Explicación del Código

Inclusión de Bibliotecas y Archivos

```
#include <iostream>
#include "Map.h"
#include "Troll.h"
#include "GiantSnake.h"
#include "Demon.h"
#include "Slime.h"
```

El código incluye la biblioteca estándar <iostream> para entrada y salida en la consola. También incluye los encabezados de clases personalizadas (Map, Troll, Giant Snake, Demon, Slime) que representan el mapa y diferentes tipos de enemigos.

Generación de Enemigos Aleatorios

```
enemy* generateRandomEnemy() {
    int random = rand() % 4; // Genera un número aleatorio entre 0 y 3
    switch (random) {
        case 0: return new Troll();
        case 1: return new GiantSnake();
        case 2: return new Demon();
        case 3: return new Slime();
    }
    return nullptr; // Caso por defecto, no debería ocurrir
}
```

Esta función genera un enemigo aleatorio de entre cuatro tipos (Troll, Giant Snake, Demon, Slime). La selección se hace utilizando un número aleatorio entre 0 y 3.

Función Principal “main”

```cpp
int main() {
    srand(time(0)); // Semilla para el generador de números aleatorios con la hora actual
    Map map(10, 10); // Crea un mapa de tamaño 10x10
    map.generate(); // Genera el mapa

    char input;
    while (true) {
        map.print(); // Imprime el estado actual del mapa
        std::cout << "Move (w/a/s/d): ";
        std::cin >> input; // Obtiene el movimiento del jugador

        map.movePlayer(input); // Mueve al jugador basado en la entrada
        if (map.checkForEncounter()) { // Verifica si hay un encuentro con un enemigo
            enemy* encounter = generateRandomEnemy(); // Genera un enemigo aleatorio
            system("cls"); // Limpia la consola (Para Windows. Use system("clear") en sistema

            // Batalla con el enemigo hasta que su HP sea 0
            while (encounter->get_HP() != 0) {
                std::cout << "You have encountered an enemy!" << std::endl;
                encounter->print(); // Imprime los detalles del enemigo
                std::cout << "HP: " << encounter->get_HP() << std::endl;
                std::cout << "Select action: " << std::endl;
                std::cout << "Press (A) to attack" << std::endl;
                std::cout << "Press (B) to flee" << std::endl;
                std::cin >> input; // Obt e la acción del jugador
```

```
            if (input == 'A' || input == 'a') { // El jugador ataca
                std::cout << "The enemy has been defeated" << std::endl;
                break;
            }

            if (input == 'B' || input == 'b') { // El jugador huye
                std::cout << "You ran successfully" << std::endl;
                break;
            } else {
                std::cout << "Enter a valid option" << std::endl; // Manejo de entrada inválid
            }
        }
        delete encounter; // Limpia la memoria del enemigo
    }

    if (map.checkForGoal()) { // Verifica si el jugador ha alcanzado la meta
        std::cout << "You have reached the goal, congratulations!!!" << std::endl;
        break; // Sale del bucle del juego
    }
}

return 0; // Termina el programa
```

Descripción Detallada:

1. **Inicialización del Juego**:
    - Se inicializa el generador de números aleatorios con srand(time(0)).
    - Se crea un mapa de 10x10 con Map map(10, 10) y se genera su contenido con map.generate().

2. **Bucle Principal del Juego**:
    - En cada iteración del bucle, el mapa se imprime en la consola.
    - Se solicita al jugador que ingrese una dirección para moverte (w para arriba, a para izquierda, s para abajo, d para derecha).

- Se mueve al jugador basado en la entrada con map.movePlayer(input).

3. **Encuentros con Enemigos**:
    - Si el jugador encuentra un enemigo (map.checkForEncounter()), se genera un enemigo aleatorio con generateRandomEnemy().
    - Se limpia la consola para una mejor experiencia de usuario.
    - Se inicia una batalla donde el jugador puede atacar (A o a) o huir (B o b).
    - La batalla termina cuando el enemigo es derrotado o el jugador huye.
    - Se libera la memoria del enemigo con delete encounter.

4. **Verificación de Meta**:
    - Si el jugador alcanza la meta (map.checkForGoal()), se imprime un mensaje de felicitación y se termina el juego.