Universidade Federal de Mato Grosso

Bacharelado em Ciência da Computação

Semestre: 2024/1

Programação 2

Lista 6

Exercício 1

No jogo de cartas "Yu-Gi-Oh!", os jogadores podem combinar cartas para criar novas cartas com habilidades aprimoradas. Essas combinações, chamadas de fusões, podem envolver diferentes tipos de cartas, como monstros, magias e equipamentos. Abaixo está uma lista de algumas fusões possíveis:

- Koumori Dragon (Monstro) + Saggi the Dark Clown (Monstro) = Reaper of the Cards (Monstro)
- Beast Fangs (Equipamento) + Beast Fangs (Equipamento) = Eatgaboon (Armadilha)
- Dark Magic (Magia) + Dragon (Monstro) = Blackland Fire Dragon (Monstro)
- Dragon (Monstro) + Machine (Equipamento) = Metal Dragon (Monstro)
- Dark Spellcaster (Magia) + Ryu-Kishin (Monstro) = Ryu-Kishin Powered (Monstro)
- Dragon Capture Jar (Magia) + Mooyan Curry (Magia) = Dragon Capture Jar (Magia)
- Hinotama (Magia) + Hinotama (Magia) = Final Flame (Magia)
- Dark Energy (Magia) + Acid Trap Hole (Magia) = Dark Hole (Magia)
- Electro-whip (Equipamento) + Metalmorph (Equipamento) = Raigeki (Magia)
- Goblin's Secret Remedy (Equipamento) + Goblin's Secret Remedy (Equipamento) = Soul of the Pure (Armadilha)
- Legendary Sword (Equipamento) + Sword of Dark Destruction (Equipamento) = Kunai with Chain (Equipamento)
- Legendary Sword (Equipamento) + Sparks (Magia) = Salamandra (Equipamento)
- Machine Conversion Factory (Equipamento) + Silver Bow & Arrow (Equipamento) = Winged Trumpeter (Equipamento)
- Castle of Dark Illusions (Equipamento) + Dark-Piercing Light (Magia) = Bright Castle (Equipamento)
- Elegant Egotist (Magia) + Horn of the Unicorn (Equipamento) = Curse of Tri-Horned Dragon (Monstro)

Objetivo

Seu objetivo é implementar um programa que simule todas essas fusões de cartas utilizando a sobrecarga do operador `+` como uma função amiga. Para isso, você deverá:

- Criar uma classe base `Card` que representará uma carta com um nome e um tipo (monstro, magia, equipamento, etc.).

- Criar classes derivadas para representar diferentes tipos de cartas: `MonsterCard`, `SpellCard` e `EquipCard`.
- Implementar a sobrecarga do operador `+` como uma função amiga das classes, permitindo a fusão de duas cartas e retornando uma nova carta com base na tabela de fusões fornecida.
- Implementar um mapa (ou dicionário) para armazenar as combinações de fusões e seus resultados.

Exemplo de Uso

```
int main() {
    // Exemplo de fusões
    MonsterCard monstro1("Koumori Dragon");
    MonsterCard monstro2("Saggi the Dark Clown");

    SpellCard magia1("Dragon Capture Jar");
    SpellCard magia2("Mooyan Curry");

    EquipCard equip1("Legendary Sword");
    EquipCard equip2("Sword of Dark Destruction");

    Card fusao1 = monstro1 + monstro2;
    Card fusao2 = magia1 + magia2;
    Card fusao3 = equip1 + equip2;

    std::cout << "Resultado da Fusão 1: " << fusao1.getName() << " (" << fusao1.getType() << ")\n";
    std::cout << "Resultado da Fusão 2: " << fusao2.getName() << " (" << fusao2.getType() << ")\n";
    std::cout << "Resultado da Fusão 3: " << fusao3.getName() << " (" << fusao3.getType() << ")\n";

    return 0;
}
```

Saída Esperada

A saída esperada desta main acima:

```
Resultado da Fusão 1: Reaper of the Cards (Monstro)
Resultado da Fusão 2: Dragon Capture Jar (Magia)
Resultado da Fusão 3: Kunai with Chain (Equipamento)
```

Exercício 2: Vetores 2D

Neste exercício, você deve implementar a classe `Vector2D` que representa um vetor bidimensional. A classe deve incluir a sobrecarga dos operadores `+`, `-`, `==`, `<<` e `>>`. A seguir está um exemplo de código para testar a implementação:

```
int main() {
    Vector2D v1(3.5, 2.0);
    Vector2D v2(1.5, 4.0);

    // Adiciona dois vetores
    Vector2D v3 = v1 + v2;
    std::cout << "Soma de v1 e v2: " << v3 << std::endl;

    // Subtrai dois vetores
    Vector2D v4 = v1 - v2;
    std::cout << "Diferenca entre v1 e v2: " << v4 << std::endl;

    // Compara dois vetores
    if (v1 == v2) {
        std::cout << "v1 e v2 sao iguais." << std::endl;
    } else {
        std::cout << "v1 e v2 sao diferentes." << std::endl;
    }

    // Leitura de vetor
    Vector2D v5;
    std::cout << "Digite um vetor (formato: x y): ";
    std::cin >> v5;
    std::cout << "Vetor lido: " << v5 << std::endl;

    return 0;
}
```

Saída Esperada:

```
Soma de v1 e v2: (5.0, 6.0)
Diferença entre v1 e v2: (2.0, -2.0)
v1 e v2 são diferentes.
Digite um vetor (formato: x y): 2.0 3.0
Vetor lido: (2.0, 3.0)
```