Manual

Técnico

Fase 1

USAC Games, Batalla naval

Contenido

Introducción

El presente documento describe los aspectos técnicos informáticos del USAC Games, Batalla naval, creado en lenguaje C++, el programa este compuesto por distintos componentes desarrollados en consola, de forma que lo vuelven intuitivo para el usuario, el documento busca familiarizar todos los aspectos técnicos del programa, como lo es su funcionamiento correcto de la aplicación, como la aplicación de los distintos flujos que el sistema crea.

Lugar de realización: Guatemala

Fecha:24/08/2022

Responsable de elaboración: Nataly Saraí Guzmán Duarte

Información destacada

La realización de la fase 1 de dicho proyecto fue de aproximadamente 10 días, cuya realización fue separada entre menús, cada función se realizó aproximadamente 2 días.

Objetivos y alcances del sistema

- Dar a conocer al programada la forma en que se realizó el programa desde su inicio cuando se instaló C++, mostrando específicamente el código y explicación de porque se realizó cada bloque de código.
- Realizar un programa para que como estudiante practique con el lenguaje C++.
- Construir aplicaciones que se ejecuten por consola con el ejecutable, viéndose de forma ordenada e intuitiva con el usuario.
- Utilizar listas dinámicas sin utilizar las listas del lenguaje.
- Realizar un juego el cual sea intuitivo con el usuario, y tenga muchas opciones de movimientos y compra de artículos.

Requerimientos

- RAM: 1 GB (mínimo).
- ROM: 250 MB (mínimo).
- Arquitectura x32 bits o x64 bits.
- Sistema operativo: Windows, Linux, MacOS.
- Lenguaje de programación: C++.

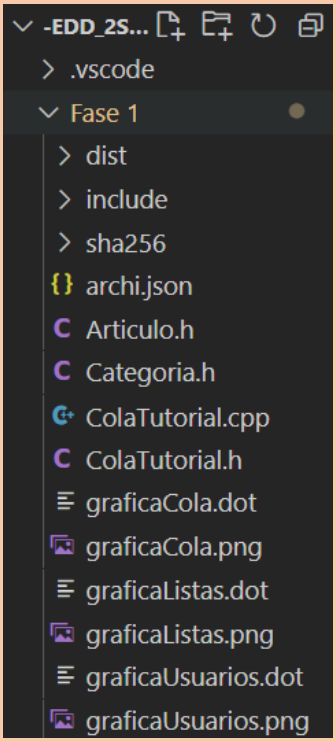
Datos técnicos para la realización del sistema

- Windows 10 Home Single Language.
- Procesador: Intel Core i5.
- RAM: 8.00 GB
- Lenguaje: C++.

Lógica del programa

El juego USAC Games, Batalla naval está constituido por medio de consola creada en el lenguaje de programación C++ es simple e intuitivo con el usuario mostraremos a continuación las distintas clases que se usaron para su creación.

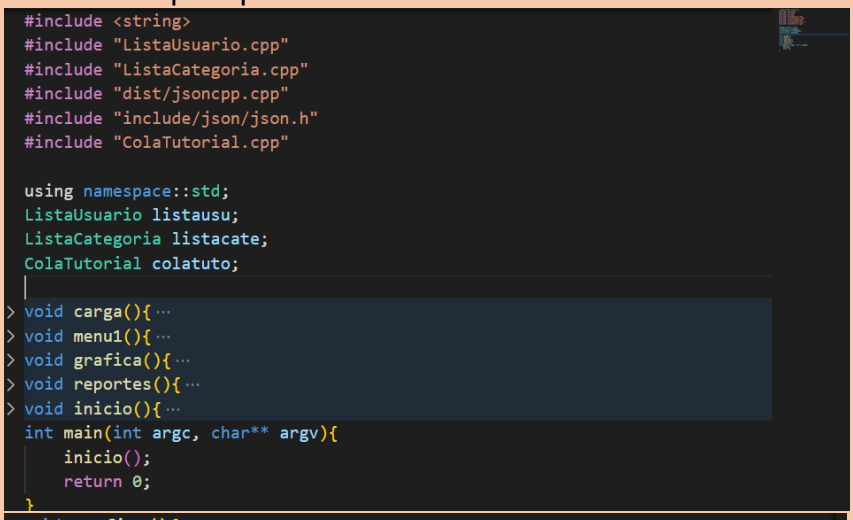
- Listado de clases creadas:



Cada clase cuenta con métodos que nos ayudan a que el sistema funciona de manera eficaz logrando así el uso correcto de la aplicación poniendo en práctica los distintitos condicionales y/o bucles, la utilización de listas dinámicas.

- Descripción de realización de cada clase:
  - Fase 1: Contiene las clases para ejecutar el programa.
  - Main: Contiene el main, menus y opciones para ejecutar el programa.
  - Usuario: Nodo para crear lista de usuarios.
  - Tutorial: Nodo para crear Cola de tutorial.
  - Movimientos: Nodo para crear Cola de movimientos.
  - Articulo: Nodo para crear lista de artículos.
  - Categoria: Nodo para crear cabecera de lista de artículos.
  - ListaUsuario: Clase donde se crea la lista circular doblemente enlazada para guardar los usuarios.
  - ListaCategoria: Clase donde se crea la lista de listas para guardar los artículos.
  - ColaTutorial: Clase donde se crea la cola para guardar los tutoriales.

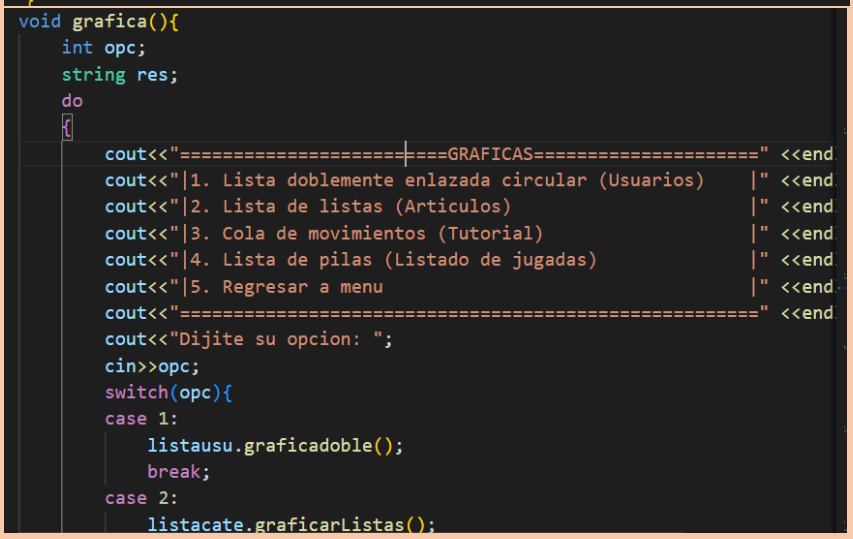
- Clase main:
  - ✓ Constructor: Llamado de función para la creación y/o inicialización del primer menú.
  - ✓ Funciones: creación de los demás menús y funciones de cada opción requerida.
  - ✓ Creación de variables globales para la utilización de estas en cualquier parte de la clase.



```
#include <string>
#include "ListaUsuario.cpp"
#include "ListaCategoria.cpp"
#include "dist/jsoncpp.cpp"
#include "include/json/json.h"
#include "ColaTutorial.cpp"

using namespace::std;
ListaUsuario listausu;
ListaCategoria listacate;
ColaTutorial colatuto;

void carga(){...
void menu1(){...
void grafica(){...
void reportes(){...
void inicio(){...
int main(int argc, char** argv){
    inicio();
    return 0;
}
```



```
void grafica(){
    int opc;
    string res;
    do
    {
        cout<<"======================GRAFICAS====================" <<end
        cout<<"|1. Lista doblemente enlazada circular (Usuarios)    |" <<end
        cout<<"|2. Lista de listas (Articulos)                      |" <<end
        cout<<"|3. Cola de movimientos (Tutorial)                   |" <<end
        cout<<"|4. Lista de pilas (Listado de jugadas)              |" <<end
        cout<<"|5. Regresar a menu                                  |" <<end
        cout<<"======================================================" <<end
        cout<<"Dijite su opcion: ";
        cin>>opc;
        switch(opc){
        case 1:
            listausu.graficadoble();
            break;
        case 2:
            listacate.graficarListas();
```

- Nodo Usuario:
  - ✓ Variables públicas: creación de variables para objeto.
  - ✓ Constructor: asignación de variables y punteros para las listas.

```cpp
#include <string>
#include <iostream>
using namespace std;
class Usuario{
    //atributos y metodos
    private:
    public:
        string nick, pass;
        int mon, edad;
        Usuario* siguiente;
        Usuario* anterior;
        Usuario(string nick, string pass,int mon, int edad){
            this->nick= nick;
            this->pass= pass;
            this->mon= mon;
            this->edad= edad;
            this->siguiente=NULL;
            this->anterior=NULL;
        }
};
```

- NodoTutorial:
  - ✓ Variables públicas: creación de variables para objeto.
  - ✓ Constructor: asignación de variables y punteros para la cola.

```cpp
#include <stddef.h>
#include <string>
#include <iostream>
using namespace std;
class Tutorial{
    //atributos y metodos
    public:
        int ancho, alto;
        string indi;
        Tutorial* siguiente;
        //constructor
        Tutorial(int ancho, int alto,string indi){
            this->alto = alto;
            this->ancho = ancho;
            this->indi = indi;
            this->siguiente = NULL;
        }
    private:
};
```

- Nodo Articulo:
  - ✓ Variables públicas: creación de variables para objeto.
  - ✓ Constructor: asignación de variables y punteros para la lista de listas

```cpp
#include <stddef.h>
#include <string>
#include <iostream>

using namespace std;
class Articulo{
    //atributos y metodos
    private:
    public:
        string nombre, src;
        int id, precio;
        Articulo* abajo;
        Articulo(int id, int precio,string nombre, string src){
            this->id= id;
            this->precio= precio;
            this->nombre= nombre;
            this->src= src;
            this->abajo=NULL;
        }
};
```

- Nodo Articulo:
  - ✓ Variables públicas: creación de variables para objeto.
  - ✓ Constructor: asignación de variables y punteros para la lista de listas

```cpp
#include <stddef.h>
#include <string>
#include <iostream>
#include "Articulo.h"

using namespace std;

class Categoria{
    //atributos y metodos
    public:
        string dato;
        Categoria* siguiente;
        Articulo* abajo;
        Categoria(string dato){
            this->dato = dato;
            this->siguiente=NULL;
            abajo;
        }
    private:
};
```

- Lista Usuarios:
  - ✓ Agregarlista: Agrega elementos a la lista circular doblemente enlazada.
  - ✓ Mostarlista: Muestra todos los elementos en la lista circular doblemente enlazada.
  - ✓ Verificar: Busca si el Nick y el password ingresado son los correcots.

```
#include "sha256/sha256.cpp"
#include <fstream>
#include <sstream>
#include <string>
#include <bits/stdc++.h>
using namespace std;
SHA256 probando;
void ListaUsuario::agregarlista(string nick, string pass,int mon, int edad){
void ListaUsuario::mostrarlista(){···
string ListaUsuario:: verificar(string nic, string pass){···

void ListaUsuario::editar(string res,string nick, string pass,int mon, int e

void ListaUsuario::eliminar(string res){···

int ListaUsuario:: obtemonedas(string nic, string pass){···

int ListaUsuario:: obtedad(string nic, string pass){···

void ListaUsuario::ordeascen(){···
```

- Lista Categoria:
  - ✓ Agregarcate: Agrega elementos a la lista simple.
  - ✓ AgregarArti:Crea una lista de listas agregando elementos a ella.
  - ✓ MostrarCate: Muestra las categorías ingresadas.
  - ✓ MostrarArti: Muestra los artículos ingresados según las categoría.

```
#include <iostream>
#include <string>
#include <sstream>
#include "ListaCategoria.h"

using namespace std;

void ListaCategoria::agregarCate(string dato){···

void ListaCategoria::mostrarCate(){···

string ListaCategoria::getCate(string dato){···

void ListaCategoria::agregarArti(int id, string categoria,int precio,string

void ListaCategoria::mostrarArti(){···

int ListaCategoria::getPrecio(int id){···

void ListaCategoria::graficarListas(){···
```

Diagrama general de la aplicación

Muestra el menú principal y tiene diferentes opciones de los menús que lleva a cada menú para su especifica funcionalidad.