UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Departamento de Informática e Matemática Aplicada

Introdução às Técnicas de Programação

◁ Exercícios - parte 8 ▷

1. Em relação ao setor de memória na Figura da página a seguir (um fictício com endereços de 16 bits):

   (a) Qual o endereço base de 3.2 e qual o do número inteiro 503?

   (b) O que é escrito na tela com printf("%s\n", p0);?

   (c) O que é escrito na tela com printf("%s\n", &p0[5]);?

   (d) O que é escrito na tela com printf("%s\n", p0+2);?

   (e) Qual o valor de *p1?

   (f) Qual o valor de p1[0]?

   (g) Qual o valor de p1+1?

   (h) Qual o valor de *(p1+1)?

   (i) Qual o valor de *(p2+1)?

   (j) Qual o valor de &x?

   (k) Qual o valor de &p0?

0x85ba	0	1	1	0	0	0	1	1	**char** c
0x85bb	0	1	1	0	0	0	0	1	**char** a
0x85bc	0	1	1	1	0	0	1	1	**char** s
0x85bd	0	1	1	0	0	0	0	1	**char** a
0x85be	0	0	0	0	0	0	0	0	**char** \0
0x85bf	0	1	1	0	0	0	0	1	**char** a
0x85c0	0	1	1	0	1	1	0	0	**char** l
0x85c1	0	1	1	0	1	0	0	1	**char** i
0x85c2	0	0	0	0	0	0	0	0	**char** \0
0x85c3	0	1	0	0	0	0	0	0	**float** 3.2
0x85c4	0	1	0	0	1	1	0	0	
0x85c5	1	1	0	0	1	1	0	0	
0x85c6	1	1	0	0	1	1	0	1	
0x85c7	0	1	0	0	0	0	0	0	**float** 5.4
0x85c8	1	0	1	0	1	1	0	0	
0x85c9	1	1	0	0	1	1	0	0	
0x85ca	1	1	0	0	1	1	0	1	
0x85cb	0	0	1	1	1	1	1	1	**float** 1.2
0x85cc	1	0	0	1	1	0	0	1	
0x85cd	1	0	0	1	1	0	0	1	
0x85ce	1	0	0	1	1	0	1	0	
0x85cf	0	1	0	0	0	0	0	1	**float** 8.4
0x85d0	0	0	0	0	0	1	1	0	
0x85d1	0	1	1	0	0	1	1	0	
0x85d2	0	1	1	0	0	1	1	0	
0x85d3	1	0	0	0	0	1	0	1	**char *p0 =** 0x85ba
0x85d4	1	0	1	1	1	0	1	0	
0x85d5	0	0	0	0	0	0	0	0	**int x =** 503
0x85d6	0	0	0	0	0	0	0	0	
0x85d7	0	0	0	0	0	0	0	1	
0x85d8	1	1	1	1	0	1	1	1	
0x85d9	0	1	1	0	0	1	0	0	**char** d
0x85da	0	0	0	0	0	0	0	0	**int** 702
0x85db	0	0	0	0	0	0	0	0	
0x85dc	0	0	0	0	0	0	1	0	
0x85dd	1	0	1	1	1	1	1	0	
0x85de	1	0	0	0	0	1	0	1	**float *p1 =** 0x85c3
0x85df	1	1	0	0	0	0	1	1	
0x85e0	1	0	0	0	0	1	0	1	**float *p2 =** 0x85cb
0x85e1	1	1	0	0	1	0	1	1	
0x85e2	1	0	0	0	0	1	0	1	**int *p3 =** 0x85ca
0x85e3	1	1	0	0	1	0	1	0	

Figura 1: Setor da memória para a primeira questão

2. ▷ Altere o ponteiro p para apontar para o início da segunda palavra da string. Assuma que a frase digitada possui ao menos uma palavra.

```c
#include <stdio.h>

int main() {

    char frase[200];

    gets(frase);

    char *p = frase;

    //altere p para que aponte para o primeiro caractere apos o primeiro espaco

    printf("%s\n", p);

    return 0;
}
```

3. ▷ Altere o valor inicial de p de forma que a saída do programa seja 5 11 12 2 8.

```c
#include <stdio.h>

int main() {

    int i;
    int v[] = {3, 14, 9, 6, 5, 11, 12, 2, 8, 13, 7, 10, 1, 4};

    int *p; //atribua um valor inicial adequado

    for(i = 0; i < 5; i++) {
        printf("%d ", p[i]);
    }

    return 0;
}
```

4. ▷ Escreva um programa que leia um inteiro **n**, leia **n números reais** e escreva na tela o índice (começando de 1) do maior entre esses **n** números reais. Assuma que não há números iguais na sequência.

Exemplo:

6
3.97 2.15 13.97 12.38 10.65 16.19
6

5. ▷ O MEC precisa de sua ajuda (de novo!) para automatizar a correção das provas objetivas do ENEM. Mas dessa vez o MEC não tem ideia do número máximo de questões que pode haver. Escreva um programa que leia um número inteiro **n** representando o número de questões. Em seguida leia as **n** respostas do gabarito e, em seguida, as **n** respostas do aluno. Assuma que as respostas estão sempre entre 1 e 5. Depois o programa deve escrever na tela quantas questões o aluno acertou e a string "acertos" ou "acerto" (para 1 acerto), conforme exemplo abaixo.

   Exemplo 2:
   ```
   4
   1 2 3 4
   1 5 3 5
   2 acertos
   ```

   Exemplo 2:
   ```
   7
   1 2 3 2 1 5 4
   3 3 3 3 3 3 3
   1 acerto
   ```

6. ▷ Escreva um programa que leia um inteiro **n**, leia **n** inteiros que estão em ordem crescente, um inteiro **m** e **m** inteiros que também estão em ordem crescente. O programa deve em seguida escrever na tela uma única sequência ordenada com os **m+n** inteiros.

   Exemplo:
   ```
   4
   1 2 3 4
   4
   4 6 7 9
   1 2 3 4 4 6 7 9
   ```

7. ▷ Durante as eleições algumas seções de votação possuem filas consideráveis. A fila é formada por ordem de chegada, mas os idosos (65 anos ou mais) possuem prioridade. Quando o fiscal chega, ele passa todos os idosos para a frente da fila, mantendo a ordem de chegada desses idosos. Enquanto o fiscal não chega, você percebeu que dá para saber a posição de qualquer pessoa após a reorganização da fila mesmo sem mudar a posição de ninguém, bastando saber a idade de cada um.

   O seu programa deve ler um inteiro **n** representando a quantidade de pessoas na fila, a posição (começando de 1) da pessoa de quem queremos saber a posição final e **n** números que representam a idade de cada um da fila, começando pelo primeiro da fila. O programa deve então escrever na tela a posição final dessa pessoa.

   Exemplo 1 (5 pessoas, queremos saber a posição final da segunda pessoa da fila):
   ```
   5 2
   30 20 25 40 19
   2
   ```

   Exemplo 2 (12 pessoas, queremos saber a posição final da quarta pessoa da fila):
   ```
   12 4
   27 65 51 65 41 62 22 55 65 46 90 63
   2
   ```

Exemplo 3 (17 pessoas, queremos saber a posição final da décima terceira pessoa da fila):

```
17 13
27 33 31 46 27 47 46 83 37 53 18 56 47 37 80 87 29
15
```

8. ▷ Um fazendeiro está construindo um cercado para os seus animais e para isso fixou algumas estacas por onde deverá passar uma cerca (veja figura). Ele já possui as coordenadas de cada uma das estacas e precisa saber quantos metros terá a cerca, assim poderá dimensionar a quantidade de material que utilizará para concluir o cercado. Entre a última e a primeira estaca não há cerca, pois ali será colocada uma porteira (segmento em pontilhado na figura). Escreva um programa que leia um inteiro **n** (assuma $n > 2$), faça a leitura de n pares de **floats** (vide exemplo) e em seguida escreva na tela o comprimento total da cerca com 2 casas decimais de precisão. Você pode assumir que as coordenadas são fornecidas na mesma ordem em que são utilizadas para formar o cercado. Utilize o tipo **float** para as coordenadas e cálculos de distância.

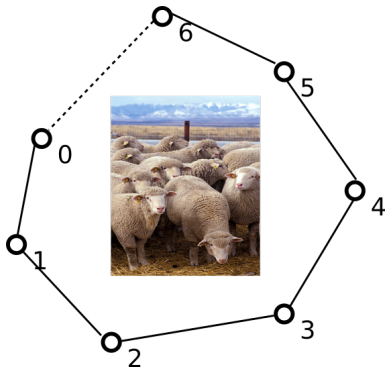

Figura 2: Exemplo com 7 estacas

O comprimento da cerca entre duas estacas é dada pela distância euclidiana entre elas:

$$\sqrt{\Delta x^2 + \Delta y^2}$$

onde Δx é a diferença entre as abscissas e Δy é a diferença entre as ordenadas.

Exemplo 1:

```
3
0.0 0.0
1.0 1.0
0.0 2.0
2.83
```

9. Escreva a seguinte função que recebe como parâmetro um vetor **v** de **n** números reais:

```
float norma(float v[], int n)
```

A função deve retornar a norma euclidiana dada por:

$$\sqrt{v_0^2 + v_1^2 + ... + v_{n-1}^2}$$

A função main deve ler do usuário um inteiro **n**, alocar dinamicamente um vetor **u** de **n** números reais e escrever na tela, com duas casas decimais de precisão, a norma de **u** fazendo uso da função norma.

10. Atoi é uma função em C que transforma uma string em número inteiro. Escreva a seguinte função obterInteiro que recebe como parâmetro um vetor **s** de caracteres (ou seja, uma string) e que retorna o inteiro contido da string observando essas regras:

    – pode haver caracteres diferentes de dígitos numéricos, mas eles são ignorados
    – o número pode ser negativo, desde que o - seja o **primeiro** caractere

    ```
    int obterInteiro(char s[])
    ```

    A função main deve ler do usuário uma palavra(30) e escrever na tela o inteiro contido na string fazendo uso da função obterInteiro. Por exemplo, se a string for ”tes100te”, a função deve retornar 100 e se a string for ”a-s1tri2g”deve retornar 12.

11. Escreva uma função para colocar uma string em caixa-alta:

    ```
    void caixaAlta(char s[])
    ```

    A função main deve ler do usuário uma frase(100) e escrever na tela a string em caixa-alta fazendo uso da função caixaAlta.

12. Escreva uma função para retornar uma **nova** string, igual à passada como parâmetro, mas em caixa-alta:

    ```
    char * caixaAlta(char s[])
    ```

    A função main deve ler do usuário uma frase(100) e escrever na tela a string em caixa-alta fazendo uso da função caixaAlta.

13. Escreva uma função que receba um inteiro **n** e retorne o endereço base de um vetor de **n** inteiros alocado dinamicamente.

    ```
    int * alocarVetor(int n)
    ```

    Escreva uma outra função que receba como parâmetros um inteiro **n1**, um inteiro **n2**, o endereço base de um vetor de **n1** inteiros, o endereço base de um vetor de **n2** inteiro e retorne o endereço base de um novo vetor com a soma dos respectivos elementos. Se $n1 \neq n2$, a função deve retornar **NULL**. Por exemplo, se um vetor contiver {1, 5, 2, 3} e o outro contiver {-2, 3, 2, 0}, o vetor resultante conterá os valores {-1, 8, 4, 3}.

    ```
    int * somaVetores(int n1, int n2, int *v1, int *v2)
    ```

    A função main deve:

    (a) ler do usuário um inteiro **n1**, um inteiro **n2**
    (b) alocar dinamicamente um vetor **u** de **n1** números inteiros e outro **v** de **n2** números inteiros
    (c) ler **n1** inteiros para **u** e **n2** inteiros para **v**
    (d) chamar a função somaVetores
    (e) escrever na tela o vetor resultante da soma dos dois vetores