Dokumentasi Program Battleship Game

Deskripsi Program

Secara umum, program ini adalah game sederhana yang menggunakan objek-objek kapal player dan musuh yang direpresentasikan oleh kelas **Ship**. Pada game loop, pemain dapat melakukan aksi seperti *attack* dan *move* untuk mengalahkan musuh. Game loop akan berlanjut sampai kesehatan pemain mencapai 0.

Program ini ditulis dalam bahasa C++ dan menggunakan library standar seperti `iostream` dan `cstdlib`. Tujuan dari program ini adalah untuk memperkenalkan konsep dasar dari pemrograman berorientasi objek (OOP).

Instalasi

Tidak diperlukan instalasi khusus untuk menjalankan program ini.

Cara Menggunakan

1. Buka file program battleship.cpp
2. Compile program menggunakan compiler C++ yang tersedia pada komputer Anda
3. Jalankan program yang sudah dicompile
4. Ikuti instruksi yang muncul pada layar untuk bermain game

Informasi Tentang Game

1. Setelah program dijalankan, pemain akan melihat peta dengan simbol '~' yang merepresentasikan lautan.
2. Pemain akan mengontrol kapal player yang direpresentasikan dengan simbol 'T'.
3. Kapal musuh direpresentasikan dengan simbol 'M' dan akan muncul pada posisi acak pada awal permainan. Pemain dapat memilih untuk menyerang musuh dengan input 'A', memindahkan kapal dengan input 'M', atau bertahan dengan input 'S'.
4. Jika pemain berhasil menyerang musuh, maka musuh akan hancur dan akan muncul musuh baru pada posisi acak.
5. Permainan berakhir jika kapal player kehilangan semua health-nya.

Aturan Permainan

1. Map berukuran 5x5 diinisialisasi dengan karakter tilde (~) untuk merepresentasikan wilayah kosong.
2. Terdapat dua objek berupa kapal yaitu PlayerShip dan EnemyShip yang dapat bergerak di dalam map.
3. Player memulai game dengan kapal PlayerShip yang memiliki simbol T, sedangkan kapal musuh EnemyShip memiliki simbol M.
4. Setiap objek kapal memiliki koordinat x dan y, health, max range, dan simbol.
5. Jika player memilih untuk attack (input A), akan dilakukan pengecekan range attack. Jika dalam range, musuh akan diserang dan health musuh berkurang sebesar 10 poin. Jika health musuh habis (<= 0), maka musuh akan ditempatkan di posisi yang acak dan health musuh di-reset menjadi 50.
6. Jika player memilih untuk move (input M), player dapat memilih arah (U/D/L/R) dan dipindahkan sesuai arah tersebut. Jika player mencoba keluar dari map, akan muncul pesan error.
7. Jika player memilih untuk stay (input S), tidak terjadi aksi apa pun dan giliran berikutnya berlanjut.
8. Jika health player habis (<= 0), permainan berakhir dan program berhenti.

Alur Program / Flowchart

1. Program dimulai dengan meng-import library `iostream`, `cstdlib`, dan `ctime`.
2. Membuat kelas **Ship** yang merepresentasikan kapal dan memiliki beberapa method untuk memanipulasi kapal seperti *attack*, *move*, *getHealth*, dan lain-lain.
3. Kelas **Ship** memiliki atribut berupa koordinat *x* dan *y*, *health*, *max_range*, dan *symbol*.
4. Kelas **PlayerShip** dan **EnemyShip** merupakan turunan dari kelas Ship dan memiliki method tambahan, seperti konstruktor untuk mengatur *symbol* kapal.
5. Pada main program, map diinisialisasi dengan simbol '~' dan dimulai game loop.
6. Pada setiap iterasi game loop, map diprint, input player diterima, dan aksi yang sesuai dilakukan.
7. Jika input player adalah 'A' (attack), program akan melakukan pengecekan range dengan memanggil method *calculateDistance*, jika dalam range, musuh akan diserang dengan memanggil method *attack* pada objek player dan update map.
8. Jika input player adalah 'M' (move), program akan menerima input arah yang dituju dan memanggil method *move* pada objek player. Kemudian, map akan diupdate dengan simbol kapal yang sudah berpindah.

9. Program akan melakukan pengecekan apakah musuh sudah mati (health <= 0). Jika sudah mati, objek *enemy* akan dihapus dan diganti dengan objek baru dengan memanggil konstruktor **EnemyShip** dan update map.
10. Game loop akan berakhir jika health player <= 0.
11. Program selesai.

Kelas dan Fungsi yang Digunakan

Kelas Ship

Kelas Ship merepresentasikan kapal dan memiliki beberapa method untuk memanipulasi kapal seperti attack, move, getHealth, dll. Kelas ini memiliki atribut x, y, health, max_range, dan symbol. Method-method yang dimiliki oleh kelas Ship adalah sebagai berikut:

1. Ship() adalah konstruktor yang menginisialisasi atribut x, y, health, dan max_range dengan nilai default 0, 0, 100, dan 2.
2. int getX() mengembalikan nilai atribut x.
3. int getY() mengembalikan nilai atribut y.
4. void setX(int new_x) mengubah nilai atribut x dengan nilai new_x.
5. void setY(int new_y) mengubah nilai atribut y dengan nilai new_y.
6. int getHealth() mengembalikan nilai atribut health.
7. void setHealth(int new_health) mengubah nilai atribut health dengan nilai new_health.
8. int getMaxRange() mengembalikan nilai atribut max_range.
9. void attack(Ship& enemy) menyerang kapal musuh dan mengurangi health kapal musuh sebesar 10. Jika jarak antara kapal player dan kapal musuh lebih besar dari max_range, maka pesan error akan ditampilkan. Method ini mengambil objek Ship musuh sebagai parameter. void printLocation() mencetak lokasi kapal pada konsol dalam format "(x,y)".
10. void move(char direction) memindahkan kapal ke arah yang ditentukan oleh parameter direction (U untuk up, D untuk down, L untuk left, dan R untuk right). Jika kapal bergerak keluar dari map (x < 0 atau x > 4 atau y < 0 atau y > 4), maka pesan error akan ditampilkan.
11. int calculateDistance(Ship& enemy) mengembalikan jarak antara kapal player dan kapal musuh dalam satuan langkah.
12. char getSymbol() mengembalikan simbol kapal.

Kelas PlayerShip dan EnemyShip

Kelas PlayerShip dan EnemyShip adalah turunan dari kelas Ship dan memiliki beberapa method tambahan. Method-method yang dimiliki oleh kelas PlayerShip dan EnemyShip adalah sebagai berikut:

1. PlayerShip() adalah konstruktor yang mengatur simbol kapal player menjadi 'T'.
2. EnemyShip() adalah konstruktor yang mengatur simbol kapal musuh menjadi 'M', serta menginisialisasi atribut x, y, health, dan max_range dengan nilai default yang berbeda dari kelas Ship.

Fungsi main

Pada fungsi main, terdapat beberapa hal yang dilakukan untuk memulai game. Pertama-tama, diinisialisasi seed untuk fungsi random dengan nilai waktu saat ini. Kemudian, objek PlayerShip dan EnemyShip dibuat. Selanjutnya, map diinisialisasi dengan karakter '~' dan posisi kapal player dan kapal musuh ditandai dengan simbol kapal masing-masing.

Setelah inisialisasi, game loop dimulai. Pada setiap iterasi game loop, posisi kapal player dan kapal musuh dicetak pada konsol. Kemudian, pemain diminta untuk memasukkan input berupa arah gerakan kapal player (U, D, L, atau R). Setelah input diterima, kapal player dipindahkan sesuai dengan arah yang dimasukkan. Setelah pemain memindahkan kapalnya, giliran kapal musuh untuk melakukan serangan atau memindahkan posisinya.

Jika health salah satu kapal telah habis, game loop akan diakhiri dan pesan kemenangan atau kekalahan akan ditampilkan sesuai dengan kondisi permainan.

Source Code

```
// Program ini adalah game sederhana yang menggunakan objek-objek berupa kapal
player dan kapal musuh.

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

// Kelas Ship merepresentasikan kapal dan memiliki beberapa method untuk
memanipulasi kapal seperti attack, move, getHealth, dll.
class Ship {
public:
    Ship() {
        x = 0;
        y = 0;
```

```
        health = 100;
        max_range = 2;
    }

    int getX() {
        return x;
    }

    int getY() {
        return y;
    }

    void setX(int new_x) {
        x = new_x;
    }

    void setY(int new_y) {
        y = new_y;
    }

    int getHealth() {
        return health;
    }

    void setHealth(int new_health) {
        health = new_health;
    }

    int getMaxRange() {
        return max_range;
    }

    void attack(Ship& enemy) {
        int distance = calculateDistance(enemy);
        if (distance > max_range) {
            cout << "Error: Enemy ship is out of range." << endl;
        } else {
            enemy.setHealth(enemy.getHealth() - 10);
            cout << "Ship attacked enemy ship! Enemy health is now " <<
enemy.getHealth() << endl;
        }
    }

    void printLocation() {
        cout << "Ship is at (" << x << "," << y << ")" << endl;
    }

    void move(char direction) {
        int new_x = x;
```

```
        int new_y = y;

        if (direction == 'U') {
            new_x--;
        } else if (direction == 'D') {
            new_x++;
        } else if (direction == 'L') {
            new_y--;
        } else if (direction == 'R') {
            new_y++;
        }

        if (new_x < 0 || new_x > 4 || new_y < 0 || new_y > 4) {
            cout << "Error: Cannot move outside the map." << endl;
        } else {
            x = new_x;
            y = new_y;
            cout << "Ship moved to (" << x << "," << y << ")" << endl;
        }
    }

    int calculateDistance(Ship& enemy) {
        int x_diff = abs(x - enemy.getX());
        int y_diff = abs(y - enemy.getY());
        return max(x_diff, y_diff);
    }

    char getSymbol() {
        return symbol;
    }

protected:
    int x;
    int y;
    int health;
    int max_range;
    char symbol;
};

// Kelas PlayerShip dan EnemyShip adalah turunan dari Ship dan memiliki beberapa
method tambahan, seperti konstruktor untuk mengatur symbol kapal.
class PlayerShip : public Ship {
public:
    PlayerShip() {
        symbol = 'T';
    }
};

class EnemyShip : public Ship {
```

```
public:
    EnemyShip() {
        symbol = 'M';
        x = rand() % 5;
        y = rand() % 5;
        health = 50;
        max_range = 1;
    }
};

// Pada main program, map diinisialisasi dan dimulai game loop.
// Pada setiap iterasi game loop, map diprint, input player diterima, dan aksi yang sesuai dilakukan.
int main() {
    srand(time(NULL));
    PlayerShip player;
    EnemyShip enemy;

    char map[5][5];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            map[i][j] = '~';
        }
    }
    map[player.getX()][player.getY()] = player.getSymbol();
    map[enemy.getX()][enemy.getY()] = enemy.getSymbol();

    int enemy_count = 1;
    while (player.getHealth() > 0) {
    // Print map
    cout << "Map:" << endl;
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            cout << map[i][j] << " ";
        }
        cout << endl;
    }

    char input;
    cout << "Enter command (A for attack, M for move, S for stay): ";
    cin >> input;

    // Jika player memilih untuk attack, akan dilakukan pengecekan range, jika dalam range, musuh akan diserang dan map diupdate.
    if (input == 'A') {
        player.attack(enemy);
        if (enemy.getHealth() <= 0) {
            map[enemy.getX()][enemy.getY()] = 'X';
            enemy_count++;
```

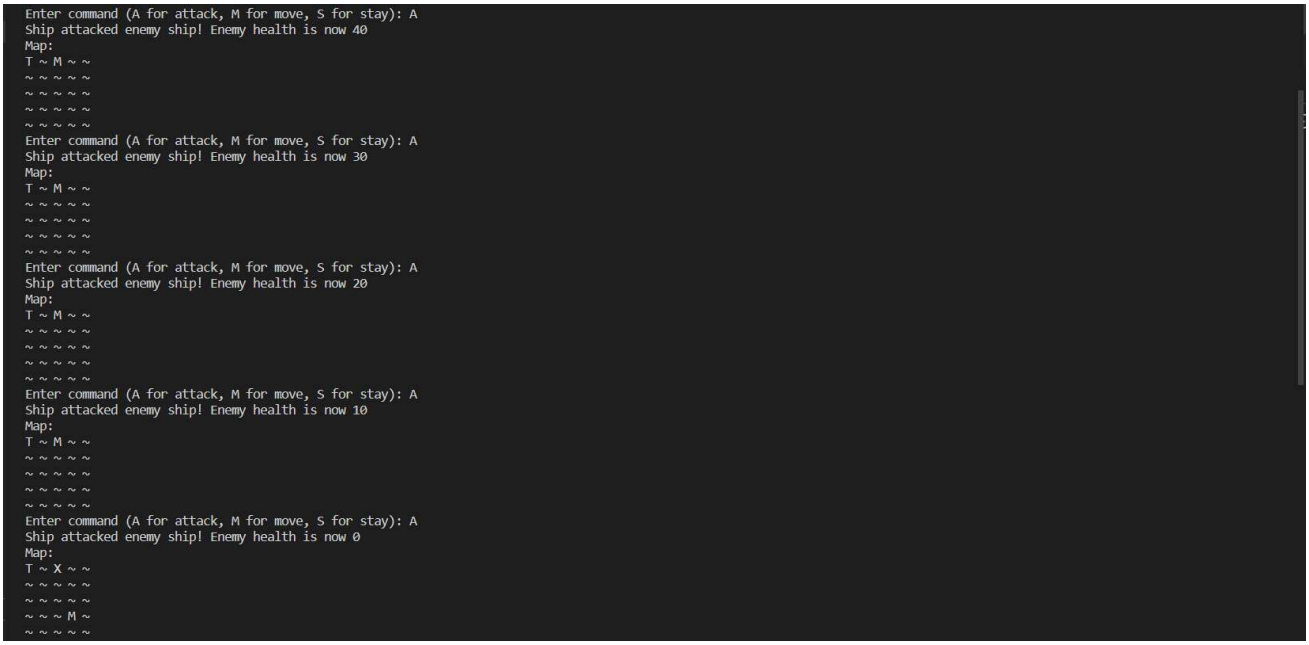
```
            enemy = EnemyShip();
            map[enemy.getX()][enemy.getY()] = enemy.getSymbol();
        }
    // Jika player memilih untuk move, player dipindahkan sesuai input, dan map
diupdate.
    } else if (input == 'M') {
        char direction;
        cout << "Enter direction (U for up, D for down, L for left, R for right):
";
        cin >> direction;
        player.move(direction);
        // Update map
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (i == player.getX() && j == player.getY()) {
                    map[i][j] = player.getSymbol();
                } else if (i == enemy.getX() && j == enemy.getY()) {
                    map[i][j] = enemy.getSymbol();
                } else {
                    map[i][j] = '~';
                }
            }
        }
    // Jika player memilih untuk stay, tidak terjadi apa-apa.
    } else if (input == 'S') {
        // Do nothing
    // Jika player memasukkan input yang tidak valid, akan muncul pesan error.
    } else {
        cout << "Error: Invalid command." << endl;
    }
}
// Game akan berakhir ketika health player mencapai 0. Jumlah musuh yang berhasil
dihancurkan akan ditampilkan.
cout << "Game over. You destroyed " << enemy_count << " enemy ships." << endl;

return 0;
}
```

Test Case

Menu Utama

Stay Command

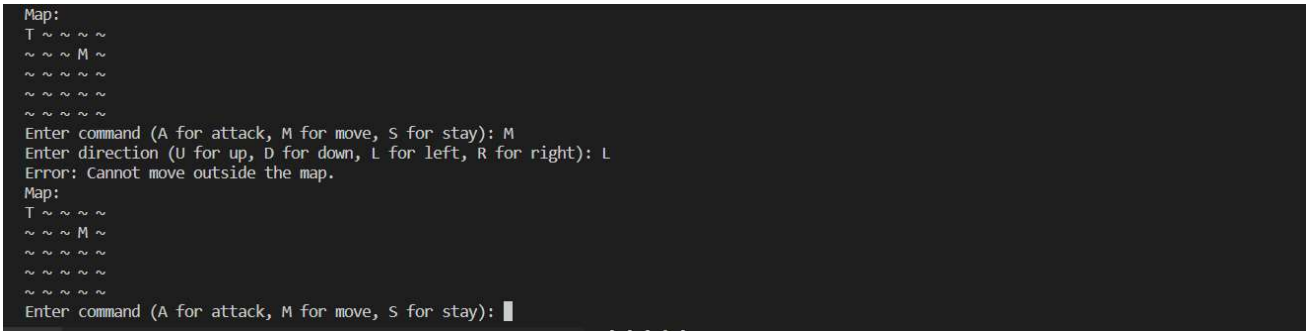
Attack Command



Attack Out of Range

Move Command

Move Command Out of Bounds



Wrong Input