Daniel Garcia Brun

1-DAM  17/12/2024

Actividad Guiada: Refactorización en C++

Contenido

[Introducción 2](#_Toc185318290)

[Instrucciones Generales 2](#_Toc185318291)

[Fases de la Actividad 2](#_Toc185318292)

[Fase 1: Versión inicial (todo en main) 2](#_Toc185318293)

[Fase 2: Refactorización con funciones sin parámetros 3](#_Toc185318294)

[Fase 3: Refactorización con parámetros por valor 5](#_Toc185318295)

[Fase 4: Refactorización con valores de retorno 5](#_Toc185318296)

[Fase 5: Refactorización final con paso por referencia 5](#_Toc185318297)

## Introducción

En esta actividad, aprenderás el concepto de refactorización, que consiste en mejorar la estructura interna de un programa sin cambiar su funcionalidad externa. El objetivo es convertir un programa básico en C++ en una versión modular, legible y fácil de mantener, siguiendo una progresión por etapas.

## Instrucciones Generales

Crearás un programa que simule un combate entre un héroe y un enemigo. Ambos personajes tendrán puntos de vida (Hit Points) y un valor de ataque. El programa deberá controlar el flujo del juego utilizando estructuras condicionales y bucles. Finalmente, refactorizarás el código en varias fases.

## Fases de la Actividad

### Fase 1: Versión inicial (todo en main)

Escribe todo el código en el método main. Utiliza variables globales para los datos del héroe y el enemigo. Puedes reusar el código del videojuego creado en el módulo de programación.

#include <iostream>

using namespace std;

string opcion;

int heroh = 2;

int enemyh = 2;

int main() {

while (heroh > 0 && enemyh > 0) {

cout << "Quieres atacar o esquivar?";

cin >> opcion;

if (opcion == "atacar" || opcion == "Atacar" || opcion == "ATACAR") {

enemyh--;

cout << "El enemigo pierde un punto de vida" << endl;

if (enemyh <= 0) {

cout << "Felicidades has ganado el combate!" << endl;

break;

}

heroh--;

cout << "Tu enemigo ataca" << endl;

if (heroh <= 0) {

cout << "Has sido derrotado y la ciudad destruida" << endl;

break;

}

}

else if (opcion == "esquivar" || opcion == "Esquivar" || opcion == "ESQUIVAR") {

cout << "Tu enemigo intenta atacarte pero lo esquivas" << endl;

}

else {

cout << "Algo ha salido mal" << endl;

}

}

return 0;

}

Texto

Descripción generada automáticamente

### Fase 2: Refactorización con funciones sin parámetros

Refactoriza el código dividiéndolo en funciones que no usen parámetros, pero trabajen con variables globales.

Ejemplo de funciones:

* void heroAttackEnemy()
* void enemyAttackHero()
* void printStatus()

#include <iostream>

using namespace std;

string opcion;

int heroh = 2;

int enemyh = 2;

void introduccion() {

cout << "La ciudad esta siendo atacada y no hay mas heroes disponibles, eres la ultima esperanza. Derrota al villano que esta asolando la ciudad." << endl;

}

void ataqueEnemigo() {

heroh--;

cout << "Tu enemigo ataca." << endl;

if (heroh <= 0) {

cout << "Has sido derrotado y la ciudad destruida." << endl;

}

}

void ataqueHeroe() {

enemyh--;

cout << "El enemigo pierde un punto de vida." << endl;

if (enemyh <= 0) {

cout << "Felicidades, has ganado el combate!" << endl;

}

}

void status() {

cout << "A tu enemigo le quedan " << enemyh << " puntos de vida." << endl;

cout << "A tu heroe le quedan " << heroh << " puntos de vida." << endl;

}

void opciones() {

cout << "Quieres atacar, esquivar, ver el estado o huir?" << endl;

cin >> opcion;

}

void logicacombate() {

while (heroh > 0 && enemyh > 0) {

opciones();

if (opcion == "atacar" || opcion == "Atacar" || opcion == "ATACAR") {

ataqueHeroe();

if (enemyh <= 0) {

break;

}

ataqueEnemigo();

if (heroh <= 0) {

cout << "Has perdido, pero no pasa nada, intentalo de nuevo!" << endl;

break;

}

}

else if (opcion == "esquivar" || opcion == "Esquivar" || opcion == "ESQUIVAR") {

cout << "Tu enemigo intenta atacarte, pero lo esquivas." << endl;

}

else if (opcion == "estado" || opcion == "Estado" || opcion == "ESTADO") {

status();

}

else if (opcion == "huir" || opcion == "Huir" || opcion == "HUIR") {

cout << "Huyes; en consecuencia, la ciudad ha sido destruida y miles de personas masacradas." << endl;

break;

}

else {

cout << "Opcion no valida. Intentalo de nuevo." << endl;

}

}

}

int main() {

introduccion();

logicacombate();

return 0;

}

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### Fase 3: Refactorización con parámetros por valor

Elimina las variables globales y utiliza parámetros por valor en las funciones.

### Fase 4: Refactorización con valores de retorno

Elimina las variables globales y utiliza parámetros por valor en las funciones.

### Fase 5: Refactorización final con paso por referencia

Introduce el uso de estructuras y referencias para hacer el código más robusto y modular.