L I S T A D E E X E R C Í C I O S

- Structs e Modularização -

Data Limite para Resolução: Data da Prova #02.
Envio das soluções em arquivo compactado (zip/rar) para o e-mail: adriano.exe@gmail.com

Exercícios de Structs

Problema #01

Faça um programa em C que cria uma *struct* **Horário**, contendo informações de Horas, Minutos e Segundos. Utilizando a constante __TIME__ faça a alimentação de uma variável do tipo Horário.
Imprima na tela a **hora** atual com base na *struct* preenchida.

Problema #02

Continue o desenvolvimento do Problema #01, agora fazendo com que o usuário também alimente um horário. Após isto, o programa deve preencher uma terceira variável do tipo Horário contendo a diferença entre a hora atual e a hora cadastrada pelo usuário. Imprima na tela o resultado do cálculo.

Problema #03

Faça um programa que preencha o cadastro de N pessoas. "Pessoa" possui os atributos "Nome" (string) e "Data de Nascimento" (Dia, Mês e Ano). O usuário deve preencher apenas o campo Nome. A "Data de Nascimento" deve ser sorteada aleatoriamente pelo próprio sistema. Use a solução do Problema #01 para coletar a data do sistema, calcular a idade de cada Pessoa, e imprimir o seguinte relatório, ordenado por idade decrescente.

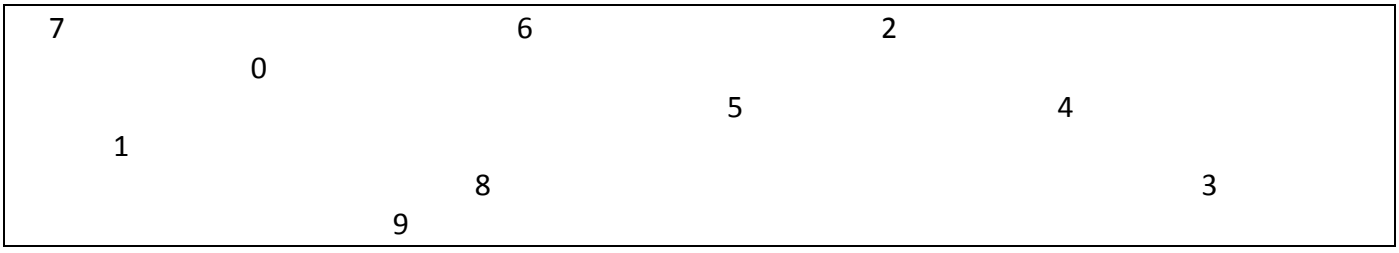
```
Nome              Data de Nascimento       Idade Atual
Xxxxx Xxxx        04/03/1986               32
Yyyyyy Yyy        01/02/1988               30
```

Problema #04

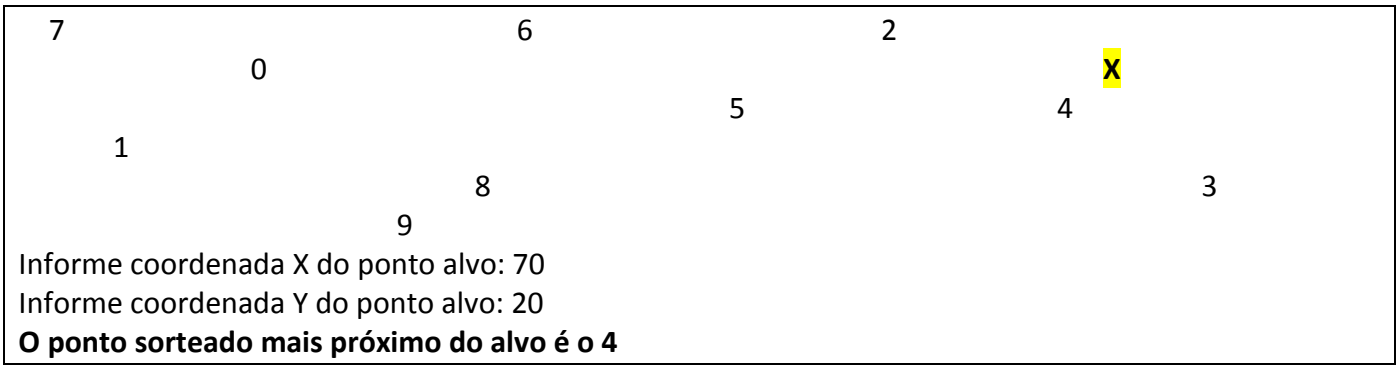
Faça um programa que realize simulações de empréstimo financeiro. Modele a *struct* Proposta: Valor Emprestado (f), Prazo (i) e Taxa de juros (f). Calcule o valor de cada parcela e o valor total pago ao final do empréstimo. Faça novas simulações até que o usuário informe Valor de Empréstimo negativo.
Obs.: Utilize a fórmula de "juros compostos" para realizar os cálculos.

Problema #05

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.: Necessário biblioteca "gconio.h" e função gotoxy().



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.



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

Problema #06

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 #07

Uma pequena mercearia deseja informatizar o seu processo de estoque e vendas. Faça um programa que alimente um vetor com “N” registros de Produto: Código de Barras (i), Descrição (s), Qtde. em Estoque (i), Valor Unitário (f).
Após cadastrar o estoque, o programa entrará em modo de venda, neste modo, o programa segue o seguinte fluxo:

- Usuário informa o Código de Barra do produto a ser vendido;
- Programa imprime as informações do produto (ou se ele não existe);
- Usuário informa a Quantidade deste item que será vendido;
- Programa verifica se possui em estoque a quantidade desejada;
- Caso seja possível a venda, o programa contabiliza o valor a ser pago e atualiza a quantidade no estoque;
- Programa repete vendas, até que Código de Barras informado seja um valor negativo;
- Caso código negativo, o programa informa o valor total da venda;

Problema #08

Uma biblioteca deseja informatizar o arquivamento e a consulta do acervo de livros que possui. Faça um programa que utiliza 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 as informações básicas dos “N” 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: Analise 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 todos os livros da biblioteca, o **programa** deverá distribuí-los em prateleiras, seguindo a seguinte lógica: 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: Analise 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 #09

Modele uma *struct* chamada Velocidade, contendo as seguintes informações: quilômetros por hora (kmH), metros por segundo (ms), milhas por hora (mph) e nós (velocidade náutica).
Peça para o usuário cadastrar uma das medidas de velocidade. O programa deve converter e preencher todas as demais medidas. Cada conversão deve ser feita através de uma função específica. Imprima os dados da *struct* com todas as velocidades convertidas.

Problema #10

Faça um programa em C que implemente as seguintes estruturas de dados:

- Horário: composto de horas, minutos e segundos.
- Data: composto de dia, mês e ano.
- Compromisso: composto de Data, Horário e Texto que descreve o compromisso.

Implemente as seguintes funcionalidades:

- Função para cadastrar compromisso.
- Função que retorna variável do tipo Horário, com a hora atual (utilize a constate __TIME__)
- Função que retorna variável do tipo Data, com a data atual (utilize a constate __DATE__)
- Função que compare se duas datas são iguais
- Função que retorna a diferença entre dois horários
- Função para consultar compromissos da data atual do sistema, informando os compromissos e quantas horas, minutos e segundos ainda faltam para cada compromisso (baseado no horário do sistema).

Problema #11

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 #12

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 #13

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. Familia, Trabalho, Amigo, etc...). O seu programa deverá cumprir os seguintes requisitos:

- Realizar o cadastro de um contato por vez, através de um procedimento/função própria para este fim.
- 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 #13

Desenvolva um Sistema de Gestão Acadêmico básico. O Sistema deverá gerenciar:

Disciplinas

- Código (I)
- Nome (S)
- Carga Horária Semanal (I)
- Qtde de Vagas (I))

Alunos

- Nome (S)
- Matriculas (Vetor de Inteiros que armazena Cód. de Disciplinas em que está matriculado).

No *menu* principal do programa, deverão ser apresentadas as seguintes opções:

1 - Cadastrar Disciplina (1 por vez)
2 - Cadastrar Aluno (1 por vez)
3 - Matricular Aluno
- Informar nome da disciplina e o sistema deverá buscar o código correspondente.
- A quantidade de matricula na disciplina não pode exceder a sua quantidade de vagas;
- Nenhum aluno poderá se matricular em disciplinas que superem o total de 40 h/a.

4 - Imprimir as Disciplinas e Alunos Matriculados, conforme o modelo abaixo.

Disciplina: Estruturas de Dados	**6 h/a**	**Vagas: 03/10**
Aluno	**Carga Horária**	
AAAAAAAAAAAAAAAA	10 h/a	
BBBBBBBBBBBBBBBB	06 h/a	
CCCCCCCCCCCCCCCC	14 h/a	
Disciplina: Redes de Computadores II	**4 h/a**	**Vagas: 02/10**
Aluno	**Carga Horária**	
AAAAAAAAAAAAAAAA	10 h/a	
DDDDDDDDDDDDDDDD	04 h/a	

Problema #15

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 #16

CRM (*Customer Relationship Management*) é um sistema de informação voltado para a Gestão de Relacionamento com Clientes. Uma empresa quer desenvolver um *software* para enviar mensagens de felicitações para os seus clientes aniversariantes. Desta forma, solicitou que você desenvolva um *software* para armazenar dados como: nome, telefone, dia e mês de aniversário e ano de nascimento.

- Crie um procedimento para o cadastro dos dados da pessoa.
- Crie uma função para validar o dia e o mês informado pelo usuário (por exemplo, não existe dia 31/04, mas existe o dia 31/05. Note que existe um "padrão" para essa distribuição).
- Crie um procedimento **aniversarioMes** que imprime a relação dos aniversariantes de um determinado mês enviado por parâmetro. Essa listagem deverá estar ordenada pelo dia de aniversário, e deve informar, em formato de uma tabela, os seguintes dados: dia de aniversário, nome do cliente, idade que completará.
- Crie uma opção no menu para imprimir um relatório com dados de todos cadastros, ordenados pelo dia e mês. (Dica: Faça uso da mesma função **aniversarioMes** para resolver esse requisito).

Problema #17

Um provedor de acesso à Internet mantém o seguinte cadastro de clientes: nome, número de horas de acesso, possui hospedagem (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 hospedagem têm acréscimo de R$ 20,00. 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 qtde. de horas de acesso). Imprima o extrato em formato de tabela (NOME - HORAS DE ACESSO - HOSPEDAGEM? - VALOR BRUTO - VALOR DESCONTO - VALOR FINAL). Ao final do extrato, imprima também o valor do faturamento total da empresa provedora de Internet.

Problema #18

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 seqüência 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 #19

Faça um programa para gerenciar as informações dos associados de uma cooperativa de crédito. Associado possui Nome, CPF, Profissao 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 #20

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.
- Racao: 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 que 50 quilos de ração por dia.
    - Possui peio maior que 18 arrobas.
- Um animal que já foi abatido, obviamente, não poderá ser abatido novamente.
- Relatório da quantidade total de leite produzida 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.