

PROGRAMAÇÃO ORIENTADA A OBJETOS

JOGO DA FORCA

PARTE III



EDUARDO HENRIQUE



PAULO MARCELO

CONCEITOS IMPLEMENTADOS

- S.O.L.I.D
- ARQUIVOS
- TEMPLATES

PRINCIPIOS DE S.O.L.I.D

* ANTES

```
class introduction{  
public:  
    int option;  
};  
  
class players : public introduction{  
private:  
    string nome;  
    string senha;  
public:  
    void setnome_senha(string nome,string senha);  
    void registration();  
    void login();  
    void difficulty();  
    void easy();  
    void normal();  
    void hard();  
};
```

> class introduction{ ...
> class players : public introduction{ ...
> class registro_player : public players{ ...
> class login_player : public players{ ...
> class choice_difficulty : public players{ ...
> class administrador{ ...
> class login_administrador : public administrador{ ...
> class lista_administrador : public login_administrador{ ...
> class pesquisar_administrador : public login_administrador{ ...

* DEPOIS

X

TEMPLATES

```
template <class GAME>
GAME estrutura_game(GAME palavra[10][25], GAME nome[10][10]);
template <class GAME>
GAME estrutura_hard_game(GAME lista_palavra[10][25]);
```

```
GAME estrutura_game(GAME lista_palavra[10][25], GAME nome[1][10])
{
    std::cout << "\n\n==== SEU TEMA E SOBRE " << nome[0] << "====\n\n";
    choice_difficulty escolha;

    // contadores
    int vida = 6, acertos = 0, erros, i, tamanho_palavra;
    char letra[1], lacuna[25] = "_";

    // escolher a palavra aleatoria
    srand((unsigned char)time(NULL));
    int palavra_aleatoria = rand() % 10;
    std::string palavra = lista_palavra[palavra_aleatoria];
    tamanho_palavra = strlen(lista_palavra[palavra_aleatoria]);

    // for para determinar o tamanho das lacunas
    for (i = 0; i < tamanho_palavra; i++)
    {
        lacuna[i] = '_';
    }

    while (vida > 0)
    {
        erros = true;
        std::cout << "\n"
            << lacuna << "\n";
        std::cout << "Digite uma letra: ";
        std::cin >> letra;
        std::cout << "\n";
        if (letra[0] == palavra[i])
        {
            lacuna[i] = letra[0];
            acertos++;
            std::cout << "Acerto! \n";
        }
        else
        {
            vida--;
            std::cout << "Erro! \n";
        }
    }
}
```

ARQUIVOS

• ARQUIVO USUARIOS.TXT

usuarios.txt X

```
1 paulo  
2 marcelo  
3
```

```
fp.open("usuarios.txt", ios::app);  
fpp.open("usuariosPontos.txt", ios::app);  
  
if (!fp.is_open())  
{  
    std::cout << "ocorreu um problema ao fazer o seu cadastro!!";  
    fp.close();  
}  
else  
{  
    fflush(stdin);  
    std::cout << "digite um nome: ";  
    cin >> nome;  
  
    fflush(stdin);  
    std::cout << "digite uma senha: ";  
    cin >> senha;  
    //-----  
    pontosInvocador = 0;  
  
    // grava as informações passadas pelo usuário  
    fp << nome << endl  
    << senha << endl;  
    fpp << nome << endl;
```

ARQUIVOS

• ARQUIVO USUARIOS_PONTOS.TXT

usuariosPontos.txt X

```
1 paulo
2 50
3
4
// estrutura para "salvar" o arquivo
typedef struct
{
    string name;
    int pontuacao_jogador = 0;
} jogador_estrutura;
jogador_estrutura jogador[50];
// abrindo conta do jogador para add os seus pontos ganhados pela vitoria.
ifstream fpp("usuariosPontos.txt");
if (!fpp.good())
{
    std::cout << ("ocorreu um problema ao adicionar os pontos para o jogador!");
}
else
{
    int i = 0;
    while (fpp)
    {
        getline(fpp, jogador[i].name);
        fpp >> jogador[i].pontuacao_jogador;
        i++;
    }
}
fpp.close();
string nome_jogador;
cout << "digite seu nome de login";
cin >> nome_jogador;
for (int i = 0; i < 50; i++)
{
    if (nome_jogador == jogador[i].name)
    {
        int pontuacao = 50;
        jogador[i].pontuacao_jogador += pontuacao;
    }
}
```

INCREMENTAÇÕES PARA ETAPA III

1. PONTUAÇÃO PARA OS JOGADORES.
2. CLASSE ADMINISTRADOR.
3. OBSERVA LISTA DE JOGADORES CADASTRADOS.
4. PROCURAR OS DADOS DE UM JOGADOR ESPECÍFICO.