TÉCNICAS DE PROGRAMAÇÃO II
EXERCÍCIOS LISTA 01: **ESTRUTURAS**
Profª Lucília Ribeiro

01 Escrever um programa que leia e escreva a informação de cem clientes de determinada empresa. Os clientes tem um nome, o número de unidades solicitadas de um produto, o preço de cada unidade e o status do pedido: em processamento, pago, atrasado.

02 Acrescentar ao programa anterior uma função que permita mostrar a fatura de todos os clientes de cada uma das categorias "processamento", "pago" ou "atrasado".

03 Modificar o programa anterior para obter os seguintes relatórios:
a) Clientes em processamento, com fatura inferior a determinada quantia e
b) Clientes com status "pago", com fatura maior que determinada quantia.

04 Deseja-se registrar uma estrutura **`PessoaFuncionario`** que contenha como membros os dados de uma pessoa, o salário e o número de horas trabalhadas por semana. Uma **`Pessoa`**, por sua vez, é outra estrutura que tem como membros o nome, a idade, a altura, o peso e a data de nascimento. Por sua vez, a data de nascimento é outra estrutura que contêm o dia, o mês e o ano. Escrever funções para ler e escrever um funcionário do tipo **`PessoaFuncionario`**.

05 Deseja-se informatizar os resultados obtidos pelas Ligas Esportivas da PUC. Cada equipe contém as seguintes informações:

- Nome da equipe
- Modalidade esportiva
- Número de vitórias
- Número de derrotas

Para as equipes de basquete, acrescente as seguintes informações:
- Número de perdas de bola
- Número de rebotes pegos
- Nome do melhor jogador de arremessos de três pontos
- Número de arremessos de três pontos do melhor jogador

Para as equipes de futebol, acrescente:
- Número de empates
- Número de gols a favor
- Número de gols contra
- Nome do goleador da equipe
- Número de gols do goleador

Escrever um programa para introduzir a informação para todas as equipes integrantes.

06 Modificar o programa anterior para obter os seguintes relatórios:

a) Listagem dos melhores jogadores de arremessos de três pontos de cada equipe
b) Máximo goleador da liga de futebol
c) Supondo que o jogo vencido equivale a 3 pontos e um empate a 1 ponto, faça uma listagem da equipe ganhadora
d) Equipe ganhadora da modalidade basquete.

07 Os protocolos IP de direcionamento da Internet definem o endereço de cada nó de uma rede como um inteiro longo, mas esse formato não é muito adequado para os usuários. Para a visualização dos usuários, costuma-se separar os quatro bytes por pontos. Escrever uma função que visualize um inteiro longo como um endereço de rede dos protocolos de Internet (quatro bytes do inteiro longo separados por pontos).

08 Uma livraria deseja ter o inventário de livros. Para isso, quer criar uma base de dados na qual são armazenadas as seguintes informações por livro: título do livro; data de publicação; autor; número total de livros existentes; número total de livros existentes em pedidos; preço de venda. Escrever funções que permitam ler os dados da base de dados e visualizar a base de dados.

ESTRUTURAS → Uma **struct** permite que seus membros dado dos mesmos ou de diferentes tipos se encapsulem em uma única implementação, ao contrário dos arrays, que são agregados de um tipo de dado simples. Os membros dado de uma estrutura são, por default, acessíveis publicamente, ainda que uma estrutura possa ter diferentes tipos de acesso ou visibilidade. Além disso, uma estrutura pode, de fato, encapsular funções que operam sobre os dados da estrutura, ainda que essas propriedades costumem reservar-se geralmente como **classe**.

Uma estrutura é um tipo de dado que contém elementos de tipos diferentes.

UNIÃO → O tipo de dado união é similar à estrutura, de maneira que é uma coleção de membros dado de tipos diferentes ou semelhantes, mas, ao contrário de uma definição struct, que aloca memória suficiente para conter todos os membros dado, uma **union** pode conter apenas um membro dado em qualquer momento e o tamanho de uma união é, como consequência, o tamanho de seu maior número.

Utilizando um tipo união, pode-se fazer que diferentes variáveis coexistam no mesmo espaço de memória. A união permite reduzir espaço de memória em seus programas.

ENUMERAÇÃO → uma enumeração **enum**, é um tipo definido pelo usuário com constantes de nome de tipo inteiro.

```
union Teste
{
	char letra;
	int elemento;
	float preco;
};

enum DiasSemana
{
	SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO
};
```

```cpp
#include <iostream>
using namespace std;

struct Aluno
{
    int matricula;
    float nota[3];
    float media;
};

int main()
{
    Aluno aluno;

    cout << "\nDigite o numero da matricula: ";
    cin >> aluno.matricula;
    aluno.nota[0] = 7.5;
    aluno.nota[1] = 5.2;
    aluno.nota[2] = 8.4;
    aluno.media = (aluno.nota[0] + aluno.nota[1] + aluno.nota[2]) / 3.0;

    cout << "\nMatricula...: " << aluno.matricula;
    cout << "\nMedia.......: " << aluno.media << endl;
    return 0;
}
```

```cpp
#include <iostream>
#include <cstring>
using namespace std;

struct Data
{
    int dia;
    int mes;
    int ano;
};

struct ColecaoCD
{
    string titulo;
    string artista;
    int qtdMusicas;
    float preco;
    Data dataCompra;
};

int main()
{
    Data dataNascimento = {30, 9, 1969};
    ColecaoCD cd1 = {
        "Musica para acampamentos",
        "Legiao Urbana",
        13,
        142.50,
        {23, 8, 2023}
    };
    ColecaoCD cd2;
    cout << "\nTitulo do CD...: " << cd1.titulo;
    cout << "\nArtista........: " << cd1.artista;
    cout << "\nAdquirido em...: " << cd1.dataCompra.dia << "/" << cd1.dataCompra.mes << "/" << cd1.dataCompra.ano;

    cout << "\n\nTamanho da estrutura de dados ColecaoCD = " << sizeof(ColecaoCD) << endl;
    cout << "Tamanho da estrutura de dados Data = " << sizeof(Data) << endl;
    cout << "Tamanho do elemento float = " << sizeof(cd1.preco) << endl;
    cout << "Tamanho do elemento inteiro = " << sizeof(cd1.qtdMusicas) << endl;
    cout << "Tamanho do elemento ano da Data = " << sizeof(dataNascimento) << endl;

    cout << "\nTitulo do CD...: ";
    getline(cin, cd2.titulo);
    cd2.artista = "Fulano de Tal";
    cd2.dataCompra = dataNascimento;

    cout << "\nTitulo do CD...: " << cd2.titulo;
    cout << "\nArtista........: " << cd2.artista;
    cout << "\nAdquirido em...: " << cd2.dataCompra.ano;
    cout << "\nQtde musicas...: " << cd2.qtdMusicas;
    cout << "\nValor..........: " << cd2.preco << endl;

    return 0;
}
```

```cpp
#include <iostream>
#include <cstring>
using namespace std;

struct Endereco
{
    string rua;
    int numero;
    string cidade;
    char estado[2];
    int cep;
};

struct Data
{
    int dia;
    int mes;
    int ano;
};

struct InfoPessoa
{
    string nome;
    Endereco endereco;
    Data dataNascimento;
};
```

```cpp
void verInfo(InfoPessoa &dados) //passagem por referência
{
    cout << "\n";
    cout << dados.nome << endl;
    cout << dados.endereco.rua << endl;
    cout << dados.dataNascimento.ano << endl;
};

int main()
{
    InfoPessoa clientes[3];

    for (int i = 0; i < 3; i++)
    {
        fflush(stdin);
        cout << "Dados do cliente " << (i + 1) << endl;
        cout << "Nome.............: ";
        getline(cin, clientes[i].nome);
        cout << "Endereco (rua).....: ";
        getline(cin, clientes[i].endereco.rua);
        cout << "Ano de nascimento..: ";
        cin >> clientes[i].dataNascimento.ano;
    }
    for (int i = 0; i < 3; i++)
    {
        verInfo(clientes[i]);
    }
    return 0;
}
```