Universidad Europea

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

GRADO EN INGENIERIA INFÓRMATICA

ACTIVIDAD COLABORATIVA 2

Programación con estructuras lineares

Ing. Jorge García

EJERCICIOS PRACTICOS DE GESTIÓN DE MEMORIA

Grupo 1:

MIGUEL SÁNCHEZ BERNUY

GERÓNIMO BASSO

JORGE CASTILLA COELLO

VICTOR MARTÍN LORENZO

DIOGO RIBEIRO da SILVA

CURSO 2022-2023

1 RESUMEN

1.1 TABLA RESUMEN

	DATOS
Nombre y apellidos:	MIGUEL SÁNCHEZ BERNUY GERÓNIMO BASSO JORGE CASTILLA COELLO VICTOR MARTÍN LORENZO DIOGO RIBEIRO DA SILVA
Título del proyecto:	Ejercicios prácticos de gestión de memoria
Directores del proyecto:	Jorge García
El proyecto se ha realizado en colaboración de una empresa o a petición de una empresa:	NO
El proyecto ha implementado un producto: (esta entrada se puede marcar junto a la siguiente)	NO
El proyecto ha consistido en el desarrollo de una investigación o innovación: (esta entrada se puede marcar junto a la anterior)	NO
Objetivo general del proyecto:	Demostrar que somos capaces de completar las tareas aplicando aptitudes adquiridas

2 ÍNDICE

3 ÍNDICE DE FIGURAS

4 RESUMEN DEL PROYECTO

4.1 CONTEXTO Y JUSTIFICACIÓN

Este proyecto ha surgido a través de la necesidad de practicar y probar los conocimientos adquiridos a lo largo del aprendizaje de la asignatura

4.2 PLANTEAMIENTO DEL PROBLEMA

Mediante este programa pretendemos demostrar y mejorar nuestras aptitudes tanto grupales como individuales en lo referido a la utilización de las diversas herramientas aprendidas a lo largo del curso

4.3 OBJETIVOS DEL PROYECTO

En resumidas cuentas, nuestros objetivos serían completar las cinco tareas asignadas, más una sexta opcional, mediante lo explicado en clase de la forma más satisfactoria y óptima posible.

5 GUÍA DE USUARIO

5.1 INTRODUCCIÓN A LA GUÍA DE USUARIO

Esta guía de usuario es un breve documento explicativo de las diferentes funciones del código mediante el cual aprenderás a moverte y recorrer el programa y sus diversas acciones de una manera amena y visual.

5.2 EJERCICIO 1

5.2.1 Menú Principal

- Al correr el programa se mostrará por pantalla le menú principal, con 4 posibles opciones cada una con su respectiva función: 1º (Insertar valores en la Pila), 2º(Ver valores almacenados), 3º(Eliminar Elementos), 4º(Salir del programa):

```
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
```

Ilustración 1: Guía 1 ej.1

5.2.2 Insertar valores en la Pila

- Al acceder a la primera opción (Insertar valores en la Pila) nos pedirá que introduzcamos el valor numérico por teclado:

```
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
1
Introduce un numero
```

Ilustración 2: Guía 2 ej.1

Al introducir el número correctamente se almacenará en la Pila y volverá a mostrarnos el menú principal, con el fin de seleccionar de nuevo una de las 4 opciones validas.

```
1
Introduce un numero
1
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
```

Ilustración 3: Guía 3 ej.1

Si el usuario por equivoco introduce un número no número le saltara una excepción por pantalla, confirmando que dicho número no es válido. El menú principal saldrá de nuevo preguntado al usuario que introduzca una de las 4 posibles opciones:

```
1
Introduce un numero
fds
Numero no validado
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
```

Ilustración 4: Guía 4 ej.1

5.2.3 Ver valores almacenados

Al seleccionar la segunda opción (ver valores almacenados), nos mostrara la posición de dicho número junto al número asignado a esa posición:

```
Posición elemento: 1° || Número: 2
Posición elemento: 2° || Número: 1000
Posición elemento: 3° || Número: 125
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
```

Ilustración 5: Guía 5 ej.1

En caso de que no haya ningún elemento introducido en la lista nos mostrara un mensaje por pantalla “No existen elementos para mostrar”:

```
2
No existen elementos para mostrar

----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
```

Ilustración 6: Guía 6 ej.1

5.2.4 Eliminar Elementos

- Al acceder a la opción 3 (Eliminar Elementos), nos saldrá un mensaje por pantalla, el cual preguntará al usuario que introduzca la posición donde se sitúa el elemento que desea eliminar. Dichas posiciones y elementos se pueden observar en la opción 2:
- Elementos actuales almacenados en la lista, verificación de la opción 2:

```
Posición elemento1° || Número: 2
Posición elemento2° || Número: 5
Posición elemento3° || Número: 18
```

Ilustración 7: Guía 7 ej.1

-

```
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
3
Introduce la posición del elemento que deseas eliminar
```

Ilustración 8: Guía 8 ej.1

- Al introducir la posición 3 se borraría el número 18 y se nos mostraría un mensaje por pantalla confirmando al usuario que el número 18 se ha borrado correctamente:

```
Introduce la posición del elemento que deseas eliminar

3
Numero: 18 ha sido eliminado
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
```

Ilustración 9: Guía 9 ej.1

- Comprobación:

```
4- Salir del Programa
2
Posición elemento1º || Número: 2
Posición elemento2º || Número: 5
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
```

Ilustración 10: Guía 10 ej.1

- Al borrar el número almacenado en la posición 1, automáticamente el número almacenado en la posición se movería a la posición 1º:

```
Posición elemento1º || Número: 2
Posición elemento2º || Número: 5
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa

3
Introduce la posición del elemento que deseas eliminar

1
Numero: 2 ha sido eliminado
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
2
Posición elemento1º || Número: 5
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
```

Ilustración 11: Guía 11 ej.1

- Si seleccionamos la opción 3 estando la lista vacía nos saldrá un mensaje advirtiéndonos de que dicha lista no contiene ningún elemento almacenado:

```
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
3
La lista esta vacia

----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
```

Ilustración 12: Guía 12 ej.1

- Si la lista contiene elementos, pero la posición que hemos seleccionado no tiene ningún elemento almacenado, se le mostrara un mensaje por pantalla al usuario advirtiéndonos que esa posición no tiene ningún elemento:

```
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
3
Introduce la posición del elemento que deseas eliminar

2
La posición indicada no pertenece a la lista

----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
```

Ilustración 13: Guía 13 ej.1

5.2.5 Salir del programa

- Al acceder a la posición 4º (Salir del Programa), se nos mostrara un mensaje por pantalla diciendo "FIN DEL PROGRAMA", dicho programa finalizara automáticamente después de seleccionar esta opción, cerrando de esta manera al bucle:

```
----Introduce la opción que deseas realizar----
1- Insertar valores en la Pila
2- Ver valores almacenados
3- Eliminar elementos
4- Salir del Programa
4
FIN DEL PROGRAMA
```

Ilustración 14: Guía 14 ej.1

5.3 EJERCICIO 2

Como menciona la letra planteada en la actividad, este programa se basa en simular una mochila, en el cual vamos a ir almacenando diferentes tipos de datos. Comenzamos el programa mostrando un menú inicial, el cual se ve así:

```
Bienvenido usuario! Elija que quiere hacer:
1. Almacenar objeto en la mochila
2. Ver objetos en la mochila
3. Eliminar objeto de la mochila
4. Finalizar programa
1
```

Ilustración 15: Guía 1 ej.2

En este menu tenemos 4 opciones, como se ve en la imagen. Vamos a hacer un ejemplo mostranado el flujo entero pasasndo por los 4 menus distintos, para dos tipos distintos de datos. El usuario ingresa el valor “1” para entrar al menu de almacenar objeto en la mochila, una vez ingresado vemos la siguiente imagen:

```
¿Que objeto quieres almacenear?
1. Boolean
2. Integer
3. Float
4. Double
5. Char
6. Volver al menu
1
```

Ilustración 16: Guía 2 ej.2

El usuario puede ver todos los tipos nativos que hay dentro de C++, y así elegir cual quiere almacenar. En este caso el usuario ingresa “1” , dado que desea almacenar un tipo de dato boolean. A continuación, el programa muestra lo siguiente:

```
Introduzca el boolean que quiere almacenar(en minusculas porfavor):
true
Boolean almacenado con éxito
```

Ilustración 17: Guía 3 ej.2

El programa pide al usuario que ingrese en formato string, lo quiere que almacene, en este caso solo serán validos los textos, “true” o “false”. Una vez ocurre esto, se muestra un mensaje de “Boolean almacenado con éxito”. Luego, automáticamente volvemos al menú donde estábamos antes. Y mostramos lo siguiente:

```
¿Que objeto quieres almacenear?
1. Boolean
2. Integer
3. Float
4. Double
5. Char
6. Volver al menu
2
```

Ilustración 18: Guía 4 ej.2

Le preguntamos al usuario si va a almacenar otro objeto, en este caso el usuario introduce “2” para almacenar un tipo de dato Integer, a lo que el programa muestra:

```
Introduzca el Integer que quiere almacenar:
99
Integer almacenado con éxito
```

Ilustración 19: Guía 5 ej.2

Pidiendo el tipo de dato Integer, el usuario ingresa “99”, si se cumplen todas las validaciones correctas dentro del backend, se muestra el mensaje “Integer almacenado con éxito”. Luego se vuelve al menú de antes, mostrando:

```
¿Que objeto quieres almacenear?
1. Boolean
2. Integer
3. Float
4. Double
5. Char
6. Volver al menu
6
```

Ilustración 20: Guía 6 ej.2

El usuario ingresa un “6” para volver al menú principal, donde tiene las 4 opciones generales que vimos antes, el programa muestra lo siguiente:

```
Bienvenido usuario! Elija que quiere hacer:
1. Almacenar objeto en la mochila
2. Ver objetos en la mochila
3. Eliminar objeto de la mochila
4. Finalizar programa
2
```

Ilustración 21: Guía 7 ej.2

El usuario selecciona “2” para ver todos los objetos que hay dentro de la mochila. A lo que el programa responde:

```
------- Boolean ---------
1
------- Integer ---------
99
------- Float ---------
------- Double ---------
------- Char ---------
```

Ilustración 22: Guía 8 ej.2

Se imprime por pantalla un listado de los objetos que hay dentro dentro de la mochila, en boolean podemos ver el "1" significando "true" que agregamos al principio, y el Integer podemos ver el "99" que fue el dato que ingresamos segundo. En Float, Double y Char no vemos nada dado que no se agregó ningún objeto de ese tipo todavía. A continuación, el programa vuelve al menú principal:

```
Bienvenido usuario! Elija que quiere hacer:
1. Almacenar objeto en la mochila
2. Ver objetos en la mochila
3. Eliminar objeto de la mochila
4. Finalizar programa
3
```

Ilustración 23: Guía 9 ej.2

En este caso vamos a probar el flujo de eliminación de objetos dentro de la mochila, por lo cual el usuario selecciona "3", vemos lo siguiente:

```
¿Que dato quieres borrar?
1. Boolean
2. Integer
3. Float
4. Double
5. Char
6. Volver al menu
1
```

Ilustración 24: Guía 10 ej.2

Desplegamos un menú con las opciones dadas sobre borrar cada tipo de dato nativo de C++. En este caso el usuario va a borrar el boolean, ingresamos "1", a lo que vamos a mostrar:

```
Introduzca la posición del elemento que quiere borrar (recuerde que compienza desde 0):
0
```

Ilustración 25: Guía 11 ej.2

El usuario introduce "0" dado que el boolean que quiere borrar fue el primer dato ingresado dentro del vector que almacena los booleans, recordar que cada vector almacena cada tipo de dato distinto, por lo tanto, cada vector de cada tipo de dato comienza desde 0. Se vuelve al menú principal y se muestra:

```
¿Que dato quieres borrar?
1. Boolean
2. Integer
3. Float
4. Double
5. Char
6. Volver al menu
2
```

Ilustración 26: Guía 12 ej.2

Volvemos al menú de borrado y en este caso vamos a seleccionar la opción “2” para borrar el Integer que habíamos almacenado antes, se muestra:

```
Introduzca la posición del elemento que quiere borrar (recuerde que compienza desde 0):
0
```

Ilustración 27: Guía 13 ej.2

A lo que el usuario escribe “0” dado que es la primera posición del vector de integers, se borra y el programa muestra:

```
¿Que dato quieres borrar?
1. Boolean
2. Integer
3. Float
4. Double
5. Char
6. Volver al menu
6
```

Ilustración 28: Guía 14 ej.2

Seleccionamos la opción “6” para volver al menú principal, el programa refleja lo siguiente:

```
Bienvenido usuario! Elija que quiere hacer:
1. Almacenar objeto en la mochila
2. Ver objetos en la mochila
3. Eliminar objeto de la mochila
4. Finalizar programa
2
```

Ilustración 29: Guía 15 ej.2

Seleccionamos la opción “2” para ver que se hayan borrado correctamente los elementos que borramos en los pasos anteriores, se muestran:

```
------- Boolean ---------
------- Integer ---------
------- Float ---------
------- Double ---------
------- Char ---------
```

Ilustración 30: Guía 16 ej.2

Por lo cual podemos ver que se borraron los elementos con éxito, y ya no se encuentran dentro de la mochila. Volvemos automaticamente al menu principal, mostramos:

```
Bienvenido usuario! Elija que quiere hacer:
1. Almacenar objeto en la mochila
2. Ver objetos en la mochila
3. Eliminar objeto de la mochila
4. Finalizar programa
4
```

Ilustración 31: Guía 17 ej.2

El usuario selecciona “4” para asi terminar la ejecucción del programa.

5.4 EJERCICIO 3

Una vez inicializado el programa, esta muestra un sencillo menú donde se listan todas las opciones disponibles.

```
Selecione una opcion:
1 Crear Matriz,
2 Sumar Matrices,
3 restar matrices,
4 multiplicar matriz,
5 matriz traspuesta,
6 determinante de la matriz(solo en los casos 1x1,2x2 y3x3),
7 salir
```

Ilustración 32: Guía 1 ej.3

La primera opción, para crear matrices, lo primero nos pregunta el número de filas y de columnas que desea para la nueva matriz, junto con los valores que debe de contener, una vez creada la matriz, nos muestra una representación del resultado

```
indique las filas
2
indique las columnas
2
introduzca el elemento: 1x1
0
introduzca el elemento: 1x2
1
introduzca el elemento: 2x1
2
introduzca el elemento: 2x2
3
 0  1
 2  3
```

Ilustración 33: Guía 2 ej.3

La opción de sumar matrices, (Solo disponible una vez que se hayan creado al menos dos matrices), nos muestra una lista de las matrices existentes y nos solicita que indiquemos que dos matrices queremos sumar, una vez realizado nos muestra la matriz resultada que se añadirá de forma automática a la lista de matrices

```
2
Matriz 0
 0  1
 2  3
Matriz 1
 0  2
 1  3
indique la matriz a sumar 1
0
indique la matriz a sumar 2
1
 0  3
 3  6
```

Ilustración 34: Guía 3 ej.3

El menú de restar matrices es igual que el de la suma únicamente que realiza la operación contraria

```
3
Matriz 0
 0  1
 2  3
Matriz 1
 0  2
 1  3
Matriz 2
 0  3
 3  6
indique la matriz a restar 1
0
indique la matriz a restar 2
2
 0  -2
 -1  -3
```

Ilustración 35: Guía 4 ej.3

El menú de multiplicar matrices es igual que el de la suma y la resta, comportándose de la misma manera

```
4
Matriz 0
 0  1
 2  3
Matriz 1
 0  2
 1  3
Matriz 2
 0  3
 3  6
Matriz 3
 0  -2
 -1  -3
indique la matriz a multiplicar 1
3
indique la matriz a multiplicar 2
2
 -6  -12
 -9  -21
```

Ilustración 36: Guía 5 ej.3

El menú de transponer es muy sencillo, dado que solo es necesario una matriz para obtener la transpuesta, actúa como los menús anteriores, mostrando al final del proceso la matriz resultado antes de añadirla a la lista de matrices disponibles para trabajar.

```
5
Matriz 0
 0  1
 2  3
indique la matriz a trasponer
```

Ilustración 37: Guía 6 ej.3

El menú para obtener el determinante de una matriz, por razones matemáticas, solo disponible para matrices cuadradas, y por limitaciones de programación, únicamente los más comunes de 1X1, 2X2, 3X3, nos pide que indiquemos de que matriz deseamos obtener el determinante, y posteriormente nos muestra el determinante por consola, esta operación no genera una matriz adicional en la lista

```
6
Matriz 0
 0  1
 2  3
Matriz 1
 0  2
 1  3
Matriz 2
 0  3
 3  6
Matriz 3
 0  -2
 -1  -3
Matriz 4
 -6  -12
 -9  -21
indique la matriz de la que quiere calcular el determinante
4
El determinante es: 18
```

Ilustración 38: Guía 7 ej.3

Una vez que hemos terminado con la aplicación, e indicamos que deseamos cerrar, se llama al destructor de cada matriz creada y del array que las contiene para liberar la memoria del free store.

```
7
Matriz destruida con exito
Matriz destruida con exito
Matriz destruida con exito
Matriz destruida con exito
Matriz destruida con exito

Process finished with exit code 0
```

Ilustración 39: Guía 8 ej.3

5.5 EJERCICIO 4

Al iniciar el programa se mostrará un menú con tres opciones las cuales son introducir un usuario, eliminar un usuario y finalizar el programa.

```
1. Dar de alta un usuario
2. Dar de baja un usuario
3. Finalizar programa
```

Ilustración 40: Guía 1 ej.4

1. Dar de alta un usuario.

Al elegir está opción se mostrará en el menú que por favor se introduzca tanto el nombre como el apellido de dicho usuario, y automáticamente se generará un código de identificación.

```
1
Introduzca el nombre del usuario:
Alejandro
Introduzca el apellido del usuario:
Morales
-----Bienvenido a nuestro sistema de altas y bajas de usuario-----
```

Ilustración 41: Guía 2 ej.4

2.Dar de baja un usuario

Al elegir está opción podrás eliminar un usuario del programa sin necesidad de perder el código de identificación del usuario, y al crear un usuario nuevo, este tomará el código del usuario anterior.

El usuario se elimina mediante el código de identificación.

```
2
----Usuarios----
0 Alejandro Morales
Indique el codigo del usuario:
0
```

Ilustración 42: Guía 3 ej.4

Ahora el usuario está borrado.

```
1
Introduzca el nombre del usuario:
Manuel
Introduzca el apellido del usuario:
Perez
```

Ilustración 43: Guía 4 ej.4

Ahora este nuevo usuario tiene el código de identificación del último usuario destruido, en este caso el código de identificación es el 0.

```
----Usuarios----
0 Manuel Perez
```

Ilustración 44: Guía 5 ej.4

Si en el programa no hay ningún usuario introducido este te dirá que debe de haber al menos algún usuario que se pueda eliminar.

```
1. Dar de alta un usuario
2. Dar de baja un usuario
3. Finalizar programa
2
Deben de existir al menos un usuario
```

Ilustración 45: Guía 6 ej.4

3. Finalizar programa

El programa no se finalizará hasta que esta opción sea seleccionada, por lo que te seguirá pidiendo que des de alta o de baja a usuarios hasta que selecciones esta opción.

```
1. Dar de alta un usuario
2. Dar de baja un usuario
3. Finalizar programa
3

Process finished with exit code 0
```

Ilustración 46: Guía 7 ej.4

5.6 EJERCICIO 5

Tras iniciar el programa el mismo generará automáticamente un número aleatorio oculto el cual el usuario deberá adivinar a continuación.

Lo primero que se mostrará por pantalla es un breve texto introductor al juego y acto seguido preguntará el número el cual el usuario piensa que podría ser.

```
------Bienvenido a Más o Menos, donde tendras que adivinar un número random entre el 1 y el 100 para ganar el juego !--------
Qué número piensas que es?
```

Ilustración 47: Guía 1 ej.5

A continuación, el usuario escribirá un primer número a modo de prueba el cual puede representar tres escenarios distintos:

- **El numero introducido es menor**: Se imprimirá por pantalla, haciendo referencia a que el número introducido es demasiado pequeño respecto al generado por el programa y volverá a preguntar el número al usuario.

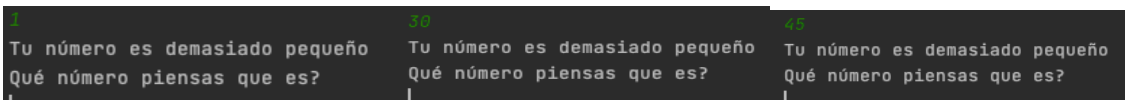

Ilustración 48: Guía 2,3,4 ej.5

- **El numero introducido es mayor:** Se imprimirá por pantalla, haciendo referencia a que el número introducido es demasiado grande respecto al generado por el programa y volverá a preguntar el número al usuario.

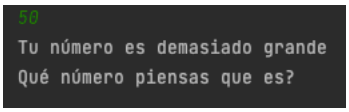

Ilustración 49: Guía 5 ej.5

- **El numero introducido es correcto:** Se imprimirá por pantalla una enhorabuena, dando por hecho que el juego ha finalizado. Además, añadirá un contador con el número de intentos requeridos.

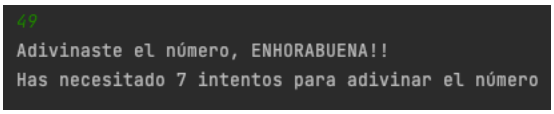

Ilustración 50: Guía 6 ej.5

Por último, existe la opción de que el usuario no comprenda que se significa el término número por lo tanto en caso de que ingrese una letra o carácter no numérico el programa corregirá el error imprimiendo por pantalla un mensaje y preguntando el número de nuevo.

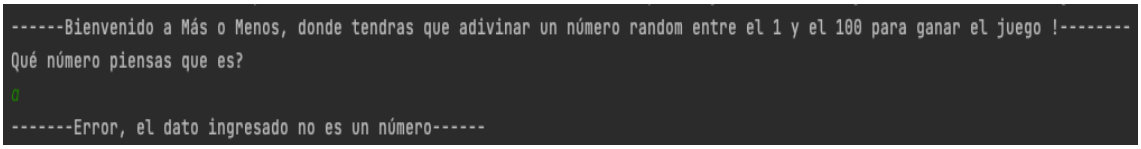

Ilustración 51: Guía 7 ej.5

5.7 EJERCICIO 6

5.7.1 CONTEXTUALIZACIÓN

Este ejercicio consta de crear un sistema de matriculación en el cual tanto alumnos como profesores tengan diversidad de opciones al respecto. Desde crear los respectivos usuarios para inicializar el programa hasta crear asignaturas y asignar diferentes usuarios a las mismas.

Tratamos de facilitar el proceso de matriculación realizando este programa con el objetivo de modernizar el sistema actual de matriculación.

Para lograr dicho objetivo hemos decidido realizar un programa el cual permita a profesores agregar asignaturas y alumnos a las mismas. También podrá revisar que usuarios están

introducidos en las diferentes materias, y a la inversa, ver en que asignaturas está matriculado un alumno. Además, serán capaces de eliminar todo tipo de datos como por ejemplo asignaturas, alumnos y profesores.

5.7.2 GUIA DE USUARIO

Comenzamos el programa mostrando por terminal un menú principal, donde tenemos nueve opciones. Estas opciones las usamos para completar todo lo que nos planteamos en la letra del problema. Como mencionamos, en las siguientes capturas mostraremos el menú principal, haciendo un recorrido atreves de todas las diferentes opciones para ver el flujo entero del programa. El usuario ingresa “1” para crear a un profesor, el programa muestra lo siguiente:

```
----- Bienvenido a nuestro sistema de matriculacion de alumnos! -------
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
1
Indique el nombre del profesor:
Mark
Indique el apellido del profesor:
Cuban
Indique el codigo de empleado del profesor:
111
```

Ilustración 52: Guía 1 ej.6

El usuario ingresa el nombre del profesor, junto con su apellido y su código de empleado, una vez ingresado esto, se vuelve al flujo del programa general. Lo cual el programa muestra:

```
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
2
Indica el nombre del alumno:
Jorge
Indique el apellido del alumno:
Castilla
Indique el numero de expediente:
222
```

Ilustración 53: Guía 2 ej.6

Luego, el usuario ingresa “2” para continuar con crear un alumno, como en el paso anterior ingresamos un nombre, apellido y un número de expediente, luego de esto, volvemos al menú principal y mostramos lo siguiente:

```
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
3
----Profesores----
0-111-Mark-Cuban
Indique la posicon dentro del vector del profesor que dara la asignatura:
0
Indique el nombre de la asignatura:
Base de datos
Indique el codigo de la asignatura:
333
Indique el anio de la asignatura:
2000
Se creo la asignatura con exito !
333 Base de datos 2000
------------------------------------------------------------
111-Mark-Cuban
------------------------------------------------------------
```

Ilustración 54: Guía 3 ej.6

Volvemos al menú principal, y el usuario ingresa el dato “3” para crear una asignatura. El programa se maneja usando posiciones de los vectores. Por lo tanto, pedimos la posición de la asignatura, luego el nombre de la asignatura, el código de la asignatura y el año de la asignatura. Volvemos al menú principal y vemos:

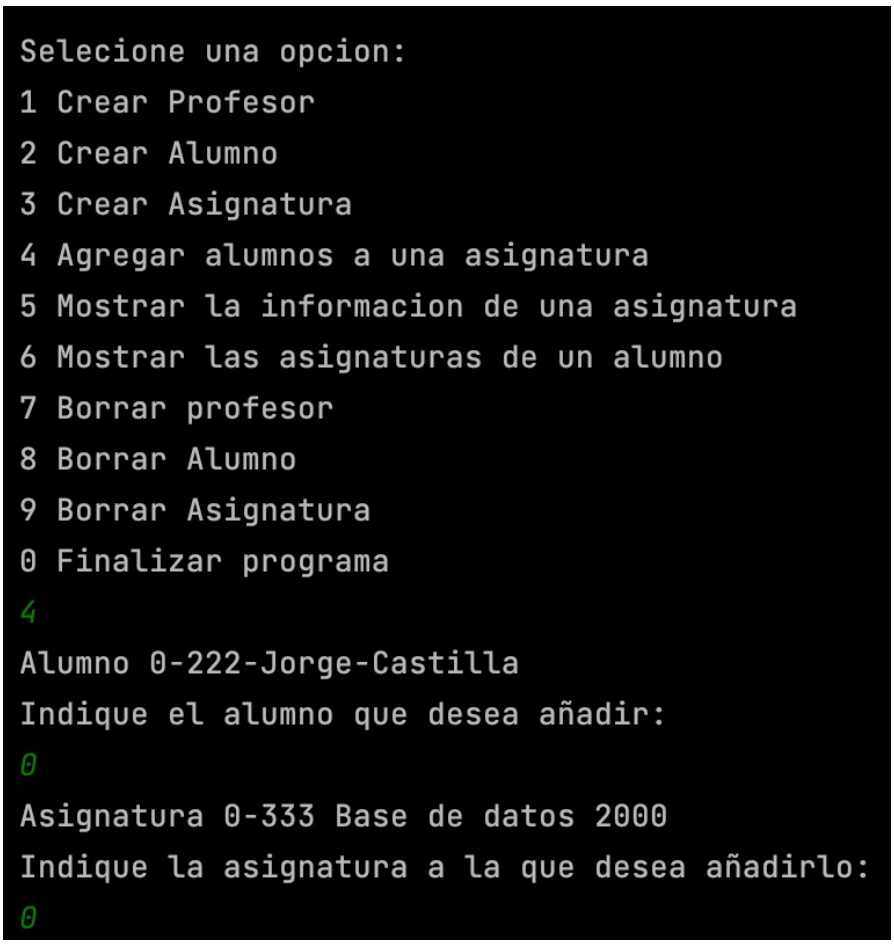

Ilustración 55: Guía 4 ej.6

El usuario ingresa un “4” para agregar una asignatura, esta materia asocia a un alumno, pasamos la posición del vector del alumno que vamos a inscribir en la materia, y luego pasamos la asignatura a la que se va a añadir. Después de esto volvemos al menú principal y mostramos:

```
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
5
Asignatura: 0-333 Base de datos 2000
indique la asignatura de la que desea ver la informacion
0
333 Base de datos 2000
------------------------------------------------------------
111-Mark-Cuban
------------------------------------------------------------
222-Jorge-Castilla
```

Ilustración 56: Guía 5 ej.6

El usuario ingresa “5” para poder mostrar la información dentro de una asignatura, a lo que pedimos la posición de la asignatura para mostrar todo lo que está dentro de ella. Volvemos al menú principal y mostramos:

```
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
6
Alumno: 0-222-Jorge-Castilla
indique el alumno del que desea ver las asignaturas
0
333 Base de datos 2000
```

Ilustración 57: Guía 6 ej.6

El usuario ingresa “6” para el cual se van a mostrar las asignaturas de un alumno, pedimos la posición para poder elegir el alumno que queremos mostrar. Volvemos al menú, y desplegamos:

```
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
7
Profesor 0-111-Mark-Cuban
Indique el profesor que desea borrar:
0
Profesor eliminado con exito
```

Ilustración 58: Guía 7 ej.6

El usuario ingresa “7”, para borrar un profesor, se pide posición del vector profesor para poder borrarlo.

```
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
8
Alumno 0-222-Jorge-Castilla
Indique el alumno que desea borrar:
0
Alumno eliminado con exito
```

Ilustración 59: Guía 8 ej.6

El usuario ingresa “8”, para borrar un alumno, se pide posición del vector alumno para poder borrarlo.

```
Alumno eliminado con exito
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
9
Asignatura 0-333 Base de datos 2000
Indique la asignatura que desea borrar:
0
Asignatura eliminada con exito
```

Ilustración 60: Guía 9 ej.6

El usuario ingresa “9”, para borrar una asignatura, se pide posición del vector asignatura para poder borrarlo.

```
Asignatura eliminada con exito
Selecione una opcion:
1 Crear Profesor
2 Crear Alumno
3 Crear Asignatura
4 Agregar alumnos a una asignatura
5 Mostrar la informacion de una asignatura
6 Mostrar las asignaturas de un alumno
7 Borrar profesor
8 Borrar Alumno
9 Borrar Asignatura
0 Finalizar programa
0
```

Ilustración 61: Guía 10 ej.6

Por último, el usuario ingresa un “0” para lo cual se termina con la ejecución del programa.

6 OBJETIVOS

6.1 OBJETIVOS GENERALES

El objetivo general de dicho proyecto es completar adecuadamente la entrega de las seis tareas.

6.2 OBJETIVOS ESPECÍFICOS

Tarea 1: Implementar en C++ un programa que sea capaz de simular la pila de usuario.

Tarea 2: Implementar en C++ un programa el cual simule una mochila, inventario o almacén de objetos.

Tarea 3: Implementar una calculadora de matrices en C++.

Tarea 4: Implementar un sistema de altas y bajas sencillo en C++.

Tarea 5: Realizar un juego para el usuario inspirado en el Frio/Calor hasta adivinar un número.

Tarea 6: Realizar un sistema de matriculación escolar completamente funcional en C++.

6.3 BENEFICIOS DEL PROYECTO

Afianzar términos, conceptos y técnicas aprendidos a lo largo del curso. Además de servir como repaso y practica de la parte práctica de la asignatura.
Por último, facilitar la creación de otros posibles códigos mediante la reutilización de los ejercicios propuestos.

7 CLASES Y MÉTODOS

7.1 EJERCICIO 1

7.1.1 Clase Pila

Témplate de una clase Pila con sus respectivos métodos, objetos y variables:

Variables:

- 1º Variable: Esta variable es un array de tipo “int” con 10 posiciones, siendo esta una variable privada.
- 2º Variable: Esta variable es un “int” llamada elementos, siendo esta una variable privada.

Métodos:

1. Método (insertNumero):

Método de tipo void con un int como parámetro, este método verificara si todos los elementos del array "listaElmentos" están ocupados, en caso de que no estén el programa asignara el numero pasado por parámetro a la última posición de la lista y sumara un elemento más, perteneciente a la lista. En caso contrario se mostrará por pantalla dicho mensaje "Error: Pilla llena".

2. Método (definirElementos):
   Método de tipo int, en este método declaramos una variable de tipo int "numero".
   Abrimos un try catch y pedimos que el usuario pase un numero por teclado asignado a la variable "numero". En caso de que el dato declarado por el usuario no sea de tipo int saltara la excepción y se mostrara. Por último, dentro del try existe un método que verifica si el numero introducido es 0, en caso de que sea 0 se mostrara un mensaje por teclado "el numero 0 no se almacenado".

3. Método (void showNumeros):
   Método de tipo void, este método comprueba al principio si existen elementos en la lista
   en caso de que no haya ningún elemento en la lista se mostrara dicho mensaje por pantalla "NO EXISTEN ELEMENTOS PARA MOSTRAR". En caso contrario el programa recoge la lista mediante un for que tendrá como límite establecido el número de elementos almacenado en la variable elementos, al recoger el array mostraremos la posición del elemento y su respectivo valor.

4. Método (eliminar): Método tipo void, solo se ejecutará sus funciones si existen elementos en la lista. En este método creamos una variable de tipo int pos (posición), pediremos al usuario que introduzca un dato tipo int que verificara el elemento de la posición. Se abrirá un try catch que verificara que el elemento introducido es de tipo número. En caso de que el número se haya introducido correctamente y de que la posición existe en la pila, se recogerá un for con su límite establecido por el número de elementos actuales de la lista. Con este for recogeremos la lista y eliminaremos el elemento en la posición que hemos introducido por teclado, asignaremos la lista con la posición borrada a la lista con todas las posiciones, y disminuiremos la variable int elementos por 1 valor.

7.1.2 Clase main:

Método main de tipo int, en este método determinaremos toda la ejecución del programa.

Variables:

- Variables 1º: variable de tipo int "num" se usará en el caso 1 para verificar si el numero obtenido es = 0
- Variables 2º: variable de tipo int "caso" se usará para determinar el case ejecutado en el switch
- Variables 3º: variable de tipo boolean "continuar, se usará para controlar el while que determina el tiempo de vida de programa

- Variables 4ª: Variable de tipo Pila, referenciando a clase pila y a sus respectivos métodos

Método While: Determinara la vida del programa

- Comentarios con una breve descripción de cada caso del switch
- metodos para borrar la cache de los datos pedidos atraves del metodo cin. (cin.clear()) y (cin.sync())

- pedimos al usuario que introduzca una de las opciones mostradas anteriormente por pantalla y la asginamos a la variable caso

- **Switch** que gestionara los 5 posibles casos:

1. Case 1: Asignaremos la variable num al metodo definirElementos, en caso de que lo que nos retorne el método (definirElementos) perteneciente a la clase Pila sea distinto de 0, insertaremos dicho elemento en la lista
   a través del método insertNumero.

2. Case 2: Se iniciará el método showNumeros, se mostrarán por pantalla todos los elementos pertenecientes a la lista

3. Case 3: Se ejecutará el método para eliminar un número especifico de la lista determinado por su posición.

4. Case 4: En este caso se mostrará un mensaje por pantalla "FIN DEL PROGRAMA" y la variable continuar se igualaría a false terminado así en bucle while y la ejecución del programa

5. Default: Este caso se ejecutará si la variable num caso no corresponde a ninguno de los 4 casos disponibles, se mostrará un mensaje por pantalla "el caso que a seleccionado no existe"

7.2 EJERCICIO 2

7.2.1 Clase Main:

Todo el código de este ejercicio se encuentra dentro del archivo “main.cpp”, por lo tanto, dentro de este archivo tenemos varios métodos auxiliarles definidos, junto con una plantilla y la función main para ejecutar el programa.

- Funciones auxiliraes:
  - int checkBool(string element){}
  - bool checkInteger(string element){}
  - bool checkFloat(string element){}

- bool checkDouble(string element){}
- bool checkChar(string element){}

Estos cinco métodos realizan tareas muy similares, lo que hacen es chequear que dado un string de argumento, verificamos el tipo de dato que es. Por ejemplo, si en el programa estoy almacenando datos de tipo Integer, y le pido al usuario que me pase un Integer para poder almacenarlo, se le pasa a este argumento a la función checkInteger() para comprobar realmente si es un Integer, esto funciona igual para el resto de las 4 funciones. Asi me aseguro que puedo guardar los datos en los vectores de forma segura sin levantar ninguna excepción.

- Plantilla:
  - template <class T>
    void impimirVector(vector<T> lista) {}

Esta plantilla lo que hace es para cualquier tipo de vector que me pasen, sea un vector que tiene almacenado integers, strings o floats, agarro este vector, y imprimo el contenido que tiene dentro por consola.

- Int main()

Por último, en el método int main(), tenemos la mayor parte de código. Se empieza por un while donde pedimos que nos ingresen la opción del menú principal, luego dependiendo de esa opción podemos entrar o no a otro tipo de menú que tengan otro while, como son los de almacenar objeto y eliminar objeto. En estos menus, tenemos varias opciones para poder cubrir todos los tipos de datos. Por último, tenemos dos opciones más las cuales son imprimir lo que hay dentro de la mochila, y finalizar el programa. Estos cuatro casos principales son los que conforman el main, con dos sub menus más dentro de los casos de almacenar objeto en la mochila y en eliminar objeto de la mochila. Estos dos submenús me permiten elegir el tipo de dato a almacenar o eliminar, para así hacer un ciclo completo.

7.3 EJERCICIO 3

7.3.1 Clase Matriz

La Clase matriz representa las matrices matemáticas con las que opera el usuario, en ella hay presentes los siguientes atributos:

- Filas: el número de filas de la matriz. Tiene un tipo int.
- Columnas: el número de columnas de la matriz. Tiene un tipo int.
- Datos: donde se almacenan los valores de la matriz. Es tipo vector<>

Existe el siguiente método privado, que dada la fila y la columna devuelve el índice del vector unidimensional.

Los siguientes métodos son todos públicos:

- Matriz(int filas, int columnas, int modo): El constructor que se encarga de crear una matriz aleatoria o vacía según el modo.

- ~Matriz(): Destructor de la matriz
- generarMtrizAleatoria(int filas, int columnas): llena la matriz con números aleatorios
- getFilas(): devuelve el número de filas
- getColumnas(): devuelve el número de columnas
- imprimirMatriz(): imprime la matriz por consola
- modificarElemento(): modifica un valor de la matriz dada su posición
- verElemento(): permite visualizar un elemento de la matriz dada su posición
- imprimirMatrizSecuencial(): este método imprime la matriz con unas elegantes elipses rodeando la posición indicada (NO SE USA EN EL PROGRAMA)

7.3.2 Clase operacionesMatriciales

La clase operacionesMatriciales incluye los métodos para realizar las operaciones con las matrices, pero no contiene ningún atributo, sus métodos, todos públicos, son los siguientes:

- ~operacionesMatriciales(): Destructor de la matriz.
- sumarMatrices(): dados los punteros a dos matrices realiza la suma, añade una matriz al array y devuelve su puntero.
- restarMatrices(): dados los punteros a dos matrices realiza la resta, añade una matriz al array y devuelve su puntero.
- transponerMatriz(): dado el puntero de una matriz realiza la transposición, añade una matriz al array y devuelve su puntero.
- multiplicarMatriz(): dados los punteros a dos matrices realiza la multiplicación, añade una matriz al array y devuelve su puntero.
- determinante(): dado el puntero a una matriz, calcula según regla directa, diagonales o formula de Sarrus el determinante de la matriz indicada y lo devuelve e imprime por consola.

7.3.3 Clase menu

La clase menu contiene el menu de la aplicación junto con el vector en el free store donde se han de guardar las matrices durante el tiempo de ejecución, contiene los siguientes atributos:

- seleccion: se utiliza en el switch case del menu, toma la entrada del usuario y lo dirige a la funcionalidad adecuada. Es un char.
- banderaSalida: se utiliza para romper el while y salir del menu. Es de tipo int.
- matrices: contiene los punteros de todas las matrices en el free store durante el tiempo de ejecución. Es de tipo vector<>*

El único método público que posee es el método menu(), que actua como constructor y menu, le muestra al usuario y le permite navegar por el menu con comandos de texto, utiliza un while y un switch case, cuando sale utilizando la opción 7, se encarga de borrar todas las matrices creadas durante la ejecución y borra también el vector que las contiene

7.4 Ejercicio 4:

Bibliotecas usadas

<lostream>: biblioteca generica.

<Vector>: introduce el uso de vectores en el sistema

7.4.1 Clase Usuario

- Variables utilizadas:
- Nombre: dato de tipo string el cual almacena el nombre del usuario y es un dato publico
- Apellido: dato de tipo string el cual almacena el apellido del usuario y es un dato publico
- Identificador: dato de tipo entero el cual almacena el numero de identificación del usuario y en caso del usuario ser borrado el identificador se mantiene para el siguiente usuario a introducir.

7.4.2 Clase sistema

- Variables utilizadas:
- vec: es un vector de tipo usuario el cual almacena todos los datos de los usuarios introducidos es un dato de tipo publico.

7.4.3 Clase main

La clase main es de tipo int .

Variables:

- Condición: de tipo boolean sirve para que el programa se pueda seguir usando tras introducir o destruir un usuario.
- OpcionMenu: de tipo char sirve para introducir las opciones que tiene el programa dentro.
- Usuarios: de tipo vector, es donde se almacenan los datos de los usuarios introducidos.

7.5 Ejercicio 5:

7.5.1 Clase main

- **Bibliotecas utilizadas:**
  - <stdlib.h> <time.h>: requeridas posteriormente para la generación del número aleatorio en C++.
  - <iostream>: biblioteca genérica.
- **Variables utilizadas:**
  - Int numero: en esta variable generamos el número aleatorio mediante el uso de “rand”.
  - Int intentos: variable utilizada para albergar los intentos requeridos para adivinar el número.
  - String datoingresado: esta variable guarda el valor ingresado en formato string
- **Método while:** Es aquí donde empieza la parte principal del código.
  - **Bloque try catc**h: Realizamos el bloque try catch el cual lee datoingresado,

- Si el string es un carácter numérico el programa prosigue con normalidad.
- Si el string contiene caracteres no numéricos existen dos opciones. El carácter no numérico es anterior a los numéricos entonces el try catch actúa de forma que emite un aviso de error y prosigue el programa con normalidad. Si el carácter no numérico es posterior a los numéricos simplemente se obvia todo lo numérico posterior al carácter no numérico.

- **Bloque if else:** Este bloque está dividido en 3 comparaciones distintas:
  - if (stoi(datoIngresado) > numero): Este primer caso apela a cuando el dato ingresado es mayor que el número generado por lo tanto el programa emitirá el correspondiente mensaje debido al exceso y añadirá +1 al contador de intentos.
  - else if (stoi(datoIngresado) < numero): Esta segunda sentencia hace referencia al caso en el cual el dato ingresado es menor que el número a adivinar, el programa escribirá el respectivo mensaje y acto seguido añadirá +1 al contador de intentos.
  - else: Hace referencia a cuando el usuario acierta el número aleatorio. El programa procede a emitir un mensaje alentador de enhorabuena, además, de añadir +1 al contador de intentos y escribir por pantalla el número requerido de los mismos.

7.6 EJERCICIO 6

7.6.1 Clase Alumno:

En esta clase tenemos tres atributos privados, que son los siguentes:

- nombre: El nombre del alumno, de tipo string
- apellido: el apellido del alumno, de tipo string
- codigoAlumno: un código numérico que se utiliza como ID único, de tipo int

Por su parte, esta clase cuenta también con varios métodos públicos, que procedo a detallar:

- Alumno(): constructor vacío de la clase, no devuelve nada porque es un constructor.
- Alumno(string, nombre, string apellido, int codigoAlumno); constructor de la clase Alumno, utilizado en la implementación
- ~Alumno(): Destructor de la clase, imprime un texto por consola para confirmar la destrucción con éxito.
- imprimirAlumno(): devuelve un string con la información del alumno
- getCodigoAlumno(): devuelve el codigo ID del alumno

7.6.2 Clase Profesor:

Esta clase, similar a la de Alumno, contiene 3 atributos privados equivalentes:

- nombre: El nombre del profesor, de tipo string
- apellido: el apellido del profesor, de tipo string
- codigoEmpleado, un número que se utiliza como ID del profesor, de tipo int

Esta clase, cuenta también con cuatro atributos públicos, los cuales son los siguientes:

- Profesor(): constructor por defecto de la clase Profesor, inicializa a unos valores por defecto.
- Profesor(string nombres, string apellido, int codigoEmpleado): constructor con atributos de la clase, utilizado en el programa
- ~Profesor(): Destructor de la clase, imprime un texto por consola para confirmar la destrucción con éxito.
- imprimirProfesor(): devuelve un string con los atributos del profesor para mostrarlo por pantalla

7.6.3 Clase Asignatura:

Siendo esta la clase insignia del proyecto, es de esperar que contenga más atributos y métodos que las anteriores, siendo en este caso 5 atributos privados, del cual uno es un puntero a un objeto personalizado, que corresponde al profesor, y el otro un puntero a un vector de alumnos, procedo a detallar los atributos aquí:

- codigo: se corresponde con el ID de la asignatura, como es posible que la organización decida asignar códigos alfanuméricos a las asignaturas, hemos decidido darle tipo string
- Nombre: el nombre que se muestra de la asignatura, es de tipo string
- anio: el año académico de impartición de la asignatura, es de tipo int
- profesorAsociado: representa el profesor asignado a impartir la asignatura, es un puntero a un objeto profesor
- alumnosAsignados: los alumnos que están matriculados en la asignatura, es un puntero a un vector de punteros a Alumnos

Para poder trabajar con los objetos de la clase, hemos creado los siguientes nueve metodos:

- Asignatura(): constructor por defecto de la clase
- Asignatura(string codigo, string nombre, int anio, Profesor * profesorAsociado): constructor con atributos para la clase.
- ~Asignatura: destructor del objeto, muestra un mensaje por consola para confirmar la destrucción con éxito
- imprimirAsignatura(): muestra por consola, de una forma elegante, la información de la asignatura, junto con su profesor asociado y sus alumnos matriculados.
- imprimirAsignaturaLite(): similar a los métodos de profesor y alumno, devuelve la información básica, pero no el profesor ni los alumnos, devuelve un string
- numeroDeAlumnos(): devuelve el número de alumnos asignados a la asignatura
- contieneAlumno(Alumno * alumno): devuelve un 1 si un alumno pasado por puntero se encuentra matriculado en la asignatura

- getCodigo(): devuelve el ID de la asignatura
- getAlumnosAsignados(): devuelve el puntero al vector que contiene los alumnos

7.6.4 Clase menu

La clase menu contiene el menu de la aplicación junto con los vectores en el free store donde se han de guardar los datos durante el tiempo de ejecución, contiene los siguientes atributos:

- seleccion: se utiliza en el switch case del menu, toma la entrada del usuario y lo dirige a la funcionalidad adecuada. Es un char.
- banderaSalida: se utiliza para romper el while y salir del menu. Es de tipo int.
- profesores: contiene los punteros de todos los profesores en el free store durante el tiempo de ejecución. Es de tipo vector<>*
- alumnos: contiene los punteros de todos los alumnos en el free store durante el tiempo de ejecución. Es de tipo vector<>*
- asignaturas: contiene los punteros de todas las asignaturas en el free store durante el tiempo de ejecución. Es de tipo vector<>*

El único método público que posee es el método menu(), que actúa como constructor y menu, le muestra al usuario y le permite navegar por el menu con comandos de texto, utiliza un while y un switch case, cuando sale utilizando la opción 0, se encarga de borrar los vectores creados durante la ejecución.

8 ENLACES Y REFERENCIAS

En este aparto, presentamos los enlaces a todos los repositorios de GitHub creados, se crearon 6 repositorios en total, uno para cada ejercicio. Los repositorios son de tipo público, por lo cual cualquier puede acceder.

- Ejercicio 1:
  - https://github.com/alucates/trabajos
- Ejercicio 2:
  - https://github.com/Geronimo-Basso/Mochila
- Ejercicio 3:
  - https://github.com/espartanlast1/ActividadGrupal23Matrices
- Ejercicio 4:
  - https://github.com/Geronimo-Basso/AltaBajaDeUsuario
- Ejercicio 5:
  - https://github.com/SrBernuy/PEL_Ej5_AC2
- Ejercicio 6:
  - https://github.com/Geronimo-Basso/SistemaDeMatriculacionDeAlumnos

9 CONCLUSIONES

9.1 CONCLUSIONES DEL TRABAJO

Basado en todo lo hecho en las páginas anteriores, podemos concluir que la actividad grupal 2 queda terminada. Como equipo consideramos que fue una actividad muy positiva, principalmente porque mejoramos nuestras habilidades en temas como, punteros, el free store y el manejo en general de C++. No solo eso, sino que también continuamos mejorando con el uso de herramientas de desarrollo como son GitHub y Git. Esperamos que el proyecto cumpla con las expectativas del docente que lo va a evaluar.