

5.1 Estrutura de repetição em algoritmo

Uma estrutura de repetição é utilizada quando um trecho do algoritmo, ou até mesmo o algoritmo inteiro, precisa ser repetido. O número de repetições pode ser fixo ou estar atrelado a uma condição. Assim, existem estruturas para tais situações, descritas a seguir.

5.1.1 Estrutura de repetição para número definido de repetições (estrutura PARA)

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que um trecho do algoritmo deve ser repetido. O formato geral dessa estrutura é:

```
PARA I ← valor_inicial ATÉ valor_final FAÇA [PASSO n]
INÍCIO
comando1
comando2
...
comandom
FIM
```

O comando1, o comando2 e o comandom serão executados utilizando-se a variável *I* como controle, e seu conteúdo vai variar do *valor_inicial* até o *valor_final*. A informação do *PASSO* está entre colchetes porque é opcional. O *PASSO* indica como será a variação da variável de controle. Por exemplo, quando for indicado *PASSO 2*, a variável de controle será aumentada em 2 unidades a cada iteração até atingir o *valor_final*. Quando a informação do *PASSO* for suprimida, isso significa que o incremento ou o decremento da variável de controle será de 1 unidade.

Quando houver apenas um comando a ser repetido, os marcadores de bloco *INÍCIO* e *FIM* poderão ser suprimidos.

Exemplos:

```
PARA I ← 1 ATÉ 10 FAÇA
  ESCRIVA I
```

O comando *ESCREVA I* será executado dez vezes, ou seja, para *I* variando de 1 a 10. Assim, os valores de *I* serão: 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10.

```
PARA J ← 1 ATÉ 9 FAÇA PASSO 2
  ESCRIVA J
```

O comando *ESCREVA J* será executado cinco vezes, ou seja, para *J* variando de 1 a 9, de 2 em 2. Assim, os valores de *J* serão: 1, 3, 5, 7 e 9.

```
PARA I ← 10 ATÉ 5 FAÇA PASSO -1
  ESCRIVA I
```

O comando `ESCREVA I` será executado seis vezes, ou seja, para i variando de 10 a 5. Assim, os valores de serão: 10, 9, 8, 7, 6 e 5.

```
PARA J ← 15 ATÉ 1 FAÇA PASSO -2
ESCREVA J
```

O comando `ESCREVA J` será executado oito vezes, ou seja, para J variando de 15 a 1, de 2 em 2. Assim, os valores de J serão: 15, 13, 11, 9, 7, 5, 3 e 1.

Existem duas instruções comumente usadas nos comandos internos das estruturas de repetição. São as instruções denominadas acumuladores e contadores.

Os acumuladores devem ser usados quando a realização de um cálculo precisa de valores obtidos a cada iteração, ou seja, o cálculo só estará pronto com a conclusão da repetição. É por isso que um acumulador deve ser inicializado com um valor neutro para a operação em que será utilizado. Por exemplo, se for usado em uma adição, deve ser inicializado com zero; se for usado em uma multiplicação, deve ser inicializado com 1.

Exemplo de acumulador:

```
SOMA ← 0           // inicialização da variável SOMA com o valor zero
PARA I ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCRVA "Digite um número: "
  LEIA NUM
  SOMA ← SOMA + NUM // acumulando o valor da variável NUM na variável SOMA
FIM
ESCREVA "Soma = ",SOMA
```

Simulação:

MEMÓRIA			TELA	
I	NUM	SOMA		
		0	Inicialização da variável SOMA com o valor zero $SOMA \leftarrow 0$	
1				Digite um número: 5
1	5			
1	5	5	Acumulando o valor da variável NUM na variável SOMA $SOMA \leftarrow SOMA + NUM$	
2				Digite um número: 3
2	3			
2	3	8	Acumulando o valor da variável NUM na variável SOMA $SOMA \leftarrow SOMA + NUM$	
3				Digite um número: 0
3	0			
3	0	8	Acumulando o valor da variável NUM na variável SOMA $SOMA \leftarrow SOMA + NUM$	
4				Digite um número: 10
4	10			
4	10	18	Acumulando o valor da variável NUM na variável SOMA $SOMA \leftarrow SOMA + NUM$	
5				Digite um número: 2
5	2			
5	2	20	Acumulando o valor da variável NUM na variável SOMA $SOMA \leftarrow SOMA + NUM$	
				Soma = 20

Exemplo de contador:

```

CONT ← 0           // inicialização da variável CONT com o valor zero
PARA I ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite um número: "
  LEIA NUM
  SE (NUM > 5)
    ENTÃO CONT ← CONT + 1 // contando mais 1 na variável CONT
FIM
ESCREVA "Quantidade de número maiores que 5 = ",CONT

```

Simulação:

MEMÓRIA			TELA
I	NUM	CONT	
		0	Inicialização da variável CONT com o valor zero CONT ← 0
1			Digite um número: 5
1	5		
1	5	0	O número digitado não é maior que 5, logo, o contador CONT não será alterado
2			Digite um número: 12
2	12		
2	12	1	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT ← CONT + 1
3			Digite um número: 8
3	8		
3	8	2	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT ← CONT + 1
4			Digite um número: 3
4	3		
4	3	2	O número digitado não é maior que 5, logo, o contador CONT não será alterado
5			Digite um número: 6
5	6		
5	6	3	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT ← CONT + 1
			Quantidade de números maiores que 5 = 3

5.1.2 Estrutura de repetição para número indefinido de repetições e teste no início (estrutura ENQUANTO)

Essa estrutura de repetição é utilizada quando não se sabe o número de vezes que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se conhece esse número.

Essa estrutura baseia-se na análise de uma condição. A repetição será feita enquanto a condição se mostrar verdadeira.

Existem situações em que o teste condicional da estrutura de repetição, que fica no início, resulta em um valor falso logo na primeira comparação. Nesses casos, os comandos escritos dentro da estrutura de repetição não serão executados.

```
ENQUANTO condição FAÇA
comando1
```

Enquanto a condição for verdadeira, o comando1 será executado.

```
ENQUANTO condição FAÇA
INÍCIO
    comando1
    comando2
    comando3
FIM
```

Enquanto a condição for verdadeira, o comando1, o comando2 e o comando3 serão executados.

Exemplos:

```
X ← 1           // inicialização da variável X com o valor 1
Y ← 5           // inicialização da variável Y com o valor 5
ENQUANTO X < Y FAÇA
INÍCIO
X ← X + 2       // contador incrementado em 2 unidades
Y ← Y + 1       // contador incrementado em 1 unidade
FIM
```

Simulação:

X	Y	
1	5	Valores iniciais
3	6	Valores obtidos dentro da estrutura de repetição
5	7	
7	8	
9	9	

No trecho do algoritmo anterior, portanto, os comandos que estão dentro da estrutura de repetição são repetidos quatro vezes.

```
X ← 1 // inicialização da variável X com o valor 1
Y ← 1 // inicialização da variável Y com o valor 1
ENQUANTO X <= 5 FAÇA
INÍCIO
Y ← Y * X // acumulador das multiplicações
X ← X + 1 // contador incrementado em 1 unidade
FIM
```

Simulação:

Y	X	
1	1	Valores iniciais
1	2	Valores obtidos dentro da estrutura de repetição
2	3	
6	4	
24	5	
120	6	

No trecho do algoritmo anterior, portanto, os comandos que se localizam na estrutura de repetição são repetidos cinco vezes. Nesse exemplo, a estrutura **ENQUANTO** é utilizada para repetir o trecho do algoritmo em um número definido de vezes.

5.1.3 Estrutura de repetição para número indefinido de repetições e teste no final (estrutura REPITA)

Essa estrutura de repetição é utilizada quando *não* se sabe o número de vezes que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se conhece esse número.

Essa estrutura baseia-se na análise de uma condição. A repetição será feita até a condição se tornar verdadeira.

A diferença entre a estrutura **ENQUANTO** e a estrutura **REPITA** é que, nessa última, os comandos serão repetidos pelo menos uma vez, já que a condição de parada se encontra no final.

```
REPITA
comandos
ATÉ condição
```

Repita os comandos até a condição se tornar verdadeira.

Exemplos:

```
X ← 1           // inicialização da variável X com o valor 1
Y ← 5           // inicialização da variável Y com o valor 5
REPITA
X ← X + 2       // contador incrementado em 2 unidades
Y ← Y + 1       // contador incrementado em 1 unidade
ATÉ X >= Y
```

Simulação:

X	Y	
1	5	Valores iniciais
3	6	Valores obtidos dentro da estrutura de repetição
5	7	
7	8	
9	9	

No trecho do algoritmo anterior, portanto, os comandos escritos dentro da estrutura de repetição são repetidos quatro vezes.

```
X ← 1           // inicialização da variável X com o valor 1
Y ← 1           // inicialização da variável Y com o valor 1
REPITA
Y ← Y * X       // acumulador das multiplicações
X ← X + 1       // contador incrementado em 1 unidade
ATÉ X = 6
```

Simulação:

Y	X	
1	1	Valores iniciais
1	2	Valores obtidos dentro da estrutura de repetição
2	3	
6	4	
24	5	
120	6	

No trecho do algoritmo anterior, portanto, os comandos que se localizam dentro da estrutura de repetição são repetidos cinco vezes. Nesse exemplo, a estrutura **REPITA** é utilizada para repetir o trecho do algoritmo em um número definido de vezes.

5.2 Estrutura de repetição em PASCAL

5.2.1 Estrutura de repetição FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que um trecho do programa deve ser repetido.

```
FOR I := valor_inicial TO valor_final DO
    comando;
```

O comando será executado utilizando-se a variável **I** como controle, e seu conteúdo vai variar do **valor_inicial** até o **valor_final**, de 1 em 1, incrementando automaticamente.

```
FOR J := valor_inicial TO valor_final DO
BEGIN
    comando1;
    comando2;
END;
```

O **comando1** e o **comando2** serão executados utilizando-se a variável **J** como controle, e seu conteúdo vai variar do **valor_inicial** até o **valor_final**, de 1 em 1, incrementando automaticamente.

```
FOR K := valor_inicial DOWNTO valor_final DO
    comando;
```

O comando será executado utilizando-se a variável **K** como controle, e seu conteúdo vai variar do **valor_inicial** até o **valor_final**, de 1 em 1, decrementando automaticamente.

```
FOR H := valor_inicial DOWNTO valor_final DO
BEGIN
    comando1;
    comando2;
    comando3;
END;
```

O **comando1**, o **comando2** e o **comando3** serão executados utilizando-se a variável **H** como controle, e seu conteúdo vai variar do **valor_inicial** até o **valor_final**, de 1 em 1, decrementando automaticamente.



Observações

Na linguagem PASCAL, a estrutura de repetição **FOR** funciona obrigatoriamente de 1 em 1, incrementando ou decrementando.

Exemplos:

```
FOR i := 1 TO 5 DO
    Writeln(i);
```

No trecho de programa anterior, o comando **Writeln(i);** será executado cinco vezes, ou seja, para **i** valendo 1, 2, 3, 4 e 5.

```
FOR i := 10 DOWNTO 1 DO
    Writeln(i);
```

No trecho de programa anterior, o comando **Writeln(i);** será executado dez vezes, ou seja, para **i** valendo 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

Existem duas instruções comumente usadas nos comandos internos das estruturas de repetição. São as instruções denominadas acumuladores e contadores.

Os acumuladores devem ser usados quando a realização de um cálculo precisa de valores obtidos a cada iteração, ou seja, o cálculo só estará pronto com a conclusão da repetição. É por isso que um acumulador deve ser inicializado com um valor neutro para a operação em que será utilizado. Por exemplo, se for usado em uma adição, deve ser inicializado com zero, e se for usado em uma multiplicação, deve ser inicializado com 1.

Exemplo de acumulador:

```
SOMA := 0;           // inicialização da variável SOMA com o valor zero
FOR I := 1 TO 5 DO
BEGIN
  WRITE('Digite um número: ');
  READLN(NUM);
  SOMA := SOMA + NUM; // acumulando o valor da variável NUM na variável SOMA
END;
WRITELN('Soma = ',SOMA);
```

Simulação:

MEMÓRIA			TELA
I	NUM	SOMA	
		0	Inicialização da variável SOMA com o valor zero SOMA := 0;
1			Digite um número: 5
1	5		
1	5	5	Acumulando o valor da variável NUM na variável SOMA SOMA := SOMA + NUM;
2			Digite um número: 3
2	3		
2	3	8	Acumulando o valor da variável NUM na variável SOMA SOMA := SOMA + NUM;
3			Digite um número: 0
3	0		
3	0	8	Acumulando o valor da variável NUM na variável SOMA SOMA := SOMA + NUM;
4			Digite um número: 10
4	10		
4	10	18	Acumulando o valor da variável NUM na variável SOMA SOMA := SOMA + NUM;
5			Digite um número: 2
5	2		
5	2	20	Acumulando o valor da variável NUM na variável SOMA SOMA := SOMA + NUM;
			Soma = 20

Exemplo de contador:

```
CONT := 0; // inicialização da variável CONT com o valor zero
FOR I := 1 TO 5 DO
BEGIN
```

```

WRITE('Digite um número: ');
READLN(NUM);
IF (NUM > 5)
THEN CONT := CONT + 1; // contando mais 1 na variável CONT
END;
Writeln('Quantidade de número maiores que 5 = ',CONT);

```

Simulação:

MEMÓRIA			TELA
I	NUM	CONT	
		0	Inicialização da variável CONT com o valor zero CONT := 0;
1			Digite um número: 5
1	5		
1	5	0	O número digitado não é maior que 5, logo, o contador CONT não será alterado
2			Digite um número: 12
2	12		
2	12	1	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT := CONT + 1;
3			Digite um número: 8
3	8		
3	8	2	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT := CONT + 1;
4			Digite um número: 3
4	3		
4	3	2	O número digitado não é maior que 5, logo, o contador CONT não será alterado
5			Digite um número: 6
5	6		
5	6	3	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT := CONT + 1;
			Quantidade de números maiores que 5 = 3

5.2.2 Estrutura de repetição WHILE

A estrutura de repetição **WHILE** é utilizada quando o número de repetições necessárias não for fixo, apesar de também ser utilizada quando se conhece a quantidade de repetições.

Nessa estrutura os comandos serão repetidos enquanto a condição for verdadeira e o teste condicional ocorre no início. Isso significa que existe a possibilidade da repetição não ser executada quando a condição assumir o valor falso logo na primeira verificação.

```

WHILE condição DO
comando;

```

Enquanto a condição for verdadeira, o comando será executado.

```

WHILE condição DO
BEGIN
    comando1;
    comando2;
END;

```

Enquanto a condição for verdadeira, o comando1 e o comando2 serão executados.

Exemplos:

```

X := 0;           // inicialização da variável X com o valor zero
WHILE X <> 5 DO
BEGIN
  WRITELN('Valor de X = ',X);
  X := X + 1;     // contador incrementado em 1 unidade
END;
WRITELN('Valor de X depois que sair da estrutura = ',X);

```

No trecho de programa anterior, os comandos `WRITELN('Valor de X = ',X)` e `X := X + 1;` foram executados cinco vezes. O teste condicional avalia `x` valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	Valor inicial
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição
Valor de X = 1	2	
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```

X := 1;           // inicialização da variável X com o valor 1
Y := 10;          // inicialização da variável Y com o valor 10
WHILE Y > X DO
BEGIN
  WRITELN('Valor de Y = ',Y);
  Y := Y - 2;     // contador decrementado em 2 unidades
END;
WRITELN('Valor de Y depois que sair da estrutura = ',Y);

```

No trecho de programa anterior, os comandos `WRITELN('Valor de Y = ',Y);` e `Y := Y - 2;` foram executados cinco vezes. O teste condicional avalia `y` valendo 10, 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	Valores iniciais
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

```

X := 1;           // inicialização da variável x com o valor 1
Y := 1;           // inicialização da variável y com o valor 1
WHILE X < Y DO
BEGIN
  WRITELN('Valor de X = ',X);

```

```
X := X + 1;    // contador incrementado em 1 unidade
END;
```

No trecho de programa anterior, os comandos `WRITELN('Valor de X = ',X);` e `X := X + 1;` não são executados, pois com os valores iniciais de `x` e `y` a condição é falsa, logo, não ocorre a entrada na estrutura de repetição para execução de seus comandos.

5.2.3 Estrutura de repetição REPEAT

A estrutura de repetição **REPEAT** é utilizada quando o número de repetições necessárias não for fixo, apesar de também ser utilizada quando se conhece o número de repetições.

Nessa estrutura, os comandos serão repetidos até a condição se tornar verdadeira, e o teste condicional ocorre no final, o que significa que os comandos internos da estrutura da repetição serão executados, no mínimo, uma vez.

```
REPEAT
comandos;
UNTIL condição;
```

Os comandos serão repetidos até que a condição se torne verdadeira.

Exemplos:

```
X := 0;          // inicialização da variável X com o valor 0
REPEAT
WRITELN('Valor de X = ',X);
X := X + 1;      // contador incrementado em 1 unidade
UNTIL X = 5;
WRITELN('Valor de X depois que sair da estrutura = ',X);
```

No trecho de programa anterior, os comandos `WRITELN('Valor de X = ',X);` e `X := X + 1;` foram executados cinco vezes. O teste condicional avalia `x` valendo 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	Valor inicial
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição
Valor de X = 1	2	
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

No trecho do algoritmo anterior, portanto, os comandos que se localizam na estrutura de repetição são repetidos cinco vezes. Nesse exemplo, a estrutura **REPITA** é utilizada para repetir o trecho do algoritmo em um número definido de vezes.

```
X := 1;          // inicialização da variável X com o valor 1
Y := 10;         // inicialização da variável Y com o valor 10
REPEAT
WRITELN('Valor de Y = ',Y);
Y := Y - 2;      // contador decrementado em 2 unidades
UNTIL Y <= X;
WRITELN('Valor de Y depois que sair da estrutura = ',Y);
```

No trecho de programa anterior, os comandos `WRITELN('Valor de Y = ',Y);` e `Y := Y - 2;` serão executados cinco vezes. O teste condicional avaliará `y` valendo 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	Valores iniciais
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

5.3 Estrutura de repetição em C/C++

5.3.1 Estrutura de repetição FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que um trecho do programa deve ser repetido.

O formato geral do comando `for` é composto por três partes:

```
for (i = valor_inicial; condição; incremento ou decremento de i)
    comando;
```

A primeira parte atribui um `valor_inicial` à variável `i`, que tem como função controlar o número necessário de repetições.

A segunda parte corresponde a uma expressão relacional que, quando assumir o valor falso, determinará o fim da repetição.

A terceira parte é responsável por alterar o valor da variável `i` (incremento ou decremento) com o objetivo de, em algum momento, fazer a condição assumir o valor falso.

Caso seja necessária a repetição de apenas um comando, o compilador entenderá que a estrutura de repetição terminará quando for encontrado o primeiro `;` (ponto e vírgula).

Exemplo:

```
for (a = 1; a <= 20; a++)
    printf("\no valor de a é: %d", a);
```

No exemplo anterior, à variável `a` é atribuído inicialmente o valor 1 (`a = 1`) que, depois, é incrementado em uma unidade (`a++`).

A cada incremento, o comando `printf` será executado. Esse processo se repete até o valor da variável `a` se tornar maior que 20 (quando a condição `a <= 20` assumir o valor falso).

Se for necessária a repetição de mais de um comando, o compilador entenderá que a estrutura de repetição começará quando for encontrado o símbolo `{` e terminará quando for encontrado o símbolo `}`.

Exemplo:

```
for (a = 15; a >= 1; a = a-2)
{
    printf("Digite um número: ");
    scanf("%d%c", &x);
}
```

No exemplo anterior, a variável `a` é inicializada com o valor 15 (`a = 15`) que, depois, é decrementada em duas unidades (`a = a - 2`).

A cada decremento, o bloco de comando que está entre chaves `{ ... }` é executado. Esse processo se repete até o valor da variável `a` se tornar menor que 1 (quando a condição `a >= 1` assumir o valor falso).

Exemplos:

```
for (i = 1; i <= 5; i++)
    printf("%d", i);
ou
for (i = 1; i <= 5; i = i+1)
    printf("%d", i);
```

Nos trechos de programa anteriores, que, apesar de utilizarem formas diferentes para aumentar o valor da variável *i*, expressam a mesma coisa, o comando `printf("%d", i);` foi executado cinco vezes, ou seja, para *i* valendo 1, 2, 3, 4 e 5.

```
for (i = 10; i >= 1; i--)
    printf("%d", i);
ou
for (i = 10; i >= 1; i = i-1)
    printf("%d", i);
```

Nos trechos de programa anterior, que, apesar de utilizarem formas diferentes para diminuir o valor da variável *i*, são exatamente a mesma coisa, o comando `printf("%d", i);` será executado dez vezes, ou seja, para *i* valendo 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

```
for (i = 0; i <= 10; i = i+2)
    printf("%d", i);
```

No trecho de programa anterior, o comando `printf("%d", i);` será executado seis vezes, ou seja, para *i* valendo 0, 2, 4, 6, 8 e 10.

```
for (i = 100; i >= 0; i = i-20)
    printf("%d", i);
```

No trecho de programa anterior, o comando `printf("%d", i);` será executado seis vezes, ou seja, para *i* valendo 100, 80, 60, 40, 20 e 0.

Existem duas instruções comumente usadas nos comandos internos das estruturas de repetição. São as instruções denominadas acumuladores e contadores.

Os acumuladores devem ser usados quando a realização de um cálculo precisa de valores obtidos a cada iteração, ou seja, o cálculo só estará pronto com a conclusão da repetição. É por isso que um acumulador deve ser inicializado com um valor neutro para a operação em que será utilizado. Por exemplo, se for usado em uma adição, deve ser inicializado com zero; se for usado em uma multiplicação, deve ser inicializado com 1.

Exemplo de acumulador:

```
SOMA = 0; // inicialização da variável SOMA com o valor zero
for(I = 1; I <= 5; I++)
{
    printf("Digite um número: ");
    scanf("%d%c", &NUM);
    SOMA = SOMA + NUM; // acumulando o valor da variável NUM na variável SOMA
}
printf("Soma = %d", SOMA);
```

Simulação:

MEMÓRIA			TELA
I	NUM	SOMA	
		0	Inicialização da variável SOMA com o valor zero SOMA = 0;
1			Digite um número: 5
1	5		
1	5	5	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
2			Digite um número: 3
2	3		
2	3	8	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
3			Digite um número: 0
3	0		
3	0	8	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
4			Digite um número: 10
4	10		
4	10	18	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
5			Digite um número: 2
5	2		
5	2	20	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
			Soma = 20

Exemplo de contador:

```

CONT = 0;           // inicialização da variável CONT com o valor zero
for(I = 1; I<= 5; I++)
{
    printf("Digite um número: ");
    scanf("%d%c",&NUM);
    if (NUM > 5)
        CONT = CONT + 1; // contando mais 1 na variável CONT
}
printf("Quantidade de número maiores que 5 = %d",CONT);

```

Simulação:

MEMÓRIA			TELA
I	NUM	CONT	
		0	Inicialização da variável CONT com o valor zero CONT = 0;
1			Digite um número: 5
1	5		
1	5	0	O número digitado não é maior que 5, logo, o contador CONT não será alterado
2			Digite um número: 12
2	12		

MEMÓRIA			TELA
I	NUM	CONT	
2	12	1	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT = CONT + 1;
3			Digite um número: 8
3	8		
3	8	2	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT = CONT + 1;
4			Digite um número: 3
4	3		
4	3	2	O número digitado não é maior que 5, logo, o contador CONT não será alterado
5			Digite um número: 6
5	6		
5	6	3	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT = CONT + 1;
			Quantidade de números maiores que 5 = 3

A terceira parte da estrutura de repetição FOR é um contador que pode ser incrementado ou decrementado.

Exemplos:

```
I = I + 1;      // contador incrementado em 1 unidade
I++;           // contador incrementado em 1 unidade
I = I - 1;     // contador decrementado em 1 unidade
I--;           // contador decrementado em 1 unidade
I = I + 2;     // contador incrementado em 2 unidades
J = J - 3;     // contador decrementado em 3 unidades
```

5.3.2 Estrutura de repetição WHILE

Trata-se de uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não for fixo. Os comandos serão repetidos até a condição assumir o valor falso.

Nesse tipo de estrutura, o teste condicional ocorre no início. Isso significa que existe a possibilidade da repetição não ser executada quando a condição assumir o valor falso logo na primeira verificação.

```
while(condição)
comando;
```

Enquanto a condição for verdadeira, o comando será executado.

```
while(condição)
{ comando1;
  comando2;
  comando3;
  ...
}
```

Enquanto a condição for verdadeira, os comandos que estão dentro das chaves serão executados (comando1, comando2, comando3...).

Exemplos:

```
x = 0;           // inicialização da variável x com o valor 0
while (x != 5)
{
    printf("Valor de X = %d",x);
    x = x + 1;    // contador incrementado em 1 unidade
}
printf("Valor de X depois que sair da estrutura = %d",x);
```

No trecho de programa anterior, os comandos `printf("Valor de X = %d",x);` e `x = x + 1;` foram executados cinco vezes. O teste condicional avalia `x` valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	Valor inicial
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição
Valor de X = 1	2	
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```
x = 1;           // inicialização da variável X com o valor 1
Y = 10;          // inicialização da variável Y com o valor 10
while (Y > x)
{
    printf("Valor de Y = %d",Y);
    Y = Y - 2;    // contador decrementado em 2 unidades
}
printf("Valor de Y depois que sair da estrutura = %d",Y);
```

No trecho de programa acima, os comandos `printf("Valor de Y = %d",Y);` e `Y = Y - 2;` foram executados cinco vezes. O teste condicional avalia `Y` valendo 10, 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	Valores iniciais
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

```
x = 1;           // inicialização da variável X com o valor 1
Y = 1;           // inicialização da variável Y com o valor 1
while (X < Y)
{
```

```
    printf("Valor de X = %d",X);
    X = X + 1; // contador incrementado em 1 unidade
}
```

No trecho de programa anterior, os comandos `printf("Valor de X = %d",X);` e `X = X + 1;` não foram executados, pois, com os valores iniciais de `x` e `y`, a condição é falsa; logo, não ocorre a entrada na estrutura de repetição para execução de seus comandos.

5.3.3 Estrutura de repetição DO-WHILE

Trata-se de uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não for fixo. Os comandos serão repetidos até a condição assumir o valor falso.

Nesse tipo de estrutura o teste condicional ocorre no fim. Isso significa que a repetição será executada, no mínimo, uma vez, quando todo o bloco for executado uma vez, e, ao final, a condição assumir o valor falso.

```
do
{
    comandos;
}
while (condição);
```

Os comandos serão repetidos até que a condição assuma valor falso.

Exemplos:

```
X = 0;           // inicialização da variável X com o valor 0
do
{
    printf("Valor de X = %d",X);
    X = X + 1; // contador incrementado em 1 unidade
}
while (X != 5);
printf("Valor de X depois que sair da estrutura = %d",X);
```

No trecho de programa anterior, os comandos `printf("Valor de X = %d",X);` e `X = X + 1;` foram executados cinco vezes. O teste condicional avalia `x` valendo 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	Valor inicial
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição
Valor de X = 1	2	
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```
X = 1;           // inicialização da variável X com o valor 1
Y = 10;          // inicialização da variável Y com o valor 10
do
{
    printf("Valor de Y = %d",Y);
    Y = Y - 2; // decrementando o contador em 2 unidades
}
while (Y > X);
printf("Valor de Y depois que sair da estrutura = %d",Y);
```


No trecho de programa anterior, os comandos `printf("Valor de Y = %d",Y);` e `Y = Y - 2;` são executados cinco vezes. O teste condicional avalia `Y` valendo 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	Valores iniciais
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

5.4 Estrutura de repetição em JAVA

5.4.1 Estrutura de repetição FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que um trecho do programa deve ser repetido.

O formato geral do comando `FOR` é composto por três partes:

```
for (i=valor_inicial; condição; incremento ou decremento de i)
    comando;
```

A primeira parte atribui um `valor_inicial` à variável `i`, que tem como função controlar o número necessário de repetições.

A segunda parte corresponde a uma expressão relacional que, quando assumir o valor falso, determinará o fim da repetição.

A terceira parte é responsável por alterar o valor da variável `i` (incremento ou decremento) com o objetivo de, em algum momento, fazer a condição assumir o valor falso.

Caso seja necessária a repetição de apenas um comando, o compilador entenderá que a estrutura de repetição terminará quando for encontrado o primeiro `;` (ponto e vírgula).

Exemplo:

```
for (a=1;a<=20;a++)
    System.out.println("O valor de a é: " +a);
```

No exemplo anterior, a variável `a` é inicializada com o valor 1 (`a=1`) e é incrementada em uma unidade (`a++`). A cada incremento, o comando `System.out.println` é executado. Esse processo se repete até o valor da variável `a` se tornar maior que 20 (quando a condição `a <= 20` assumir o valor falso).

Se for necessária a repetição de mais de um comando, a linguagem entenderá que a estrutura de repetição começará quando for encontrado o símbolo `{` e terminará quando for encontrado o símbolo `}`.

Exemplo:

```
for (a=15;a>=1;a=a-2)
{
    System.out.println("Digite um número: ");
    dado = new Scanner(System.in);
    x = dado.nextInt();
}
```

No exemplo anterior, a variável `a` começa com o valor 15 (`a=15`) e é decrementada em duas unidades (`a=a-2`). A cada decremento, o bloco de comando que está entre chaves `{ ... }` é executado. Esse processo se repete até o valor da variável `a` se tornar menor que 1 (quando a condição `a >= 1` assumir o valor falso).

Exemplos:

```
for (i = 1; i <= 5; i++)
    System.out.println(i);
ou
for (i = 1; i <= 5; i=i+1)
    System.out.println(i);
```

Nos trechos de programa anteriores, que apesar de utilizarem formas diferentes para aumentar o valor da variável `i`, expressam a mesma coisa, o comando `System.out.println(i);` foi executado cinco vezes, ou seja, para `i` valendo 1, 2, 3, 4 e 5.

```
for (i = 10; i >= 1; i--)
    System.out.println(i);

ou
for (i = 10; i >= 1; i=i-1)
    System.out.println(i);
```

Nos trechos de programa anteriores, apesar de utilizarem formas diferentes para diminuir o valor da variável `i`, são exatamente a mesma coisa, o comando `System.out.println(i);` foi executado dez vezes, ou seja, para `i` valendo 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

```
for (i = 0; i <= 10; i=i+2)
    System.out.println(i);
```

No trecho de programa anterior, o comando `System.out.println(i);` foi executado seis vezes, ou seja, para `i` valendo 0, 2, 4, 6, 8 e 10.

```
for (i = 100; i >= 0; i=i-20)
    System.out.println(i);
```

No trecho de programa acima, o comando `System.out.println(i);` foi executado seis vezes, ou seja, para `i` valendo 100, 80, 60, 40, 20 e 0.

Existem duas instruções comumente usadas nos comandos internos das estruturas de repetição. São as instruções denominadas acumuladores e contadores.

Os acumuladores devem ser usados quando a realização de um cálculo precisa de valores obtidos a cada iteração, ou seja, o cálculo só estará concluído com a conclusão da repetição. É por isso que um acumulador deve ser inicializado com um valor neutro para a operação em que será utilizado. Por exemplo, se for usado em uma adição, deve ser inicializado com zero; se for usado em uma multiplicação, deve ser inicializado com 1.

Exemplo de acumulador:

```
SOMA = 0; // inicialização da variável SOMA com o valor zero
for(I = 1; I <= 5; I++)
{
    System.out.print("Digite um número: ");
    NUM = dado.nextInt();
    SOMA = SOMA + NUM; // acumulando o valor da variável NUM na variável SOMA
}
System.out.println ("Soma = "+SOMA);
```

Simulação:

MEMÓRIA			TELA
I	NUM	SOMA	
		0	Inicialização da variável SOMA com o valor zero SOMA = 0;
1			Digite um número: 5
1	5		
1	5	5	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
2			Digite um número: 3
2	3		
2	3	8	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
3			Digite um número: 0
3	0		
3	0	8	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
4			Digite um número: 10
4	10		
4	10	18	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
5			Digite um número: 2
5	2		
5	2	20	Acumulando o valor da variável NUM na variável SOMA SOMA = SOMA + NUM;
			Soma = 20

Exemplo de contador:

```

CONT = 0;           // inicialização da variável CONT com o valor zero
for(I = 1; I <= 5; I++)
{
    System.out.print("Digite um número: ");
    NUM = dado.nextInt();
    if (NUM > 5)
        CONT = CONT + 1; // contando mais 1 na variável CONT
}
System.out.println("Quantidade de número maiores que 5 = " + CONT);

```

Simulação:

MEMÓRIA			TELA
I	NUM	CONT	
		0	Inicialização da variável CONT com o valor zero CONT = 0;
1			Digite um número: 5
1	5		
1	5	0	O número digitado não é maior que 5, logo, o contador CONT não será alterado
2			Digite um número: 12
2	12		
2	12	1	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT = CONT + 1;

MEMÓRIA			TELA
I	NUM	CONT	
3			Digite um número: 8
3	8		
3	8	2	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT = CONT + 1;
4			Digite um número: 3
4	3		
4	3	2	O número digitado não é maior que 5, logo, o contador CONT não será alterado
5			Digite um número: 6
5	6		
5	6	3	O número digitado é maior que 5, logo, o contador CONT será incrementado em 1 unidade CONT = CONT + 1;
			Quantidade de números maiores que 5 = 3

A terceira parte da estrutura de repetição FOR é um contador que pode ser incrementado ou decrementado.

Exemplos:

```
I = I + 1;      // contador incrementado em 1 unidade
I++;           // contador incrementado em 1 unidade
I = I - 1;     // contador decrementado em 1 unidade
I--;           // contador decrementado em 1 unidade
I = I + 2;     // contador incrementado em 2 unidades
J = J - 3;     // contador decrementado em 3 unidades
```

5.4.2 Estrutura de repetição WHILE

Trata-se de uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não for fixo. Os comandos serão repetidos até a condição assumir o valor falso.

Nesse tipo de estrutura o teste condicional ocorre no início, o que significa que existe a possibilidade da repetição não ser executada quando a condição assumir o valor falso logo na primeira verificação.

```
while (condição)
    comando;
```

Enquanto a condição for verdadeira, o comando será executado.

```
while (condição)
{
    comando1;
    comando2;
    comando3;
    ...
}
```

Enquanto a condição for verdadeira, os comandos que estão dentro das chaves serão executados (comando1, comando2, comando3...).

Exemplos:

```
x = 0;          // inicialização da variável X com o valor 0
while (x != 5)
{
```

```

    System.out.println("Valor de X = "+X);
    x = x + 1; // contador incrementado em 1 unidade
}
System.out.println("Valor de X depois que sair da estrutura = "+X);

```

No trecho de programa anterior, os comandos `System.out.println("Valor de X = "+X);` e `x = x + 1;` foram executados cinco vezes. O teste condicional avalia `x` valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	Valor inicial
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição
Valor de X = 1	2	
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```

x = 1; // inicialização da variável X com o valor 1
y = 10; // inicialização da variável Y com o valor 10
while (y > x)
{
    System.out.println("Valor de Y = "+Y);
    y = y - 2; // contador decrementado em 2 unidades
}
System.out.println("Valor de Y depois que sair da estrutura = "+Y);

```

No trecho de programa anterior, os comandos `System.out.println("Valor de Y = "+Y);` e `y = y - 2;` são executados cinco vezes. O teste condicional avalia `y` valendo 10, 8, 6, 4, 2 e 0.

TELA	X	Y	
	1	10	Valores iniciais
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

```

x = 1; // inicialização da variável X com o valor 1
y = 1; // inicialização da variável Y com o valor 1
while (x < y)
{
    System.out.println("Valor de x = "+X);
    x = x + 1; // contador incrementado em 1 unidade
}

```

No trecho de programa, os comandos `System.out.println("Valor de x = "+X);` e `x = x + 1;` não serão executados, pois com os valores iniciais de `x` e `y` a condição é falsa. Logo, não ocorrerá a entrada na estrutura de repetição para execução dos seus comandos.

5.4.3 Estrutura de repetição DO-WHILE

Trata-se de uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não for fixo. Os comandos serão repetidos até a condição assumir o valor falso.

Nesse tipo de estrutura o teste condicional ocorre no fim. Isso significa que a repetição será executada, no mínimo, uma vez, quando todo o bloco for executado uma vez, e, ao final, a condição assumir o valor falso.

```
do
{
    comandos;
}
while (condição);
```

Os comandos serão repetidos até que a condição assuma o valor falso.

Exemplos:

```
x = 0;                // inicialização da variável X com o valor 0
do
{
    System.out.println("Valor de X = " +X);
    x = x + 1;        // contador incrementado em 1 unidade
}
while (x != 5);
System.out.println("Valor de X depois que sair da estrutura = "+X);
```

No trecho de programa anterior, os comandos `System.out.println ("Valor de x = "+X);` e `x = x + 1;` foram executados cinco vezes. O teste condicional avalia x valendo 1, 2, 3, 4 e 5.

TELA	X	
	0	Valor inicial
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição
Valor de X = 1	2	
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```
x = 1;                // inicialização da variável X com o valor 1
y = 10;               // inicialização da variável Y com o valor 10
do
{
    System.out.println("Valor de Y = " + Y);
    Y = Y - 2;        // contador decrementado em 2 unidades
}
while (Y > X);
System.out.println("Valor de Y depois que sair da estrutura = " + Y);
```

No trecho de programa anterior, os comandos `System.out.println ("Valor de Y = " + Y);` e `Y = Y - 2;` foram executados cinco vezes. O teste condicional avalia y valendo 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	Valores iniciais
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

EXERCÍCIOS RESOLVIDOS

- 1.** Um funcionário de uma empresa recebe, anualmente, aumento salarial. Sabe-se que:
- Esse funcionário foi contratado em 2005, com salário inicial de R\$ 1.000,00.
 - Em 2006, ele recebeu aumento de 1,5% sobre seu salário inicial.
 - A partir de 2007 (inclusive), os aumentos salariais sempre corresponderam ao dobro do percentual do ano anterior.

Faça um programa que determine o salário atual desse funcionário.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE i, ano_atual, salario NUMÉRICO
        novo_salario, percentual NUMÉRICO
LEIA ano_atual
salario ← 1000
percentual ← 1.5/100
novo_salario ← salario + percentual * salario
PARA i ← 2007 ATÉ ano_atual FAÇA
INÍCIO
    percentual ← 2 * percentual
    novo_salario ← novo_salario + percentual * novo_salario
FIM
ESCREVA novo_salario
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX1_A.PAS e \EXERC\CAP5\PASCAL\EX1_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX1_B.PAS e \EXERC\CAP5\PASCAL\EX1_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX1_A.CPP e \EXERC\CAP5\C++\EX1_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX1_B.CPP e \EXERC\CAP5\C++\EX1_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX1_A.java e \EXERC\CAP5\JAVA\EX1_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX1_B.java e \EXERC\CAP5\JAVA\EX1_B.class

- 2.** Faça um programa que leia um valor N inteiro e positivo. Calcule e mostre o valor de E, conforme a fórmula a seguir:

$$E = 1 + 1/1! + 1/2! + 1/3! + \dots + 1/N!$$

ALGORITMO SOLUÇÃO:

```
ALGORITMO
DECLARE n, e, i, j, fat NUMÉRICO
LEIA n
e ← 1
PARA i ← 1 ATÉ n FAÇA
  INÍCIO
    fat ← 1
    PARA j ← 1 ATÉ i FAÇA
      INÍCIO
        fat ← fat * j
      FIM
    e ← e + 1/fat
  FIM
ESCREVA e
FIM_ALGORITMO.
```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX2_A.PAS e \EXERC\CAP5\PASCAL\EX2_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX2_B.PAS e \EXERC\CAP5\PASCAL\EX2_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX2_A.CPP e \EXERC\CAP5\C++\EX2_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX2_B.CPP e \EXERC\CAP5\C++\EX2_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX2_A.java e \EXERC\CAP5\JAVA\EX2_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX2_B.java e \EXERC\CAP5\JAVA\EX2_B.class

- 3.** Faça um programa que leia um número N que indica quantos valores inteiros e positivos devem ser lidos a seguir. Para cada número lido, mostre uma tabela contendo o valor lido e o fatorial desse valor.

ALGORITMO SOLUÇÃO:

```
ALGORITMO
DECLARE n, num, i, j, fat NUMÉRICO
LEIA n
PARA i ← 1 ATÉ n FAÇA
  INÍCIO
    LEIA num
    fat ← 1
    PARA j ← 1 ATÉ num FAÇA
      INÍCIO
        fat ← fat * j
      FIM
    ESCREVA fat
  FIM
FIM_ALGORITMO.
```


PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX3_A.PAS e \EXERC\CAP5\PASCAL\EX3_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX3_B.PAS e \EXERC\CAP5\PASCAL\EX3_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX3_A.CPP e \EXERC\CAP5\C++\EX3_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX3_B.CPP e \EXERC\CAP5\C++\EX3_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX3_A.java e \EXERC\CAP5\JAVA\EX3_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX3_B.java e \EXERC\CAP5\JAVA\EX3_B.class

- 4.** Foi feita uma estatística em cinco cidades brasileiras para coletar dados sobre acidentes de trânsito. Foram obtidos os seguintes dados:

- código da cidade;
- número de veículos de passeio;
- número de acidentes de trânsito com vítimas.

Deseja-se saber:

- qual é o maior e qual é o menor índice de acidentes de trânsito e a que cidades pertencem;
- qual é a média de veículos nas cinco cidades juntas;
- qual é a média de acidentes de trânsito nas cidades com menos de 2.000 veículos de passeio.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE cont, cod, num_vei, num_acid NUMÉRICO
      maior, cid_maior, menor, cid_menor NUMÉRICO
      media_vei, soma_vei, media_acid NUMÉRICO
      soma_acid, cont_acid NUMÉRICO
soma_vei ← 0
soma_acid ← 0
cont_acid ← 0
PARA cont ← 1 ATÉ 5 FAÇA
  INÍCIO
    LEIA cod, num_vei, num_acid
    SE cont = 1
      ENTÃO INÍCIO
        maior ← num_acid
        cid_maior ← cod
        menor ← num_acid
        cid_menor ← cod
      FIM
    SENÃO INÍCIO
      SE num_acid > maior
        ENTÃO INÍCIO
          maior ← num_acid
          cid_maior ← cod
        FIM
      SE num_acid < menor
        ENTÃO INÍCIO
          menor ← num_acid
    FIM
  FIM

```

```

        cid_menor ← cod
    FIM

    FIM
    soma_vei ← soma_vei + num_vei
SE num_vei < 2000
ENTÃO INÍCIO
    soma_acid ← soma_acid + num_acid
    cont_acid ← cont_acid + 1
    FIM
FIM

FIM
ESCREVA maior, cid_maior
ESCREVA menor, cid_menor
media_vei ← soma_vei/5
ESCREVA media_vei
SE cont_acid = 0
    ENTÃO ESCREVA "Não foi digitada nenhuma cidade com menos de 2000 veículos"
    SENÃO INÍCIO
        media_acid ← soma_acid/cont_acid
        ESCREVA media_acid
    FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX4_A.PAS e \EXERC\CAP5\PASCAL\EX4_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX4_B.PAS e \EXERC\CAP5\PASCAL\EX4_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX4_A.CPP e \EXERC\CAP5\C++\EX4_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX4_B.CPP e \EXERC\CAP5\C++\EX4_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX4_A.java e \EXERC\CAP5\JAVA\EX4_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX4_B.java e \EXERC\CAP5\JAVA\EX4_B.class

- 5.** Faça um programa que leia o número de termos e um valor positivo para X. Calcule e mostre o valor da série a seguir:

$$s = \frac{-X^2}{1!} + \frac{+X^3}{2!} - \frac{X^4}{3!} + \frac{+X^5}{4!} - \frac{X^6}{3!} + \frac{+X^7}{2!} - \frac{X^8}{1!} + \frac{+X^9}{2!} - \frac{X^{10}}{3!} + \frac{+X^{11}}{4!} - \dots$$

ALGORITMO Solução:

```

ALGORITMO
DECLARE fim, i, j, x, expoente, num_termos NUMÉRICO
        den, denominador, fat, sNUMÉRICO
LEIA num_termos, x
s ← 0
denominador ← 1
PARA i ← 1 TO num_termos FAÇA
INÍCIO
    fim ← denominador
    fat ← 1

```

```

    PARA j ← 1 ATÉ fim FAÇA
        INÍCIO
            fat ← fat * j
        FIM
    expoente ← i + 1
    SE RESTO (expoente/2) = 0
    ENTÃO s ← s - xexpoente/fat
    SENÃO s ← s + xexpoente/fat
    SE denominador = 4
    ENTÃO den ← -1
    SE denominador = 1
    ENTÃO den ← 1
    SE den = 1
    ENTÃO denominador ← denominador + 1
    SENÃO denominador ← denominador - 1
    FIM
    ESCRIVA s
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX5_A.PAS e \EXERC\CAP5\PASCAL\EX5_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX5_B.PAS e \EXERC\CAP5\PASCAL\EX5_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX5_A.CPP e \EXERC\CAP5\C++\EX5_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX5_B.CPP e \EXERC\CAP5\C++\EX5_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX5_A.java e \EXERC\CAP5\JAVA\EX5_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX5_B.java e \EXERC\CAP5\JAVA\EX5_B.class

6. Uma empresa possui dez funcionários com as seguintes características: código, número de horas trabalhadas no mês, turno de trabalho (M — matutino; V — vespertino; ou N — noturno), categoria (O — operário; ou G — gerente), valor da hora trabalhada. Sabendo-se que essa empresa deseja informatizar sua folha de pagamento, faça um programa que:

- Leia as informações dos funcionários, exceto o valor da hora trabalhada, não permitindo que sejam informados turnos e nem categorias inexistentes. Trabalhe sempre com a digitação de letras maiúsculas.
- Calcule o valor da hora trabalhada, conforme a tabela a seguir. Adote o valor de R\$ 450,00 para o salário mínimo.

CATEGORIA	TURNO	VALOR DA HORA TRABALHADA
G	N	18% do salário mínimo
G	M ou V	15% do salário mínimo
O	N	13% do salário mínimo
O	M ou V	10% do salário mínimo

- Calcule o salário inicial dos funcionários com base no valor da hora trabalhada e no número de horas trabalhadas.
- Calcule o valor do auxílio alimentação recebido pelo funcionário de acordo com seu salário inicial, conforme a tabela a seguir.

SALÁRIO INICIAL	AUXÍLIO ALIMENTAÇÃO
Até R\$ 300,00	20% do salário inicial
Entre R\$ 300,00 e R\$ 600,00	15% do salário inicial
Acima de R\$ 600,00	5% do salário inicial

- e) Mostre o código, número de horas trabalhadas, valor da hora trabalhada, salário inicial, auxílio alimentação e salário final (salário inicial + auxílio alimentação).

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE cont, codigo, nht, valor NUMÉRICO
        sal_min, sal_inicial, aux, sal_final NUMÉRICO
        turno, categoria LITERAL
sal_min ← 450
PARA cont ← 1 ATÉ 10 FAÇA
INÍCIO
    LEIA codigo, nht, turno, categoria
    ENQUANTO turno ≠ "M" E turno ≠ "V" E turno ≠ "N" FAÇA
        INÍCIO
            LEIA turno
        FIM
    ENQUANTO categoria ≠ "G" E categoria ≠ "O" FAÇA
        INÍCIO
            LEIA categoria
        FIM
    SE categoria = "G"
    ENTÃO INÍCIO
        SE turno = "N"
            ENTÃO valor ← sal_min * 18/100
            SENÃO valor ← sal_min * 15/100
        FIM
    SENÃO INÍCIO
        SE turno = "N"
            ENTÃO valor ← sal_min * 13/100
            SENÃO valor ← sal_min * 10/100
        FIM
    sal_inicial ← nht * valor
    SE sal_inicial ≤ 300
        ENTÃO aux ← sal_inicial * 20/100
    SENÃO SE sal_inicial < 600
        ENTÃO aux ← sal_inicial * 15/100
        SENÃO aux ← sal_inicial * 5/100
    sal_final ← sal_inicial + aux
    ESCREVA codigo, nht, valor, sal_inicial, aux, sal_final
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX6_A.PAS e \EXERC\CAP5\PASCAL\EX6_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX6_B.PAS e \EXERC\CAP5\PASCAL\EX6_B.EXE

C/C++1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX6_A.CPP e \EXERC\CAP5\C++\EX6_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX6_B.CPP e \EXERC\CAP5\C++\EX6_B.EXE

JAVA1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX6_A.java e \EXERC\CAP5\JAVA\EX6_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX6_B.java e \EXERC\CAP5\JAVA\EX6_B.class

7. Faça um programa que monte os oito primeiros termos da sequência de Fibonacci.

0-1-1-2-3-5-8-13-21-34-55...

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE cont, num1, num2, res NUMÉRICO
num1 ← 0
num2 ← 1
ESCREVA num1
ESCREVA num2
PARA cont ← 3 ATÉ 8 FAÇA
    INÍCIO
        res ← num1 + num2
        ESCRIVA res
        num1 ← num2
        num2 ← res
    FIM
FIM_ALGORITMO.

```

PASCAL1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX7_A.PAS e \EXERC\CAP5\PASCAL\EX7_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX7_B.PAS e \EXERC\CAP5\PASCAL\EX7_B.EXE

C/C++1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX7_A.CPP e \EXERC\CAP5\C++\EX7_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX7_B.CPP e \EXERC\CAP5\C++\EX7_B.EXE

JAVA1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX7_A.java e \EXERC\CAP5\JAVA\EX7_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX7_B.java e \EXERC\CAP5\JAVA\EX7_B.class

8. Faça um programa que leia o número de termos, determine e mostre os valores de acordo com a série a seguir:

Série = 2, 7, 3, 4, 21, 12, 8, 63, 48, 16, 189, 192, 32, 567, 768...

ALGORITMO Solução:

```

ALGORITMO
DECLARE i, num_termos, num1, num2, num3 NUMÉRICO
LEIA num_termos
num1 ← 2
num2 ← 7
num3 ← 3
ESCREVA num1
ESCREVA num2
ESCREVA num3
i ← 4
enquanto i ≠ num_termos FAÇA
INÍCIO
    num1 ← num1 * 2
    ESCRIVA num1
    i ← i + 1
    SE i ≠ num_termos
    ENTÃO INÍCIO
        num2 ← num2 * 3
        ESCRIVA num2
        i ← i + 1
        SE i ≠ num_termos
        ENTÃO INÍCIO
            num3 ← num3 * 4
            ESCRIVA num3
            i ← i + 1
        FIM
    FIM
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX8_A.PAS e \EXERC\CAP5\PASCAL\EX8_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX8_B.PAS e \EXERC\CAP5\PASCAL\EX8_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX8_A.CPP e \EXERC\CAP5\C++\EX8_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX8_B.CPP e \EXERC\CAP5\C++\EX8_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX8_A.java e \EXERC\CAP5\JAVA\EX8_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX8_B.java e \EXERC\CAP5\JAVA\EX8_B.class

9. Faça um programa que receba duas notas de seis alunos. Calcule e mostre:

- a média aritmética das duas notas de cada aluno; e
- a mensagem que está na tabela a seguir:

MÉDIA ARITMÉTICA	MENSAGEM
Até 3	Reprovado
Entre 3 e 7	Exame
De 7 para cima	Aprovado

- o total de alunos aprovados;
- o total de alunos de exame;
- o total de alunos reprovados;
- a média da classe.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE cont, n1, n2, media, ta, te, tr NUMÉRICO
        media_classe, total_classe NUMÉRICO
total_classe ← 0
PARA cont ← 1 ATÉ 6 FAÇA
    INÍCIO
        LEIA n1, n2
        media ← (n1 + n2) / 2
        ESCRIVA media
        SE media <= 3
            ENTÃO INÍCIO
                tr ← tr + 1
                ESCRIVA "Reprovado"
            FIM
        SE media > 3 E media < 7
            ENTÃO INÍCIO
                te ← te + 1
                ESCRIVA "Exame"
            FIM
        SE media >= 7
            ENTÃO INÍCIO
                ta ← ta + 1
                ESCRIVA "Aprovado"
            FIM
        total_classe ← total_classe + media
    FIM
ESCREVA tr
ESCREVA te
ESCREVA ta
media_classe ← total_classe/6
ESCREVA media_classe
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX9_A.PAS e \EXERC\CAP5\PASCAL\EX9_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX9_B.PAS e \EXERC\CAP5\PASCAL\EX9_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX9_A.CPP e \EXERC\CAP5\C++\EX9_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX9_B.CPP e \EXERC\CAP5\C++\EX9_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX9_A.java e \EXERC\CAP5\JAVA\EX9_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX9_B.java e \EXERC\CAP5\JAVA\EX9_B.class

10. Em um campeonato de futebol existem cinco times e cada um possui onze jogadores. Faça um programa que receba a idade, o peso e a altura de cada um dos jogadores, calcule e mostre:

- a quantidade de jogadores com idade inferior a 18 anos;
- a média das idades dos jogadores de cada time;
- a média das alturas de todos os jogadores do campeonato; e
- a porcentagem de jogadores com mais de 80 kg entre todos os jogadores do campeonato.

ALGORITMO Solução:

```
ALGORITMO
DECLARE cont_time, cont_jog, idade NUMÉRICO
      peso, alt, qtde, media_idade NUMÉRICO
      media_altura, porc, tot80 NUMÉRICO
qtde ← 0
tot80 ← 0
PARA cont_time ← 1 ATÉ 5 FAÇA
  INÍCIO
    media_idade ← 0
    PARA cont_jog ← 1 ATÉ 11 FAÇA
      INÍCIO
        leia idade, peso, alt
        SE idade < 18
          ENTÃO qtde ← qtde + 1
          media_idade ← media_idade + idade
          media_altura ← media_altura + alt
          SE peso > 80
            ENTÃO tot80 ← tot80 + 1
      FIM
    media_idade
    ESCRIVA media_idade ← media_idade/11
  FIM
ESCREVA qtde
media_altura ← media_altura/55
ESCREVA media_altura
porc ← tot80 * 100/55
ESCREVA porc
FIM_ALGORITMO.
```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX10_A.PAS e \EXERC\CAP5\PASCAL\EX10_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX10_B.PAS e \EXERC\CAP5\PASCAL\EX10_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX10_A.CPP e \EXERC\CAP5\C++\EX10_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX10_B.CPP e \EXERC\CAP5\C++\EX10_B.EXE

JAVA1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX10_A.java e \EXERC\CAP5\JAVA\EX10_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX10_B.java e \EXERC\CAP5\JAVA\EX10_B.class

- 11.** Faça um programa que receba um número inteiro maior que 1, verifique se o número fornecido é primo ou não e mostre uma mensagem de número primo ou de número não primo.
Um número é primo quando é divisível apenas por 1 e por ele mesmo.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE i, num, qtde NUMÉRICO
LEIA num
qtde ← 0
PARA i ← 1 ATÉ num FAÇA
    INÍCIO
        SE RESTO(num/i) = 0
            ENTÃO qtde ← qtde + 1
        FIM
SE qtde > 2
    ENTÃO ESCREVA "Número não primo"
SENÃO ESCREVA "Número primo"
FIM_ALGORITMO.

```

PASCAL1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX11_A.PAS e \EXERC\CAP5\PASCAL\EX11_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX11_B.PAS e \EXERC\CAP5\PASCAL\EX11_B.EXE

C/C++1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX11_A.CPP e \EXERC\CAP5\C++\EX11_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX11_B.CPP e \EXERC\CAP5\C++\EX11_B.EXE

JAVA1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX11_A.java e \EXERC\CAP5\JAVA\EX11_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX11_B.java e \EXERC\CAP5\JAVA\EX11_B.class

- 12.** Em uma fábrica trabalham homens e mulheres divididos em três classes:

- trabalhadores que fazem até 30 peças por mês — classe 1;
- trabalhadores que fazem de 31 a 50 peças por mês — classe 2;
- trabalhadores que fazem mais de 50 peças por mês — classe 3.

A classe 1 recebe salário mínimo. A classe 2 recebe salário mínimo mais 3% deste salário por peça, acima das 30 peças iniciais. A classe 3 recebe salário mínimo mais 5% desse salário por peça, acima das 30 peças iniciais.

Faça um programa que receba o número do operário, o número de peças fabricadas no mês, o sexo do operário, e que também calcule e mostre:

- o número do operário e seu salário;
- o total da folha de pagamento da fábrica;
- o número total de peças fabricadas no mês;

- a média de peças fabricadas pelos homens;
- a média de peças fabricadas pelas mulheres; e
- o número do operário ou operária de maior salário.

A fábrica possui 15 operários.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE num_op, pecas_op, num_maior, cont_m, cont_f NUMÉRICO
        tot_pecas, cont, media_m, salario_maior NUMÉRICO
        media_f, salario_op, tot_folha NUMÉRICO
        sexo_op LITERAL
tot_folha ← 0
tot_pecas ← 0
media_m ← 0
media_f ← 0
cont_m ← 0
cont_f ← 0
PARA cont ← 1 ATÉ 15 FAÇA
    INÍCIO
        ESCREVA "Digite o número do ", cont, "º operário "
        LEIA num_op
        ESCREVA "Digite o sexo do operário (M ou F) "
        LEIA sexo_op
        ESCREVA "Digite o total de peças fabricadas pelo ", cont, "º operário "
        LEIA pecas_op
        SE pecas_op <= 30
            ENTÃO salario_op ← 450
        SE pecas_op > 30 E pecas_op <= 50
            ENTÃO salario_op ← 450 + ((pecas_op-30) * 3 / 100 * 450)
        SE pecas_op > 50
            ENTÃO salario_op ← 450 + ((pecas_op-30) * 5 / 100 * 450)
        ESCREVA "O operário de número ", num_op, " recebe salário = ", salario_op
        tot_folha ← tot_folha + salario_op
        tot_pecas ← tot_pecas + pecas_op
        SE sexo_op = "M"
            ENTÃO INÍCIO
                media_m ← media_m + pecas_op
                cont_m ← cont_m + 1
            FIM
        SENÃO INÍCIO
            media_f ← media_f + pecas_op
            cont_f ← cont_f + 1
        FIM
    SE cont = 1
        ENTÃO INÍCIO
            salario_maior ← salario_op
            num_maior ← num_op
        FIM
    SENÃO INÍCIO
        SE (salario_op > salario_maior)
            ENTÃO INÍCIO
                salario_maior ← salario_op
                num_maior ← num_op
            FIM
        FIM
FIM

```

```

ESCREVA "Total da folha de pagamento = ", tot_folha
ESCREVA "Total de peças fabricadas no mês = ", tot_pecas
SE cont_m = 0
ENTÃO ESCREVA "NENHUM HOMEM"
SENÃO INÍCIO
    media_m ← media_m / cont_m
    ESCREVA "Média de peças fabricadas por homens = ", media_m
FIM
SE cont_f = 0
ENTÃO ESCREVA "NENHUMA MULHER"
SENÃO INÍCIO
    media_f ← media_f / cont_f
    ESCREVA "Média de peças fabricadas por mulheres = ", media_f
FIM
ESCREVA "O número do operário com maior salário é ", num_maior
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX12_A.PAS e \EXERC\CAP5\PASCAL\EX12_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX12_B.PAS e \EXERC\CAP5\PASCAL\EX12_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX12_A.CPP e \EXERC\CAP5\C++\EX12_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX12_B.CPP e \EXERC\CAP5\C++\EX12_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX12_A.java e \EXERC\CAP5\JAVA\EX12_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX12_B.java e \EXERC\CAP5\JAVA\EX12_B.class

13. Foi feita uma pesquisa para determinar o índice de mortalidade infantil em certo período. Faça um programa que:

- leia o número de crianças nascidas no período;
- identifique o sexo (M ou F) e o tempo de vida de cada criança nascida.

O programa deve calcular e mostrar:

- a percentagem de crianças do sexo feminino mortas no período;
- a percentagem de crianças do sexo masculino mortas no período;
- a percentagem de crianças que viveram 24 meses ou menos no período.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE i, num_cri, meses, porc_f, porc_m, tot_f NUMÉRICO
        tot_m, tot_24, porc_24 NUMÉRICO
        sexo LITERAL
ESCREVA "Digite o número de crianças nascidas no período "
LEIA num_cri
tot_m ← 0
tot_f ← 0
tot_24 ← 0
PARA i ← 1 ATE num_cri FAÇA

```

```

INÍCIO
  ESCREVA "Digite o sexo da ", i, "ª criança"
  LEIA sexo
  ESCREVA "Digite o tempo de vida (em meses) da ", i, "ª criança"
  LEIA meses
  SE sexo = "M"
    ENTÃO tot_m ← tot_m + 1
  SE sexo = "F"
    ENTÃO tot_f ← tot_f + 1
  SE meses ≤ 24
    ENTÃO tot_24 ← tot_24 + 1
FIM
SE num_cri = 0
  ENTÃO ESCREVA "NENHUMA CRIANÇA DIGITADA"
SENÃO INÍCIO
  porc_m ← tot_m * 100 / num_cri
  porc_f ← tot_f * 100 / num_cri
  porc_24 ← tot_24 * 100 / num_cri
  ESCREVA "Percentual de crianças do sexo feminino mortas ", porc_f
  ESCREVA "Percentual de crianças do sexo masculino mortas ", porc_m
  ESCREVA "Percentual de crianças com 24 meses ou menos mortas
  ↳ no período ", porc_24
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX13_A.PAS e \EXERC\CAP5\PASCAL\EX13_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX13_B.PAS e \EXERC\CAP5\PASCAL\EX13_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX13_A.CPP e \EXERC\CAP5\C++\EX13_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX13_B.CPP e \EXERC\CAP5\C++\EX13_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX13_A.java e \EXERC\CAP5\JAVA\EX13_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX13_B.java e \EXERC\CAP5\JAVA\EX13_B.class

- 14.** Faça um programa que receba o valor de uma dívida e mostre uma tabela com os seguintes dados: valor da dívida, valor dos juros, quantidade de parcelas e valor da parcela.

Os juros e a quantidade de parcelas seguem a tabela:

QUANTIDADE DE PARCELAS	% DE JUROS SOBRE O VALOR_INICIAL DA DÍVIDA
1	0
3	10
6	15
9	20
12	25

Exemplo de saída do programa:

VALOR DA DÍVIDA	VALOR DOS JUROS	QUANTIDADE DE PARCELAS	VALOR DA PARCELA
R\$ 1.000,00	0	1	R\$ 1.000,00
R\$ 1.100,00	100	3	R\$ 366,67
R\$ 1.150,00	150	6	R\$ 191,67

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE valor_inicial, juros, valor_parc NUMÉRICO
      total, valor_juros, num_parc, i NUMÉRICO
ESCREVA "Digite o valor_inicial da dívida"
LEIA valor_inicial
juros ← 0
num_parc ← 1
total ← valor_inicial
valor_parc ← valor_inicial
ESCREVA total
ESCREVA juros
ESCREVA num_parc
ESCREVA valor_parc
juros ← juros + 10
num_parc ← num_parc + 2
PARA i ← 1 ATÉ 4 FAÇA
  INÍCIO
    valor_juros ← valor_inicial * juros / 100
    total ← valor_inicial + valor_juros
    valor_parc ← total / num_parc
    ESCREVA total
    ESCREVA valor_juros
    ESCREVA num_parc
    ESCREVA valor_parc
    juros ← juros + 5
    num_parc ← num_parc + 3
  FIM
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX14_A.PAS e \EXERC\CAP5\PASCAL\EX14_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX14_B.PAS e \EXERC\CAP5\PASCAL\EX14_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX14_A.CPP e \EXERC\CAP5\C++\EX14_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX14_B.CPP e \EXERC\CAP5\C++\EX14_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX14_A.java e \EXERC\CAP5\JAVA\EX14_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX14_B.java e \EXERC\CAP5\JAVA\EX14_B.class

15. Faça um programa que receba o preço unitário, a refrigeração (S para os produtos que necessitem de refrigeração e N para os que não necessitem) e a categoria (A — alimentação; L — limpeza; e V — vestuário) de doze produtos, e que calcule e mostre:

- O custo de estocagem, calculado de acordo com a tabela a seguir.

PREÇO UNITÁRIO	REFRIGERAÇÃO	CATEGORIA	CUSTO DE ESTOCAGEM
Até 20		A	R\$ 2,00
		L	R\$ 3,00
		V	R\$ 4,00
Entre 20 e 50 (inclusive)	S		R\$ 6,00
	N		R\$ 0,00
Maior que 50	S	A	R\$ 5,00
		L	R\$ 2,00
		V	R\$ 4,00
	N	A ou V	R\$ 0,00
		L	R\$ 1,00

- O imposto calculado de acordo com as regras a seguir:

Se o produto **não preencher** nenhum dos requisitos a seguir, seu imposto será de 2% sobre o preço unitário; caso contrário, será de 4%.

Os requisitos são: categoria — A e refrigeração — S.

- O preço final, ou seja, preço unitário mais custo de estocagem mais imposto.
- A classificação calculada usando a tabela a seguir.

PREÇO FINAL	CLASSIFICAÇÃO
Até R\$ 20,00	Barato
Entre R\$ 20,00 e R\$ 100,00 (inclusive)	Normal
Acima de R\$ 100,00	Caro

- A média dos valores adicionais, ou seja, a média dos custos de estocagem e dos impostos dos doze produtos.
- O maior preço final.
- O menor preço final.
- O total dos impostos.
- A quantidade de produtos com classificação barato.
- A quantidade de produtos com classificação caro.
- A quantidade de produtos com classificação normal.

ALGORITMO Solução:

```

ALGORITMO
DECLARE i, preco, custo_est, imp, preco_final, adicional NUMÉRICO
        maior_p, menor_p, tot_imp, qtd_b, qtd_n, qtd_c NUMÉRICO
        refri, categ LITERAL
adicional ← 0
tot_imp ← 0
qtd_b ← 0
qtd_n ← 0
qtd_c ← 0
PARA i ← 1 ATÉ 12 FAÇA
INÍCIO

```

```

LEIA preco
LEIA refri
LEIA categ
SE preco <= 20
    ENTÃO INÍCIO
        SE categ = "A"
            ENTÃO custo_est ← 2
        SE categ = "L"
            ENTÃO custo_est ← 3
        SE categ = "V"
            ENTÃO custo_est ← 4
        FIM
SE preco > 20 E preco <= 50
    ENTÃO INÍCIO
        SE refri = "S"
            ENTÃO custo_est ← 6
        SENÃO custo_est ← 0
        FIM
SE preco > 50
    ENTÃO INÍCIO
        SE refri = "S"
            ENTÃO INÍCIO
                SE categ = "A"
                    ENTÃO custo_est ← 5
                SE categ = "L"
                    ENTÃO custo_est ← 2
                SE categ = "V"
                    ENTÃO custo_est ← 4
                FIM
            SENÃO INÍCIO
                SE categ = "A" OU categ = "V"
                    ENTÃO custo_est ← 0
                SE categ = "L"
                    ENTÃO custo_est ← 1
            FIM
        FIM
SE categ ≠ "A" E refri ≠ "S"
    ENTÃO imp ← preco * 2 / 100
    SENÃO imp ← preco * 4 / 100
preco_final ← preco + custo_est + imp
ESCREVA custo_est
ESCREVA imp
ESCREVA preco_final
SE preco_final <= 20
    ENTÃO INÍCIO
        qtd_b ← qtd_b + 1
        ESCREVA "Classificação Barato"
        FIM
SE preco_final > 20 E preco_final <= 100
    ENTÃO INÍCIO
        qtd_n ← qtd_n + 1
        ESCREVA "Classificação Normal"
        FIM
SE preco_final > 100
    ENTÃO INÍCIO
        qtd_c ← qtd_c + 1

```

```

        ESCRIVA "Classificação Caro"
        FIM
    adicional ← adicional + custo_est + imp
    tot_imp ← tot_imp ← imp
    SE i = 1
        ENTÃO INÍCIO
            maior_p ← preco_final
            menor_p ← preco_final
            FIM
        SENÃO INÍCIO
            SE preco_final > maior_p
                ENTÃO maior_p ← preco_final
            SE preco_final < menor_p
                ENTÃO menor_p ← preco_final
            FIM
    FIM
    adicional ← adicional / 12
    ESCRIVA adicional
    ESCRIVA maior_p
    ESCRIVA menor_p
    ESCRIVA tot_imp
    ESCRIVA qtd_b
    ESCRIVA qtd_n
    ESCRIVA qtd_c
    FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX15_A.PAS e \EXERC\CAP5\PASCAL\EX15_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX15_B.PAS e \EXERC\CAP5\PASCAL\EX15_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX15_A.CPP e \EXERC\CAP5\C++\EX15_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX15_B.CPP e \EXERC\CAP5\C++\EX15_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX15_A.java e \EXERC\CAP5\JAVA\EX15_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX15_B.java e \EXERC\CAP5\JAVA\EX15_B.class

- 16.** Faça um programa para calcular a área de um triângulo e que não permita a entrada de dados inválidos, ou seja, medidas menores ou iguais a 0.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE base, altura, área NUMÉRICO
REPITA
    LEIA base
    ATÉ base > 0
    REPITA
        LEIA altura

```



```

ATÉ altura > 0
area ← base * altura / 2
ESCREVA area
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\PASCAL\EX16_A.PAS e \EXERC\CAP5\PASCAL\EX16_A.EXE
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

```
\EXERC\CAP5\PASCAL\EX16_B.PAS e \EXERC\CAP5\PASCAL\EX16_B.EXE
```

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\C++\EX16_A.CPP e \EXERC\CAP5\C++\EX16_A.EXE
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

```
\EXERC\CAP5\C++\EX16_B.CPP e \EXERC\CAP5\C++\EX16_B.EXE
```

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\JAVA\EX16_A.java e \EXERC\CAP5\JAVA\EX16_A.class
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

```
\EXERC\CAP5\JAVA\EX16_B.java e \EXERC\CAP5\JAVA\EX16_B.class
```

- 17.** Faça um programa que receba o salário de um funcionário chamado Carlos. Sabe-se que outro funcionário, João, tem salário equivalente a um terço do salário de Carlos. Carlos aplicará seu salário integralmente na caderneta de poupança, que rende 2% ao mês, e João aplicará seu salário integralmente no fundo de renda fixa, que rende 5% ao mês. O programa deverá calcular e mostrar a quantidade de meses necessários para que o valor pertencente a João iguale ou ultrapasse o valor pertencente a Carlos.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE sal_carlos, sal_joao, meses NUMÉRICO
LEIA sal_carlos
sal_joao ← sal_carlos / 3
meses ← 0
ENQUANTO sal_joao < sal_carlos FAÇA
INÍCIO
    sal_carlos ← sal_carlos + (sal_carlos * 2 / 100)
    sal_joao ← sal_joao + (sal_joao * 5 / 100)
    meses ← meses + 1
FIM
ESCREVA meses
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\PASCAL\EX17_A.PAS e \EXERC\CAP5\PASCAL\EX17_A.EXE
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

```
\EXERC\CAP5\PASCAL\EX17_B.PAS e \EXERC\CAP5\PASCAL\EX17_B.EXE
```

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\C++\EX17_A.CPP e \EXERC\CAP5\C++\EX17_A.EXE
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

```
\EXERC\CAP5\C++\EX17_B.CPP e \EXERC\CAP5\C++\EX17_B.EXE
```

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX17_A.java e \EXERC\CAP5\JAVA\EX17_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX17_B.java e \EXERC\CAP5\JAVA\EX17_B.class

- 18.** Faça um programa que leia um conjunto não determinado de valores e mostre o valor lido, seu quadrado, seu cubo e sua raiz quadrada. Finalize a entrada de dados com um valor negativo ou zero.

ALGORITMO Solução:

```

ALGORITMO
DECLARE num, quad, cubo, raiz NUMÉRICO
LEIA num
ENQUANTO num > 0 FAÇA
INÍCIO
quad ← num * num
cubo ← num * num * num
raiz ←  $\sqrt{\text{num}}$ 
ESCREVA quad, cubo, raiz
LEIA num
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX18_A.PAS e \EXERC\CAP5\PASCAL\EX18_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX18_B.PAS e \EXERC\CAP5\PASCAL\EX18_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX18_A.CPP e \EXERC\CAP5\C++\EX18_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX18_B.CPP e \EXERC\CAP5\C++\EX18_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX18_A.java e \EXERC\CAP5\JAVA\EX18_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX18_B.java e \EXERC\CAP5\JAVA\EX18_B.class

- 19.** Faça um programa que leia um número não determinado de pares de valores [m,n], todos inteiros e positivos, um par de cada vez, e que calcule e mostre a soma de todos os números inteiros entre m e n (inclusive). A digitação de pares terminará quando m for maior ou igual a n.

ALGORITMO Solução:

```

ALGORITMO
DECLARE m, n, soma, i NUMÉRICO
LEIA m
LEIA n
ENQUANTO m < n FAÇA
INÍCIO
    soma ← 0
    PARA i m ATÉ n FAÇA
    INÍCIO
        soma ← soma + i
    FIM
FIM

```

```

        ESCRIBA soma
        LEIA m
        LEIA n
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\PASCAL\EX19_A.PAS e \EXERC\CAP5\PASCAL\EX19_A.EXE
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

```
\EXERC\CAP5\PASCAL\EX19_B.PAS e \EXERC\CAP5\PASCAL\EX19_B.EXE
```

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\C++\EX19_A.CPP e \EXERC\CAP5\C++\EX19_A.EXE
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

```
\EXERC\CAP5\C++\EX19_B.CPP e \EXERC\CAP5\C++\EX19_B.EXE
```

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\JAVA\EX19_A.java e \EXERC\CAP5\JAVA\EX19_A.class
```

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

```
\EXERC\CAP5\JAVA\EX19_B.java e \EXERC\CAP5\JAVA\EX19_B.class
```

- 20.** Faça um programa para ler o código, o sexo (M — masculino; F — feminino) e o número de horas/aula dadas mensalmente pelos professores de uma universidade, sabendo-se que cada hora/aula vale R\$ 30,00. Emita uma listagem contendo o código, o salário bruto e o salário líquido (levando em consideração os descontos explicados a seguir) de todos os professores. Mostre também a média dos salários líquidos dos professores do sexo masculino e a média dos salários líquidos dos professores do sexo feminino. Considere:

- desconto para homens, 10%, e, para mulheres, 5%;
- as informações terminarão quando for lido o código = 99999.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE cod, num_h, sal_b, sal_l, media_m, media_f NUMÉRICO
        cont_m, cont_f NUMÉRICO
        sexo LITERAL
LEIA cod
cont_m ← 0
cont_f ← 0
ENQUANTO cod ≠ 99999 FAÇA
INÍCIO
    LEIA sexo
    LEIA num_h
    sal_b ← num_h * 30
    SE sexo = "M"
        ENTÃO INÍCIO
            sal_l ← sal_b - (sal_b * 10 / 100)
            media_m ← media_m + sal_l
            cont_m ← cont_m + 1
        FIM
    SE sexo = "F"
        ENTÃO INÍCIO

```

```

        sal_l ← sal_b - (sal_b * 5 / 100)
        media_f ← media_f + sal_l
        cont_f ← cont_f + 1
    FIM
    ESCRIVA cod
    ESCRIVA sal_b
    ESCRIVA sal_l
    LEIA cod
FIM
SE cont_m = 0
ENTÃO ESCRIVA "Nenhum professor do sexo masculino"
SENÃO INÍCIO
    media_m ← media_m / cont_m
    ESCRIVA media_m
    FIM
SE cont_f = 0
ENTÃO ESCRIVA "Nenhum professor do sexo feminino"
SENÃO INÍCIO
    media_f ← media_f / cont_f
    ESCRIVA media_f
    FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX20_A.PAS e \EXERC\CAP5\PASCAL\EX20_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX20_B.PAS e \EXERC\CAP5\PASCAL\EX20_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX20_A.CPP e \EXERC\CAP5\C++\EX20_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX20_B.CPP e \EXERC\CAP5\C++\EX20_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX20_A.java e \EXERC\CAP5\JAVA\EX20_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX20_B.java e \EXERC\CAP5\JAVA\EX20_B.class

21. Faça um programa que receba vários números, calcule e mostre:

- a soma dos números digitados;
- a quantidade de números digitados;
- a média dos números digitados;
- o maior número digitado;
- o menor número digitado;
- a média dos números pares;
- a porcentagem dos números ímpares entre todos os números digitados.

Finalize a entrada de dados com a digitação do número 30.000.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE num, soma, qtd, maior, menor, qtd_par NUMÉRICO
        media_par, soma_par, qtd_impar, media, perc NUMÉRICO
qtd ← 0
qtd_par ← 0
soma_par ← 0
qtd_impar ← 0
soma ← 0
LEIA num
ENQUANTO num ≠ 30000 FAÇA
INÍCIO
    SE qtd = 0
        ENTÃO INÍCIO
            maior ← num
            menor ← num
        FIM
    SENÃO INÍCIO
        SE num > maior
            ENTÃO maior ← num
        SE num < menor
            ENTÃO menor ← num
        FIM
    soma ← soma + num
    qtd ← qtd + 1
    SE RESTO(num/2) = 0
        ENTÃO INÍCIO
            soma_par ← soma_par + num
            qtd_par ← qtd_par + 1
        FIM
    SENÃO qtd_impar ← qtd_impar + 1
LEIA num
FIM
SE qtd = 0
    ENTÃO ESCREVA "Nenhum número digitado"
SENÃO INÍCIO
    ESCREVA soma
    ESCREVA qtd
    media ← soma / qtd
    ESCREVA media
    ESCREVA maior
    ESCREVA menor
    SE qtd_par = 0
        ENTÃO ESCREVA "nenhum par"
    SENÃO INÍCIO
        media_par ← soma_par / qtd_par
        ESCREVA media_par
    FIM
    perc ← qtd_impar * 100 / qtd
    ESCREVA perc
FIM
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX21_A.PAS e \EXERC\CAP5\PASCAL\EX21_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX21_B.PAS e \EXERC\CAP5\PASCAL\EX21_B.EXE



1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX21_A.CPP e \EXERC\CAP5\C++\EX21_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX21_B.CPP e \EXERC\CAP5\C++\EX21_B.EXE



1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX21_A.java e \EXERC\CAP5\JAVA\EX21_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX21_B.java e \EXERC\CAP5\JAVA\EX21_B.class

22. Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas em seu quadro de funcionários. Supondo que você seja o programador dessa empresa, faça um programa que leia, para cada candidato, a idade, o sexo (M ou F) e a experiência no serviço (S ou N). Para encerrar a entrada de dados, digite zero para a idade.

O programa também deve calcular e mostrar:

- o número de candidatos do sexo feminino;
- o número de candidatos do sexo masculino;
- a idade média dos homens que já têm experiência no serviço;
- a porcentagem dos homens com mais de 45 anos entre o total dos homens;
- o número de mulheres com idade inferior a 21 anos e com experiência no serviço;
- a menor idade entre as mulheres que já têm experiência no serviço.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE idade, tot_f, tot_m, soma1, cont_m1, cont_m2, tot NUMÉRICO
        cont_f1, media_idade, perc, menor_idade NUMÉRICO
        sexo, exp LITERAL

tot ← 0
tot_f ← 0
tot_m ← 0
soma1 ← 0
cont_m1 ← 0
cont_m2 ← 0
cont_f1 ← 0
LEIA idade
ENQUANTO idade ≠ 0 FAÇA
INÍCIO
    LEIA sexo
    LEIA exp
    SE sexo = "F" E exp = "S"
        ENTÃO INÍCIO
            SE tot = 0
                ENTÃO INÍCIO
                    menor_idade ← idade
                    tot ← 1
                FIM
            SENÃO SE idade < menor_idade
                ENTÃO menor_idade ← idade
            FIM
        FIM
    FIM

```

```

SE sexo = "M"
    ENTÃO tot_m ← tot_m + 1
SE sexo = "F"
    ENTÃO tot_f ← tot_f + 1
SE sexo = "F" E idade < 21 E exp = "S"
    ENTÃO cont_f1 ← cont_f1 + 1
SE sexo = "M" E idade > 45
    ENTÃO cont_m1 ← cont_m1 + 1
SE sexo = "M" E exp = "S"
    ENTÃO INÍCIO
        somal ← somal + idade
        cont_m2 ← cont_m2 + 1
    FIM
LEIA idade
FIM
ESCREVA tot_f
ESCREVA tot_m
SE cont_m2 = 0
    ENTÃO ESCREVA "Nenhum homem com experiência"
SENÃO INÍCIO
    media_idade ← somal / cont_m2
    ESCREVA media_idade
FIM
SE tot_m = 0
    ENTÃO ESCREVA "Nenhum homem"
SENÃO INÍCIO
    perc ← cont_m1 * 100 / tot_m
    ESCREVA perc
FIM
ESCREVA cont_f1
ESCREVA menor_idade
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX22_A.PAS e \EXERC\CAP5\PASCAL\EX22_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX22_B.PAS e \EXERC\CAP5\PASCAL\EX22_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX22_A.CPP e \EXERC\CAP5\C++\EX22_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX22_B.CPP e \EXERC\CAP5\C++\EX22_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX22_A.java e \EXERC\CAP5\JAVA\EX22_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX22_B.java e \EXERC\CAP5\JAVA\EX22_B.class

23. Faça um programa que receba o valor do salário mínimo, uma lista contendo a quantidade de quilowatts gasta por consumidor e o tipo de consumidor (1 — residencial; 2 — comercial; ou 3 — industrial) e que calcule e mostre:

- o valor de cada quilowatt, sabendo que o quilowatt custa um oitavo do salário mínimo;
- o valor a ser pago por consumidor (conta final mais acréscimo). O acréscimo encontra-se na tabela a seguir:

TIPO	% DE ACRÉSCIMO SOBRE O VALOR GASTO
1	5
2	10
3	15

- o faturamento geral da empresa;
- a quantidade de consumidores que pagam entre R\$ 500,00 e R\$ 1.000,00.

Termine a entrada de dados com quantidade de quilowatts igual a zero.

ALGORITMO Solução:

```

ALGORITMO
DECLARE sal, qtd, tipo, valor_kw, gasto, acresc NUMÉRICO
        total, tot_geral, qtd_cons NUMÉRICO
tot_geral ← 0
qtd_cons ← 0
LEIA sal, qtd
valor_kw ← sal / 8
ENQUANTO qtd ≠ 0 FAÇA
INÍCIO
    gasto ← qtd * valor_kw
    LEIA tipo
    SE tipo = 1
        ENTÃO acresc ← gasto * 5 / 100
    SE tipo = 2
        ENTÃO acresc ← gasto * 10 / 100
    SE tipo = 3
        ENTÃO acresc ← gasto * 15 / 100
    total ← gasto + acresc
    tot_geral ← tot_geral + total
    SE total >= 500 E total <= 1000
        ENTÃO qtd_cons ← qtd_cons + 1
    ESCRIVA gasto
    ESCRIVA acresc
    ESCRIVA total
    LEIA qtd
FIM
ESCREVA tot_geral
ESCREVA qtd_cons
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX23_A.PAS e \EXERC\CAP5\PASCAL\EX23_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX23_B.PAS e \EXERC\CAP5\PASCAL\EX23_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX23_A.CPP e \EXERC\CAP5\C++\EX23_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX23_B.CPP e \EXERC\CAP5\C++\EX23_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX23_A.java e \EXERC\CAP5\JAVA\EX23_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX23_B.java e \EXERC\CAP5\JAVA\EX23_B.class

- 24.** Faça um programa que apresente o menu de opções a seguir, permita ao usuário escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. Verifique a possibilidade de opção inválida e não se preocupe com restrições do tipo salário inválido.

Menu de opções:

1. Imposto
 2. Novo salário
 3. Classificação
 4. Finalizar o programa
- Digite a opção desejada.

Na opção 1: receber o salário de um funcionário, calcular e mostrar o valor do imposto usando as regras a seguir.

SALÁRIOS	% DO IMPOSTO
Menor que R\$ 500,00	5
De R\$ 500,00 a R\$ 850,00	10
Acima de R\$ 850,00	15

Na opção 2: receber o salário de um funcionário, calcular e mostrar o valor do novo salário usando as regras a seguir.

SALÁRIOS	AUMENTO
Maiores que R\$ 1.500,00	R\$ 25,00
De R\$ 750,00 (inclusive) a R\$ 1.500,00 (inclusive)	R\$ 50,00
De R\$ 450,00 (inclusive) a R\$ 750,00	R\$ 75,00
Menores que R\$ 450,00	R\$ 100,00

Na opção 3: receber o salário de um funcionário e mostrar sua classificação usando esta tabela:

SALÁRIOS	CLASSIFICAÇÃO
Até R\$ 700,00	Mal remunerado
Maiores que R\$ 700,00	Bem remunerado

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE op, sal, imp, aum, novo_sal NUMÉRICO
REPITA
  ESCREVA " MENU DE OPÇÕES"
  ESCREVA "1- Imposto"
  ESCREVA "2- Novo Salário"
  ESCREVA "3- Classificação"
  ESCREVA "4- Finalizar o programa"
  ESCREVA "Digite a opção desejada"
LEIA op
SE op > 4 OU op < 1
  ENTÃO ESCREVA "Opção inválida !"
SE op = 1

```

```

ENTÃO INÍCIO
    LEIA sal
    SE sal < 500
        ENTÃO imp ← sal * 5/100
    SE sal >= 500 E sal <= 850
        ENTÃO imp ← sal * 10/100
    SE sal > 850
        ENTÃO imp ← sal * 15/100
    ESCRIVA imp
FIM
SE op = 2
    ENTÃO INÍCIO
        LEIA sal
        SE sal > 1500
            ENTÃO aum ← 25
        SE sal >= 750 E sal <= 1500
            ENTÃO aum ← 50
        SE sal >= 450 E sal < 750
            ENTÃO aum ← 75
        SE sal < 450
            ENTÃO aum ← 100
        novo_sal ← sal + aum
        ESCRIVA novo_sal
    FIM
SE op = 3
    ENTÃO INÍCIO
        LEIA sal
        SE sal <= 700
            ENTÃO ESCRIVA "Mal Remunerado"
        SENÃO ESCRIVA "Bem Remunerado"
    FIM
ATÉ op = 4
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX24_A.PAS e \EXERC\CAP5\PASCAL\EX24_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX24_B.PAS e \EXERC\CAP5\PASCAL\EX24_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX24_A.CPP e \EXERC\CAP5\C++\EX24_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX24_B.CPP e \EXERC\CAP5\C++\EX24_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX24_A.java e \EXERC\CAP5\JAVA\EX24_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX24_B.java e \EXERC\CAP5\JAVA\EX24_B.class

- 25.** Faça um programa que receba os dados a seguir de vários produtos: preço unitário, país de origem (1 – Estados Unidos; 2 — México; e 3 — outros), meio de transporte (T — terrestre; F — fluvial; e A — aéreo), carga perigosa (S — sim; N — não), finalize a entrada de dados com um preço inválido, ou seja, menor ou igual a zero. O programa deve calcular e mostrar os itens a seguir.

- O valor do imposto, usando a tabela a seguir.

PREÇO UNITÁRIO	PERCENTUAL DE IMPOSTO SOBRE O PREÇO UNITÁRIO
Até R\$ 100,00	5%
Maior que R\$ 100,00	10%

- O valor do transporte usando a tabela a seguir.

CARGA PERIGOSA	PAÍS DE ORIGEM	VALOR DO TRANSPORTE
S	1	R\$ 50,00
	2	R\$ 21,00
	3	R\$ 24,00
N	1	R\$ 12,00
	2	R\$ 21,00
	3	R\$ 60,00

- O valor do seguro, usando a regra a seguir.

Os produtos que vêm do México e os produtos que utilizam transporte aéreo pagam metade do valor do seu preço unitário como seguro.

- O preço final, ou seja, preço unitário mais imposto mais valor do transporte mais valor do seguro.
- O total dos impostos.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE preco, imp, transp, seguro, final NUMÉRICO
        total_imp, origem NUMÉRICO
        meio_t, carga LITERAL
LEIA preco
ENQUANTO preco > 0 FAÇA
INÍCIO
    LEIA origem
    LEIA meio_t
    LEIA carga
    SE preco <= 100
        ENTÃO imp ← preco * 5 / 100
        SENÃO imp ← preco * 10 / 100
    SE carga = "S"
        ENTÃO INÍCIO
            SE origem = 1
                ENTÃO transp ← 50
            SE origem = 2
                ENTÃO transp ← 21
            SE origem = 3
                ENTÃO transp ← 24
        FIM
    SE carga = "N"
        ENTÃO INÍCIO
            SE origem = 1
                ENTÃO transp ← 12
            SE origem = 2
                ENTÃO transp ← 21
            SE origem = 3
                ENTÃO transp ← 60
        FIM

```

```

SE origem = 2 OU meio_t = "A"
    ENTÃO seguro ← preco/2
    SENÃO seguro ← 0
final ← preco + imp + transp + seguro
total_imp ← total_imp + imp
ESCREVA imp
ESCREVA transp
ESCREVA seguro
ESCREVA final
LEIA preco
FIM
ESCREVA total_imp
FIM_ALGORITMO.

```

PASCAL

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX25_A.PAS e \EXERC\CAP5\PASCAL\EX25_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX25_B.PAS e \EXERC\CAP5\PASCAL\EX25_B.EXE

C/C++

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX25_A.CPP e \EXERC\CAP5\C++\EX25_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX25_B.CPP e \EXERC\CAP5\C++\EX25_B.EXE

JAVA

1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX25_A.java e \EXERC\CAP5\JAVA\EX25_A.class

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX25_B.java e \EXERC\CAP5\JAVA\EX25_B.class

EXERCÍCIOS PROPOSTOS

1. Faça um programa que leia cinco grupos de quatro valores (A, B, C, D) e mostre-os na ordem lida. Em seguida, organize-os em ordem crescente e decrescente.
2. Uma companhia de teatro deseja montar uma série de espetáculos. A direção calcula que, a R\$ 5,00 o ingresso, serão vendidos 120 ingressos, e que as despesas serão de R\$ 200,00. Diminuindo-se em R\$ 0,50 o preço dos ingressos, espera-se que as vendas aumentem em 26 ingressos. Faça um programa que escreva uma tabela de valores de lucros esperados em função do preço do ingresso, fazendo-se variar esse preço de R\$ 5,00 a R\$ 1,00, de R\$ 0,50 em R\$ 0,50. Escreva, ainda, para cada novo preço de ingresso, o lucro máximo esperado, o preço do ingresso e a quantidade de ingressos vendidos para a obtenção desse lucro.
3. Faça um programa que receba a idade de oito pessoas, calcule e mostre:
 - a) a quantidade de pessoas em cada faixa etária;
 - b) a porcentagem de pessoas na primeira faixa etária com relação ao total de pessoas.
 - c) a porcentagem de pessoas na última faixa etária com relação ao total de pessoas

FAIXA ETÁRIA	IDADE
1ª	Até 15 anos
2ª	De 16 a 30 anos
3ª	De 31 a 45 anos
4ª	De 46 a 60 anos
5ª	Acima de 60 anos

- 4.** Faça um programa que receba um número, calcule e mostre a tabuada desse número.

Exemplo:

Digite um número: 5

$$5 \times 0 = 0$$

$$5 \times 1 = 5$$

$$5 \times 2 = 10$$

$$5 \times 3 = 15$$

$$5 \times 4 = 20$$

$$5 \times 5 = 25$$

$$5 \times 6 = 30$$

$$5 \times 7 = 35$$

$$5 \times 8 = 40$$

$$5 \times 9 = 45$$

$$5 \times 10 = 50$$

- 5.** Faça um programa que mostre as tabuadas dos números de 1 a 10.
- 6.** Uma loja utiliza o código V para transação à vista e P para transação a prazo. Faça um programa que receba o código e o valor de quinze transações, calcule e mostre:
- o valor total das compras à vista;
 - o valor total das compras a prazo;
 - o valor total das compras efetuadas; e
 - o valor da primeira prestação das compras a prazo juntas, sabendo-se que serão pagas em três vezes.
- 7.** Faça um programa que receba a idade, a altura e o peso de cinco pessoas, calcule e mostre:
- a quantidade de pessoas com idade superior a 50 anos;
 - a média das alturas das pessoas com idade entre 10 e 20 anos;
 - a porcentagem de pessoas com peso inferior a 40 kg entre todas as pessoas analisadas.
- 8.** Faça um programa que receba a idade, o peso, a altura, a cor dos olhos (A — azul; P — preto; V — verde; e C — castanho) e a cor dos cabelos (P — preto; C — castanho; L — louro; e R — ruivo) de seis pessoas, e que calcule e mostre:
- a quantidade de pessoas com idade superior a 50 anos e peso inferior a 60 kg;
 - a média das idades das pessoas com altura inferior a 1,50 m;
 - a porcentagem de pessoas com olhos azuis entre todas as pessoas analisadas; e
 - a quantidade de pessoas ruivas e que não possuem olhos azuis.
- 9.** Faça um programa que receba dez idades, pesos e alturas, calcule e mostre:
- a média das idades das dez pessoas;
 - a quantidade de pessoas com peso superior a 90 kg e altura inferior a 1,50 metro; e
 - a porcentagem de pessoas com idade entre 10 e 30 anos entre as pessoas que medem mais de 1,90 m.
- 10.** Faça um programa que receba dez números, calcule e mostre a soma dos números pares e a soma dos números primos.
- 11.** Faça um programa que receba o valor de um carro e mostre uma tabela com os seguintes dados: preço final, quantidade de parcelas e valor da parcela. Considere o seguinte:
- o preço final para compra à vista tem desconto de 20%;
 - a quantidade de parcelas pode ser: 6, 12, 18, 24, 30, 36, 42, 48, 54 e 60; e
 - os percentuais de acréscimo encontram-se na tabela a seguir.

QUANTIDADE DE PARCELAS	PERCENTUAL DE ACRÉSCIMO SOBRE O PREÇO FINAL
6	3%
12	6%
18	9%
24	12%
30	15%
36	18%
42	21%
48	24%
54	27%
60	30%

- 12.** Faça um programa que receba dez números inteiros e mostre a quantidade de números primos dentre os números que foram digitados.
- 13.** Faça um programa que receba a idade e o peso de quinze pessoas, e que calcule e mostre as médias dos pesos das pessoas da mesma faixa etária. As faixas etárias são: de 1 a 10 anos, de 11 a 20 anos, de 21 a 30 anos e de 31 anos para cima.
- 14.** Cada espectador de um cinema respondeu a um questionário no qual constava sua idade e sua opinião em relação ao filme: ótimo — 3; bom — 2; regular — 1. Faça um programa que receba a idade e a opinião de quinze espectadores, calcule e mostre:
- a média das idades das pessoas que responderam ótimo;
 - a quantidade de pessoas que responderam regular; e
 - a percentagem de pessoas que responderam bom, entre todos os espectadores analisados.
- 15.** Uma empresa fez uma pesquisa de mercado para saber se as pessoas gostaram ou não de um novo produto lançado. Para isso, forneceu o sexo do entrevistado e sua resposta (S — sim; ou N — não). Sabe-se que foram entrevistadas dez pessoas. Faça um programa que calcule e mostre:
- o número de pessoas que responderam sim;
 - o número de pessoas que responderam não;
 - o número de mulheres que responderam sim; e
 - a percentagem de homens que responderam não, entre todos os homens analisados.
- 16.** Faça um programa que receba várias idades, calcule e mostre a média das idades digitadas. Finalize digitando idade igual a zero.
- 17.** Foi feita uma pesquisa sobre a audiência de canal de TV em várias casas de uma cidade, em determinado dia. Para cada casa consultada foi fornecido o número do canal (4, 5, 7, 12) e o número de pessoas que estavam assistindo àquele canal. Se a televisão estivesse desligada, nada era anotado, ou seja, essa casa não entrava na pesquisa. Faça um programa que:
- leia um número indeterminado de dados (número do canal e número de pessoas que estavam assistindo); e
 - calcule e mostre a porcentagem de audiência de cada canal.
- Para encerrar a entrada de dados, digite o número do canal ZERO.
- 18.** Foi feita uma pesquisa entre os habitantes de uma região. Foram coletados os dados de idade, sexo (M/F) e salário. Faça um programa que calcule e mostre:
- a média dos salários do grupo;
 - a maior e a menor idade do grupo;
 - a quantidade de mulheres com salário até R\$ 200,00;
 - a idade e o sexo da pessoa que possui o menor salário.
- Finalize a entrada de dados ao ser digitada uma idade negativa.

19. Faça um programa que receba o tipo da ação, ou seja, uma letra a ser comercializada na bolsa de valores, o preço de compra e o preço de venda de cada ação e que calcule e mostre:

- o lucro de cada ação comercializada;
- a quantidade de ações com lucro superior a R\$ 1.000,00;
- a quantidade de ações com lucro inferior a R\$ 200,00;
- o lucro total da empresa.

Finalize com o tipo de ação 'F'.

20. Faça um programa que apresente o menu de opções a seguir:

Menu de opções:

1. Média aritmética
2. Média ponderada
3. Sair

Digite a opção desejada.

Na opção 1: receber duas notas, calcular e mostrar a média aritmética.

Na opção 2: receber três notas e seus respectivos pesos, calcular e mostrar a média ponderada.

Na opção 3: sair do programa.

Verifique a possibilidade de opção inválida. Nesse caso, o programa deverá mostrar uma mensagem.

21. Em uma eleição presidencial existem quatro candidatos. Os votos são informados por meio de código. Os códigos utilizados são:

1, 2, 3, 4	Votos para os respectivos candidatos
5	Voto nulo
6	Voto em branco

Faça um programa que calcule e mostre:

- o total de votos para cada candidato;
- o total de votos nulos;
- o total de votos em branco;
- a porcentagem de votos nulos sobre o total de votos; e
- a porcentagem de votos em branco sobre o total de votos.

Para finalizar o conjunto de votos, tem-se o valor zero e, para códigos inválidos, o programa deverá mostrar uma mensagem.

22. Faça um programa que receba a idade e a altura de várias pessoas, calcule e mostre a média das alturas daquelas com mais de 50 anos. Para encerrar a entrada de dados, digite idade menor ou igual a zero.

23. Faça um programa que apresente o menu de opções a seguir, que permita ao usuário escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. Verifique a possibilidade de opção inválida e não se preocupe com as restrições como salário inválido.

Menu de opções:

1. Novo salário
2. Férias
3. Décimo terceiro
4. Sair

Digite a opção desejada.

Na opção 1: receber o salário de um funcionário, calcular e mostrar o novo salário usando as regras a seguir:

SALÁRIOS	PERCENTAGEM DE AUMENTO
Até R\$ 210,00	15%
De R\$ 210,00 a R\$ 600,00 (inclusive)	10%
Acima de R\$ 600,00	5%

Na opção 2: receber o salário de um funcionário, calcular e mostrar o valor de suas férias. Sabe-se que as férias equivalem a seu salário acrescido de um terço do salário.

Na opção 3: receber o salário de um funcionário e o número de meses de trabalho na empresa, no máximo doze, calcular e mostrar o valor do décimo terceiro. Sabe-se que o décimo terceiro equivale a seu salário multiplicado pelo número de meses de trabalho dividido por 12.

Na opção 4: sair do programa.

24. Faça um programa que receba um conjunto de valores inteiros e positivos, calcule e mostre o maior e o menor valor do conjunto. Considere que:

- para encerrar a entrada de dados, deve ser digitado o valor zero;
- para valores negativos, deve ser enviada uma mensagem;
- os valores negativos ou iguais a zero não entrarão nos cálculos.

25. Uma agência bancária possui vários clientes que podem fazer investimentos com rendimentos mensais, conforme a tabela a seguir:

TIPO	DESCRIÇÃO	RENDIMENTO MENSAL
1	Poupança	1,5%
2	Poupança plus	2%
3	Fundos de renda fixa	4%

Faça um programa que leia o código do cliente, o tipo do investimento e o valor investido, e que calcule e mostre o rendimento mensal de acordo com o tipo do investimento. No final, o programa deverá mostrar o total investido e o total de juros pagos.

A leitura terminará quando o código do cliente digitado for menor ou igual a 0.