

Estruturas de Dados

Exercícios – Capítulos 5 e 7



Questão 1

A média aritmética, A , e a média harmônica, H , de um conjunto de valores são dadas pelas seguintes fórmulas:

$$A = \frac{1}{n} \sum v_i \qquad \frac{1}{H} = \frac{1}{n} \sum \frac{1}{v_i}$$

Implemente uma função que calcule a média aritmética e a média harmônica de um conjunto de valores armazenado num vetor. Esta função deve obedecer ao protótipo abaixo, recebendo como parâmetros o número de elementos do vetor, o endereço inicial do vetor e os endereços das variáveis onde os valores das médias, isto é, os valores de A e H , devem ser armazenados.

```
void medias (int n, float* v, float* pA, float* pH);
```

Questão 1

```
/* Cálculo da média aritmética e da média harmônica */
#include <stdio.h>
/* Função medias */
void medias (int n, float* v, float* pA, float* pH)
{ float sA = 0;
  float sH = 0;
  int i;
  for(i=0;i<n;i++)
  {      sA = sA + v[i];
        sH = sH + 1/v[i];
  }
  *pA = sA/n;
  *pH = n/sH;
}
```

Questão 1

```
/* Função principal (para teste) */  
int main (void) {  
    float v[] = {3.0, 10.0, 5.0, 3.0};  
    int n = 4;  
    float pA, pH;  
    medias(n, v, &pA, &pH);  
    printf("%f %f\n", pA, pH);  
    return 0;  
}
```

Questão 2

Implemente uma função que receba como parâmetro um vetor v de n números inteiros e retorne um novo vetor w , alocado dinamicamente, cujos elementos são definidos pelas fórmulas:

$$w_0 = v_0$$

$$w_i = v_i + w_{i-1}, \quad 0 < i < n$$

Essa função não deve alterar o conteúdo do vetor original v e seu protótipo deve ser:

```
int* somatorios (int n, int* v);
```

Questão 2

```
/* Cálculo de somatório */  
  
int* somatorios (int n, int* v)  
{  
    int i;  
    int *w;  
    /* alocação dinâmica */  
    w = (int*) malloc(n*sizeof(int));  
    w[0] = v[0];  
    for (i = 1; i < n; i++)  
        w[i] = v[i] + w[i-1];  
    return w;  
}
```

Questão 2

- Uso incorreto de apontador:

```
/* Cálculo de somatório */  
  
int* somatorios (int n, int* v)  
{  
    int i;  
    int *w;  
    /* alocação dinâmica */  
    w = (int*) malloc(n*sizeof(int));  
    *w[0] = *v[0];  
    for (i = 1; i < n; i++)  
        *w[i] = *v[i] + *w[i-1];  
    return &w;  
}
```

Questão 3

Implemente uma função que receba como parâmetros uma cadeia de caracteres s e um caractere c , e retorne o índice da última ocorrência do caractere c em s . Por exemplo, se forem passados para essa função a cadeia “Rio de Janeiro” e o caractere ‘i’, a função deve retornar o valor 11. Caso não haja ocorrências do caractere procurado, a função deve retornar -1. Essa função deve ter o seguinte protótipo:

```
int ultima_ocorrencia (char* s, char c);
```

Questão 3

```
/* Última ocorrência de caractere em cadeia de caracteres */

#include <stdio.h>
#include <string.h>

int ultima_ocorrencia (char* s, char c)
{
    int i;
    int n = strlen(s);    /* comprimento da cadeia de entrada */
    for (i=n-1; i!=-1; i--) /* s[0] é o primeiro caractere de s */
        if (s[i] == c) break; /* s[n-1] é o último caractere de s */
    return i;             /* saída do “for” com i=-1, se c não ocorre em s */
                        /* saída do “for” via “break” com s[i]==c */
}
```

Questão 3

```
int main (void)
{
    char c;
    char cidade[] = "Rio de Janeiro";
    printf("Entre com o caractere a pesquisar:\n");
    scanf("%c", &c);
    printf("%d", ultima_ocorrencia(cidade, c));
    return 0;
}
```

Questão 3

- Símbolo incorreto para o operador de comparação:

```
int ultima_ocorrencia (char* s, char c)
{
    int i;
    int n = strlen(s); /* comprimento da cadeia de entrada */
    for (i=n-1; i!=-1; i--)
        if (s[i] = c) break;
    return i;
}
```

Questão 3

- Condição de parada incorreta:

```
int ultima_ocorrencia (char* s, char c)
{
    int i;
    int n = strlen(s);    /* comprimento da cadeia de entrada      */
    for (i=n-1; i!=0; i--) /* não compara c com o primeiro caractere de s */
        if (s[i] == c) break;
    return i;
}
```

Questão 4

Implemente uma função que crie uma cadeia de caracteres inversa, trocando a ordem dos caracteres. A função deve receber como parâmetro de entrada uma cadeia de caracteres e retornar uma nova cadeia, cujo espaço de memória deve ser alocado pela função, contendo a cadeia original invertida: o primeiro caractere passa a ser o último, o segundo o penúltimo, e assim por diante. Assim, se for fornecido a cadeia “puc-rio”, deve-se retornar a cadeia “oir-cup”. A cadeia de caracteres original não pode ser alterada. A assinatura da função deve ser:

```
char* inverte (char* s);
```

Questão 4

- Tentativa de Solução 1:

```
char* inverta (char* s)
{
    int i;
    int n = strlen(s);    /* comprimento da cadeia de entrada */
    char* w = (char*) malloc((n)*sizeof(char));
    for (i=n; i!=-1; i--) /* copia s para w do fim para o início */
        w[n-i] = s[i];
    w[n] = '\0';          /* fecha a cadeia w */
    return w;
}
```

– Qual o erro?

Questão 4

- Tentativa de Solução 1:
 - erro: ignorar que uma cadeia s de comprimento n é um vetor de **n+1** posições onde s[0] a s[n-1] são os caracteres e s[n]='\0';

```
char* inverte (char* s)
{
    int i;
    int n = strlen(s);    /* comprimento da cadeia de entrada */
    char* w = (char*) malloc((n)*sizeof(char));
    for (i=n; i!=-1; i--) /* copia s para w do fim para o início */
        w[n-i] = s[i];
    w[n] = '\0';         /* fecha a cadeia w – fora de w !    */
    return w;
}
```

Questão 4

- Tentativa de Solução 1:

```
char* invertre (char* s)
{
    int i;
    int n = strlen(s);    /* comprimento da cadeia de entrada */
    char* w = (char*) malloc((n+1)*sizeof(char));
    for (i=n; i!=-1; i--) /* copia s para w do fim para o início */
        w[n-i] = s[i];
    w[n] = '\0';         /* fecha a cadeia w – agora dentro de w */
    return w;
}
```

- Qual o erro?

Questão 4

- Tentativa de Solução 2:
 - erro: ignorar que `s[n]='\0'` não deve ser copiado para `w[0]`, ou seja, que `s[0]` a `s[n-1]` são os caracteres e `s[n]='\0'`

```
char* inverte (char* s)
{
    int i;
    int n = strlen(s);    /* comprimento da cadeia de entrada */
    char* w = (char*) malloc((n+1)*sizeof(char));
    for (i=n; i!=-1; i--) /* copia s para w do fim para o início */
        w[n-i] = s[i];
    w[n] = '\0';         /* fecha a cadeia w */
    return w;
}
```

Questão 4

```
/* Inversão de cadeia de caracteres */

char* inverte (char* s)
{
    int i;
    int n = strlen(s);      /* comprimento da cadeia de entrada */
    char* w = (char*) malloc((n+1)*sizeof(char));
    for (i=n-1; i!=-1; i--) /* copia s para w do fim para o início */
        w[n-1-i] = s[i];
    w[n] = '\0';           /* fecha a cadeia w */
    return w;
}
```

Questão 4

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

char* invert(char* s);

int main (void)
{
    char v[121];
    char* w;
    printf("Entre com a cadeia até 120 caracteres e tecla enter:\n");
    scanf(" %[^\\n]", v);
    w = invert(v);
    printf("%s", w);
    return 0;
}
```

Questão 4

- Solução alternativa:

```
char* inverte (char* s)
{
    int i;
    int n = strlen(s);      /* comprimento da cadeia de entrada */
    char* w = (char*) malloc((n+1)*sizeof(char));
    for (i=0; s[i]!='\0'; i++) /* copia s para w do início para o fim */
        w[n-1-i] = s[i];
    w[n] = '\0';           /* fecha a cadeia w */
    return w;
}
```

Questão 5

Considere uma função, chamada `minmax`, que determina o menor e o maior elemento de um vetor *qualquer* de valores reais. O programa abaixo ilustra a utilização dessa função. Se fosse executado, este programa imprimiria os valores `-3.2` e `5.5`, que representam, respectivamente, o menor e o maior elemento do vetor em questão.

```
#include <stdio.h>
int main (void) {
    float v[5] = {-3.2, 4.5, 2.0, 5.5, -1.5};
    float a, b;
    minmax( 5, v, &a, &b ); /* chama minmax */
    printf("%.1f  %.1f\n", a, b);
    return 0;
}
```

Escreva a função `minmax` para que o programa acima funcione de maneira correta. Note que a função deve funcionar para vetores de valores reais de qualquer dimensão.

Questão 5

```
void minmax(int n, float* vet, float* pmin, float* pmax)
{
    int i;
    if (n<=0) return;
    *pmin=*pmax=vet[0];
    for(i=1;i<n;i++)
    {
        if (*pmin>vet[i]) *pmin=vet[i];
        if (*pmax<vet[i]) *pmax=vet[i];
    }
}
```

Questão 5

```
#include <stdio.h>

void minmax(int n, float* vet, float* pmin, float* pmax);

int main( )
{
    float v[5]={-3.2f, 4.5f, 2.0f, 5.5f, -1.5f};
    float a,b;
    minmax(5,v,&a,&b);
    printf("%.1f %.1f\n",a,b);
    return 0;
}
```

Questão 6

Uma forma muito simples de codificarmos uma mensagem consiste em deslocar o alfabeto de uma posição. Assim, a letra 'a' passa a ser representada pela letra 'b', a letra 'b' pela letra 'c', e assim por diante. Como o deslocamento é cíclico, a letra 'z' passa a ser representada pela letra 'a'. Uma regra análoga é aplicada às letras maiúsculas ('A' para 'B', 'B' para 'C', ..., 'Z' para 'A'). A codificação afeta apenas as letras da cadeia, preservando os demais caracteres. Assim, a codificação da cadeia "PUC-Rio" resulta em "QVD-Sjp".

(a) Escreva uma função para codificar uma cadeia. A função deve receber como parâmetro de entrada uma cadeia de caracteres (*string*) que deve ser codificada, conforme apresentado acima. A função deve obedecer ao protótipo abaixo:

```
void codifica (char* s);
```

(b) Escreva um programa (função main) para testar sua função de codificação. Esse programa deve capturar uma cadeia com até 80 caracteres, fornecida pelo usuário via teclado, e imprimir a cadeia codificada.

Dica: para capturar uma cadeia com eventuais brancos com até 80 caracteres, use o formato: "%80[^\n]".

Questão 6

```
void codifica(char *s)
{
    int i, n=strlen(s);
    for(i=0;i<n;i++)
    {
        if ((s[i]>='a'&& s[i]<'z') || (s[i]>='A'&& s[i]<'Z')) s[i]++;
        if (s[i]=='z') s[i]='a';
        if (s[i]=='Z') s[i]='A';
    }
}
```

Questão 6

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

void codifica(char* s);

int main ( )
{
    char msg[81];
    printf("Mensagem: ");
    scanf(" %80[^\n]",msg);
    codifica(msg);
    printf("%s\n",msg);
}
```