```
175     }
176
177     printf( "Modificando os dados do array (dentro da funcao)...\n" );
178     for ( int i = 0; i < n; i++ ) {
179         a[i] += 2;
180     }
181
182     printf( "Dados do array apos modificacao (dentro da funcao):\n" );
183     for ( int i = 0; i < n; i++ ) {
184         printf( "a[%d] = %d\n", i, a[i] );
185     }
186
187 }
188
189 /*
190  * implementação da função imprimirNumeros
191  */
192 void imprimirNumeros() {
193
194     for ( int i = 0; i <= 10; i++ ) {
195         printf( "%d ", i );
196     }
197
198     pularLinha();
199
200 }
```

7.2 Exercícios

Exercício 7.1: Escreva um programa que leia 5 valores inteiros e imprima para cada um o seu valor absoluto. Para obter o valor absoluto do número utilize a função "absoluto", especificada abaixo:

- **Nome:** `absoluto`
- **Descrição:** Calcula o valor absoluto do número fornecido.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** O valor absoluto de n (`int`).

Arquivo com a solução: ex7.1.c

Entrada

```
n0: 5
n1: 6
n2: -7
n3: 8
n4: 9
```

Saída

```
absoluto(5) = 5
absoluto(6) = 6
absoluto(-7) = 7
absoluto(8) = 8
absoluto(9) = 9
```

Exercício 7.2: Escreva um programa que leia o valor do raio de um círculo. O programa deve calcular e imprimir a área e o perímetro do círculo representado por esse raio. Para obter o valor da área do círculo o programa deverá chamar a função "areaCirculo" e para obter o valor do seu perímetro o programa deverá invocar a função "circunferenciaCirculo". Para o valor de π, use o dialeto indicado no Capítulo sobre a biblioteca matemática padrão.

- **Nome:** `areaCirculo`
- **Descrição:** Calcula a área do círculo representado pelo raio fornecido.
- **Entrada/Parâmetro(s):** `float raio`
- **Saída/Retorno:** A área do círculo (`float`).

- **Nome:** `circunferenciaCirculo`
- **Descrição:** Calcula a circunferência do círculo representado pelo raio fornecido.
- **Entrada/Parâmetro(s):** `float raio`
- **Saída/Retorno:** A circunferência do círculo (`float`).

Arquivo com a solução: ex7.2.c

Entrada

```
Raio: 5
```

Saída

```
Area = 78.54
Circunferencia = 31.42
```

Exercício 7.3: Escreva um programa que leia 5 pares de valores decimais. Todos os valores lidos devem ser positivos. Caso um valor menor ou igual a zero for fornecido, esse valor deve ser lido novamente. Para cada par lido deve ser impresso o valor do maior elemento do par ou a frase "Eles sao iguais" se os valores do par forem iguais. Para obter o maior elemento do par utilize a função "maiorNumero".

- **Nome:** `maiorNumero`
- **Descrição:** Calcula o maior valor entre os dois valores.
- **Entrada/Parâmetro(s):** `float n1, float n2`
- **Saída/Retorno:** Retorna o maior valor os dois fornecidos ou -1 caso sejam iguais (`int`).
- **Observação:** Considere que os valores de entrada serão sempre positivos.

Arquivo com a solução: ex7.3.c

Entrada

```
n1[0]: 2
n2[0]: 3
n1[1]: 4
n2[1]: 6
n1[2]: 5
n2[2]: 5
n1[3]: -6
Entre com um valor positivo!
n1[3]: 4
n2[3]: -7
Entre com um valor positivo!
n2[3]: -8
Entre com um valor positivo!
n2[3]: 3
n1[4]: 4
n2[4]: 2
```

Saída

```
2.00, 3.00: O maior valor e 3.00
4.00, 6.00: O maior valor e 6.00
5.00, 5.00: Eles sao iguais
4.00, 3.00: O maior valor e 4.00
4.00, 2.00: O maior valor e 4.00
```

Exercício 7.4: Escreva um programa que leia 5 números inteiros positivos, utilizando, para isso, a função "lePositivo". Para cada valor lido escrever o somatório dos inteiros de 1 ao número informado. O resultado do cálculo desse somatório deve ser obtido através da função "somatorio".

- **Nome:** `lePositivo`
- **Descrição:** Faz a leitura de um valor. Se ele for negativo ou zero, a leitura deve ser repetida até que o valor lido seja positivo.
- **Entrada/Parâmetro(s):** nenhum.
- **Saída/Retorno:** Retorna o valor lido (`int`).

- **Nome:** `somatorio`
- **Descrição:** Calcula o somatório dos inteiros de 1 ao número fornecido como parâmetro.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** O valor do somatório (`int`).

Arquivo com a solução: ex7.4.c

Entrada

```
n[0]: 5
n[1]: 4
n[2]: 9
n[3]: -7
Entre com um valor positivo: 8
n[4]: -8
Entre com um valor positivo: -9
Entre com um valor positivo: -4
Entre com um valor positivo: 3
```

Saída

```
Somatorio de 1 a 5: 15
Somatorio de 1 a 4: 10
Somatorio de 1 a 9: 45
Somatorio de 1 a 8: 36
Somatorio de 1 a 3: 6
```

Exercício 7.5: Escreva um programa que leia dois números 5 vezes. O programa deve verificar se o primeiro número fornecido é par e se o primeiro número é divisível pelo segundo, ou seja, se o resto da divisão do primeiro pelo segundo é zero. Para fazer tais verificações, utilize os métodos estáticos "ehPar" e "ehDivisivel".

- **Nome:** `ehPar`
- **Descrição:** Verifica se o número fornecido é ou não par.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** `true` caso o número seja par, `false` caso contrário.

- **Nome:** `ehDivisivel`
- **Descrição:** Verifica se um número é divisível por outro.
- **Entrada/Parâmetro(s):** `int dividendo, int divisor`
- **Saída/Retorno:** `true` caso o dividendo seja divisível pelo divisor, `false` caso contrário.

Arquivo com a solução: ex7.5.c

Entrada

```
n1[0]: 8
n2[0]: 4
n1[1]: 7
n2[1]: 3
n1[2]: 21
n2[2]: 7
n1[3]: 9
n2[3]: 5
n1[4]: 10
n2[4]: 5
```

Saída

```
8 eh par e 8 eh divisivel por 4
7 eh impar e 7 nao eh divisivel por 3
21 eh impar e 21 eh divisivel por 7
9 eh impar e 9 nao eh divisivel por 5
10 eh par e 10 eh divisivel por 5
```

Exercício 7.6: Escreva um programa que leia 5 números inteiros positivos (utilizar "lePositivo"). Para cada número informado escrever a soma de seus divisores (exceto ele mesmo). Utilize a função "somaDivisores" para obter a soma.

- **Nome:** `somaDivisores`
- **Descrição:** Calcula a soma dos divisores do número informado, exceto ele mesmo.

- **Exemplo:** Para o valor 8, tem-se que $1+2+4=7$

- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** A soma dos divisores do número fornecido.

Arquivo com a solução: ex7.6.c

Entrada

```
n[0]: 8
n[1]: 10
n[2]: 5
n[3]: -8
Entre com um valor positivo: 9
n[4]: -7
Entre com um valor positivo: -8
Entre com um valor positivo: -7
Entre com um valor positivo: 50
```

Saída

```
Soma dos divisores de 8: 7
Soma dos divisores de 10: 8
Soma dos divisores de 5: 1
Soma dos divisores de 9: 4
Soma dos divisores de 50: 43
```

Exercício 7.7: Escreva um programa que imprima na tela os números primos existentes entre 1, inclusive, e 20, inclusive. Para verificar se um número é primo utilize a função "ehPrimo".

- **Nome:** `ehPrimo`
- **Descrição:** Verifica se um número é ou não primo.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** `true` caso o número seja primo, `false` caso contrário.

Arquivo com a solução: ex7.7.c

Saída

```
1: nao eh primo
2: eh primo
3: eh primo
4: nao eh primo
5: eh primo
6: nao eh primo
7: eh primo
8: nao eh primo
9: nao eh primo
10: nao eh primo
11: eh primo
12: nao eh primo
13: eh primo
14: nao eh primo
15: nao eh primo
16: nao eh primo
17: eh primo
18: nao eh primo
19: eh primo
20: nao eh primo
```

Exercício 7.8: Escreva um programa que leia 5 pares de valores positivos ("lePositivo"). Imprima se os elementos de cada par são números amigos ou não. Dois números "a" e "b" são amigos se a soma dos divisores de "a" excluindo "a" é igual a "b" e a soma dos divisores de "b" excluindo "b" é igual a "a". Para verificar se dois números são amigos utilize a função "saoAmigos".

- **Nome:** `saoAmigos`
- **Descrição:** Verifica se dois números são amigos.

- **Observação:** Utilize a função "somaDividores" do exercício anterior.
- **Exemplo:** 220 e 284 são amigos, pois:
  - **220:** $1+2+4+5+10+11+20+22+44+55+110=284$
  - **284:** $1+2+4+71+142=220$
- **Entrada/Parâmetro(s):** `int n1, int n2`
- **Saída/Retorno:** `true` caso os números sejam amigos, `false` caso contrário.

Arquivo com a solução: ex7.8.c

Entrada

```
n1[0]: 220
n2[0]: 284
n1[1]: 128
n2[1]: 752
n1[2]: 789
n2[2]: 568
n1[3]: 1184
n2[3]: 1210
n1[4]: 874
n2[4]: 138
```

Saída

```
220 e 284 sao amigos
128 e 752 nao sao amigos
789 e 568 nao sao amigos
1184 e 1210 sao amigos
874 e 138 nao sao amigos
```

Exercício 7.9: Escreva um programa que leia as medidas dos lados de 5 triângulos. Para cada triângulo imprimir a sua classificação (Não é triângulo, Triângulo Equilátero, Isósceles ou Escaleno). O programa deve aceitar apenas valores positivos para as medidas dos lados (utilizar "lePositivo"). Para verificar se as medidas formam um triângulo chamar a função "ehTriangulo". Para obter o código da classificação utilize a função "tipoTriangulo".

- **Nome:** `ehTriangulo`
- **Descrição:** Verifica se as 3 medidas informadas permitem formar um triângulo. Essa condição de existência já foi apresentada em um Capítulo anterior.
- **Entrada/Parâmetro(s):** `int ladoA, int ladoB, int ladoC`
- **Saída/Retorno:** `true` caso os valores representam um triângulo, `false` caso contrário.

- **Nome:** `tipoTriangulo`
- **Descrição:** A partir das medidas dos lados de um triângulo, verifica o tipo do triângulo.
- **Entrada/Parâmetro(s):** `int ladoA, int ladoB, int ladoC`
- **Saída/Retorno:** Um inteiro, sendo que:
  - **0:** se não formam um triângulo;
  - **1:** se for um triângulo equilátero;

- **2:** se for um triângulo isósceles;
- **3:** se for um triângulo escaleno.

Arquivo com a solução: ex7.9.c

Entrada

```
ladoA[0]: 2
ladoB[0]: 2
ladoC[0]: 2
ladoA[1]: 2
ladoB[1]: 3
ladoC[1]: -10
Entre com um valor positivo: -5
Entre com um valor positivo: 5
ladoA[2]: 3
ladoB[2]: 4
ladoC[2]: 5
ladoA[3]: 7
ladoB[3]: 7
ladoC[3]: -15
Entre com um valor positivo: -19
Entre com um valor positivo: 8
ladoA[4]: 1
ladoB[4]: 10
ladoC[4]: 20
```

Saída

```
Valores 2, 2 e 2: triangulo equilatero
Valores 2, 3 e 5: nao formam um triangulo
Valores 3, 4 e 5: triangulo escaleno
Valores 7, 7 e 8: triangulo isosceles
Valores 1, 10 e 20: nao formam um triangulo
```

Exercício 7.10: Escreva um programa que leia um valor inteiro de 1 a 9999. Para cada número imprima o seu correspondente dígito verificador. O programa é encerrado ao ser fornecido um número fora da faixa estabelecida. Para obter o valor do dígito verificador utilize a função "calculaDigito".

- **Nome:** `calculaDigito`
- **Descrição:** Calcula o dígito verificador de um número. Para evitar erros de digi-

tação em números de grande importância, como código de uma conta bancária, geralmente se adiciona ao número um dígito verificador. Por exemplo, o número 1841 é utilizado normalmente como 18414, onde o 4 é o dígito verificador. Ele é calculado da seguinte forma:
  - **a)** Cada algarismo do número é multiplicado por um peso começando em 2, da direita para a esquerda. Para cada algarismo o peso é acrescido de 1. Soma-se então os produtos obtidos. Exemplo: $1*5+8*4+4*3+1*2=51$
  - **b)** Calcula-se o resto da divisão desta soma por 11: $51\%11=7$
  - **c)** Subtrai-se de 11 o resto obtido: $11-7=4$
  - **d)** Se o valor obtido for 10 ou 11, o dígito verificador será o 0, nos outros casos, o dígito verificador é o próprio valor encontrado.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** O dígito verificador do número (`int`).

Arquivo com a solução: ex7.10.c

Entrada
```
Numero: 1841
```

Saída
```
Digito verificador de 1841: 4
```

Entrada
```
Numero: 6857
```

Saída
```
Digito verificador de 6857: 8
```

Entrada
```
Numero: 751
```

Saída
```
Digito verificador de 751: 0
```

Exercício 7.11: Escreva um programa que leia um valor inteiro de 10 a 99999, onde o último algarismo representa o seu dígito verificador. Imprima uma mensagem

indicando se ele foi digitado corretamente ou não. O programa é encerrado ao ser fornecido um número fora da faixa estabelecida. Utilize a função "numeroCorreto" para verificar se o número está correto.

- **Nome:** `numeroCorreto`
- **Descrição:** Verifica se um número, em conjunto com seu dígito, está correto.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** `true` se o número está correto, `false` caso contrário.
- **Observação:** Use as funções abaixo: "obtemNumero", "obtemDigito" e "calculaDigito".

- **Nome:** `obtemDigito`
- **Descrição:** Separa o dígito verificador (a unidade) do número.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** O último algarismo do número (`int`).
- **Exemplo:** Para o valor 1823, o dígito é 3.

- **Nome:** `obtemNumero`
- **Descrição:** Separa o número do dígito verificador.
- **Entrada/Parâmetro(s):** `int n`
- **Saída/Retorno:** O número sem o valor da unidade (`int`).
- **Exemplo:** Para o valor 1823, o número é 182.

Arquivo com a solução: ex7.11.c

Entrada

```
Numero: 18414
```

Saída

```
Numero completo: 18414
Numero: 1841
Digito: 4
Digito calculado: 4
O numero fornecido esta correto!
```

Entrada

```
Numero: 68577
```

Saída

```
Numero completo: 68577
Numero: 6857
Digito: 7
Digito calculado: 8
O numero fornecido esta incorreto!
```

Entrada

```
Numero: 7510
```

Saída

```
Numero completo: 7510
Numero: 751
Digito: 0
Digito calculado: 0
O numero fornecido esta correto!
```

Exercício 7.12: Escreva um programa que leia 3 duplas de valores inteiros. Exibir cada dupla em ordem crescente. A ordem deve ser impressa através da chamada da função "classificaDupla" especificada abaixo:

- **Nome:** `classificaDupla`
- **Descrição:** Imprime em ordem crescente dois valores inteiros.
- **Entrada/Parâmetro(s):** `int n1, int n2`
- **Saída/Retorno:** nenhum.

Arquivo com a solução: ex7.12.c

Entrada

```
n1[0]: 7
n2[0]: 9
n1[1]: 10
n2[1]: 5
n1[2]: 2
n2[2]: 2
```

Saída

```
7 e 9: 7 <= 9
10 e 5: 5 <= 10
2 e 2: 2 <= 2
```

Exercício 7.13: Escreva um programa que leia 3 trincas de valores inteiros. Exibir cada trinca em ordem crescente. A ordem deve ser impressa através da chamada da função "classificaDupla" especificada abaixo:

- **Nome:** `classificaTrinca`
- **Descrição:** Imprime em ordem crescente três valores inteiros.
- **Entrada/Parâmetro(s):** `int n1, int n2, int n3`
- **Saída/Retorno:** nenhum.

Arquivo com a solução: ex7.13.c

Entrada

```
n1[0]: 9
n2[0]: 5
n2[0]: 1
n1[1]: 8
n2[1]: 7
n2[1]: 6
n1[2]: 3
n2[2]: 3
n2[2]: 3
```

Saída

```
9, 5 e 1: 1 <= 5 <= 9
8, 7 e 6: 6 <= 7 <= 8
3, 3 e 3: 3 <= 3 <= 3
```

Exercício 7.14: Escreva um programa que leia 5 duplas de valores inteiros. Após a leitura de todos os elementos, imprimir as duplas que foram armazenadas nas posições pares em ordem crescente e aquelas armazenadas nas posições ímpares em ordem decrescente. Utilize a função "imprimeDuplaClassificada" especificada abaixo para escrever os elementos na ordem desejada.

- **Nome:** `imprimeDuplaClassificada`

- **Descrição:** Imprime os dois inteiros fornecidos na ordem desejada. A ordem é especificada através do parâmetro “emOrdemCrescente”.
- **Entrada/Parâmetro(s):** `int n1, int n2, bool emOrdemCrescente`
- **Saída/Retorno:** nenhuma.

Arquivo com a solução: ex7.14.c

Entrada

```
n1[0]: 7
n2[0]: 9
n1[1]: 9
n2[1]: 7
n1[2]: 8
n2[2]: 2
n1[3]: 6
n2[3]: 4
n1[4]: 9
n2[4]: 9
```

Saída

```
7 e 9: 7 <= 9
9 e 7: 9 >= 7
8 e 2: 2 <= 8
6 e 4: 6 >= 4
9 e 9: 9 <= 9
```