Manual Técnico

Fase 1

ESTRUCTURA DE DATOS

B+

Edgar Rolando Alvarez Rodriguez
202001144

AGOSTO 2022

INTRODUCCION

El presente manual abordará el tema del funcionamiento del código de la Proyecto (Fase 1) el cual consiste en realizar la lectura de un archivo json así como generar a partir de ella diversas listas, gráficos, entre otras.

El manual esta dirigido a técnicos y programadores que manejan el soporte del software y puedan darle mantenimiento, añadir funcionalidades al programa o revisar el propósito de los archivos que componen el programa.

FUNCIONAMIENTO DEL SISTEMA

interfaz:

Se genera una interfaz en consola, un menú que se repite hasta que se escoge salir de el. Este posee submenús para cada una de sus opciones y muestra en consola los resultados en texto, y para los graficos este mismo inicia la aplicación predeterminada para mostrar las imágenes.

```
********** MENU **********
 1. Carga Masiva
 2. Registrar Usuario
 3. Login
 4. Reportes
 0. SALIR del juego
Ingrese una opcion a ejecutar: 0
```

FUNCIONAMIENTO DEL SISTEMA

interfaz (Código):

Se genera una interfaz en consola por medio del usos de un do-while y swich para mostrar y navegar en el menu ingresando la opción a elegir manualmente.

```
do
{
    // FUNCIONA
    // pr prueba;
    // prueba.Mostrar(3);
    cout << "********** MENU **********\n";
    cout << " 1. Carga Masiva \n";
    cout << " 2. Registrar Usuario \n";
    cout << " 3. Login \n";
    cout << " 4. Reportes \n";
    cout << " 0. SALIR del juego \n";

    cout << "Ingrese una opcion a ejecutar: ";
    cin >> opcion;
```

Para navegar por el menu principal se utilizan numeros para ingresar a cada opcion

```
case 3:{

   cout << " 3. Login \n";
   cout << "\nIngrese su nombre de usuario: ";
   cin >> numberUser;
   cout << "Ingrese su contraseña: ";
   cin >> contrass;

   bool encontrado = listaCircu.BuscarUsuario(numberUser,contrass);
   int monedas = listaCircu.BuscarUsuario(numberUser,contrass);
   // cout << "MONEDAS: " << monedas;
   if (encontrado == 1){
       do
       {
           cout << "\n USUARIO ENCONTRADO\n";
           cout << "    a. Editar Informacion \n";
           cout << "    b. Eliminar Cuenta \n";
           cout << "    c. Ver el Tutorial \n";
           cout << "    d. Ver articulos de la tienda \n";
           cout << "    e. Realizaqr Movimientos \n";
           cout << "    f. Salir al menu principal \n";
           cout << "Ingrese una opcion: ";
           cin >> opcionLogin;

           switch (opcionLogin)
           {
```

Mientras que para navegar por los submenus se utilizan letras minusculas.

CARGA DE DATOS

LECTURA DE ARCHIVOS:

Para ingresar los archivos JSON al sistema en esta ocasion se opto por la opcion de ingresar el nombre del archivo a leer manualmente en el sistema, mediante consola, por ello el archivo en cuestion debe estar ubicado en la carpeta origen donde se ubican los archivo del programa.

Luego de ello se redirigieron los datos a otras funciones para que estas retornaron los datos necesarios según las necesidades del sistema.

```
int lectorJson(){

    cout << "\nIngrese el nombre del archivo a leer:  ";
    string nameArchivo;
    cin >> nameArchivo;

    nameArchivo = nameArchivo + ".json";

    // Usando el fstream para tomar la ruta del archivo
    ifstream file(nameArchivo);

    if (file.fail())
    {
       cout <<"\nERROR - archivo dañado o no encontrado\n";
    }else {

    Json::Value actualJson;
    Json::Reader reader;

    // Usando Reader para parsear el Json osea leerlo
    reader.parse(file, actualJson);
```

CARGA DE DATOS

LECTURA DE ARCHIVOS:

Para leer el archivo JSON y extraer los datos dentro de este se utilizaron los archivos de la librería JSONcpp, estos archivos se ingresaron de manera manual a la carpeta que contiene los archivos de programa y se utilizaron sus funciones segun los requerimientos del sistema.

```
// USUARIOS
const Json::Value users = actualJson["usuarios"];
for(int i = 0; i < users.size(); i = i + 1){

    string nick = users[i]["nick"].asString();

    string edadTemp = (users[i]["edad"].asString());
    int edad = std::atoi(edadTemp.c_str());

    string monTemp = (users[i]["monedas"].asString());
    int monedas = std::atoi(monTemp.c_str());

    string password = users[i]["password"].asString();

    // cout << edad << endl;
    PersonaA usuario(nick,edad,monedas,password);
    // cout << usuario.getNombre() << " "  << usuario.getEdad() << " " << usuario.g

    listaCircu.InsertarFinal(usuario);
    listaSimple_UsuariosASC.InsertarEnOrdenReporte(usuario);
    listaSimple_UsuariosDESC.InsertarEnOrdenReporteDESC(usuario);

}
```

```
const Json::Value tuto = actualJson["tutorial"];
string anchoTemp = tuto["ancho"].asString();
int ancho = std::atoi(anchoTemp.c_str());

string altoTemp = tuto["alto"].asString();
int alto = std::atoi(altoTemp.c_str());

const Json::Value movs = tuto["movimientos"];
for(int i=0; i < movs.size(); i = i + 1){
    string xM = movs[i]["x"].asString();
    string yM = movs[i]["y"].asString();

    string coordenada = xM +  "," + yM;
    // cout << coordenada;

    TutorialA tutorial = TutorialA(ancho,alto,coordenada);
    pilaTutorial.push(tutorial);
}

}
```

CLASES

Persona

Por medio de una clase se instancio el objeto "PersonaA" el cual es el encargado de recaudar los datos de los usuarios para crear un objeto que posteriormente es enviado a una lista de personas.

Articulo

Por medio de una clase se instancio el objeto "ArticuloA" el cual es el encargado de recaudar los datos de los artículos para crear un objeto que posteriormente es enviado a una lista de artículos.

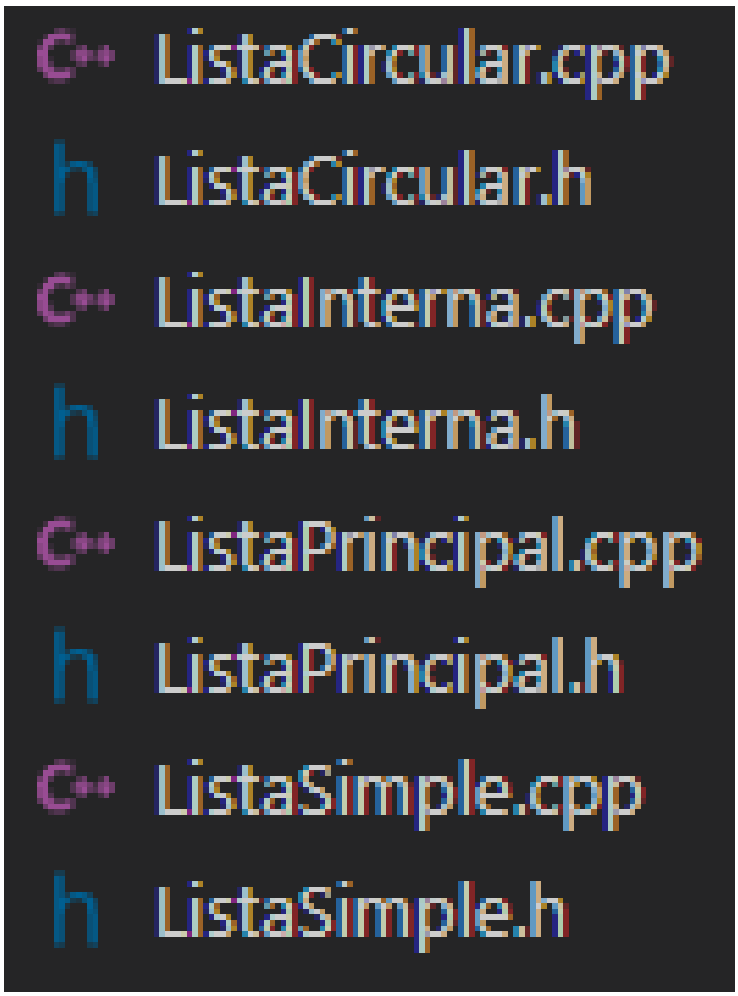
Tuturial

Por medio de una clase se instancio el objeto "TutorialA" el cual es el encargado de recaudar los datos de los artículos para crear un objeto que posteriormente es enviado a una lista de tutorial.

```
#include "TuturialA.h"
#include "PersonaA.h"
#include "ArticuloA.h"
```

Instancias y Funciones

Se crearon distintos archivos para el correcto orden y manejo de las funciones del sistema, de manera que cada archivo se encarga de distintos métodos de ordenamiento y creación de una lista por medio de punteros.



GENERAR LOS GRAFICOS

Para generar los graficos de cada una de las listas que crea el sistema se utilizo Graphviz que es una librería que nos permite generar graficos por medio de su lenguaje.

Para cada lista se instancio una funcion dentro de su respectivo archivo, estas funciones son las encargadas de crear el archivo de la lista solo si son llamadas desde el menu principal.

Todos las funciones siguen la estructura que se presenta a continuación con sus leves modificaciones segun corresponda.

```
void ListaCircular::GrafoASCUsuarios() {
    string dot = "";
    dot = dot + "\ndigraph G {\n";
    dot = dot + "label=\"Lista de Usuarios\";\n";
    dot = dot + "node [shape=box];\n";

    nodocircu*actual = Inicio;
    dot = dot + "//agregar nodos\n";

    while (actual != NULL) {
        // cout <<"[" << (aux->p1).getEdad() << "]->";
        dot = dot + "U" + ((actual->p1).getNombre()) + "[label=\"" + " Nombre: "
        actual = actual->sig;
    }

    dot = dot + "//Enlazar imagenes\n";
    dot = dot + "{rank=same;\n";
    /**CONECTANDO NODOS DE PRINCIPIO A FIN**/
    actual = Inicio;
    while (actual != NULL) {
        dot = dot + "U" + ((actual->p1).getNombre());
        if (actual->sig != NULL) {
            dot = dot + "->";
        }
    actual = actual->sig;
```

```
//------->escribir archivo
ofstream file;
file.open("UsuarioOrdenAsc.dot");
file << dot;
file.close();

//------->generar png
system(("dot -Tpng UsuarioOrdenAsc.dot -o  UsuarioOrdenAsc.png"));

//------>abrir archivo
system(("UsuarioOrdenAsc.png"));
```