Centro Universitario de Guatemala
Universidad Mariano Gálvez
Segundo Semestre Sección “A”
Ingeniería En Sistemas
Curso: Algoritmos
Catedrático: Inge. Alan Gonzalo

PROYECTO FINAL

Estudiantes que conforma el grupo:

Joshua Iván André Méndez Vásquez ——————————No. Carné: 9490-22-4032
Cristhian Sebastián Rodas Arriola ——————————No. Carné 9490-22-523
Pablo Andre Melendez Melendez ——————————No. Carné 9490-22-611
Erich Walter Dahinten Colmeranes ----------No. Carné 9490-22-2786
Alder Isaac Solis De Leon ——————————No. Carné 9490-22-227
Cristofer Gabriel Guerrero Fajardo ----------No. Carné 9490-22-696

Guatemala, octubre de 2022

ÍNDICE

Conceptos

En su núcleo, C + + es un lenguaje de programación que proviene de la extensión del lenguaje C para que pudiese manipular objetos. A pesar de ser un lenguaje con muchos años, su gran potencia lo convierte en uno de los lenguajes de programación más demandados.

Una biblioteca es un lugar donde podemos encontrar libros o cualquier otro soporte de un texto, como publicaciones, revistas, documentos, catálogos, etcétera. Una biblioteca puede ser a la vez la pieza de mobiliario donde colocamos los libros (que usualmente está formada por varios estantes horizontales) o el edificio en sí, donde podemos consultar y tomar prestados libros y demás.
Desde la última versión realizada de nuestro código, C + + no requiere un archivo de configuración para agrupar este proyecto. Sin embargo, es increíblemente configurable para adaptarse mejor a sus necesidades.

Para comenzar solo necesitas entenderlo Conceptos básicos:

- Entrada
- Salida
- Declaraciones
- Funciones
- Bibliotecas
- Compatibilidad del sistema

Este documento está destinado a dar un alto nivel descripción general de estos conceptos, al tiempo que proporciona una explicación a casos detallados del uso específico del concepto.

Para una mejor comprensión de las ideas detrás de los agrupadores de módulos y cómo funcionan bajo el capó, consulte estos recursos:

- Información general de módulos
- Módulo, importar, exportar
- Módulos con nombre

Entrada

Un programa en C + + puede realizar operaciones de entrada y salida de varias formas distintas. Un flujo de entrada no es más que la serie de entradas que alimentan un ordenador para que el programa las utilice.
Supondremos que la entrada proviene del teclado por lo que esta sección son las salidas se envían a la pantalla de un terminal por ejemplo:

```cpp
#include "iostream"
#include "string"

using namespace std;

int main()
{
    cout << "Hola! Este es un ejemplo en C++" << "\n" << "Por favor ingrese su nombre:" << "\n";
    //La instrucción \n es un salto de línea Mostrando los textos separados

   string nombre;//En esta variable estará almacenado el nombre ingresado.
   cin >> nombre; //Se lee el nombre

   cout << "Bienvenido al sistema " << nombre << ". Gracias por usar nuestra aplicación" << "\n";


   return 0;
}
```

Aprende más de los puntos de entradas.

Salida

La salida de datos en C + + es simple (al menos para los casos más comunes). Con esto hago una comparación entre C + + y Java, pues leer una entrada del usuario por teclado en C + + es bastante simple mientras que en Java implica una serie de conceptos adicionales que la hacen un tanto más complicada. Por suerte para nosotros en esta sección veremos cómo leer por teclado en C + +, en otras palabras, asignar a una variable el valor que el usuario escriba por el teclado. Adicional a esto veremos también cómo mostrar texto por pantalla en C + +, es decir, salida de datos.

Mostrar texto por pantalla en C++ es muy simple. Para imprimir una salida de texto en C++ se hace uso de la instrucción *cout*, junto con <<. Es importante tener en cuenta que la instrucción *cout* siempre va acompañada de << para controlar el flujo de datos que sale. No te fijes mucho en ellos, solo ten siempre presente que *cout* viene acompañado de << para tener *cout* << como resultado.

```cpp
#include "iostream"

using namespace std;

int main()
{
    //Se muestra un mensaje por pantalla.
    cout << "Hola Mundo" << " Desde ProgramarYa." << "\n";

    return 0;
}
```

Aprende más de los puntos de salida.

Declaraciones

El programa de C + + consta de varias entidades, como variables, funciones, tipos y espacios de nombres. Cada una de estas entidades debe declararse para que puedan usarse. Una declaración especifica un nombre único para la entidad, junto con información sobre su tipo y otras características. En C + +, el punto en el que se declara un nombre es el punto en el que se vuelve visible para el compilador. No puede referirse a una función o clase que se declara en un punto posterior en la unidad de compilación. Las variables se deben declarar lo más cerca posible antes del punto en el que se usan cabe recalcar que hay diferentes tipos de librerías las cuales tiene diferentes tipos de declaraciones en C + +.

```
#include <string>

int f(int i); // forward declaration

int main()
{
    const double pi = 3.14; //OK
    int i = f(2); //OK. f is forward-declared
    C obj; // error! C not yet declared.
    std::string str; // OK std::string is declared in <string> header
    j = 0; // error! No type specified.
    auto k = 0; // OK. type inferred as int by compiler.
}

int f(int i)
{
    return i + 42;
}

namespace N {
   class C{/*...*/};
}
```

La configuración está definida por unas propiedades para un ese módulo con esas propiedades requeridas están para toda prueba y uso. Esto le dice al compilador de C + + que muestre algunas declaraciones.
Aprende más sobre las declaraciones.

Funciones

Una función es un bloque de código que realiza alguna operación. Una función puede definir opcionalmente parámetros de entrada que permiten a los llamadores pasar argumentos a la función. Una función también puede devolver un valor como salida. Las funciones son útiles para encapsular las operaciones comunes en un solo bloque reutilizable, idealmente con un nombre que describa claramente lo que hace la función. La siguiente función acepta dos enteros de un autor de llamada y devuelve su suma; *a* y *b* son *parámetros* de tipo`int`.

C++ Copiar

```
int sum(int a, int b)
{
    return a + b;
}
```

La función puede ser invocada, o *llamada*, desde cualquier lugar del programa. Los valores que se pasan a la función son los *argumentos*, cuyos tipos deben ser compatibles con los tipos de los parámetros en la definición de la función.

No hay ningún límite práctico para la longitud de la función, pero un buen diseño tiene como objetivo funciones que realizan una sola tarea bien definida. Los algoritmos complejos deben dividirse en funciones más sencillas y fáciles de comprender siempre que sea posible.

Las funciones definidas en el ámbito de clase se denominan funciones miembro. En C + +, a diferencia de otros lenguajes, una función también puede definirse en el ámbito de espacio de nombres (incluido el espacio de nombres global implícito). Estas funciones se denominan *funciones libres* o *funciones no miembros*; se utilizan de forma exhaustiva en la biblioteca estándar.

Las funciones pueden ser *sobrecargadas*, lo que significa que diferentes versiones de una función pueden compartir el mismo nombre si difieren por el número y/o tipo de parámetros formales. Para obtener más información, consulte Sobrecarga de funciones.

Bibliotecas

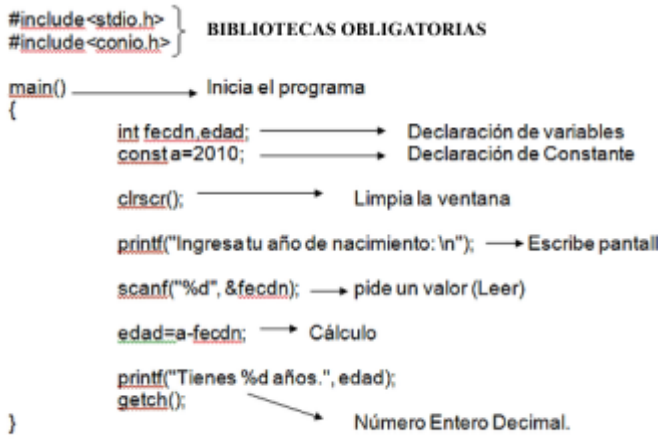
Se han efectuado diversas mejoras en el proyecto en las bibliotecas que se suministran, incluida la biblioteca estándar de C + +, para que sean más seguras y prácticas.
Varios métodos de la biblioteca estándar de C + + se identificaron como potencialmente faltantes porque provocan, la insuficiente cantidad de proceso de información por lo que resulta en una saturación del búfer u otro defecto de código. El uso de estos métodos son necesarios y se crearon nuevos métodos más seguros para reemplazar a todos estos métodos.

La biblioteca de C++ cumple prácticamente con las mismas convenciones que la biblioteca estándar de C, además de algunas más que se describen aquí.
Una implementación tiene cierta libertad en cómo declara los tipos y funciones de la biblioteca de C++:

- Los nombres de funciones de la biblioteca estándar de C pueden tener la vinculación extern "C++" o extern "C". Incluya el encabezado estándar de C adecuado, en lugar de declarar una entidad de biblioteca insertada.

- Un nombre de función miembro en una clase de biblioteca puede tener firmas de función adicionales además de las que aparecen en este documento. Puede asegurarse de que una llamada a función que se describe aquí se comporte según lo esperado, pero no puede tomar de forma fiable la dirección de una función miembro de biblioteca. (El tipo podría no ser el esperado).

- Una clase de biblioteca puede tener clases base (no virtuales) sin documentar. Una clase documentada según se deriva de otra clase puede, de hecho, derivarse de esa clase a través de otras clases sin documentar.



Aprende más sobre las bibliotecas.

Compatibilidad del sistema

Esta Biblioteca realizada en C + + es compatible con cualquier tipo de máquina no importando su sistema operativo ni la potencia del mismo en lo único que se diferenciaría son en los tiempos de ejecución la cual nosotros no nos tomamos demasiado tiempo en ejecutarlo y esto se afirma por la claridad de lo que es el código.

Environment

C++ se ejecuta en el programa DEV-C++ en la versión 6.30

Puntos de Entrada

Como se mencionó anteriormente en entrada hay multitud de formas en las cuales definir lo que quiere el usuario, le mostraremos las formas en que se configuró la propiedad de entrada, además de explicar por qué fue útil en este programa.

Sintaxis de entrada

<iostream> es un componente de la biblioteca estándar (STL) del lenguaje de programación C + + que es utilizado para operaciones de entrada/salida. Su nombre es un acrónimo de Input/Output Stream. El flujo de entrada y salida de datos en C + + (y su predecesor C) no se encuentra definida dentro de la sintaxis básica y se provee por medio de librerías de funciones especializadas como iostream. iostream define los siguientes objetos:

- cin : Flujo de entrada
- cout : Flujo de salida
- Cerr : Flujo de error no almacenado.
- Clog : Flujo de error almacenado.

Todos los objetos derivados de iostream forman parte del espacio de nombres std.

Por lo que en nuestro programa declaramos los objetos que controlan la lectura y escritura en los flujos estándar. Esta inclusión suele ser el único encabezado que necesita incluir para realizar la entrada y salida de esta biblioteca.
Tras realizar determinadas operaciones en un flujo, como la entrada estándar, no es posible realizar operaciones de otra orientación en el mismo flujo. Por lo tanto, un programa no puede funcionar indistintamente tanto con cin como con wcin, por ejemplo.
Todos los objetos que se declaran en este encabezado tienen en común una propiedad peculiar: se puede suponer que se construyen antes que cualquier objeto estático que se defina, en una unidad de traducción que incluye <iostream>. De igual manera, se puede suponer que estos objetos no se destruyen antes que los destructores de todos esos objetos estáticos que se definen. (Sin embargo, los flujos de salida se vacían durante la finalización del programa). Por lo tanto, puede leer o escribir sin ningún riesgo en los flujos estándar antes de iniciar el programa y después de la finalización del programa.

windows.h es un archivo cabecera específico de Windows para la programación en lenguaje C/C++ que contiene las declaraciones de todas las funciones de la biblioteca de este programa, todas las macros utilizadas por los programadores de aplicaciones para Windows, y todas las estructuras de datos utilizadas en gran cantidad de funciones y subsistemas. La Win32 API se agregó en este proyecto de programación en C haciendo la inclusión de la cabecera <windows.h>

string.h es un archivo de la Biblioteca estándar del lenguaje de programación C que contiene la definición de macros, constantes, funciones y tipos y algunas operaciones de manipulación de memoria lo que hace manejable esta biblioteca.

Las funciones declaradas en string.h se han agregado, por lo que están garantizadas para cualquier plataforma que soporte C. Además, las funciones para cadenas de caracteres sólo trabajan con conjuntos de caracteres ASCII o extensiones ASCII compatibles por lo que el encabezado definió el tipo de variable y macros en varias funciones para manipular distintas matrices de variables.

<cstdlib> Este encabezado definió varias funciones de propósito general, incluida la gestión dinámica de la memoria, la generación aleatoria de números, la comunicación con el medio ambiente, la aritmética entera, la búsqueda, la clasificación y la conversión. Incluye el encabezado <stdlib.h> de la biblioteca estándar de C que agrega los nombres asociados al espacio de nombres std. Incluye este encabezado también garantización de los nombres declarados mediante vinculación externa en el encabezado de la Biblioteca Estándar C se declaren en el espacio de nombres std.

<vector> Los vectores fueron los contenedores de secuencia que representan las matrices que pueden cambiar de tamaño. Al igual que las matrices, los vectores usan ubicaciones de almacenamiento contiguas para sus elementos, lo que significa que también que ahora se accederá a sus elementos en las reglas. Pero a diferencia de las matrices, su tiempo puede cambiar dinámicamente, con su alma en un momento automático por el contenido.

El encabezado <algoritmo> define una colección de funciones que fueron especialmente diseñadas para ser utilizadas en rangos de elementos extendidos en esta biblioteca. El rango es cualquier seguridad de objetos a los que se puede acceder a los trabajos de iteradores o juegos, como una matriz o una instancia de algunos de los Contenedores STL Sin embargo, tomemos en cuenta que los algoritmos se operaron a través de iteradores directamente en los valores, sin afectar de ninguna manera la estructura de ningún contenedor posible (nunca afecta el tamaño o la asignación de almacenamiento del contenedor).

<fstream> Los objetos de esta clase mantienen al búfer de flujo interno, que realiza operaciones de entrada / salida en el archivo con el que están asociados (si corresponde).
Las secuencias de archivos que están asociadas con archivos en calidad de la construcción, o llamando a un miembro abierto.

Sintaxis de Objeto

Una clase es en general un modelo, una receta o plantilla que define el estado y comportamiento de cierto tipo de objetos. Las clases se usaron para pensarse como una colección de variables (atributos o propiedades) y funciones (métodos) que permiten representar el conjunto de datos y especificar las operaciones o procedimientos que permitirán manipular tales datos. Se puede inclusive entender las clases como un tipo de dato personalizado, similar a las estructuras (structs), donde cada programador define los miembros que va a tener su tipo de dato. De hecho, los tipos de datos nativos de C + + son en realidad clases.
Los objetos fueron una instancia de una clase, es decir una entidad que se construye a partir de las descripciones consignadas en una clase (datos y funciones). Por tanto, el objeto se debe entender como una "variable" que se declara del tipo de dato de cierta clase

y el objeto es como tal la entidad tangible que permite acceder a los datos y funciones modeladas al interior de la clase dando como resultado el tipo de manejo de índole de dato entonces el manejo de las librerías usadas en esta biblioteca se mantuvieron de manera funcional y ordenada.

En el C y C ++ lenguajes de programación, <unistd.h> este nombre de tipo archivo de encabezado proporciona acceso a la POSIX sistema operativo API. Está definido por el estándar POSIX.1, la base del. Especificación de unix único, y, por lo tanto, debe estar disponible en cualquier sistema operativo compatible con POSIX y compilador. Por ejemplo, esto incluye. Unix y Unix-like sistemas operativos, como variantes GNU, distribuciones de Linux y BSD, y macOS, y compiladores como GCC y LLVM.

En sistemas tipo Unix, la interfaz definida por unistd.h está compuesto generalmente de funciones de envoltura de llamadas del sistema como tenedor, tubería and E / S primitivos (read, write, close, etc.).
Capaces de compatibilidad Unix como Cygwin y MinGW También proporcionan sus propias versiones de unistd.h. De hecho, esos sistemas nos proporcionan junto con las bibliotecas de traducción que implementan sus funciones en términos de funciones win32. Por lo tanto, unistd.h es solo una capa adaptativa definida genéricamente para basarse en definiciones específicas del sistema y del compilador ya existentes. Esto tiene la ventaja general de no tener un conjunto posiblemente concurrente de archivos de encabezado definidos, sino uno que se construye sobre la misma raíz que, por esta razón, plantea muchas menos preocupaciones en los casos de uso combinado.

<conio.h> es un archivo de cabecera escrito en C que se formuló mayormente para proveer un sistema de E/S por consola. Éste no es parte de la biblioteca estándar de C o ISO C, ni está definida por POSIX pero esta cabecera declara varias funciones útiles que se usaron para mejorar el rendimiento de la «entrada y salida por consola» desde este programa. La mayoría de los compiladores de C creados tienen esta biblioteca que suministran las funciones de la biblioteca asociadas en la biblioteca por defecto de C. La mayoría de los compiladores C creados no tienen esta biblioteca y no suministran las funciones de esta biblioteca. Algunos sistemas embebidos usan una biblioteca compatible.

Las funciones de biblioteca declaradas por conio.h varían ligeramente dependiendo el compilador. Originalmente implementada en Lattice C, las funciones mapeadas directamente a las primeras pocas funciones INT 21H de DOS. La biblioteca provista por Borland Turbo C no usaba la DOS API pero accedía a la memoria de vídeo directamente para la salida y usaba llamadas de interrupción de la BIOS. Esta biblioteca contiene funciones adicionales inspiradas en las funciones de Turbo Pascal.

Objeto de inscripción de entrada

El objeto de descripción del punto de entrada se usó para especificar las siguientes propiedades.

DependOn: los puntos de entrada de los que depende el punto de entrada actual. Deben cargarse antes de cargar este punto de entrada.

Filename: especifica el nombre de cada archivo de salida en la consola.

Import: Módulo(s) que incluso fueron personalizados para cargar al inicio.

Biblioteca: especifique las opciones de la biblioteca para agrupar una biblioteca desde la entrada actual.

Runtime: el nombre del fragmento de tiempo de ejecución.

Módulos

En la programación modular, todos los que trabajamos en el programa dividimos los programas en fragmentos discretos de funcionalidad llamados a módulos.

Cada módulo tiene un área de superficie más pequeña que un programa completo, lo que hace que la verificación, la depuración y las pruebas sean triviales módulos proporcionar abstracciones sólidas y límites de encapsulación, de modo que cada módulo tenga un diseño coherente y un propósito claro dentro de la aplicación general.

Se ha admitido la programación modular casi desde su inicio. Sin embargo, en la compilación, es compatible con esta variedad de beneficios y limitaciones. C + + se basa en las lecciones aprendidas de estos sistemas y aplica el concepto de módulos a cualquier archivo en su proyecto.

Objetivos

Debido a que C + + se puede escribir tantas líneas de código aunque para su práctica no es lo mejor que se puede aplicar en ella ya que no se debe de romper el principio de responsabilidad única pero de igual manera el objetivo del programa sea conseguido de manera efectiva todo lo aplicado en esto se logró a base de conceptos investigados, interpretados de cierta manera de las clases impartidas.

Configuración

Fuera de la biblioteca, C + + no requerirá que use un archivo de configuración. Sin embargo asumirá que el punto de entrada y el resultado bajo ciertos parámetros optimizados para la realización del programa por lo que todo será interno y bajo cierto control de seguridad para que nada que no esté dentro del programa salga y/o se pueda modificar de tal forma que el programa ya no sea útil.

Configurar un nuevo inicio de proyecto

Al ser C + + el lenguaje de programación tiene un gran conjunto de opciones que además de ser abrumadoras, aprovecharán todo lo que sea lo que tiene y no dejará nada que no

esté en su límite para poder realizar el mismo. Por lo que anteriormente se mencionó que los requisitos del proyecto no son de suma importancia ya que en cualquier sistema en donde se encuentre el programa dev-c++ no habrá ningún problema ya que en la instalación del mismo incluirán todos los plugins y complementos necesarios para ello.

Lenguajes de configuración

C + + acepta archivos que forman parte de la extensión ".h" que son generalmente los archivos de encabezado utilizados con los lenguajes de programación C ++ o C. Los archivos .h son llamados como los 'archivos de cabecera' por los programadores. Pueden consistir en definiciones de variables externas, prototipos de funciones y constantes para manejar muchos tipos diferentes de archivos de configuración.

De cierta manera tenemos los archivos .win, que anteriormente se habló de esta integración que se da por la introducción a Win32 a C + + por lo que el API de Windows para plataformas de 32 bits, llamada informalmente Win32, es el nombre dado por Microsoft al conjunto de interfaces de programación de aplicaciones disponibles en el núcleo de los sistemas operativos Windows. la agregamos para usarse en el lenguaje de programación C, y por lo tanto, se adapta a C + + sin dificultad (aunque la mayoría de los lenguajes soporta enlazados con código en C). Involuntariamente, es la forma más directa de que esta aplicación interactúa con el sistema operativo Windows. Un acceso a bajo nivel al sistema operativo (usado principalmente en el desarrollo de manejadores de dispositivos) que provee.

El archivo CSV se conoce comúnmente como formato de archivo de texto, donde los valores están separados por comas en cada línea. Las líneas se denominan registros de datos y, por lo general, cada registro consta de más de un campo, separados por comas. El formato CSV se usó principalmente para almacenar datos tabulares, por lo que contiene un número igual de delimitadores de coma en cada registro. Sin embargo, tenga en cuenta que, dado que existen múltiples implementaciones en la práctica, uno podría encontrar, por ejemplo, espacios entre los campos separados por comas, o puede que no haya espacios entre los campos, etc.

Los archivos binarios que hasta ahora nos hemos centrado en los ficheros de texto, que son sencillos de crear y de leer. Pero también podemos manejar ficheros que contengan información de cualquier tipo. Los archivos binarios se trataron con la clase fstream. En este caso, especificamos si se desea entrada y salida, o sólo entrada o sólo salida. Uno de los usos que se aplicaron a este es utilizar el archivo como una pequeña base de datos, utilizando un registro como referencia. Por lo que tratamos de que todos los registros deben contuviesen el mismo tamaño.

Y por último los archivos .o que son algunos archivos que se crearon durante el proceso de desarrollo este programa o software que no están vinculados específicamente a ningún programa o función. Por lo que el archivo O. Son los archivos con esta breve extensión son creados por un compilador de C como, por ejemplo, GNU Compiler Collection (GCC). Que da el caso en que, durante el proceso de programación, un archivo no se guarde como archivo de programa válido. Cuando eso ocurre, es posible que en su lugar se cree un archivo de objeto O. Posteriormente, pueden vincularse de nuevo para generar el archivo ejecutable deseado desde el principio.

Los archivos O contienen nuestro código en C o C + + que el compilador con el que se han creado utiliza para procesar otros archivos de código fuente. Los archivos O contienen código más compacto, pues son tipo de archivos binarios, y pueden vincularse para generar

una biblioteca o un archivo ejecutable. La extensión de archivo O se implementó como archivo de objeto general y está asociada con numerosos programas y compiladores distintos.

Optimización

Optimización guiada por perfiles

Visual Studio admite la Optimización guiada por perfiles (PGO). Esta optimización permite usar datos de perfil de ejecuciones de entrenamiento de una versión instrumentada de una aplicación para controlar la optimización posterior de la aplicación. Por lo que utilización de PGO puede ocupar mucho tiempo; por tanto, no es correcto para utilizar todos los desarrolladores, sino que se recomienda utilizar PGO para la versión de lanzamiento final de un producto. Para más información, vea Optimizaciones guiadas por perfiles.
Además, se ha mejorado la optimización de todo el programa (que también se conoce como Generación de código en tiempo de vínculo) y las optimizaciones /O1 y /O2 . En general, una aplicación compilada con una de estas opciones es más rápida que la misma aplicación compilada con un compilador anterior.
Para obtener más información, vea /GL (Optimización de todo el programa) y /O1, /O2 (Minimizar tamaño, maximizar velocidad).

¿Qué nivel de optimización usamos?

Todo fue posible gracias a las últimas versiones de lanzamiento que se compilaron con las optimizaciones guiadas por perfiles. Si no es posible generar con PGO, ya sea debido a una infraestructura insuficiente para ejecutar las compilaciones instrumentadas, o bien por no tener acceso a los escenarios, se recomienda realizar la generación con la optimización de todo el programa.

El modificador /Gy también resulta muy útil. Genera un COMDAT independiente para cada función, dando más flexibilidad al vinculador cuando quita COMDAT sin referencia y plegamiento de COMDAT. El único inconveniente de usar /Gy es que puede producir problemas durante la depuración. Por consiguiente, por lo general se recomienda utilizarlo. Para obtener más información, vea /Gy(Habilitar vinculación en el nivel de función).

Para vinculaciones en entornos de 64 bits, se recomienda utilizar la opción del enlazador /OPT:REF,ICF y, en entornos de 32 bits, se recomienda /OPT:REF . Para obtener más información, consulte el artículo /OPT (Optimizaciones).

También se recomienda encarecidamente generar símbolos de depuración, incluso con versiones de lanzamiento optimizadas. Esto no afecta al código generado y facilita la depuración de la aplicación, si fuera necesario.

Caché

En la terminología C/C++, los flujos que son almacenados en la memoria caché (H5.2) se denominan "buffered". Los compiladores C/C++ disponen de este propio sistema de caché para ficheros (de disco o de otro tipo). Esta caché se denomina de ejecución (runtime), para distinguirla de la caché del Sistema. Así mismo, disponen de recursos en la Librería Estándar para forzar su vaciado en caso necesario. Para esto se recurre a la funciones fflush (para ficheros abiertos con fopen) y flush (para los flujos de salida, "ostreams").

Sin embargo, no olvide que el vaciado de la caché del compilador se realiza sobre la del Sistema, que está por debajo (recuerde que el Software tiene una estructura de capas E1.7a), y que el SO decide por su cuenta cuando es el momento oportuno para realizar físicamente la escritura de los discos. Esto significa que una seguridad total solo se alcanza forzando la escritura de la caché del Sistema, y esto naturalmente depende de la plataforma utilizada.

Guías

Esta sección contiene guías para comprender y dominar la amplia variedad de herramientas y características que ofrece nuestro programa. La primera es una guía que te lleva Empezando.

Los guías se vuelven más avanzados a medida que avanza. La mayoría sirve como punto de partida, y una vez completado, debe sentirse más cómodo sumergiéndose en la documentación real.

Empezando

Se ha realizado el programa para la biblioteca Souvenir, el cual ha sido desarrollado usando el lenguaje c++, a continuación se mostrara los requerimientos mínimos del sistema:

Biblioteca Souvenir - CLI	
Requerimiento	**Descripción**
Memoria	**512 mb**
Almacenamiento	**177 MB**
Software Específico	**C++**
Compiler arguments	**-g3 -g3 -std=c++0x**
Procesadores	**1**
Arquitectura soportada	**32 / 64 bits**

Getting Started

Para poder inicializar por primera vez el proyecto es necesario realizar primero clonar el proyecto el cual se encuentra versionado en el siguiente repositorio:

Repositorio: https://github.com/aisolis/Library-el-suvenir

Dentro del cual se encuentran los archivos del proyecto necesarios para realizar la compilación del programa, no necesitas nada más para poder iniciar al proyecto, dentro del se encuentra el archivo .exe el cual no es necesario que se realice la compilación del proyecto nuevamente, por lo que basta con darle doble click al ejecutable y empezar a utilizarlo.

Módulos

El programa se encuentra construido en base a módulos, los cuales nos dan acceso a distintos submenús dentro de los cuales se manejan roles de usuario, estos roles de usuario van orientados a la gestión de permisos lo cuales usan como input un número entero que va correlacionado con el perfil del usuario:

Para ello obtenemos la información del usuario y validamos el rol que posee, mismo que será devuelto al handler y seteado como rol de usuario para el resto de los componentes.

```
switch(aux.getRol()){

    case 1:{
        LoginForm::userPermission = 1;
        break;
    }

    case 2:{
        LoginForm::userPermission = 2;

        break;
     }

    case 3:{
        LoginForm::userPermission = 3;
        break;
     }

 }
```

con el rol seteado esto nos permitirá indicar al programa cual módulo o submódulo desplegar dependiendo del rol que sea seteado.

dicho programa está hecho con los principios de la programación orientada a objetos usando las clases como su principal vía de desarrollo, haciendo uso de las clases podemos realizar la instancia de los objetos

Clases del programa

Las clases del sistema las podemos clasificar en 4 tipos:

- Clases helpers
- Clases de modelo
- Clases de servicio
- Clases de datos

Dentro de las clases helpers podemos encontrar las siguientes:
- LoginForm
- AdminPanel
- SupervisorPanel
- OperatorPanel

Dentro de las clases de modelo tenemos:
- User
- Selling
- Book

Dentro de las clases de servicio tenemos:
- InventoryModule
- MasiveModule
- UserModule
- ReportsModule

Dentro de las clases de datos tenemos:
- binFilesHandler

Cada una de las clases cumple con su propia responsabilidad, lo que nos permite tener un mejor control de nuestras clases.

Administración de la capa de datos

Para esta capa de datos, se utiliza como base de datos un archivo en formato binario (.bin) de tal manera que la lectura de estos datos no pueden ser leídos con un editor de textos convencional, por lo que es perfecto para resguardar la seguridad de nuestros datos, estos son manejados a través de nuestro binFilesHandler el cual es el encargado de manejar nuestros datos para los binarios estos son utiles para nuestros inventarios, carga de datos de los inventarios, usuarios y cargas y descargas masivas…

para ello usamos las ventajas de los lectores y librerias como lo son <fstream> para poder aperturar el archivo y poder leer su contenido usando sus modificadores de lectura como lo pueden ser los ios::in, ios::out, ios::ap e ios::binary, esto nos permite poder realizar distintos

tipos de lectura en nuestro documento, también hacemos uso de la libreria <ctime> para poder realizar paseos de fechas y los vectores que seran recorridos en cada iteración para poder realizar el relleno de los vectores de los objetos User, Selling y Book, los cuales nos permitiran realizar la actualización de nuestros datos.

Hemos de resaltar que la clase más compleja es la clase del inventory Module, la cual contiene toda la lógica importante para poder manejar los datos, validarlos y enviarlos hacia la capa de datos, lo que nos va a permitir tener siempre datos limpios y validos, para esto hacemos uso de nuestro método auxiliar validateForm la cual validará y corrobora que mandemos datos sanitizados y posteriormente la enviará a el método fixFormData la cual nos permitirá corregir los datos que vengan con inconvenientes en su estructura o información, por lo que finalmente se devolverá al método validate form hasta que todos los datos estén correctamente ingresados.

Código de división

Para este programa se realizaron distintos métodos para cada clase, estos nos permiten fraccionar nuestros problemas en problemas más cortos, entre estos podemos destacar los más importantes que se encuentra en todas la clases y es el que nos permite posicionar el puntero en nuestra CLI para poder hacer la impresión o toma de datos, el cual es conocido comúnmente como gotoxy o Goto XY porque toma de referencias coordenadas para posicionarse en un lugar específico.

```
COORD Coord = {0,0};

void gotoxy(int x, int y){
	Coord.X = x;
	Coord.Y = y;
	SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),Coord);
}
```

esta poderosa herramienta nos permite realizar entornos gráficos en cli de una manera mas sofisticada y ordenada, a diferencia de si todo lo pusiéramos en fila cosa que permite poder redireccionar el puntero de la consola en cualquier parte que nosotros decidamos.

Cada uno de los métodos realizados cumplen con una función determinada, los mas comunes de ver son los display Module(), estos son encargados de visualizar el módulo de acuerdo a los roles de usuario establecidos, estos se encuentran en las siguientes clases:

1. InventoryModule
    a. void displayModule()
    b. void displayInventoryModuleSupervisor()
    c. displayInventoryModuleOperator()
    d.
2. MasiveModule
    a. void displayModule()
    b. void displayMasiveModuleSupervisor()

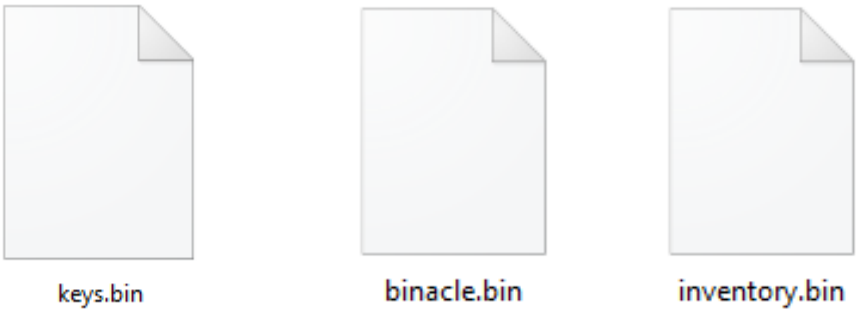
3. UserModule
    a. void displayModule()
    b. void displayModuleSupervisor()

4. ReportsModule
    a. void displayModule()

Todos los módulos son del tipo void porque no tienen que retornar nada, estos son los encargados de facilitar los accesos a los demás métodos, los cuales realizan toda la lógica del sistema.

Autorización de escritura de la biblioteca

Para asegurar la seguridad e integridad de los datos los archivos son manejados en formato binario, para que no pueda ser leído por editores de texto comunes, así mismo se ha limitado la escritura en la base de datos a los Administradores y Supervisores, los operadores no tendrán habilitado estas opciones.



Los únicos elementos sin seguridad son los que son disponibles para que el usuario realice la carga masiva, y los documentos que son resultados de descargas de inventario y / o reportes generados, mismos que son sobre escritos para evitar que se genere duplicidad de la información y / o que los datos sean extraídos de manera inadecuada.

Diagrama de clases

Diagrama de clases composición de la clase main

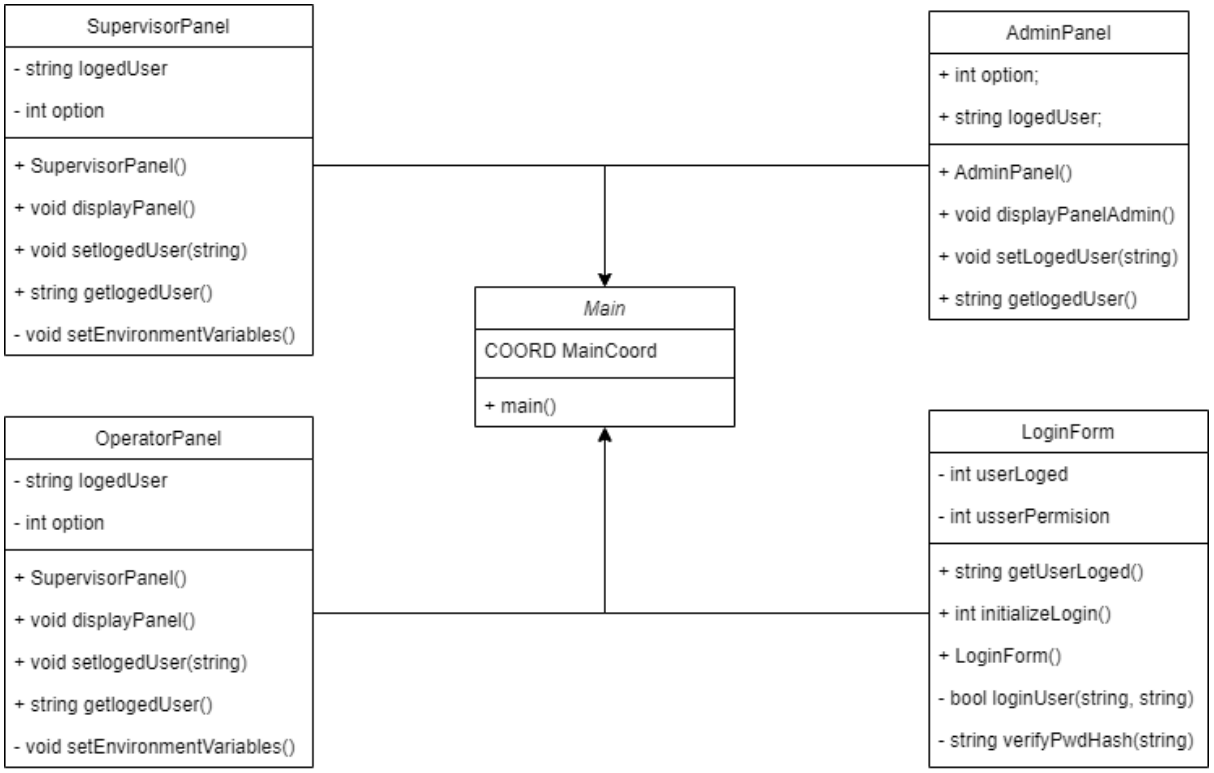


Diagrama de clases composición de la clase PanelAdmin

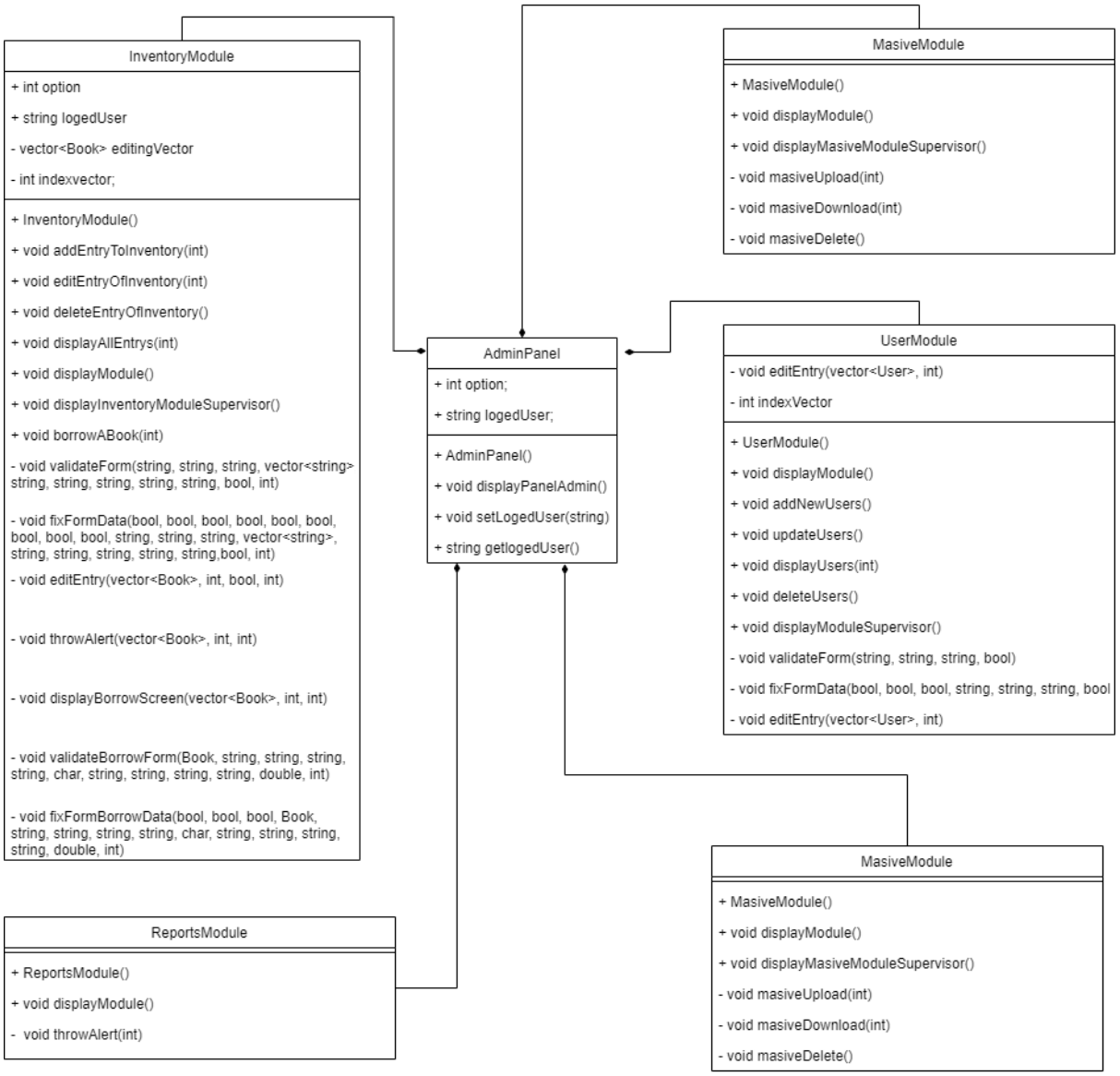


Diagrama de clases composición de la clase InventoryModule

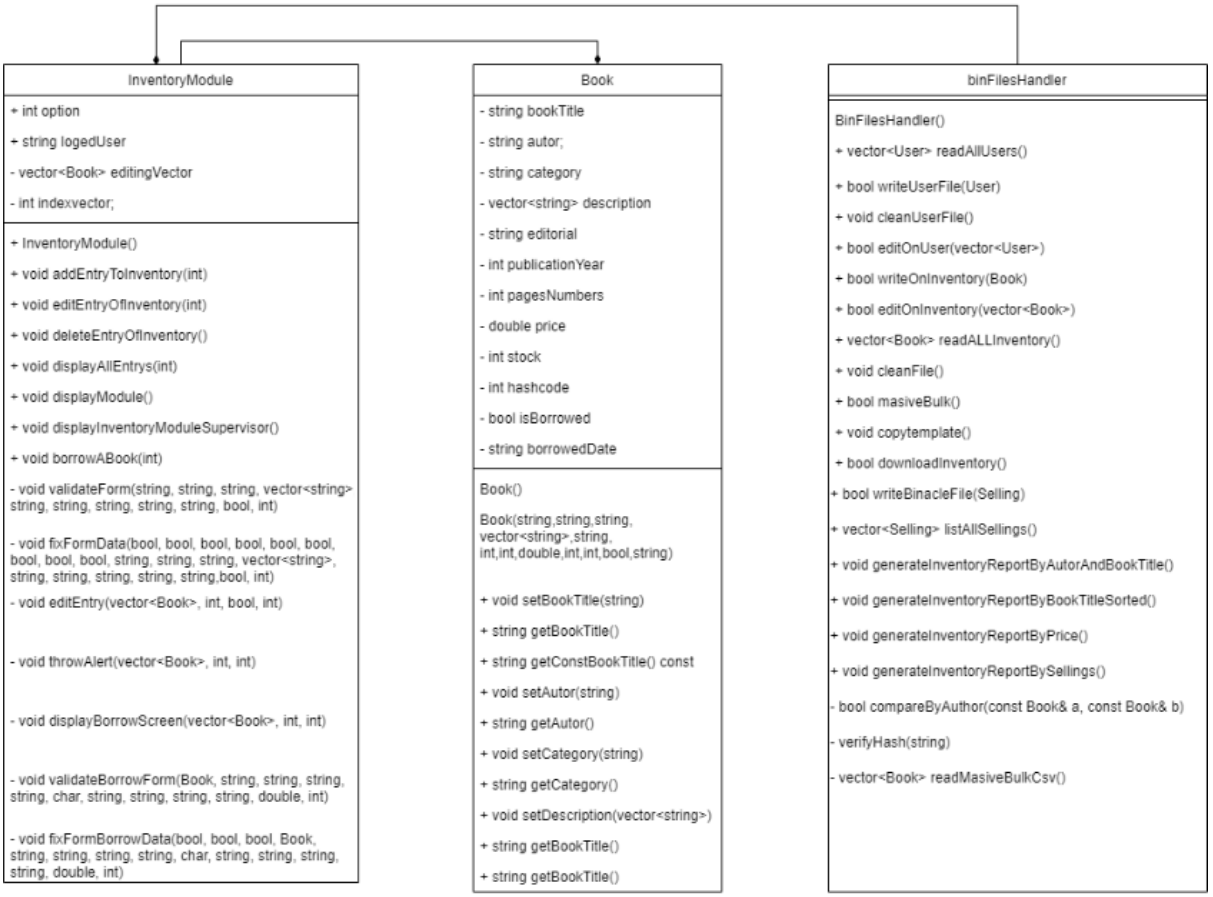


Diagrama de clases composición de la clase masiveModule

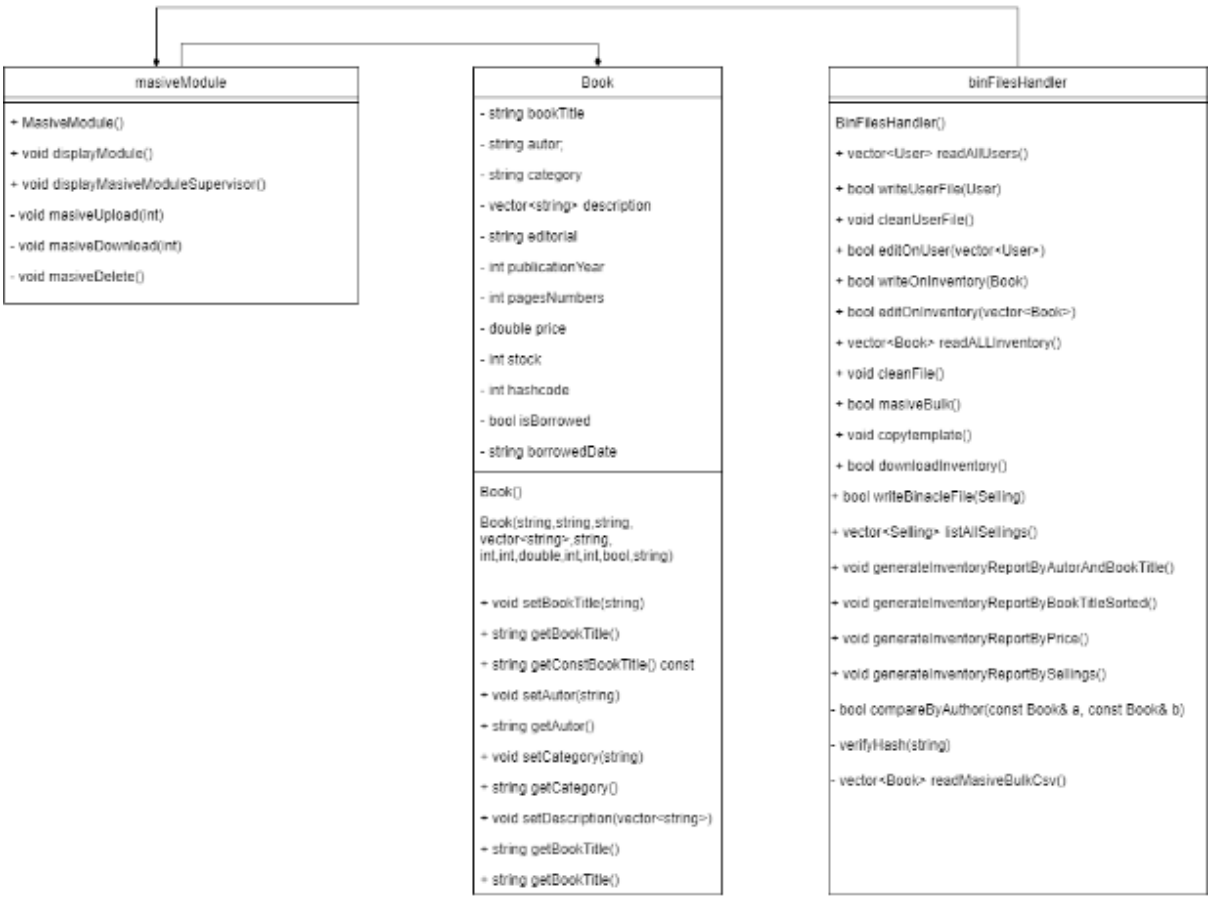


Diagrama de clases composición de la clase userModule

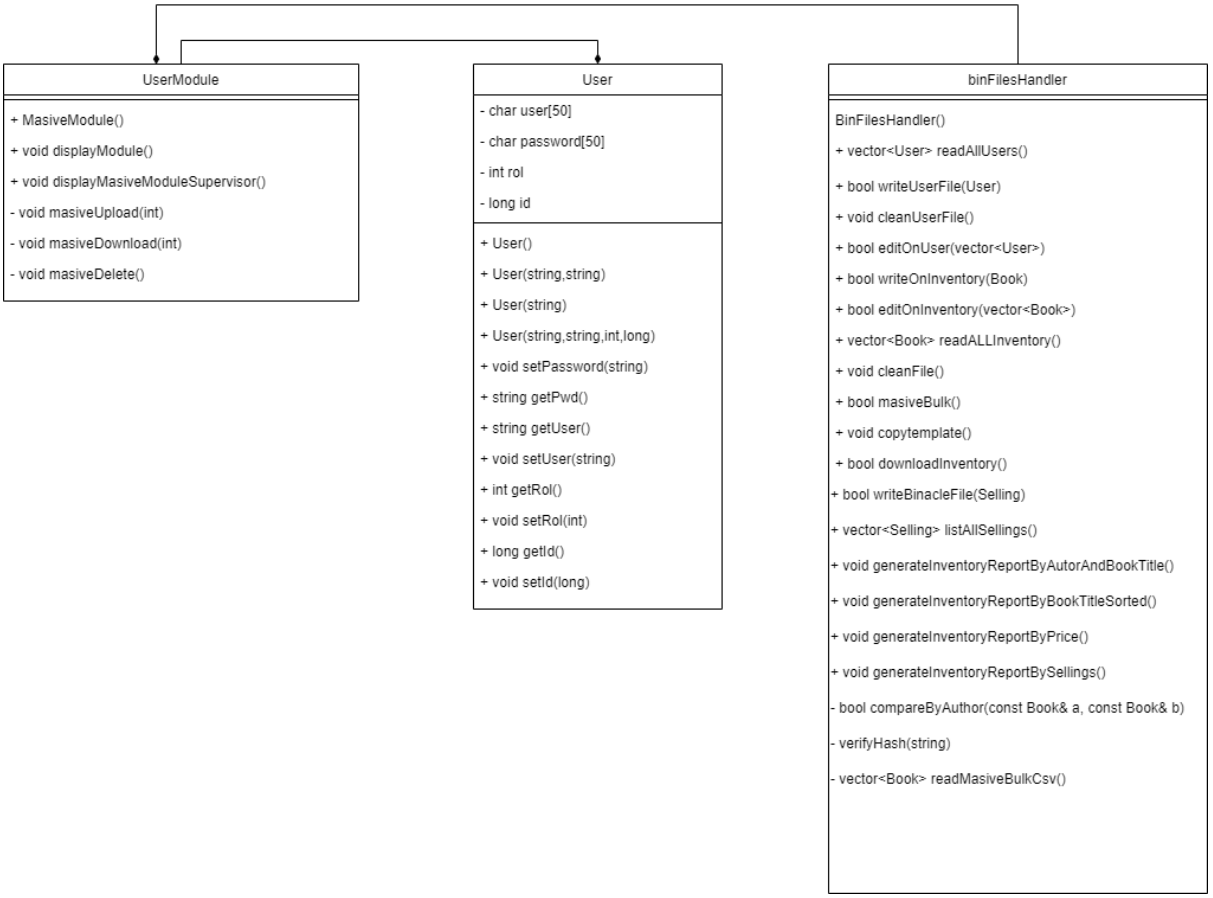


Diagrama de clases composición de la clase reportModule