

## CAPÍTULO 2

### Comandos de Atribuição (=), Entrada (*cin*) e Saída (*cout*)

|  |    |
|--|----|
| COMANDO DE ATRIBUIÇÃO.....                                     | 22 |
| <i>SINTAXE DO COMANDO DE ATRIBUIÇÃO EM PROGRAMAS</i> .....     | 22 |
| COMANDOS DE ENTRADA E SAÍDA.....                               | 24 |
| <i>COMANDO DE ENTRADA</i> .....                                | 24 |
| <i>SINTAXE DO COMANDO DE ENTRADA EM ALGORITMOS</i> .....       | 24 |
| <i>SINTAXE DO COMANDO DE ENTRADA EM PROGRAMAS EM C++</i> ..... | 24 |
| <i>COMANDO DE SAÍDA</i> .....                                  | 25 |
| <i>SINTAXE DO COMANDO DE SAÍDA EM ALGORITMOS</i> .....         | 26 |
| <i>SINTAXE DO COMANDO DE SAÍDA EM PROGRAMAS EM C++</i> .....   | 26 |
| MÉTODO PARA CONSTRUÇÃO DE PROGRAMAS.....                       | 28 |
| INDENTAÇÃO .....   | 30 |
| PRIMEIRO PROGRAMA EM C++ .....                                 | 30 |
| EXERCÍCIOS PROPOSTOS DO CAPÍTULO 2: .....                      | 32 |

## CAPÍTULO 2

### COMANDO DE ATRIBUIÇÃO

O comando de atribuição permite que se forneça um valor a uma certa variável, onde a natureza deste valor tem de ser compatível com o tipo da variável na qual o valor está sendo armazenado. É a maneira de se armazenar um dado/uma informação numa variável. Fazendo uma analogia: é guardar um objeto em uma certa gaveta própria para aquele objeto.

O comando de atribuição é representado nos algoritmos pelo símbolo  $\leftarrow$ , e, em geral, nos programas pelo símbolo  $=$ .

Em todo programa cada comando é finalizado com o sinal de ponto-e-vírgula, que objetiva separar uma ação da outra e auxiliar a organização sequencial das ações, pois após encontrar um (;) deve-se executar o próximo comando da sequência.

### SINTAXE DO COMANDO DE ATRIBUIÇÃO EM PROGRAMAS

```
nome_da_variável = expressão;
```

O resultado da expressão, que pode ser aritmético, lógico ou literal, é atribuído à variável. A natureza do valor atribuído tem que ser do mesmo tipo da variável declarada.

```
nome_da_variável1 = nome_da_variável2;
```

O conteúdo da variável2 é atribuído à variável1. Elas têm que ser de tipos de dado compatíveis.

```
nome_da_variável = constante;
```

O valor da constante é atribuído à variável. O valor da constante e a variável têm que ser de tipos de dado compatíveis.

#### Exemplos:

Atribui-se às variáveis, os valores fornecidos e/ou obtidos à direita do símbolo de atribuição (=).

```
k = 1;
cor = "verde";
cod = (sqrt(k) + 1 >= 5);
teste = false;
media = soma/n;
sim = (x == 0 && y != 2);
a = b;
```

**Exercício 2.1)** Encontre os erros dos seguintes comandos de atribuição (=):

```
{
    bool a;
    int b;
    float c, d;
    char e;

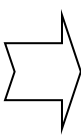
    d = b;
    c+1 = b+c;
    a = b == c;
    c&&b = 3.5;
    e = (c >= d);
}
```

A operação de atribuição é uma operação destrutiva, pois qualquer valor que a variável possuía, antes do processamento da operação de atribuição, se perde, sendo substituído pelo novo valor. Então, se a sequência de operações:

```
a = 16;
a = -27;
a = 1;
```

fosse executada, o valor da variável a após as três operações seria 1. Os valores 16 e -27 foram destruídos. Note que, quando tem-se uma sequência de atribuições, as operações individuais são sempre executadas na ordem em que aparecem.

**Exemplo)** Escreva um programa que calcule o preço total da compra do produto, fornecido o seu preço unitário e a quantidade comprada.

|  |   |
|--|---|
| <pre>int main ()    // programa exemplo1 {     float precounit, precotot; // preço de cada produto e preço total     float quant;              // quantidade do produto adquirido      precounit = 5.5;     quant = 10;     precotot = precounit * quant;      return 0; }</pre> |  <p><b>necessidade de entrar com mais valores (???)</b></p> <p><b>necessidade de mostrar o resultado (???)</b></p> |
|--|---|

Recomenda-se que a declaração de variáveis e os comandos contenham comentários que facilitem o entendimento de seus significados.

O programa acima resolveu o problema, mas não mostrou o resultado. Um programa que não informa ao usuário o resultado de seu processamento, não tem utilidade. É necessário disponibilizar meios de mostrar em alguma unidade de saída (tela do computador, impressora, etc..) os resultados dos processamentos realizados pelo programa. Para isto foram criados os comandos de saída de dados.

O programa acima não está bom, pois toda vez que é executado, ele calcula o mesmo preço total, porque os valores das variáveis precounit e quant não são alterados. O correto seria que estes valores pudessem ser fornecidos pelo usuário toda vez que o programa fosse executado, aumentando a sua flexibilidade. Os comandos de entrada de dados foram criados para suprir esta necessidade.

## COMANDOS DE ENTRADA E SAÍDA

Sabe-se que as unidades de entrada e saída são dispositivos que possibilitam a comunicação entre o usuário e o computador. O programador é quem determina o momento da entrada dos dados para o programa e a saída dos resultados obtidos para o usuário.

### COMANDO DE ENTRADA

Através de uma unidade de entrada (teclado, mouse, etc..) o usuário dá entrada ao programa e aos dados a serem manipulados pelo programa na memória do computador.

O comando de entrada é responsável em receber dados digitados pelo usuário. Os dados recebidos são armazenados nas posições de memória das variáveis cujos nomes aparecem na frente do comando de entrada. O comando de entrada é representado nos algoritmos pela palavra reservada **leia** e nos programas em C++, pela palavra **cin**.

### SINTAXE DO COMANDO DE ENTRADA EM ALGORITMOS

**leia** lista de identificadores ;

onde:

**leia** é uma palavra reservada

**lista de identificadores** são nomes de variáveis onde serão armazenados os valores provenientes do meio de entrada. Usa-se a vírgula (,) para separar as variáveis.

; para encerrar o comando de entrada de dados

### SINTAXE DO COMANDO DE ENTRADA EM PROGRAMAS EM C++

**cin >>** lista de identificadores ;

onde:

**cin** é uma palavra reservada

**>>** é o separador do cin

**lista de identificadores** são nomes de variáveis onde serão armazenados os valores provenientes do meio de entrada. Usa-se >> para separar as variáveis.

; para encerrar o comando de entrada de dados

A operação de entrada é destrutiva do mesmo modo que a operação de atribuição. Qualquer valor que a variável possuía, anteriormente, é destruído.

Exemplo: Tem-se um número de matrícula e uma nota a ser digitada na unidade de entrada. Quais deveriam ser os comandos para receberem estas entradas?

|               |
|---------------|
| 09795<br>88.2 |
|---------------|

```
{
    float nota;    // armazena nota
    int mat;       // armazena o número de matrícula do aluno

    cin >> mat;
    cin >> nota;
}
```

mat

nota

|       |
|-------|
| 09795 |
|-------|

|      |
|------|
| 88.2 |
|------|

**Exercício 2.2)** Foram digitados 6 dados de nomes de alunos e suas notas:

|   |
|---|
| PAULO<br>100.0<br>MARIA<br>75<br>JOSÉ<br>60.5 |
|---|

Escreva os comandos de entrada que leiam estas linhas e armazenem os valores na memória (não se esqueça de declarar as variáveis que irão armazenar estes valores). Dica: Iremos necessitar de 6 variáveis.

## COMANDO DE SAÍDA

O computador pode emitir resultados e mensagens para o usuário através de uma unidade de saída (impressora, vídeo ou outro) e o comando de saída é o meio pelo qual as informações contidas na memória dos computadores são colocadas nestes dispositivos de saída. Esse comando é representado nos algoritmos pela palavra reservada **escreva** e nos programas em C++, pela palavra **cout**. E os dados/informações a serem mostrados podem ser conteúdos de variáveis e/ou mensagens.

## SINTAXE DO COMANDO DE SAÍDA EM ALGORITMOS

**escreva** lista de identificadores;

ou

**escreva** expressões;

ou

**escreva** "literal";

onde:

**escreva** é uma palavra reservada

**lista de identificadores** é um conjunto de nomes de variáveis, separadas por vírgulas, cujos conteúdos serão mostrados através de um meio de saída

**"literal"** é um dado do tipo literal (*string*), limitado por apóstrofes (" ") que é escrito diretamente no dispositivo de saída

**expressões** são os resultados de expressões, que podem ser aritméticos, lógicos ou literais e que são escritos no dispositivo de saída

No algoritmo a cada execução de um comando de saída (um **escreva**), uma nova linha é colocada à disposição para escrita.

Exemplos:

**escreva** a, "x", "35";

**escreva** "Nome ", nom, ", você pesa ", x\*2, " quilos.";

## SINTAXE DO COMANDO DE SAÍDA EM PROGRAMAS EM C++

**cout** << lista de identificadores;

ou

**cout** << expressões;

ou

**cout** << "literal";

onde:

**cout** é uma palavra reservada

<< é o separador do cout

Há possibilidade de se misturar nomes de variáveis, *strings* e expressões no mesmo comando de saída. O comando é executado da esquerda para a direita; se um nome de

variável for encontrado, a informação da mesma é pega na memória e colocada no dispositivo de saída (como se fosse uma fotocópia); no caso do *string*, o mesmo é escrito diretamente no dispositivo de saída e no caso de expressão, seu resultado é também escrito na unidade de saída.

No programa, a cada execução de um comando de saída (um **cout**), para que uma nova linha seja colocada à disposição para escrita, deve-se utilizar o **endl** ou “\n”.

Exemplos:

```
cout << a << " " << x << "35" << endl;
```

```
cout << "Nome " << nom;           // não pula linha aqui
```

```
cout << ", você pesa " << x*2 << " quilos." << "\n";
```

**Exercício 2.3)** Utilizando as variáveis do exercício anterior (2.2), escreva os comandos de saída que imprimam os conteúdos das posições de memória conforme lay-out abaixo:

|                            |
|----------------------------|
| JOSÉ 60 PAULO 100 MARIA 75 |
|----------------------------|

|                                  |
|----------------------------------|
| JOSÉ 60<br>MARIA 75<br>PAULO 100 |
|----------------------------------|

|   |
|---|
| MARIA<br>75<br>JOSÉ<br>60<br>PAULO<br>100 |
|---|

Uma preocupação constante de um bom programador deve ser a de fazer programas “amigáveis”, que tenham uma boa interface com o usuário (meio pelo qual usuário e programa “conversam”). Regras básicas a serem aplicadas:

- 1) toda vez que um programa estiver esperando que o usuário forneça a ele um dado, o programa deve enviar uma mensagem ao usuário, através do comando **cout <<** dizendo qual a informação que está sendo requisitada;
- 2) o programa deve também escrever uma mensagem antes de enviar qualquer resultado ao usuário.

Verificando agora o programa **exemplo1**, pode-se fazer as seguintes alterações, após ter aprendido os comandos de entrada e saída de dados.

```
int main ()    // programa exemplo1
{
    float precounit, precotot; // preço de cada produto e preço total
    float quant;              // quantidade do produto adquirido

    // entrada de dados
    cout << "Entre com o preço unitário do produto e com a quantidade comprada ";
    cin >> precounit >> quant;

    // processamento
    precotot = precounit * quant;

    // saída de dados
    cout << "O preço total calculado é " << precotot << endl;

    return 0;
}
```

## MÉTODO PARA CONSTRUÇÃO DE PROGRAMAS

- 1o. passo) Ler atentamente o enunciado do problema
- 2o. passo) Retirar do enunciado a relação das entradas de dados
- 3o. passo) Retirar do enunciado a relação das saídas de dados
- 4o. passo) Determinar o que deve ser feito para transformar as entradas determinadas nas saídas especificadas
- 5o. passo) Construir o programa.
- 6o. passo) Executar o programa. Verificar se os resultados obtidos correspondem ao esperado.

**Exemplo)** Utilizando o método para construção de programas, escreva um programa que calcule a média entre dois números quaisquer que serão fornecidos pelo usuário.

### MÉTODO:

#### 1o. passo) Ler o enunciado com atenção

O enunciado pede para fazer um programa que calcule a média aritmética entre dois números quaisquer.

#### 2o. passo) Relação das entradas de dados

Dois números quaisquer: num1 e num2. Devem ser do tipo numérico (float, de preferência).

#### 3o. passo) Relação das saídas de dados

A média aritmética que deverá ser também do tipo numérico/float.



**4o. passo) O que deve ser feito para transformar a entrada em saída**

Somar os dois números e dividir por 2

**5o. passo) Construir o programa:**

```
int main ()    // programa que calcula média
{
    float num1, num2, media;           // número 1, número 2 e média

    // entrada de dados
    cout << "Entre com valores para os 2 números ";
    cin >> num1 >> num2;

    // processamento
    media = (num1 + num2) / 2;         // cálculo da média aritmética

    // saída de dados
    cout << "A média calculada é " << media << endl;

    return 0;
}
```

**6o. passo) Executar o programa e verificar se ele está fazendo o esperado.****NUM1****NUM2****MEDIA**

**Exercício 2.4)** Utilizando o método para construção de programas, escreva um programa que calcule o valor de y como função de x, segundo a função  $y = 3x + 2$ .

## INDENTAÇÃO

Algumas linhas dos programas começam mais à direita, e outras, mais à esquerda. É que existe um certo padrão de alinhamento entre elas. Isto é feito para melhorar a legibilidade dos programas. Este procedimento recebe o nome de **indentação**. Fica visualmente fácil de ver, apenas observando o alinhamento das linhas, onde começa e termina a declaração de variáveis, e o início e final do programa, o início e o final dos comandos.

Como os comentários, as regras de indentação não são únicas, cada autor apresenta suas próprias regras. O padrão adotado nesta apostila é fácil de aprender. Acompanhe os exemplos e exercícios e faça da mesma forma.

## PRIMEIRO PROGRAMA EM C++

Um programa em C++ consiste em uma ou várias funções. A forma geral de uma função em C++ é a seguinte:

```

tipo nomeFunc (declaração dos parâmetros)
{
    declaração1;
    .....
    declaraçãom;

    instrução1;
    .....
    instruçãon;

    return var_tipo;
}

```

Iremos começar um programa simples. Para isso iniciaremos o Falcon C++ e abriremos um novo Arquivo C++ .

Ou você pode abrir o esqueleto de programa (baixe o **esqueleto\_programa.cpp**):

```

#include <iostream>           // padrão de entrada e saída no C++
#include <cmath>              // inclui funções matemáticas
#include <cstring>             // inclui funções de manipulação de string
#include <locale>              // inclui função para se usar palavras em português
#include <iomanip>             // inclui funções de configuração de saída

using namespace std;         // para usar bibliotecas

int main()
{
    setlocale(LC_ALL,"Portuguese"); // habilita a estrutura ortográfica em português

    // declarações
    // entrada de dados
    // processamento
    // saída de dados

    return 0;
}

```

A função **main** é a função que inicia a execução do programa. Em todo programa C++ deve existir uma única função chamada de *main*. Ela marca o ponto de partida do programa. E este é encerrado quando for encerrada a execução da função *main*.

A diretiva **#include** provoca a inclusão de outro arquivo no programa-fonte. Na verdade, o compilador substitui a linha contendo essa diretiva pelo conteúdo do arquivo indicado. Esta diretiva insere as bibliotecas (com funções) que serão utilizadas no programa.

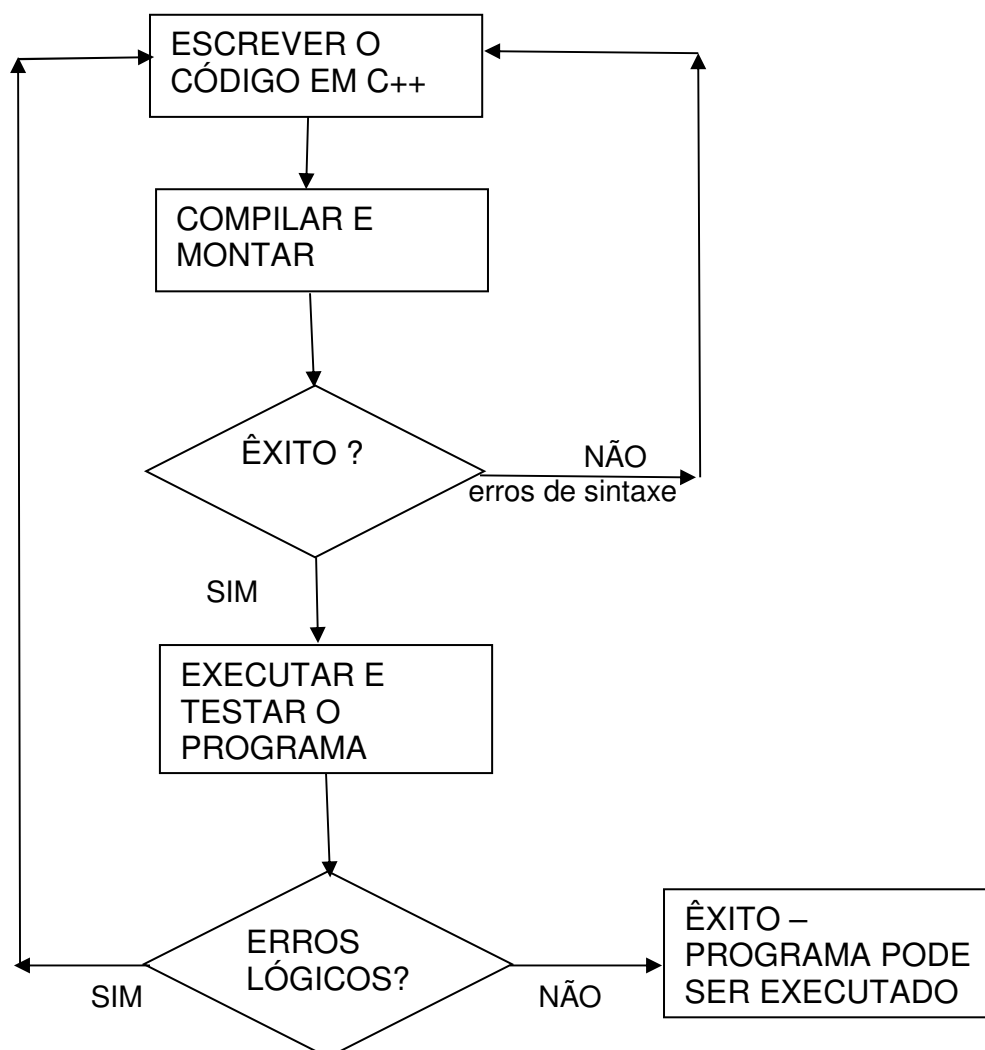
A diretiva **using namespace std** serve para se utilizar alguns comandos (objetos) no programa sem ter que precedê-los com (*std::*). Exemplo: *std::cout << "Primeiro programa";*

Para limitar/mostrar a quantidade de casas decimais de uma variável do tipo **double** ou **float**, usamos o comando **setprecision**. É preciso inserir a biblioteca **#include <iomanip>**

```
float var = 4.55623;
cout << fixed << setprecision(3);
cout << var << endl;
```

Nesse caso a saída do valor terá 3 casas decimais.

A figura abaixo mostra o **Processo completo de conclusão de um programa em C++**. Os **erros de sintaxe são detectados na Compilação**, enquanto **os erros lógicos são detectados quando o programa é executado/testado** e percebe-se que o mesmo não está fazendo o desejado.



### Erros mais comuns que ocorrem na programação:

- Na hora do **cout**, colocar o nome da variável entre aspas
- Esquecer de colocar o ponto e vírgula no final do comando
- Esquecer de incluir as bibliotecas
- Confundir **==** (igual) com **=** (atribuição)
- Esquecer dos caracteres chave **{ }** de abertura e fechamento de bloco
- Esquecer das aspas ao fechar uma cadeia de caracteres
- Para colocar uma condição depois de **else**, precisa do **if**
- Não colocar **endl** quando se quer mudar de linha
- Usar **end** ao invés de usar o **endl**
- Não comentar o código
- Não indentar o código

### Exercícios propostos do Capítulo 2:

P2.1) Elabore um programa que leia um nome de pessoa na forma **nome de batismo** seguido por **sobrenome** e imprima o nome na forma **sobrenome** seguido por **,nome de batismo**. Exemplo:

Na entrada de dados: **Alberto Capistrano**

Na saída: **Capistrano, Alberto**

P2.2) Faça um programa que receba um número, calcule e mostre:

- a) o número digitado ao quadrado;
- b) o número digitado ao cubo;

P2.3) Faça um programa que receba um número, calcule e mostre (apenas em um programa):

- a) sua parte inteira;
- b) sua parte fracionária;
- c) o arredondamento desse número para inteiro maior.

P2.4) O programa abaixo determina o valor da função  $f(x,y) = 2x + 3y^2$ . Verifique o seu desenvolvimento e aponte 3 erros cometidos na construção do mesmo.

```
int main ()    // programa que calcula função de 2 variáveis
{
    float x, f(xy), y;    // valor de x, resultado da função e valor de y

    // entrada de dados
    cout << "Entre com os valores de X e Y ";
    cin >> x >> y;

    // processamento
    f(xy) = 2x + 3y2;    // cálculo da função

    return 0;
}
```

P2.5) Desenvolva um programa que efetue o cálculo da área (a) de uma circunferência apresentando a medida da área calculada através de um dado raio (r). ( $a = \pi \cdot r^2$  e considerar  $\pi = 3.14159$ )

P2.6) Faça um programa para ler uma temperatura em graus Centígrados e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é  $F = (9C + 160)/5$  onde F é a temperatura em Fahrenheit e C é a temperatura em Centígrados.

P2.7) Faça um programa que receba o raio, calcule e mostre (em apenas 1 programa):

- 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  $V = 4/3\pi R^3$ .

P2.8) Prepare um programa para ler os comprimentos dos três lados de um triângulo (L1, L2 e L3), calcular e mostrar a área do triângulo de acordo com a fórmula:

$$\text{área} = \sqrt{T(T - L1)(T - L2)(T - L3)} \text{ onde}$$

$$T = \frac{L1 + L2 + L3}{2}$$

P2.9) 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.

P2.10) Dadas as três notas em três aspectos do ano escolar: laboratório, exame intermediário e exame final, para um estudante e o seu nome, elabore um programa que calcule e escreva a média final com pesos de 20%, 30% e 50%, respectivamente. O relatório de saída tem o seguinte formato:

|                         |
|-------------------------|
| NOME: Rodolfo           |
| Nota de Laboratório: 72 |
| Exame Intermediário: 68 |
| Exame Final: 65         |
| Média Final: 67.3       |

P2.11) Sabe-se que para iluminar de maneira correta os cômodos de uma casa, para cada  $m^2$ , deve-se usar 18W 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 iluminação (potência total) que deverá ser utilizada.

P2.12) Construa um programa que leia uma quantidade inteira expressa em milímetros, calcule e imprima a quantidade correspondente expressa em metros, decímetros, centímetros e milímetros.

Exemplo: Na entrada: 82453  
Na saída: 82 metros, 4 decímetros, 5 centímetros e 3 milímetros.

P2.13) Dado um número inteiro de 3 algarismos, inverta a ordem de seus algarismos.

Exemplo: Na entrada: 891  
Na saída: 198

P2.14) Construa um programa que efetue o cálculo do salário líquido mensal de um professor. Dados de entrada: o nome do professor, o valor hora aula, o número de aulas dadas no mês e o percentual de desconto do INSS. Apresente o nome, o salário bruto e o salário líquido.

P2.15) Faça um programa que receba o salário-base de um funcionário, calcule e mostre o salário a receber, sabendo que esse funcionário tem gratificação de 5% sobre o salário base e paga imposto de 7% sobre o salário total (salário-base + gratificação).

P2.16) Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:

- a) a idade dessa pessoa atualmente
- b) quantos anos essa pessoa terá em 2030.

P2.17) Desenvolva um programa que efetue o cálculo da quantidade de litros de combustível gasta em uma viagem, utilizando-se um automóvel que faz 11 km por litro. Serão fornecidos: o tempo gasto na viagem (em horas) e a velocidade média (em km/hora) durante a mesma. O programa deverá apresentar os valores da velocidade média, do tempo gasto, da distância percorrida e da quantidade de litros de combustível utilizada.

P2.18) Faça um programa para ler dois valores para as variáveis A e B, efetuar a troca dos valores de forma que a variável A passe a possuir o valor da variável B e que a variável B passe a possuir o valor da variável A. Apresente os valores trocados.

P2.19) Sabe-se que: 1 pé = 12 polegadas; 1 jarda = 3 pés; 1 milha = 1760 jardas

Faça um programa que receba uma medida em pés, faça as devidas conversões e mostre os resultados em polegadas, jardas e em milhas.