**Requisitos funcionais**

***Todos os atributos e funções membros devem estar relacionados a classe***

1. Pelo menos 4 atributos

Em Picareta.h

int tipoN;

string tipoS;

Bloco matMine;

Em Ferramenta.h

int tipoFerramenta;

1. Pelo menos 4 funções membros sem incluir get e set

Em Ferramenta.h

virtual void jogarNoChao();

virtual void checarEstado() const;

virtual inline void destruir();

virtual bool quebrarBloco(Bloco &);

**Requisitos de implementação**

1. Todos os atributos devem ser inicializados. Fez validação de dados

Atributos da classe Bloco:

string nome;

vector<float> resMat;

vector<bool> colherMat;

Inicialização dos atributos no construtor:

Bloco::Bloco(string nome, float resMat[5], bool colherMat[5]) {

this->setNome(nome);

this->setResMat(resMat);

this->setColherMat(colherMat);

}

Atributos da classe Ferramenta:

int tipoFerramenta;

int maxRes;

int atualRes;

Inicialização dos atributos no construtor:

Ferramenta::Ferramenta(int tipoFerramenta, int maxRes, int atualRes) {

this->tipoFerramenta = tipoFerramenta;

this->setMaxRes(maxRes);

this->setAtualRes(atualRes);

}

Atributos da classe Picareta:

int tipoN;

string tipoS;

Bloco matMine;

static int numPicaretasQuebradas;

1. Três construtores, incluindo um construtor de cópia e construtor com parâmetros defaults. Verifica alocação dentro do construtor de cópia.

Três construtores para Picareta:

Picareta(string = "madeira", int = 0, int = 60, int = 60, Bloco = Bloco());

Picareta(const Picareta &);

Picareta(int);

1. Deve ter um atributo string

Atributo string em Picareta.h:

string tipoS;

1. Um atributo static. Correta modelagem dos statics?

Inicialização do atributo static(Na classe Picareta):

int Picareta::numPicaretasQuebradas = 0;

Utilização do atributo static:

virtual inline void destruir(){

numPicaretasQuebradas++;

cout << "Sua Picareta de " << this->tipoS << " quebrou." << endl;

cout << "Construa uma picareta Nova" << endl;

}

1. Um atributo const static
2. Dois métodos constantes (não pode ser get)
3. Um array
4. Uma função inline (não pode ser get ou set)
5. Método com passagem por referência usando ponteiro
6. Método static – deve ser chamado no main
7. Composição com a classe Data. Fez uso do objeto criado?
8. O que é const deve ser const.
9. Alocação dinâmica de memória. A memória é desalocada?
10. friend Operator<<
11. Operator=
12. vector push\_back

Requisitos para as classes adicionais (pelo menos duas)

* Operator =
* Alocação dinâmica - se houver vazamento de memória a classe toda é desconsiderada
* Usar o destrutor
* Construtor de cópia
* Operator << friend
* Um const static

**Requisitos herança**

* Diagrama de classes (obrigatório salvar também o png do diagrama no gitHub)
  + Herança pública
  + Construtor de cópia, e sobrecargas dos operadores de atribuição (=) e << (cout << base) para a classe base e derivada
  + Usar Protected acessando diretamente os atributos na classe derivada
  + Alocação dinâmica de memória na classe base e derivada
  + Sobrescrita de método: chamar dentro do método da classe derivada o método correspondente da classe base usando ::
  + No main: criar um ponteiro da classe base para alocar memória para a classe derivada e chamar os vários métodos implementados