CAPÍTULO

3 Estrutura sequencial

3.1 Estrutura sequencial em algoritmos

```
ALGORITMO
     DECLARE nome_da_variável tipo_da_variável
     bloco_de_comandos
FIM_ALGORITMO.
```

3.1.1 Declaração de variáveis em algoritmos

As *variáveis* são declaradas após a palavra DECLARE e os tipos mais utilizados são: NUMÉRICO (para variáveis que receberão números), LITERAL (para variáveis que receberão caracteres) e LÓGICO (para variáveis que receberão apenas dois valores: verdadeiro ou falso).

Exemplo:

```
DECLARE X NUMÉRICO
        Y, Z LITERAL
        TESTE LÓGICO
```

3.1.2 Comando de atribuição em algoritmos

O *comando de atribuição* é utilizado para conceder valores ou operações a variáveis, sendo representado pelo símbolo ←.

Exemplo:

```
x ← 4
x ← x + 2
y ← "aula"
teste ← falso
```

3.1.3 Comando de entrada em algoritmos

O *comando de entrada* é utilizado para receber dados digitados pelo usuário, que serão armazenados em variáveis. Esse comando é representado pela palavra LEIA.

Exemplo:

```
LEIA X
```

Um valor digitado pelo usuário será armazenado na variável X.

```
LEIA Y
```

Um ou vários caracteres digitados pelo usuário serão armazenados na variável Y.

3.1.4 Comando de saída em algoritmos

O *comando de saída* é utilizado para mostrar dados na tela ou na impressora. Esse comando é representado pela palavra ESCREVA, e os dados podem ser conteúdos de variáveis ou mensagens.

Exemplo:

```
ESCREVA X
```

Mostra o valor armazenado na variável X.

```
ESCREVA "Conteúdo de Y = ",Y
```

Mostra a mensagem "Conteúdo de Y = " e, em seguida, o valor armazenado na variável Y.

3.2 Estrutura sequencial em PASCAL

```
PROGRAM nome;
USES nomes_das_unidades;
VAR nomes_das_variáveis: tipo;
BEGIN
bloco_de_comandos;
END.
```

As unidades são bibliotecas utilizadas pela linguagem PASCAL para a correta execução do programa. A unidade CRT é obrigatória em todos os programas, pois faz a adequação do *hardware* com seu programa.

3.2.1 Declaração de variáveis em PASCAL

As *variáveis* são declaradas após a palavra VAR e os tipos mais utilizados são: INTEGER (para números inteiros), REAL (para números reais), CHAR (para um caractere), STRING (para vários caracteres) e BOOLEAN (para verdadeiro ou falso).

Exemplo:

```
VAR X: INTEGER;
    Y,Z: REAL;
    NOME: STRING;
    SEXO: CHAR;
    TESTE: BOOLEAN;
```

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas e unidades, entre outras.

As regras básicas para a formação dos identificadores são:

- podem ter qualquer tamanho. Entretanto, apenas os 63 primeiros caracteres são utilizados pelo compilador;
- os caracteres que podem ser utilizados na formação dos identificadores são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado;
- o compilador não faz distinção entre letras maiúsculas e minúsculas, portanto, o identificador NUM é exatamente igual ao identificador num;
- o primeiro caractere deve ser sempre uma letra ou o caractere sublinhado;
- não são permitidos espaços em branco e caracteres especiais (@, $, +, –, %, !); e
- não é permitido usar palavras reservadas.

Palavras reservadas são nomes utilizados pelo compilador para representar comandos, operadores e nomes de seções de programas. As palavras reservadas da linguagem PASCAL são:		
`and`	`goto`	`program`
`asm`	`if`	`record`
`array`	`implementation`	`repeat`
`begin`	`in`	`set`
`case`	`inherited`	`shl`
`const`	`inline`	`shr`
`constructor`	`interface`	`string`
`destructor`	`label`	`then`
`div`	`library`	`to`
`do`	`mod`	`type`
`downto`	`nil`	`unit`
`else`	`not`	`until`
`end`	`object`	`uses`
`exports`	`of`	`var`
`file`	`or`	`while`
`for`	`packed`	`with`
`function`	`procedure`	`xor`

Os tipos de dados mais utilizados na linguagem PASCAL estão descritos na tabela a seguir:

TIPO	FAIXA DE VALORES	TAMANHO (aproximado)
`shortint`	–128 a 127	8 bits
`integer`	–32.768 a 32.767	16 bits
`longint`	–2.147.483.648 a 2.147.483.647	32 bits
`byte`	0 a 255	8 bits
`word`	0 a 65.535	16 bits
`real`	$2{,}9 \times 10^{-39}$ a $1{,}7 \times 10^{38}$ (11 a 12 dígitos com sinal)	6 bytes
`single`	$1{,}5 \times 10^{-45}$ a $3{,}4 \times 10^{38}$ (7 a 8 dígitos com sinal)	4 bytes
`double`	$5{,}0 \times 10^{-324}$ a $1{,}7 \times 10^{308}$ (15 a 16 dígitos com sinal)	8 bytes
`extended`	$3{,}4 \times 10^{-4932}$ a $1{,}1 \times 10^{4932}$ (19 a 20 dígitos com sinal)	10 bytes
`comp`	$-9{,}2 \times 10^{18}$ a $9{,}2 \times 10^{18}$ (19 a 20 dígitos com sinal)	8 bytes
`boolean`	true ou false	8 bits
`wordbool`	true ou false	16 bits
`longbool`	true ou false	32 bits
`bytebool`	true ou false	8 bits
`char`	1 caractere qualquer	1 byte
`string`	cadeia de caracteres (no máximo 255)	tantos bytes quantos forem os caracteres

3.2.2 Declaração de constantes em PASCAL

As *constantes* são declaradas após a palavra `CONST` e seus valores não podem ser alterados durante a execução do programa.

Exemplo:

```
CONST X = 8;
      Y = 2.8;
      NOME = 'MARIA';
      SEXO = 'm';
      TESTE = TRUE;
```

3.2.3 Comando de atribuição em PASCAL

O *comando de atribuição* é utilizado para conceder valores ou operações às variáveis, sendo representado por `:=` (os sinais de dois pontos e de igualdade).

Exemplo:

```
x := 4;
x := x + 2;
y := 2.5;
nome:='AULA';
sexo := 'F';
teste := false;
```

Observações

a) Em PASCAL, os caracteres literais são representados entre apóstrofos.
b) Os números reais utilizam o ponto como separador decimal.
c) Cada comando é finalizado com o sinal de ponto e vírgula.

3.2.4 Comando de entrada em PASCAL

O *comando de entrada* é utilizado para receber dados digitados pelo usuário. Esses dados são armazenados em variáveis. Esse comando é representado pela palavra `READLN`. Sua sintaxe está representada a seguir:

Sintaxe:

```
READLN(nome_da_variável);
READLN(nome_da_variável1,nome_da_variável2);
```

Exemplo:

```
READLN(X);
```

Um valor digitado pelo usuário será armazenado na variável `X`.

```
READLN(NOME);
```

Um ou vários caracteres digitados pelo usuário serão armazenados na variável `NOME`.

```
READLN(Y, Z);
```

O primeiro valor digitado será armazenado na variável `Y` e o segundo valor será armazenado na variável `Z`. Os valores podem ser digitados seguidos de enter ou separados por espaço e finalizados com enter.

3.2.5 Comando de saída em PASCAL

O *comando de saída* é utilizado para mostrar dados na tela ou na impressora. Esse comando é representado pelas palavras `WRITE` ou `WRITELN` e os dados podem ser conteúdos de variáveis ou mensagens.

Sintaxe:

```
WRITE(nome_da_variável);
WRITELN(nome_da_variável);
WRITE('mensagem');
WRITELN('mensagem');
WRITE('mensagem', nome_da_variável);
WRITELN('mensagem', nome_da_variável);
```

Exemplo:

```
WRITELN(X);
WRITE(X);
```

Mostra o valor armazenado na variável X.

```
WRITELN('Conteúdo de Y = ',Y);
WRITE('Conteúdo de Y = ',Y);
```

Mostra a mensagem "Conteúdo de Y = " e em seguida o valor armazenado na variável Y.

A diferença entre esses comandos é que o comando WRITELN mostra seu conteúdo e passa o cursor para a linha de baixo, enquanto o comando WRITE mantém o cursor na mesma linha após mostrar a mensagem.

Exemplo:

```
WRITE('aula ');
WRITE('fácil');
```

Os comandos dados anteriormente geram a saída a seguir:
aula fácil

```
WRITELN('aula ');
WRITE('fácil');
```

Os comandos dados anteriormente geram a saída a seguir:

```
aula
fácil
```

Nos comandos de saída, é possível ainda fazer a formatação de variáveis do tipo real, single, double, extended e comp. Após o nome da variável, coloca-se :m:n, onde m significa a quantidade de espaços da tela e n o número de caracteres gastos com a parte fracionária do número. O ponto, que é o separador decimal, ocupará um caractere do total de caracteres. Os espaços que sobram à esquerda serão preenchidos com branco, e, quando faltam espaços, o compilador completa com a necessidade para mostrar o resultado.

Exemplo:

```
WRITE(X:5:2);
```

No exemplo anterior, o conteúdo da variável X ocupará 5 espaços na tela. Dois espaços para a parte fracionária, um espaço para o ponto e dois espaços para a parte inteira.

```
WRITE(X:7:3);
```

No exemplo anterior, o conteúdo da variável X ocupará 7 espaços na tela. Três espaços para a parte fracionária, um espaço para o ponto e três espaços para a parte inteira.

3.2.6 Comentários em PASCAL

Os comentários não são interpretados pelo compilador, servem apenas para esclarecer o programador. Constituem excelentes instrumentos de documentação e devem sempre estar entre {.....} ou entre (*.....*).

3.2.7 Operadores e funções predefinidas em PASCAL

A linguagem PASCAL possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

Operador	Exemplo	Comentário
:=	x := y	O conteúdo da variável Y é atribuído à variável X (A uma variável pode ser atribuído o conteúdo de outra, um valor constante ou, ainda, o resultado de uma função).
+	x + y	Soma o conteúdo de X e de Y.
–	x – y	Subtrai o conteúdo de Y do conteúdo de X.
*	x * y	Multiplica o conteúdo de X pelo conteúdo de Y.
/	x / y	Obtém o quociente da divisão de X por Y.
div	x div y	Obtém o quociente inteiro da divisão de X por Y.
mod	x mod y	Obtém o resto da divisão de X por Y.

Observações

a) Os operadores `div` e `mod` só podem ser aplicados com operandos do tipo inteiro.
b) O operador `/` sempre conduz a um resultado real.
c) Com os operadores +, –, `*` e `/`, se pelo menos um dos operandos for real, então o resultado será real.

Operador	Exemplo	Comentário
=	x = y	O conteúdo de X é igual ao conteúdo de Y.
<>	x <> y	O conteúdo de X é diferente do conteúdo de Y.
<=	x <= y	O conteúdo de X é menor ou igual ao conteúdo de Y.
>=	x >= y	O conteúdo de X é maior ou igual ao conteúdo de Y.
<	x < y	O conteúdo de X é menor que o conteúdo de Y.
>	x > y	O conteúdo de X é maior que o conteúdo de Y.

Funções matemáticas		
Função	Exemplo	Comentário
abs	abs(x)	Obtém o valor absoluto de X.
exp	exp(x)	Obtém o logaritmo natural e elevado à potência X.
log	log(X)	Obtém o logaritmo natural de X.
trunc	trunc(x)	Obtém a parte inteira do número real armazenado em X.
frac	frac(x)	Obtém a parte fracionária do número real armazenado em X.
round	round(x)	Arredonda X. Para parte fracionária inferior a 0,5, o arredondamento é para baixo. Para parte fracionária de 0,5 para cima, o arredondamento é para cima. Exemplo: round(1.2) = 1 round(1.5) = 2 round(1.8) = 2
sin	sin(x)	Calcula o seno de X (X deve estar representado em radianos).
cos	cos(x)	Calcula o cosseno de X (X deve estar representado em radianos).
pi	Pi	Retorna o valor de π.
sqrt	sqrt(x)	Calcula a raiz quadrada de X.
sqr	sqr(x)	Calcula X elevado ao quadrado.
inc	inc(x,y)	Incrementa a variável X com o valor da variável Y.
dec	dec(x,y)	Decrementa a variável X com o valor da variável Y.

Observação

Por não existir o operador de potenciação, temos:

A^B = EXP(B*LN(A))

Exemplo:

3^4 = exp(4*ln(3))

5^{10} = exp(10*ln(5))

Observação

As funções SIN e COS esperam receber argumentos no formato de radianos; para receber argumentos em graus, siga o próximo exemplo. Na linguagem PASCAL, não existe uma função para tangente; assim, utilize seno/cosseno.

Exemplo para o cálculo do seno de um ângulo fornecido em graus e utilizando uma variável para o valor de π:

```
VALORPI := 3.1415;
READLN(X); { X EM GRAUS }
Y := SIN ((VALORPI * X) / 180);
```

Exemplo para o cálculo do seno de um ângulo fornecido em graus e utilizando a função pi:

```
READLN(X); { X EM GRAUS }
Y := SIN ((PI * X) / 180);
```

As prioridades entre os operadores são:

1ª) ( )

2ª) funções

3ª) *, /, DIV, MOD

4ª) +, –

Quando se tem uma expressão em que os operadores possuem a mesma prioridade, a expressão é resolvida da esquerda para a direita.

Exemplos:

```
2 + 3 – 4 = 5 – 4 = 1
2 * 4/2 = 8/2 = 4
```

3.3 Estrutura sequencial em C/C++

```
#include <nome_da_biblioteca>
int main()
{
   bloco_de_comandos;
   return 0;
}
```

Bibliotecas são arquivos contendo várias funções que podem ser incorporadas aos programas escritos em C/C++. A diretiva `#include` faz o texto contido na biblioteca especificada ser inserido no programa.

A biblioteca `stdio.h` permite a utilização de diversos comandos de entrada e saída.

É importante salientar que a linguagem C/C++ é sensível a letras maiúsculas e minúsculas, ou seja, considera que letras maiúsculas são diferentes de minúsculas (por exemplo, *a* é diferente de *A*). Sendo assim, todos os comandos devem, obrigatoriamente, ser escritos com letras minúsculas.

3.3.1 Declaração de variáveis em C/C++

As *variáveis* são declaradas após a especificação de seus tipos. Os tipos de dados mais utilizados são: `int` (para números inteiros), `float` (para números reais) e `char` (para um caractere). A linguagem C/C++ não possui tipo de dados `boolean` (que pode assumir os valores verdadeiro ou falso), pois considera verdadeiro qualquer valor diferente de 0 (zero). A linguagem C não possui um tipo especial para armazenar cadeias de caracteres (`strings`). Deve-se, quando necessário, utilizar um vetor contendo vários elementos do tipo `char`. Os vetores serão tratados no Capítulo 6.

Exemplo:

```
float X;
```

Declara uma variável chamada `X` em que pode ser armazenado um número real.

```
float Y, Z;
```

Declara duas variáveis chamadas `Y` e `Z` em que podem ser armazenados dois números reais.

```
char SEXO;
```

Declara uma variável chamada `SEXO` em que pode ser armazenado um caractere.

```
char NOME[40];
```

Declara uma variável chamada `NOME` em que podem ser armazenados até 39 caracteres. O 40º caractere será o \0, que indica final da cadeia de caracteres.

A linguagem C/C++ possui quatro tipos básicos que podem ser utilizados na declaração das variáveis: `int`, `float`, `double` e `char`. A partir desses tipos básicos, podem ser definidos outros, conforme apresentado na tabela a seguir.

Tipo	Faixa de valores	Tamanho (aproximado)
`char`	−128 a 127	8 bits
`unsigned char`	0 a 255	8 bits
`int`	−32.768 a 32.767	16 bits
`unsigned int`	0 a 65.535	16 bits
`short int`	−32.768 a 32.767	16 bits
`long`	−2.147.483.648 a 2.147.483.647	32 bits
`unsigned long`	0 a 4.294.967.295	32 bits
`float`	3.4×10^{-38} a 3.4×10^{38}	32 bits
`double`	1.7×10^{-308} a 1.7×10^{308}	64 bits
`long double`	3.4×10^{-4932} a 1.1×10^{4932}	80 bits

É importante ressaltar que, de acordo com o processador ou compilador C/C++ utilizado, o tamanho e a faixa de valores podem variar. As faixas apresentadas seguem o padrão ANSI e são consideradas mínimas.

3.3.2 Declaração de constantes em C/C++

As *constantes* são declaradas depois das bibliotecas e seus valores não podem ser alterados durante a execução do programa. A declaração de uma constate deve obedecer à seguinte sintaxe:

```
#define nome valor
```

Exemplo:

```
#define x 7
```

Define uma constante com identificador `x` e valor 7.

```
#define y 4.5
```

Define uma constante com identificador `y` e valor 4.5.

```
#define nome "MARIA"
```

Define uma constante com identificador `nome` e valor `MARIA`.

3.3.3 Comando de atribuição em C/C++

O *comando de atribuição* é utilizado para conceder valores ou operações a variáveis, sendo representado por = (sinal de igualdade).

Exemplo:

```
x = 4;
x = x + 2;
y = 2.5;
sexo = 'F';
```

Caso seja necessário armazenar uma cadeia de caracteres dentro de uma variável, deve-se utilizar uma função para manipulação de caracteres, conforme apresentado a seguir:

```
strcpy(nome, "João");
```

Para que seja possível a utilização da função `strcpy`, deve-se inserir no programa, por meio da diretiva `include`, a biblioteca `string.h`. As funções de manipulação de strings serão abordadas no Capítulo 9.

Observações

Em C/C++, os caracteres são representados entre apóstrofos ('). As cadeias de caracteres devem ser representadas entre aspas (").

Em C/C++, cada comando é finalizado com o sinal de ponto e vírgula.

Em C/C++, a parte inteira e a parte fracionária do número são separadas por um ponto.

3.3.4 Comando de entrada em C/C++

O *comando de entrada* é utilizado para receber dados digitados pelo usuário. Os dados recebidos são armazenados em variáveis. Um dos comandos de entrada mais utilizados na linguagem C/C++ é o `scanf`.

Exemplo:

```
scanf("%d%*c",&X);
```

Um valor inteiro, digitado pelo usuário, será armazenado na variável `X`.

```
scanf("%f%*c",&Z);
```

Um valor real, digitado pelo usuário, será armazenado na variável `Z`.

```
scanf("%s%*c",&NOME);
```

Um ou mais caracteres, digitados pelo usuário, serão armazenados na variável `NOME`.

```
scanf("%c%*c",&Y);
```

Um caractere, digitado pelo usuário, será armazenado na variável `Y`.

No comando `scanf`, é necessário indicar o tipo de variável que será lida: `%f` para variáveis que armazenam números reais; `%d` para variáveis que armazenam números inteiros; `%c` para variáveis que armazenam um único caractere; e `%s` para variáveis que armazenam um conjunto de caracteres.

O comando `scanf` armazena em um buffer o conteúdo digitado pelo usuário e armazena também a tecla enter utilizada pelo usuário para encerrar a entrada de dados. Para que o buffer seja esvaziado depois da atribuição do conteúdo à variável, utiliza-se `%*c`.

3.3.5 Comando de saída em C/C++

O *comando de saída* é utilizado para mostrar dados na tela ou na impressora. Um dos comandos de saída mais utilizado na linguagem C/C++ é o `printf`.

Exemplo:

```
printf("%d",Y);
```

Mostra o número inteiro armazenado na variável `Y`.

```
printf("Conteúdo de Y = %d",Y);
```

Mostra a mensagem "`Conteúdo de Y =` " e, em seguida, o número inteiro armazenado na variável `Y`.

```
printf("%f", X);
```

Mostra o número real armazenado na variável `X`.

```
printf("%5.2f", X);
```

Mostra o número real armazenado na variável `X` utilizando cinco caracteres da tela, e, destes, dois serão utilizados para a parte fracionária e um para o ponto, que é o separador da parte inteira e da parte fracionária.

```
printf("Conteúdo de X = %7.3f", X);
```

Mostra a mensagem "`Conteúdo de X = ` e, em seguida, o número real armazenado na variável `X`, utilizando sete caracteres da tela, e, destes, três serão utilizados para a parte fracionária e um para o ponto, que é o separador da parte inteira e da parte fracionária.

No comando `printf` é necessário indicar o tipo de variável que será mostrada: `%f` para variáveis que armazenam números reais, `%d` para variáveis que armazenam números inteiros, `%c` para variáveis que armazenam um único caractere, e `%s` para variáveis que armazenam um conjunto de caracteres.

No comando `printf` pode-se utilizar caracteres para posicionar a saída, por exemplo, `\n`, que passa o cursor para a próxima linha, ou `\t`, que avança o cursor uma tabulação.

Exemplo:

```
printf("aula ");
printf("fácil");
```

Os comandos dados anteriormente geram a saída a seguir:
aula fácil

```
printf("aula ");
printf("\nfácil");
```

Os comandos dados anteriormente geram a saída a seguir:
aula
fácil

```
printf("aula ");
printf("\tfácil");
```

Os comandos dados anteriormente geram a saída a seguir:
aula	fácil

3.3.6 Comentários em C/C++

Comentários são textos que podem ser inseridos em programas com o objetivo de documentá-los. Eles não são analisados pelo compilador.

Os comentários podem ocupar uma ou várias linhas, devendo ser inseridos nos programas utilizando-se os símbolos `/* ..... */` ou `//`.

Exemplo:

```
/*
linhas de comentário
linhas de comentário
*/
```

A região de comentários é aberta com os símbolos `/*` e encerrada com os símbolos `*/`.

```
// comentário
```

A região de comentários é aberta com os símbolos `//` e encerrada automaticamente ao final da linha.

3.3.7 Operadores e funções predefinidas em C/C++

A linguagem C/C++ possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

Operador	Exemplo	Comentário
=	`x = y`	O conteúdo da variável Y é atribuído à variável X (A uma variável pode ser atribuído o conteúdo de outra, um valor constante ou, ainda, o resultado de uma função).
+	`x + y`	Soma o conteúdo de X e de Y.
–	`x – y`	Subtrai o conteúdo de Y do conteúdo de X.
*	`x * y`	Multiplica o conteúdo de X pelo conteúdo de Y.
/	`x / y`	Obtém o quociente da divisão de X por Y. Se os operandos são inteiros, o resultado da operação será o quociente inteiro da divisão. Se os operandos são reais, o resultado da operação será a divisão. Por exemplo: int z = 5/2; → a variável z receberá o valor 2. float z = 5.0/2.0; → a variável z receberá o valor 2.5.
%	`x % y`	Obtém o resto da divisão de X por Y.

O operador % só pode ser utilizado com operandos do tipo inteiro.

Operador	Exemplo	Comentário
+=	x += y	Equivale a X = X + Y.
–=	x –= y	Equivale a X = X – Y.
*=	x *= y	Equivale a X = X * Y.
/=	x /= y	Equivale a X = X / Y.
%=	x %= y	Equivale a X = X % Y.
++	x++	Equivale a X = X + 1.
++	y = ++x	Equivale a X = X + 1 e depois Y = X.
++	y = x++	Equivale a Y = X e depois X = X +1.
––	x––	Equivale a X = X – 1.
––	y = ––x	Equivale a X = X – 1 e depois Y = X.
––	y = x––	Equivale a Y = X e depois X = X – 1.

Os operadores matemáticos de atribuição são utilizados para representar de maneira sintética uma operação aritmética e, posteriormente, uma operação de atribuição. Por exemplo, na tabela anterior o operador **+=** está sendo usado para realizar a operação **`x + y`** e, posteriormente, atribuir o resultado obtido à variável **`x`**.

Operador	Exemplo	Comentário
==	`x == y`	O conteúdo de X é igual ao conteúdo de Y.
!=	`x != y`	O conteúdo de X é diferente do conteúdo de Y.
<=	`x <= y`	O conteúdo de X é menor ou igual ao conteúdo de Y.
>=	`x >= y`	O conteúdo de X é maior ou igual ao conteúdo de Y.
<	`x < y`	O conteúdo de X é menor que o conteúdo de Y.
>	`x > y`	O conteúdo de X é maior que o conteúdo de Y.

Funções matemáticas		
Função	**Exemplo**	**Comentário**
`ceil`	`ceil(X)`	Arredonda um número real para cima. Por exemplo, ceil(3.2) é 4.
`cos`	`cos(X)`	Calcula o cosseno de X (X deve estar representado em radianos).
`exp`	`exp(X)`	Obtém o logaritmo natural e elevado à potência X.
`abs`	`abs(X)`	Obtém o valor absoluto de X.
`floor`	`floor(X)`	Arredonda um número real para baixo. Por exemplo, floor(3.2) é 3.
`log`	`log(X)`	Obtém o logaritmo natural de X.
`log10`	`log10(X)`	Obtém o logaritmo de base 10 de X.
`modf`	`z = modf(X,&Y)`	Decompõe o número real armazenado em X em duas partes: Y recebe a parte fracionária e z, a parte inteira do número.
`pow`	`pow(X,Y)`	Calcula a potência de X elevado a Y.
`sin`	`sin(X)`	Calcula o seno de X (X deve estar representado em radianos).
`sqrt`	`sqrt(X)`	Calcula a raiz quadrada de X.
`tan`	`tan(X)`	Calcula a tangente de X (X deve estar representado em radianos).

Observação

As funções `sin`, `cos` e `tan` esperam receber argumentos no formato de radianos; para receberem argumentos em graus, siga o exemplo a seguir.

Exemplo para o cálculo do seno de um ângulo fornecido em graus e utilizando uma variável para o valor de π:

```
VALORPI = 3.1415;
scanf("%f%*c",&X); //X EM GRAUS
Y = SIN ((VALORPI * X) / 180);
```

A linguagem C/C++ possui muitas outras funções matemáticas que podem ser observadas detalhadamente na documentação da biblioteca `math.h`.

Palavras reservadas são nomes utilizados pelo compilador para representar comandos de controle do programa, operadores e diretivas. As palavras reservadas da linguagem C/C++ são:

`asm`	`auto`	`break`	`case`	`cdecl`	`char`
`class`	`const`	`continue`	`_cs`	`default`	`delete`
`do`	`double`	`_ds`	`else`	`enum`	`_es`
`export`	`extern`	`far`	`_fastcall`	`float`	`friend`
`goto`	`huge`	`for`	`if`	`inline`	`int`
`interrupt`	`_loadds`	`long`	`near`	`new`	`operator`
`pascal`	`private`	`protected`	`public`	`register`	`return`
`_saveregs`	`_seg`	`short`	`signed`	`sizeof`	`_ss`
`static`	`struct`	`switch`	`template`	`this`	`typedef`
`union`	`unsigned`	`virtual`	`void`	`volatile`	`while`

3.4 Estrutura sequencial em JAVA

```
import nome_do_pacote_das_classes;
public class nome
{
      public static void main (String args[])
      {
        bloco_de_comandos;
      }
}
```

Os pacotes de classes são arquivos contendo diferentes classes que possuem vários métodos, ou seja, funções, os quais podem ser utilizados nos programas escritos em JAVA. A diretiva `import` permite que o programa reconheça as classes do pacote e, consequentemente, a utilização de seus métodos.

É importante salientar que a linguagem JAVA é sensível a letras maiúsculas e minúsculas, ou seja, considera letras maiúsculas diferentes de minúsculas (por exemplo, *a* é diferente de *A*). Sendo assim, cada comando tem a própria sintaxe, que, às vezes, é somente com letras minúsculas e outras vezes com letras maiúsculas e minúsculas.

3.4.1 Declaração de variáveis em JAVA

As *variáveis* são declaradas após a especificação de seus tipos. Os tipos de dados mais utilizados são: `int` (para números inteiros), `float` e `double` (para números reais), `char` (para um caractere), `String` (para vários caracteres) e `boolean` (para verdadeiro ou falso).

Exemplo:

```
float X;
```

Declara uma variável chamada `X` em que pode ser armazenado um número real.

```
double Y, Z;
```

Declara duas variáveis chamadas `Y` e `Z` em que podem ser armazenados dois números reais.

```
char SEXO;
```

Declara uma variável chamada `SEXO` em que pode ser armazenado um caractere.

```
String NOME;
```

Declara uma variável chamada `NOME` em que podem ser armazenados vários caracteres.

A linguagem JAVA possui os tipos primitivos de dados listados a seguir.

Tipo	Faixa de valores	Tamanho (aproximado)
`byte`	−128 a 127	8 bits
`char`	0 a 65.535	16 bits
`short`	−32.768 a 32.767	16 bits
`int`	−2.147.483.648 a 2.147.483.647	32 bits
`long`	−9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	64 bits
`float`	-3.4×10^{-38} a 3.4×10^{38}	32 bits
`double`	-1.7×10^{-308} a 1.7×10^{308}	64 bits
`boolean`	true ou false	indefinido

3.4.2 Declaração de constantes em JAVA

Constantes em JAVA podem ser declaradas em diferentes locais e isso define o seu escopo. Por escopo, pode-se entender as partes de um programa em que uma constante e também variáveis são compreendidas e podem ser utilizadas, sem acarretar erros de compilação. Sendo assim, se a constante for declarada fora de qualquer método (por exemplo, o `main`), ela terá escopo mais abrangente e poderá ser utilizada em qualquer ponto da classe. Se a constante for declarada dentro de um método, seu escopo ficará restrito a esse método.

A declaração de uma constante deve obedecer à seguinte sintaxe:

```
final tipo_da_constante nome_da_constante = valor_da_constante;
```

Exemplo:

```
final int X = 8;
```

Declaração da constante X, que é do tipo int, com valor 8.

```
final String NOME = "MARIA";
```

Declaração da constante NOME, que é do tipo String, com valor MARIA.

3.4.3 Comando de atribuição em JAVA

O *comando de atribuição* é utilizado para conceder valores ou operações às variáveis, sendo representado por = (sinal de igualdade).

Exemplo:

```
x = 4;
x = x + 2;
y = 2.5;
sexo = 'F';
```

Observações

Em JAVA, os caracteres são representados entre apóstrofos ('). As cadeias de caracteres devem ser representadas entre aspas (").

Nessa linguagem, cada comando é finalizado com o sinal de ponto e vírgula.

Em JAVA, os números reais tem a parte inteira separada da parte fracionária por um ponto.

3.4.4 Comando de entrada em JAVA

O *comando de entrada* é utilizado para receber dados digitados pelo usuário. Os dados recebidos são armazenados em variáveis. Uma das formas de entrada utilizada na linguagem JAVA é por meio da classe `Scanner`, que requer a importação do pacote `java.util`.

Exemplos:

```
int n1;
Scanner dado;
dado = new Scanner(System.in);
n1 = dado.nextInt();
```

Um valor inteiro digitado pelo usuário será armazenado na variável `n1`.

```
float x;
Scanner dado;
dado = new Scanner(System.in);
x = dado.nextFloat();
```

Um valor real digitado pelo usuário será armazenado na variável `x`.

```
String nome;
Scanner dado;
dado = new Scanner(System.in);
nome = dado.next();
```

Um valor literal digitado pelo usuário será armazenado na variável `nome`.

É importante salientar que todas as entradas são recebidas pela linguagem JAVA como um conjunto de caracteres. Assim, esses caracteres deverão ser convertidos por funções de conversão de tipos. Seguem algumas dessas funções.

Função	Funcionalidade
next()	Aguarda uma entrada em formato String com uma única palavra.
nextLine()	Aguarda uma entrada em formato String com uma ou várias palavras.
nextInt()	Aguarda uma entrada em formato inteiro.
nextByte()	Aguarda uma entrada em formato inteiro.
nextLong()	Aguarda uma entrada em formato inteiro.
nextFloat()	Aguarda uma entrada em formato número fracionário.
nextDouble()	Aguarda uma entrada em formato número fracionário.

O tratamento de cadeia de caracteres será mais detalhado no Capítulo 9.

3.4.5 Comando de saída em JAVA

O *comando de saída* é utilizado para mostrar dados na tela ou na impressora. Os comandos de saída mais utilizados na linguagem JAVA são `System.out.println` e `System.out.print`.

Exemplo:

```
System.out.println(X);
```

Mostra o valor armazenado na variável `x`.

```
System.out.println("Conteúdo de X = "  + X);
```

Mostra a mensagem "`Conteúdo de X =` " e, em seguida, o valor armazenado na variável `x`.

A diferença entre esses comandos é que o comando `System.out.println` mostra seu conteúdo e passa o cursor para a linha de baixo, enquanto o comando `System.out.print` mantém o cursor na mesma linha após mostrar a mensagem.

Exemplo:

```
System.out.print("aula ");
System.out.print("fácil");
```

Os comandos dados anteriormente geram a saída a seguir:
aula fácil

```
System.out.println("aula ");
System.out.println("fácil");
```

Os comandos dados anteriormente geram a saída a seguir:
aula
fácil

3.4.6 Comentários em JAVA

Comentários são textos que podem ser inseridos em programas com o objetivo de documentá-los. Eles são ignorados pelo interpretador.

Os comentários podem ocupar uma ou várias linhas, devendo ser inseridos nos programas utilizando-se os símbolos /* ..... */ ou //.

Exemplo:

```
/*
linhas de comentário
linhas de comentário
*/
```

A região de comentários é aberta com os símbolos /* e encerrada com os símbolos */.

```
// comentário
```

A região de comentários é aberta com os símbolos // e encerrada automaticamente ao final da linha.

3.4.7 Operadores e funções predefinidas em JAVA

A linguagem JAVA possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

Operador	Exemplo	Comentário
=	`x = y`	O conteúdo da variável Y é atribuído à variável X (A uma variável pode ser atribuído o conteúdo de outra variável, um valor constante ou, ainda, o resultado de uma função).
+	`x + y`	Soma o conteúdo de X e de Y.
−	`x − y`	Subtrai o conteúdo de Y do conteúdo de X.
*	`x * y`	Multiplica o conteúdo de X pelo conteúdo de Y.
/	`x / y`	Obtém o quociente da divisão de X por Y. Se os operandos são inteiros, o resultado da operação será o quociente inteiro da divisão. Se os operandos são reais, o resultado da operação será a divisão. Por exemplo: int z = 5/2; → a variável z receberá o valor 2. double z = 5.0/2.0; → a variável z receberá o valor 2.5.
%	`x % y`	Obtém o resto da divisão de X por Y.
+=	`x += y`	Equivale a X = X + Y.
−=	`x −= y`	Equivale a X = X − Y.
*=	`x *= y`	Equivale a X = X * Y.
/=	`x /= y`	Equivale a X = X / Y.
%=	`x %= y`	Equivale a X = X % Y.
++	`x++`	Equivale a X = X + 1.
++	`y = ++x`	Equivale a X = X + 1 e depois Y = X.
++	`y = x++`	Equivale a Y = X e depois X = X +1.
−−	`x−−`	Equivale a X = X − 1.
−−	`y = −−x`	Equivale a X = X − 1 e depois Y = X.
−−	`y = x−−`	Equivale a Y = X e depois X = X − 1.

Os operadores matemáticos de atribuição são utilizados para representar, de maneira sintética, uma operação aritmética e, posteriormente, uma operação de atribuição. Por exemplo, na tabela anterior, o operador += está sendo usado para realizar a operação `x + y` e, posteriormente, atribuir o resultado obtido à variável `x`.

Operador	Exemplo	Comentário
==	`x == y`	O conteúdo de X é igual ao conteúdo de Y.
!=	`x != y`	O conteúdo de X é diferente do conteúdo de Y.
<=	`x <= y`	O conteúdo de X é menor ou igual ao conteúdo de Y.
>=	`x >= y`	O conteúdo de X é maior ou igual ao conteúdo de Y.
<	`x < y`	O conteúdo de X é menor que o conteúdo de Y.
>	`x > y`	O conteúdo de X é maior que o conteúdo de Y.

Funções matemáticas		
Função	**Exemplo**	**Comentário**
`ceil`	`Math.ceil(X)`	Arredonda um número real para cima. Por exemplo, ceil(3.2) é 4.
`cos`	`Math.cos(X)`	Calcula o cosseno de X (X deve estar representado em radianos).
`exp`	`Math.exp(X)`	Obtém o logaritmo natural e elevado à potência X.
`abs`	`Math.abs(X)`	Obtém o valor absoluto de X.
`floor`	`Math.floor(X)`	Arredonda um número real para baixo. Por exemplo, floor(3.2) é 3.
`log`	`Math.log(X)`	Obtém o logaritmo natural de X.
`log10`	`Math.log10(X)`	Obtém o logaritmo de base 10 de X.
`pow`	`Math.pow(X,Y)`	Calcula a potência de X elevado a Y.
`sin`	`Math.sin(X)`	Calcula o seno de X (X deve estar representado em radianos).
`sqrt`	`Math.sqrt(X)`	Calcula a raiz quadrada de X.
`cbrt`	`Math.cbrt(X)`	Calcula a raiz cúbica de X.
`tan`	`Math.tan(X)`	Calcula a tangente de X (X deve estar representado em radianos).
`PI`	`Math.PI`	Retorna o valor de π.
`toDegrees`	`Math.toDegrees(X)`	Converte a medida de X de radianos para graus.
`toRadians`	`Math.toRadians(X)`	Converte a medida de X de graus para radianos.

Observação

Os métodos `sin`, `cos` e `tan` esperam receber argumentos no formato de radianos; para receberem argumentos em graus, siga o próximo exemplo.

```
dado = new Scanner(System.in);
x = dado.nextDouble();
y = Math.sin(Math.toRadians(x));
```

A linguagem JAVA possui muitas outras funções matemáticas que podem ser observadas detalhadamente na documentação da classe `Math`.

Palavras reservadas são nomes utilizados pela linguagem para representar comandos de controle do programa, operadores e diretivas.				
`abstract`	`continue`	`for`	`new`	`switch`
`assert`	`default`	`if`	`package`	`synchronized`
`boolean`	`do`	`goto`	`private`	`this`
`break`	`double`	`implements`	`protected`	`throw`
`byte`	`else`	`import`	`public`	`throws`
`case`	`enum`	`instanceof`	`return`	`transient`
`catch`	`extends`	`int`	`short`	`try`
`char`	`final`	`interface`	`static`	`void`
`class`	`finally`	`long`	`strictfp`	`volatile`
`const`	`float`	`native`	`super`	`while`

Tem-se ainda em JAVA os literais reservados e que, assim como as palavras reservadas, não podem ser usados como identificadores, pois apresentarão erro de compilação. São eles: `null`, `true` e `false`.

EXERCÍCIOS RESOLVIDOS

1. Faça um programa que receba quatro números inteiros, calcule e mostre a soma desses números.

ALGORITMO Solução:

```
ALGORITMO
     DECLARE n1, n2, n3, n4, soma NUMÉRICO
     LEIA n1, n2, n3, n4
     soma ← n1 + n2 + n3 + n4
     ESCREVA soma
FIM_ALGORITMO.
```

1ª solução:

\EXERC\CAP3\PASCAL\EX1_A.PAS **e** \EXERC\CAP3\PASCAL\EX1_A.EXE

2ª solução:

\EXERC\CAP3\PASCAL\EX1_B.PAS **e** \EXERC\CAP3\PASCAL\EX1_B.EXE

1ª solução:

\EXERC\CAP3\C++\EX1_A.CPP **e** \EXERC\CAP3\C++\EX1_A.EXE

2ª solução:

\EXERC\CAP3\C++\EX1_B.CPP **e** \EXERC\CAP3\C++\EX1_B.EXE

1ª solução:

\EXERC\CAP3\JAVA\EX1_A.java **e** \EXERC\CAP3\JAVA\EX1_A.class

2ª solução:

\EXERC\CAP3\JAVA\EX1_B.java **e** \EXERC\CAP3\JAVA\EX1_BA.class

2. Faça um programa que receba três notas, calcule e mostre a média aritmética.

ALGORITMO 1ª solução:

```
ALGORITMO
     DECLARE nota1, nota2, nota3, media NUMÉRICO
     LEIA nota1, nota2, nota3
     media ← (nota1 + nota2 + nota3)/3
     ESCREVA media
FIM_ALGORITMO.
```

2ª solução:

```
ALGORITMO
     DECLARE nota1, nota2, nota3, soma, media NUMÉRICO
     LEIA nota1, nota2, nota3
     soma ← nota1 + nota2 + nota3
     media ← soma/3
     ESCREVA media
FIM_ALGORITMO.
```

1ª solução:

\EXERC\CAP3\PASCAL\EX2_A.PAS **e** \EXERC\CAP3\PASCAL\EX2_A.EXE

2ª solução:

\EXERC\CAP3\PASCAL\EX2_B.PAS **e** \EXERC\CAP3\PASCAL\EX2_B.EXE

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números. Se isso não for feito, eles serão apresentados com formatação científica.

Exemplo de números com formatação científica:

```
1.5000000000E+04 = 15000
7.0000000000E+00 = 7
```

Exemplo de formatação:

```
X:6:2  A variável X será mostrada com seis caracteres: dois caracteres para a parte fracionária, um caractere para o ponto e os outros três caracteres restantes para a parte inteira.
Y:8:3  A variável Y será mostrada com oito caracteres: três caracteres para a parte fracionária, um caractere para o ponto e os outros quatro caracteres restantes para a parte inteira.
```

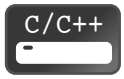
Variável: número total de caracteres: número de caracteres da parte fracionária

O primeiro parâmetro da formatação corresponde ao número total de caracteres mostrados na tela; o segundo, ao total de caracteres ocupados pela parte fracionária. O ponto, que é o separador entre a parte inteira e fracionária, também ocupa um caractere.

Os caracteres à direita serão preenchidos com zeros e os caracteres à esquerda serão preenchidos com espaços em branco.

3ª SOLUÇÃO:

```
\EXERC\CAP3\PASCAL\EX2_C.PAS e \EXERC\CAP3\PASCAL\EX2_C.EXE
```



1ª SOLUÇÃO:

```
\EXERC\CAP3\C++\EX2_A.CPP e \EXERC\CAP3\C++\EX2_A.EXE
```

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números para definir quantas casas decimais devem ser mostradas. Assim, no comando de saída `printf`, a formatação é especificada imediatamente antes da letra que define o tipo da variável que será mostrada.

Exemplo:

```
printf("Conteúdo de variável X é: %6.3f",X);
```

No comando dado anteriormente, `%f` especifica que será mostrado um número real, e 6.3 significa que serão utilizados seis caracteres para mostrar o número, e, destes, três caracteres para a parte fracionária, um caractere para o ponto e os outros dois caracteres restantes para a parte inteira.

2ª SOLUÇÃO:

```
\EXERC\CAP3\C++\EX2_B.CPP e \EXERC\CAP3\C++\EX2_B.EXE
```

3ª SOLUÇÃO:

```
\EXERC\CAP3\C++\EX2_C.CPP e \EXERC\CAP3\C++\EX2_C.EXE
```

1ª SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX2_A.java e \EXERC\CAP3\JAVA\EX2_A.class
```

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números para definir a quantidade de casas decimais que devem ser mostradas.

Deve-se utilizar o método `DecimalFormat`, conforme apresentado a seguir:

```
DecimalFormat casas;
casas = new DecimalFormat("0.00");
System.out.println("Média = "+casas.format(media));
```

No exemplo anterior, a formatação permitirá que sejam mostradas duas casas decimais para o valor da variável média. Para a utilização do método `DecimalFormat`, deve-se incluir o pacote de classes `text`, ou seja, `import java.text.*;`

2ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX2_B.java **e** \EXERC\CAP3\JAVA\EX2_B.class

3ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX2_C.java **e** \EXERC\CAP3\JAVA\EX2_C.class

3. Faça um programa que receba três notas e seus respectivos pesos, calcule e mostre a média ponderada.

ALGORITMO 1ª SOLUÇÃO:

```
ALGORITMO
  DECLARE nota1, nota2, nota3, peso1, peso2, peso3, media NUMÉRICO
  LEIA nota1, nota2, nota3, peso1, peso2, peso3
  media←(nota1*peso1+nota2*peso2+nota3*peso3)/(peso1+peso2+peso3)
  ESCREVA media
FIM_ALGORITMO.
```

2ª SOLUÇÃO:

```
ALGORITMO
  DECLARE nota1, nota2, nota3, peso1, peso2, peso3 NUMÉRICO
          soma1, soma2, soma3, total, media NUMÉRICO
  LEIA nota1, nota2, nota3, peso1, peso2, peso3
  soma1 ← nota1 * peso1
  soma2 ← nota2 * peso2
  soma3 ← nota3 * peso3
  total ← peso1 + peso2 + peso3
  media ←(soma1 + soma2 + soma3)/total
  ESCREVA media
FIM_ALGORITMO.
```

PASCAL 1ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX3_A.PAS **e** \EXERC\CAP3\PASCAL\EX3_A.EXE

2ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX3_B.PAS **e** \EXERC\CAP3\PASCAL\EX3_B.EXE

3ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX3_C.PAS **e** \EXERC\CAP3\PASCAL\EX3_C.EXE

C/C++ 1ª SOLUÇÃO:

\EXERC\CAP3\C++\EX3_A.CPP **e** \EXERC\CAP3\C++\EX3_A.EXE

2ª SOLUÇÃO:

\EXERC\CAP3\C++\EX3_B.CPP **e** \EXERC\CAP3\C++\EX3_B.EXE

3ª SOLUÇÃO:

\EXERC\CAP3\C++\EX3_C.CPP **e** \EXERC\CAP3\C++\EX3_C.EXE

JAVA 1ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX3_A.java **e** \EXERC\CAP3\JAVA\EX3_A.class

2ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX3_B.java **e** \EXERC\CAP3\JAVA\EX3_B.class

3ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX3_C.java **e** \EXERC\CAP3\JAVA\EX3_C.class

4. Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%.

ALGORITMO 1ª SOLUÇÃO:

```
ALGORITMO
  DECLARE sal, novosal NUMÉRICO
  LEIA sal
  novosal ← sal + sal * 25/100
  ESCREVA novosal
FIM_ALGORITMO.
```

2ª SOLUÇÃO:

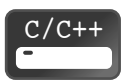
```
ALGORITMO
  DECLARE sal, aumento, novosal NUMÉRICO
  LEIA sal
  aumento ← sal * 25/100
  novosal ← sal + aumento
  ESCREVA novosal
FIM_ALGORITMO.
```

1ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX4_A.PAS **e** \EXERC\CAP3\PASCAL\EX4_A.EXE

2ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX4_B.PAS **e** \EXERC\CAP3\PASCAL\EX4_B.EXE



1ª SOLUÇÃO:

\EXERC\CAP3\C++\EX4_A.CPP **e** \EXERC\CAP3\C++\EX4_A.EXE

2ª SOLUÇÃO:

\EXERC\CAP3\C++\EX4_B.CPP **e** \EXERC\CAP3\C++\EX4_B.EXE

1ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX4_A.java **e** \EXERC\CAP3\JAVA\EX4_A.class

2ª SOLUÇÃO:

\EXERC\CAP3\JAVA\EX4_B.java **e** \EXERC\CAP3\JAVA\EX4_B.class

5. Faça um programa que receba o salário de um funcionário e o percentual de aumento, calcule e mostre o valor do aumento e o novo salário.

ALGORITMO SOLUÇÃO:

```
ALGORITMO
  DECLARE sal, perc, aumento, novosal NUMÉRICO
  LEIA sal, perc
  aumento ← sal * perc/100
  ESCREVA aumento
  novosal ← sal + aumento
  ESCREVA novosal
FIM_ALGORITMO.
```

SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX5.PAS **e** \EXERC\CAP3\PASCAL\EX5.EXE

SOLUÇÃO:

\EXERC\CAP3\C++\EX5.CPP **e** \EXERC\CAP3\C++\EX5.EXE

Solução:

```
\EXERC\CAP3\JAVA\EX5.java e \EXERC\CAP3\JAVA\EX5.class
```

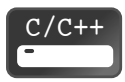
6. Faça um programa que receba o salário base de um funcionário, calcule e mostre o salário a receber, sabendo-se que o funcionário tem gratificação de 5% sobre o salário base e paga imposto de 7% também sobre o salário base.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE sal, salreceber, grat, imp NUMÉRICO
  LEIA sal
  grat ← sal * 5/100
  imp ← sal * 7/100
  salreceber ← sal + grat − imp
  ESCREVA salreceber
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX6.PAS e \EXERC\CAP3\PASCAL\EX6.EXE
```



Solução:

```
\EXERC\CAP3\C++\EX6.CPP e \EXERC\CAP3\C++\EX6.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX6.java e \EXERC\CAP3\JAVA\EX6.class
```

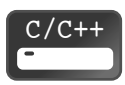
7. Faça um programa que receba o salário base de um funcionário, calcule e mostre seu salário a receber, sabendo-se que o funcionário tem gratificação de R$ 50 e paga imposto de 10% sobre o salário base.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE sal, salreceber, imp NUMÉRICO
  LEIA sal
  imp ← sal * 10/100
  salreceber ← sal + 50 − imp
  ESCREVA salreceber
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX7.PAS e \EXERC\CAP3\PASCAL\EX7.EXE
```



Solução:

```
\EXERC\CAP3\C++\EX7.CPP e \EXERC\CAP3\C++\EX7.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX7.java e \EXERC\CAP3\JAVA\EX7.class
```

8. Faça um programa que receba o valor de um depósito e o valor da taxa de juros, calcule e mostre o valor do rendimento e o valor total depois do rendimento.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE dep, taxa, rend, total NUMÉRICO
  LEIA dep, taxa
```

```
  rend ← dep * taxa/100
  total ← dep + rend
  ESCREVA rend
  ESCREVA total
FIM_ALGORITMO.
```

Solução:

\EXERC\CAP3\PASCAL\EX8.PAS **e** \EXERC\CAP3\PASCAL\EX8.EXE

Solução:

\EXERC\CAP3\C++\EX8.CPP **e** \EXERC\CAP3\C++\EX8.EXE

Solução:

\EXERC\CAP3\JAVA\EX8.java **e** \EXERC\CAP3\JAVA\EX8.class

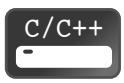
9. Faça um programa que calcule e mostre a área de um triângulo. Sabe-se que: Área = (base * altura)/2.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE base, altura, area NUMÉRICO
  LEIA base, altura
  area ← (base * altura)/2
  ESCREVA area
FIM_ALGORITMO.
```

Solução:

\EXERC\CAP3\PASCAL\EX9.PAS **e** \EXERC\CAP3\PASCAL\EX9.EXE



Solução:

\EXERC\CAP3\C++\EX9.CPP **e** \EXERC\CAP3\C++\EX9.EXE

Solução:

\EXERC\CAP3\JAVA\EX9.java **e** \EXERC\CAP3\JAVA\EX9.class

10. Faça um programa que calcule e mostre a área de um círculo. Sabe-se que: Área = π * R^2.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE area, raio NUMÉRICO
  LEIA raio
  area ← 3.1415 * raio²
  ESCREVA area
FIM_ALGORITMO.
```

1ª solução:

\EXERC\CAP3\PASCAL\EX10_A.PAS **e** \EXERC\CAP3\PASCAL\EX10_A.EXE

2ª solução:

\EXERC\CAP3\PASCAL\EX10_B.PAS **e** \EXERC\CAP3\PASCAL\EX10_B.EXE

3ª SOLUÇÃO:

```
\EXERC\CAP3\PASCAL\EX10_C.PAS e \EXERC\CAP3\PASCAL\EX10_C.EXE
```

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 3.2.7.

1ª SOLUÇÃO:

```
\EXERC\CAP3\C++\EX10_A.CPP e \EXERC\CAP3\C++\EX10_A.EXE
```

2ª SOLUÇÃO:

```
\EXERC\CAP3\C++\EX10_B.CPP e \EXERC\CAP3\C++\EX10_B.EXE
```

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 3.3.7.

1ª SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX10_A.java e \EXERC\CAP3\JAVA\EX10_A.class
```

2ª SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX10_B.java e \EXERC\CAP3\JAVA\EX10_B.class
```

Esse programa usou alguns métodos da linguagem JAVA que estão descritos na Seção 3.4.7.

11. Faça um programa que receba um número positivo e maior que zero, calcule e mostre:

a) o número digitado ao quadrado;
b) o número digitado ao cubo;
c) a raiz quadrada do número digitado;
d) a raiz cúbica do número digitado.

SOLUÇÃO:

```
ALGORITMO
  DECLARE num, quad, cubo, r2, r3 NUMÉRICO
  LEIA num
  quad ← num²
  cubo ← num³
  r2 ← ∛num
  r3 ← ∜num
  ESCREVA quad, cubo, r2, r3
FIM_ALGORITMO.
```

SOLUÇÃO:

```
\EXERC\CAP3\PASCAL\EX11.PAS e \EXERC\CAP3\PASCAL\EX11.EXE
```

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 3.2.7.

SOLUÇÃO:

```
\EXERC\CAP3\C++\EX11.CPP e \EXERC\CAP3\C++\EX11.EXE
```

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 3.3.7.

SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX11.java e \EXERC\CAP3\JAVA\EX11.class
```

Esse programa usou alguns métodos da linguagem JAVA que estão descritos na Seção 3.4.7.

12. Faça um programa que receba dois números maiores que zero, calcule e mostre um elevado ao outro.

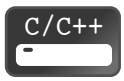
ALGORITMO Solução:

```
ALGORITMO
  DECLARE num1, num2, r1, r2 NUMÉRICO
  LEIA num1, num2
  r1 ← num1^num2
  r2 ← num2^num1
  ESCREVA r1, r2
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX12.PAS e \EXERC\CAP3\PASCAL\EX12.EXE
```

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 3.2.7.



Solução:

```
\EXERC\CAP3\C++\EX12.CPP e \EXERC\CAP3\C++\EX12.EXE
```

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 3.3.7.

Solução:

```
\EXERC\CAP3\JAVA\EX12.java e \EXERC\CAP3\JAVA\EX12.class
```

Esse programa usou alguns métodos da linguagem JAVA que estão descritos na Seção 3.4.7.

13. Sabe-se que:
pé = 12 polegadas
1 jarda = 3 pés
1 milha = 1,760 jarda
Faça um programa que receba uma medida em pés, faça as conversões a seguir e mostre os resultados.

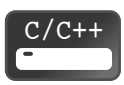
a) polegadas;
b) jardas;
c) milhas.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE pes, polegadas, jardas, milhas NUMÉRICO
  LEIA pes
  polegadas ← pes * 12
  jardas ← pes / 3
  milhas ← jardas / 1760
  ESCREVA polegadas, jardas, milhas
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX13.PAS e \EXERC\CAP3\PASCAL\EX13.EXE
```



Solução:

```
\EXERC\CAP3\C++\EX13.CPP e \EXERC\CAP3\C++\EX13.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX13.java e \EXERC\CAP3\JAVA\EX13.class
```

14. Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:
a) a idade dessa pessoa;
b) quantos anos ela terá em 2050.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE ano_atual, ano_nascimento, idade_atual, idade_2050 NUMÉRICO
  LEIA ano_atual
  LEIA ano_nascimento
  idade_atual ← ano_atual − ano_nascimento
  idade_2050 ← 2050 − ano_nascimento
  ESCREVA idade_atual
  ESCREVA idade_2050
FIM_ALGORITMO.
```

PASCAL Solução:

```
\EXERC\CAP3\PASCAL\EX14.PAS e \EXERC\CAP3\PASCAL\EX14.EXE
```

C/C++ Solução:

```
\EXERC\CAP3\C++\EX14.CPP e \EXERC\CAP3\C++\EX14.EXE
```

JAVA Solução:

```
\EXERC\CAP3\JAVA\EX14.java e \EXERC\CAP3\JAVA\EX14.class
```

15. O custo ao consumidor de um carro novo é a soma do preço de fábrica com o percentual de lucro do distribuidor e dos impostos aplicados ao preço de fábrica. Faça um programa que receba o preço de fábrica de um veículo, o percentual de lucro do distribuidor e o percentual de impostos, calcule e mostre:
a) o valor correspondente ao lucro do distribuidor;
b) o valor correspondente aos impostos;
c) o preço final do veículo.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE p_fab, perc_d, perc_i, vlr_d, vlr_i, p_final NUMÉRICO
  LEIA p_fab
  LEIA perc_d
  LEIA perc_i
  vlr_d ← p_fab * perc_d / 100
  vlr_i ← p_fab * perc_i / 100
  p_final ← p_fab + vlr_d + vlr_i
  ESCREVA vlr_d
  ESCREVA vlr_i
  ESCREVA p_final
FIM_ALGORITMO.
```

PASCAL Solução:

```
\EXERC\CAP3\PASCAL\EX15.PAS e \EXERC\CAP3\PASCAL\EX15.EXE
```

C/C++ Solução:

```
\EXERC\CAP3\C++\EX15.CPP e \EXERC\CAP3\C++\EX15.EXE
```

Solução:

\EXERC\CAP3\JAVA\EX15.java **e** \EXERC\CAP3\JAVA\EX15.class

16. Faça um programa que receba o número de horas trabalhadas e o valor do salário mínimo, calcule e mostre o salário a receber, seguindo estas regras:

a) a hora trabalhada vale a metade do salário mínimo.
b) o salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada.
c) o imposto equivale a 3% do salário bruto.
d) o salário a receber equivale ao salário bruto menos o imposto.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE horas_t, vlr_sal_min, vlr_hora_t NUMÉRICO
          vlr_sal_bru, imp, vlr_sal_liq NUMÉRICO
  LEIA horas_t
  LEIA vlr_sal_min
  vlr_hora_t ← vlr_sal_min / 2
  vlr_sal_bru ← vlr_hora_t * horas_t
  imp ← vlr_sal_bru * 3 / 100
  vlr_sal_liq ← vlr_sal_bru – imp
  ESCREVA vlr_sal_liq
FIM_ALGORITMO.
```

Solução:

\EXERC\CAP3\PASCAL\EX16.PAS **e** \EXERC\CAP3\PASCAL\EX16.EXE

Solução:

\EXERC\CAP3\C++\EX16.CPP **e** \EXERC\CAP3\C++\EX16.EXE

Solução:

\EXERC\CAP3\JAVA\EX16.java **e** \EXERC\CAP3\JAVA\EX16.class

17. Um trabalhador recebeu seu salário e o depositou em sua conta bancária. Esse trabalhador emitiu dois cheques e agora deseja saber seu saldo atual. Sabe-se que cada operação bancária de retirada paga CPMF de 0,38% e o saldo inicial da conta está zerado.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE salario, cheque1, cheque2, cpmf1, cpmf2, saldo NUMÉRICO
  LEIA salario
  LEIA cheque1
  LEIA cheque2
  cpmf1 ← cheque1 * 0.38 / 100
  cpmf2 ← cheque2 * 0.38 / 100
  saldo ← salario – cheque1 – cheque2 – cpmf1 – cpmf2
  ESCREVA saldo
FIM_ALGORITMO.
```

Solução:

\EXERC\CAP3\PASCAL\EX17.PAS **e** \EXERC\CAP3\PASCAL\EX17.EXE

Solução:

```
\EXERC\CAP3\C++\EX17.CPP e \EXERC\CAP3\C++\EX17.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX17.java e \EXERC\CAP3\JAVA\EX17.class
```

18. Pedro comprou um saco de ração com peso em quilos. Ele possui dois gatos, para os quais fornece a quantidade de ração em gramas. A quantidade diária de ração fornecida para cada gato é sempre a mesma. Faça um programa que receba o peso do saco de ração e a quantidade de ração fornecida para cada gato, calcule e mostre quanto restará de ração no saco após cinco dias.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE peso_saco, racao_gato1, racao_gato2, total_final NUMÉRICO
  LEIA peso_saco
  LEIA racao_gato1
  LEIA racao_gato2
  racao_gato1 ← racao_gato1 / 1000
  racao_gato2 ← racao_gato2 / 1000
  total_final ← peso_saco − 5 * (racao_gato1 + racao_gato2)
  ESCREVA total_final
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX18.PAS e \EXERC\CAP3\PASCAL\EX18.EXE
```

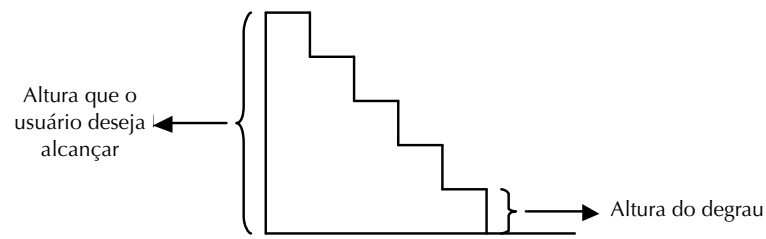
Solução:

```
\EXERC\CAP3\C++\EX18.CPP e \EXERC\CAP3\C++\EX18.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX18.java e \EXERC\CAP3\JAVA\EX18.class
```

19. Cada degrau de uma escada tem X de altura. Faça um programa que receba essa altura e a altura que o usuário deseja alcançar subindo a escada, calcule e mostre quantos degraus ele deverá subir para atingir seu objetivo, sem se preocupar com a altura do usuário. Todas as medidas fornecidas devem estar em metros.



ALGORITMO Solução:

```
ALGORITMO
  DECLARE a_degrau, a_usuario, qtd_degraus NUMÉRICO
  LEIA a_degrau
  LEIA a_usuario
  qtd_degraus ← a_usuario / a_degrau
  ESCREVA qtd_degraus
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX19.PAS e \EXERC\CAP3\PASCAL\EX19.EXE
```

Solução:

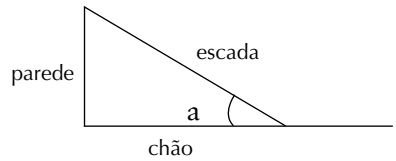
```
\EXERC\CAP3\C++\EX19.CPP e \EXERC\CAP3\C++\EX19.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX19.java e \EXERC\CAP3\JAVA\EX19.class
```

20. Faça um programa que receba a medida do ângulo (em graus) formado por uma escada apoiada no chão e encostada na parede e a altura da parede onde está a ponta da escada. Calcule e mostre a medida dessa escada.

Observação: as funções trigonométricas implementadas nas linguagens de programação trabalham com medidas de ângulos em radianos.

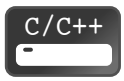


ALGORITMO Solução:

```
ALGORITMO
  DECLARE ang, alt, escada, radiano NUMÉRICO
  LEIA ang
  LEIA alt
  radiano ← ang * 3.14 / 180
  escada ← alt / seno(radiano)
  ESCREVA escada
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX20.PAS e \EXERC\CAP3\PASCAL\EX20.EXE
```



Solução:

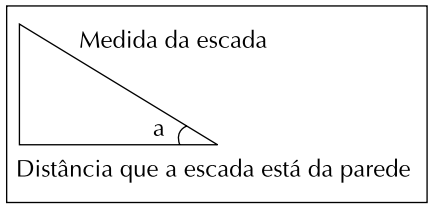
```
\EXERC\CAP3\C++\EX20.CPP e \EXERC\CAP3\C++\EX20.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX20.java e \EXERC\CAP3\JAVA\EX20.class
```

21. Uma pessoa deseja pregar um quadro em uma parede. Faça um programa para calcular e mostrar a que distância a escada deve estar da parede. A pessoa deve fornecer o tamanho da escada e a altura em que deseja pregar o quadro.

Lembre-se de que o tamanho da escada deve ser maior que a altura que se deseja alcançar.



X – Altura em que deseja pregar o quadro

Y – Distância em que deverá ficar a escada

Z – Tamanho da escada

ALGORITMO Solução:

```
ALGORITMO
  DECLARE X, Y, Z NUMÉRICO
  LEIA Z
  LEIA X
  Y ← Z² − X²
  Y ← ²√Y
  ESCREVA Y
FIM_ALGORITMO.
```

PASCAL Solução:

```
\EXERC\CAP3\PASCAL\EX21.PAS e \EXERC\CAP3\PASCAL\EX21.EXE
```

C/C++ Solução:

```
\EXERC\CAP3\C++\EX21.CPP e \EXERC\CAP3\C++\EX21.EXE
```

JAVA Solução:

```
\EXERC\CAP3\JAVA\EX21.java e \EXERC\CAP3\JAVA\EX21.class
```

22. Sabe-se que o quilowatt de energia custa um quinto do salário mínimo. Faça um programa que receba o valor do salário mínimo e a quantidade de quilowatts consumida por uma residência. Calcule e mostre:

a) o valor de cada quilowatt;
b) o valor a ser pago por essa residência;
c) o valor a ser pago com desconto de 15%.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE vlr_sal, qtd_kw, vlr_kw, vlr_reais, desc, vlr_desc NUMÉRICO
  LEIA vlr_sal
  LEIA qtd_kw
  vlr_kw ← vlr_sal / 5
  vlr_reais ← vlr_kw * qtd_kw
  desc ← vlr_reais * 15 / 100
  vlr_desc ← vlr_reais − desc
  ESCREVA vlr_kw
  ESCREVA vlr_reais
  ESCREVA vlr_desc
FIM_ALGORITMO.
```

PASCAL Solução:

```
\EXERC\CAP3\PASCAL\EX22.PAS e \EXERC\CAP3\PASCAL\EX22.EXE
```

Solução:

```
\EXERC\CAP3\C++\EX22.CPP e \EXERC\CAP3\C++\EX22.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX22.java e \EXERC\CAP3\JAVA\EX22.class
```

23. Faça um programa que receba um número real, encontre e mostre:

a) a parte inteira desse número;
b) a parte fracionária desse número;
c) o arredondamento desse número.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE num, i, f, a NUMÉRICO
  LEIA num
  i ← parte inteira de num
  f ← num − i
  a ← arredonda (num)
  ESCREVA i
  ESCREVA f
  ESCREVA a
FIM_ALGORITMO.
```

Solução (Arredondando o número como na matemática):

```
\EXERC\CAP3\PASCAL\EX23.PAS e \EXERC\CAP3\PASCAL\EX23.EXE
```

1ª solução (arredondando o número para cima):

```
\EXERC\CAP3\C++\EX23_A.CPP e \EXERC\CAP3\C++\EX23_A.EXE
```

2ª solução (arredondando o número para baixo):

```
\EXERC\CAP3\C++\EX23_B.CPP e \EXERC\CAP3\C++\EX23_B.EXE
```

1ª solução (arredondando o número para cima):

```
\EXERC\CAP3\JAVA\EX23_A.java e \EXERC\CAP3\JAVA\EX23_A.class
```

2ª solução (arredondando o número para baixo):

```
\EXERC\CAP3\JAVA\EX23_B.java e \EXERC\CAP3\JAVA\EX23_B.class
```

24. Faça um programa que receba uma hora formada por hora e minutos (um número real), calcule e mostre a hora digitada apenas em minutos. Lembre-se de que:

- para quatro e meia, deve-se digitar 4.30;
- os minutos vão de 0 a 59.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE hora, h, m, conversao NUMÉRICO
  LEIA hora
  h ← pegar a parte inteira da variável hora
  m ← hora − h
  conversao ← (h * 60) + (m * 100)
  ESCREVA conversao
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX24.PAS e \EXERC\CAP3\PASCAL\EX24.EXE
```

Solução:

```
\EXERC\CAP3\C++\EX24.CPP e \EXERC\CAP3\C++\EX24.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX24.java e \EXERC\CAP3\JAVA\EX24.class
```

25. Faça um programa que receba o custo de um espetáculo teatral e o preço do convite desse espetáculo. Esse programa deverá calcular e mostrar a quantidade de convites que devem ser vendidos para que, pelo menos, o custo do espetáculo seja alcançado.

ALGORITMO Solução:

```
ALGORITMO
  DECLARE custo, convite, qtd NUMÉRICO
  LEIA custo
  LEIA convite
  qtd ← custo / convite
  ESCREVA qtd
FIM_ALGORITMO.
```

Solução:

```
\EXERC\CAP3\PASCAL\EX25.PAS e \EXERC\CAP3\PASCAL\EX25.EXE
```

Solução:

```
\EXERC\CAP3\C++\EX25.CPP e \EXERC\CAP3\C++\EX25.EXE
```

Solução:

```
\EXERC\CAP3\JAVA\EX25.java e \EXERC\CAP3\JAVA\EX25.class
```

EXERCÍCIOS PROPOSTOS

1. Faça um programa que receba dois números, calcule e mostre a subtração do primeiro número pelo segundo.

2. Faça um programa que receba três números, calcule e mostre a multiplicação desses números.

3. Faça um programa que receba dois números, calcule e mostre a divisão do primeiro número pelo segundo. Sabe-se que o segundo número não pode ser zero, portanto, não é necessário se preocupar com validações.

4. Faça um programa que receba duas notas, calcule e mostre a média ponderada dessas notas, considerando peso 2 para a primeira e peso 3 para a segunda.

5. Faça um programa que receba o preço de um produto, calcule e mostre o novo preço, sabendo-se que este sofreu um desconto de 10%.

6. Um funcionário recebe um salário fixo mais 4% de comissão sobre as vendas. Faça um programa que receba o salário fixo do funcionário e o valor de suas vendas, calcule e mostre a comissão e seu salário final.

7. Faça um programa que receba o peso de uma pessoa, calcule e mostre:

a) o novo peso, se a pessoa engordar 15% sobre o peso digitado;

b) o novo peso, se a pessoa emagrecer 20% sobre o peso digitado.

8. Faça um programa que receba o peso de uma pessoa em quilos, calcule e mostre esse peso em gramas.

9. Faça um programa que calcule e mostre a área de um trapézio.

Sabe-se que: A = ((base maior + base menor) * altura)/2

10. Faça um programa que calcule e mostre a área de um quadrado. Sabe-se que: A = lado * lado.

11. Faça um programa que calcule e mostre a área de um losango. Sabe-se que: A = (diagonal maior * diagonal menor)/2.

12. Faça um programa que receba o valor do salário mínimo e o valor do salário de um funcionário, calcule e mostre a quantidade de salários mínimos que esse funcionário ganha.

13. Faça um programa que calcule e mostre a tabuada de um número digitado pelo usuário.

Exemplo:

Digite um número: *5*

$5 \times 0 = 0$
$5 \times 1 = 5$
$5 \times 2 = 10$
$5 \times 3 = 15$
$5 \times 4 = 20$
$5 \times 5 = 25$
$5 \times 6 = 30$
$5 \times 7 = 35$
$5 \times 8 = 40$
$5 \times 9 = 45$
$5 \times 10 = 50$

14. Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:

a) a idade dessa pessoa em anos;
b) a idade dessa pessoa em meses;
c) a idade dessa pessoa em dias;
d) a idade dessa pessoa em semanas.

15. João recebeu seu salário e precisa pagar duas contas atrasadas. Em razão do atraso, ele deverá pagar multa de 2% sobre cada conta. Faça um programa que calcule e mostre quanto restará do salário de João.

16. Faça um programa que receba o valor dos catetos de um triângulo, calcule e mostre o valor da hipotenusa.

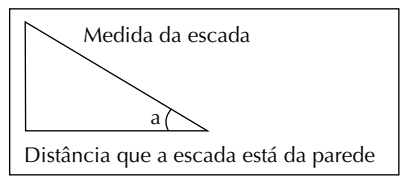
17. Faça um programa que receba o raio, calcule e mostre:

a) o comprimento de uma esfera; sabe-se que $C = 2 * \pi R$;
b) a área de uma esfera; sabe-se que $A = \pi R^2$;
c) o volume de uma esfera; sabe-se que $V = ¾ * \pi R^3$.

18. Faça um programa que receba uma temperatura em Celsius, calcule e mostre essa temperatura em Fahrenheit. Sabe-se que F = 180*(C + 32)/100.

19. Sabe-se que, para iluminar de maneira correta os cômodos de uma casa, para cada m^2, deve-se usar 18 W de potência. Faça um programa que receba as duas dimensões de um cômodo (em metros), calcule e mostre a sua área (em m^2) e a potência de iluminação que deverá ser utilizada.

20. Faça um programa que receba a medida do ângulo formado por uma escada apoiada no chão e a distância em que a escada está da parede, calcule e mostre a medida da escada para que se possa alcançar sua ponta.



21. Faça um programa que receba o número de horas trabalhadas, o valor do salário mínimo e o número de horas extras trabalhadas, calcule e mostre o salário a receber, de acordo com as regras a seguir:

a) a hora trabalhada vale 1/8 do salário mínimo;
b) a hora extra vale 1/4 do salário mínimo;

c) o salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada;
d) a quantia a receber pelas horas extras equivale ao número de horas extras trabalhadas multiplicado pelo valor da hora extra;
e) o salário a receber equivale ao salário bruto mais a quantia a receber pelas horas extras.

22. Faça um programa que receba o número de lados de um polígono convexo, calcule e mostre o número de diagonais desse polígono. Sabe-se que ND = N * (N − 3)/2, em que N é o número de lados do polígono.

23. Faça um programa que receba a medida de dois ângulos de um triângulo, calcule e mostre a medida do terceiro ângulo. Sabe-se que a soma dos ângulos de um triângulo é 180 graus.

24. Faça um programa que receba a quantidade de dinheiro em reais que uma pessoa que vai viajar possui. Ela vai passar por vários países e precisa converter seu dinheiro em dólares, marco alemão e libra esterlina. Sabe-se que a cotação do dólar é de R$ 1,80; do marco alemão, de R$ 2,00; e da libra esterlina, de R$ 3,57. O programa deve fazer as conversões e mostrá-las.

25. Faça um programa que receba uma hora (uma variável para hora e outra para minutos), calcule e mostre:
a) a hora digitada convertida em minutos;
b) o total dos minutos, ou seja, os minutos digitados mais a conversão anterior;
c) o total dos minutos convertidos em segundos.