ED1 2025/1: Lista de Exercícios n.º 2
Estrutura de Dados I
Professor(es): Abrantes Araújo Silva Filho Entrega: 2025-03-04 07:15

As questões e exercícios de programação a seguir foram retiradas do livro "*Programming Abstractions in C: A Second Course in Computer Science*", de Eric S. Roberts (Addison-Wesley, 1998).

1 Questões de Revisão

Responda às questões abaixo, de forma **manuscrita**, em papel almaço. Entregue diretamente ao professor na data indicada.

(a) Defina cada um dos seguintes termos: **ponteiro**, **array** e **registro**.

(b) Que definição de tipo você usaria para criar uma nova enumeração chamada "`poligono_t`" consistindo dos seguintes elementos: `triangulo`, `quadrado`, `pentagono`, `hexagono` e `octogono`? Como você poderia alterar a definição para que a representação interna de cada nome constante correspondesse ao número de lados para aquele polígono?

(c) Quais as vantagens de se usar uma enumeração ao invés de usar constantes simbólicas criadas com `#define` para alcançar funcionalidade semelhante?

(d) Explique como a palavra chave "**`typedef`**" funciona.

(e) Defina os seguintes termos: **bit**, **byte**, **word**, **endereço**.

(f) Qual a função do operador **`typeof`**? Quando ele é avaliado? Como é utilizado?

(g) Cite quatro razões para o uso de ponteiros.

(h) O que é um "lvalue"? E um "rvalue"?

(i) Quais os tipos das variáveis `p1` e `p2`, criadas com a seguinte declaração:

```
int * p1, p2;
```

(j) Como você faria a declaração de uma variável chamada "`p_flag`" como um ponteiro para um valor Booleano?

(k) Quais são as duas operações fundamentais dos ponteiros? Que operação corresponde ao termo "desrefereciar"?

(l) Explique a diferença entre a atribuição de ponteiro e a atribuição de valor.

(m) Desenhe diagramas mostrando o conteúdo da memória após **cada linha** do código a seguir:

```
v1 = 10; v2 = 25; p1 = &v1; p2 = &v2;
*p1 += *p2;
p2 = p1;
*p2 = *p1 + *p2;
```

(n) Qual a representação interna da constante `NULL`?

(o) O que significa *call by reference*?

(p) Escreva as declarações dos seguintes arrays:

- Um array chamado `array_real` consistindo de 100 valores de ponto flutuante;
- Um array chamado `em_uso` consistindo de 16 valores Booleanos;
- Um array chamado `linhas` que pode armazenar até 1000 strings.

Lembre-se de que os limites superiores desses arrays devem ser definidos como constantes para facilitar alterações posteriores.

(q) Qual a diferença entre o **tamanho alocado** e o **tamanho efetivo** de um array?

(r) Assumindo que o endereço base para o array `retangular` é, em hexadecimal, `0x1000`, e que valores do tipo `int` ocupam 4 bytes na memória, desenhe um diagrama que mostre o endereço de cada elemento do array declarado como:

```
int retangular[2][3];
```

(s) Assumindo que `int_array` foi declarado como

```
int int_array[10];
```

e que `j` é uma variável inteira, descreva os passos necessários que o computador terá que realizar para determinar o valor da seguintes expressão:

```
&int_array[j + 3];
```

(t) Se `arr` foi declarado como um array, descreva em detalhes a distinção entre as seguintes expressões: "`arr[2]`" e "`arr + 2`".

(u) Assumindo que uma variável do tipo `double` ocupa 8 bytes de memória RAM, e que o endereço base de um array que armazena valores desse tipo de dado e chamado de `double_array` é, em hexadecimal, `0x1000`, qual é o endereço de:

```
double_array + 5;
```

(v) Explique se a seguinte afirmação é verdadeira ou falsa: se `p` é um variável do tipo ponteiro, a expressão `p++` adiciona 1 à representação interna de `p`.

(w) Que passos são necessários para declarar uma variável do tipo registro?

(x) Se a variável `p` é declarada como um ponteiro para um registro que contém um campo chamado `custo`, o que está errado com a expressão

```
*p.custo
```

como um meio de seguirmos o ponteiro `p` para seu valor e, então, selecionar o campo `custo`? Que expressão você escreveria em C para realizar essa operação de desreferenciamento e seleção?

(y) O que é a *Heap*?

(z) Qual é a função do tipo "`void *`"?

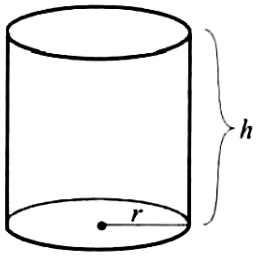
(aa) O que significa *garbage collection*?

2 Exercícios de Programação

Utilize a linguagem **C** para resolver os problemas abaixo. Se necessário, utilize as funções das bibliotecas CRpaic e CSLIB.

(a) Defina uma enumeração chamada de `dia_da_semana_t`, cujos elementos são os dias da semana. Escreva as funções "`dia_seguinte`" e "`dia_anterior`", que recebem um valor do tipo `dia_da_semana_t` e retornam o dia da semana que está após ou antes da data especificada, respectivamente. Por exemplo: "`dia_anterior(Domingo)`" deve retornar `Sábado`. Escreva também a função "`incrementar_dia(dia_inicial, delta)`", que retorna o dia da semana que ocorre `delta` dias a partir do `dia_inicial`. Assim, por exemplo, `incrementar_dia(Quinta, 4)` deve retornar `Segunda`. A implementação de `incrementar_dia` deve funcionar se `delta` for negativo, situação na qual os dias são contados para trás.

(b) Escreva um programa que calcule a área de superfície e o volume de um cilindro com altura h e raio da base r, como no seguinte diagrama:



As fórmulas para o cálculo da área de superfície e do volume são:

$$A = 2\pi rh + 2\pi r^2 \tag{1}$$

$$V = \pi hr^2 \tag{2}$$

O seu programa deve consistir obrigatoriamente de 3 chamadas de funções: uma para ler os dados de entrada, outra para calcular os resultados, e a última para exibir as respostas. Quando apropriado, utilize *call by reference* para comunicar dados entre as funções e o programa `main`.

(c) Em um concurso ou competição cuja nota é dada por juízes, não se pode descartar a possibilidade de que alguns juízes tenham algum viés individual e forneçam notas muito mais baixas ou muito mais altas do que a maioria dos outros juízes. Por isso é uma prática comum descartar a maior e a menor nota dada pelos juízes antes de se calcular a pontuação final dada pela média das notas. Modifique o programa "`juizes.c`" para que a maior e a menor notas sejam descartadas antes do cálculo da média.

(d) Escreva um predicado chamado de "`esta_ordenado(array, n)`" que recebe um array de inteiros e seu tamanho efetivo e retorna TRUE se o array estiver ordenado em ordem não decrescente.

(e) No século III a.C. o astrônomo grego Eratóstenes desenvolveu um algoritmo para encontrar todos os números primos até algum limite superior N. Para aplicar o algoritmo, você começa escrevendo todos os números inteiros entre 2 e N. Por exemplo: se $N = 20$, você começa o algoritmo escrevendo todos os números entre 2 e 20:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Agora você circula o primeiro número da lista, indicando que você encontrou o primeiro número primo. Depois disso você percorre todos os números da lista e faz uma cruz ou um "X" em todos os múltiplos do número que você circulou, indicando que nenhum deles é primo. Assim, depois de circular o número 2 e eliminar todos os múltiplos de 2, você terá a seguinte situação:

(2) 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~

A partir desse ponto você simplesmente repete esse processo, circulando o primeiro número da lista que não está em um círculo e que não esteja cruzado, e depois cruzando seus múltiplos. Neste exemplo o próximo número que você circulará é o 3 e os números que você cruzará são os múltiplos de 3, resultando em:

(2) (3) ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

Eventualmente, todo número da lista estará circulado (indicando que é primo) ou estará cruzado (indicando que não é primo, ou seja, é composto):

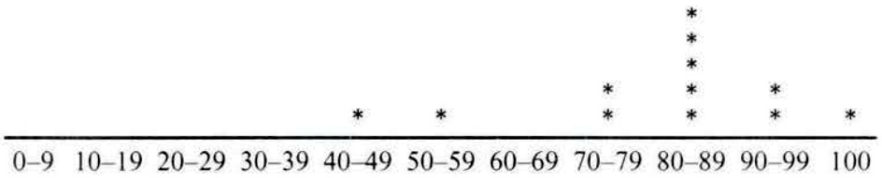
(2) (3) ~~4~~ (5) ~~6~~ (7) ~~8~~ ~~9~~ ~~10~~ (11) ~~12~~ (13) ~~14~~ ~~15~~ ~~16~~ (17) ~~18~~ (19) ~~20~~

Esse algoritmo é conhecido por **Crivo de Eratóstenes**. Escreva um programa que implemente o Crivo de Eratóstenes para imprimir uma lista de todos os números primos entre 2 e 1000.

(f) Um **histograma** é uma maneira visual de apresentar uma coleção de dados, dividindo-se os dados em faixas numéricas separadas e então indicando quantos dados estão em cada faixa. Por exemplo, dado o seguinte conjunto de notas em uma prova

100 95 47 88 86 92 75 89 81 70 55 80

um histograma tradicional teria o seguinte formato:



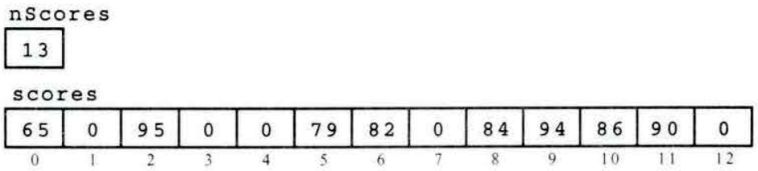
Os asteriscos no histograma acima indicam que uma nota está na faixa entre 40–49, uma nota na faixa entre 50–59, cinco notas estão entre 80–89 e assim por diante.

Quando geramos histogramas em um computador, entretanto, é muito mais fácil exibi-los de forma horizontal na página, como por exemplo:

```
  0:
 10:
 20:
 30:
 40: *
 50: *
 60:
 70: **
 80: *****
 90: **
100: *
```

Sua tarefa é escrever um programa que leia um array de inteiros informados pelo usuário, com as notas de uma prova que podem variar de 0 a 100, e mostrar um histograma desses números divididos nas faixas 0–9, 10–19, 20–29 e assim por diante, até a faixa que termina contém apenas a nota 100. O usuário pode entrar as notas em qualquer ordem e pode informar qualquer quantidade de notas.

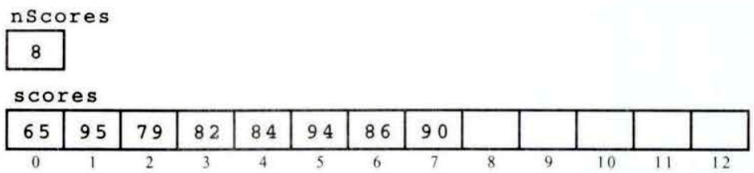
(g) Escreva a função "`remover_elementos_zero(array, n)`", que percorre um array de inteiro e elimina todos os elementos cuja valor seja 0 (zero). Como essa operação altera o tamanho efetivo do array, a função deve retornar o novo tamanho efetivo como resultado. Por exemplo: suponha que `scores` é um array que contém as notas de um exame opcional, e que `nScores` indica o tamanho efetivo do array, como na figura abaixo:



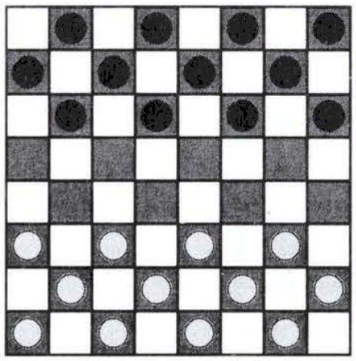
Em algum ponto, se o seu programa chamar

```
nScores = remover_elementos_zero(scores, nScores);
```

o resultado será a remoção de todos os elementos cujo valor seja 0, resultando em:



(h) O estado inicial de um Jogo de Damas está ilustrado na figura abaixo:



Os quadrados pretos do tabuleiro nas três linhas inferiores são ocupados pelas peças vermelhas; os quadrados pretos do tabuleiro nas três linhas superiores são ocupados pelas peças pretas. As duas linhas centrais se iniciam sem nenhuma peça.

Se você quiser armazenar o estado de um tabuleiro de dadas em um programa de computador, você precisa de um array bidimensional indexado por linhas e colunas. Os elementos do array poderiam ser de diferentes tipos, mas uma abordagem razoável — como no "Jogo da Velha" — é usar caracteres. Por exemplo, você poderia usar a letra "`b`" para representar as peças brancas, e a letra "`r`" (*red*) para representar as peças vermelhas. Casas vazias do tabuleiro seriam representadas por espaços ou hifens, dependendo se a cor da casa é branca ou preta, respectivamente.

Crie um subprograma chamado de "`inicializar_tabuleiro`", que inicializa um array que corresponde à posição inicial das peças de um Jogo de Damas. Crie um segundo subprograma chamado de "`mostrar_tabuleiro`" que mostre o estado atual do tabuleiro do jogo, como na ilustração abaixo (que está exibindo o estado inicial do jogo):

```
  b   b   b   b
b   b   b   b
  b   b   b   b
-   -   -   -
  -   -   -   -
r   r   r   r
  r   r   r   r
r   r   r   r
```

(i) Crie um programa que receberá do usuário uma quantidade variável de números de ponto flutuante não negativos que serão armazenados em um array. O usuário encerrará a entrada de dados quando ele digitar o valor sentinela -1. Crie também um subprograma que pesquisará no array o menor e o maior número, e retornará **ambos os valores simultaneamente**. Por exemplo, se o usuário digitar

67 78 75 70 71 80 69 86 65 54 76 78 70 68 77 -1

seu programa deve imprimir a seguinte mensagem:

```
A faixa de valores é 54-86.
```

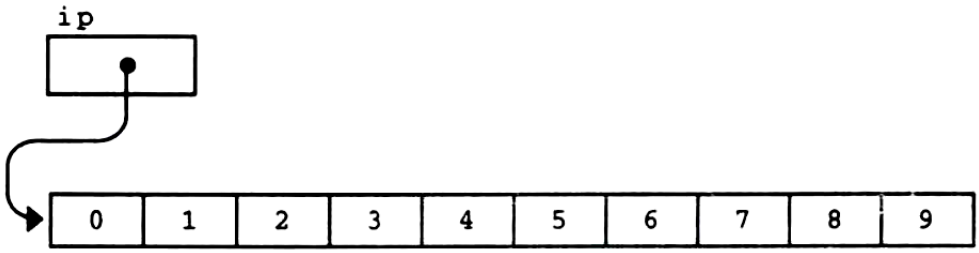
(j) Escreva uma função chamada de "`array_indice(n)`" que retorna um **ponteiro** para um array de inteiros alocado dinamicamente, com n elementos, cada um dos quais inicializado para o valor de seu próprio índice. Por exemplo, assumindo que `ip` é declarado como

```
int *ip;
```

a sentença

```
ip = array_indice(10);
```

deve produzir a seguinte configuração de memória:



(k) Considere que você criou um registro (implementado como um `struct`) que armazena os dados de um funcionário de uma empresa (o tipo `string` está definido na biblioteca CRpaic) e criou um novo tipo que é um ponteiro para esse registro:

```
struct st_empregado
{
    string nome;
    string funcao;
    string cpf;
    double salario;
    int matricula;
};

typedef struct st_empregado *empregado_t;
```

Isso cria para o programa o novo tipo de dado "`empregado_t`", que é do tipo **ponteiro para struct st_empregado**. Agora que você já tem esse novo tipo de dado, você pode declarar variáveis desse tipo como qualquer outra variável, por exemplo, se você fizer

```
empregado_t joao;
empregado_t maria;
```

estará definindo as variáveis "`joao`" e "`maria`", ambas do tipo `empregado_t` (lembre-se de que esse tipo é, na verdade, um **ponteiro para struct st_empregado** que contém os dados de cada funcionário). Como a definição do tipo `empregado_t` inclui o ponteiro, as variáveis "`joao`" e "`maria`" serão automaticamente declaradas como ponteiros também, mesmo que não tenham asterisco ao lado do nome.

Agora note o seguinte: "`joao`" e "`maria`" são apenas ponteiros para registros, mas **os registros ainda não existem**. Para que você possa criar os registros você precisa **alocar memória** para cada um, da seguinte maneira:

```
joao = malloc(sizeof(struct st_empregado));
maria = malloc(sizeof(struct st_empregado));
```

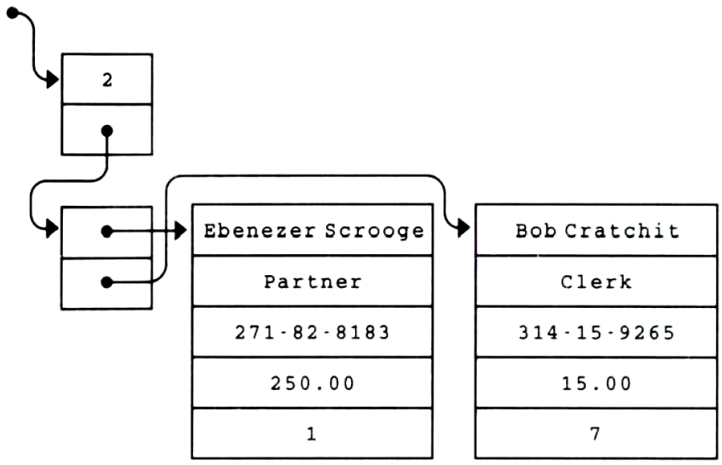
As sentenças acima alocam o espaço necessário de memória para armazenar os registros na *Heap* e retornam um ponteiro para essas áreas de memória. Lembre-se de que, ao trabalhar com alocação de memória, você sempre tem que testar se a alocação foi bem-sucedida antes de começar a utilizar os ponteiros. Lembre-se de que, ao final do programa, você deve desalocar a memória alocada para esses registros.

Lembre-se também de que o tipo "`string`" é, na verdade, apenas um apelido para o tipo "`char *`", ou seja, um ponteiro para "`char`". Isso significa que você precisará **alocar e desalocar memória** para criar e remover as strings (exceto se você usar a biblioteca CRpaic e a função "`get_string`", que já faz a alocação e a desalocação automaticamente para você). Esclareça suas dúvidas quanto a isso com o professor ou com o monitor.

Sua tarefa é criar um novo tipo de dados, chamado de "`folha_pag_t`", para representar a folha de pagamentos dos funcionários dessa empresa, uma lista de funcionários sendo que cada funcionário será do tipo "`empregado_t`".

O tipo "`folha_pag_t`" **deve ser um ponteiro** que aponta para um registro contendo o número de funcionários e um array dinâmico dos valores atuais dos empregados cadastrados, ou

seja, dos "`empregado_t`" como ilustrado no seguinte diagrama:



Após preparar todos os tipos de dados necessários, escreva um subprograma apropriado chamado de "`obter_folha_pag`" que recebe do usuário o número de funcionários que serão cadastrados, e os dados de cada funcionário, como por exemplo:

```
Quantos funcionários serão cadastrados? 2
Empregado 1:
  Nome: Ebenezer Scrooge
  Função: Partner
  CPF: 271-82-8183
  Salário: 250.00
  Matrícula: 1
Empregado 2:
  Nome: Bob Cratchit
  Função: Clerk
  CPF: 314-15-9265
  Salário: 15.00
  Matrícula: 7
```

Depois que os dados forem digitados pelo usuário o subprograma "`obter_folha_pag`" deve retornar um valor do tipo "`folha_pag_t`" cuja estrutura seja a mesma do diagrama acima.

(l) Escreva a função

```
int *obter_array_dinamico(int sentinela, int *pN);
```

que retorna um array dinamicamente alocado de inteiros que serão informados pelo usuário. O "`sentinela`" indicará o valor que será utilizado para terminar a entrada. O segundo argumento "`nN`" é um ponteiro para um número inteiro que será utilizado para retornar o tamanho efetivo do array. Note que é impossível saber previamente quantos números o usuário informará e, por isso, sua implementação deve alocar novos espaços para o array a medida em que isso for necessário.

(m) Suponha que você recebeu a tarefa de criar um sistema para registrar o catálogo de cartão de livros de uma biblioteca. Como primeiro tarefa seu supervisor pediu para que você desenvolvesse um protótipo capaz de armazenar as seguintes informações de cada um dos 1000 livros da biblioteca:

- Título
- Uma lista de até 5 autores
- O registro ISBN
- Uma lista de até 5 palavras-chave dos assuntos que o livro trata
- A editora
- O ano de publicação
- Se o livro pode sem emprestado ou não

Projete as estruturas de dados que seriam necessárias para manter todas essas informações. Em seu programa deve ser possível escrever a declaração

```
biblioteca_t livros;
```

e fazer com que a variável "`livros`" contenha todas as informações necessárias para manter os dados dos 1000 livros. Lembre-se de que o número real de livros pode ser menor do que esse limite superior. Essa variável será o "banco de dados" de livros.

Sua próxima tarefa é a seguinte: escreva o procedimento "`busca_por_chave`", que recebe como argumentos o banco de dados de livros e uma string que corresponde a uma palavra-chave. Esse procedimento deve fazer uma busca nos livros procurando pela palavra-chave. Para todo livro que tiver a palavra-chave seu programa deve imprimir o título, o nome do primeiro autor e o código ISBN do livro.