Lista 08

1-

#include <iostream>

class Autor {

public:

    std::string nome;

    std::string nacionalidade;

    Autor(const std::string& nome, const std::string& nacionalidade)

        : nome(nome), nacionalidade(nacionalidade) {}

};

class Livro {

public:

    std::string titulo;

    Autor\* autor;

    Livro(const std::string& titulo, Autor\* autor)

        : titulo(titulo), autor(autor) {}

    void mostrarInformacoes() {

        std::cout << "Livro: " << titulo << std::endl;

        std::cout << "Autor: " << autor->nome << ", Nacionalidade: " << autor->nacionalidade << std::endl;

    }

};

int main() {

    Autor autor("José de Alencar", "Brasileiro");

    Livro livro("O Guarani", &autor);

    livro.mostrarInformacoes();

    return 0;

}

2-

#include <iostream>

class Medico {

public:

    std::string nome;

    std::string especialidade;

    std::string crm;

    Medico(const std::string& nome, const std::string& especialidade, const std::string& crm)

        : nome(nome), especialidade(especialidade), crm(crm) {}

};

class Paciente {

public:

    std::string nome;

    std::string cpf;

    std::string data\_nascimento;

    Paciente(const std::string& nome, const std::string& cpf, const std::string& data\_nascimento)

        : nome(nome), cpf(cpf), data\_nascimento(data\_nascimento) {}

};

class Consulta {

public:

    std::string data;

    std::string hora;

    Medico\* medico;

    Paciente\* paciente;

    Consulta(const std::string& data, const std::string& hora, Medico\* medico, Paciente\* paciente)

        : data(data), hora(hora), medico(medico), paciente(paciente) {}

    void mostrarInformacoes() {

        std::cout << "Consulta agendada para: " << data << " às " << hora << std::endl;

        std::cout << "Médico: " << medico->nome << ", Especialidade: " << medico->especialidade << std::endl;

        std::cout << "Paciente: " << paciente->nome << ", CPF: " << paciente->cpf << std::endl;

    }

};

int main() {

    Medico medico1("Dr. Silva", "Cardiologia", "CRM12345");

    Paciente paciente1("João", "123.456.789-00", "1990-01-01");

    Consulta consulta1("2023-11-13", "15:00", &medico1, &paciente1);

    consulta1.mostrarInformacoes();

    return 0;

}

3-

#include <iostream>

class Quarto {

public:

    int numero;

    std::string tipo;

    double preco\_diaria;

    bool disponivel;

    Quarto(int numero, const std::string& tipo, double preco\_diaria)

        : numero(numero), tipo(tipo), preco\_diaria(preco\_diaria), disponivel(true) {}

};

class Reserva {

public:

    std::string data\_entrada;

    std::string data\_saida;

    std::string nome\_cliente;

    Quarto\* quarto;

    Reserva(const std::string& data\_entrada, const std::string& data\_saida, const std::string& nome\_cliente, Quarto\* quarto)

        : data\_entrada(data\_entrada), data\_saida(data\_saida), nome\_cliente(nome\_cliente), quarto(quarto) {

            quarto->disponivel = false; // Marcar o quarto como não disponível ao realizar a reserva

    }

    void mostrarInformacoes() {

        std::cout << "Reserva para: " << nome\_cliente << std::endl;

        std::cout << "Data de entrada: " << data\_entrada << ", Data de saída: " << data\_saida << std::endl;

        std::cout << "Quarto: Número " << quarto->numero << ", Tipo: " << quarto->tipo << std::endl;

    }

    void cancelarReserva() {

        quarto->disponivel = true; // Marcar o quarto como disponível ao cancelar a reserva

    }

};

int main() {

    Quarto quarto(101, "Deluxe", 200.0);

    Reserva reserva("2023-11-13", "2023-11-15", "Maria", &quarto);

    reserva.mostrarInformacoes();

    return 0;

}

4-

#include <iostream>

class Produto {

public:

    std::string nome;

    double preco;

    std::string descricao;

    Produto(const std::string& nome, double preco) : nome(nome), preco(preco), descricao("") {}

};

class Pedido {

public:

    int numero;

    std::string data;

    Produto\*\* produtos;

    int numProdutos;

    Pedido(int numero) : numero(numero), data(""), produtos(new Produto\*[10]), numProdutos(0) {}

    void adicionarProduto(const Produto& produto) {

        produtos[numProdutos++] = new Produto(produto);

    }

    double calcularTotal() {

        double total = 0.0;

        for (int i = 0; i < numProdutos; ++i) {

            total += produtos[i]->preco;

        }

        return total;

    }

    ~Pedido() {

        for (int i = 0; i < numProdutos; ++i) {

            delete produtos[i];

        }

        delete[] produtos;

    }

};

int main() {

    Pedido pedido(123);

    pedido.adicionarProduto(Produto("Caneta", 1.50));

    pedido.adicionarProduto(Produto("Caderno", 5.00));

    std::cout << "Total do Pedido: " << pedido.calcularTotal() << std::endl;

    return 0;

}

5-

#include <iostream>

class Curso {

public:

    std::string nome;

    std::string codigo;

    Curso(const std::string& nome, const std::string& codigo) : nome(nome), codigo(codigo) {}

};

class Departamento {

public:

    std::string nome;

    std::string localizacao;

    Curso\*\* cursos;

    int numCursos;

    Departamento(const std::string& nome, const std::string& localizacao)

        : nome(nome), localizacao(localizacao), cursos(new Curso\*[10]), numCursos(0) {}

    void adicionarCurso(const Curso& curso) {

        cursos[numCursos++] = new Curso(curso);

    }

    void removerCurso(const std::string& codigo) {

        for (int i = 0; i < numCursos; ++i) {

            if (cursos[i]->codigo == codigo) {

                delete cursos[i];

                cursos[i] = cursos[numCursos - 1];

                cursos[--numCursos] = nullptr;

                break;

            }

        }

    }

    ~Departamento() {

        for (int i = 0; i < numCursos; ++i) {

            delete cursos[i];

        }

        delete[] cursos;

    }

};

int main() {

    Departamento departamento("Ciências", "Bloco 1 (único)");

    departamento.adicionarCurso(Curso("Matemática", "MAT101"));

    departamento.adicionarCurso(Curso("Física", "FIS102"));

    std::cout << "Departamento: " << departamento.nome << std::endl;

    for (int i = 0; i < departamento.numCursos; ++i) {

        std::cout << "Curso: " << departamento.cursos[i]->nome << ", Código: " << departamento.cursos[i]->codigo << std::endl;

    }

    return 0;

}

6-

#include <iostream>

class Motorista {

public:

    std::string nome;

    std::string cnh;

    Motorista(const std::string& nome, const std::string& cnh)

        : nome(nome), cnh(cnh) {}

};

class Veiculo {

public:

    std::string modelo;

    std::string ano;

    std::string placa;

    Veiculo(const std::string& modelo, const std::string& ano, const std::string& placa)

        : modelo(modelo), ano(ano), placa(placa) {}

};

class Viagem {

public:

    std::string destino;

    double distancia;

    Motorista\* motorista;

    Veiculo\* veiculo;

    Viagem(const std::string& destino, Motorista& motorista, Veiculo& veiculo)

        : destino(destino), distancia(0.0), motorista(&motorista), veiculo(&veiculo) {}

    void mostrarInformacoes() {

        std::cout << "Viagem para: " << destino << std::endl;

        std::cout << "Motorista: " << motorista->nome << ", CNH: " << motorista->cnh << std::endl;

        std::cout << "Veículo: Modelo " << veiculo->modelo << ", Placa: " << veiculo->placa << std::endl;

    }

};

int main() {

    Motorista motorista("João", "ABC1234");

    Veiculo veiculo("Carro", "2023", "XYZ-9999");

    Viagem viagem("Rio de Janeiro", motorista, veiculo);

    viagem.mostrarInformacoes();

    return 0;

}

Lista 11

1-

#include <iostream>

int main() {

    int numero = 10; // Variável inteira

    int\* ponteiro\_numero = &numero; // Ponteiro para a variável inteira

    std::cout << "Variável inteira: " << numero << std::endl;

    std::cout << "Valor do ponteiro: " << ponteiro\_numero << std::endl; // Endereço de memória armazenado no ponteiro

    std::cout << "Valor apontado pelo ponteiro: " << \*ponteiro\_numero << std::endl; // Valor apontado pelo ponteiro

    return 0;

}

2-

#include <iostream>

int main() {

    int numero = 10; // Variável inteira

    int& referencia\_numero = numero; // Referência para a variável inteira

    std::cout << "Variável inteira: " << numero << std::endl;

    std::cout << "Referência para a variável: " << &referencia\_numero << std::endl; // Endereço de memória da referência

    std::cout << "Valor da referência: " << referencia\_numero << std::endl; // Valor da variável referenciada

    return 0;

}

3-

#include <iostream>

int main() {

    int array[] = {33, 55, 77, 99, 11}; // Array de inteiros

    int\* ponteiro = array; // Ponteiro para o primeiro elemento do array

    for (int i = 0; i < 5; ++i) {

        std::cout << "Valor " << i + 1 << " do array (usando ponteiro): " << \*ponteiro << std::endl;

        ponteiro++; // Avança para o próximo elemento do array

    }

    return 0;

}

4-

#include <iostream>

int\* criarArray() {

    int\* array = new int[5]{33, 55, 77, 99, 11}; // Aloca dinamicamente um array de inteiros

    return array;

}

int main() {

    int\* ponteiro = criarArray(); // Recebe o ponteiro para o array criado dinamicamente

    for (int i = 0; i < 5; ++i) {

        std::cout << "Valor " << i + 1 << " do array: " << ponteiro[i] << std::endl;

        std::cout << "Endereço do valor " << i + 1 << ": " << &ponteiro[i] << std::endl;

    }

    delete[] ponteiro; // Libera a memória alocada dinamicamente

    return 0;

}

5-

#include <iostream>

int\* capture\_array() {

    int\* array = new int[5]{33, 55, 77, 99, 11}; // Aloca dinamicamente um array de inteiros

    return array;

}

void imprime\_array(int\* ptr, int tamanho) {

    std::cout << "Valores do array:" << std::endl;

    for (int i = 0; i < tamanho; ++i) {

        std::cout << ptr[i] << " ";

    }

    std::cout << std::endl;

}

void trocar\_valores(int\* ptr, int index1, int index2) {

    int temp = ptr[index1];

    ptr[index1] = ptr[index2];

    ptr[index2] = temp;

}

int main() {

    int\* ptr = capture\_array();

    imprime\_array(ptr, 5);

    trocar\_valores(ptr, 1, 3);

    imprime\_array(ptr, 5);

    delete[] ptr; // Libera a memória alocada dinamicamente

    return 0;

}

6-

#include <iostream>

void modificaComPonteiro(int\* ptr) {

    \*ptr = 20; // Modifica o valor apontado pelo ponteiro

}

void modificaComReferencia(int& ref) {

    ref = 20; // Modifica diretamente o valor referenciado pela referência

}

int main() {

    int valor = 10; // Valor inicial

    modificaComPonteiro(&valor); // Chama a função com ponteiro

    std::cout << "Valor modificado com ponteiro: " << valor << std::endl;

    modificaComReferencia(valor); // Chama a função com referência

    std::cout << "Valor modificado com referência: " << valor << std::endl;

    return 0;

}

7-

#include <iostream>

bool check\_vogais(char myc) {

    return myc == 'a' ||

           myc == 'e' ||

           myc == 'i' ||

           myc == 'o' ||

           myc == 'u';

}

int contar\_vogais(char\* txt) {

    int count = 0;

    while (\*txt != '\0') { // Enquanto não chegarmos ao final da string ('\0')

        if (check\_vogais(\*txt)) { // Verifica se o caractere atual é uma vogal

            count++; // Se sim, incrementa o contador de vogais

        }

        txt++; // Move para o próximo caractere na string

    }

    return count;

}

void ler\_teclado(char\* texto, int tamanho) {

    std::cout << "Digite uma frase (max " << tamanho - 1 << " caracteres): ";

    std::cin.getline(texto, tamanho);

}

int main() {

    char texto[50];

    ler\_teclado(texto, sizeof(texto));

    std::cout << "Você digitou a frase abaixo:\n";

    std::cout << texto << std::endl;

    int num = contar\_vogais(texto);

    std::cout << "Total de vogais: " << num << std::endl;

    return 0;

}

8-

#include <iostream>

void encontrarMaiorValor(int\* array, int tamanho) {

    int maiorValor = \*array; // Inicializa a variável maiorValor com o primeiro elemento do array

    int indice = 0; // Inicializa o índice do maior valor como 0

    for (int i = 1; i < tamanho; ++i) {

        if (\*(array + i) > maiorValor) { // Compara cada elemento do array com o maiorValor atual

            maiorValor = \*(array + i); // Se o elemento atual for maior, atualiza o maiorValor

            indice = i; // Atualiza o índice do maior valor

        }

    }

    std::cout << "Maior valor encontrado: " << maiorValor << std::endl;

    std::cout << "Índice do maior valor: " << indice << std::endl;

}

int main() {

    int array[] = {10, 5, 30, 15, 25}; // Exemplo de array de inteiros

    int tamanho = sizeof(array) / sizeof(array[0]); // Calcula o tamanho do array

    encontrarMaiorValor(array, tamanho); // Chama a função para encontrar o maior valor no array

    return 0;

}

9-

#include <iostream>

void achar\_menor(int\* arr, int tamanho) {

    int\* ptr = arr;

    int menor = INT\_MAX; // Valor inicial do menor como o maior valor possível de um inteiro

    int idx = -1;

    for (int i = 0; i < tamanho; ++i) {

        if (\*(ptr + i) < menor) {

            menor = \*(ptr + i);

            idx = i;

        }

    }

    std::cout << "Menor valor encontrado: " << menor << std::endl;

    std::cout << "Índice do menor valor: " << idx << std::endl;

}

int main() {

    int meu\_array[] = {88, 22, -11, 44, -77, 99};

    achar\_menor(meu\_array, 6);

    return 0;

}

10-

#include <iostream>

class Estudante {

private:

    std::string nome;

    int idade;

public:

    Estudante(std::string nome, int idade) : nome(nome), idade(idade) {}

    int getIdade() const {

        return idade;

    }

    std::string getNome() const {

        return nome;

    }

};

void estudante\_maior(Estudante\* lista, int tamanho) {

    int maiorIdade = -1;

    int idx = -1;

    for (int i = 0; i < tamanho; ++i) {

        if (lista[i].getIdade() > maiorIdade) {

            maiorIdade = lista[i].getIdade();

            idx = i;

        }

    }

    if (idx != -1) {

        std::cout << "Estudante com maior idade: " << lista[idx].getNome() << std::endl;

        std::cout << "Idade: " << lista[idx].getIdade() << std::endl;

        std::cout << "Posição no array: " << idx << std::endl;

    } else {

        std::cout << "Nenhum estudante encontrado!" << std::endl;

    }

}

int main() {

    Estudante e1 = Estudante("nome1", 22);

    Estudante e2 = Estudante("nome2", 33);

    Estudante e3 = Estudante("nome3", 19);

    Estudante e4 = Estudante("nome4", 26);

    Estudante e5 = Estudante("nome5", 30);

    Estudante estudantes[] = {e1, e2, e3, e4, e5};

    estudante\_maior(estudantes, sizeof(estudantes) / sizeof(estudantes[0]));

    return 0;

}

11-

#include <iostream>

#include <cmath>

class Ponto {

private:

    int x, y;

public:

    Ponto(int x, int y) : x(x), y(y) {}

    int distancia(const Ponto\* outroPonto) const {

        int dx = outroPonto->x - x;

        int dy = outroPonto->y - y;

        return sqrt(dx \* dx + dy \* dy);

    }

};

int main() {

    Ponto ptoA(1, 1);

    Ponto ptoB(5, 5);

    int d1 = ptoA.distancia(&ptoB);

    int d2 = ptoB.distancia(&ptoA);

    std::cout << "d1: " << d1 << std::endl;

    std::cout << "d2: " << d2 << std::endl;

    return 0;

}

12-

#include <iostream>

#include <string>

class Pessoa {

private:

    std::string nome;

    int idade;

public:

    Pessoa(std::string nome, int idade) : nome(nome), idade(idade) {}

    void mostra\_info() {

        std::cout << "Nome: " << nome << ", Idade: " << idade << std::endl;

    }

    int getIdade() const {

        return idade;

    }

};

Pessoa\* criar\_pessoa() {

    std::string nome;

    int idade;

    std::cout << "Digite o nome da pessoa: ";

    std::cin >> nome;

    std::cout << "Digite a idade da pessoa: ";

    std::cin >> idade;

    return new Pessoa(nome, idade);

}

void my\_init(Pessoa\* todos[], int TOTAL\_PESSOAS) {

    std::cout << "\nCriando todas pessoas...\n";

    for (int idx = 0; idx < TOTAL\_PESSOAS; idx++) {

        todos[idx] = criar\_pessoa();

    }

}

void my\_print\_all(Pessoa\* todos[], int TOTAL\_PESSOAS) {

    std::cout << "\nPrint all...\n";

    for (int idx = 0; idx < TOTAL\_PESSOAS; idx++) {

        todos[idx]->mostra\_info();

    }

}

Pessoa\* verifica\_maior\_idade(Pessoa\* todos[], int TOTAL\_PESSOAS) {

    Pessoa\* pessoaMaiorIdade = todos[0];

    for (int idx = 1; idx < TOTAL\_PESSOAS; idx++) {

        if (todos[idx]->getIdade() > pessoaMaiorIdade->getIdade()) {

            pessoaMaiorIdade = todos[idx];

        }

    }

    return pessoaMaiorIdade;

}

int main() {

    const int TOTAL\_PESSOAS = 5;

    Pessoa\* pessoas[TOTAL\_PESSOAS];

    my\_init(pessoas, TOTAL\_PESSOAS);

    my\_print\_all(pessoas, TOTAL\_PESSOAS);

    Pessoa\* result = verifica\_maior\_idade(pessoas, TOTAL\_PESSOAS);

    std::cout << "\nMaior idade:" << std::endl;

    result->mostra\_info();

    // Liberar a memória alocada dinamicamente

    for (int i = 0; i < TOTAL\_PESSOAS; ++i) {

        delete pessoas[i];

    }

    return 0;

}

13-

#include <iostream>

#include <string>

class Aluno {

private:

    std::string nome;

    int idade;

    int notas[3];

public:

    Aluno(std::string nome, int idade, int nota1, int nota2, int nota3) : nome(nome), idade(idade) {

        notas[0] = nota1;

        notas[1] = nota2;

        notas[2] = nota3;

    }

    void mostra\_info() {

        std::cout << "Nome: " << nome << ", Idade: " << idade << ", Notas: ";

        for (int i = 0; i < 3; ++i) {

            std::cout << notas[i] << " ";

        }

        std::cout << std::endl;

    }

    int getIdade() const {

        return idade;

    }

    int getNotaMaisAlta() const {

        int maxNota = -1;

        for (int i = 0; i < 3; ++i) {

            if (notas[i] > maxNota) {

                maxNota = notas[i];

            }

        }

        return maxNota;

    }

};

class Turma {

private:

    Aluno\* alunos[3];

public:

    Turma() {

        alunos[0] = new Aluno("Ana", 25, 8, 9, 7);

        alunos[1] = new Aluno("João", 30, 7, 6, 9);

        alunos[2] = new Aluno("Maria", 20, 9, 8, 8);

    }

    Aluno\* alunoMaiorIdade() {

        Aluno\* alunoMaior = alunos[0];

        for (int i = 1; i < 3; ++i) {

            if (alunos[i]->getIdade() > alunoMaior->getIdade()) {

                alunoMaior = alunos[i];

            }

        }

        return alunoMaior;

    }

    Aluno\* alunoNotaMaisAlta() {

        Aluno\* alunoNotaAlta = alunos[0];

        for (int i = 1; i < 3; ++i) {

            if (alunos[i]->getNotaMaisAlta() > alunoNotaAlta->getNotaMaisAlta()) {

                alunoNotaAlta = alunos[i];

            }

        }

        return alunoNotaAlta;

    }

    ~Turma() {

        for (int i = 0; i < 3; ++i) {

            delete alunos[i];

        }

    }

};

int main() {

    Turma turma;

    Aluno\* alunoMaiorIdade = turma.alunoMaiorIdade();

    std::cout << "Aluno de maior idade: ";

    alunoMaiorIdade->mostra\_info();

    Aluno\* alunoNotaMaisAlta = turma.alunoNotaMaisAlta();

    std::cout << "Aluno com a nota mais alta: ";

    alunoNotaMaisAlta->mostra\_info();

    return 0;

}