LISTA DE EXERCÍCIOS

- Structs e Modularização -

"may the force be with you"

Exercícios de Structs

Problema #01

Todo computador possui informações básicas, sendo as principais: Arquitetura, Modelo da CPU, Qtde de Núcleos, Velocidade do Clock (em GHz), Espaço de Memória RAM e de Memória Secundária (em GBs), se possui GPU dedicada, Modelo GPU…
Modele uma struct para representar todas essas informações sobre um computador.
Faça o cadastro de N structs (até o usuário informar Qtde. CPUs <= 0).
Após a etapa de cadastro, liste todos os equipamentos cadastrados em formato de relatório.

Problema #02

Faça um programa em C que, utilizando a biblioteca customizada "`datetime.h`" preenche um array contendo 10 horários aleatórios: `dt_rand_time()`. Imprima todos os horários obtidos.
Após isso, obtenha a hora atual do sistema (pela própria biblioteca) e imprima qual dos horários existentes no array é o mais próximo do horário atual do sistema, tanto do passado quanto do futuro (Imagine como se fosse um alerta para o último compromisso e próximo compromisso).

Problema #03

Um ano tem 365 dias (às vezes 366). Qual a chance de duas pessoas fazerem aniversário no mesmo dia?
O paradoxo do aniversário diz que, se em uma festa tiver pelo menos 23 convidados, a chance de duas pessoas fazerem aniversário no mesmo dia é de 50%. Se tiver 30 pessoas, a chance aumenta para 70%, e se tiver 50 pessoas apenas, a chance é de 97%. Interessante não?!
Faça um programa que leia um valor N correspondente ao número de convidados de uma festa.
Gere N datas aleatórias - `dt_rand_birth()`, verifique se há ou não pessoas que podem comemorar juntos, e que dia é esse. Informe também, dentre todas as datas geradas, qual dia da semana em que mais pessoas nasceram, e que dia da semana é esse.

Problema #04

Faça um programa que preencha o cadastro de N pessoas. "Pessoa" possui os atributos "Nome" (string) e "Data de Nascimento". O usuário deve preencher apenas a informação "Nome".
"Data de Nascimento" deve ser gerada aleatoriamente através da biblioteca "`datetime.h`".
Imprima o relatório de pessoas cadastradas, ORDENADAS pelas suas idades atuais (Obs.: verifique se a pessoa já completou ou não o aniversário no ano atual).

Problema #05

O gestor de arquivos de qualquer sistema operacional apresenta informações básicas dos arquivos em um diretório, tais como: Nome do Arquivo (s), Tamanho em Bytes (i), Data de Criação do Arquivo (datetime), Tipo de Arquivo (Imagem, Documento, Vídeo, Diretório, etc.). Faça um programa que simule o gestor de arquivos de um S.O. Na primeira etapa, faça o cadastro de N arquivos. Encerre essa fase quando o usuário informar a palavra "exit" para o nome do arquivo. Na segunda etapa, o programa deverá listar todos os arquivos na ordem de preferência do usuário. Digitando a letra 'N' o sistema deverá imprimir a lista ordenada considerando os Nomes. Digitando 'B' considere a quantidade de Bytes. Digitando 'D' considere a data de criação, e digitando 'T' imprima a lista considerando o tipo. Repita várias opções de listagem, até informar 'X' para encerrar a execução.

Problema #06

Uma pequena mercearia deseja informatizar o processo de estoque e venda de produtos.
Faça um programa que alimente um vetor com até 100 registros de Produto: Código de Barras (i), Descrição (s), Qtde. em Estoque (i), Valor Unitário (f).
Após cadastrar o estoque existente, o programa entrará no modo de venda (PDV). Neste modo, o programa seguirá o seguinte fluxo:

- O caixa informa o Código de Barra do produto a ser vendido;
- O programa verifica e imprime todas informações do produto (ou se não existe/leitura incorreta);
- O caixa informa a Quantidade que deseja comprar deste item;
- O programa verifica se possui a quantidade desejada em estoque;
- Caso seja possível realizar a venda, o programa contabiliza o valor a ser pago e atualiza a quantidade daquele item no estoque;

O Programa deve repetir todo o fluxo acima para vários itens, até que o Código de Barras informado seja um valor negativo; Caso seja informado um código negativo, o programa deve então imprimir o valor total da venda realizada, receber o valor pago pelo cliente, e calcular o troco a ser devolvido.
Iniciar uma nova operação de venda logo em seguida.

Problema #07

Toda conta de usuário em redes sociais contém algumas informações básicas, tais como:
@ (identificador único), endereço de e-mail, senha de acesso e o nome de apresentação do usuário.
Faça um programa que, utilizando Structs, realize o cadastro de N contas nesta "rede".
O Identificador @ deve ser validado, pois deve ser exclusivo para cada cadastro.
A senha deve conter mais de 6 dígitos e pelo menos uma letra e um símbolo numérico.
Após o processo de cadastro, implemente uma tela que simula o login nesta rede, pedindo ao usuário uma string (que pode ser tanto o e-mail ou o @) e a senha.
Informe como resultado: o nome de apresentação da pessoa que se autenticou (se a conta existe e senha confere), se a conta não existe, ou se a senha informada está incorreta.

Problema #08

CRM (*Customer Relationship Management*) é um sistema de informação voltado para a Gestão de Relacionamento com Clientes. Desenvolva um *software* para ajudar no envio de mensagens de felicitações para os aniversariantes.
Modele uma struct Cliente contendo: nome, telefone, email, e data de nascimento (geradas aleatoriamente pela biblioteca "`datetime.h`").
Faça o cadastro de vários clientes (até informar "`exit`" para encerrar).
Após a fase de cadastro, imprima a relação dos aniversariantes de um determinado mês informado pelo usuário. Essa listagem deverá estar ordenada pelo dia de aniversário, e deve informar, em formato de uma tabela, os seguintes dados: dia/mês/ano de nascimento, nome e e-mail, e idade que completará.
Repita a operação para novos meses, à escolha do usuário. Finalize quando informar um valor <= 0.

Problema #09

Faça um programa que detalhe o consumo de energia dos eletrodomésticos de sua casa. Modele uma struct contendo: nome do dispositivo, potência nominal (Watts) e tempo ativo aproximado por dia (em minutos). Cadastre vários eletrodomésticos (até o usuário desejar interromper).
Leia o valor médio do KWh cobrado pela concessionária de energia, e imprima relatório contendo:

- Consumo kWh diário (1 kWh == 1000 watts por hora).
- Consumo kWh mensal.
- Participação percentual relativa do consumo de cada eletrodoméstico, em relação ao total.
- Valor aproximado correspondente do equipamento na conta de energia.

Problema #10

Uma reta em um plano bidimensional pode ser representada a partir de dois pontos: ponto A, composto pelas coordenadas Xa e Ya, e ponto B, representado pelas coordenadas Xb e Yb.
Faça um programa que modele *structs* do tipo **Ponto** e tipo **Reta2D**.

Faça com que o usuário cadastre duas retas, utilizando a definição de **Reta2D**.

O programa deve calcular informar se as duas retas informadas pelo usuário são **PARALELAS**, **PERPENDICULARES** ou **NENHUMA** das duas.

Obs.: Utilize o conceito de "coeficiente angular de reta" para determinar estas condições.

Problema #11

Faça um programa que gere aleatoriamente 10 pontos na tela do seu monitor. Cada PONTO é formado pelo par de valores: COORDENADA X e COORDENADA Y (considere que a tela de impressão possua no máximo 20 linhas e 80 colunas).

Após sorteados, os pontos (de 0 a 9) deverão ser impressos no seu terminal através do seu próprio identificador. Como no exemplo abaixo. Obs.: É necessário a biblioteca "`gconio.h`"

```
 7                        6                   2
           0
                                     5                  4
    1
                        8                                      3
                    9
```

Em seguida, o usuário deverá informar para o programa o PONTO ALVO "X". O programa deverá imprimir o valor X na posição do ponto alvo e informar qual dos pontos sorteados é o mais próximo deste alvo.

```
 7                        6                   2
           0                                              X
                                     5                 4
    1
                        8                                      3
                    9
Informe coordenada X do ponto alvo: 70
Informe coordenada Y do ponto alvo: 20
O ponto sorteado mais próximo do alvo é o 4
```

DICA: *Cálculo da distância entre dois pontos: SQRT((x2-x1) ^ 2 + (y2-y1) ^ 2)*

Problema #12

Uma biblioteca deseja informatizar o arquivamento e a consulta do acervo de livros que possui.

Faça um programa que utilize *structs* do tipo Livro: Título (s), Autor (s), Área do Conhecimento (s), Ano (i) e Localização (i), para cadastrar "N" Livros.

A informação de **Localização não deve ser preenchida pelo usuário**, mas pelo próprio sistema, dadas as seguintes regras...

- Após o usuário cadastrar todos os Livros, o programa deverá ordená-los pela: "**Área do conhecimento**" e, dentro de cada Área, pelo "**Título do livro**", como o exemplo abaixo:

Área de Conhecimento: Análise de Sistemas

Título	**Autor**	**Ano**	**Localização**
Algoritmos	José	2012	---
Banco de Dados	Maria	2013	---
Cálculo Numérico	João	2011	---
Estrutura de Dados	Adriano	2016	---

Área de Conhecimento: Biologia

Título	**Autor**	**Ano**	**Localização**
Citologia	Antônio	2016	---
Genética	Lucas	2003	---

Área de Conhecimento: Engenharia Civil

Título	**Autor**	**Ano**	**Localização**
Cálculo	João	2014	---
Desenho	Pedro	2002	---

Uma vez organizados, o **programa** deverá distribuir os livros automaticamente nas prateleiras, seguindo a seguinte lógica: As prateleiras são sequenciais. Cada prateleira suporta no máximo 03 livros. Livros de "Áreas de Conhecimento" diferentes não podem ficar na mesma prateleira.

O resultado do cenário de exemplo seria, portanto...

Área de Conhecimento: Análise de Sistemas			
Título	**Autor**	**Ano**	**Localização**
Algoritmos	José	2012	01
Banco de Dados	Maria	2013	01
Cálculo Numérico	João	2011	01
Estrutura de Dados	Adriano	2016	02
Área de Conhecimento: Biologia			
Título	**Autor**	**Ano**	**Localização**
Citologia	Antônio	2016	03
Genética	Lucas	2003	03
Área de Conhecimento: Engenharia Civil			
Título	**Autor**	**Ano**	**Localização**
Cálculo	João	2014	04
Desenho	Pedro	2002	04

Exercícios de Structs e Funções

Problema #13

Refatore o código do problema da Mercearia anterior (venda de produtos), modularizando-o ao máximo. Crie um menu interativo com as seguintes opções:

1 - Cadastrar Produto
2 - Relatório de Produtos
3 - Atualizar Estoque de Produto
4 - Atualizar Preço de Produto
5 - Vender Produtos
6 - Sangria de Caixa

Cada opção acima estará vinculada à uma única função responsável pelo processamento da tarefa. Ao final de cada tarefa, o programa deverá retornar a este menu inicial.

Problema #14

Faça um programa que modele uma struct ContaBancaria contendo: Nome do Titular, Número da Conta e Saldo Atual.

Número da Conta deve ser gerado pelo próprio sistema, seguindo a regra de validação: **ABCD-V**, onde:

V é um dígito verificador, no qual: V = |A - B + C - D|
se V >= 10, então V == d1+d2 (a soma dos dois dígitos).

A interface do programa deve conter as opções:

- Abrir Nova Conta
- Mostrar Dados da Conta
- Retirada de Dinheiro
- Depósito de Dinheiro

Problema #15

Faça um programa em C que implemente uma struct “Compromisso”, contendo: Texto que descreve algum compromisso, Data/Hora do compromisso (utilize a estrutura `datetime` definida pela biblioteca "`datetime.h`")

Implemente as seguintes funcionalidades:

- Cadastrar compromisso (use a função `dt_create()` para criar a datetime);
- Listar todos os compromissos do dia atual do sistema, ordenado pelo horário (ordem crescente).
- Função para consultar apenas o compromisso mais próximo, considerando a data/horário atual.
- Função para excluir um determinado compromisso cadastrado anteriormente.
- Função que, dada uma data específica, lista todos os compromissos daquele dia, ordenados pelos horários (ordem crescente).

Problema #16

Faça um programa que controle o fluxo de voos nos aeroportos do Brasil. Modele as structs:

Aeroporto: Código IATA: 3 letras que representam o aeroporto (e não podem se repetir), nome da cidade, porte (pequeno, médio ou grande).

Voo: Código do Voo (Inteiro), Aeroporto de Origem (código IATA), Aeroporto de Destino (código IATA), tamanho da aeronave (P, M ou G).

- Faça funções para cadastro de Aeroportos e Voos. Todas as informações e restrições devem ser validadas.
- Faça função para imprimir a quantidade de voos que se originam e a quantidade de voos que se destinam a cada aeroporto cadastrado.
- Faça função para pesquisar Voos cuja Origem é um determinado código IATA (informado pelo usuário).
- Faça função para pesquisar Voos cujo Destino é um determinado código IATA (informado pelo usuário).

Problema #17

Desenvolva um aplicativo "Livro de Receitas". Crie uma estrutura RECEITA com as seguintes informações: NOME da receita, TEMPO de preparo (em minutos), DIFICULDADE (F - M - D) e INGREDIENTES. Cada ingrediente possui as seguintes informações: DESCRIÇÃO do ingrediente, QUANTIDADE. Uma receita suporta até 30 ingredientes.

Seu aplicativo deve oferecer as seguintes funcionalidades:

- Cadastrar Receita (Uma por vez).
- Consultar uma Receita (O usuário deve informar o nome ou apenas parte do nome de uma receita buscada).
- Consultar todas as receitas que possuem um determinado ingrediente (ou parte do nome de um ingrediente) informado pelo usuário.

Problema #18

Modele uma struct para controlar as ações de uma Bolsa de Valores, com as seguintes informações: Nome da Ação (String), Histórico de Preços (Array de Floats). O histórico de preços de cada ação deve armazenar os valores dos últimos 365 dias, sendo o índice 0 do array o preço atual (gere aleatoriamente estes valores, dentro de um intervalo entre R$ 15,01 e R$ 45,99 reais).

Implemente funções específicas para:

- Consultar o Valor Atual, os Valores Históricos dos últimos N dias dessa ação, e a variação % de cada dia. (N deve ser informado pelo usuário).
- Imprimir em ordem decrescente, as ações que mais se valorizaram percentualmente considerando os últimos N dias. (N deve ser informado pelo usuário).
- Imprimir em ordem decrescente, as ações com maior preço de mercado.
- Imprimir em ordem crescente, as ações com menor preço de mercado.

Obs: Otimize ao máximo sua função de ordenação para uso em todos os casos necessários.

Problema #19 [Questão de Prova]

Uma grande empresa de transportes de mercadorias deseja que você desenvolva um programa para gerenciar a logística de entrega de produtos. O programa deverá ser capaz de armazenar ROTAS entre cidades do país.

Cada ROTA possui as seguintes informações: cidade de ORIGEM, cidade de DESTINO, DISTÂNCIA (em km.) e TEMPO de viagem (em horas). Mas atenção! **Nenhuma** cidade pode ser ORIGEM para duas rotas distintas, ou seja, cada rota deve possuir uma origem exclusiva, e seu programa deve garantir essa restrição. Seu aplicativo deve oferecer as seguintes funcionalidades:

- Cadastrar rotas (Sendo um único cadastro por vez).
- Consultar a **distância**, o **tempo** de entrega total, e a **quantidade de viagens** que um produto deve sofrer para ser entregue, considerando uma cidade de ORIGEM e uma cidade de DESTINO informada pelo usuário.
- OBS.: Lembre que a rota também pode não ser possível...

Problema #20 [Questão de Prova]

O Google quer te contratar, e solicitou que você programe uma ferramenta para administração de contatos do Gmail. O sistema deverá permitir o cadastro de até 100 itens, contendo as informações: nome do contato, e-mail, telefone e grupo de afinidade (p.e. Família, Trabalho, Amigo, etc...). O seu programa deverá cumprir os seguintes requisitos:

- Realizar o cadastro de um contato por vez.
- Fazer a validação do endereço de e-mail através de uma função **validaEmail**. Para um e-mail ser válido, deve possuir um único símbolo “@”, pelo menos um símbolo ponto (.) após o símbolo “@” e não possuir espaços em branco.
- O cadastro deve aceitar somente endereços de e-mails válidos.
- Antes de o usuário cadastrar o grupo de afinidade para um novo contato, o programa deve exibir todos os grupos de afinidade já cadastrados em contatos anteriores (Dica: Crie um procedimento exclusivamente para isso). Desta forma, o usuário poderá ver os grupos que já foram criados antes de escolher o grupo do novo contato.
- Desenvolva um relatório (em procedimento próprio) que imprima na tela todos os contatos de um determinado grupo de afinidade, informado pelo usuário.
- Desenvolva um procedimento que localize na base de dados um determinado e-mail (não se esqueça de validar o endereço). Imprima na tela todos os dados do contato, ou informe se o contato não existe.

Problema #21

Faça um programa para gerenciar o resultado da Maratona Internacional de São Paulo. Cada maratonista possui as seguintes informações: NOME, NÚMERO de inscrição, CATEGORIA (amador masculino, amador feminino, profissional masculino e profissional feminino) e TEMPO de chegada (que deve ser modelado como uma *struct* TEMPO, com os campos HORAS, MINUTOS e SEGUNDOS.

Seu programa deve permitir:

- Cadastro das informações de maratonistas (um cadastro por vez, e limite máximo de 50).
- Resultado final da maratona, apresentando o ranking em ordem de chegada das 04 categorias.

Exemplo de Saída:

```
Resultado: Profissional Masculino
ONNONN ONON NONO        65        [1h 25m 52s]
NONNON NONO ONON        57        [1h 32m 14s]
(...)
Resultado: Profissional Feminino
ANNAN NNAANA NANAN      14        [1h 54m 41s]
(...)
```

Problema #22

Faça um programa para gerenciar as informações dos associados de uma cooperativa de crédito. Associado possui Nome, CPF, Profissão e Renda Mensal.

- O programa deve realizar um cadastro de associado por vez.
- O programa somente aceitará o cadastro do associado caso o CPF informado seja válido (crie a função **validaCPF** [pergunte ao *teacher* como fazer]).
- O programa não deve aceitar o cadastro de CPFs repetidos (crie uma função para verificar a ocorrência de repetições).
- O programa deve oferecer opções de relatório, em formato de tabela, no *menu* principal:
    - Listagem de todos os associados;
    - Listagem dos associados através da busca pelo nome. (Obs. permitir busca parcial de um nome -> crie uma função facilitar essa busca).
    - Listagem dos associados com renda acima do valor X (informado pelo usuário);
    - Listagem do associado com maior renda;

Atenção: Perceba que todos os relatórios possuem uma **atribuição em comum, que é a impressão dos dados de associado em formato de tabela**. Portanto, faça o isolamento desta sub-rotina em um procedimento próprio, evitando a duplicação do mesmo código em vários lugares.

Problema #23

Um provedor de acesso à Internet possui o seguinte cadastro de clientes: nome, número de horas de acesso, possui endereço IP público (S-sim ou N-não). Elabore um programa que:

- Cadastre clientes (um por vez);
- Imprime o extrato de faturamento da empresa. Para isto, considere que as primeiras 100 horas de acesso de cada cliente têm custo de R$ 80,00; Horas excedentes têm o custo de R$ 1,85 / hora e os clientes que possuem endereço IP público têm acréscimo de R$ 50,00. O cliente que possui a maior quantidade de horas de acesso tem direito a desconto de 20% sobre o total bruto (Dica: faça uma função que encontre o cliente com a maior quantidade de horas de acesso).
  Imprima o extrato em formato de tabela (NOME - HORAS DE ACESSO - IP PÚBLICO? - VALOR BRUTO - VALOR DESCONTO - VALOR FINAL). Ao final do extrato, imprima também o valor do faturamento total da empresa provedora de Internet.

Problema #24

Um baralho convencional possui 52 cartas únicas. Cada carta possui um valor entre 1 e 13, e um naipe dentro dos 04 possíveis (Copa, Espada, Paus e Ouro).

Faça um programa que simule o seguinte jogo...

N amigos (sendo N >= 2 e N <=5) recebem aleatoriamente 04 cartas cada.

O vencedor do jogo é o primeiro amigo que conseguir obter um dos seguintes conjuntos de cartas:

- 02 pares de cartas com valores idênticos, por exemplo... 3, 4, 3, 4 ou 10, 10, 13, 13.
- 03 cartas em sequência numérica de valores, de naipes distintos. P.Ex.. 8->9->10 ou 2->3->4.
- 04 cartas com naipes distintos (independentemente dos seus valores).

A cada rodada em que não houver um vencedor, cada jogador poderá trocar uma carta da sua mão por outra que ainda não foi sorteada.

O seu programa deve exibir, em cada rodada, as cartas que estão em poder de cada amigo!

Dica.: Faça uma função que recebe um número entre 1 e 4; e imprima o símbolo do Naipe correspondente, sabendo que os valores que representam os símbolos são: “\u2660”, “\u2663”, “\u2665” e “\u2666”.

Problema #25

Desenvolva um programa que auxilie no controle de uma fazenda de bovinos. As informações para registro e controle dos animais são:

- Código: código da cabeça de gado.
- Leite: número de litros de leite produzido por semana.
- Ração: quantidade de alimento ingerida por semana - em quilos.
- Peso: peso do animal, em arrobas.
- Nascimento: mês e ano de nascimento do animal.
- Abate: "N" (não) ou "S" (sim). Obs.: Este campo **NÃO** deverá ser preenchido pelo usuário, mas calculado pelo próprio sistema.

Pede-se os seguintes requisitos:

- Cadastro de animais (um cadastro por vez).
- Alteração dos dados de animais (busca através do “Código”).
- Relatório de animais para abate, sendo que, um animal é enviado para abate quando atinge alguma das seguintes condições...
    - Possui mais de 05 anos de idade.
    - Produza menos de 40 litros de leite por semana.
    - Produza menos de 70 litros de leite por semana e ingira mais de 50 quilos de ração por dia.
    - Possui peso maior que 18 arrobas.
- Um animal que já foi abatido, obviamente, não poderá ser abatido novamente.
- Relatório da quantidade total de leite produzido por semana.
- Relatório do peso total de animais abatidos.
- Relatório da quantidade total de ração necessária por semana.

Deve ser apresentado para o usuário um *menu* contendo todas as opções de funcionalidades do sistema.