Nome: Tiago Bertoline

Lista de Exercícios Ponteiros

1) R:

```
#include <iostream>
#include <locale>

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

using namespace std;

struct Computador {     char marca[30];   // Array de
caracteres para a marca     char modelo[30];  // Array de
caracteres para o modelo     float preco;      // Variável do
tipo float para o preço
};


int main(int argc, char** argv) {
        setlocale(LC_ALL, "Portuguese");
int qtd;

        // Chama o struct.
        Computador p;

        // Ponteiro que armazena o struct
        Computador* ponteiro = &p;



        cout << "Informe quantidade que deseja armazenar:";
cin >> qtd;
        cin.ignore();// Limpa o buffer do teclado

        // Alocando os elementos usando o ponteiro
    for (int i = 0; i < qtd; i++) {
                        cout << "Informe a marca do computador:";
```

```
                                cin >> p.marca;
        cin.ignore();                                cout <<
"Informe a modelo do computador:";
        cin >> p.modelo;

                                cout << "Informe o preço do computador:";
                        cin >> p.preco;

    }

    // Exibir os dados armazenados
        for (int i = 0; i < qtd; i++) {         cout <<
"Computador " << i + 1 << ": \n";         cout <<
"Marca: " << p.marca << "\n";         cout <<
"Modelo: " << p.modelo << "\n";         cout <<
"Preço: " << p.preco << "\n\n";
    }

            delete ponteiro; // Libera a memória



        return 0;
}
```

2) R:

```
#include <iostream>
#include <locale>
#include <cstring>  // Para strcpy()

using namespace std;

struct Carro {     char marca[30];  // Array de caracteres para armazenar a
marca do carro     int ano;         // Ano do carro
};

// Função para alterar o ano do carro void
alterarAno(Carro carros[], int indice, int novoAno) {
carros[indice].ano = novoAno;
```

```
}
int main() {     setlocale(LC_ALL,
"Portuguese");

    int qtd;

    cout << "Informe a quantidade de carros que deseja armazenar: ";
cin >> qtd;     cin.ignore();  // Limpa o buffer do teclado

    // Criar um array dinâmico para armazenar os carros
    Carro* carros = new Carro[qtd];

    // Entrada de dados dos carros      for (int i = 0; i < qtd;
i++) {         cout << "Informe a marca do carro " << i + 1
<< ": ";         cin.getline(carros[i].marca, 30);

        cout << "Informe o ano do carro " << i + 1 << ": ";
cin >> carros[i].ano;
        cin.ignore();  // Limpa o buffer
    }

    // Escolher um carro para alterar o ano     int indice;     cout << "\nEscolha o
índice do carro que deseja alterar (0 a " << qtd - 1 <<
"): ";
    cin >> indice;

    // Validar índice     if (indice < 0 ||
indice >= qtd) {         cout << "Índice
inválido!\n";         delete[] carros;  //
Liberar memória         return 1;
    }

    // Pedir o novo ano     int novoAno;     cout << "Digite o novo ano para o
carro " << carros[indice].marca << ": ";     cin >> novoAno;
    // Chamar a função para alterar o ano
alterarAno(carros, indice, novoAno);

    // Mostrar os carros atualizados     cout << "\nLista de carros
atualizada:\n";     for (int i = 0; i < qtd; i++) {         cout << i << " - " <<
carros[i].marca << " (" << carros[i].ano << ")\n";
```

```
    }

    // Liberar memória alocada
delete[] carros;

    return 0;
}
```

3) R:

```
#include <iostream>
#include <locale>

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

//No main(), declare um array com 3 produtos e um ponteiro para esse array.
//Use o ponteiro
//para percorrer os produtos e imprimir suas informações.

using namespace std;

struct Produto {     char nome[30];   // Array de
caracteres para a nome     float preco;      // Variável
do tipo float para o preço
};


int main(int argc, char** argv) {
        setlocale(LC_ALL, "Portuguese");

    // Chama o struct e passa o array de produtos.
    Produto produtos[3] = {
        {"Arroz", 20.0},
        {"Feijão", 7.0},
        {"Colher", 2.0}
    };

    // Criando um ponteiro para o array de produtos
    Produto* ponteiro = produtos;
```

```
    // Percorrendo o array com o ponteiro e exibindo os produtos
for (int i = 0; i < 3; i++) {         cout << "Produto: " << (ponteiro +
i)->nome << "\n";         cout << "Preço: R$ " << (ponteiro + i)-
>preco << "\n\n";
    }
                    delete ponteiro; // Libera a
memória.

          return 0;
}
```

4) R:

```
#include <iostream>

using namespace std;

// Estrutura que representa um nó da lista struct
No {
    int dado;       // Valor armazenado no nó
    No* proximo;    // Ponteiro para o próximo nó
};

// Função para inserir um elemento no início da lista void
inserirInicio(No*& cabeca, int valor) {     No* novoNo = new No;  //
Aloca um novo nó      novoNo->dado = valor; // Atribui o valor
novoNo->proximo = cabeca; // Aponta para o antigo primeiro nó
cabeca = novoNo;       // Atualiza a cabeça da lista
}

// Função para exibir os elementos da lista
void exibirLista(No* cabeca) {     No* atual
= cabeca;     while (atual != nullptr) {
cout << atual->dado << " -> ";         atual =
atual->proximo;
    }
    cout << "NULL" << endl;
}
```

```
int main() {
    No* lista = nullptr; // Inicializa a lista vazia

    // Adicionando 3 números à lista
inserirInicio(lista, 10);     inserirInicio(lista,
20);     inserirInicio(lista, 30);

    // Exibindo a lista
cout << "Lista ligada: ";
exibirLista(lista);

    delete lista; // Libera a memória.

    return 0;
}
```

5) R:

```
#include <iostream>
#include <locale>

using namespace std;

// Estrutura struct
Aluno {     char
nome[50];
float nota;
};

int main() {
    setlocale(LC_ALL, "Portuguese");

    int qtd;

    cout << "Informe a quantidade de alunos que deseja armazenar: ";
cin >> qtd;     cin.ignore(); // Limpa o buffer do teclado

    // Alocando dinamicamente um array de Alunos
    Aluno* ponteiro = new Aluno[qtd];
```

```
    // Inserção dos dados     for (int i = 0; i < qtd; i++) {
cout << "\nInforme o nome do aluno " << i + 1 << ": ";
cin.getline(ponteiro[i].nome, 50);

        cout << "Informe a nota do aluno " << i + 1 << ": ";
cin >> ponteiro[i].nota;         cin.ignore(); // Limpa o
buffer do teclado
    }

    // Exibição dos dados armazenados     cout <<
"\n=== Lista de Alunos ===\n";     for (int i = 0; i <
qtd; i++) {         cout << "Nome: " <<
ponteiro[i].nome << "\n";         cout << "Nota: " <<
ponteiro[i].nota << "\n";         cout << "----------------
------\n";
    }

    // Liberação de memória
delete[] ponteiro;

    return 0;
}
```

6) R:

```
#include <iostream>
#include <locale>

using namespace std;

// Estrutura struct
Funcionario {
char nome[50];
float salario;
};

int main() {     setlocale(LC_ALL,
"Portuguese");

    // Alocando dinamicamente um array de Alunos
    Funcionario* ponteiro = new Funcionario[5];
```

```
    // Preenchendo os valores dos funcionários
    for (int i = 0; i < 5; i++ ){         cout << "Digite o nome do
funcionario " << i + 1 << ": ";         cin.ignore(); // Limpa o
buffer antes de usar getline
cin.getline(ponteiro[i].nome, 50);

        cout << "Digite o salario do funcionario " << i + 1 << ": ";
cin >> ponteiro[i].salario;
    }

    // Exibindo as informações dos funcionários
cout << "\nLista de Funcionarios:\n";     for (int
i = 0; i < 5; i++) {
        cout << "Nome: " << ponteiro[i].nome << " - Salario: R$ " <<
ponteiro[i].salario << endl;
    }
    // Liberação de memória
delete[] ponteiro;

    return 0;
}
```

7) R:

```
#include <iostream>
#include <locale>
#include <cstring> // Para usar strcpy()

using namespace std;

// Estrutura corrigida struct Livro {     char titulo[50]; // Agora
pode armazenar um título completo     int ano;
};

// Função para preencher os dados do livro void
criarLivro(Livro* l, const char* titulo, int ano) {     strcpy(l-
>titulo, titulo); // Copia o título para a estrutura     l->ano
= ano;
}
```

```
int main() {     setlocale(LC_ALL,
"Portuguese");

    // Alocando dinamicamente um array de Livros
    Livro* ponteiro = new Livro[5];

    // Preenchendo os valores com os livros     for (int i = 0; i < 5; i++) {
char tituloTemp[50]; // Variável temporária para armazenar o título
int anoTemp;

        cout << "Digite o título do livro " << i + 1 << ": ";
cin.ignore(); // Limpa o buffer antes de usar getline
cin.getline(tituloTemp, 50);

        cout << "Digite o ano do livro " << i + 1 << ": ";
cin >> anoTemp;

        // Chamando a função para preencher a estrutura
criarLivro(&ponteiro[i], tituloTemp, anoTemp);
    }

    // Exibindo as informações dos livros
cout << "\nLista de livros cadastrados:\n";
    for (int i = 0; i < 5; i++) {
        cout << "Título: " << ponteiro[i].titulo << " - Ano: " << ponteiro[i].ano <<
endl;
    }

    // Liberação de memória
delete[] ponteiro;

    return 0;
}
```

8) **R:**

```
#include <iostream>
#include <cstring> // Para strcpy

using namespace std;
```

```
// Definição da struct Pessoa
struct Pessoa {     char
nome[50];     int idade;
};

// Função que modifica a idade através de ponteiro duplo
void modificarIdade(Pessoa** p, int novaIdade) {
    (*p)->idade = novaIdade;
}

int main() {
    // Criando um objeto Pessoa
    Pessoa pessoa;

    // Definindo valores iniciais
strcpy(pessoa.nome, "João");     pessoa.idade
= 25;

    // Criando um ponteiro para a struct
    Pessoa* ptr = &pessoa;

    // Exibindo os valores antes da modificação
    cout << "Antes da modificação: " << endl;     cout << "Nome: " <<
pessoa.nome << ", Idade: " << pessoa.idade << endl;

    // Chamando a função para modificar a idade
modificarIdade(&ptr, 30);

    // Exibindo os valores após a modificação     cout << "Depois da
modificação: " << endl;     cout << "Nome: " << pessoa.nome << ", Idade: "
<< pessoa.idade << endl;

    return 0;
}
```

9) R:

```
#include <iostream>

using namespace std;
```

```
// Definição da struct No
struct No {     int dado;
No* prox;
};

// Função para adicionar um elemento no início da lista
void inserirNoInicio(No*& cabeca, int valor) {     No*
novoNo = new No;     novoNo->dado = valor;
novoNo->prox = cabeca;     cabeca = novoNo;
}

// Função para remover um elemento da lista void
removerElemento(No*& cabeca, int valor) {
    No* atual = cabeca;
    No* anterior = nullptr;

    // Caso especial: remoção do primeiro nó     if
(atual != nullptr && atual->dado == valor) {
cabeca = atual->prox; // Novo início da lista
delete atual;         return;
    }

    // Percorre a lista buscando o nó a ser removido
while (atual != nullptr && atual->dado != valor) {
anterior = atual;         atual = atual->prox;
    }

    // Se não encontrou o valor, sai da função
if (atual == nullptr) return;

    // Ajusta os ponteiros e remove o nó
anterior->prox = atual->prox;     delete
atual;
}

// Função para exibir a lista ligada
void exibirLista(No* cabeca) {
No* temp = cabeca;     while (temp
!= nullptr) {         cout << temp-
```

```
>dado << " -> ";         temp =
temp->prox;
    }
    cout << "NULL" << endl;
}

int main() {
    No* cabeca = nullptr;

    // Criando uma lista com 3 valores: 10 -> 20 -> 30
inserirNoInicio(cabeca, 30);     inserirNoInicio(cabeca,
20);     inserirNoInicio(cabeca, 10);

    cout << "Lista antes da remoção: ";
exibirLista(cabeca);

    // Removendo um elemento (por exemplo, 20)
removerElemento(cabeca, 20);

    cout << "Lista depois da remoção: ";
exibirLista(cabeca);

    return 0;
}
```

10) R:

```
#include <iostream>

using namespace std;

// Definição da struct Ponto struct
Ponto {
    int x;
int y;
};

// Função para trocar os valores de x e y entre dois pontos
void trocarPontos(Ponto* a, Ponto* b) {
```

```
    // Troca os valores de x
int temp = a->x;     a->x =
b->x;     b->x = temp;

    // Troca os valores de y
temp = a->y;     a->y = b-
>y;     b->y = temp;
}

int main() {
    // Criando dois pontos
    Ponto p1 = {3, 5};
    Ponto p2 = {8, 12};

    // Exibindo os valores antes da troca     cout << "Antes da
troca:" << endl;     cout << "Ponto 1: (" << p1.x << ", " <<
p1.y << ")" << endl;     cout << "Ponto 2: (" << p2.x << ", " <<
p2.y << ")" << endl;

    // Chamando a função para trocar os valores
trocarPontos(&p1, &p2);

    // Exibindo os valores depois da troca     cout <<
"\nDepois da troca:" << endl;     cout << "Ponto 1: (" << p1.x
<< ", " << p1.y << ")" << endl;     cout << "Ponto 2: (" << p2.x
<< ", " << p2.y << ")" << endl;

    return 0;
}
```