**Nome: Bruno Deluca Satil Cassiano**

**1 - Armazene em uma estrutura de dados struct dados a respeito de um prédio de 4 apartamentos. É preciso armazenar os seguintes dados do apartamento: número de portas, números de janelas e metragem quadrada. No final o programa deve dar o valor em real dos 4 apartamentos individualmente que são calculados de acordo com as seguintes diretrizes: 1 porta = +R$ 200,00, 1 janela = +R$ 100,00, metro quadrado construído = +R$ 4500,00 o metro quadrado.**

#include <iostream>

using namespace std;

struct Apartamento{

double QuantidadePortas;

double QuantidadeJanelas;

double MetrosQuadrados;

void insere(double qtd\_portas, double qtd\_janelas, double metros\_quadrados) {

QuantidadePortas = qtd\_portas;

QuantidadeJanelas = qtd\_janelas;

MetrosQuadrados = metros\_quadrados;

}

void calcular(int apartamento) {

int total = 0;

total = (QuantidadePortas \* 200) + (QuantidadeJanelas \* 100) + (MetrosQuadrados \* 4500);

cout << "O valor total do " <<apartamento << "º apartamento é R$" <<total << endl;

}

};

int main(){

Apartamento ap1, ap2, ap3, ap4;

Apartamento \*apartamento = new Apartamento[4];

apartamento[0] = ap1;

apartamento[1] = ap2;

apartamento[2] = ap3;

apartamento[3] = ap4;

for (int x = 0; x < 4; x++) {

apartamento[x].insere((x + 1), (x + 1 \* 20), (x + 1 \* 14954));

apartamento[x].calcular(x + 1);

}

return 0;

}

**2 - Imagine que você está criando um jogo de videogame do gênero RPG e precisa de um programa para armazenar os dados de 5 inimigos diferentes, cada inimigo possui 6 atributos diferentes.**

**A grandeza desses atributos é definida por um valor que vai 1 até 10. Os atributos são os seguintes: força, destreza, constituição, inteligência, sabedoria e carisma. O nome do inimigo também deve ser armazenado.**

**Faça um programa que o usuário entre com todos esses valores e após adicionados a uma estrutura de dados do tipo struct, o valor de todos os inimigos devem ser exibidos na tela.**

#include <iostream>

using namespace std;

bool verificar\_valor(int valor) {

if (valor < 0 || valor > 10) {

cout<<"Valor do status inválido, o mínimo é 0 e o máximo é 10, tente novamente!" <<endl <<endl <<endl;

return false;

} else {

return true;

}

}

string nome\_status(int idStatus) {

switch (idStatus) {

case 1:

return "a força";

case 2:

return "a destreza";

case 3:

return "a constituição";

case 4:

return "a inteligência";

case 5:

return "a sabedoria";

case 6:

return "o carisma";

default:

return "PERSONAGEM CORROMPIDO, FUJAM PRAS COLINAS!";

}

}

string nome\_inimigo(int idInimigo) {

switch (idInimigo) {

case 1:

return "Hunter";

case 2:

return "Reaper";

case 3:

return "Demolisher";

case 4:

return "Bomber";

case 5:

return "Crumbler";

default:

return "Monstro Surpresa";

}

}

struct Inimigos {

int Forca;

int Destreza;

int Constituicao;

int Inteligencia;

int Sabedoria;

int Carisma;

void inserir\_status(int forca, int destreza, int constituicao, int inteligencia, int sabedoria, int carisma) {

Forca = forca;

Destreza = destreza;

Constituicao = constituicao;

Inteligencia = inteligencia;

Sabedoria = sabedoria;

Carisma = carisma;

}

void menu\_status(int inimigo) {

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

cout<<"Status naturais do " <<nome\_inimigo(inimigo) <<": " <<endl;

cout<<" - Força: " <<Forca <<endl;

cout<<" - Destreza: " <<Destreza <<endl;

cout<<" - Constituição: " <<Constituicao <<endl;

cout<<" - Inteligência: " <<Inteligencia <<endl;

cout<<" - Sabedoria: " <<Sabedoria <<endl;

cout<<" - Carisma: " <<Carisma <<endl;

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

}

};

int main()

{

Inimigos hunter, reaper, demolisher, bomber, crumbler;

Inimigos \*inimigos = new Inimigos[5];

inimigos[0] = hunter;

inimigos[1] = reaper;

inimigos[2] = demolisher;

inimigos[3] = bomber;

inimigos[4] = crumbler;

int statusDoInimigo[6];

for (int x = 0; x < 5; x++) {

if (x == 0) {

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

cout<<"| Preencha os status do primeiro inimigo! |" <<endl;

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

} else {

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

cout<<"| Preencha os status do próximo inimigo! |" <<endl;

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

}

for (int y = 0; y < 6;) {

cout<<"Insira " <<nome\_status(y+1) <<" do " <<nome\_inimigo(x+1) <<": ";

cin>>statusDoInimigo[y];

cout<<endl <<endl;

if (verificar\_valor(statusDoInimigo[y]) == false) {

continue;

} else {

y++;

}

}

inimigos[x].inserir\_status(

statusDoInimigo[0],

statusDoInimigo[1],

statusDoInimigo[2],

statusDoInimigo[3],

statusDoInimigo[4],

statusDoInimigo[5]

);

}

for (int x = 0; x < 5; x++) {

inimigos[x].menu\_status(x+1);

}

return 0;

}

**3 - Implemente um TAD Jogador de futebol.**

**Cada jogador possui os campos, número de jogos, gols, assistências e chutes a gol.**

**Implemente as operações:**

**Atribuir valores para os campos citados acima**

**Imprimir os dados/estatísticas do jogador**

**Aproveitamento do jogador (chutes a gol/gols)**

**E através de comentários defina: O que é a aplicação, especificação e implementação.**

Arquivo main.cpp:

//Aplicação

#include <iostream>

#include "Jogadores.hpp"

using namespace std;

int main()

{

Jogadores Kaka;

Kaka.inserir(2,3,4,5);

Kaka.exibir\_estatisticas();

cout<<endl <<"O aproveitamento foi de " <<Kaka.aproveitamento() <<"%";

return 0;

}

Arquivo Jogadores.cpp:

//Implementação

#include "Jogadores.hpp"

#include <iostream>

using namespace std;

void Jogadores::inserir(int numero\_de\_jogos, int gols, int assistencias, int chutes\_ao\_gol) {

NumeroDeJogos = numero\_de\_jogos;

Gols = gols;

Assistencias = assistencias;

ChutesAoGol = chutes\_ao\_gol;

}

void Jogadores::exibir\_estatisticas() {

cout<<endl <<endl;

cout<<"Este jogador participou de " <<NumeroDeJogos <<" jogos." <<endl;

cout<<"Tentou " <<ChutesAoGol <<" chutes ao gol," <<" e marcou " <<Gols <<" gols." <<endl;

cout<<"Além de que também realizou " <<Assistencias <<" assistências.";

}

double Jogadores::aproveitamento() {

if (ChutesAoGol != 0 && Gols != 0) {

return (Gols \* 100) / ChutesAoGol;

} else {

return 0;

}

}

Arquivo Jogadores.hpp:

//Especificação

struct Jogadores {

int NumeroDeJogos;

int Gols;

int Assistencias;

int ChutesAoGol;

void inserir(int numero\_de\_jogos, int gols, int assistencias, int chutes\_ao\_gol);

void exibir\_estatisticas();

double aproveitamento();

};

**4 - Implemente um TAD avião particular**

**Devem ser levadas em consideração dados do avião em relação a sua manutenção, considere motores, aviônica, fuselagem (número de horas que uma peça irá precisar de manutenção) além do seu nome e do seu ano de fabricação.**

**Implemente as operações:**

**Definir número de horas de vôo para a manutenção dos motores, da aviônica e da fuselagem.**

**Inserir em um looping infinito o número de horas de voo do avião (vôo após vôo).**

**Se determinada peça atingir o número de horas de vôo limite para a sua manutenção, deve ser impresso um alarme na tela falando para o avião fazer a sua manutenção**

**E através de comentários defina: O que é a aplicação, especificação e implementação.**

//Main.cpp

//Aplicação

#include "Avioes.hpp"

#include <iostream>

#include <string>

using namespace std;

int main()

{

string nomeAviao = "";

cout<<"Qual o nome do avião? ";

cin>>nomeAviao;

cout<<endl <<endl;

int dadosAviao[4];

cout<<"Qual o ano de fabricação do avião? ";

cin>>dadosAviao[0];

cout<<endl <<endl;

cout<<"Qual a duração dos motores do avião? ";

cin>>dadosAviao[1];

cout<<endl <<endl;

cout<<"Qual a duração da aviônica do avião? ";

cin>>dadosAviao[2];

cout<<endl <<endl;

cout<<"Qual a duração da fuselagem do avião? ";

cin>>dadosAviao[3];

cout<<endl <<endl;

Avioes aviao;

aviao.inserir(nomeAviao, dadosAviao[0], dadosAviao[1], dadosAviao[2], dadosAviao[3]);

bool tester = false;

int horas;

while (tester == false) {

cout<<"Insira 0 (ZERO) para parar o programa ou o tempo de vôo do " <<aviao.Nome <<": ";

cin>>horas;

cout<<endl;

if (horas == 0) {

tester = true;

} else {

aviao.tempo\_de\_voo(horas);

aviao.condicao\_das\_pecas();

}

}

return 0;

}

//Avioes.cpp

//Implementação

#include "Avioes.hpp"

#include <iostream>

#include <string>

using namespace std;

void Avioes::inserir(

string nome,

int ano\_de\_fabricacao,

int duracao\_motores,

int duracao\_avionica,

int duracao\_fuselagem

) {

Nome = nome;

AnoDeFabricacao = ano\_de\_fabricacao;

DuracaoMotores = duracao\_motores;

DuracaoAvionica = duracao\_avionica;

DuracaoFuselagem = duracao\_fuselagem;

}

void Avioes::condicao\_das\_pecas() {

cout<<endl;

if (DuracaoMotores == 0 || DuracaoMotores < 0) {

cout<<"Os motores chegaram ao fim da vida útil recomendada, faça a manutenção e troque-o!" <<endl;

} else {

cout<<"Os motores ainda possuem " <<DuracaoMotores <<" horas de vida útil." <<endl;

}

if (DuracaoAvionica == 0 || DuracaoAvionica < 0) {

cout<<"A aviônica chegou ao fim da vida útil recomendada, faça a manutenção e troque-a!" <<endl;

} else {

cout<<"A aviônica ainda possuem " <<DuracaoAvionica <<" horas de vida útil." <<endl;

}

if (DuracaoFuselagem == 0 || DuracaoFuselagem < 0) {

cout<<"A fuselagem chegou ao fim da vida útil recomendada, faça a manutenção e troque-a!" <<endl;

} else {

cout<<"A fuselagem ainda possui " <<DuracaoFuselagem <<" horas de vida útil." <<endl;

}

cout<<endl;

}

void Avioes::tempo\_de\_voo(int horas) {

DuracaoMotores = DuracaoMotores - horas;

DuracaoAvionica = DuracaoAvionica - horas;

DuracaoFuselagem = DuracaoFuselagem - horas;

}

//Avioes.hpp

//Especificação

#include <string>

using namespace std;

struct Avioes {

string Nome;

int AnoDeFabricacao;

int DuracaoMotores;

int DuracaoAvionica;

int DuracaoFuselagem;

void inserir(

string nome,

int ano\_de\_fabricacao,

int duracao\_motores,

int duracao\_avionica,

int duracao\_fuselagem

);

void condicao\_das\_pecas();

void tempo\_de\_voo(int horas);

};