Escribir un resumen del capitulo 1 del libro "Data Structures and Algorith Analysis in C++" siguiendo los criterios de la rúbrica que se agrega en esta tarea. Recuerden que si algún concepto no les queda claro pueden mencionarlo en su resumen para profundizarlo en la clase pero también es necesario que además del libro añadan algunas otras fuentes de información.

Recuerden que incluso los programadores detrás de juegos épicos como Fortnite o Minecraft comenzaron desde cero, como un personaje de nivel 1 en un MMORPG. No se desanimen por los errores; cada bug resuelto los hace más fuertes, como la evolución constante de Kratos en God of War.

Resumen Cap. 1

Estructura de datos

Jonathan Nava Noguez IPV1412

Data Structures & Algorith Analysis

Mark Allen Weiss.

El libro comienza explicando la base aritmética y matemática que se necesita para comprender la programación empezando por los logaritmos, introducidos por John Napier, son herramientas fundamentales que convierten operaciones de multiplicación y división en sumas y restas, simplificando cálculos complejos. Las series, secuencias infinitas de números, ofrecen un vasto campo de estudio, desde la serie geométrica hasta la serie de Taylor, revelando la complejidad y la belleza de las matemáticas.

La aritmética modular, que estudia los restos de la división de números enteros, encuentra aplicaciones en criptografía y teoría de números, ofreciendo una nueva perspectiva sobre la estructura de los enteros.

Los métodos de demostración, como la demostración directa y la inducción matemática, son herramientas esenciales para construir y justificar afirmaciones matemáticas, permitiendo a los matemáticos establecer conexiones y descubrir verdades fundamentales.

El concepto básico de C++ es una clase que consta de miembros que pueden ser datos o funciones. Estos miembros definen el comportamiento y la información de los objetos creados a partir de la clase. Las funciones que pertenecen a una clase se denominan funciones miembro o métodos.


```
1  /**
2   * A class for simulating an integer memory cell.
3   */
4  class IntCell
5  {
6    public:
7      /**
8       * Construct the IntCell.
9       * Initial value is 0.
10      */
11     IntCell( )
12       { storedValue = 0; }
13
14     /**
15      * Construct the IntCell.
16      * Initial value is initialValue.
17      */
18     IntCell( int initialValue )
19       { storedValue = initialValue; }
20
21     /**
22      * Return the stored value.
23      */
24     int read( )
25       { return storedValue; }
26
27     /**
28      * Change the stored value to x.
29      */
30     void write( int x )
31       { storedValue = x; }
32
33   private:
34     int storedValue;
35 };
```

Figure 1.5 A complete declaration of an `IntCell` class

La clase IntCell que simula un dispositivo de memoria utilizado para almacenar números enteros. Cada objeto IntCell contiene un miembro de datos denominado valor almacenado. Esta clase también incluye métodos para leer y escribir en esta celda de memoria y constructores para inicializar objetos IntCell. La visibilidad de los miembros de la clase está controlada por los modificadores públicos y privados. Se puede acceder a los miembros públicos mediante cualquier método de cualquier clase, mientras que a los miembros privados solo se puede acceder mediante métodos de su propia clase. Esta diferencia le permite ocultar los aspectos internos de la implementación de la clase, lo que hace que el código sea más fácil de mantener y modificar.

Además, se tratan algunas mejoras de sintaxis y mejores prácticas para trabajar con clases C. Estos incluyen el uso de parámetros predeterminados en los constructores, listas de inicialización para inicializar elementos materiales, declarar constructores de un solo argumento explícitos para evitar conversiones implícitas innecesarias y marcar funciones miembro persistentes como accesores para indicar que no cambiarán el estado del objeto.

Finalmente, analizamos la separación de las clases de interfaz y de implementación en archivos .h y .cpp, respectivamente, para mejorar el modularidad y la legibilidad del código. También se mencionan las clases de vectores y cadenas proporcionadas por la biblioteca estándar de C, que proporcionan una funcionalidad mejorada sobre las cadenas y matrices nativas de C.

Los templates de función son patrones para funciones que pueden trabajar con cualquier tipo de datos. Se muestra un ejemplo de un template de función llamado findMax, que encuentra el máximo elemento en un vector de elementos. Este template toma un argumento de tipo Comparable, que puede ser cualquier tipo que admita la comparación. El código del template se expande automáticamente según sea necesario, pero puede generar un exceso de código en grandes proyectos. Por otro lado, los templates de clase funcionan de manera similar a los templates de función, pero se aplican a clases en lugar de funciones. Se muestra un ejemplo de un template de clase llamado MemoryCell, que simula una celda de memoria para almacenar cualquier tipo de dato. Al igual que con los templates de función, los templates de clase permiten la creación de estructuras de datos independientes del tipo.

Se aborda el tema de los objetos de función, que son utilizados para resolver limitaciones en los algoritmos genéricos que dependen de operaciones de comparación específicas. Se muestra cómo se puede usar un objeto de función como argumento en un algoritmo genérico, lo que permite una mayor flexibilidad en la comparación de elementos.

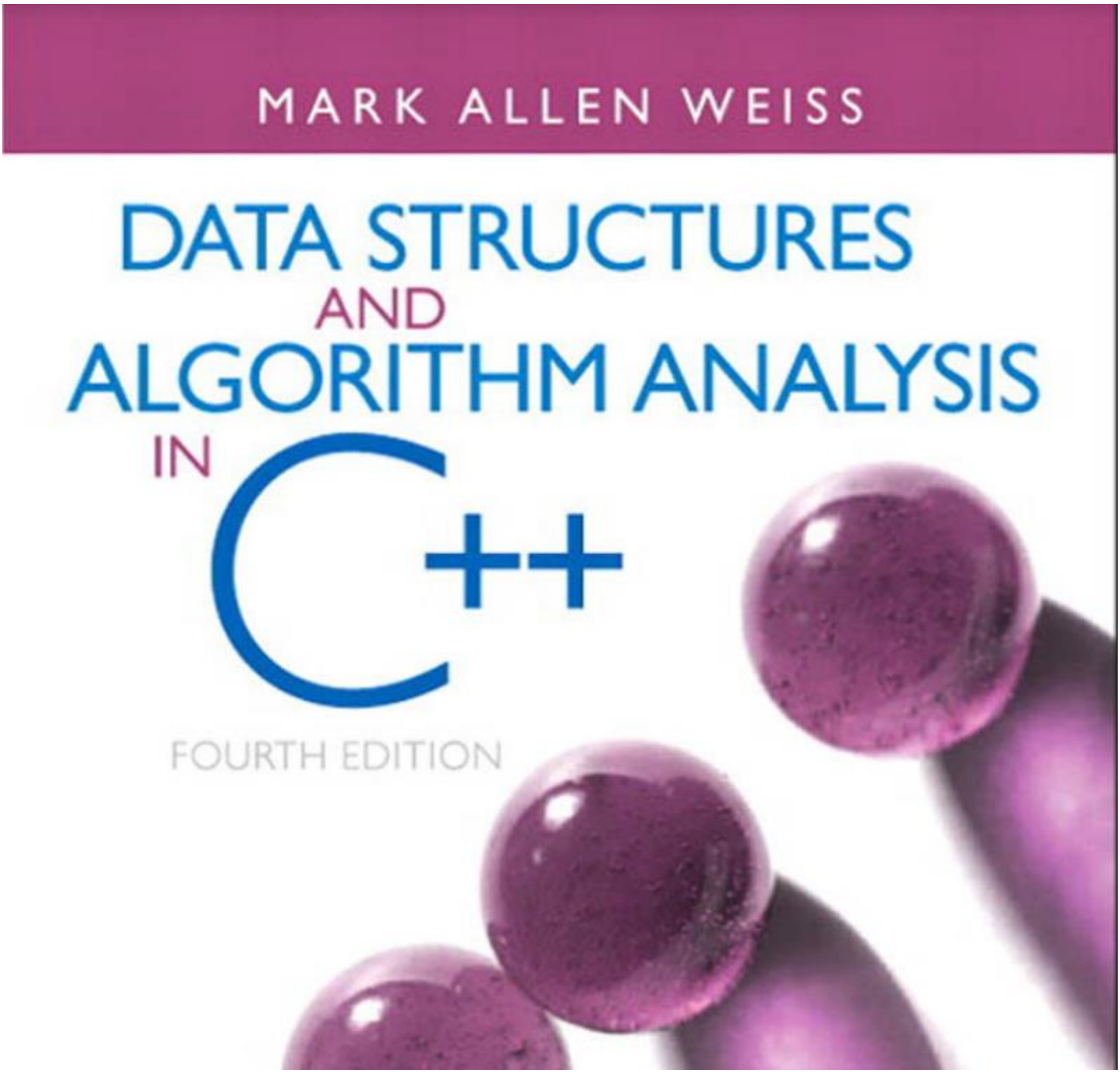
Se introduce la noción de algoritmos recursivos, ejemplificando con el cálculo de los números de Fibonacci. Se detalla cómo un algoritmo recursivo puede transformarse en uno iterativo para mejorar su eficiencia temporal. Además, se ofrecen ejercicios prácticos para reforzar los conceptos presentados, como la implementación de algoritmos para resolver problemas de selección de palabras y contar el número de unos en la representación binaria de un número.

El texto aborda también la complejidad del tiempo de ejecución de los algoritmos, presentando ejercicios para estimar sumas infinitas y calcular expresiones exponenciales módulo un número dado. Estos ejercicios sirven como práctica para comprender conceptos matemáticos fundamentales que se aplicarán en el análisis de algoritmos a lo largo del libro.

En la sección 1.6, se introduce el concepto de plantillas en C++, que permiten escribir algoritmos y estructuras de datos independientes del tipo de datos con el que operan. Se describen las plantillas de funciones y clases, con ejemplos prácticos como la implementación de una función findMax que encuentra el máximo elemento en un vector,

independientemente del tipo de datos contenido en el vector. Además, se discuten los "objetos de función", que permiten pasar funciones como parámetros a otras funciones. Esto se ilustra con el ejemplo de una clase CaseInsensitiveCompare, que se utiliza para comparar cadenas de texto sin tener en cuenta las diferencias de mayúsculas y minúsculas.

Avanzando un poco más en la lectura, se aborda la implementación de matrices en C++. Se proporciona una clase matrix que utiliza un vector de vectores para representar una matriz bidimensional. Se detallan aspectos importantes como la sobrecarga del operador de indexación (operator[]) y se discute la necesidad de devolver una referencia constante en algunos casos para garantizar la integridad de los datos.



Después de haber leído el libro de "Data Structures & Algorithm Analysis" de Mark Allen Weiss, me he dado cuenta de la falta de conocimiento que tengo en el área de programación logística, especialmente en inglés. Sin embargo, esta lectura también me ha proporcionado información de calidad en programación, como los conceptos de constructores, punteros, variables y funciones, entre otras cosas. Los ejemplos prácticos incluidos en el libro han sido de gran ayuda para mi comprensión. A pesar de haber adquirido este conocimiento, todavía me siento inexperto en este aspecto. Nunca he sido bueno para memorizar características específicas y tiendo a olvidar detalles con el tiempo. Sé que, con el tiempo, podré olvidar cómo usar correctamente lo que he aprendido. Sin embargo, estoy decidido a poner en práctica lo que he leído, incluso si es solo a un nivel básico. Mi objetivo es asegurarme de que este conocimiento permanezca en mi mente el mayor tiempo posible.

La lectura de este libro me ha abierto los ojos a mis deficiencias en programación logística y habilidades lingüísticas en inglés. Aunque aún me siento inseguro en este campo y sé que puedo olvidar lo aprendido con el tiempo, estoy comprometido a aplicar lo que he aprendido para afianzar mi comprensión y retener este conocimiento de la mejor manera posible.

IEEE:

1. "Data Structures & Algorith Analysis Mark Allen Weiss," *Google Classroom*. https://classroom.google.com/u/0/c/NjY2MjkwMzg0ODgz/m/NjY3NzEyMzc1MDM1/details
2. "El lenguaje C++ — Fundamentos de Programación en C++." https://www2.eii.uva.es/fund_inf/cpp/temas/1_introduccion/introduccion.html