

MASTERMIND

André Gomes Lomba Júnior
Engenharia Mecatrônica
Universidade Federal de Santa Catarina
Joinville-SC
andre.lomba.jr@hotmail.com

Luana Aparecida Gomes
Engenharia Mecatrônica
Universidade Federal de Santa Catarina
Joinville-SC
luana00gomes@gmail.com

Resumo — O seguinte trabalho tem como objetivo construir o jogo Mastermind, sendo este realizado com base na programação orientada ao objeto por meio da linguagem C++.

Palavras-chave — Mastermind, orientação ao objeto, templates, composição, polimorfismo, herança.

Introdução

Este relatório refere-se ao desenvolvimento do jogo Mastermind. Sua implementação foi realizada em C++, linguagem na qual pudemos usufruir de classes, composição de funções, heranças, templates e polimorfismos para criar e definir objetos.

Esse jogo trata-se de um desafio em que um jogador, que neste trabalho chamaremos de Hacker, tenta adivinhar um código de cores definido anteriormente pelo *mastermind*.

Regras do jogo:

-Primeiramente são definidos os parâmetros do jogo: tamanho da senha a ser desvendada, número máximo de tentativas, quantidade de cores disponíveis e uma possível repetição entre elas são definidos para formar a senha. Esses parâmetros podem ser ajustados pelo usuário que escolher jogar no modo personalizado ou podem ser pré-definidos pelo programa caso o usuário queira escolher um determinado nível de dificuldade. Cada partida pode ser jogada por um ou dois usuários, sendo que caso exista apenas um jogador, ele será obrigatoriamente o *hacker*.

-Assim que todos os parâmetros já estiverem definidos, uma senha é determinada pelo Mastermind e então inicia-se o jogo;

-O usuário deverá escolher uma senha com o tamanho já definido, utilizando as cores também já estabelecidas;

-Para cada tentativa, o *mastermind* retornará peças pretas em mesma quantidade que o número de peças que o *hacker* escolher com a cor certa na posição certa;

-O *mastermind* retornará peças brancas em mesma quantidade de peças que o *hacker* escolher com a cor certa, mas na posição errada,;

-Após retornar o resultado, o usuário joga novamente. Isso se repete até o momento em que ele acerte a senha e vença o jogo ou acabe acabe seu número de tentativas e perca o jogo.

Desenvolvimento

Foram feitas 6 classes para criar objetos que serão utilizados no jogo:

Classe Partida: Esta classe é a base de todo o código. Ela é responsável por definir os atributos da partida (tamanho de código, número de cores, repetição ou não de cores, número de palpites e nível de dificuldade). Além disso, possui métodos para encontrar e setar valores para esses parâmetros (métodos getters e setters). Ela também possui o método *inicia()*. Quando este método é chamado, ele pede ao usuário os parâmetros da partida.

Classe Chave: Esta é uma classe herdeira de Partida. É responsável por armazenar um vetor de cores com o tamanho escolhido pelo usuário e também seus palpites realizados em cada rodada. Esta classe tem o método *setSenha()* que na classe Partida foi definida como virtual. Posteriormente será feito um overload deste método para os casos em que a partida tenha 1 ou 2 jogadores. Este tipo de implementação é considerado polimorfismo. Ademais, ela possui o método *freeAtributos()* que deleta os ponteiros criados no construtor de um objeto chave. Ela ainda possui um método *getT_jogador()* que é implementado com o uso de template onde na chamada deste método deve ser retornado o objeto *Um_Jogador* ou *Dois_Jogadores* (de acordo de como foi definido pelo usuário). O objeto do tipo classe é o primeiro criado na *main()* com o objetivo de receber os parâmetros de jogo, a partir do método *inicia()* da classe base.

Classe Um_Jogador e Classe Dois_Jogadores: Estas classes herdam a classe Chave (e consequentemente a classe Partida). Elas são responsáveis por setar a senha (no overload do método *setSenha()*), e fazer a interação de cada rodada, onde o computador pede ao *hacker* para setar um palpite (usando o método herdado da classe Chave) e faz a correção do palpite, pelo método *corrigePalpite()*, seja pedindo os dados ao *mastermind* (quando houver dois jogadores), ou fazendo isto por si próprio (quando a partida tiver um jogador).

Também é herdeira da classe Chave e possui os mesmos métodos da classe *Um_Jogadore*. A diferença consiste no fato da classe *Dois_Jogadores* ser utilizada quando o usuário escolhe jogar em dois.

Classe Jogo: Nesta classe são feitas composições com a classes chave e uma classe do tipo T, que é um template. Na construção do objeto jogo, elas são passadas por parâmetro. O tipo T assumirá a forma de *Um_Jogador* ou *Dois_Jogadores*, dependendo da quantidade de jogadores na partida.

A classe Jogo possui o método *start()*. Nele é chamado o método *iniciaT()* que passará para o atributo *t_jogador* os dados do objeto chave. Após isso, o objeto *t_jogador* construirá o vetor de cores

e iniciará a senha a ser desvendada. Por fim, o método start() fará interações sucessivas para receber e verificar os palpites dados e decidirá quem ganhou o jogo.

Discussões

A realização do trabalho exige um bom estruturamento para que o código fique organizado e simples de entender. Na seção desenvolvimento, discute-se sobre o uso de características da linguagem, como por exemplo classes; e isso será analisado novamente a seguir.

O arquivo "teste.h" por exemplo é responsável pela declaração das classes junto com os atributos dos objetos. Neste arquivo também há o uso de herança, polimorfismo e templates.

Conclusão

Após a realização do trabalho, pode-se dizer que o mesmo foi proveitoso para um melhor entendimento de programação orientada ao objeto, afinal, fez-se a utilização de conhecimentos teóricos obtidos em sala de aula para criar um jogo.

Além disso, também foi possível notar que a interação entre homem e computador é um recurso poderoso pois por meio do mesmo foi possível estabelecer uma conexão com o "mundo abstrato" e trazê-lo, mesmo que pouco, para o mundo real.

ANEXOS

Imagem 1 - Classe Partida

```
1 #include <string>
2 #include <iostream>
3
4 using namespace std;
5
6 class Chave;
7
8 class Partida{
9     protected:
10         int numCores;
11         int tamCodigo;
12         bool coresRepitidas;
13         int numJogadores;
14         int maxPalpites;
15         int level;
16         int *senha;
17
18     public:
19         Partida(const int &numCores, const int &tamCodigo, const int &maxPalpites, const bool
20 &coresRepitidas, const int &numJogadores, const int &level);
21         Partida();
22         void inicia();
23
24         void setNumCores(const int &numCores);
25         void setTamCodigo(const int &tamCodigo);
26         void setMaxPalpites(const int &maxPalpites);
27         void setCoresRepitidas(const bool &coresRepitidas);
28         void setNumJogadores(const int &numJogadores);
29         void setLevel(const int &level);
30
31         int getNumCores();
32         int getTamCodigo();
33         int getMaxPalpites();
34         bool getCoresRepitidas();
35         int getNumJogadores();
36         int getLevel();
37
38         void freeSenha();
39         virtual void setSenha()=0;
40 };
41
```

Fonte: Autores

Imagem 2 - Classe Chave

```
41 class Chave: public Partida{
42     protected:
43         string *cores;
44         int *palpite;
45         int numPalpites=0;
46         int numPretas=0;
47         int numBranças=0;
48
49     public:
50         Chave();
51         void setCores();
52         void setSenha();
53         void setPalpite();
54
55         int* getPalpite();
56         int* getSenha();
57         int getNumPretas();
58         int getNumBranças();
59
60         void freeAtributos();
61
62     template <typename T>
63         T getT_jogador();
64 };
65
```

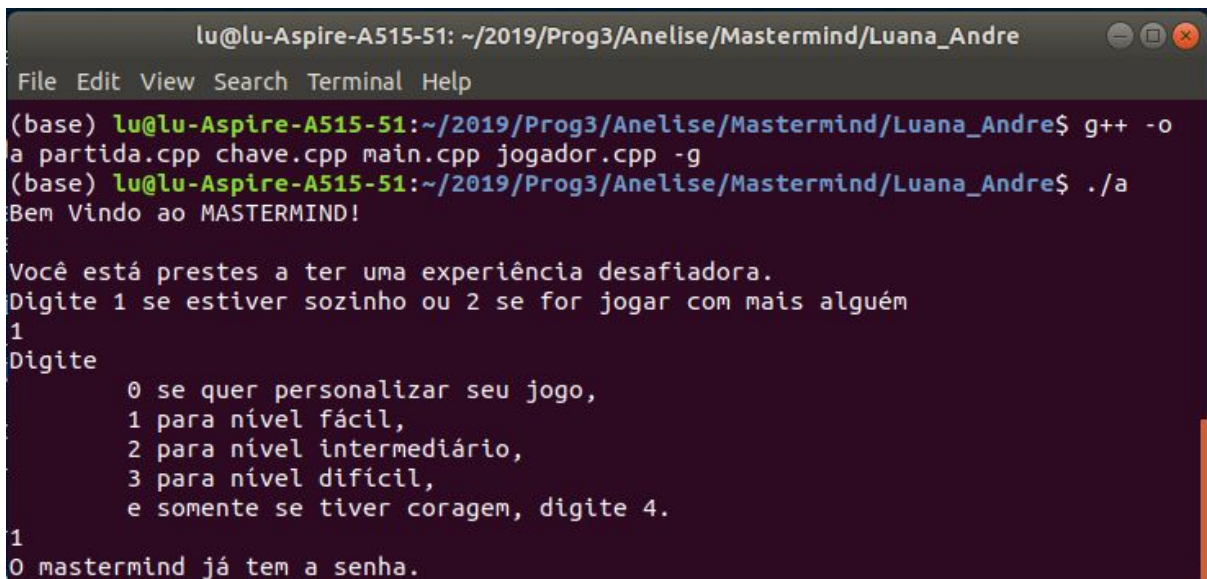
Fonte: Autores

Imagem 2 - Classe Um_Jogador e Dois_Jogadores

```
77 class Um_Jogador: public Chave{
78     public:
79         void setSenha();
80         void interage();
81         void corrigePalpite();
82 };
83
84
85 class Dois_Jogadores: public Chave{
86     public:
87         void interage();
88         void setSenha();
89         void corrigePalpite();
90 };
```

Fonte: Autores

Imagem 3 - Jogo funcionando



```
lu@lu-Aspire-A515-51: ~/2019/Prog3/Anelise/Mastermind/Luana_Andre
File Edit View Search Terminal Help
(base) lu@lu-Aspire-A515-51:~/2019/Prog3/Anelise/Mastermind/Luana_Andre$ g++ -o
a partida.cpp chave.cpp main.cpp jogador.cpp -g
(base) lu@lu-Aspire-A515-51:~/2019/Prog3/Anelise/Mastermind/Luana_Andre$ ./a
Bem Vindo ao MASTERMIND!

Você está prestes a ter uma experiência desafiadora.
Digite 1 se estiver sozinho ou 2 se for jogar com mais alguém
1
Digite
    0 se quer personalizar seu jogo,
    1 para nível fácil,
    2 para nível intermediário,
    3 para nível difícil,
    e somente se tiver coragem, digite 4.
1
0 mastermind já tem a senha.
```

Fonte: Autores

Imagem 4 - Jogo funcionando

```
lu@lu-Aspire-A515-51: ~/2019/Prog3/Anelise/Mastermind/Luana_Andre
File Edit View Search Terminal Help
Restam 10tentativas.

*****
Hacker, faça seu chute.
Cores:
    (0) Verde
    (1) Preto
    (2) Branco
    (3) Rosa
Escolha a cor da posicao 0.
0
Cores:
USADO    (0) Verde
         (1) Preto
         (2) Branco
         (3) Rosa
Escolha a cor da posicao 1.
1
Cores:
USADO    (0) Verde
USADO    (1) Preto
         (2) Branco
         (3) Rosa
```

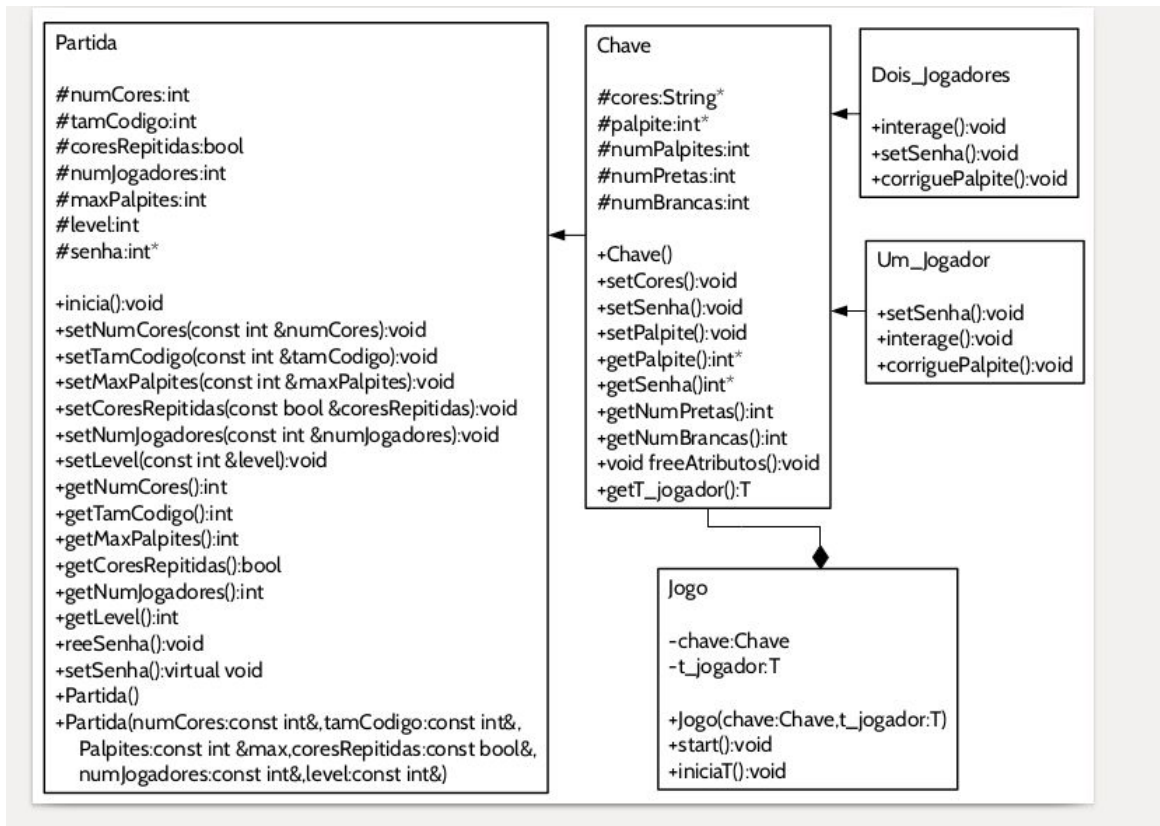
Fonte: Autores

Imagem 5 - Jogo funcionando

```
lu@lu-Aspire-A515-51: ~/2019/Prog3/Anelise/Mastermind/Luana_Andre
File Edit View Search Terminal Help
    (2) Branco
    (3) Rosa
Escolha a cor da posicao 1.
1
Cores:
USADO    (0) Verde
USADO    (1) Preto
         (2) Branco
         (3) Rosa
Escolha a cor da posicao 2.
2
Cores:
USADO    (0) Verde
USADO    (1) Preto
USADO    (2) Branco
         (3) Rosa
Escolha a cor da posicao 3.
3
Palpite:
Verde Preto Branco Rosa
Correcao:      0Pretas
Correcao:      0Brancas
Mastermind venceu!
(base) lu@lu-Aspire-A515-51:~/2019/Prog3/Anelise/Mastermind/Luana_Andre$
```

Fonte: Autores

UML de classes



Fonte: Autores