Laboratório de ATP I
Lista: Structs e Registros - UNESP, IBILCE

1. Escreva uma programa que receba um número inteiro (que representa um intervalo de tempo medido em minutos) e que transforme esse intervalo no correspondente número de horas e minutos. Exemplo: converte 131 minutos em 2 horas e 11 minutos. Grave o resultado em uma *struct* tal como definida a seguir:

Algorithm 1: Programa converte minutos.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  //Definição da struct 'horamin'
4  typedef struct horamin
5  {
6      int horas;
7      int minutos;
8  } horamin;
9  //Programa principal
10 int main()
11 ...
```

2. Crie um programa que leia e permita armazenar o **nome**, **altura** e **peso** de 5 pessoas. Cada pessoa deve ser representada por uma *struct*, que deverá ser alocada dentro de um vetor (vetor de *structs*). Após isso, o programa deverá:

(a) Ler um valor de altura e um de peso específico;
(b) Listar todos os nomes com alturas maiores daquela que foi fornecida em (a);
(c) Listar todos os nomes com respectivos pesos acima do peso fornecido em (a).

3. Crie uma *struct* capaz de armazenar datas com três campos do tipo inteiro (**dia**, **mês** e **ano**). Em seguida, faça um programa que leia cada um dos campos de uma *struct* do tipo acima e que imprima na tela a seguinte formatação de data:

- dia/mês por extenso/ano (Exemplo: 10/julho/2022).

4. Considerando a *struct* do *Algoritmo 2* para representar um número complexo, implemente um programa que calcule:

(a) a soma entre dois números complexos;
(b) o produto entre eles;
(c) o módulo de um número complexo;
(d) o conjugado de um número complexo;
(e) o ângulo (argumento) de um número complexo.

Algorithm 2: Definição da *struct* número complexo.

```
1 //Definição da struct numero 'complexo'
2 typedef struct complexo
3 {
4     float re;
5     float im;
6 } complexo;
```

5. Utilizando *struct*, fazer um programa que permita a entrada de **nome**, **endereço** e **celular** (apenas números, sem DDD) de 5 pessoas. Imprimir os dados de cada uma das pessoas cadastradas. Utilize o seguinte formato de saída de dados:

Fulano | Rua dos Toddynhos 185, Jd Yakult | 995674235

Goku | Rua do Kamisama 555, Planeta Vegeta | 992636789

e assim sucessivamente.

6. Faça um programa usando *struct* que controle o consumo de energia dos eletrodomésticos de uma residência. Mais especificamente,

(a) Criar e ler 10 eletrodomésticos de uma casa. Cada eletrodoméstico deverá conter: **nome** (máximo de 15 letras), **potencial** (real, em kW), e **tempo ativo por dia** (real, em horas).

(b) Ler um tempo t (em dias), calcular e mostrar o consumo total da residência, e os consumos relativos de cada eletrodoméstico (consumo/consumo total) nesse período de tempo. Apresente este último dado em porcentagem.

7. Crie uma *struct* (cliente banco) que contenha os seguintes dados bancários de um cliente: **nome** (string), **agência** (inteiro), **conta** (inteiro), e **saldo** (real). Em seguida:

(a) Leia os dados de um cliente, gravando-os em uma variável *struct*. Leia também a operação bancária desejada pelo cliente, isto é: (1) para saque, (2) para depósito.

(b) Imprimir na tela o valor atualizado após o saque (caso selecione (1)), ou o valor agregado final na conta após o depósito (caso selecione (2)).

8. Crie um programa que converta coordenadas polares para cartesianas, isto é:

(a) Crie uma *struct* que represente um ponto em coordenadas polares: um raio (r), e um argumento (ângulo $theta$), dado em radianos.

(b) Crie uma *struct* que represente um ponto em coordenadas cartesianas: duas coordenadas reais, x e y.

(c) No programa principal, leia um ponto em coordenada polar e converta-o para coordenadas cartesianas através da fórmula: $x = r\,cos(theta)$ e $y = r\,sen(theta)$. Grave o resultado em uma variável do tipo coordenada cartesiana.

9. Crie um programa que leia um vetor com os dados de 10 carros (uma *struct*): **marca** (máximo de 20 caracteres), **ano**, e **preço**. No programa principal, leia um valor p e mostre as informações de todos os carros com preço menor ou igual que p. O programa deve repetir esse processo até que seja digitado o valor $p = 0$, quando o mesmo deverá ser encerrado.

10. Considere a seguinte *struct*:

Algorithm 3: Definição da *struct* paciente.

```
1 typedef struct
2 {
3     char nome[100];
4     char sexo;    //'m': masculino, 'f': feminino
5     float peso;
6     float altura;
7     char cpf[12]; //apenas numeros
8 } paciente;
```

Leia os dados de 10 pacientes (vetor de *struct*). Dado um cpf digitado pelo usuário, identificar no vetor a pessoa detentora desse cpf, calcular e mostrar na tela o seu IMC.

Fórmula IMC: $peso/(alutra \times altura)$