

# Lista 7 - Vetores, Strings e Matrizes

1. Seja `w` um vetor contendo 9 elementos inteiros. Supondo que `i` seja uma variável inteira valendo 5, que valores estarão armazenados em `w` após as atribuições a seguir?
  - a. `w[0] = 17 ;`
  - b. `w[i/2] = 9 ;`
  - c. `w[2*i-2] = 95 ;`
  - d. `w[i-1] = w[8]/2 ;`
  - e. `w[i] = w[2] ;`
  - f. `w[i+ 1] = w[i]+ w[i-1] ;`
  - g. `w[w[2]-2] = 78 ;`
  - h. `w[w[i]-1] = w[1]*w[i] ;`
2. Codifique um programa para solicitar 5 números, via teclado, e exibi-los na ordem inversa àquela em que foram fornecidos.
3. Codifique um programa que indique a quantidade mínima de cédulas equivalente a uma dada quantia em dinheiro. Considere apenas valores inteiros e cédulas de 1, 5, 10, 50 e 100 reais.

```
Quantia? R$ 209
2 cédulas de R$100,00
1 cédula de R$5,00
4 cédulas de R$1,00
```

4. Codifique a função `minimax(v, n, a, b)`, que recebe um vetor `v` contendo `n` números reais e devolve em `a` e `b`, respectivamente, os valores mínimo e máximo entre aqueles armazenados em `v`.

5. Usando ponteiros, codifique a função `carrega(v,n)`, que preenche um vetor `v` com `n` valores lidos do teclado.
6. Usando ponteiros, codifique a função `puts(s)`, que exibe uma string `s` no vídeo e, depois, muda o cursor de linha.
7. Codifique a função `strcpy(s,t)`, que copia o conteúdo da string `t` para a string `s`. Essa função é útil quando precisamos realizar atribuição entre strings; por exemplo, para atribuir a constante "teste" a uma string `x`, basta escrever `strcpy(x,"teste")`.
8. Codifique a função `strlen(s)`, que devolve o número de caracteres armazenados na string `s`. Lembre-se de que o terminador `'\0'` não faz parte da string e, portanto, não deve ser contado.
9. Codifique a função `strcat(s,t)`, que concatena a string `t` ao final da string `s`. Por exemplo, se `x` armazena "facil" e `y` armazena "idade", após a chamada `strcat(x,y)`, `x` estará armazenando "facilidade".
10. Codifique a função `strpos(s,c)`, que devolve a posição da primeira ocorrência do caracter `c` na string `s`; ou `-1`, caso ele não ocorra em `s`.
11. Codifique a função `strdel(s,p)`, que remove o caracter existente na posição `p` da string `s`. Se a posição `p` não existe em `s`, nada é feito.
12. Codifique um programa para ler uma matriz quadrada de ordem `n` e exibir apenas os elementos da diagonal principal.
13. Crie um programa que inicializa e exibe uma matriz representando um tabuleiro para o "jogo da velha", conforme a seguir:

```

      | o | x
----+---+---
      | x | o
----+---+---
      |   |

```

14. Codifique um programa para jogar campo minado. O programa deve solicitar uma posição ao usuário e mostrar o campo, repetidamente, até que a posição (0,0) seja fornecida ou que o usuário jogue numa posição minada.
15. Usando as funções `randomize()` e `random()`, ambas definidas em `stdlib.h`, crie uma função para preencher, aleatoriamente, uma matriz 10×10

representando um campo minado 8×8. Os limites da matriz devem ser preenchidos com uns e o interior, com 0's nas posições livres e 9's nas posições contendo bombas. Utilize um parâmetro adicional na sua função para especificar a "facilidade" do campo minado, de tal modo que quanto maior for a facilidade, menor seja número de bombas no campo.

16. Crie uma função que recebe um campo minado e marca, para cada posição livre, o número de posições adjacentes que contêm bombas, conforme exemplificado a seguir

*1001*3*
331012*2
**100232
23212*2*
*11*3221
22323*21
1*4*223*
2**21011

Codifique uma função que recebe como argumentos um campo minado marcado, vide exercício anterior, e as coordenadas de uma posição dele. A função deve marcar a posição indicada como "aberta" e exibir o campo minado no vídeo, conforme ilustração a seguir, mostrando apenas as posições já "abertas". Além disso, a função deve devolver 1 se a posição estiver livre e 0 se tiver uma bomba.

	12345678
1	◆◆◆◆◆◆◆◆
2	◆◆◆01◆◆◆
3	◆◆1002◆◆
4	◆◆◆12◆◆◆
5	◆11◆3221
6	◆◆◆◆◆◆◆◆
7	◆◆4◆◆◆◆◆
8	◆◆◆◆◆◆◆◆

17. Além da adição de ponteiros com inteiros, uma outra operação possível é a subtração entre ponteiros do mesmo tipo. Quando um ponteiro é subtraído

de outro, o resultado é o número de elementos existentes no espaço de memória compreendido entre os endereços apontados por eles. Usando essa idéia, codifique a função `strlen(s)`, que devolve o número de caracteres existentes numa string `s`.

18. A biblioteca padrão da linguagem C oferece uma função polimórfica para ordenação de vetores cujo protótipo é o seguinte:

```
void qsort(void *v, int n, int t, int (*c)(const void *,const void *));
```

sendo `v` um ponteiro para o vetor a ser ordenado,

`n` o número de elementos no vetor, `t` o tamanho em bytes de cada elemento e `c` um ponteiro para a função de comparação a ser usada durante ordenação. A novidade nesse protótipo é o tipo `void*`, que serve para declarar ponteiros capazes de receber qualquer tipo de endereço. Isso quer dizer que, por exemplo, o parâmetro `v` pode receber como valor inicial tanto o endereço de um vetor de caracteres quanto o endereço de um vetor de números reais ou ainda um vetor de estruturas. O único problema com ponteiros `void` é que não é permitido o acesso a dados a partir deles e, portanto, temos que usar casts com eles. Com base nessas informações, crie um vetor de estruturas e tente ordená-lo por cada um de seus campos, um de cada vez, usando a função `qsort()`.