



Caderno de Exercícios

Estrutura de Dados I

1ª Edição

Professor: Eder Stone Fontoura

Conteúdo

Exercícios de Ponteiros	3
Listas Encadeadas	13
Filas e Pilhas	14

Exercícios de Ponteiros

1. Escreva um programa em C que declara e inicializa uma variável double, uma variável int e uma variável char. Logo após, declarar e inicializar um ponteiro para cada uma das variáveis declaradas anteriormente. O programa deverá exibir o endereço, o valor armazenado e o tamanho (número de bytes) ocupados para cada uma das 6 variáveis criadas.
2. Responda as questões contidas no código abaixo.

```
#include<stdio.h>
#include<stdlib.h>

void teste(int x){
    int y = x;
    /**
     * Imprimir o endereço e o valor de x e y aqui
     */
    x = 2;
    /**
     * Imprimir o endereço e o valor de x e y aqui
     */
    /*
     * O valor de x, declarado nesta função, foi modificado? Justifique.
     * O valor de y, declarado nesta função, foi modificado? Justifique.
     */
}

int main(){
    int x = 10;
    /**
     * Imprimir o endereço e o valor de x aqui
     */
    teste(x);
    /**
     * Imprimir o endereço e o valor de x aqui
     */
    /*
     * O valor de x, declarado nesta função, foi modificado? Justifique.
     */
}
```

3. Qual instrução deve ser adicionada ao programa abaixo para que ele funcione corretamente, considerando que ptr_x deve ser um ponteiro para x?

```
int main ( ) {
    int x, *ptr_x;
    *ptr_x = 3;
}
```

4. A função **faz_algo** foi chamada na função main da seguinte forma: `faz_algo(&i,&j)`, onde os valores de `i` e `j` no momento da chamada eram, respectivamente, 5 e 12. A função possui um erro. Qual é este erro?

```
void faz_algo (int *p, int *q){  
    int *temp;  
    *temp = *p;  
    *p = *q;  
    *q = *temp;  
}
```

5. Faça um desenho representando o que esta ocorrendo na memória, com a aplicação dos comandos abaixo:

```
int i = 99 , j;  
int *p;  
p = &i;  
j = *p + 100;
```

6. Faça um desenho representando o que esta ocorrendo na memória, com a aplicação dos comandos abaixo:

```
int a = 5, b = 12;  
int *p;  
int *q;  
p = &a;  
q = &b;  
int c = *p + *q;
```

7. Faça um desenho representando o que esta ocorrendo na memória, com a aplicação dos comandos abaixo:

```
int i=7, j=3;  
int *p;  
int **r;  
p = &i;  
r = &p;  
int c = **r + j;
```

8. Escreva um procedimento (void) em C que receba um vetor de inteiros e retorne o maior elemento e o menor elemento. Observe que o método deve retornar ao local da chamada os dois valores (não imprimir ao usuário). Portanto, você precisará usar passagem de parâmetro por referência, já que os procedimentos não podem retornar valor.

9. Forneça os valores serão apresentados com a execução do código abaixo. Além disso,

forneça o desenho da memória representando as variáveis e os seus respectivos valores.

```
int main(){
    int vet[]={5, 4, 6, 3};
    int aux, i; int *ptr1 = vet; int *ptr2 = vet;
    ptr2++;

    if (*ptr1 < *ptr2){
        aux = *ptr2; *ptr2 = *ptr1; *ptr1 = aux;
    }
    ptr1++; ptr2++;
    if (*ptr1 < *ptr2){
        aux = *ptr2; *ptr2 = *ptr1; *ptr1 = aux;
    }
    for(i=0; i<4; i++){
        printf("%d ", vet[i]);
    }
    system("pause");
}
```







10. Assumindo que vet[] é um vetor do tipo int, e que pulo é um ponteiro apontando para o primeiro elemento do vetor, quais das seguintes expressões seria a correta para recuperar o valor contido na terceira posição do vetor?

a) *(pulo + 2) b) *(pulo + 4) c) pulo + 4 d) pulo + 2

11. Forneça os valores serão apresentados com a execução do código abaixo. Além disso, forneça o desenho da memória representando as variáveis e os seus respectivos valores.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int * ptr, ** ptr2 = &ptr, *** ptr3 = & ptr2;
    int x = 10, y = 5;
    ptr = &x;
    x++;
    printf("%d, %d \n", *ptr, x);
    printf("%d, %d, %d \n", *ptr, **ptr2, ***ptr3);
    ptr = &y;
    ***ptr3 = (**ptr2) * (*ptr);
    printf("%d, %d \n", x, y);
}
```

12. Considerando as declarações abaixo forneça o valor correto das sentenças abaixo, considerando que um tipo int ocupa 2 bytes e um tipo float 4 bytes de memória.

X	Y	ptr_x	ptr_y	ptr_ptr_x	ptr_ptr_y
					
104	300	550	800	1001	111 1

int x = 28, *ptr_x = &x, **ptr_ptr_x = &ptr_x;

float y = 39.3, *ptr_y = &y, **ptr_ptr_y = &ptr_y;

- | | | |
|-------------|------------------|-----------------|
| a) x= | g) *ptr_ptr_x = | m) &ptr_ptr_y = |
| b) *ptr_y = | h) ptr_y = | n) ptr_x++ = |
| c) ptr_x = | i) &x = | |
| d) &y = | j) ptr_y++ = | |
| e) *ptr_x = | k) &ptr_ptr_x = | |
| f) y= | l) **ptr_ptr_y = | |

13. Declare e inicie um vetor de strings. Mostre todos os elementos do vetor criado, utilizando apenas ponteiros.

14. Qual das instruções abaixo é correta para declarar um ponteiro para inteiro?

- a. *int pti;
- b. *pti;
- c. &i;
- d. int_pti pti;
- e. int *pti;

15. Seja a seguinte sequência de instruções em um programa C:

```
int *pti;  
int i = 10;  
pti = &i;
```

Qual afirmativa é falsa?

- a. pti armazena o endereço de i
- b. *pti é igual a 10
- c. ao se executar *pti = 20; i passará a ter o valor 20
- d. ao se alterar o valor de i, *pti será modificado
- e. pti é igual a 10

16. Se i e j são variáveis inteiras e pi e pj são ponteiros para inteiro, qual atribuição é ilegal?

- a. pi = &i;
- b. *pj = &j;
- c. pj = *&j;
- d. i = *&j;
- e. i = (*pi)+++*pj;

17. Seja a seguinte sequência de instruções em um programa C:

```
int *pti;  
int veti[]={ 10,7,2,6,3};  
pti = veti;
```

Qual afirmativa é falsa?

- a. *pti é igual a 10

- b. $*(pti+2)$ é igual a 2
- c. $pti[4]$ é igual a 3
- d. $pti[1]$ é igual a 10
- e. $*(veti+3)$ é igual a 6

18. Na sequência de instruções abaixo:

```
float f;  
float *pf;  
pf = &f;  
scanf("%f", pf);
```

- a. Efetuamos a leitura de f
- b. Não efetuamos a leitura de f
- c. Temos um erro de sintaxe
- d. Deveríamos estar usando &pf no scanf
- e. Nenhuma das opções anteriores

19. Seja a seguinte sequência de instruções

```
int i=10, j=20;  
int *pti, *ptj;  
pti = &i;  
ptj = &j;  
Qual expressão não é válida?
```

- a. $j = pti == ptj$;
- b. $i = pti - ptj$;
- c. $pti += ptj$;
- d. $pti++$;
- e. $i = pti \parallel ptj$;

20. Seja a declaração:

```
int matr[][4] = { 1,2,3,4,5,6,7,8,9,10,11,12 }  
Qual afirmativa é falsa?
```

- a. $**matr$ é igual a 1
- b. $*(matr+1)+2$ é igual a 7
- c. $*(matr[2]+3)$ é igual a 12
- d. $((matr+2))[2]$ é igual a 11
- e. $((**matr)+1)$ é igual a 5

21. Preencha o valor das tabelas 1 e 2, baseando-se no Programa 1:

Programa 1

```
main()
{
    int i, j, *p_1, *p_2, **p_p_1, **p_p_2;
    i = 4;
    j = 5;
    p_1 = &i;
    p_2 = &j;
    p_p_1 = &p_2;
    p_p_2 = &p_1;
}
```

Tabela 1

Nome Variável	i	j	p_1	p_2	p_p_1	p_p_2
Conteúdo	4	5				
Endereço	1000	1007	1030	1053	1071	1079

Tabela 2

Expressão	i	*p_2	&i	&p_2	**p_p_1	*p_p_2	&*p_1	j	*p_1	*&p_1
Resultado										

22. Preencha o valor das tabelas 3 e 4, baseando-se no Programa 2:

Programa 2

```
main()
{
    int i, j, *p_1, *p_2, **p_3, ***p_4;
    i = 4;
    j = 5;
    p_1 = &j;
    p_2 = &i;
    p_3 = &p_1;
    p_4 = &p_3;
}
```

Tabela 3

Nome Variável	i	j	p_1	p_2	p_3	p_4
Conteúdo	4	5				
Endereço	1000	1007	1030	1053	1071	1079

Tabela 4

Expressão	i	*p_2	&j	&p_2	**p_3	**p_4	***p_4	*p_1	*&p_2	*p_4
Resultado										

23. Responda as perguntas abaixo. Se necessário, faça testes no programa em C.

a) Explique a diferença entre

p++; (*p)++; *(p++);

b) O que quer dizer *(p+10);?

24. O que será impresso com a execução do programa abaixo.

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
    int i = 10 , j = 20;
    int temp ;
    int *p1 , * p2 ;
    p1 = &i;
    p2 = &j;
    temp = * p1 ;
    * p1 = * p2 ;
    * p2 = temp ;
    printf ("%d %d\n" , i , j );
    return 0;
}
```

25. Qual o valor de y no final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. A seguir, escreva um /*comentário */ em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda do '=' após sua execução.

26.

27. int main()

28. {

29. int y, *p, x;

30. y = 0;

31. p = &y;

32. x = *p;

33. x = 4;

34. (*p)++;

35. x¬¬;

36. (*p) += x;

37. printf ("y = %d\n", y);

38. return(0);

39. }

- 40.
41. 1. Assumindo que `M1[]` é um vetor do tipo `int`, quais das seguintes expressões referenciam o valor do terceiro elemento de `M1`?
42. a) `*(M1 + 2)` b) `*(M1 + 4)` c) `M1 + 4` d) `M1 + 2`
- 43.
- 44.
45. Seja `vet` um vetor de 4 elementos: TIPO `vet[4]`. Supor que depois da declaração, `vet` esteja armazenado no endereço de memória 4092 (ou seja, o endereço de `vet[0]`). Supor também que na máquina usada uma variável do tipo `char` ocupa 1 byte, do tipo `int` ocupa 2 bytes, do tipo `float` ocupa 4 bytes e do tipo `double` ocupa 8 bytes.
- 46.
47. Qual o valor de `vet+1`, `vet+2` e `vet+3` se:
- 48.
49. a) `vet` for declarado como `char`?
50. b) `vet` for declarado como `int`?
51. c) `vet` for declarado como `float`?
52. d) `vet` for declarado como `double`?
- 53.
54. Verifique o programa abaixo. Encontre o seu erro e corrija-o para que escreva o numero 10 na tela.
55. `#include <stdio.h>`
56. `int main()`
57. {
58. `int x, *p, **q;`
59. `p = &x;`
60. `q = &p;`
61. `x = 10;`
62. `printf("\n%d\n", &q);`
63. `return(0);`
64. }
- 65.
66. Complete o programa abaixo. Este programa usa a função `void troca (int *a, int *b)`. Esta função troca os valores apontados por `a` e `b`.
- 67.
68. `#include <stdio.h>`
69. `void troca (int *a , int *b)`
70. {
71. `int temp ;`
- 72.
73. }
- 74.
75. `int main (void)`
76. {
77. `int x , y;`
78. `scanf ("%d %d" , &x , &y);`
79. `troca (&x , &y);`
80. `printf (" Troquei ----> %d %d\n" , x , y);`
81. `return 0;`
82. }

83.

84. O que imprime o programa abaixo.

85.

86. `#include <stdio.h>`

87. `char * misterio (char frase [])`

88. `{`

89. `char * pfrase ;`

90. `int i = 0;`

91. `while (frase[i] != ' ')`

92. `{`

93. `i ++;`

94. `}`

95. `i ++;`

96. `pfrase = &frase[i];`

97. `return pfrase ;`

98. `}`

99.

100. `int main (void)`

101. `{`

102. `char frase[80];`

103. `char *p;`

104. `strcpy(frase, "A casa caiu!");`

105. `p = misterio(frase);`

106. `puts (p);`

107. `return 0;`

108. `}`

109.

110.

111.

112. Complete o programa abaixo, que converte um número inteiro positivo da base 10 para a base 2. A função recebe o número na variável `numeroBase10` e retorna todos os 32 bits no vetor

113. `numeroBase2`. O número na base 2 deve ser armazenado no vetor da seguinte maneira: bit 31 na posição 31, bit 30 na posição 30 e assim sucessivamente.

114.

115. Entrada

116.

117. A entrada é composta de vários conjuntos de teste. Cada caso de teste é um número inteiro positivo. A entrada termina quando um número inteiro negativo for digitado.

118.

119. Saída

120.

121. Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste n", onde n é numerado a partir de 1. A

122. segunda linha deve conter o número lido. A terceira linha deve conter todos os bits do número convertido, inclusive os zeros à esquerda. A quarta linha deve ser deixada em branco. A formato

123. mostrado no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

124.
125. Exemplo de entrada e saída

```
126.  
127.  
128.  
129. #include <stdio.h>  
130. #include <stdlib.h>  
131.  
132. void Converter ( int numeroBase10 , int numeroBase2 [32])  
133. {  
134.  
135. }  
136.  
137. int main ( int argc , char * argv [])  
138. {  
139.     int nb10 , nb2 [32] , i , teste = 1;  
140.     while (1){  
141.         scanf("%d" , &nb10);  
142.         if( nb10 < 0 ) break;  
143.  
144.         Converter ( nb10 , nb2 );  
145.         printf ( " Teste %d\n" , teste ++);  
146.         printf ("%d\n" , nb10 );  
147.  
148.         /* Imprimir o vetor aqui*/  
149.  
150.         printf ("\n\n" );  
151.     }  
152.     return 0;  
153. }
```

154.
155.
156. Usando apenas ponteiros e alocação de memória, criar um vetor de números inteiros de tamanho 3, preenche-lo com valores fornecidos pelo usuário e, logo após, apresentar os valores armazenados.

157.
158. Usando apenas ponteiros e alocação de memória, criar uma matriz de números inteiros de tamanho 4 X 4, preenche-la com valores fornecidos pelo usuário e, logo após, apresentar os valores armazenados. Lembre-se que uma matriz necessita de duas dimensões para acessar uma determinada posição.

Listas Encadeadas

1. Criar uma função que receba uma lista encadeada de números reais(float ou double) e retorne a soma de todos os elementos existentes na lista.
2. Criar uma função que receba uma lista encadeada de números inteiros e retorne quantidade de elementos existentes na lista.
3. Criar uma função que receba uma lista encadeada de números reais(float ou double) e retorne a média de todos os elementos existentes na lista.
4. Criar uma função para verifique se duas listas simplesmente encadeadas são iguais. Listas iguais são aquelas que têm a mesma sequência de elementos. O protótipo da função deve ser `int comparar_listas (lista * l1, lista * l2)`, retornando 1 para listas iguais e 0 para listas diferentes.
5. Considerando uma lista encadeada simples que armazena números inteiros, implemente uma função que receba um vetor de valores inteiros com n elementos e construa uma lista encadeada contendo os mesmos elementos do vetor e na mesma ordem. Use o seguinte protótipo para a função: **lista* construirLista(int n, int * v)** onde lista é o tipo de dado que define um elemento da lista.
6. Criar uma função para verifique se duas listas simplesmente encadeadas são iguais. Listas iguais são aquelas que têm a mesma sequência de elementos. O protótipo da função deve ser `int igual (No* l1, No* l2)`, retornando 1 para listas iguais e 0 para listas diferentes.
7. Considerando uma lista simplesmente encadeada que armazena caracteres, desenvolva uma função que receba uma string e retorne uma lista contendo os caracteres da string enviada.
8. Considerando uma lista simplesmente encadeada que armazena número inteiros, criar uma função que receba por parâmetro uma lista e retorne em uma nova lista, os elementos em ordem inversa a lista original.
9. Considerando uma lista simplesmente encadeada que armazena número inteiros, criar uma função que receba uma lista e um número n. A função deverá remover os primeiros n elementos da lista e retornar a lista alterada. Use o seguinte protótipo para a função: **lista* removerElementos(int n, lista * inicio)** onde lista é o tipo de dado que define um elemento da lista.
10. Criar uma função que crie uma cópia de uma lista encadeada, ou seja, a lista retornada por esta função é uma nova lista contendo os mesmos elementos(valores) da lista passada por parâmetro. O protótipo da função deve ser dado por: `lista * copiar_lista (lista* l)`;
11. Faça uma função para unir duas listas encadeadas simples **L1** e **L2**, recebidas por parâmetro, criando a lista encadeada simples **L3** que deve ser o retorno desta função. Cada elemento da lista é formado pela informação código e pelo ponteiro para o próximo elemento. A lista **L3** não pode possuir elementos com o mesmo código. Assim, se um elemento de **L1** está em **L2** este só deve aparecer uma única vez em **L3**.

Filas e Pilhas

1. Criar uma função que receba por parâmetro um vetor de números inteiros e retorne uma pilha contendo os mesmos elementos.
2. Utilizando somente operações de empilhar e desempilhar, escreva um programa que remove um item com chave c fornecida pelo usuário da pilha. Ao final da execução da função, a pilha deve ser igual à original, exceto pela ausência do item removido.
3. Escreva um programa que tenha uma fila cujos elementos possuem um campo inteiro representando sua prioridade. Quanto menor o valor deste campo, maior a prioridade do elemento. Insira n elementos com prioridades diversas na fila e depois divida a fila em duas, uma com elementos cuja prioridade é menor ou igual ao valor p fornecido pelo usuário e outra com os elementos restantes.
4. Escreva uma função que inverte os elementos de uma fila usando uma pilha.

```
void inverte_fila(Fila *fila);
```

5. Escrever um algoritmo que leia um número indeterminado de valores inteiros. O valor 0 (zero) finaliza a entrada de dados. Para cada valor lido, determinar se ele é um número par ou ímpar. Se o número for par, então incluí-lo na FILA PAR; caso contrário, incluí-lo na FILA ÍMPAR. Após o término da entrada de dados, retirar um elemento de cada fila alternadamente (iniciando-se pela FILA ÍMPAR) até que ambas as filas estejam vazias. Se o elemento retirado de uma das filas for um valor positivo, então incluí-lo em uma PILHA; caso contrário, remover um elemento da PILHA. Finalmente, escrever o conteúdo da pilha.
159. Criar um validador de escopos de expressões matemáticas usando uma pilha. O programa deve receber como entrada uma expressão matemática do tipo $(a * b) + (b * c) \{[(a * b) + (b * c)] - (a * c)\} * j$ e validar se a abertura e fechamento de chaves, colchetes e parênteses estão corretas (a função não necessita validar se as variáveis e operadores matemáticos estão corretos). O programa deverá apenas validar se os escopos estão corretamente delimitados se para cada “(” ou “[” ou “{” existe um “)” ou “]” ou “}” na ordem correta.