2 A Programação Básica

2.1 A Entrada de Dados

Para um computador poder cumprir as ordens que um programador criou, ele precisa conhecer algumas informações iniciais. Assim como o ensinamento de uma criança nos princípios dos primeiros passos, onde um adulto precisa orientar e direcioná-la, um programa de computador precisa receber do usuário informações iniciais conhecidas como entrada de dados.

É fundamental que o programador tenha consciência que o usuário na maioria das vezes quer uma resposta rápida para o tipo de processo que está trabalhando. Sendo assim, um programa de computador que necessita de entrada de dados externos, não pode de maneira alguma pedir grandes quantidades de informações iniciais. O melhor programa é aquele que necessita de uma quantidade mínima de informações externas para seu funcionamento.

Um algoritmo é como uma receita de bolo, isso foi que me ensinaram no tempo de graduação. O nome da receita é o nome do programa. Os ingredientes são as definições dos tipos de variáveis que o programador vai usar. E o algoritmo em si, é a forma como o bolo será feito.

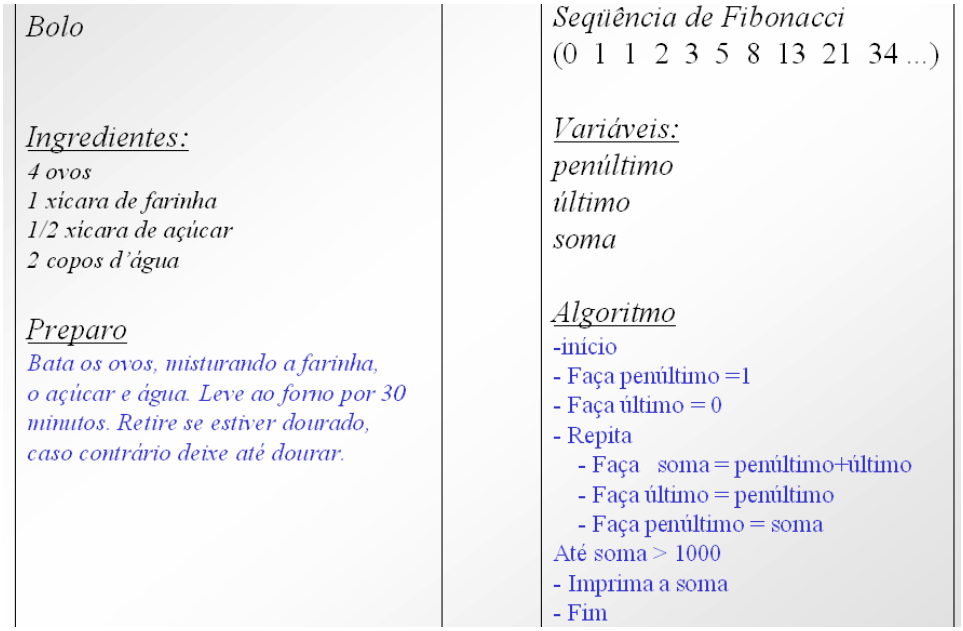


Figura 2.1 - Equivalência Receita de bolo e algoritmo

Desde os primórdios dos tempos de tecnologia de programação, um programa de computador precisa passar por algumas seqüências de avaliações. O *editor de texto* é onde o usuário escreve o que ele deseja que o computador realize. Depois de escrever o programa, para que o mesmo converse com o computador, não pode haver erros na colocação das regras dentro da linguagem escolhida. Para isso as averiguações de inconsistências entre o que o programador deseja e o que ele pode usar naquela linguagem escolhida é realizada pelo *compilador*. Caso haja alguma inconsistência entre o programa escrito e a notação dos símbolos e regras permitidas pela linguagem, uma mensagem de erro é enviada pelo compilador.

A última fase é o processo de *linkagem* realizada pelo *linkador* que vai atrelar ao programa todas as bibliotecas de funções que o programa poderá acessar. Exemplo: a impressão de um gráfico precisa de uma rotina da linguagem que permita a impressão. Se a linguagem não tem essa rotina ou função que permita o programa imprimir um gráfico ou imagem, o programa não poderá realizar a tarefa e uma mensagem de erro será emitida para o usuário.

Todos esses alertas podem deixar o leitor iniciante confortável, achando que seus erros serão alertados pelo computador. Não, infelizmente o pior dos erros nunca é alertado pelo computador. O erro lógico de pensamento do programador nunca á encontrado pelo computador. Exemplo: se o programador enviou uma ordem para que o computador fazer a operação 2 + 3 = 4, se a regra de soma estiver de acordo com as permissões da linguagem, o computador vai enviar ao usuário a mensagem que dois mais três é quatro e não cinco. O problema é que o computador recebeu uma ordem lógica errada, mas com comandos certos. Assim, caro leitor, lembre-se que seus erros de programação somente um ser humano poderá encontrar (no VBA-Excel pelo menos).

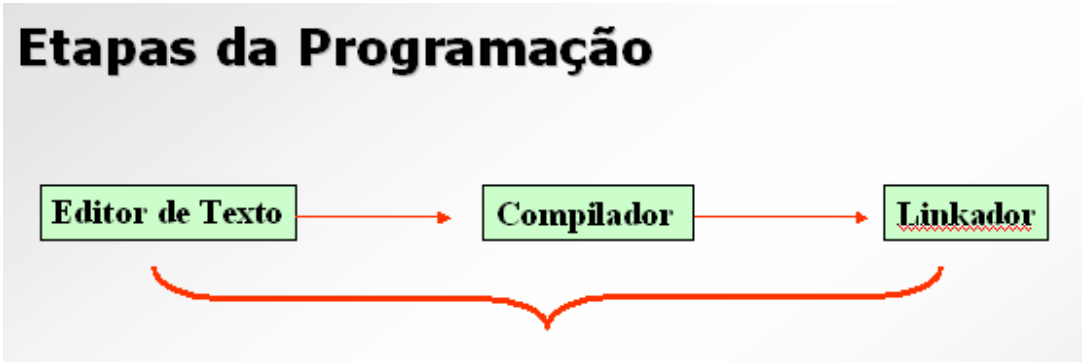


Figura 2.2- O processo de programação

No entanto, no VBA-Excel esse processo é muito rápido e o leitor nem perceberá qual dos processos está ocorrendo. Nos primórdios dos primeiros computadores linguagens as vezes demoravam minutos e até horas para compilar e mais ou menos o mesmo tempo para linkar. Exemplo disso é a linguagem cientifica que dominou o mundo acadêmico até a década de 1990, o FORTRAN, onde os compiladores eram programas externos a linguagem e os linkadores também. Para a execução de um programa, o programador tinha que esperar bastante tempo e torcer para não acusar erros em sua programação.

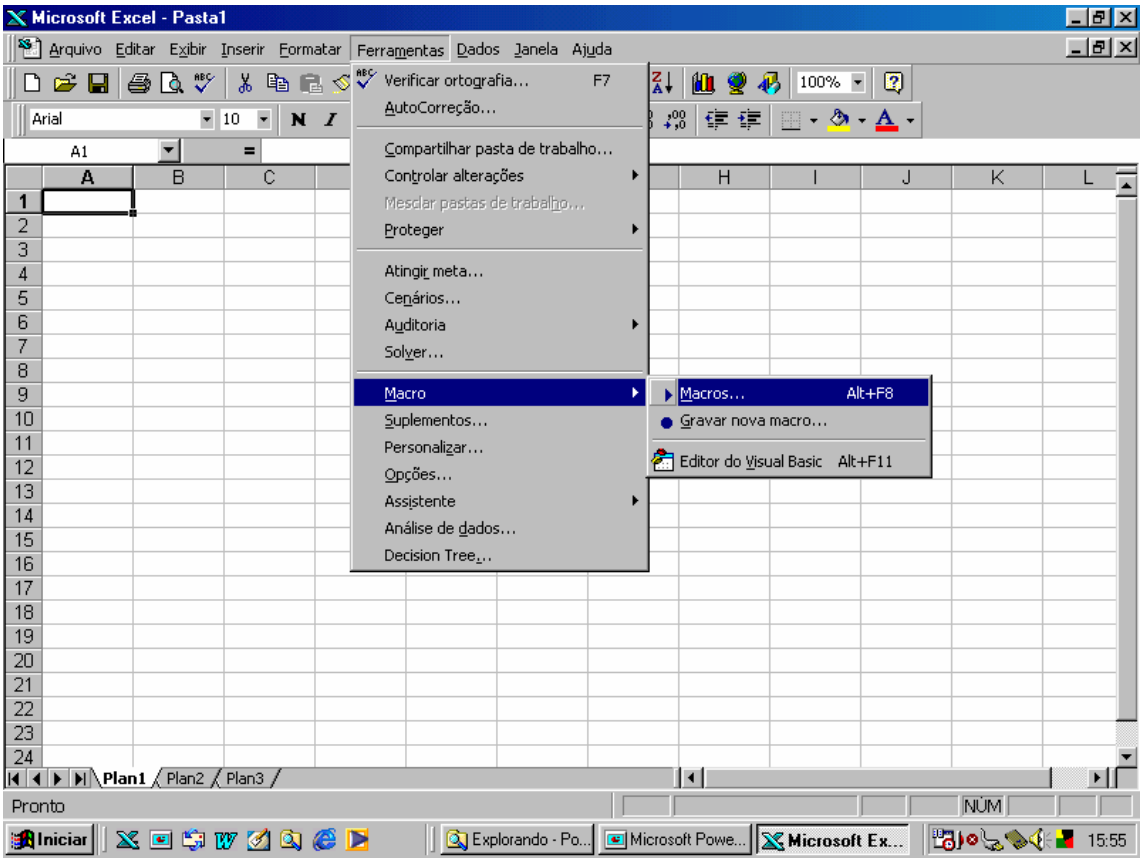


Figura 2.3- Acessando o Visual Basic da planilha do Excel

No caso do VBA-Excel, o ambiente de programação das *macros* se encontra no caminho *ferramentas* → *macros*→ *editor do visual basic.* Então aparecerá a tela de programação ainda vazia com tela a esquerda mostrando a raiz e as planilhas contidas no arquivo em Excel *(janela do project explorer)* e a *janela de propriedades* que serão exploradas nos capítulos posteriores.

Clica-se então no caminho *ferramentas* → *macros* e então ao abrir a nova tela de nomes, coloca-se um nome para o programa (ver Figura 2.5). Ao clicar no botão "criar" será aberta a tela onde o programa deverá ser escrito. É nessa tela que começa toda a programação dos algoritmos e será nessa tela que o computador interpretará os comandos do programador (Figura 2.6).

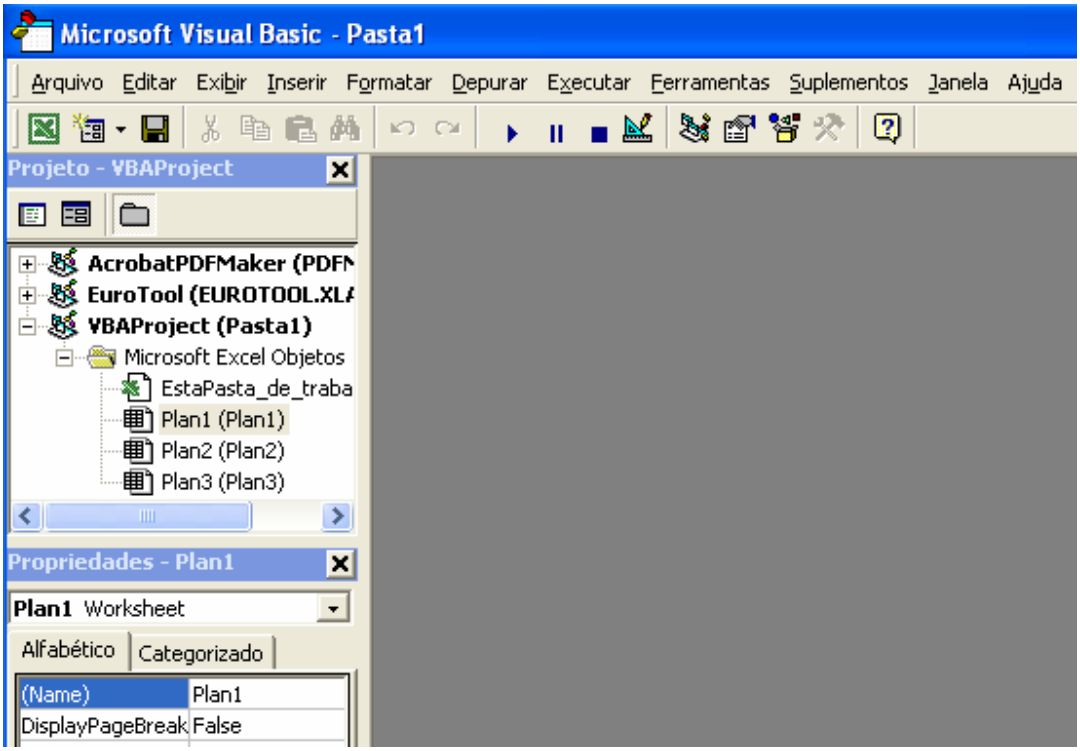


Figura 2.4- Acessando o ambiente de macros

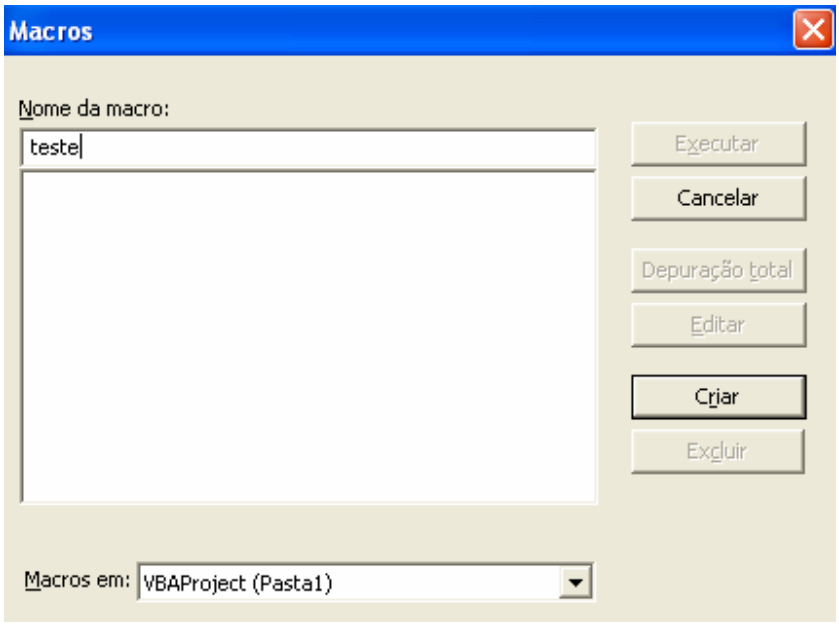


Figura 2.5- Escrevendo o nome do programa

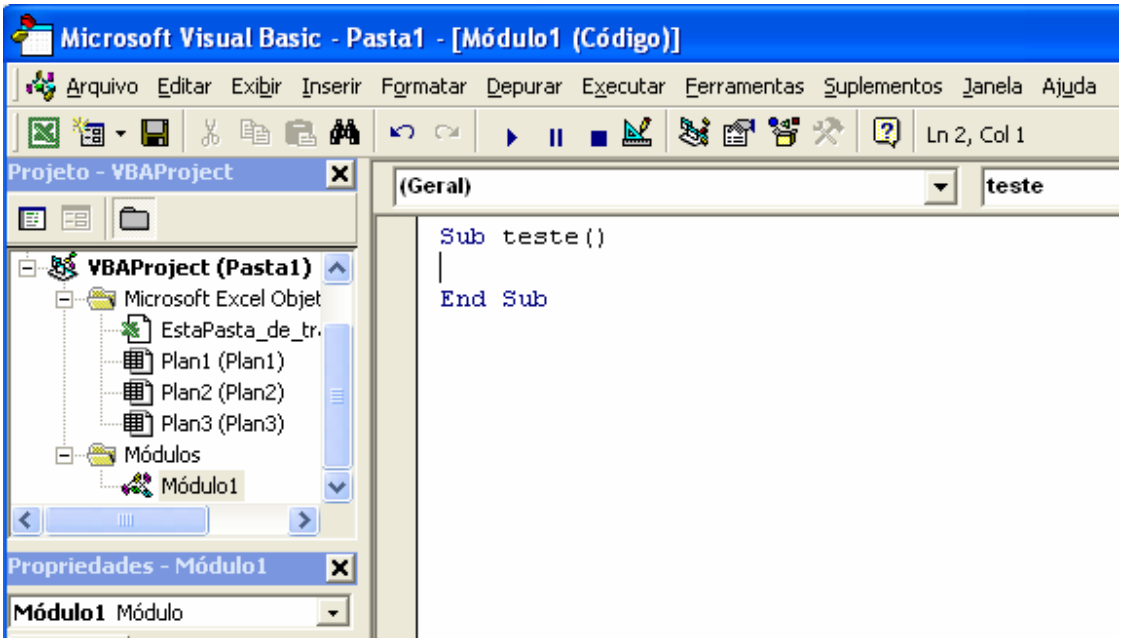


Figura 2.6 – Tela onde o algoritmo deverá ser escrito.

Algumas observações importantes devem ser feitas em relação ao nome do programa e nomes de variáveis. Tanto os nomes dos programas quanto de variáveis devem seguir a seguinte regra de nomeação:

•Não pode começar com NÚMERO

Exemplo: 1prog, 10xpe, 2123cal, 1aula,

•

•Não pode ter espaços em branco

Exemplo: aula 1
prog comp

•Não pode ter acentos, c cedilha, ou qualquer pontuação

Exemplo: aulão1
cçwqr

•Não pode passar de 8 caracteres

Exemplo: programadecomputação1
Auladecomp

•PODE misturar letras e números

Exemplo: prog1
pr23rtu

Uma vez no módulo de programação o usuário está pronto para colocar os algoritmos em funcionamento, utilizando-se para isso da linguagem de programação **V***isual* **B***asic* for **A***plications* (VBA). O primeiro ponto de contato do leitor com a programação é que será necessário informar o computador sobre dados de entradas das variáveis de problemas. O computador precisará sempre de uma informação inicial para poder processar os algoritmos. Essa informação inicial que se chama entrada de um programa de computador. Antes será necessário informar ao computador, quais são as variáveis de que ele irá dispor no programa. A declaração das variáveis é feita através da nomenclatura ***"dim"*** que vem da palavra ***dimension***, a qual dimensiona o tipo de variável a ser usada pelo computador. Observe o trecho do programa teste a seguir:

```
Sub teste()
Dim x As Integer

End Sub
```

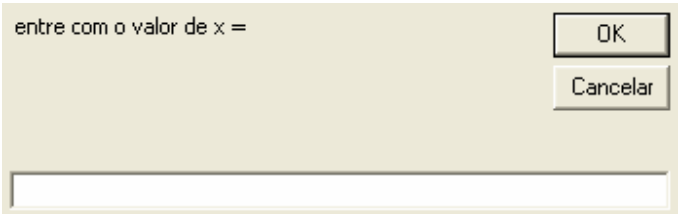
O programa está informando ao computador que ele terá que usar uma variável chamada "x" e ainda que essa variável é do tipo *Integer*, ou seja, uma variável do tipo inteira, onde somente receberá e usará valores inteiros (...,-2,-1,0,1,2,...). A partir de agora o computador precisará ter a informação de entrada sobre qual é esse valor de x. Essa informação será passada pelo usuário através de uma caixa de texto, com nome *InputBox*. Essa caixa ainda poderá definir ao usuário qual o tipo de dado está entrando no programa. Por exemplo, se o dado for valor inteiro (*Integer*) então a caixa de entrada receberá uma nomeação *Cint* que significa (Change Integer). Outros tipos de caixa poderão ser inseridos com tipos de variáveis diferentes:

- **CInt() –** dados inteiros.
- **CSng()** – dados reais.
- **CDbl()** – dados com dupla precisão.
- **CStr()** – dados texto (*String*).

Então o programa teste para a entrada do valor de x será:

```
Sub teste()
Dim x As Integer

x = CInt(InputBox(" entre com o valor de x = "))

End Sub
```

O texto entre aspas é o que o usuário irá receber na caixa de texto. Dentro das aspas o programador poderá programar e escrever qualquer tipo de texto, pois a caixa reproduzirá toda informação entre aspas. O resultado para o usuário será:



Ao entrar com os dados o usuário informa ao computador o valor da variável x e então o computador poderá fazer os cálculos necessários pois conhecerá quem é essa variável. Mas, como o usuário visualizará o resultado das operações programadas. Isso é o que se chama saída de dados.

2.2 A saída de Dados

A entrada de dados fornece ao computador as condições para o programa funcionar e fazer cálculos. Mas a visualização dos resultados é permitirá uma "conversa" entre o usuário e o computador. A saída de dados é muito fácil no caso do VBA-Excel usando para isso uma caixa de mensagens, chamada de MsgBox( ). Novamente, o texto entre aspas é o que o usuário visualizará do resultado. Mas além do texto, o valor da variável deverá aparecer ao usuário. E como o próprio nome diz, variável é a variante do programa e sempre está relacionada com algum cálculo. Então, como o computador exportará para o usuário o resultado de uma variável? Utiliza-se sempre para variáveis na programação VBA o símbolo & para indicar que depois dele o valor depende de cálculos anteriores. Uma vez o programa estando pronto ele deve ser executado para que os passos de *compilação* → *execução* sejam realizados. Para se executar o programa e ver o resultado da programação o usuário deve ir até a caixa de ferramentas superior dentro do módulo onde o programa está sendo programado e clicar em Executar através do botão  ou na palavra executar contida na barra padrão conforme Figura 2.6.

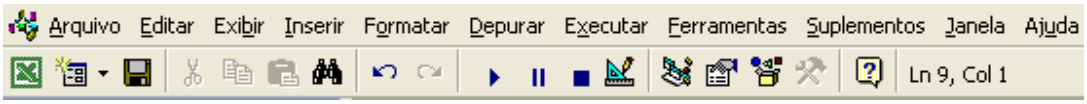


Figura 2.6 Barra de ferramentas padrão do módulo de programação VBA

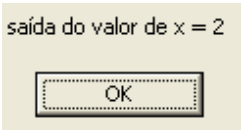
O exemplo a seguir mostra como a entrada de x é apresentada na saída ao usuário através dos dois comandos *InputBox* e *MsgBox*.

```
Sub teste()
Dim x As Integer

x = CInt(InputBox(" entre com o valor de x = "))

MsgBox ("saída do valor de x = " & x)

End Sub
```

O resultado será a caixa de mensagem a seguir informando ao usuário o valor 2 que o usuário entrou na caixa InputBox.

A entrada de dados para o valor x = 2 (*via InputBox*)

A saída de dados para o valor x = 2 (*via MsgBox*)



Exemplo 2.1

Fazer um programa para calcular e mostrar ao usuário o valor de y dado pela equação $y = 2x + 4$.

Quem é a entrada desse programa? Claro que será a variável x enquanto a variável y será a saída que deverá ser visualizada pelo usuário. E quais os tipos de resultados? Para qualquer valor de x, teremos valores de y desde que eles respeitem a regra da equação proposta. Então x poderá ser inteiro, racional ou real. Logo tanto a variável de entrada x quanto a variável de saída y serão números reais. A macro programada nesse caso será a apresentada a seguir.

```
Sub teste()
Dim x As Single
Dim y As Single

x = CInt(InputBox(" entre com o valor de x = "))

y = 2 * x + 4


MsgBox ("O valor de y é " & y)

End Sub
```

O leitor iniciante perceberá que o produto deve ser indicado pelo símbolo “ * ” . Sem esse símbolo o computador não saberá que a constante x deve ser multiplicada por dois. Aqui cabe um parêntesis. Na verdade y não é igual a 2*x+4, mas sim ela está recebendo dessa operação um certo valor que deverá ser armazenada numa certa posição de memória. O mais correto e aplicado durante muito tempo em cursos de computação tradicionais era que

$$y \leftarrow 2 * x + 4$$

Essa sim é a representação mais correta para não confundir principiantes de que se y=x então x=y em qualquer parte do programa. Se na linha de baixo do programa, aparecer uma outra operação envolvendo y, o valor de x não será alterado. Observe o exemplo a seguir 2.2.

A pergunta nesse algoritmo será: Qual o valor final de x quando o usuário entra com o valor x = 5? O valor de x será 1 ou 5? Observe que quem recebe o valor de x é y e não x recebe o valor de y. A notação mais correta é

$$y \leftarrow x$$

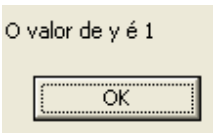
que indica que y está recebe o valor de y, mas nada é informado ao computador sobre o valor de x. Logo essa variável x terá na memória o mesmo valor que o usuário entrou. Ou seja, no exemplo 2.2 se o usuário entrar com x = 5 a saída será y = 1, pois y está recendo antes da apresentação na caixa de mensagens o valor 1 ou

$$y \leftarrow 1$$

Exemplo 2.2

```
Sub teste()
Dim x As Single
Dim y As Single

x = CInt(InputBox(" entre com o valor de x = "))

y = x

y = 1

MsgBox ("O valor de y é " & y)

End Sub
```

A execução desse programa usando x = 5 tem como resultado y = 1 conforme mostrado nas duas telas de execução:



Exemplo 2.3

Elaborar um programa que onde o usuário entre com o valor de x e o programa mostre o valor de y respeitando a equação $y = x^2 + 2x + 5$.

Aqui a novidade é apenas como elevar a variável ao quadrado, onde para isso o programador deverá usar o símbolo "^" entre o x e a potência. A solução será:

```
Sub teste()
Dim x As Single
Dim y As Single

x = CInt(InputBox(" entre com o valor de x = "))

y = x ^ 2 + 2 * x + 5

MsgBox ("O valor de y é " & y)

End Sub
```

Exemplo 2.4

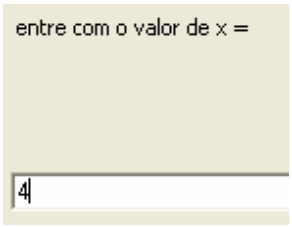
O usuário deverá entrar com x e o programa imprimir o resultado da equação

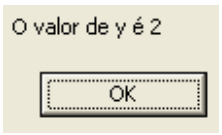
$$y = \sqrt{x}$$

A novidade nesse programa é que a linguagem VBA assim como outras linguagens uma função pré-definida para extrair a raiz quadrada de um número. No caso do VBA a função é ***Sqr( )***, que vem da palavra em inglês *Square Root* (raiz quadrada).

```
Sub teste()
Dim x As Single
Dim y As Single

x = CInt(InputBox(" entre com o valor de x = "))

y = Sqr(x)

MsgBox ("O valor de y é " & y)

End Sub
```

O resultado para x = 4 serão as telas a seguir:





Exemplo 2.5

{Descobrindo o ano de nascimento}

Fazer um algoritmo onde o computador deverá descobrir o ano de nascimento do usuário.

Esse algoritmo causa a sensação no usuário, de que o computador "pensa" ao responder. Logicamente é só uma sensação pois o computador deverá seguir lógica de linhas de pensamento humanas para poder dar a resposta. Assim, quando desejamos saber em que ano uma pessoa nasceu, qual a primeira pergunta que surge? É óbvio que será: qual sua idade. E depois o que se faz é subtrair o ano em que se está da idade da pessoa. Então a linha de passos que o computador deverá seguir será:

1. Usuário deverá entrar com a idade.
2. Ano atual subtraído da idade.
3. Ano de nascimento recebe valor da operação ano atual subtraído da idade.
4. Mostre o valor de ano de nascimento na caixa de mensagens.

Em termos de algoritmo a solução será:

1. Leia idade.
2. anonasc = 2007 – idade
3. imprima anonasc

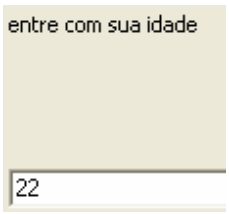
Deve ser observado que nunca se pode usar uma variável com nomes muito grande ou separada por espaços em branco. Para o programador é sempre interessante usar nomes sugestivos da variável ou se for o caso, usar o sinal de subscrição "_" e nunca o hífen "-". A variável ano de nascimento poderia ser: anonasc, anonac, ano_nas, ano_de_nascimento, etc. Mas nunca "ano de nascimento" pois isso ocasionaria o chamado **erro de compilação** como pode ser observado a seguir.

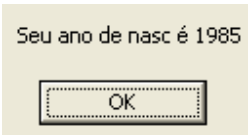
```
Sub teste()
Dim ano de nascimento As Single


End Sub
```

Então a solução para esse algoritmo é muito simples e a menos do cuidado dos nomes das variáveis, acaba sendo mais simples do que os demais, mas causa a sensação de que o computador está realmente "pensando" sobre seu ano de nascimento.

```
Sub teste()
Dim anonasc As Single
Dim idade As Single

idade = CSng(InputBox("entre com sua idade"))

anonasc = 2007 - idade

MsgBox (" Seu ano de nasc é " & anonasc)


End Sub
```

O resultado da execução será:





Exemplo 2.6
{Descobrindo a idade}

Fazer um algoritmo para descobrir a idade do usuário.

Esse algoritmo é o problema inverso do exemplo anterior, onde agora o computador deve descobrir a idade do usuário. A pergunta que o algoritmo deve fazer é sobre o ano de nascimento do usuário. O algoritmo fica:

```
Sub teste()
Dim anonasc As Single
Dim idade As Single

anonasc = CSng(InputBox("entre com o ano de nascimento"))

idade = 2007 - anonasc

MsgBox (" Você tem " & idade & " anos de idade")

End Sub
```

O leitor pode observar que a novidade nesse algoritmo é a impressão na caixa de mensagem de texto após a impressão da variável. Sempre que se desejar continuar uma impressão de texto após o aparecimento do valor das variáveis, deve-se usar o símbolo & antes da variável e atrás da variável, para indicar que a caixa conterá um texto logo a seguir.

Exemplo 2.7
{Raio de um círculo}

Fazer um algoritmo onde o usuário entra com o raio de círculo e o programa mostra ao usuário a área do círculo.

A área de um círculo é calculada com base no raio, ou seja,

$$Area = \pi r^2$$

Então, a pergunta básica que o computador deve fazer ao usuário é qual o valor do raio do círculo. E o valor do π? O programador pode usar uma aproximação para a variável pi adotando por exemplo o valor 3,14. Mas o cuidado que deve ter é que em programação

não existe vírgula separando as casas decimais como numa planilha comum do Excel com ortografia em Português. A separação deve ser feita com ponto decimal (.). Assim a constante pi deve ser 3.14 e não 3,14 na programação. A solução é apresentada na macro a seguir.

```
Sub area_circulo()
Dim raio As Single
Dim pi As Single

pi = 3.14

raio = CSng(InputBox(" Raio = "))

area = pi * raio ^ 2

MsgBox (" Area = " & area)

End Sub
```

Exemplo 2.8
{O retorno médio de um investimento }

Um usuário possui aplicado dinheiro em três tipos de investimentos que fornecem três tipos de retornos diferentes conforme a tabela a seguir:

Retorno	**Probabilidade**	**Evento**
12	*1/3*	*1*
9	*1/3*	*2*
6	*1/3*	*3*

Sabendo-se que a fórmula do retorno médio é

$$\overline{R} = \frac{\sum_{i=1}^{3} r_i}{3}$$

Fazer um algoritmo que peça ao usuário os três retornos de investimento e assumindo uma probabilidade de ocorrência de sucesso igual para os três investimentos iguais a "p" calcule o retorno médio pela fórmula anterior.

Nesse caso, a probabilidade "p" deve ser solicitada pelo programa ao usuário e a soma dos três retornos r_1, r_2 e r_3 multiplicada por ela. O algoritmo é
:

```
Sub retorno()
Dim p As Single
Dim r1 As Single
Dim r2 As Single
Dim r3 As Single
Dim retmd As Single

p = CSng(InputBox("probabilidade="))
r1 = CSng(InputBox("r1="))
r2 = CSng(InputBox("r2="))
r3 = CSng(InputBox("r3="))
retmd = (r1 + r2 + r3) * p
MsgBox ("o resultado é = " & retmd)

End Sub
```

Exemplo 2.9

{ Algoritmo da adivinhação}

1. *Peça ao usuário para digitar a idade dele.*
2. *Mande uma mensagem para ele pensar num número diferente de zero.*
3. *Mande uma mensagem para ele multiplicar o número pensado por 2.*
4. *Mande ele somar mentalmente o número resultante pelo número 6.*
5. *Mande uma mensagem para ele dividir o resultado pelo número 2.*
6. *Envie uma mensagem para ele subtrair o resultado obtido do valor do primeiro número pensado.*
7. *Mande a mensagem: "O resultado é 3"*

```
Sub ADVINHA()
Dim idade As Integer

idade = CInt(InputBox("Digite sua idade"))
MsgBox ("PENSE NUM NUMERO")
MsgBox ("MULTIPLIQUE O NUMERO POR 2")
MsgBox ("SOME 6 AO RESULTADO")
MsgBox ("DIVIDA O RESULTADO POR 2")
MsgBox ("SUBTRAIA O REULTADO DO NUMERO PENSADO")
MsgBox ("DEU " & 3)
End Sub
```

Por que esse algoritmo funciona e sempre fornece o resultado 3? Observe o que acontece com a matemática das operações que está por trás da brincadeira desse algoritmo.

1. x = fornecido pelo usuário
2. y = 2*x
3. y = y + 6
4. $z = y/2 = \frac{(2*x+6)}{2} = x+3$
5. Resultado = z – x = (x + 3) – x = 3

ogo para qualquer número pensado o resultado sempre será o número 3. O leitor deve estar perguntando porque a variável idade entrou no programa. Para nada! Ela apenas entra para o usuário pensar que a idade tem alguma importância no programa, mas na verdade ela não é usada.

2.3 A Lógica Condicional

Algoritmos onde se efetuam cálculos são bastante parecidos com calculadoras comuns e não demonstram todo o poder da computação na resolução de problemas. A figura de uma condição divisória de eventos na programação é o que faz a lógica de um algoritmo se aproximar da mente humana na resolução de problemas.

Todas as linguagens de programação possuem a estrutura de desvio lógico de um algoritmo. A função para o desvio lógico no VBA é a função SE. Na planilha do Excel ela aparece em língua portuguesa e tem a característica de colocações das lógicas necessárias no parêntesis =SE(condição; verdade; falso). No caso da programação a função é *If* e tem a estrutura

```
If condicao Then
   comando1
   comando2
   .......
   comando n
 Else
   comando1
   comando2
   ........
   comando n
 End If
```

onde os comandos são lógicas operacionais do programa que estarão sujeitas a condição ser verdadeira ou falsa depois do *If*. Se a *condição* for verdadeira *então* a linha logo abaixo do *Then* é verdade e serão obedecidos esses comandos no programa. Caso seja falso depois do *Else* entra os comandos para essa condição. A estrutura *if-then-else* tem um resultado lógico binário que divide o programa em “ramos” verdadeiros e falsos. Mas atenção todo *if* deve terminar com um *End If*.

Exemplo 2.10

Fazer um algoritmo que leia um número inteiro. Se esse número for menor ou igual do que 10 o programa envia uma mensagem ao usuário "prejuízo". Caso contrário mande uma mensagem "lucro".

```
Sub lucro()
Dim numero As Integer

numero = CInt(InputBox("numero = "))

If numero <= 10 Then
   MsgBox ("prejuizo")
Else
   MsgBox ("lucro")
End If

End Sub
```

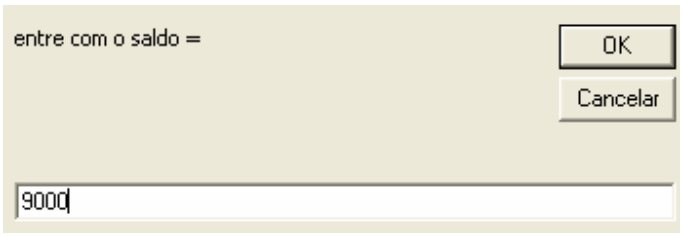
Esse algoritmo desvia depois da leitura o programa para a linha abaixo depois do *Then* caso o número fornecido pelo usuário seja menor que 10. Caso contrário, o compilador manda o programa pular todas as ordens até encontrar o comando *Else* onde então executará esses comandos até encontrar o *End If*.

Exemplo 2.11

Se uma conta bancária contém saldo inferior a R$10.000,00 o banco pagará juros de 2% ao mês na remuneração da conta-poupança. Se o saldo for superior a R$10.000,00 o banco pagará juros de 3% ao mês. Fazer um programa onde o usuário entra com o saldo de sua conta bancária e o programa informa o saldo corrigido.

```
Sub banco()
Dim saldo As Single
Dim saldo_novo As Single
Dim taxa As Single

saldo = CSng(InputBox("entre com o saldo = "))

If saldo < 10000 Then
taxa = 0.02
Else
taxa = 0.03
End If

saldo_novo = saldo * (1 + taxa)
MsgBox ("seu saldo será " & saldo_novo)

End Sub
```

O resultado será



Muitas vezes é necessário encadear muitos "*if*" para traduzir o pensamento lógico desejado na solução de um problema. Além da estrutura *if-then-else* existe a possibilidade do uso do "*elseif*" que significa *caso contrário se*. Esse comando lógico deve ser usado quando é indicado ao computador que ao negar uma afirmação ele o faça em troca de outra pergunta lógica. Isso não significa que sempre que ocorrer uma falsidade na afirmação deve-se utilizar o comando "*elseif*". Às vezes o uso de vários "*if*" um após o outro pode resolver o problema da mesma forma.

Exemplo 2.12

No exemplo 2.10, suponha agora que se a conta possuir saldo inferior a R$10.000,00 o banco pagará juros de 2%. Mas diferente do exemplo anterior, assuma agora que para cada R$5.000,00 adicional acima de R$10.000,00 o banco pagará 0,5% a mais até atingir a taxa limite de 4%. Fazer um programa para calcular quanto uma conta bancária nessas condições terá no final do mês.

```
Sub banco()
Dim saldo As Single
Dim saldo_novo As Single
Dim taxa As Single

saldo = CSng(InputBox("entre com o saldo = "))

If saldo < 10000 Then
taxa = 0.02
   ElseIf saldo < 15000 Then
   taxa = 0.025
        ElseIf saldo < 20000 Then
        taxa = 0.03
          ElseIf saldo < 25000 Then
          taxa = 0.035
          Else
          taxa = 0.04
End If

saldo_novo = saldo * (1 + taxa)
MsgBox ("seu saldo será " & saldo_novo)

End Sub
```

Algumas observações importantes devem ser feitas a cerca desse exemplo com o uso do *'elseif"*. Se o saldo for inferior a R$10.000,00 a taxa assumida será de 0,02 que corresponde a 2%. Mas caso contrário, ou seja, se a taxa é igual ou superior o computador deve perguntar se ela é menor que R$15.000,00. Então a diferença entre um *else* e um *elseif* é que quando existe uma negação e o computador tem um *elseif*, ele não está somente verificando a negação da condição anterior, mas também perguntando se com a primeira negação a segunda condição é satisfeita. Então a parte do programa a seguir

```
If saldo < 10000 Then
taxa = 0.02
   ElseIf saldo < 15000 Then
   taxa = 0.025
```

tem o mesmo significado matemático de um intervalo de validade, ou

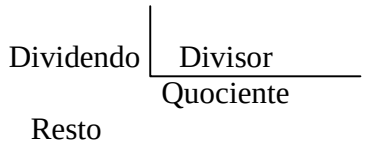
$$10.000 < saldo < 15.000$$

Exercício 2.13

Fazer um programa onde o computador leia um número inteiro fornecido pelo usuário e envia uma mensagem se esse número é par ou ímpar.

Discussão da resolução

Existem diversas maneiras de programar e pensar no algoritmo para resolver esse simples problema. O primeiro é recordar dos tempos colegiais como estão relacionados os termos de uma divisão: divisor, dividendo, quociente e resto.



A relação matemática entre resto e os demais fatores é

$$resto = Dividendo - Divisor * Quociente$$

Então o que se deve fazer é primeiro fazer a divisão entre o número (dividendo) pelo número 2 (divisor) e guardar o resultado numa variável quociente. Definindo essa variável como inteira o resultado sempre será truncado e inteiro. Então se faz a subtração do número que o usuário entrou (dividendo) por duas vezes o quociente. Então se usa a regra lógica para perguntar se o resto é zero ou não. Se for zero o computador envia uma mensagem ao usuário dizendo que o número é par, caso contrário que o número é impar. O programa resolvido é apresentado a seguir.

```
Sub npar_impar()
Dim num As Integer
Dim resto As Integer
Dim quociente As Integer

num = CInt(InputBox("entre com o número = "))

quociente = num / 2
resto = num - 2 * quociente

If resto = 0 Then
   MsgBox ("número par")
Else
   MsgBox ("número impar")
End If

End Sub
```

Uma outra forma de resolver esse problema é fazendo o uso da função *mod* que toda linguagem de programação possui. Essa função já fornece o resto da divisão entre dois números e sua utilização segue a seguinte regra:

x = num1 *mod* num2

e o resultado para x será o resto da divisão de num1 por num2. Então para o exemplo do número par ou ímpar o programa fica como a seguir.

```
Sub npar_impar()
Dim num As Integer
Dim resto As Integer


num = CInt(InputBox("entre com o número = "))

resto = num Mod 2

If resto = 0 Then
   MsgBox ("número par")
Else
   MsgBox ("número impar")
End If

End Sub
```

A última solução apresentada é a mais simples de todas. Para saber se o número é par ou ímpar basta usar a regra matemática da potenciação do número (-1). Ou seja, todo número negativo elevado a número par será positivo e elevado a ímpar será negativo. Então basta programar uma condição lógica que verifica se o sinal dessa operação é positivo ou negativo. A solução é apresentada a seguir.

```
Sub npar_impar()
Dim num As Integer
Dim x As Integer

num = CInt(InputBox("entre com o número = "))

x = (-1) ^ num

If x >= 0 Then
   MsgBox ("número par")
Else
   MsgBox ("número impar")
End If

End Sub
```

Nesse ponto o leitor pode ficar confuso e se perguntar sobre qual solução “é a mais correta?”. A resposta é que todas as formas são corretas desde que o resultado final seja correto. Claro que existem algumas soluções mais rebuscadas e confusas com passos a mais que outras mais claras e objetivas. Esse tipo de análise é que faz a diferença num programador, que se faz entender aos outros ou somente a si próprio em sua programação. No entanto, sob o ponto de vista de análise, se o programa solucionou o problema de forma correta em todos os passos, tanto faz a forma de resolver. O diferencial é que o melhor programador se destaca perante aos outros se além de mais elegante e simples seu programa economizar memória, o que significa economizar energia e tempo dispensado no trabalho.

2.4 Exercícios

(1) Fazer um algoritmo e programa em VBA-Excel onde o usuário entra com um número e o programa imprime uma mensagem dizendo se o número é positivo ou negativo.

(2) Fazer um algoritmo e programa em VBA-Excel onde o usuário entra com 3 números usando InputBox e o programa calcula a média.

(3) Projetar um algoritmo onde o usuário entra com 3 números usando InputBox e o programa descobre e imprime uma mensagem dizendo qual é o maior e qual é o menor.

(4) Fazer um programa onde o usuário entra via InputBox com dois números reais. Se os dois números forem iguais o programa deve enviar uma mensagem com o valor da soma de outras duas variáveis X =2 e Y=1 .Se os dois números forem diferentes o programa deve imprimir uma mensagem mostrando o valor da subtração de X e Y.

(5) Dados 3 lados de um triângulo qualquer, fazer um programa onde ele descobre se o triângulo é eqüilátero, isóceles ou escaleno.

(6) Um comerciante deseja entrar com o valor de compra de uma mercadoria, o valor de venda da mesma e descobrir se o lucro foi <10%, entre 10% e 20%, ou ainda superior a 20%. Fazer um programa que imprima a mensagem dizendo em qual faixa a mercadoria do comerciante se localiza.

(8) A fórmula de conversão de graus Farenheit para centígrados é obtida por

$$C = \frac{5(F - 32)}{9}$$

Fazer um algoritmo e programa em VBA-Excel onde o usuário entra com o grau F (Farenheit) e o programa mostra o valor em centígrados.

(9) Fazer um programa onde o usuário entra com dois números e programa imprime uma mensagem dizendo quantos % o primeiro é menor que o segundo, ou caso contrário, quantos % o primeiro é maior que o segundo.

(10) Lembrar do programa do retorno médio da aula. Fazer um programa onde não mais os eventos tem mesma probabilidade de ocorrer. Ou seja, suponha o caso onde

Retorno	Probabilidade	Evento
12	0,5	1
9	0,4	2
6	0,1	3

Quais são as entradas. Quais são as saídas. Programar novamente o algoritmo levando em conta que as probabilidades dos eventos são diferentes.

(11) Fazer um programa onde o usuário entra com 03 números x1, x2 e x3 e o programa mostra numa MsgBox o valor da media dos números e em outra MsgBox o valor do desvio padrão.

(12) Fazer um programa onde o usuário entra com os limites de um intervalo (inferior e superior) e logo em seguida o computador pede um número qualquer a esse usuário. Se o número pertence ao intervalo do usuário o programa manda uma mensagem “pertence” caso contrário uma mensagem “não pertence”.

Exemplo: Liminf=10
Limsup = 20
Num = 15 ----→ “pertence”
Num = 25 -----→ “não pertence”

(13) O suporte de uma ação é calculado como 30% do intervalo histórico de uma ação (máximo subtraído do mínimo). A resistência é calculada como o valor de 60% do intervalo histórico. Veja o exemplo:

Baixa histórica: 1,50
Alta histórica: 2,20
Suporte: 1,5 + (2,20-1,50)*0,3
Resistência: 1,5 + (2,20 – 1,5)*0,6

Faça um programa onde o usuário forneça o valor mais baixo historicamente e o valor mais alto e o programa diz qual é o suporte e qual é a resistência.

(14) Considere uma equação do segundo grau

$$Ax^2 + Bx + C = 0$$

Utilizando-se da variável $DELTA = B^2 - 4AC$ escreva uma macro que calcule as raízes da equação tal que:

(i) Se não houver raízes (DELTA<0) o programa retorna a mensagem "não existem raízes reais".

(ii) Se houver uma única raiz (DELTA = 0) o programa mostra ao usuário a única raiz calculada como

X1 = -B / (2*a)

(iii) Se houver duas raízes mostre ao usuário calculando-as da forma:

X1 = (-B + Sqr(DELTA)) / (2*A)

X2 = (-B - Sqr(DELTA)) / (2*A)

Onde *Sqr* é a função do VBA para calcular a raiz quadrada de um número real.

(15) Fazer um programa que leia um número. Se o valor for negativo, inverter-lhe o sinal. Em qualquer caso, imprimir a raiz quadrada do número resultante.

(16) Em uma disciplina, a nota final é calculada pela média ponderada das três notas do semestre: trabalho, prova escrita e laboratório. O peso das notas na média é, respectivamente 2, 5 e 3. As notas variam de 0 a 10 e o aluno é considerado aprovado se a nota final for maior ou igual a 6. Elaborar um programa para calcular a nota final de um aluno e enviar uma mensagem se ele foi aprovado ou não.

(17) Fazer um programa para ler três valores {A, B,C}. Se A + B for menor do que C, enviar a mensagem " dados errados". Caso contrário, enviar a mensagem "dados corretos".

(18) Fazer um programa para ler dois números inteiros A e B. Se A = B, atribuir para uma variável X o valor 1.5 e para uma variável Y o valor 2.5. Caso contrário, atribuir os valores -1.5 e -2.5. Em todos os casos, imprimir os valores de X e Y.

(19) Fazer um programa para ler três valores inteiros, A, B e C. Enviar uma mensagem de qual deles é o menor número.

(20) A tabela abaixo fornece os descontos de uma compra. Fazer um programa que leia o valor de uma compra, determine o desconto a ser aplicado e calcule o valor a ser pago pelo cliente.
Tabela:

Valor da Compra (R$)	Desconto (%)
Entre 0 e 20,00	5
Entre 21 e 50	10
Entre 51 e 100	15
Entre 101 e 1000	20
Maior que 1000	30

(21) Escreva um programa para ler um valor de um ângulo em graus, converter esse valor em radianos e enviar o valor em uma mensagem para o usuário.

(22) Dados três pontos (x1,y1), (x2,y2) e (x3,y3), escrever um programa para verificar se esses pontos estão alinhados. A condição de alinhamentos de três pontos é:

$$\frac{y_3 - y_1}{x_3 - x_1} = \frac{y_2 - y_3}{x_2 - x_3}$$