Tomo siguiente termino  . . . . FinMientras  . . . . Devuelvo valor total  . . . Sino  . . . MuestroMensaje NUMERO\_INVALIDO  . . Sino  . . MuestroMensaje POLINOMIO\_YA\_EXISTE  . Sino  . MuestroMensaje NOMBRE\_INVALIDO  Sino  MuestroMensaje PARAMETROS\_INVALIDOS |

|  |  |
| --- | --- |
| **EsRaiz** | |
| **Entrada:** | Lista de string con parámetros cargados |
| **Salida:** | Enumerado TipoMensaje |
| **Método:** | Obtengo cantidad de elementos de la lista de entrada.  Si Cantidad de elemento es igual a 3  . Tomo segundo elemento  . Si EsValidoNombre de segundo elemento es OK  . . Si ExistePolinomio de elemento es NO\_EXISTE\_POLINOMIO  . . . Tomo tercer elemento  . . . Si EsValidoNumero es OK  . . . . Creo variable Boolean en FALSE  . . . . Si Evaluar de valor ingresado es igual a 0  . . . . Guardo variable en TRUE  . . . . FinSi  . . . . Retorno variable Boolean  . . . .  . . . .  . . . Sino  . . . MuestroMensaje NUMERO\_INVALIDO  . . Sino  . . MuestroMensaje POLINOMIO\_YA\_EXISTE  . Sino  . MuestroMensaje NOMBRE\_INVALIDO  Sino  MuestroMensaje PARAMETROS\_INVALIDOS |

|  |  |
| --- | --- |
| MostrarABBPolinomio | |
| Entrada: | ABB polinomios |
| Salida: | Salida por pantalla |
| Método: | Si el ABB no es vacio  MostrarABBPolinomio ABB hijo izquierdo  Tomo Polinomio del ABB  MostrarPolinomio tomado del ABB  Desplego en pantalla salto de linea  MostrarABBPolinomio ABB hijo derecho  FinSi  Sino  MostrarMensaje LISTA\_VACIA  FinSino |

|  |  |
| --- | --- |
| GuardarPolinomio | |
| Entrada: | Lista de string cargada con parámetros |
| Salida: | Lista de términos del polinomio almacenada como string en el archivo |
| Método: | Defino f como archivo en modo ‘wb’  Defino variable str de tipo string  Tomo segundo valor de la lista de parámetros  Si el valor no es nulo  Si EsValidoNombre es OK  Si ExistePolinomio es TRUE  Obtengo polinomio  Tomo Lista de Terminos  Tomo primer valor de la lista  Mientras que la lista no se nula  Convierto TerminoAString y lo guardo en variable str  Agrego a str “ “  Tomo siguiente valor de la lista  FinMientras  Quito el ultimo carácter “ “ del string  Agrego a la variable str el carácter ‘\0’  GuardarString en archivo f  Sino  MostrarMensaje POLINOMIO\_NO\_EXISTE  Sino  MostrarMensaje NOMBRE\_INVALIDO  Sino  MostrarMensaje “PARAMETROS\_INVALIDOS” |

|  |  |
| --- | --- |
| RecuperarPolinomio | |
| Entrada: | Lista de string cargada con parámetros |
| Salida: | String con comando crear polinomio |
| Método: | Obtengo cantidad de elementos de la lista de entrada.  Si Cantidad de elemento es igual a 3  Si EsValidoNombre es OK  Si ExistePolinomio es TRUE  Tomo tercer valor de la lista  Si EsValidoNombreArchivo  Defino f como archivo en modo ‘rb’  Defino variable comando de tipo string  Cargo en variable comando “crear ”  Agrego a variabe commando el string con el nomde del polinomio + “ “  Leo primer carácter del archivo  Mientras que no sea el final del archivo y que aux no sea ‘\0’  Guardo el carácter en string comando  Tomo siguiente carácter  FinMientras  Guardo en string comando carácter ‘\0’  Parseo el String comando y obtengo lista de Parametros  CrearPolinomio con la lista de parametros  FinSi  Sino  MostrarMensaje “NOMBRE\_ARCHIVO\_INVALIDO”  Sino  MostrarMensaje “POLINOMIO\_NO\_EXISTE”  Sino  MostrarMensaje “NOMBRE\_INVALIDO”  Sino  MostrarMesaje “PARAMETROS\_INVALIDO” |

|  |  |
| --- | --- |
| Salir | |
| Entrada: | Lista de string  ABBPolinomio |
| Salida: | Fin del programa |
| Método: | Tomo primer elemento de la lista  Si no es nulo  Mientras que el siguiente elemento no sea nulo  Defino aux = siguiente elemento  Delete elemento de lista  Lista = aux  Tomo elemento de lista  FinMientras  Delete aux  Sino  Delete elemento  BorrarABB ABBPolinomio |

## Funciones y Procedimiento auxiliares

|  |  |
| --- | --- |
| Guardar\_String | |
| Entrada: | String Referencia a archivo |
| Salida: | String en el archivo |
| Método: | Tomo primer caracter  Mientas que carácter sea diferenta a fin de linea  Escribo carácter en archivo  Tomo siguiente caracter  FinMientras  Escribo carácter fin de linea en archivo |

|  |  |
| --- | --- |
| Levantar\_String | |
| Entrada: | Referencia a String Referencia a archivo |
| Salida: | String en variable |
| Método: | Creo un string auxiliar con tamaño max  Leo el primer caracter  Mientas que carácter sea diferente a fin de línea y no llegue al final del archivo  Tomo siguiente caracter  Leo carácter en archivo  FinMientras  Copio lo del string auxiliar al string a retornar  Elimino string auxiliar |

|  |  |
| --- | --- |
| Parsear | |
| Entrada: | String de comandos |
| Salida: | Lista de Strings |
| Método: | Creo nuevo string  Recorro String de comandos  Mientras caracter sea distinto a fin de línea  Si carácter es distinto a espacio  Cargo String con carácter  Sino  Agrego caracter fin de línea  Guardo nuevo String en lista de String  Creo nuevo string  FinMientras |

|  |  |
| --- | --- |
| ValidarComando | |
| Entrada: | String |
| Salida: | Enumerado TipoMensaje |
| Método: | Defino variable check en COMANDO\_INVALIDO  Tomo primer enumerado.  Mientras haya enumerados de comandos Y valor de check sea COMANDO\_INVALIDO  Si el String de entrada es igual al enumerado de comandos  Seteo OK en variable check  Sino  Tomo el proximo enumerado  FinMientras  Devuelvo check |

|  |  |
| --- | --- |
| EsValidoNombre | |
| Entrada: | String cargado |
| Salida: | Enumerado TipoMensaje |
| Método: | Recorro String  Defino variable check de tipo TipoMensaje = OK  Mientras carácter sea distinto a carácter fin de línea y variable sea OK  Si carácter >= ‘a’ y caracter <= ‘z’  o carácter >= ‘A’ y caracter <= ‘Z’  o carácter >= ‘0’ y caracter <= ‘9’ entonces  Seteo variable check en OK  Sino  Seteo variable check en NOMBRE\_INVALIDO  FinMientras  Devuelvo check |

|  |  |
| --- | --- |
| EsValidoNumero | |
| Entrada: | String cargado |
| Salida: | Enumerado TipoMensaje |
| Método: | Defino variable check de tipo TipoMensaje = OK  Recorro String  Mientras carácter sea distinto a carácter fin de línea y variable check OK  Si carácter es ‘-’ entonces tomo el siguiente caracter  Si carácter >= ‘0’ y carácter <= ‘9’ entonces  Seteo variable check en OK  Sino  Seteo variable check en TERMINO\_INVALIDO  Sino  Si carácter >= ‘0’ y carácter <= ‘9’ entonces  Seteo variable check en OK  Sino  Seteo variable en TERMINO\_INVALIDO  FinMientras  Devuelvo check |

|  |  |
| --- | --- |
| ConvertirStringATermino | |
| Entrada: | String |
| Salida: | Termino |
| Método: | Defino nuevo termino vacio.  Recorro string  Mientras carácter sea distinto a fin de línea  Si carácter es ‘-‘  Guardo signo en Signo  Sino  Guardo signo ‘+’ en Signo  ConvertirCharANumero  Guardo valor en base  FinSi  FinMientras |

|  |  |
| --- | --- |
| CantParametros | |
| Entrada: | Lista de tipo ListaParam cargada |
| Salida: | Int |
| Método: | Declaro variable cant  Recorro lista  Mientras haya ítems en lista  Sumo uno a variable cant  FinMientras |

|  |  |
| --- | --- |
| CantTerminos | |
| Entrada: | Lista de tipo ListaTerm cargada |
| Salida: | Int |
| Método: | Declaro variable cant  Recorro lista  Mientras haya ítems en lista  Sumo uno a variable cant  FinMientras |

|  |  |
| --- | --- |
| ConvertirCharANumero | |
| Entrada: | String cargado |
| Salida: | Int |
| Método: | Creo variables cantItems, contador, resultado, multiplicador en 1, i  Recorro String  Mientras carácter sea distinto a carácter fin de línea  Sumo uno a variable cantItems  FinMientras  Asigno valor de cantItems a variable i  Para cada elemento de String mientras contador sea menor a i  Asigno a resultado, (valor de resultado mas valor de carácter en posicion cantItems menos 1) – 48 por multiplicador  Asigno a multiplicador valor de multiplicador, multiplicado por 10  Resto uno a cantItems  FinParaCada  Retorno variable resultado |

|  |  |
| --- | --- |
| MostrarPolinomio | |
| Entrada: | Polinomio |
| Salida: | Salida por pantalla |
| Método: | Muestro nombre polinomio  Muestro “ = “  Tomo lista de terminos  Tomo primer termino de la lista  Mientras que la lista no sea vacía  MuestroTermino de la lista  Despliego “ “ por pantalla  Tomo el proximo termino de la lista  FinMientras |

|  |  |
| --- | --- |
| MostrarMensaje | |
| Entrada: | Enumerado TipoMensaje |
| Salida: | Mensaje en pantalla |
| Método: | Según sea  Si caso es PARAMETROS\_INVALIDOS entonces  Desplego “No ingreso todos los parametros”  Si caso es NOMBRE\_INVALIDO entonces  Desplego “El nombre ingresado no es valido”  Si caso es NUMERO\_INVALIDO  Desplego “El numero ingresado no es valido”  Si caso es COMANDO\_INVALIDO  Desplego “El comando ingresado no es valido”  Si caso es YA\_EXISTE\_POLINOMIO  Desplego “El polinomio ya existe en memoria”  Si caso es NO\_EXISTE\_POLINOMIO  Desplego “El polinomio solicitado no existe en memoria”  Si caso es NO\_INGRESO\_NOMBRE  Desplego “No ingreso un nombre para el polinomio”  Si caso es NO\_INGRESO\_TERMINO  Desplego “No ingreso termino”  Si caso es LISTA\_VACIA  Desplego “La lista esta vacia”  Si caso es COEF\_IGUAL\_CERO  Desplego “El primero coeficiente no puede ser 0”  FinSegunSea |

|  |  |
| --- | --- |
| BorrarABB | |
| Entrada: | ABBPolinomio |
| Salida: | Memoria ocupada por ABB se libera |
| Método: | Si el ABB no es vacío  BorrarABB ABB hijo izquierdo  BorrarABB ABB hijo derecho  Delete nodo  FinSi |

# Planificación – Gantt

# 

# 