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

Contenido

[Elección de estructuras de datos 2](#_Toc1121902)

[Justificación 2](#_Toc1121903)

[Escritura en C++ y cabezales 3](#_Toc1121904)

[Diagrama de inclusiones 6](#_Toc1121905)

[Seudocódigo 7](#_Toc1121906)

[Comandos del programa 7](#_Toc1121907)

[Funciones y Procedimiento auxiliares 14](#_Toc1121908)

[Planificación – Gantt 19](#_Toc1121909)

[19](#_Toc1121910)

# 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, NO\_INGRESO\_NOMBRE, NO\_INGRESO\_TERMINO} 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 "String.h"

#include "ListaTerminos.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);

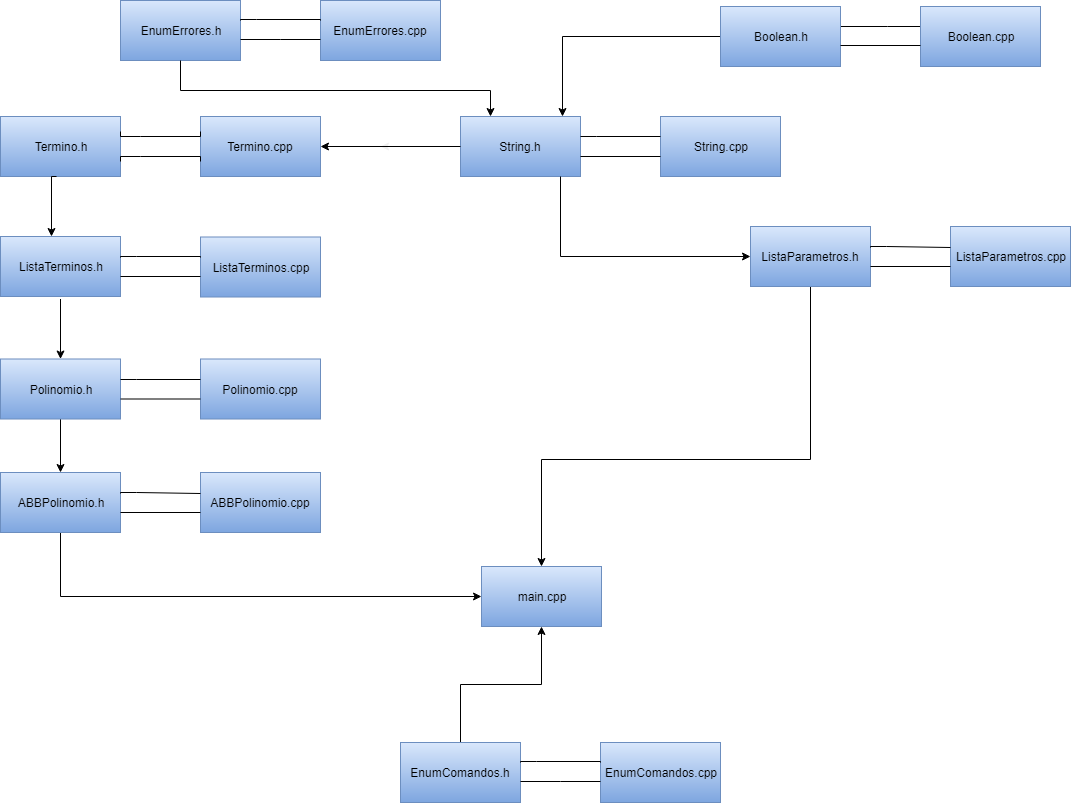
Boolean ExistePolinomio(Arbol &ABBPoli, String nombre\_poli);

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

**EnumComandos.h**

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

# Diagrama de inclusiones



# Seudocódigo

## Comandos del programa

|  |  |
| --- | --- |
| CrearPolinomio | |
| Entrada: | Lista de String cargada parámetros |
| 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 EsValidoNombre es OK  Si  Recorro Lista de String a partir de segundo parámetro  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 numero 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  Sino  MuestroMensaje de nombre invalido  Creo polinomio con nombre y la nueva lista de términos  Agrego polinomio a ABB |

|  |  |
| --- | --- |
| Sumar | |
| Entrada: | Dos polinomios a ser sumados  Polinomio vacío por referencia |
| Salida: |  |
| Método: | Obtengo términos de primer polinomio  Obtengo términos de segundo polinomio  Creo variable Booleana en FALSE  Creo variable j  Obtengo cantidad de elementos de cada lista de términos.  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  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 |

|  |  |
| --- | --- |
| Multiplicar | |
| Entrada: | 2 Polinomios String |
| Salida: | Polinomio cargado |
| Método: | Obtengo lista terminos de 1er polinomio  Obtengo lista términos de 2do polinomio  Creo variable char Signo  Creo Lista de terminos1, Lista de términos 2  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  Llamo a Procedimiento sumar pasando ambas listas de términos  Creo nuevo polinomio con String de nombre y Lista cargada con suma de términos. |

|  |  |
| --- | --- |
| Evaluar | |
| Entrada: | Polinomio Entero para evaluación |
| Salida: | Valor del polinomio dado el entero de entrada |
| Método: | Creo variable total = 0  Creo variable temp  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 por pantalla |

|  |  |
| --- | --- |
| **EsRaiz** | |
| **Entrada:** | Polinomio  Entero para calcular |
| **Salida:** | TRUE, FALSE |
| **Método:** | Creo variable Boolean en FALSE  Si Evaluar de valor ingresado es igual a 0  Guardo variable en TRUE  Retorno variable Boolean |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| GuardarPolinomio | |
| Entrada: | Polinomio String con el nombre del archivo |
| Salida: |  |
| Método: | Defino f como archivo en modo ‘wb’  Defino variable str de tipo string  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 |

|  |  |
| --- | --- |
| RecuperarPolinomio | |
| Entrada: | String con nombre del nuevo polinomio String con el nombre del archivo |
| Salida: | String con comando crear polinomio |
| Método: | 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 |

## Funciones y Procedimiento auxiliares

|  |  |
| --- | --- |
| GuardarString | |
| 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 |

|  |  |
| --- | --- |
| GuardarTermino | |
| Entrada: | Termino Referencia a archivo |
| Salida: | Termino en el archivo |
| Método: | Tomo signo del termino y la guardo en el archivo  Tomo base del termino y la guardo en el archivo  Tomo exponente del termino y lo guardo en el archivo |

|  |  |
| --- | --- |
| IngresarComando | |
| Entrada: | Caracteres ingresados por teclado |
| Salida: | - |
| Método: | Creo un string nulo (strcrear) Cargo lo ingresado en un string dinámico (scan) Parsear el string ingresado |

|  |  |
| --- | --- |
| 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: | TRUE, FALSE |
| Método: | Guardo valor FALSE  Tomo primer enumerado.  Mientras haya enumerados de comandos Y valor guardado sea FALSE  Si el String de entrada es igual al enumerado de comandos  Guardo valor TRUE  Sino  Tomo el proximo enumerado  FinMientras  SI el valor guardado es TRUE  Devuelvo TRUE  Sino  Devuelvo FALSE |

|  |  |
| --- | --- |
| EsValidoNombre | |
| Entrada: | String cargado |
| Salida: | TRUE, FALSE |
| Método: | Recorro String  Mientras carácter sea distinto a carácter fin de línea y variable sea TRUE  Si carácter >= ‘a’ y caracter <= ‘z’  o carácter >= ‘A’ y caracter <= ‘Z’  o carácter >= ‘0’ y caracter <= ‘9’ entonces  Seteo variable en TRUE  Sino  Seteo variable en FALSE  FinMientras |

|  |  |
| --- | --- |
| EsValidoNumero | |
| Entrada: | String cargado |
| Salida: | TRUE, FALSE |
| Método: | Recorro String  Mientras carácter sea distinto a carácter fin de línea y variable sea TRUE  Si carácter es ‘-’ entonces tomo el siguiente caracter  Si carácter >= ‘0’ y carácter <= ‘9’ entonces  Seteo variable en TRUE  Sino  Seteo variable en FALSE  Sino  Si carácter >= ‘0’ y carácter <= ‘9’ entonces  Seteo variable en TRUE  Sino  Seteo variable en FALSE  FinMientras |

|  |  |
| --- | --- |
| 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 vasia  MuestroTermino de la lista  Despliego “ “ por pantalla  Tomo el proximo termino de la lista  FinMientras |

|  |  |
| --- | --- |
| TerminoAString | |
| Entrada: | Termino |
| Salida: | String |
| Método: | // FALTA HACER |

# Planificación – Gantt

# 