Lista de Exercícios

ICC 1 – Estruturas de dados heterogêneas

Professor Cláudio Fabiano Motta Toledo

- Todos os exercícios devem ser resolvidos utilizando ***structs*, ponteiros e alocação dinâmica**. Não utilize variáveis globais.
- Libere a memória dos ponteiros criados em todos os exercícios no final do programa.
- Funções que criam vetores devem retorná-los pelo `return`.
- Os programas devem ser modularizados: **cada tarefa dos exercícios deve ser feita em uma função separada da** `main()`.

1. (Fácil) Faça um programa que faça operações simples de números complexos:

- Crie e leia dois números complexos *z* e *w*, compostos por *parte real* e *parte imaginária*.
- Apresente a soma, subtração e produto entre *z* e *w*, nessa ordem, bem como o módulo de ambos.

2. (Fácil) Faça um programa que converta coordenadas polares para cartesianas:

- Crie e leia um ponto em coordenada polar, composto por *raio (r)* e *argumento (a)* em radianos.
- Crie outro ponto, agora em coordenada cartesiana, composto por *x* e *y*, sabendo que *(x = r.cos a)* e *(y = r.sin a)*.

Na sua função main(), mostre as coordenadas de ambos os pontos.

3. (Fácil) Faça um programa que faça operações simples de frações:

- Crie e leia duas frações *p* e *q*, compostas por *numerador* e *denominador*.
- Encontre o máximo divisor comum entre o numerador e o denominador, e simplifique as frações.
- Apresente a soma, a subtração, o produto e o quociente entre as frações lidas.

4. (Fácil) Faça um programa que leia um inteiro *n* e:

- Crie e leia um vetor com os dados de n carros: marca (máximo 15 letras), ano e preço.
- Leia um valor p e mostre as informações de todos os carros com preço menor que *p*. Repita este processo até que seja lido um valor *p = 0.*

5. (Fácil) Faça um programa que leia um inteiro *n* e:

- Crie e leia um vetor com dados de *n* livros: título (máximo 30 letras), autor (máximo 15 letras) e ano.
- Procure um livro por título, perguntando ao usuário qual título deseja buscar. Mostre os dados de todos os livros encontrados.

6. (Fácil) Faça um programa que:

- Leia dois inteiros *n* e *m*, *c*rie e leia uma estrutura de dados que é uma matriz de inteiros **positivos**, contendo suas dimensões *n* e *m* bem como seus elementos.
- Leia um inteiro *x* e procure na matriz, mostrando na tela a linha e a coluna em que está. Repita esse processo até ler um número menor que zero.

7. (Fácil) Faça um programa que leia um número *n* e:

- Crie e leia um vetor de alunos, sendo que cada aluno contém os dados: *nome* (máximo 15 letras), *notas* de 3 provas, *média final* e *nível* (inteiro). Este último campo **não** deve ser lido agora.
- Preencha o campo *nível*. Seu valor deve ser igual à parte inteira de *(5*média final/média da sala)*.

Na sua função `main()`, mostre o nome e o nível de cada aluno.

8. (Fácil) Faça um programa que seja uma agenda de compromissos. Leia um inteiro *n* e:

- Crie e leia um vetor de *n* estruturas de dados com: compromisso (máximo 60 letras) e data. A data deve ser outra estrutura de dados contendo dia, mês e ano.
- Leia dois inteiros *m* e *a* e mostre todos os compromissos do mês *m* do ano *a*. Repita o procedimento até ler *m = 0.*

Dica: use `fgets(string, tamanho, stdin)` para ler uma string, precedido imediatamente por `fflush(stdin)`.

9. (Fácil) Faça um programa para calcular a corrente em um circuito elétrico resistivo simples:

- Crie e leia os dados de uma fonte de tensão real composta por: *força eletromotriz (E)* e *resistência interna (ri).*
- Crie e leia os dados de um receptor composto por: *resistência interna (ri)* e *consumo (E')*
- Calcule e mostre a corrente que passa no circuito composto pela fonte e pelo receptor lidos, sabendo que $E = E' + R.i$, onde R é a soma das resistências internas.

10. (Fácil) Faça um programa que controla o consumo de energia dos eletrodomésticos de uma casa. Leia um inteiro *n* e:

- Crie e leia *n* eletrodomésticos que contém *nome* (máximo 15 letras), *potencia* (real, em kW) e *tempo ativo* por dia (real, em horas).
- Leia um tempo *t* (em dias), calcule e mostre o consumo total na casa e o consumo relativo de cada eletrodoméstico (*consumo/consumo total)* nesse período de tempo. Apresente este último dado em porcentagem.

11. (Fácil) Faça um programa que gerencie o estoque de um mercado. Leia um inteiro *n* e:

- Crie e leia um vetor de *n* produtos, com os dados: código (inteiro), nome (máximo 15 letras), preço e quantidade.
- Leia um pedido, composto por um código de produto e a quantidade. Localize este código no vetor e, se houver quantidade suficiente para atender ao pedido **integralmente**, atualize o estoque e informe o usuário. Repita este processo até ler um código igual a zero.

Se por algum motivo não for possível atender ao pedido, mostre uma mensagem informando qual erro ocorreu.

12. (Médio) Faça um programa que calcule distância entre pontos no espaço. Leia um inteiro *n* e:

- Crie e leia um vetor de *n* pontos, contendo as coordenadas *x*, *y* e *z* (reais).
- Crie e construa uma matriz *D* de distâncias, sendo o elemento d_{ab} a distância entre os pontos n_a e n_b. A distância entre os pontos é dada por $\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}$

Na sua função `main()`, mostre a matriz *D* no formato `"%.2f"`.

13. (Médio) Faça um programa que controle contas de banco. Leia um inteiro *n* e:

- Crie e leia um vetor de contas de banco, com *código* (inteiro), *cliente* (máximo 15 letras), *saldo*.
- Leia um inteiro. Se for lido 1, execute **depósito**. Se for lido 2, execute **saque**. Se for lido 0, finalize o programa. Repita este processo enquanto não for lido um valor válido.
- **Depósito:** leia um código de conta e um valor. Some o valor lido no saldo da conta lida. Mostre o nome do cliente e o saldo resultante na tela.
- **Saque:** leia um código de conta e um valor. Se o saldo for suficiente, deduza o valor lido no saldo da conta lida. Mostre o nome do cliente e o saldo resultante na tela.

14. (Médio) Faça um programa que:

- Leia um inteiro *n* ($2 \leq n \leq 3$), crie e leia uma estrutura que é uma matriz quadrada de ordem *n*, contendo sua dimensão e seus elementos.
- Crie a matriz adjunta da matriz lida. A matriz adjunta é composta pelos co-fatores da matriz geradora. O co-fator do elemento A_{ij} é o determinante da matriz que se obtém eliminando-se as linhas *i* e *j* da matriz original.

Exemplo de cálculo do co-fator do elemento A_{12}:

```
      3 5 7              2 5          2.3 - 5.2 = -4
      2 4 5              2 3
      2 1 3
  Matriz original   Matriz restante    Co-fator de A12
```

15. (Médio) Faça um programa que contenha uma pilha estática. Uma pilha é uma estrutura de dados que segue a política FILO (*First in, last out)*, ou seja, só é possível retirar elementos do topo da pilha (os últimos inseridos).

- Faça uma estrutura chamada *pilha* que contém a sua dimensão máxima *n*, um vetor com seus elementos e seu *topo* (a posição do último elemento). Use-a para armazenar inteiros.
- Inicialize a pilha lendo o seu tamanho, alocando seu vetor e inicializando o topo como -1. Note que (*topo == -1)* significa *pilha vazia*.

Em seguida, leia números digitados pelo usuário. Cada número será uma operação sobre sua pilha:

- **Inserir:** caso o usuário digite 1, leia um valor *x* e insira-o no topo da pilha. Atualize o topo para *(topo + 1)*. Se não houver mais espaço, informe ao usuário e suspenda a operação.
- **Remover:** caso o usuário digite 2, remova o valor do topo da pilha e retorne-o ao usuário. Atualize o topo para *(topo – 1)*. Se a pilha estiver vazia, informe ao usuário e retorne -1.
- **Sair:** caso o usuário digite 0, saia do programa.

Na função `main()`, sempre que o usuário remover um elemento da pilha, imprima-o.

16. (Médio) Faça um programa que controle o fluxo de vôos nos aeroportos de um país. Leia dois inteiros *v* (vôos) e *a* (aeroportos) e:

- Crie e leia um vetor de vôos, sendo que cada vôo contém um código de aeroporto de origem e um de destino.
- Crie um vetor de aeroportos, sendo que cada aeroporto contém seu código, quantidade de vôos que saem e quantidade de vôos que chegam.

Nota: Cada aeroporto é identificado por um código inteiro entre 0 e *(a-1)*. Não aceite aeroportos de código inexistente.

17. (Médio) Uma empresa de correios tem uma entrega distante para fazer, composta por vários caminhos em vários meios de transporte. Faça um programa que leia dois inteiros *n* e *m* e:

- Crie e leia um vetor de *n* meios de transporte, contendo *nome* (máximo 15 letras), *velocidade média* (km/h) e *consumo médio* (L/km).
- Crie e leia um vetor de *m* caminhos, contendo o meio de transporte usado (posição dele no vetor criado anteriormente) e seu *comprimento*.
- Calcule e mostre o tempo que se leva e o consumo de combustível para percorrer cada caminho com o meio de transporte dado, bem como o trajeto todo. Exemplo:

```
2 2
** Transporte **
Carro 100 10
Moto 150 25
** Caminhos **
0 300
1 600
Caminho 0 (Carro): 3h, 30L
Caminho 1 (Moto): 4h, 24L
Total: 7h, 54L
```

18. (Médio) Faça um programa que simule um dicionário estático. Dicionário é uma estrutura de dados que mapeia uma chave única a algum outro dado. Neste exercício, você deve mapear uma string a um inteiro.

- Leia um número *n* para ser o tamanho máximo de mapas. Então leia um vetor de *n* mapas contendo a chave (string de até 10 letras) e seu valor (que será inteiro).

A seguir, leia números digitados pelo usuário. Cada número será uma operação.

- **Consulta:** caso digite 1, leia uma chave do usuário, procure por seu valor no dicionário e mostre-o. Se a chave não for encontrada, informe ao usuário.
- **Redefinição:** caso digite 2, leia uma chave e um valor do usuário e redefina o valor desta chave. Se a chave não existir, informe ao usuário.
- Caso digite zero, saia do programa.

19. (Médio) Faça um programa para armazenar um livro de receitas. Leia um inteiro *n* e:

- Crie um vetor de *n* receitas, que deve ter *nome* (máximo 25 letras), *quantidade de ingredientes* e *ingredientes*.
- Para cada receita, leia seu *nome* e a *quantidade de ingredientes*. Então crie e leia o vetor de *ingredientes*, sendo que cada ingrediente contém *nome* e *quantidade*.
- Procure receita por nome, mostrando seus ingredientes se encontrar. Se não encontrar, informe ao usuário. Repita o processo até digitar uma string vazia.

20. (Médio) Faça um programa que armazena filmes produzidos por vários diretores. Leia um inteiro *n* e:

- Crie e leia um vetor de *n* diretores, cada um contendo *nome* (máximo 20 letras), *quantidade de filmes* e *filmes*. O membro *filmes* é um vetor, que deve ser criado após ter lido *quantidade de filmes*. Cada filme é composto por *nome*, *ano* e *duração*.
- Procure um diretor por nome, mostrando todos os filmes que ele já produziu. Repita o processo até digitar uma string vazia.

21. (Médio) O coiote precisa descobrir a velocidade média do papa-léguas para seu novo plano. Para isso ele usará os pontos de referência por onde o pássaro passa. Ele estava parado comendo um prato de alpiste deixado pelo audaz canino do deserto. Quando saiu em disparada (como sempre), um cronômetro foi disparado. Leia um inteiro *n* de referências e:

- Leia um vetor de *n* referências, compostas por sua posição (x, y) no deserto e o momento que o papa-léguas o atingiu desde que o coiote disparou o cronômetro.
- Calcule a velocidade média desenvolvida pelo papa-léguas entre cada ponto de referência, bem como no percurso todo. Mostre esses dados na tela.

22. (Difícil) Faça um programa que simule uma rede social. Leia um inteiro *n* de usuários e:

- Leia um vetor de *n* usuários, sendo que cada um tem *nome* (máximo de 15 letras), *quantidade de amigos* e *lista de amigos*. A lista de amigos é um vetor de strings com nomes dos amigos (máximo de 15 letras cada nome).
- Calcule e mostre a popularidade de cada usuário da rede social, sendo que a popularidade é o número de vezes que aparece na lista de amigos de todos os usuários. Exemplo:

Entrada	Saída
** n ** 3 ** usuário 0 (nome/quantidade/amigos) ** Marcelo 3 Hossomi Yukio Hitomi ** usuário 1 (nome/quantidade/amigos) ** Yukio 2 Marcelo Hossomi ** usuário 2 (nome/quantidade/amigos) ** Hossomi 3 Marcelo Yure Hitomi	** Popularidade ** Marcelo 1 Yukio 1 Hossomi 2

Notas: nem todos os amigos na lista de amigos precisam ser necessariamente usuários da rede social. Não é preciso calcular popularidade de não-usuários.

23. (Difícil) Uma fábrica de chocolates precisa acelerar o processo de produção de suas caixas de chocolates. Ela tem *n* tipos de caixas, cada um com chocolates diferentes. Ela também tem *m* tipos de chocolates diferentes.

- Leia um vetor de *m* chocolates, sendo que cada chocolate tem um *nome* (máximo de 20 letras) e um *preço*.
- Leia um vetor de *n* tipos de caixas, cada uma contendo seu *preço* (calculado posteriormente), uma quantidade *q* e um vetor de *q itens*. Cada item tem o *tipo de chocolate* (identificado pela posição dele no vetor criado anteriormente) e sua *quantidade*.
- Calcule o preço de cada tipo de caixa. Mostre na tela os chocolates que cada tipo de caixa tem, bem como suas quantidades, e seu preço total. Exemplo na página seguinte.

Entrada	Saída
** (n/m) ** 2 3 ** (nome/preço) ** Sonho de Valsa 0,75 Ouro Branco 0,60 Lancy 0,50 ** caixa 0 (q) ** 3 ** itens (chocolate/quantidade) ** 0 2 1 3 2 4 ** caixa 1 (q) ** 2 ** itens (chocolate/quantidade) ** 0 5 1 5	** caixa 0 ** 2 Sonho de Valsa 3 Ouro Branco 4 Lancy R$5,30 ** caixa 1 ** 5 Sonho de Valsa 5 Ouro Branco R$6,75

24. (Difícil) Faça um programa que leia uma expressão polinomial homogênea (igual a zero). Leia o número *n* de termos e:

- Crie e leia um vetor de *n* termos, sendo que cada termo possui *coeficiente*, *incógnita* e *expoente*. A expressão pode ter quantas incógnitas o usuário desejar, que deve ser representada por uma letra apenas. Trate os coeficientes e o expoente como inteiros.
- Crie um novo vetor, mas simplificando a expressão somando todos os termos com mesma incógnita e mesma potência.

Na sua função `main()`, mostre a equação original e a equação simplificada.

25. (Difícil) Faça um programa que seja uma fila de inteiros estática. Fila é uma estrutura de dados que segue a política FIFO (*first in, first out*), ou seja, sempre que se insere um novo elemento, ele entra no final. Mas quando se remove um elemento, remove-se o primeiro. Então:

- Faça uma estrutura *fila* que contém seu tamanho *n*, um vetor de inteiros e seu *fim*, que indicará a posição do último elemento da fila.
- Crie e inicialize sua fila, lendo o tamanho *n*, alocando o vetor e inicializando seus valores com zero, bem como o *fim* com -1. Note que *(fim == -1)* significa fila vazia.

Deve-se, então, ler números digitados pelo usuário. Cada número será uma operação da lista.

- **Inserção:** caso digite 1, leia um novo elemento *a* e insira na primeira posição vaga, que será sempre *(fim + 1)*. Atualize o valor de *fim* para *(fim + 1)*. Se não houver espaço, avise ao usuário e suspenda a operação.
- **Remoção:** caso digite 2, remova o primeiro elemento da fila e desloque todos os demais elementos para a esquerda (de modo a não deixar nenhum "buraco" no vetor). Atualize o *fim* para *(fim – 1)* e retorne ao usuário o elemento removido. Se a fila estiver vazia, avise ao usuário e retorne -1.
- Caso digite 0, saia do programa.

Sempre que o usuário remover um elemento da pilha, imprima o elemento retornado na tela.

```
** exemplo de fila **
7
** fila inicializada **
0 0 0 0 0 0 0          fim: -1
** inserir **
1
10
10 0 0 0 0 0 0         fim: 0
1
15
10 15 0 0 0 0 0        fim: 1
1
10
10 15 10 0 0 0 0       fim: 2
** remover **
2
Retornado: 10
15 10 0 0 0 0 0        fim: 1
2
Retornado: 15
10 0 0 0 0 0 0         fim: 0
2
Retornado: 4
0 0 0 0 0 0 0          fim: -1
0
```

Nota: não é preciso imprimir a fila em momento algum, se não quiser. A implementação real da fila estática é um pouco diferente do que foi descrito acima, mas o conceito é o mesmo.