Documentação Jogo Segundo Semestre TP

Fernanda Yasmim Antunes Vaz

Kayque Fernando Borges

Raphael de Campos Borges

Versão 2.0

# Sumário

**Namespaces**

**Lista de Namespaces**

Esta é a lista de todos os Namespaces com suas respectivas descrições:

**exceptions**

**ext**

# Índice Hierárquico

## Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Batalha

std::exception

exceptions::letra\_invalida

exceptions::numero\_invalido

Inimigo

ProfAnalNumerica

ProfCalculo2

ProfFundMec

ProfPDS2

ProfSD

Item

Personagem

# Índice dos Componentes

## Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

**Batalha (Classe que representa uma batalha entre um Personagem e um Inimigo )**

**Inimigo (Classe base para Inimigos )**

**Item (Classe para representar um Item )**

**exceptions::letra\_invalida (Exceção para letra inválida )**

**exceptions::numero\_invalido (Exceção para número inválido )**

**Personagem (Classe para representar um Personagem )**

**ProfAnalNumerica (Classe que representa um Inimigo do tipo "Professor de Análise Numérica". Herda da classe Inimigo )**

**ProfCalculo2 (Classe que representa um Inimigo do tipo "Professor de Cálculo 2". Herda da classe Inimigo )**

**ProfFundMec (Classe que representa um Inimigo do tipo "Professor de Fundamentos Mecânicos". Herda da classe Inimigo )**

**ProfPDS2 (Classe que representa um Inimigo do tipo "Professor de Projeto e Desenvolvimento de Sistemas 2". Herda da classe Inimigo )**

**ProfSD (Classe que representa um Inimigo do tipo "Professor de Sistemas Digitais". Herda da classe Inimigo )**

# Índice dos Arquivos

## Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

**código/Batalha.cpp (Implementação da classe Batalha )**

**código/Batalha.hpp**

**código/Exceptions.hpp**

**código/Extra.hpp**

**código/Inimigo.cpp (Implementação da classe Inimigo )**

**código/Inimigo.hpp**

**código/Item.cpp (Implementação da classe Item )**

**código/Item.hpp**

**código/main.cpp (Arquivo principal contendo a função main e funções relacionadas ao jogo )**

**código/Personagem.cpp (Implementação da classe Personagem )**

**código/Personagem.hpp**

**código/ProfAnalNumerica.cpp**

**código/ProfAnalNumerica.hpp**

**código/ProfCalculo2.cpp (Implementação da classe ProfCalculo2 )**

**código/ProfCalculo2.hpp**

**código/ProfFundMec.cpp (Implementação da classe ProfFundMec )**

**código/ProfFundMec.hpp**

**código/ProfPds2.cpp (Implementação da classe ProfPDS2 )**

**código/ProfPds2.hpp**

**código/ProfSistemasDigitais.cpp (Implementação da classe ProfSD )**

**código/ProfSistemasDigitais.hpp**

# Namespace

## Refência do Namespace exceptions

### Componentes

class **letra\_invalida***Exceção para letra inválida.*

class **numero\_invalido***Exceção para número inválido.*

## Refência do Namespace ext

### Funções

* void **iniciar** ()

*Função para exibir mensagem inicial.*

* void **creditos** ()

*Função para exibir créditos do grupo.*

* void **reprovacao** ()

*Função para exibir mensagem de reprovação.*

* void **print\_a** ()

*Função para imprimir nota A.*

* void **print\_b** ()

*Função para imprimir nota B.*

* void **print\_c** ()

*Função para imprimir nota C.*

* void **print\_d** ()

*Função para imprimir nota D.*

### Funções

#### void ext::creditos ()

Função para exibir créditos do grupo.

36 {

37 std::cout << "\nGRUPO:\nFernanda Vaz\nRaphael Borges\nKayque Borges\n" << endl;

38 std::cout << "PROFESSOR: Thiago Noronha\n" << endl;

39 }

43 void reprovacao() {

44 std::cout << "Que pena! parece que voce reprovou nessa materia...";

45 std::cout << "Infelizmente isso te desmotivou muito e voce acabou trancando o semestre, tente de novo no proximo!\n" << endl;

46 exit(0);

47 }

48

52 void print\_a() {

53 std::cout << R"(

54 .----------------.

55 | .--------------. |

56 | | \_\_ | |

57 | | / \ | |

58 | | / /\ \ | |

59 | | / \_\_\_\_ \ | |

60 | | \_/ / \ \\_ | |

61 | ||\_\_\_\_| |\_\_\_\_|| |

62 | | | |

63 | '--------------' |

64 '----------------' )" << std::endl;

65 }

69 void print\_b() {

70 std::cout << R"(

71 .----------------.

72 | .--------------. |

73 | | \_\_\_\_\_\_ | |

74 | | |\_ \_ \ | |

75 | | | |\_) | | |

76 | | | \_\_'. | |

77 | | \_| |\_\_) | | |

78 | | |\_\_\_\_\_\_\_/ | |

79 | | | |

80 | '--------------' |

81 '----------------'

82 )" << std::endl;

83

84 }

88 void print\_c() {

89 std::cout << R"(

90

91 .----------------.

92 | .--------------. |

93 | | \_\_\_\_\_\_ | |

94 | | .' \_\_\_ | | |

95 | | / .' \\_| | |

96 | | | | | |

97 | | \ `.\_\_\_.'\ | |

98 | | `.\_\_\_\_\_.' | |

99 | | | |

100 | '--------------' |

101 '----------------'

102

103 )" << std::endl;

104

105 }

109 void print\_d() {

110 std::cout << R"(

111

112

113 .----------------.

114 | .--------------. |

115 | | \_\_\_\_\_\_\_\_ | |

116 | | |\_ \_\_\_ `. | |

117 | | | | `. \ | |

118 | | | | | | | |

119 | | \_| |\_\_\_.' / | |

120 | | |\_\_\_\_\_\_\_\_.' | |

121 | | | |

122 | '--------------' |

123 '----------------'

124

125

126 )" << std::endl;

127

128 }

129

130 }

131

132 #endif;

#### void ext::iniciar ()

Função para exibir mensagem inicial.

11 {

12 std::cout << R"(

13

14 \_ \_ \_

15 | | (\_) | |

16 | |\_\_ \_\_\_ \_ \_\_ \_\_\_ \_\_ \_\_\_ \_ \_\_ \_\_| | \_\_\_ \_\_ \_ \_\_\_

#### void ext::print\_a ()

Função para imprimir nota A.

37 :\nFernanda Vaz\nRaphael Borges\nKayque Borges\n" << endl;

38 std::cout << "PROFESSOR: Thiago Noronha\n" << endl;

39 }

#### void ext::print\_b ()

Função para imprimir nota B.

43 {

44 std::cout << "Que pena! parece que voce reprovou nessa materia...";

45 std::cout << "Infelizmente isso te desmotivou muito e voce acabou trancando o semestre, tente de novo no proximo!\n" << endl;

46 exit(0);

#### void ext::print\_c ()

Função para imprimir nota C.

52 {

53 std::cout << R"(

#### void ext::print\_d ()

Função para imprimir nota D.

#### void ext::reprovacao ()

Função para exibir mensagem de reprovação.

# Classes

## Referência da Classe Batalha

Classe que representa uma batalha entre um **Personagem** e um **Inimigo**.

#include <Batalha.hpp>

### Membros Públicos

* **Batalha** (**Personagem** \*personagem, **Inimigo** \*inimigo)

*Construtor da classe* ***Batalha****.*

* void **executaTurno** (int escolha)

*Executa um turno da batalha com base na escolha do jogador.*

* bool **terminou** ()

*Verifica se a batalha terminou.*

* void **executaTurnoInimigo** ()

*Executa o turno do* ***Inimigo****.*

### Descrição detalhada

Classe que representa uma batalha entre um **Personagem** e um **Inimigo**.

### Construtores e Destrutores

#### Batalha::Batalha (Personagem \* *personagem*, Inimigo \* *inimigo*)

Construtor da classe **Batalha**.

##### Parâmetros

| *personagem* | Ponteiro para o **Personagem**. |
| --- | --- |
| *inimigo* | Ponteiro para o **Inimigo**. |
| *personagem* | Ponteiro para o objeto **Personagem**. |
| *inimigo* | Ponteiro para o objeto **Inimigo**. |

14 : personagem(personagem), inimigo(inimigo) {}

### Documentação das funções

#### void Batalha::executaTurno (int *escolha*)

Executa um turno da batalha com base na escolha do jogador.

##### Parâmetros

| *escolha* | Escolha do jogador (1 ou 2). |
| --- | --- |

20 {

21 int dano = personagem->ataca(escolha);

22 inimigo->recebeDano(dano);

23 std::cout << "Voce foi um bom aluno e a aula do professor ficou " << dano << " vezes mais facil !\n" << std::endl;

24

25 if (!inimigo->estaVivo()) {

26 personagem->printInfo();

27 inimigo->printInfo();

28 std::cout << "Voce passou nessa materia!\n" << std::endl;

29 return;

30 }

31

32 dano = inimigo->ataca();

33 inimigo->falar();

34 personagem->recebeDano(dano);

35 std::cout << "Mas o professor nao facilitou e voce perdeu " << dano << " de NSG!\n" << std::endl;

36

37 if (!personagem->estaVivo()) {

38 personagem->printInfo();

39 inimigo->printInfo();

40 }

41 }

#### void Batalha::executaTurnoInimigo ()

Executa o turno do **Inimigo**.

Executa o turno do inimigo.

54 {

55 if (!personagem->estaVivo() || !inimigo->estaVivo()) {

56 return;

57 }

58

59 int ataqueInimigo = inimigo->ataca();

60 personagem->recebeDano(ataqueInimigo);

61 inimigo->falar();

62 std::cout << "O professor nao foi legal e voce perdeu " << ataqueInimigo << " de NSG" << std::endl;

63 }

#### bool Batalha::terminou ()

Verifica se a batalha terminou.

##### Retorna

true se a batalha terminou, false caso contrário.

True se a batalha terminou, False caso contrário.

47 {

48 return !personagem->estaVivo() || !inimigo->estaVivo();

49 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**Batalha.hpp**
* código/**Batalha.cpp**

## Referência da Classe Inimigo

Classe base para Inimigos.

#include <Inimigo.hpp>

Diagrama de hierarquia da classe Inimigo:

### Membros Públicos

* **Inimigo** (std::string nome)

*Construtor da classe* ***Inimigo****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Inimigo****.*

* bool **estaVivo** ()

*Verifica se o* ***Inimigo*** *está vivo.*

* void **printInfo** ()

*Imprime informações do* ***Inimigo****.*

* int **getAtaque** ()

*Retorna o ataque do* ***Inimigo****.*

* std::string **getNome** ()

*Retorna o nome do* ***Inimigo****.*

* virtual int **ataca** ()=0

*Função puramente virtual para o ataque do* ***Inimigo****.*

* virtual void **falar** ()=0

*Função puramente virtual para a fala do* ***Inimigo****.*

* int **numeroAleatorio** ()

*Função para gerar número aleatório de 1 a 3.*

### Atributos Protegidos

* std::string **\_nome**
* int **\_vida**
* int **\_ataque**

### Descrição detalhada

Classe base para Inimigos.

### Construtores e Destrutores

#### Inimigo::Inimigo (std::string *nome*)

Construtor da classe **Inimigo**.

##### Parâmetros

| *nome* | Nome do **Inimigo**. |
| --- | --- |
| *nome* | Nome do inimigo. |

12 : \_nome(nome) {}

### Documentação das funções

#### virtual int Inimigo::ataca ()[pure virtual]

Função puramente virtual para o ataque do **Inimigo**.

##### Retorna

Valor do ataque.

Implementado por **ProfAnalNumerica** (*p.*), **ProfCalculo2** (*p.*), **ProfFundMec** (*p.*), **ProfPDS2** (*p.*) e **ProfSD** (*p.*).

#### bool Inimigo::estaVivo ()

Verifica se o **Inimigo** está vivo.

Verifica se o inimigo está vivo.

##### Retorna

true se estiver vivo, false caso contrário.

True se o inimigo está vivo, False caso contrário.

29 {

30 return \_vida > 0;

31 }

#### virtual void Inimigo::falar ()[pure virtual]

Função puramente virtual para a fala do **Inimigo**.

Implementado por **ProfAnalNumerica** (*p.*), **ProfCalculo2** (*p.*), **ProfFundMec** (*p.*), **ProfPDS2** (*p.*) e **ProfSD** (*p.*).

#### int Inimigo::getAtaque ()

Retorna o ataque do **Inimigo**.

Obtém o valor de ataque do inimigo.

##### Retorna

Valor do ataque.

Valor de ataque do inimigo.

37 {

38 return \_ataque;

39 }

#### string Inimigo::getNome ()

Retorna o nome do **Inimigo**.

Obtém o nome do inimigo.

##### Retorna

Nome do **Inimigo**.

Nome do inimigo.

45 {

46 return \_nome;

47 }

#### int Inimigo::numeroAleatorio ()

Função para gerar número aleatório de 1 a 3.

Gera um número aleatório entre 1 e 3.

##### Retorna

Número aleatório.

Número aleatório gerado.

62 {

63 return rand() % 3 + 1;

64 }

#### void Inimigo::printInfo ()

Imprime informações do **Inimigo**.

Imprime informações sobre o inimigo.

52 {

53 cout << "------------------------------------------------------------" << endl;

54 cout << \_nome << " | Dificuldade (vida): " << \_vida << endl;

55 cout << "------------------------------------------------------------\n" << endl;

56 }

#### void Inimigo::recebeDano (int *dano*)

Função para reduzir a vida do **Inimigo**.

Recebe dano no inimigo.

##### Parâmetros

| *dano* | Valor do dano a ser aplicado. |
| --- | --- |
| *dano* | Quantidade de dano a ser recebido. |

18 {

19 \_vida -= dano;

20 if (\_vida < 0) {

21 \_vida = 0;

22 }

23 }

### Atributos

#### int Inimigo::\_ataque[protected]

Ataque do **Inimigo**.

#### std::string Inimigo::\_nome[protected]

Nome do **Inimigo**.

#### int Inimigo::\_vida[protected]

Vida do **Inimigo**.

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**Inimigo.hpp**
* código/**Inimigo.cpp**

## Referência da Classe Item

Classe para representar um **Item**.

#include <Item.hpp>

### Membros Públicos

* **Item** (std::string nome, int valorDeRegeneracao)

*Construtor da classe* ***Item****.*

* std::string **getNome** ()

*Retorna o nome do* ***Item****.*

* int **getValorDeRegeneracao** ()

*Retorna o valor de regeneração do* ***Item****.*

### Descrição detalhada

Classe para representar um **Item**.

### Construtores e Destrutores

#### Item::Item (std::string *nome*, int *valorDeRegeneracao*)

Construtor da classe **Item**.

##### Parâmetros

| *nome* | Nome do **Item**. |
| --- | --- |
| *valorDeRegeneracao* | Valor de regeneração de vida/NSG. |
| *nome* | Nome do item. |
| *valorDeRegeneracao* | Valor de regeneração associado ao item. |

13 : nome(nome), valorDeRegeneracao(valorDeRegeneracao) {}

### Documentação das funções

#### std::string Item::getNome ()

Retorna o nome do **Item**.

Obtém o nome do item.

##### Retorna

Nome do **Item**.

Nome do item.

19 {

20 return nome;

21 }

#### int Item::getValorDeRegeneracao ()

Retorna o valor de regeneração do **Item**.

Obtém o valor de regeneração associado ao item.

##### Retorna

Valor de regeneração.

Valor de regeneração do item.

27 {

28 return valorDeRegeneracao;

29 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**Item.hpp**
* código/**Item.cpp**

## Referência da Classe exceptions::letra\_invalida

Exceção para letra inválida.

#include <Exceptions.hpp>

Diagrama de hierarquia da classe exceptions::letra\_invalida:

### Membros Públicos

* const char \* **what** () const throw ()

### Descrição detalhada

Exceção para letra inválida.

### Documentação das funções

#### const char \* exceptions::letra\_invalida::what () const throw ( ) [inline]

22 {

23 return "Ops, parece que você não sabe a diferença entre números e letras, tudo bem! Tente de novo (número)\n";

24 }

#### A documentação para essa classe foi gerada a partir do seguinte arquivo:

* código/**Exceptions.hpp**

## Referência da Classe exceptions::numero\_invalido

Exceção para número inválido.

#include <Exceptions.hpp>

Diagrama de hierarquia da classe exceptions::numero\_invalido:

### Membros Públicos

* const char \* **what** () const throw ()

### Descrição detalhada

Exceção para número inválido.

### Documentação das funções

#### const char \* exceptions::numero\_invalido::what () const throw ( ) [inline]

11 {

12 return "Parece que você ainda não aprendeu os números direitinho, escolha um número válido";

13 }

#### A documentação para essa classe foi gerada a partir do seguinte arquivo:

* código/**Exceptions.hpp**

## Referência da Classe Personagem

Classe para representar um **Personagem**.

#include <Personagem.hpp>

### Membros Públicos

* **Personagem** (std::string nome)

*Construtor da classe* ***Personagem****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Personagem****.*

* void **printInfo** ()

*Imprime informações do* ***Personagem****.*

* int **ataca** (int escolha)

*Função para realizar um ataque com base no número escolhido.*

* void **adicionarItem** (**Item** item)

*Função para adicionar um* ***Item*** *ao inventário do* ***Personagem****.*

* int **usarItem** (int index)

*Função para usar um* ***Item*** *do inventário e regenerar vida.*

* bool **estaVivo** ()

*Verifica se o* ***Personagem*** *está vivo.*

* std::string **getNome** ()

*Retorna o nome do* ***Personagem****.*

### Descrição detalhada

Classe para representar um **Personagem**.

### Construtores e Destrutores

#### Personagem::Personagem (std::string *nome*)

Construtor da classe **Personagem**.

##### Parâmetros

| *nome* | Nome do **Personagem**. |
| --- | --- |
| *nome* | O nome do personagem. |

12 : \_nome(nome) {}

### Documentação das funções

#### void Personagem::adicionarItem (Item *item*)

Função para adicionar um **Item** ao inventário do **Personagem**.

##### Parâmetros

| *item* | **Item** a ser adicionado. |
| --- | --- |

#### int Personagem::ataca (int *tipo*)

Função para realizar um ataque com base no número escolhido.

Função para realizar um ataque.

##### Parâmetros

| *escolha* | Escolha do jogador (1 ou 2). |
| --- | --- |

##### Retorna

Valor do dano causado.

##### Parâmetros

| *tipo* | O tipo de ataque a ser realizado (1 ou 2). |
| --- | --- |

##### Retorna

O valor do dano causado no ataque.

39 {

40 if (tipo == 1)

41 return \_ataque1;

42 else

43 return \_ataque2;

44 }

#### bool Personagem::estaVivo ()

Verifica se o **Personagem** está vivo.

Função para verificar se o personagem está vivo.

##### Retorna

true se estiver vivo, false caso contrário.

true se o personagem estiver vivo, false caso contrário.

50 {

51 return \_vida > 0;

52 }

#### string Personagem::getNome ()

Retorna o nome do **Personagem**.

Função para obter o nome do personagem.

##### Retorna

Nome do **Personagem**.

O nome do personagem.

66 {

67 return \_nome;

68 }

#### void Personagem::printInfo ()

Imprime informações do **Personagem**.

Função para imprimir informações do personagem.

28 {

29 cout << "------------------------------------------------------------" << endl;

30 cout << \_nome << " | seu NSG: " << \_vida << " / " << \_vida\_maxima << endl;

31 cout << "------------------------------------------------------------" << endl;

32 }

#### void Personagem::recebeDano (int *dano*)

Função para reduzir a vida do **Personagem**.

Função para receber dano no personagem.

##### Parâmetros

| *dano* | Valor do dano a ser aplicado. |
| --- | --- |
| *dano* | O valor do dano a ser recebido. |

18 {

19 \_vida -= dano;

20 if (\_vida < 0) {

21 \_vida = 0;

22 }

23 }

#### void Personagem::usarItem (int *indiceItem*)

Função para usar um **Item** do inventário e regenerar vida.

Função para usar um item do inventário.

##### Parâmetros

| *index* | Índice do **Item** a ser usado. |
| --- | --- |

##### Retorna

Valor de regeneração de vida.

##### Parâmetros

| *indiceItem* | O índice do item a ser utilizado. |
| --- | --- |

74 {

75 if (indiceItem < 0 || indiceItem >= inventario.size()) {

76 return;

77 }

78

79 \_vida += inventario[indiceItem].getValorDeRegeneracao();

80 if (\_vida > 100) {

81 \_vida = 100;

82 }

83 std::cout << "O item regenerou " << inventario[indiceItem].getValorDeRegeneracao() << " do seu NSG\n" << std::endl;

84 printInfo();

85 inventario.erase(inventario.begin() + indiceItem);

86 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**Personagem.hpp**
* código/**Personagem.cpp**

## Referência da Classe ProfAnalNumerica

Classe que representa um **Inimigo** do tipo "Professor de Análise Numérica". Herda da classe **Inimigo**.

#include <ProfAnalNumerica.hpp>

Diagrama de hierarquia da classe ProfAnalNumerica:

### Membros Públicos

* **ProfAnalNumerica** (string nome)

*Construtor da classe* ***ProfAnalNumerica****.*

* int **ataca** () override

*Função que representa o ataque do Professor de Análise Numérica.*

* void **falar** () override

*Função que representa a fala do Professor de Análise Numérica.*

#### Membros Públicos herdados de Inimigo

* **Inimigo** (std::string nome)

*Construtor da classe* ***Inimigo****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Inimigo****.*

* bool **estaVivo** ()

*Verifica se o* ***Inimigo*** *está vivo.*

* void **printInfo** ()

*Imprime informações do* ***Inimigo****.*

* int **getAtaque** ()

*Retorna o ataque do* ***Inimigo****.*

* std::string **getNome** ()

*Retorna o nome do* ***Inimigo****.*

* int **numeroAleatorio** ()

*Função para gerar número aleatório de 1 a 3.*

### Outros membros herdados

#### Atributos Protegidos herdados de Inimigo

* std::string **\_nome**
* int **\_vida**
* int **\_ataque**

### Descrição detalhada

Classe que representa um **Inimigo** do tipo "Professor de Análise Numérica". Herda da classe **Inimigo**.

### Construtores e Destrutores

#### ProfAnalNumerica::ProfAnalNumerica (string *nome*)

Construtor da classe **ProfAnalNumerica**.

##### Parâmetros

| *nome* | Nome do Professor de Análise Numérica. |
| --- | --- |
| *nome* | O nome do professor de Análise Numérica. |

12 : Inimigo(nome) {

13 this->\_vida = 50;

14 this->\_ataque = 15;

15 }

### Documentação das funções

#### int ProfAnalNumerica::ataca ()[override], [virtual]

Função que representa o ataque do Professor de Análise Numérica.

Função para realizar o ataque do professor de Análise Numérica.

##### Retorna

Valor do ataque.

O valor do dano causado no ataque.

Implementa **Inimigo** (*p.*).

21 {

22 return \_ataque;

23 }

#### void ProfAnalNumerica::falar ()[override], [virtual]

Função que representa a fala do Professor de Análise Numérica.

Função para o professor de Análise Numérica falar durante a batalha.

O professor pode mencionar tópicos como "Matrizes", "Interpolação" ou "Pégaso".

Implementa **Inimigo** (*p.*).

29 {

30 int fala = numeroAleatorio();

31 if (fala == 1) {

32 cout << "Matrizes!\n" << endl;

33 }

34 else if (fala == 2) {

35 cout << "Interpolação\n" << endl;

36 }

37 else {

38 cout << "Pégaso!\n" << endl;

39 }

40 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**ProfAnalNumerica.hpp**
* código/**ProfAnalNumerica.cpp**

## Referência da Classe ProfCalculo2

Classe que representa um **Inimigo** do tipo "Professor de Cálculo 2". Herda da classe **Inimigo**.

#include <ProfCalculo2.hpp>

Diagrama de hierarquia da classe ProfCalculo2:

### Membros Públicos

* **ProfCalculo2** (string nome)

*Construtor da classe* ***ProfCalculo2****.*

* int **ataca** () override

*Função que representa o ataque do Professor de Cálculo 2.*

* void **falar** () override

*Função que representa a fala do Professor de Cálculo 2.*

#### Membros Públicos herdados de Inimigo

* **Inimigo** (std::string nome)

*Construtor da classe* ***Inimigo****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Inimigo****.*

* bool **estaVivo** ()

*Verifica se o* ***Inimigo*** *está vivo.*

* void **printInfo** ()

*Imprime informações do* ***Inimigo****.*

* int **getAtaque** ()

*Retorna o ataque do* ***Inimigo****.*

* std::string **getNome** ()

*Retorna o nome do* ***Inimigo****.*

* int **numeroAleatorio** ()

*Função para gerar número aleatório de 1 a 3.*

### Outros membros herdados

#### Atributos Protegidos herdados de Inimigo

* std::string **\_nome**
* int **\_vida**
* int **\_ataque**

### Descrição detalhada

Classe que representa um **Inimigo** do tipo "Professor de Cálculo 2". Herda da classe **Inimigo**.

### Construtores e Destrutores

#### ProfCalculo2::ProfCalculo2 (string *nome*)

Construtor da classe **ProfCalculo2**.

##### Parâmetros

| *nome* | Nome do Professor de Cálculo 2. |
| --- | --- |
| *nome* | O nome do professor de Cálculo 2. |

12 : Inimigo(nome) {

13 this-> \_vida = 40;

14 this->\_ataque = 10;

15 }

### Documentação das funções

#### int ProfCalculo2::ataca ()[override], [virtual]

Função que representa o ataque do Professor de Cálculo 2.

Função para realizar o ataque do professor de Cálculo 2.

##### Retorna

Valor do ataque.

O valor do dano causado no ataque.

Implementa **Inimigo** (*p.*).

21 {

22 return \_ataque;

23 }

#### void ProfCalculo2::falar ()[override], [virtual]

Função que representa a fala do Professor de Cálculo 2.

Função para o professor de Cálculo 2 falar durante a batalha.

O professor pode mencionar tópicos como "Convergência de Séries", "Coordenadas Polares" ou "Gradiente de Vetor".

Implementa **Inimigo** (*p.*).

29 {

30 int fala = numeroAleatorio();

31 if (fala == 1) {

32 cout << "Convergencia de Séries!\n" << endl;

33 }

34 else if (fala == 2) {

35 cout << "Coordenadas Polares!\n" << endl;

36 }

37 else {

38 cout << "Gradiente de Vetor!\n" << endl;

39 }

40 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**ProfCalculo2.hpp**
* código/**ProfCalculo2.cpp**

## Referência da Classe ProfFundMec

Classe que representa um **Inimigo** do tipo "Professor de Fundamentos Mecânicos". Herda da classe **Inimigo**.

#include <ProfFundMec.hpp>

Diagrama de hierarquia da classe ProfFundMec:

### Membros Públicos

* **ProfFundMec** (string nome)

*Construtor da classe* ***ProfFundMec****.*

* int **ataca** () override

*Função que representa o ataque do Professor de Fundamentos Mecânicos.*

* void **falar** () override

*Função que representa a fala do Professor de Fundamentos Mecânicos.*

#### Membros Públicos herdados de Inimigo

* **Inimigo** (std::string nome)

*Construtor da classe* ***Inimigo****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Inimigo****.*

* bool **estaVivo** ()

*Verifica se o* ***Inimigo*** *está vivo.*

* void **printInfo** ()

*Imprime informações do* ***Inimigo****.*

* int **getAtaque** ()

*Retorna o ataque do* ***Inimigo****.*

* std::string **getNome** ()

*Retorna o nome do* ***Inimigo****.*

* int **numeroAleatorio** ()

*Função para gerar número aleatório de 1 a 3.*

### Outros membros herdados

#### Atributos Protegidos herdados de Inimigo

* std::string **\_nome**
* int **\_vida**
* int **\_ataque**

### Descrição detalhada

Classe que representa um **Inimigo** do tipo "Professor de Fundamentos Mecânicos". Herda da classe **Inimigo**.

### Construtores e Destrutores

#### ProfFundMec::ProfFundMec (string *nome*)

Construtor da classe **ProfFundMec**.

##### Parâmetros

| *nome* | Nome do Professor de Fundamentos Mecânicos. |
| --- | --- |
| *nome* | O nome do professor de Fundamentos de Mecânica. |

12 : Inimigo(nome) {

13 this->\_vida = 40;

14 this->\_ataque = 10;

15 }

### Documentação das funções

#### int ProfFundMec::ataca ()[override], [virtual]

Função que representa o ataque do Professor de Fundamentos Mecânicos.

Função para realizar o ataque do professor de Fundamentos de Mecânica.

##### Retorna

Valor do ataque.

O valor do dano causado no ataque.

Implementa **Inimigo** (*p.*).

21 {

22 return \_ataque;

23 }

#### void ProfFundMec::falar ()[override], [virtual]

Função que representa a fala do Professor de Fundamentos Mecânicos.

Função para o professor de Fundamentos de Mecânica falar durante a batalha.

O professor pode mencionar tópicos como "Maçã de Newton", "Conservação de Energia" ou "Rotação".

Implementa **Inimigo** (*p.*).

29 {

30 int fala = numeroAleatorio();

31 if (fala == 1) {

32 cout << "Maçã de Newton!\n" << endl;

33 }

34 else if (fala == 2) {

35 cout << "Conservacao de Energia!\n" << endl;

36 }

37 else {

38 cout << "Rotacao!\n" << endl;

39 }

40 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**ProfFundMec.hpp**
* código/**ProfFundMec.cpp**

## Referência da Classe ProfPDS2

Classe que representa um **Inimigo** do tipo "Professor de Projeto e Desenvolvimento de Sistemas 2". Herda da classe **Inimigo**.

#include <ProfPds2.hpp>

Diagrama de hierarquia da classe ProfPDS2:

### Membros Públicos

* **ProfPDS2** (string nome)

*Construtor da classe* ***ProfPDS2****.*

* int **ataca** () override

*Função que representa o ataque do Professor de Projeto e Desenvolvimento de Sistemas 2.*

* void **falar** () override

*Função que representa a fala do Professor de Projeto e Desenvolvimento de Sistemas 2.*

#### Membros Públicos herdados de Inimigo

* **Inimigo** (std::string nome)

*Construtor da classe* ***Inimigo****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Inimigo****.*

* bool **estaVivo** ()

*Verifica se o* ***Inimigo*** *está vivo.*

* void **printInfo** ()

*Imprime informações do* ***Inimigo****.*

* int **getAtaque** ()

*Retorna o ataque do* ***Inimigo****.*

* std::string **getNome** ()

*Retorna o nome do* ***Inimigo****.*

* int **numeroAleatorio** ()

*Função para gerar número aleatório de 1 a 3.*

### Outros membros herdados

#### Atributos Protegidos herdados de Inimigo

* std::string **\_nome**
* int **\_vida**
* int **\_ataque**

### Descrição detalhada

Classe que representa um **Inimigo** do tipo "Professor de Projeto e Desenvolvimento de Sistemas 2". Herda da classe **Inimigo**.

### Construtores e Destrutores

#### ProfPDS2::ProfPDS2 (string *nome*)

Construtor da classe **ProfPDS2**.

##### Parâmetros

| *nome* | Nome do Professor de Projeto e Desenvolvimento de Sistemas 2. |
| --- | --- |
| *nome* | O nome do professor de Programação de Sistemas Digitais 2. |

12 : Inimigo(nome) {

13 this->\_vida = 60;

14 this->\_ataque = 8;

15 }

### Documentação das funções

#### int ProfPDS2::ataca ()[override], [virtual]

Função que representa o ataque do Professor de Projeto e Desenvolvimento de Sistemas 2.

Função para realizar o ataque do professor de Programação de Sistemas Digitais 2.

##### Retorna

Valor do ataque.

O valor do dano causado no ataque.

Implementa **Inimigo** (*p.*).

21 {

22 return \_ataque;

23 }

#### void ProfPDS2::falar ()[override], [virtual]

Função que representa a fala do Professor de Projeto e Desenvolvimento de Sistemas 2.

Função para o professor de Programação de Sistemas Digitais 2 falar durante a batalha.

O professor pode mencionar tópicos como "C++", "Polimorfismo" ou "Programacao orientada a objetos".

Implementa **Inimigo** (*p.*).

29 {

30 int fala = numeroAleatorio();

31 if (fala == 1) {

32 cout << "C++!\n" << endl;

33 }

34 else if (fala == 2) {

35 cout << "Polimorfismo!\n" << endl;

36 }

37 else {

38 cout << "Programação orientada a objetos!\n" << endl;

39 }

40 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**ProfPds2.hpp**
* código/**ProfPds2.cpp**

## Referência da Classe ProfSD

Classe que representa um **Inimigo** do tipo "Professor de Sistemas Digitais". Herda da classe **Inimigo**.

#include <ProfSistemasDigitais.hpp>

Diagrama de hierarquia da classe ProfSD:

### Membros Públicos

* **ProfSD** (string nome)

*Construtor da classe* ***ProfSD****.*

* int **ataca** () override

*Função que representa o ataque do Professor de Sistemas Digitais.*

* void **falar** () override

*Função que representa a fala do Professor de Sistemas Digitais.*

#### Membros Públicos herdados de Inimigo

* **Inimigo** (std::string nome)

*Construtor da classe* ***Inimigo****.*

* void **recebeDano** (int dano)

*Função para reduzir a vida do* ***Inimigo****.*

* bool **estaVivo** ()

*Verifica se o* ***Inimigo*** *está vivo.*

* void **printInfo** ()

*Imprime informações do* ***Inimigo****.*

* int **getAtaque** ()

*Retorna o ataque do* ***Inimigo****.*

* std::string **getNome** ()

*Retorna o nome do* ***Inimigo****.*

* int **numeroAleatorio** ()

*Função para gerar número aleatório de 1 a 3.*

### Outros membros herdados

#### Atributos Protegidos herdados de Inimigo

* std::string **\_nome**
* int **\_vida**
* int **\_ataque**

### Descrição detalhada

Classe que representa um **Inimigo** do tipo "Professor de Sistemas Digitais". Herda da classe **Inimigo**.

### Construtores e Destrutores

#### ProfSD::ProfSD (string *nome*)

Construtor da classe **ProfSD**.

##### Parâmetros

| *nome* | Nome do Professor de Sistemas Digitais. |
| --- | --- |
| *nome* | O nome do professor de Sistemas Digitais. |

12 : Inimigo(nome) {

13 this->\_vida = 30;

14 this->\_ataque = 6;

15 }

### Documentação das funções

#### int ProfSD::ataca ()[override], [virtual]

Função que representa o ataque do Professor de Sistemas Digitais.

Função para realizar o ataque do professor de Sistemas Digitais.

##### Retorna

Valor do ataque.

O valor do dano causado no ataque.

Implementa **Inimigo** (*p.*).

21 {

22 return \_ataque;

23 }

#### void ProfSD::falar ()[override], [virtual]

Função que representa a fala do Professor de Sistemas Digitais.

Função para o professor de Sistemas Digitais falar durante a batalha.

O professor pode mencionar tópicos como "Projeto Lógico Combinacional", "Máquina de Estados Finitos" ou "Unidade Lógica Aritmética".

Implementa **Inimigo** (*p.*).

30 {

31 int fala = numeroAleatorio();

32 if (fala == 1) {

33 cout << "Projeto Lógico Combinacional!\n" << endl;

34 }

35 else if (fala == 2) {

36 cout << "Máquina de Estados Finitos!\n" << endl;

37 }

38 else {

39 cout << "Unidade Lógica Aritmética!\n" << endl;

40 }

41 }

#### A documentação para essa classe foi gerada a partir dos seguintes arquivos:

* código/**ProfSistemasDigitais.hpp**
* código/**ProfSistemasDigitais.cpp**

# Arquivos

## Referência do Arquivo código/Batalha.cpp

Implementação da classe **Batalha**.

#include <iostream>

#include "Batalha.hpp"

### Descrição detalhada

Implementação da classe **Batalha**.

## Referência do Arquivo código/Batalha.hpp

#include "Personagem.hpp"

#include "Inimigo.hpp"

### Componentes

class **Batalha***Classe que representa uma batalha entre um* ***Personagem*** *e um* ***Inimigo****.*

## Batalha.hpp

Ir para a documentação desse arquivo.

1 #ifndef BATALHA\_HPP

2 #define BATALHA\_HPP

3

4 #include "Personagem.hpp"

5 #include "Inimigo.hpp"

6

10 class Batalha {

11 Personagem\* personagem;

12 Inimigo\* inimigo;

14 public:

20 Batalha(Personagem\* personagem, Inimigo\* inimigo);

21

26 void executaTurno(int escolha);

27

32 bool terminou();

33

37 void executaTurnoInimigo();

38 };

39

40 #endif

## Referência do Arquivo código/Exceptions.hpp

#include <exception>

### Componentes

class **exceptions::numero\_invalido***Exceção para número inválido.*

class **exceptions::letra\_invalida***Exceção para letra inválida.*

### Namespaces

* namespace **exceptions**

## Exceptions.hpp

Ir para a documentação desse arquivo.

1 #include <exception>

2

3 namespace exceptions {

4

8 class numero\_invalido : public std::exception {

9 public:

10

11 const char\* what() const throw() {

12 return "Parece que você ainda não aprendeu os números direitinho, escolha um número válido";

13 }

14 };

15

19 class letra\_invalida : public std::exception {

20 public:

21

22 const char\* what() const throw() {

23 return "Ops, parece que você não sabe a diferença entre números e letras, tudo bem! Tente de novo (número)\n";

24 }

25 };

26 }

## Referência do Arquivo código/Extra.hpp

#include <iostream>

### Namespaces

* namespace **ext**

### Funções

* void **ext::iniciar** ()

*Função para exibir mensagem inicial.*

* void **ext::creditos** ()

*Função para exibir créditos do grupo.*

* void **ext::reprovacao** ()

*Função para exibir mensagem de reprovação.*

* void **ext::print\_a** ()

*Função para imprimir nota A.*

* void **ext::print\_b** ()

*Função para imprimir nota B.*

* void **ext::print\_c** ()

*Função para imprimir nota C.*

* void **ext::print\_d** ()

*Função para imprimir nota D.*

## Extra.hpp

Ir para a documentação desse arquivo.

1

2 #ifndef EXTRA\_HPP

3 #define EXTRA\_HPP

4

5 #include <iostream>

6

7 namespace ext {

11 void iniciar() {

12 std::cout << R"(

13

14 \_ \_ \_

15 | | (\_) | |

16 | |\_\_ \_\_\_ \_ \_\_ \_\_\_ \_\_ \_\_\_ \_ \_\_ \_\_| | \_\_\_ \_\_ \_ \_\_\_

17 | '\_ \ / \_ \ '\_ ` \_ \ \ \ / / | '\_ \ / \_` |/ \_ \ / \_` |/ \_ \

18 | |\_) | \_\_/ | | | | | \ V /| | | | | (\_| | (\_) | | (\_| | (\_) |

19 |\_.\_\_/ \\_\_\_|\_| |\_| |\_| \\_/ |\_|\_| |\_|\\_\_,\_|\\_\_\_/ \\_\_,\_|\\_\_\_/

20

21

22 \_\_\_ \_\_\_ \_\_\_ \_ \_ \_ \_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_ \_\_ \_\_\_ \_\_\_ \_\_\_\_\_ \_\_\_ \_\_\_

23 / \_\_| | \_\_| / \_\_| | | | | | \| | | \ / \_ \ / \_\_| | \_\_| | \/ | | \_\_| / \_\_| |\_ \_| | \_ \ | \_\_|

24 \\_\_ \ | \_| | (\_ | | |\_| | | .` | | |) | | (\_) | \\_\_ \ | \_| | |\/| | | \_| \\_\_ \ | | | / | \_|

25 |\_\_\_/ |\_\_\_| \\_\_\_| \\_\_\_/ |\_|\\_| |\_\_\_/ \\_\_\_/ |\_\_\_/ |\_\_\_| |\_| |\_| |\_\_\_| |\_\_\_/ |\_| |\_|\_\ |\_\_\_|

26

27 )" << '\n';

28

29

30 std::cout << "--------------SEGUNDO SEMESTRE --------------\n" << endl;

31 }

36 void creditos() {

37 std::cout << "\nGRUPO:\nFernanda Vaz\nRaphael Borges\nKayque Borges\n" << endl;

38 std::cout << "PROFESSOR: Thiago Noronha\n" << endl;

39 }

43 void reprovacao() {

44 std::cout << "Que pena! parece que voce reprovou nessa materia...";

45 std::cout << "Infelizmente isso te desmotivou muito e voce acabou trancando o semestre, tente de novo no proximo!\n" << endl;

46 exit(0);

47 }

48

52 void print\_a() {

53 std::cout << R"(

54 .----------------.

55 | .--------------. |

56 | | \_\_ | |

57 | | / \ | |

58 | | / /\ \ | |

59 | | / \_\_\_\_ \ | |

60 | | \_/ / \ \\_ | |

61 | ||\_\_\_\_| |\_\_\_\_|| |

62 | | | |

63 | '--------------' |

64 '----------------' )" << std::endl;

65 }

69 void print\_b() {

70 std::cout << R"(

71 .----------------.

72 | .--------------. |

73 | | \_\_\_\_\_\_ | |

74 | | |\_ \_ \ | |

75 | | | |\_) | | |

76 | | | \_\_'. | |

77 | | \_| |\_\_) | | |

78 | | |\_\_\_\_\_\_\_/ | |

79 | | | |

80 | '--------------' |

81 '----------------'

82 )" << std::endl;

83

84 }

88 void print\_c() {

89 std::cout << R"(

90

91 .----------------.

92 | .--------------. |

93 | | \_\_\_\_\_\_ | |

94 | | .' \_\_\_ | | |

95 | | / .' \\_| | |

96 | | | | | |

97 | | \ `.\_\_\_.'\ | |

98 | | `.\_\_\_\_\_.' | |

99 | | | |

100 | '--------------' |

101 '----------------'

102

103 )" << std::endl;

104

105 }

109 void print\_d() {

110 std::cout << R"(

111

112

113 .----------------.

114 | .--------------. |

115 | | \_\_\_\_\_\_\_\_ | |

116 | | |\_ \_\_\_ `. | |

117 | | | | `. \ | |

118 | | | | | | | |

119 | | \_| |\_\_\_.' / | |

120 | | |\_\_\_\_\_\_\_\_.' | |

121 | | | |

122 | '--------------' |

123 '----------------'

124

125

126 )" << std::endl;

127

128 }

129

130 }

131

132 #endif;

## Referência do Arquivo código/Inimigo.cpp

Implementação da classe **Inimigo**.

#include "Inimigo.hpp"

### Descrição detalhada

Implementação da classe **Inimigo**.

## Referência do Arquivo código/Inimigo.hpp

#include <iostream>

### Componentes

class **Inimigo***Classe base para Inimigos.*

## Inimigo.hpp

Ir para a documentação desse arquivo.

1 #ifndef INIMIGO\_HPP

2 #define INIMIGO\_HPP

3

4 #include <iostream>

5

9 class Inimigo {

10 protected:

11 std::string \_nome;

12 int \_vida;

13 int \_ataque;

15 public:

20 Inimigo(std::string nome);

21

26 void recebeDano(int dano);

27

32 bool estaVivo();

33

37 void printInfo();

38

43 int getAtaque();

44

49 std::string getNome();

50

55 virtual int ataca() = 0;

56

60 virtual void falar() = 0;

61

66 int numeroAleatorio();

67 };

68

69 #endif

## Referência do Arquivo código/Item.cpp

Implementação da classe **Item**.

#include "Item.hpp"

### Descrição detalhada

Implementação da classe **Item**.

## Referência do Arquivo código/Item.hpp

#include <string>

### Componentes

class **Item***Classe para representar um* ***Item****.*

## Item.hpp

Ir para a documentação desse arquivo.

1 #ifndef ITEM\_HPP

2 #define ITEM\_HPP

3

4 #include <string>

5

9 class Item {

10 public:

16 Item(std::string nome, int valorDeRegeneracao);

17

22 std::string getNome();

23

28 int getValorDeRegeneracao();

29

30 private:

31 std::string nome;

32 int valorDeRegeneracao;

33 };

34

35 #endif

## Referência do Arquivo código/main.cpp

Arquivo principal contendo a função main e funções relacionadas ao jogo.

#include <string>

#include <iostream>

#include <sstream>

#include "Personagem.hpp"

#include "ProfCalculo2.hpp"

#include "ProfAnalNumerica.hpp"

#include "ProfFundMec.hpp"

#include "ProfSistemasDigitais.hpp"

#include "ProfPds2.hpp"

#include "Batalha.hpp"

#include "Item.hpp"

#include <cstdlib>

#include <ctime>

#include "Extra.hpp"

#include "Exceptions.hpp"

### Funções

* int **lerInt** ()

*Função para ler um número inteiro da entrada padrão.*

* int **escolhaMenu** ()

*Função para obter a escolha do menu principal.*

* void **vitoria** (int vida)

*Função para imprimir uma mensagem de vitória.*

* void **iniciarJogo** ()

*Função para iniciar o jogo.*

* int **main** ()

*Função principal (main) do programa.*

### Variáveis

* **Item** **itens** [10]

### Descrição detalhada

Arquivo principal contendo a função main e funções relacionadas ao jogo.

### Funções

#### int escolhaMenu ()

Função para obter a escolha do menu principal.

##### Retorna

A escolha do menu.

65 {

66

67 int escolhaMenu = 0;

68

69 while (escolhaMenu > 3 || escolhaMenu < 1) {

70 try {

71 std::cout << "Escolha uma opcao:\n";

72 std::cout << "1. Primeiro dia de aula (inicia o jogo)" << std::endl;

73 std::cout << "2. Creditos" << std::endl;

74 std::cout << "3. Trancar o curso (sair)\n" << std::endl;

75 escolhaMenu = lerInt();

76 if (escolhaMenu > 3 || escolhaMenu < 1) {

77 throw exceptions::numero\_invalido();

78 }

79 } catch (const exceptions::numero\_invalido& e) {

80 std::cout << e.what() << "\n" << std::endl;

81 escolhaMenu = 0; // Reseta o valor de 'escolhaMenu' para garantir que o loop continue

82 }

83 }

84

85 return escolhaMenu;

86 }

#### void iniciarJogo ()

Função para iniciar o jogo.

112 {

113 std::string nome;

114

115 std::cout << "Antes de comecar, me diga seu nome: " << std::endl;

116 while (true) {

117 std::getline(std::cin, nome);

118

119 // Verifica se o nome contém apenas letras

120 bool apenasLetras = true;

121 for (char c : nome) {

122 if (!isalpha(c) && !isspace(c)) {

123 apenasLetras = false;

124 break;

125 }

126 }

127

128 if (apenasLetras) {

129 break;

130 } else {

131 try {

132 throw exceptions::letra\_invalida();

133 } catch (const exceptions::letra\_invalida& e) {

134 std::cout << e.what() << std::endl;

135 }

136 }

137 }

138

139 Personagem personagem(nome);

140

141 // Crie os inimigos em um vetor

142 Inimigo\* inimigos[5] = { new ProfCalculo2("Professor de Calculo2"),

143 new ProfAnalNumerica("Professor de Analise Numerica"),

144 new ProfFundMec("Professor de Fundamentos de Mecanica"),

145 new ProfSD("Professor de Sistemas Digitais"),

146 new ProfPDS2("Professor de PDS2") };

147

148 int escolha;

149 for (int i = 0; i < 5; ++i) {

150 std::cout << "Voce entrou na sala do " << inimigos[i]->getNome() << "\n" << std::endl;

151 Batalha batalha(&personagem, inimigos[i]);

152

153 while (!batalha.terminou()) {

154 personagem.printInfo();

155 inimigos[i]->printInfo();

156 std::cout << "O que voce quer fazer?\n1.Fazer pergunta\n2.Sentar na primeira cadeira\n3.Usar um item especial\n4.Trancar o curso" << std::endl;

157 escolha = lerInt();

158

159 if (escolha != 1 && escolha != 2 && escolha != 3 && escolha != 4) {

160 std::cout << "Escolha um numero dentro das opcoes, nem vem..." << std::endl;

161 }

162

163 if (escolha == 3) {

164 personagem.exibirInventario();

165 if (!personagem.getInventario().empty()) {

166 std::cout << "Escolha um item para usar: " << std::endl;

167 escolha = lerInt();

168 personagem.usarItem(escolha); // tratamento de excessao

169 }

170 batalha.executaTurnoInimigo();

171 } else if (escolha == 1 || escolha == 2) {

172 batalha.executaTurno(escolha);

173 } else if (escolha == 4) {

174 std::cout << "Ja no segundo semestre? ok, nao vou te impedir, adeus!" << std::endl;

175 exit(0);

176 }

177 }

178

179 if (!personagem.estaVivo()) {

180 std::cout << "Voce reprovou na materia...\n" << std::endl;

181 ext::reprovacao();

182 break;

183 } else {

184 if (i != 4) {

185 std::cout << "Parabens!\n" << std::endl;

186 }

187 if (i < 4) { // se não for a última batalha

188 std::cout << "O que voce quer fazer agora?\n1. Ir para a proxima aula\n2. Dar uma passadinha no DA\n3. Trancar o curso\n";

189 escolha = lerInt();

190 switch (escolha) {

191 case 1: // Continuar batalhando

192 break;

193 case 2: { // Pegar item

194 int itemAleatorio = rand() % 10; // Gerando um índice aleatório

195 personagem.addItem(itens[itemAleatorio]);

196 std::cout << "Na sua passadinha pelo hall da engenharia voce adquiriu " << itens[itemAleatorio].getNome() << "!\n" << std::endl;

197 std::cout << "Mas agora voce esta super atrasado pra sua aula entao voce correu e...\n" << std::endl;

198 break;

199 }

200 case 3: // Sair

201 std::cout << "Ja no primeiro semestre? ok, nao vou te impedir, adeus!" << std::endl;

202 exit(0);

203 break;

204 default: {

205 std::cout << "Escolha um numero dentro das opcoes, nem vem..." << std::endl;

206 }

207 }

208 } else if (i == 4) {

209 vitoria(personagem.getVida());

210 exit(0);

211 }

212 }

213 }

214

215 if (personagem.estaVivo()) {

216 std::cout << "Vitoria! Voce derrotou todos os inimigos!\n";

217 }

218

219 // deletando os inimigos

220 for (int i = 0; i < 5; ++i) {

221 delete inimigos[i];

222 }

223 }

#### int lerInt ()

Função para ler um número inteiro da entrada padrão.

##### Retorna

O número inteiro lido.

40 {

41 std::string input = "";

42 int valor = 0;

43

44 while (true) {

45 std::getline(std::cin, input);

46

47 std::stringstream myStream(input);

48 if (myStream >> valor)

49 break;

50

51 // Lança a exceção 'letra\_invalida'

52 try {

53 throw exceptions::letra\_invalida();

54 } catch (const exceptions::letra\_invalida& e) {

55 std::cout << e.what() << "\n" << std::endl;

56 }

57 }

58 return valor;

59 }

#### int main ()

Função principal (main) do programa.

##### Retorna

0 se a execução ocorreu corretamente.

229 {

230 srand(time(0));

231

232 ext::iniciar();

233

234 int escolhaMenuInicial = escolhaMenu();

235 bool sairDoJogo = false;

236

237 while (!sairDoJogo) {

238 switch (escolhaMenuInicial) {

239 case 1: {

240 iniciarJogo();

241 break;

242 }

243 case 2: {

244 bool voltarMenu = false;

245 while (!voltarMenu) {

246 ext::creditos();

247 std::cout << "Agora que voce viu nossos nomes, decida:\n1. Primeiro dia de aula (inicia o jogo)\n2. Trancar o curso (sair)\n" << std::endl;

248 int escolhaAlunos = lerInt();

249 switch (escolhaAlunos) {

250 case 1: {

251 iniciarJogo();

252 break;

253 }

254 case 2: {

255 sairDoJogo = true;

256 std::cout << "Ja no primeiro semestre? ok, nao vou te impedir, adeus!" << std::endl;

257 exit(0);

258 break;

259 }

260 default: {

261 std::cout << "vou te mostrar nossos nomes de novo, mas escolha 1 ou 2 e pare de inventar hein..\n" << std::endl;

262 }

263 }

264 }

265 break;

266 }

267 case 3: {

268 std::cout << "Ja no primeiro semestre? ok, nao vou te impedir, adeus!\n" << std::endl;

269 sairDoJogo = true;

270 break;

271 }

272 default: {

273 std::cout << "Escolha um numero dentro das opcoes, nem vem...!\n" << std::endl;

274 break;

275 }

276 }

277 }

278

279 return 0;

280 }

#### void vitoria (int *vida*)

Função para imprimir uma mensagem de vitória.

##### Parâmetros

| *vida* | O valor da vida do personagem. |
| --- | --- |

92 {

93 std::cout << "------------------------------------------------------------- \n" << std::endl;

94 std::cout << "Uauu! Parece que voce conseguiu passar do segundo semestre\n" << std::endl;

95 std::cout << "Seu NSG final e de " << vida << " o que significa que no boletim voce tem um: " << std::endl;

96 if (vida >= 90) {

97 ext::print\_a();

98 } else if (vida >= 80 && vida < 90) {

99 ext::print\_b();

100 } else if (vida >= 70 && vida < 80) {

101 ext::print\_c();

102 } else if (vida >= 60 && vida < 70) {

103 ext::print\_d();

104 } else {

105 std::cout << "Parece que mesmo passando, seu NSG ficou abaixo de 60, mas por misericordia os professores arrendondaram, agradeca\n" << std::endl;

106 }

107 }

### Variáveis

#### Item itens[10]

**Valor inicial:**= {

Item("Calculadora", 10),

Item("Video aula no youtube", 15),

Item("Cafezinho", 20),

Item("Pao de queijo", 25),

Item("Dica de veterano", 30),

Item("Monitoria", 35),

Item("Calculadora cientifica", 45),

Item("Pao de queijo recheado", 45),

Item("Abraco da mamae", 50),

Item("Listas do professor", 30)

}

23 {

24 Item("Calculadora", 10),

25 Item("Video aula no youtube", 15),

26 Item("Cafezinho", 20),

27 Item("Pao de queijo", 25),

28 Item("Dica de veterano", 30),

29 Item("Monitoria", 35),

30 Item("Calculadora cientifica", 45),

31 Item("Pao de queijo recheado", 45),

32 Item("Abraco da mamae", 50),

33 Item("Listas do professor", 30)

34 };

## Referência do Arquivo código/Personagem.cpp

Implementação da classe **Personagem**.

#include "Personagem.hpp"

### Descrição detalhada

Implementação da classe **Personagem**.

## Referência do Arquivo código/Personagem.hpp

#include <iostream>

#include "Item.hpp"

#include <vector>

### Componentes

class **Personagem***Classe para representar um* ***Personagem****.*

## Personagem.hpp

Ir para a documentação desse arquivo.

1 #ifndef PERSONAGEM\_HPP

2 #define PERSONAGEM\_HPP

3

4 #include <iostream>

5 #include "Item.hpp"

6 #include <vector>

7

11 class Personagem {

12 std::string \_nome;

13 int \_vida;

14 int \_vida\_maxima;

15 int \_ataque1;

16 int \_ataque2;

17 std::vector<Item> inventario;

19 public:

24 Personagem(std::string nome);

25

30 void recebeDano(int dano);

31

35 void printInfo();

36

42 int ataca(int escolha);

43

48 void adicionarItem(Item item);

49

55 int usarItem(int index);

56

61 bool estaVivo();

62

67 std::string getNome();

68 };

69

70 #endif

## Referência do Arquivo código/ProfAnalNumerica.cpp

#include "ProfAnalNumerica.hpp"

## Referência do Arquivo código/ProfAnalNumerica.hpp

#include "Inimigo.hpp"

#include <iostream>

### Componentes

class **ProfAnalNumerica***Classe que representa um* ***Inimigo*** *do tipo "Professor de Análise Numérica". Herda da classe* ***Inimigo****.*

## ProfAnalNumerica.hpp

Ir para a documentação desse arquivo.

1 #ifndef PROFANALNUMERICA\_HPP

2 #define PROFANALNUMERICA\_HPP

3

4 #include "Inimigo.hpp"

5 #include <iostream>

6

11 class ProfAnalNumerica : public Inimigo {

12 int \_ataque;

14 public:

19 ProfAnalNumerica(string nome);

20

25 int ataca() override;

26

30 void falar() override;

31 };

32

33 #endif

## Referência do Arquivo código/ProfCalculo2.cpp

Implementação da classe **ProfCalculo2**.

#include "ProfCalculo2.hpp"

### Descrição detalhada

Implementação da classe **ProfCalculo2**.

## Referência do Arquivo código/ProfCalculo2.hpp

#include "Inimigo.hpp"

### Componentes

class **ProfCalculo2***Classe que representa um* ***Inimigo*** *do tipo "Professor de Cálculo 2". Herda da classe* ***Inimigo****.*

## ProfCalculo2.hpp

Ir para a documentação desse arquivo.

1 #ifndef PROFCALCULO2\_HPP

2 #define PROFCALCULO2\_HPP

3

4 #include "Inimigo.hpp"

5

10 class ProfCalculo2 : public Inimigo {

11 int \_ataque;

13 public:

18 ProfCalculo2(string nome);

19

24 int ataca() override;

25

29 void falar() override;

30 };

31

32 #endif

## Referência do Arquivo código/ProfFundMec.cpp

Implementação da classe **ProfFundMec**.

#include "ProfFundMec.hpp"

### Descrição detalhada

Implementação da classe **ProfFundMec**.

## Referência do Arquivo código/ProfFundMec.hpp

#include "Inimigo.hpp"

#include <iostream>

### Componentes

class **ProfFundMec***Classe que representa um* ***Inimigo*** *do tipo "Professor de Fundamentos Mecânicos". Herda da classe* ***Inimigo****.*

## ProfFundMec.hpp

Ir para a documentação desse arquivo.

1 #ifndef PROFFUNDMEC\_HPP

2 #define PROFFUNDMEC\_HPP

3

4 #include "Inimigo.hpp"

5 #include <iostream>

6

11 class ProfFundMec : public Inimigo {

12 int \_ataque;

14 public:

19 ProfFundMec(string nome);

20

25 int ataca() override;

26

30 void falar() override;

31 };

32

33 #endif

## Referência do Arquivo código/ProfPds2.cpp

Implementação da classe **ProfPDS2**.

#include "ProfPDS2.hpp"

### Descrição detalhada

Implementação da classe **ProfPDS2**.

## Referência do Arquivo código/ProfPds2.hpp

#include "Inimigo.hpp"

#include <iostream>

### Componentes

class **ProfPDS2***Classe que representa um* ***Inimigo*** *do tipo "Professor de Projeto e Desenvolvimento de Sistemas 2". Herda da classe* ***Inimigo****.*

## ProfPds2.hpp

Ir para a documentação desse arquivo.

1 #ifndef PROFPDS\_HPP

2 #define PROFPDS\_HPP

3

4 #include "Inimigo.hpp"

5 #include <iostream>

6

11 class ProfPDS2 : public Inimigo {

12 int \_ataque;

14 public:

19 ProfPDS2(string nome);

20

25 int ataca() override;

26

30 void falar() override;

31 };

32

33 #endif

## Referência do Arquivo código/ProfSistemasDigitais.cpp

Implementação da classe **ProfSD**.

#include "ProfSistemasDigitais.hpp"

### Descrição detalhada

Implementação da classe **ProfSD**.

## Referência do Arquivo código/ProfSistemasDigitais.hpp

#include "Inimigo.hpp"

#include <iostream>

### Componentes

class **ProfSD***Classe que representa um* ***Inimigo*** *do tipo "Professor de Sistemas Digitais". Herda da classe* ***Inimigo****.*

## ProfSistemasDigitais.hpp

Ir para a documentação desse arquivo.

1 #ifndef ProfSD\_HPP

2 #define ProfSD\_HPP

3

4 #include "Inimigo.hpp"

5 #include <iostream>

6

11 class ProfSD : public Inimigo {

12 int \_ataque;

14 public:

19 ProfSD(string nome);

20

25 int ataca() override;

26

30 void falar() override;

31 };

32

33 #endif

# Sumário

INDEX