Vetores

1. Criar um vetor para armazenar números pares e outro para armazenar números ímpares. Pedir para o usuário digitar 10 números, armazenando-os nos vetores criados, conforme a sua paridade. Mostrar os vetores obtidos. Note que esses podem não estar totalmente preenchidos.

2. Pedir para o usuário preencher um vetor com 10 posições, mostrar os elementos que são múltiplos de 3 e as suas respectivas posições.

3. Pedir para o usuário preencher dois vetores com 5 posições cada um. Criar um vetor com 10 posições, intercalando os elementos dos dois vetores anteriores. Mostrar os três vetores.

4. Criar um vetor com 10 posições, pedir para o usuário digitar uma lista de números, encerrando ou quando o vetor estiver preenchido ou quando o usuário digitar zero. Mostrar o vetor e dizer quantos elementos ele possui.

5. Fazer um programa em C que lê dois vetores de 10 posições e faz a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostrar o vetor resultante.

6. Pedir para o usuário preencher um vetor de inteiros com 10 posições. Substituir os pares por 0 e os ímpares por 1. Mostrar o vetor obtido.

7. Desenvolver um programa que, no momento de preencher um vetor com oito números inteiros que o usuário digita, já os armazena de forma ordenada crescente. Mostrar o vetor resultante.

Funções

1. Simule o funcionamento (no papel) do seguinte programa. Quais valores são impressos na saída padrão (tela)?

```c
#include <stdio.h>
#include <stdlib.h>
void f1(int a){
   printf ("%d", a);
   a += a;
   printf ("%d", a);
}

int main(int argc, char **argv) {
   int a = 2;
   int b = 3;
   f1 (a);
   f1 (b);
   printf ("%d %d", a, b);
   return 0;
}
```

2) Simule o funcionamento (no papel) do seguinte programa. Quais valores são impressos na saída padrão?

```c
#include <stdio.h>
#include <stdlib.h>
void f2 (int a[]){
   printf ("%d", a[1]);
   a[0]++;
   printf ("%d", a[0]);
}
```

```
int main(int argc, char **argv) {
   int x[] = {3,2};
   f2(x);
   printf ("%d %d", x[0], x[1]);
   return 0;
}
```

3. Simule o funcionamento (no papel) do seguinte programa. Quais valores são impressos na saída padrão?

```
#include <stdio.h>
#include <stdlib.h>
int f3(int a [], int b){
	int i;
	for (i = 0; i < b; i++){
		if (a[i] > 4){
			a [i] += 2;
		}
		else{
			a[i] -= 2;
		}
		a[i]++;
	}
	b++;
	return a[0] + a[1];
}

int main(int argc, char **argv) {
	int x [] = {5, 6, 4, 1};
	int b = 4;
	int resultado = f3(x, b);
	printf ("%d %d %d %d %d %d %d", x[0], x[1], x[2], x[3], x[4], b, resultado);
	return 0;
}
```

4) Escrever uma função que recebe um valor real como parâmetro que representa uma temperatura em graus Fahrenheit e devolve este valor convertido para graus Celsius. Escrever outra função que recebe um valor real como parâmetro que representa uma temperatura em graus Celsius e devolve este valor convertido para graus Fahrenheit. Na função main, escrever um loop que permite ao usuário escolher que tipo de conversão deseja fazer. Este loop termina quando o usuário digitar a opção -1. Caso o usuário digite 1, fazer a conversão de Celsius para Fahrenheit, usando a função escrita, e mostrar o resultado. Caso o usuário digite 2, fazer a conversão de Fahrenheit para Celsius, usando a função escrita, e mostrar o resultado. Somente a função main pode fazer chamadas às funções printf e scanf.

As fórmulas para conversão são:
°F = °C × 1.8 + 32
°C = (°F − 32) / 1.8

5. No cabeçalho de função <stdlib.h> existe uma função chamada rand. Esta função permite a geração de números pseudoaleatórios entre 0 e RAND_MAX, que é um número inteiro definido dentro do próprio cabeçalho que, para muitas implementações do compilador é 32768. Ou seja, a cada vez que seu programa chamar a função rand, que não recebe parâmetro algum, ela devolverá um número entre 0 e 32768. É natural a necessidade de números aleatórios para implementar um jogo, por exemplo. Mas muitas vezes é preciso limitar o intervalo dos números que são gerados. Por exemplo, se você quiser escrever um simulador de lançamento de dados, provavelmente vai querer utilizar números aleatórios

entre 1 e 6, para representar cada face de um dado. Com algumas manipulações matemáticas, é possível escolher este intervalo. Considere a seguinte função, ainda sem implementação:

```
int gera_aleatorio (int n){

}
```

5.1 Implemente a função gera_aleatorio. Ela deve devolver um número inteiro aleatório entre 0 e n-1. Dica: Considere o operador módulo.

5.2 Implemente a segunda versão de gera_aleatorio a seguir. Ela deve devolver um número inteiro entre primeiro e ultimo.

```
int gera_aleatorio2 (int primeiro, int ultimo){

}
```

6. Utilizando o conhecimento adquirido na questão 7, resolva os seguintes exercícios.

6.1 Implemente um simulador de lançamento de dados que faça o lançamento de um dados 6000 vezes. Imprima uma tabela que mostra o número do lançamento ao lado da face obtida. As faces são geradas randomicamente e variam entre 1 e 6. O simulador deve ser implementado por uma função chamada lança_dado.

6.2 Implemente um simulador que faça o lançamento de um dado e de uma moeda 10000 vezes. O simulador deve ser implementado por uma função chamada lança_dado_e_moeda. As faces do dado podem variar de 1 a 6 e os números representando cara ou coroa são 0 ou 1 respectivamente. Imprima uma tabela que mostra o número do lançamento seguido da face do dado obtida seguido de cara ou coroa.

7. Considere o seguinte programa:

```
void f1 (){
      f2();
}

void f2(){
      f1();
}

int main(int argc, char **argv) {
      f1();
}
```

Estude o que este programa faz. Sem executá-lo no computador, explique o problema existente.

Structs

1. Considere a estrutura definida a seguir, que representa um ponto no plano cartesiano:

```
struct ponto {
   float x;
   float y;
};
```

A partir dessa estrutura é possível definir um círculo com um ponto central e um raio.

a. definir a estrutura círculo como descrito acima
b. criar uma função que recebe 2 pontos e devolve a distância entre eles $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
c. criar uma função que recebe um círculo e um ponto e devolve se o ponto pertence ou não ao círculo
d. criar uma aplicação para testar as suas funções

2. Agora, considere a estrutura definida a seguir, que representa um número racional ou fração:

```
struct fracao {
   int numerador;
   int denominador;
};
```

a. criar uma função para ler uma fração
b. criar uma função para exibir uma fração (exemplo: 3/4)
c. criar uma função que recebe 2 frações e devolve qual a maior (a primeira ou a segunda)
d. criar uma função que recebe 2 frações e devolve a multiplicação entre elas
e. criar uma aplicação para testar as suas funções