TECNOBYTE INFORMÁTICA

Lógica de programação

Usando as linguagens Pascal e C

Daniel P. Guimarães

Ji-Paraná – Rondônia, maio de 2014

Sumário

Introdução

Lógica de programação corresponde à lógica aplicada no desenvolvimento de programas para computador. Os comandos necessários para montagem de um programa para computador são escritos seguindo o padrão definido por uma linguagem de programação.

Linguagem de programação

Uma linguagem de programação é como um idioma usado para escrever programas para computador. Existem várias linguagens de programação e cada uma possui uma sintaxe e um conjunto de comandos, com os quais podemos escrever programas para uma infinidade de propósitos, tais como automação comercial, gestão de negócios, websites, etc. Veja algumas das linguagens de programação mais usadas atualmente:

- Pascal
- C
- C++
- C# (C sharp)
- Java
- Java Script
- PHP

Código-fonte

Chama-se código-fonte o conjunto de comandos digitados pelo programador, conforme as regras de uma linguagem de programação, a partir do qual se obtém o programa que será entregue ao usuário final.

Exemplo em Pascal:

```
program AloMundo;
begin
  WriteLn('Alo mundo!');
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  printf("Alo mundo!\n");
  getch();
}
```

Comentários

Comentários são informações que o programador insere no código-fonte, mas que são ignoradas pela linguagem de programação, ou seja, não interfere na execução do programa. Geralmente usa-se comentários para inserir anotações de interesse do próprio programador ou para informar a outros programadores sobre algo relevante em determinado trecho do código-fonte.

Exemplo em Pascal:

```
program ExComentario;
begin
  // Isto é um comentário.
  WriteLn('Primeiro texto');
  { Outra forma de comentário,
    delimitado por chaves. }
  WriteLn('Segundo texto');
  (* Comentário também pode ser
     desta forma, delimitado por
     parêntesis e asteriscos. *)
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  // Isto é um comentário.
  printf("Primeiro texto\n");
  /* Outra forma de comentário,
     delimitado por barras e asteriscos. */
  printf("Segundo texto\n");
  getch();
}
```

Interpretador

Chamamos de interpretador qualquer programa que seja capaz de ler o código-fonte de um programa e executar os comandos nele contidos. Algumas linguagens de programação são chamadas de ***interpretadas*** porque o código-fonte é executado diretamente por um programa interpretador, como é o caso das linguagens **Java Script** e **PHP**.

Compilador (Compiler)

Compilador é o nome que se dá a um programa de computador capaz de traduzir os comandos escritos em uma linguagem de programação para uma linguagem que pode ser executada pelo computador. Uma linguagem é chamada de ***compilada*** quando o código-fonte precisa ser compilado para gerar o programa que será executado pelo usuário, como acontece nas linguagens Pascal, C e C++.

Ligador (Linker)

O ligador é um programa de computador capaz de montar um arquivo executável a partir de um ou mais módulos previamente compilados. Em muitos ambientes de programação a compilação e ligação são feitas por um único programa.

Entrada e saída de dados

Geralmente um programa recebe dados de uma fonte externa, processa estes dados e apresenta um resultado. O ato de receber dados de uma fonte externa é chamado de **entrada de dados** e a apresentação de resultados é chamada de **saída de dados**. A entrada de dados mais comum é feita por meio da digitação, enquanto que as saída de dados na maioria dos casos se dá por meio da escrita de dados na tela do computador ou em papel por meio da impressora.

Exemplo em Pascal:

```
program ExEntradaSaida;
var
  Nome: string;
begin
  Write('Digite seu nome: ');           // Saída
  ReadLn(Nome);                         // Entrada
  WriteLn('Voce digitou isto: ', Nome); // Saída
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  char Nome[30];
  printf("Digite seu nome: ");             // Saída
  gets(Nome);                              // Entrada
  printf("Voce digitou isto: %s\n", Nome); // Saída
  getch();
}
```

Tipos de dados

Um tipo de dado define o conjunto de valores possíveis que pode ser armazenado em um determinado espaço da memória do computador.

Principais tipos de dados em Pascal:

- Char (caractere)
- Integer (número inteiro)
- String (sequência de caracteres)
- Real (número real)
- Boolean (lógico – verdadeiro/falso – true/false)

Principais tipos de dados em C:

- char (caractere)
- int (número inteiro)
- float (número real)
- double (número real duplo)

Operadores

Um operador é um sinal que executa uma operação sobre um ou mais valores ou expressões e retorna um valor de acordo com a operação executada. Em uma linguagem de programação, os operadores se dividem em aritméticos, relacionais, lógicos e de atribuição.

Principais operadores em Pascal

Operador	Operação	Exemplo	Resultado
+	Soma	5 + 2	7
-	Subtração	5 – 2	3
*	Multiplicação	5 * 2	10
/	Divisão de real	5 / 2	2,5
div	Divisão de inteiro	5 div 2	2
mod	Resto da divisão de inteiro	5 mod 2	1
=	Igual a	5 = 2	False
<>	Diferente de	5 <> 2	True
>	Maior que	5 > 2	True
>=	Maior ou igual a	5 >= 5	True
<	Menor que	5 < 2	False
<=	Menor ou igual a	5 <=5	True
not	Negação	not True	False
and	E	True and False	False
or	Ou	True or False	True
xor	Ou exclusivo	True xor True	False
:=	Atribuição	X := 5	X recebe 5

Principais operadores em C

Operador	Operação	Exemplo	Resultado
+	Soma	5 + 2	7
-	Subtração	5 – 2	3
*	Multiplicação	5 * 2	10
/	Divisão de real	5 / 2	2,5
%	Resto da divisão de inteiro	5 % 2	1
==	Igual a	5 == 2	false
!=	Não igual a	5 != 2	true
>	Maior que	5 > 2	true
>=	Maior ou igual a	5 >= 5	true
<	Menor que	5 < 2	false
<=	Menor ou igual a	5 <=5	true

!	Negação	!(true)	false
&&	E	true && false	false
||	Ou	true || false	true
=	Atribuição	X = 5	X recebe 5
++	Incremento	X++	X recebe X + 1
--	Decremento	X--	X recebe X – 1
+=	Soma e atribuição	X += 2	X recebe X + 2
-=	Subtração e atribuição	X -= 2	X recebe X - 2
*=	Multiplicação e atribuição	X *= 2	X recebe X * 2
/=	Divisão e atribuição	X /= 2	X recebe X / 2

Variáveis

Uma variável é um espaço reservado temporariamente na memória do computador onde podemos guardar um dado de um tipo específico. Uma variável pode ser utilizada para receber uma entrada de dados (digitação, por exemplo), bem como para armazenar o resultado de uma operação.

Exemplo em Pascal:

```
program ExVariavel;
var
  Parcela1, Parcela2, Soma: Real;       // Variáveis
begin
  Write('Digite a primeira parcela: '); // Saída
  ReadLn(Parcela1);                     // Entrada
  Write('Digite a segunda parcela: ');  // Saída
  ReadLn(Parcela2);                     // Entrada
  Soma := Parcela1 + Parcela2;          // Calcula a soma
  WriteLn('Resultado: ', Soma);         // Saída do resultado
  ReadLn;
end.
```

Exemplo em C:

```c
#include <stdio.h>
main()
{
  float Parcela1, Parcela2, Soma;         // Variáveis
  printf("Digite a primeira parcela: "); // Saída
  scanf("%f", &Parcela1);                 // Entrada
  printf("Digite a segunda parcela: ");  // Saída
  scanf("%f", &Parcela2);                 // Entrada
  Soma = Parcela1 + Parcela2;             // Calcula a soma
  printf("Resultado: %f", Soma);          // Saída do resultado
  printf("\n");                           // Quebra de linha
  getch();
}
```

Constantes

Uma constante corresponde a um identificador (nome) que associamos a um dado específico, de modo que possamos usar este identificador sempre que quisermos trabalhar com o dado representado pela constante.

Exemplo em Pascal:

```pascal
program ExConstante;
const
  PI = 3.14;  // Constante
var
  Raio, Area: Real;  // Variáveis
begin
  Write('Digite o comprimento do raio: ');
  ReadLn(Raio);
  Area := PI * Raio * Raio;
  WriteLn('Area do círculo: ', Area);
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
#define PI 3.14; // Constante
main()
{
  float Raio, Area; // Variáveis
  printf("Digite o comprimento do raio: ");
  scanf("%f", &Raio);
  Area = Raio * Raio * PI;
  printf("Area do circulo: %f\n", Area);
  getch();
}
```

Desvios condicionais

Desvio condicional é a mudança no fluxo de execução de um programa conforme determinada condição. Isto é útil quando o programa precisa decidir se determinado comando ou conjunto de comandos deve ou não ser executado. Se, por exemplo, considerarmos um programa que calcula o quociente entre dois números, tal programa deverá primeiro verificar se o divisor é diferente de zero, pois sendo zero a divisão não poderá ser feita.

IF..ELSE

O desvio condição do tipo **if..else** (se..senão) geralmente tem as seguintes formas:

- **if <condição>**
  <comando>

  Se <condição> for verdadeira, então <comando> será executado.

Exemplo em Pascal:

```
program ExSe;
var
  X: Integer;
begin
  Write('Digite um numero: ');
  ReadLn(X);
  if X > 10 then
    WriteLn('Numero maior que dez.');
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  int X;
  printf("Digite um numero: ");
  scanf("%d", &X);
  if (X > 10)
    printf("Numero maior que dez.\n");
  getch();
}
```

- **if <condição>**
  <comando1>
  else
  <comando2>

  Se <condição> for verdadeira, o <comando1> será executado. Caso contrário, <comando2> será executado:

Exemplo em Pascal:

```
program ExSeSenao;
var
  Dividendo, Divisor: Real;
begin
  Write('Digite o dividendo: ');
  ReadLn(Dividendo);
  Write('Digite o divisor: ');
  ReadLn(Divisor);
  if Divisor = 0 then
    WriteLn('Impossivel divisao por zero.')
  else
    WriteLn('Quociente: ', Dividendo / Divisor);
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  float Dividendo, Divisor;
  printf("Digite o dividendo: ");
  scanf("%f", &Dividendo);
  printf("Digite o divisor: ");
  scanf("%f", &Divisor);
  if (Divisor == 0)
    printf("Impossivel divisao por zero.\n");
  else
    printf("Quociente: %f\n", Dividendo / Divisor);
  getch();
}
```

- **if <condição1>**
  <comando1>
  else if <condição2>
  <comando2>
  else
  <comando3>

  Se <condição1> for verdadeira, <comando1> será executado. Caso contrário, se <condição2> for verdadeira, <comando2> será executado. Caso contrário, <comando3> será executado.

Exemplo em Pascal:

```
program SeSenaoSe;
var
  Sexo: Char;
begin
  Write('Digite o sexo (M/F): ');
  ReadLn(Sexo);
  if Sexo = 'M' then
    WriteLn('Masculino')
  else if Sexo = 'F' then
    WriteLn('Feminino')
  else
    WriteLn('Indefinido');
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  char Sexo;
  printf("Digite o sexo (M/F): ");
  scanf("%c", &Sexo);
  if (Sexo == 'M')
    printf("Masculino\n");
  else if (Sexo == 'F')
    printf("Feminino\n");
  else
    printf("Indefinido\n");
  getch();
}
```

SWITCH/CASE

Quando temos que desviar o fluxo de execução de um programa levando em consideração os vários valores que um seletor (variável ou expressão) pode assumir, podemos usar o desvio condicional do tipo **switch/case**. A forma como este tipo de desvio é escrito varia um pouco de uma linguagem de programação para outra, mas o resultado prático é basicamente o mesmo.

> *Nas linguagens Pascal e C, a variável ou expressão a ser comparada com o comando switch/case deve ser de um tipo de dado ordinal. Tipo de dado ordinal é aquele em que podemos determinar o sucessor ou antecessor de um valor especificado (exemplos: int, integer, char).*

Exemplo em Pascal:

```
program ExCase;
var
  EstadoCivil: Integer;
begin
  WriteLn('Opções de estado civil');
  WriteLn('----------------------');
  WriteLn('1 = Solteiro(a)');
  WriteLn('2 = Casado(a)');
  WriteLn('3 = Viuvo(a)');
  WriteLn('4 = Separado(a)');
  WriteLn('5 = Divorciado(a)');
  WriteLn('----------------------');
  Write('Informe seu estado civil: ');
  ReadLn(EstadoCivil);
  Write('Seu estado civil: ');
  case EstadoCivil of
    1: WriteLn('Solteiro(a)');
    2: WriteLn('Casado(a)');
    3: WriteLn('Viuvo(a)');
    4: WriteLn('Separado(a)');
    5:WriteLn('Divorciado(a)');
  else
    WriteLn('Indefinido');
  end;
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  int EstadoCivil;
  printf("Opções de estado civil\n");
  printf("----------------------\n");
  printf("1 = Solteiro(a)\n");
  printf("2 = Casado(a)\n");
  printf("3 = Viuvo(a)\n");
  printf("4 = Separado(a)\n");
  printf("5 = Divorciado(a)\n");
  printf("----------------------\n");
  printf("Informe seu estado civil: ");
  scanf("%d", &EstadoCivil);
  printf("Seu estado civil: ");
  switch (EstadoCivil)
  {
    case 1:
      printf("Solteiro(a)\n");
      break;
    case 2:
      printf("Casado(a)\n");
      break;
    case 3:
      printf("Viuvo(a)\n");
      break;
    case 4:
      printf("Separado(a)\n");
      break;
    case 5:
      printf("Divorciado(a)\n");
      break;
    default:
      printf("Indefinido\n");
  }
  getch();
}
```

Repetições

Quando um programa precisa executar repetidas vezes um mesmo comando ou grupo de comandos, utilizamos uma ***estrutura de repetição***, também conhecida como ***laço*** ou ***laço de repetição***. Existem basicamente dois tipos de laços, que são laços de contagem e laços condicionais. Os laços condicionais

podem se apresentar com a condição no início ou no fim da estrutura. Vejamos alguns exemplos.

Laço de contagem crescente: mostrar os números de 0 a 9.

Exemplo em Pascal:

```
program ExLacoFor;
var
  I: Integer;
begin
  for I := 0 to 9 do // Para I de 0 a 9 faça...
    WriteLn(I);
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  int I;
  for (I = 0; I <= 9; I++)
    printf("%d\n", I);
  getch();
}
```

Laço de contagem decrescente: mostrar os números de 9 a 0:

Exemplo em Pascal:

```
program ExForDownTo;
var
  I: Integer;
begin
  for I := 9 downto 0 do
    WriteLn(I);
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  int I;
  for (I = 9; I >= 0; I--)
    printf("%d\n", I);
  getch();
}
```

Laço condicional com condição no início: neste exemplo mostraremos os números pares de 0 a 10, testando a condição no início do laço.

Exemplo em Pascal:

```
program ExWhile;
var
  X: Integer;
begin
  X := 0;
  while X <= 10 do // Enquanto X <= 10 faça...
  begin
    WriteLn(X);
    X := X + 2;
  end;
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  int X;
  X = 0;
  while (X <= 10) // Enquanto X <= 10
  {
    printf("%d\n", X);
    X = X + 2;
  }
  getch();
}
```

Laço condicional com condição no fim: neste exemplo também mostraremos os números pares de 0 a 10, mas testando a condição no final do laço.

Exemplo em Pascal:

```
program ExRepeatUntil;
var
  X: Integer;
begin
  X := 0;
  repeat // Repetir...
    WriteLn(X);
    X := X + 2;
  until X > 10; // .. até que X > 10
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>
main()
{
  int X;
  X = 0;
  do // Faça...
  {
    printf("%d\n", X);
    X = X + 2;
  } while (X <= 10); // enquanto X <= 10
  getch();
}
```

Matrizes e vetores

Uma matriz é uma coleção de dados de um mesmo tipo, acessíveis por meio de um único nome (variável ou constante) e armazenados contiguamente na memória. Para acessar cada elemento de uma matriz devemos informar o nome da matriz (variável ou constante) e um ou mais índices que identificam a posição do elemento dentro da matriz. Uma matriz de uma única dimensão é chamada de vetor.

Um vetor é semelhante a uma lista, contendo ***n*** elementos de um mesmo tipo de dado. Uma matriz bidimensional é semelhante a uma tabela, em que as células representam os elementos de dados, todos de um mesmo tipo.

Ao declarar um vetor ou matriz na linguagem Pascal, devemos informar os índices do primeiro e último elemento cada dimensão. Já na linguagem C devemos informar apenas o tamanho de cada dimensão do vetor ou matriz, pois o índice do primeiro elemento será sempre zero.

Exemplo em Pascal:

```
var
  // Vetor para 5 valores do tipo real
  MeuVetor: array[1..5] of Real;

  // Matriz para uma tabela com 2 linhas e 5 colunas
  MinhaMatriz: array[1..2,1..5] of Integer;
```

Exemplo em C:

```
// Vetor para 5 elementos do tipo float
float MeuVetor[5];

// Matriz para uma tabela com 2 linhas e 5 colunas
int MinhaMatriz[2][5];
```

Para acessar um elemento de um vetor devemos informar o índice do elemento, ou seja, a posição do elemento dentro do vetor. No caso da matriz, por ter duas ou mais dimensões, devemos informar um índice para cada dimensão da matriz, de modo que seja possível chegar ao elemento desejado a partir dos índices informados.

Exemplo em Pascal – Vetor:

```
program ExVetor;
var
  MeuVetor: array[1..3] of Integer;
begin
  MeuVetor[1] := 60;
  MeuVetor[2] := 70;
  MeuVetor[3] := 90;
  WriteLn('Primeiro elemento: ', MeuVetor[1]);
  WriteLn('Segundo elemento: ', MeuVetor[2]);
  WriteLn('Terceiro elemento: ', MeuVetor[3]);
  ReadLn;
end.
```

Exemplo em C – Vetor:

```c
#include <stdio.h>
main()
{
  int MeuVetor[3];
  MeuVetor[0] = 60;
  MeuVetor[1] = 70;
  MeuVetor[2] = 90;
  printf("Primeiro elemento: %d\n", MeuVetor[0]);
  printf("Segundo elemento: %d\n", MeuVetor[1]);
  printf("Terceiro elemento: %d\n", MeuVetor[2]);
  getch();
}
```

Exemplo em Pascal – Matriz:

```pascal
program ExMatriz;
var
  // Declara matriz para tabela 2 x 3
  Tab: array[0..1,0..2] of Integer;
begin
  // Preenche a primeira linha da tabela.
  Tab[0,0] := 60;
  Tab[0,1] := 70;
  Tab[0,2] := 80;

  // Preenche a segunda linha da tabela.
  Tab[1,0] := 100;
  Tab[1,1] := 110;
  Tab[1,2] := 120;

  // Mostra a tabela.
  WriteLn(Tab[0,0]:5, Tab[0,1]:5, Tab[0,2]:5);
  WriteLn(Tab[1,0]:5, Tab[1,1]:5, Tab[1,2]:5);

  ReadLn;
end.
```

Exemplo em C – Matriz:

```
#include <stdio.h>
main()
{
  // Declara matriz para tabela 2 x 3
  int Tab[2][3];

  // Preenche a primeira linha da tabela.
  Tab[0][0] = 60;
  Tab[0][1] = 70;
  Tab[0][2] = 80;

  // Preenche a segunda linha da tabela.
  Tab[1][0] = 100;
  Tab[1][1] = 110;
  Tab[1][2] = 120;

  // Mostra a tabela.
  printf("%5d %5d %5d\n", Tab[0][0], Tab[0][1], Tab[0][2]);
  printf("%5d %5d %5d\n", Tab[1][0], Tab[1][1], Tab[1][2]);

  getch();
}
```

Estruturas de dados heterogêneas

Como vimos anteriormente, um vetor ou matriz é uma estrutura de dados homogênea, ou seja, todos os elementos da estrutura são de um mesmo tipo de dado. Muitas linguagens de programação suportam também estruturas de dados heterogêneas, ou seja, estruturas com elementos de tipos diferentes. Como exemplo, considere uma ficha cadastral de uma pessoa, onde temos atributos como código, nome, peso, etc.

Exemplo em Pascal:

```
program ExRecord;

// Declara estrutura Pessoa.
type
  Pessoa = record
    Codigo: Integer;
    Nome: string;
    Peso: Real;
  end;

var
  // Declara variável P do tipo Pessoa.
  P: Pessoa;
begin

  // Armazena dados na estrutura.
  P.Codigo := 20;
  P.Nome := 'Daniel';
  P.Peso := 75.5;

  // Apresenta os dados da estrutura.
  WriteLn('CADASTRO');
  WriteLn;
  WriteLn('Código...: ', P.Codigo);
  WriteLn('Nome.....: ', P.Nome);
  WriteLn('Peso.....: ', P.Peso:0:3, 'kg');
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>

// Declara estrutura Pessoa.
struct Pessoa
{
  int Codigo;
  char Nome[30];
  float Peso;
};

main()
{
  // Declara variável P do tipo Pessoa.
  struct Pessoa P;

  // Armazena dados na estrutura.
  P.Codigo = 20;
  strcpy(P.Nome, "Daniel"); // Copia "Daniel" p/ P.Nome
  P.Peso = 75.5;

  // Apresenta os dados da estrutura.
  printf("CADASTRO\n\n");
  printf("Codigo...: %d\n", P.Codigo);
  printf("Nome.....: %s\n", P.Nome);
  printf("Peso.....: %0.3fkg\n\n", P.Peso);
  getch();
}
```

Funções e procedimentos

Em programação de computador, é comum dividir o código-fonte em partes lógicas chamadas funções e procedimentos, que são conjuntos de comandos que executam tarefas específicas.

Uma função ou procedimento possui um nome e nenhum ou vários argumentos (parâmetros). Argumentos são os dados que devemos passar para a função ao executá-la. Por exemplo, ao chamar (executar) uma função que calcula a soma de dois números, devemos passar para ela os dois números que devem ser somados.

A diferença entre função e procedimento é que função retorna um valor para a rotina que a chamou.

*Na linguagem C não existe procedimento, mas existe função sem retorno, que corresponde ao procedimento da linguagem Pascal. Para declarar na linguagem C uma função que não tem retorno devemos informar a palavra **void** no tipo de retorno da função.*

Exemplo em Pascal – Função sem parâmetro:

```
program ExFuncaoSemParam;

function SolicitaPeso: Real;
var
  Retorno: Real;
begin
  Write('Informe seu peso: ');
  ReadLn(Retorno);
  SolicitaPeso := Retorno;
end;

var
  Peso: Real;
begin
  Peso := SolicitaPeso;
  WriteLn('Peso informado: ', Peso:10:3, 'kg');
  ReadLn;
end.
```

Exemplo em C – Função sem parâmetro:

```
#include <stdio.h>

float SolicitaPeso()
{
  float Retorno;
  printf("Informe seu peso: ");
  scanf("%f", &Retorno);
  return Retorno;
}

main()
{
  float Peso;
  Peso = SolicitaPeso();
  printf("Peso informado: %10.3fkg\n", Peso);
  getch();
}
```

Exemplo em Pascal – Função com parâmetros:

```
program ExFuncaoComParam;

function CalcAreaRetangulo(A, B: Real): Real;
begin
  CalcAreaRetangulo := A * B;
end;

var
  LadoA, LadoB, Area: Real;
begin
  WriteLn('Calcula area do retangulo');
  WriteLn;
  Write('Lado A: ');
  ReadLn(LadoA);
  Write('Lado B: ');
  ReadLn(LadoB);
  Area := CalcAreaRetangulo(LadoA, LadoB);
  WriteLn('Area: ', Area:10:2);
  ReadLn;
end.
```

Exemplo em C – Função com parâmetros:

```
#include <stdio.h>

float CalcAreaRetangulo(float A, float B)
{
  return A * B;
}

main()
{
  float LadoA, LadoB, Area;
  printf("Calcula area do retangulo\n");
  printf("Lado A: ");
  scanf("%f", &LadoA);
  printf("Lado B: ");
  scanf("%f", &LadoB);
  Area = CalcAreaRetangulo(LadoA, LadoB);
  printf("Area: %10.2f\n", Area);
  getch();
}
```

Exemplo em Pascal – Procedimento sem parâmetro:

```
program ExProcSemParam;

procedure MostraDados;
begin
  WriteLn('Programa: ExProcSemParam');
  WriteLn('Versão..: 1.0');
end;

begin
  MostraDados;
  ReadLn;
end.
```

Exemplo em C – Função sem retorno e sem parâmetro:

```
#include <stdio.h>

void MostraDados()
{
  printf("Programa: ExProcSemParam\n");
  printf("Versao..: 1.0\n");
}

main()
{
  MostraDados();
  getch();
}
```

Exemplo em Pascal – Procedimento com parâmetros:

```
program ExProcComParam;

procedure MostraCampo(Legenda, Valor: string);
begin
  while Length(Legenda) < 20 do
    Legenda := Legenda + '.';
  WriteLn(Legenda, ': ', Valor);
end;

begin
  MostraCampo('Nome', 'Daniel');
  MostraCampo('Home-page', 'www.tecnobyte.com.br');
  MostraCampo('E-mail', 'daniel@tecnobyte.com.br');
  ReadLn;
end.
```

Exemplo em C – Função sem retorno e com parâmetro:

```
#include <stdio.h>

void MostraCampo(char* Legenda, char* Valor)
{
  int I;
  printf("%s", Legenda);
  for (I = strlen(Legenda); I < 20; I++)
    printf("%c", '.');
  printf(": %s\n", Valor);
}

main()
{
  MostraCampo("Nome", "Daniel");
  MostraCampo("Home-page", "www.tecnobyte.com.br");
  MostraCampo("E-mail", "daniel@tecnobyte.com.br");
  getch();
}
```

Parâmetros por valor e por referência

Parâmetros podem ser passados para funções e procedimentos de duas formas: **por valor** ou **por referência**. Na passagem de parâmetro por valor, uma cópia do conteúdo é enviada para a função ou procedimento, enquanto que na passagem de parâmetro por referência é enviado somente o endereço de memória que armazena o conteúdo do parâmetro.
Se um parâmetro passado por valor for modificado dentro da função ou procedimento, esta alteração não será refletida para fora da função ou procedimento, pois estamos trabalhando com uma cópia do conteúdo passado por parâmetro.

Por outro lado, se um parâmetro passado por referência for modificado dentro da função ou procedimento, esta alteração será refletida para fora da função ou procedimento, visto que estaremos alterando os dados que estão armazenados no endereço de memória que foi passado para a função ou procedimento.

Na linguagem Pascal, usamos a palavra ***var*** na declaração de um parâmetro quando queremos passá-lo por referência. Outra forma é passar um ponteiro (endereço de memória), como se faz na linguagem C. Os detalhes técnicos envolvidos no uso de ponteiros estão fora do escopo desta obra e, portanto, não serão abordados aqui.

Nos exemplos a seguir, mostrarei como obter o retorno de uma função ou procedimento usando a passagem de parâmetro por referência.

Exemplo em Pascal:

```
program ExParamPorRef;

procedure CalculaSoma(P1, P2: Integer; var R: Integer);
begin
  R := P1 + P2;
end;

var
  A, B, C: Integer;
begin
  A := 2;
  B := 3;
  CalculaSoma(A, B, C);
  WriteLn('A = ', A);
  WriteLn('B = ', B);
  WriteLn('C = ', C);
  ReadLn;
end.
```

Exemplo em C:

```
#include <stdio.h>

int CalculaSoma(int P1, int P2, int *R)
{
  *R = P1 + P2;
}

main()
{
  int A, B, C;
  A = 2;
  B = 3;
  CalculaSoma(A, B, &C);
  printf("A = %d\n", A);
  printf("B = %d\n", B);
  printf("C = %d\n", C);
  getch();
}
```

Exercícios resolvidos

1. Escreva um programa que desenhe um retângulo na tela do computador usando os caracteres hífen e barra vertical.

Solução em Pascal

```
program ExRetangulo;
begin
  WriteLn('|--------------|');
  WriteLn('|              |');
  WriteLn('|              |');
  WriteLn('|--------------|');
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  printf("|--------------|\n");
  printf("|              |\n");
  printf("|              |\n");
  printf("|--------------|\n");
  getch();
}
```

2. Escreva um programa que receba dois números inteiros, calcule a soma e mostre o resultado na tela.

Solução em Pascal

```
program ExSoma;
var
  A, B: Integer;
begin
  Write('Informe um numero: ');
  ReadLn(A);
  Write('Informe outro numero: ');
  ReadLn(B);
  WriteLn('Soma: ', A + B);
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  int A, B;
  printf("Informe um numero: ");
  scanf("%d", &A);
  printf("Informe outro numero: ");
  scanf("%d", &B);
  printf("Soma: %d\n", A + B);
  getch();
}
```

3. Escreva um programa que receba um nome e mostre como resultado a quantidade de caracteres contidos no nome informado.

Solução em Pascal

```
program ExTamanhoNome;
var
  Nome: string;
begin
  Write('Digite seu nome: ');
  ReadLn(Nome);
  WriteLn('Seu nome tem ', Length(Nome), ' caractere(s).');
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  char Nome[30];
  printf("Digite seu nome: \n");
  gets(Nome);
  printf("Seu nome tem %d caractere(s).\n", strlen(Nome));
  getch();
}
```

4. Escreva um programa que recebe a nota do aluno. Se a nota informada for maior ou igual a 7,0, mostre na tela a palavra “Aprovado”, caso contrário mostre a palavra “Reprovado”.

Solução em Pascal

```
program ExNotaProva;
var
  Nota: Real;
begin
  Write('Informe a nota: ');
  ReadLn(Nota);
  if Nota >= 7.0 then
    WriteLn('Aprovado')
  else
    WriteLn('Reprovado');
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  float Nota;
  printf("Informe a nota: ");
  scanf("%f", &Nota);
  if (Nota >= 7.0)
    printf("Aprovado\n");
  else
    printf("Reprovado\n");
  getch();
}
```

5. Escreva um programa que receba uma string qualquer e mostre como resultado a mesma string, porém invertida (exemplo: recebe RONDÔNIA e mostra AINÔDNOR).

Solução em Pascal

```
program ExInverteNome;
var
  Nome: string;
  I: Integer;
begin
  Write('Informe um nome: ');
  ReadLn(Nome);
  for I := Length(Nome) downto 1 do
    Write(Nome[I]);
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  char Nome[30];
  int I;
  printf("Informe um nome: ");
  gets(Nome);
  for (I = strlen(Nome); I > 0; I--)
    printf("%c", Nome[I - 1]);
  printf("\n");
  getch();
}
```

6. Escreva um programa que receba um número inteiro e como resultado apresente a tabuada de multiplicação de 0 a 10 do número informado, na forma abaixo (considerando que 3 foi o número informado):

```
3 x 0 = 0
3 x 1 = 3
...
3 x 10 = 30
```

Solução em Pascal

```
program ExTabuada;
var
  Numero, I: Integer;
begin
  Write('Informe um numero: ');
  ReadLn(Numero);
  for I := 0 to 10 do
    WriteLn(Numero, ' x ', I, ' = ', Numero * I);
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  int Numero, I;
  printf("Informe um numero: ");
  scanf("%d", &Numero);
  for (I = 0; I <= 10; I++)
    printf("%d x %d = %d\n", Numero, I, Numero * I);
  getch();
}
```

7. Escreva um programa que receba o valor do salário e a alíquota do INSS, calcule e mostre o valor do INSS formatado com 2 casas decimais. Em seguida o programa deverá perguntar se quer fazer um novo cálculo e se o usuário responder com "S" o programa deverá repetir o processo. Se o usuário responder a pergunta com algo diferente de "S", o programa deverá encerrar.

Solução em Pascal

```
program ExCalculaINSS;
var
  ValorSalario, AliquotaINSS, ValorINSS: Real;
  Resposta: Char;
begin
  repeat
    Write('Valor do salario: ');
    ReadLn(ValorSalario);
    Write('Aliquota do INSS: ');
    ReadLn(AliquotaINSS);
    ValorINSS := ValorSalario * AliquotaINSS / 100;
    WriteLn('Valor do INSS: ', ValorINSS:0:2);
    Write('Calcular novamente? (S/N): ');
    ReadLn(Resposta);
  until Resposta <> 'S';
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  float ValorSalario, AliquotaINSS, ValorINSS;
  char Resposta;
  do
  {
    printf("Valor do salario: ");
    scanf("%f", &ValorSalario);
    printf("Aliquota do INSS: ");
    scanf("%f", &AliquotaINSS);
    ValorINSS = ValorSalario * AliquotaINSS / 100;
    printf("Valor do INSS: %.2f\n", ValorINSS);
    printf("Calcular novamente? (S/N): ");
    fflush(stdin);
    scanf("%c", &Resposta);
  } while (Resposta == 'S');
}
```

8. Escreva um programa que mostre todos os números pares de 0 a 30.

Soluções em Pascal

```
// Solução usando for
program ExParesFor;
var
  N: Integer;
begin
  for N := 0 to 15 do
    WriteLn(N * 2);
  ReadLn;
end.

// Solução com while
program ExParesWhile;
var
  N: Integer;
begin
  N := 0;
  while N <= 30 do
  begin
    WriteLn(N);
    N := N + 2;
  end;
  ReadLn;
end.

// Solução com repeat..until
program ExParesRepeat;
var
  N: Integer;
begin
  N := 0;
  repeat
    WriteLn(N);
    N := N + 2;
  until N > 30;
  ReadLn;
end.
```

Soluções em C++

```
// Solução usando for
#include <stdio.h>
main()
{
  int N;
  for (N = 0; N <= 30; N += 2)
    printf("%d\n", N);
  getch();
}

// Solução usando while
#include <stdio.h>
main()
{
  int N = 0;
  while (N <= 30)
  {
    printf("%d\n", N);
    N += 2;
  }
  getch();
}

// Solução usando do..while
#include <stdio.h>
main()
{
  int N = 0;
  do
  {
    printf("%d\n", N);
    N += 2;
  } while (N <= 30);
  getch();
}
```

9. Escreva um programa que receba 5 números inteiros e em seguida mostre uma listagem com o dobro de cada número informado anteriormente.

Solução em Pascal

```
program ExDobroNumeros;
var
  I: Integer;
  Numeros: array[1..5] of Integer;
begin
  for I := 1 to 5 do
  begin
    Write('Informe um numero: ');
    ReadLn(Numeros[I]);
  end;
  WriteLn;
  WriteLn('Dobro dos numeros:');
  for I := 1 to 5 do
    WriteLn(Numeros[I] * 2);
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>
main()
{
  int I, Numeros[5];
  for (I = 1; I <= 5; I++)
  {
    printf("Informe um numero: ");
    scanf("%d", &Numeros[I]);
  }
  printf("\nDobro dos numeros:\n\n");
  for (I = 1; I <= 5; I++)
    printf("%d\n", Numeros[I] * 2);
  getch();
}
```

10. Usando uma estrutura de dados heterogênea, faça um programa que receba os dados de uma pessoa (código, nome e telefone) e em seguida mostre os dados informados no formato abaixo:

```
Código..: 1234
Nome....: MARIA DA SILVA
Telefone: 98764321
```

Solução em Pascal:

```
program ExCadastro;

type
  TCadastro = record
    Codigo: Integer;
    Nome: string;
    Telefone: string;
  end;

var
  Cad: TCadastro;
begin
  WriteLn('Digite os dados');
  WriteLn;
  Write('Codigo: ');
  ReadLn(Cad.Codigo);
  Write('Nome: ');
  ReadLn(Cad.Nome);
  Write('Telefone: ');
  ReadLn(Cad.Telefone);
  WriteLn;
  WriteLn('Dados informados');
  WriteLn;
  WriteLn('Codigo..: ', Cad.Codigo);
  WriteLn('Nome....: ', Cad.Nome);
  WriteLn('Telefone: ', Cad.Telefone);
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>

struct Cadastro
{
  int Codigo;
  char Nome[30];
  char Telefone[15];
};

main()
{
  struct Cadastro Cad;
  printf("Digite os dados\n\n");
  printf("Codigo: ");
  scanf("%d", &Cad.Codigo);
  printf("Nome: ");
  scanf("%s", &Cad.Nome);
  printf("Telefone: ");
  scanf("%s", Cad.Telefone);
  printf("\nDados informados\n\n");
  printf("Codigo..: %d\n", Cad.Codigo);
  printf("Nome....: %s\n", Cad.Nome);
  printf("Telefone: %s\n", Cad.Telefone);
  getch();
}
```

11. Escreva um programa que tenha uma função chamada CalculaDobro. Esta função deverá receber um número real e retornar o dobro deste número. O programa deverá usar esta função para calcular o dobro de um número informado pelo usuário e mostrar o resultado na tela.

Solução em Pascal:

```
program ExFuncaoDobro;

function CalculaDobro(X: Real): Real;
begin
  CalculaDobro := X * 2;
end;

var
  Numero: Real;
begin
  Write('Informe um numero real: ');
  ReadLn(Numero);
  WriteLn('Dobro do numero: ', CalculaDobro(Numero));
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>

float CalculaDobro(float X)
{
  return X * 2;
}

main()
{
  float Numero;
  printf("Informe um numero real: ");
  scanf("%f", &Numero);
  printf("Dobro do numero: %f\n", CalculaDobro(Numero));
  getch();
}
```

12. Escreva um programa que tenha uma função chamada PadR. Esta função deverá receber um texto, um número inteiro e um caractere, e retornar o texto completado à direita com o caractere informado até completar o número de caracteres informado no segundo parâmetro. O programa deverá usar esta função para mostrar o resultado abaixo:

```
Código....:
Nome......:
```

Solução em Pascal:

```
program ExFuncaoPadR;

function  PadR(Texto:  string;  Tamanho:  Integer;  Caractere:
Char): string;
begin
  Texto := Copy(Texto, 1, Tamanho);
  while Length(Texto) < Tamanho do
    Texto := Texto + Caractere;
  PadR := Texto;
end;

begin
  WriteLn(PadR('Codigo do cliente', 10, '.') + ':');
  WriteLn(PadR('Nome', 10, '.') + ':');
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>

char *PadR(char *Texto, int Tamanho,
  char Caractere, char *Retorno)
{
  int I;
  strncpy(Retorno, Texto, Tamanho);
  for (I = strlen(Retorno); I < Tamanho; I++)
    Retorno[I] = Caractere;
  Retorno[Tamanho] = 0; // Terminador nulo
  return Retorno;
}

main()
{
  char Temp[11]; // String + terminador nulo.
  printf("%s:\n", PadR("Codigo", 10, '.', Temp));
  printf("%s:\n",  PadR("Nome", 10, '.', Temp));
  getch();
}
```

13. Escreva uma função que receba como parâmetro o número de um mês (1 a 12) e que retorne o nome do respectivo mês. Para testar a função, o programa deverá solicitar a digitação do número de um mês e usar a função para mostrar o nome do mês digitado.

Solução em Pascal:

```
program ExFuncaoPegaNomeMes;

function PegaNomeMes(Numero: Integer): string;
begin
  case Numero of
    1: PegaNomeMes := 'Janeiro';
    2: PegaNomeMes := 'Fevereiro';
    3: PegaNomeMes := 'Marco';
    4: PegaNomeMes := 'Abril';
    5: PegaNomeMes := 'Maio';
    6: PegaNomeMes := 'Junho';
    7: PegaNomeMes := 'Julho';
    8: PegaNomeMes := 'Agosto';
    9: PegaNomeMes := 'Setembro';
    10: PegaNomeMes := 'Outubro';
    11: PegaNomeMes := 'Novembro';
    12: PegaNomeMes := 'Dezembro';
  else
    PegaNomeMes := 'Invalido';
  end;
end;

var
  NumMes: Integer;
begin
  Write('Informe o numero do mes: ');
  ReadLn(NumMes);
  WriteLn('Mes informado: ', PegaNomeMes(NumMes));
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>

char *PegaNomeMes(int Numero, char *Nome, int TamNome)
{
  switch (Numero)
  {
    case 1: strncpy(Nome, "Janeiro", TamNome - 1); break;
    case 2: strncpy(Nome, "Fevereiro", TamNome - 1); break;
    case 3: strncpy(Nome, "Marco", TamNome - 1); break;
    case 4: strncpy(Nome, "Abril", TamNome - 1); break;
    case 5: strncpy(Nome, "Maio", TamNome - 1); break;
    case 6: strncpy(Nome, "Junho", TamNome - 1); break;
    case 7: strncpy(Nome, "Julho", TamNome - 1); break;
    case 8: strncpy(Nome, "Agosto", TamNome - 1); break;
    case 9: strncpy(Nome, "Setembro", TamNome - 1); break;
    case 10: strncpy(Nome, "Outubro", TamNome - 1); break;
    case 11: strncpy(Nome, "Novembro", TamNome - 1); break;
    case 12: strncpy(Nome, "Dezembro", TamNome - 1); break;
    default: strncpy(Nome, "Invalido", TamNome - 1);
  }
  return Nome;
}

main()
{
  int NumMes;
  char NomeMes[20];
  printf("Numero: ");
  scanf("%d", &NumMes);
  printf("Nome : %s\n", PegaNomeMes(NumMes, NomeMes, 20));
  getch();
}
```

14. Escreva uma função com o nome Confirma, que receberá uma mensagem como parâmetro. Esta função deverá exibir na tela a mensagem informada e solicitar uma resposta do usuário. Se a resposta for "S", a função deverá retornar True. Se a resposta for "N", a função deverá retornar False. Para qualquer outra resposta, a função deverá exibir novamente a mensagem e solicitar nova resposta. Para testar a função use um trecho de código-fonte semelhante a este a seguir:

```
se Confirma("Deseja excluir os dados?")
  Escreva("Exclusão confirmada.")
senão
  Escreva("Exclusão negada.");
```

Solução em Pascal:

```
program ExFuncaoConfirma;

function Confirma(Mensagem: string): Boolean;
var
  Resposta: Char;
begin
  repeat
    Write(Mensagem);
    ReadLn(Resposta);
    Resposta := UpCase(Resposta);
  until (Resposta = 'S') or (Resposta = 'N');
  Confirma := Resposta = 'S';
end;

begin
  if Confirma('Deseja excluir os dados?') then
    WriteLn('Exclusão confirmada.')
  else
    WriteLn('Exclusão negada.');
  ReadLn;
end.
```

Solução em C:

```
#include <stdio.h>

int Confirma(char *Mensagem)
{
  char Resposta;
  do
  {
    printf("%s", Mensagem);
    fflush(stdin);
    scanf("%c", &Resposta);
    Resposta = toupper(Resposta);
  } while (Resposta != 'S' && Resposta != 'N');
  return Resposta == 'S';
}

main()
{
  if (Confirma("Deseja excluir os dados?"))
    printf("Exclusão confirmada.\n");
  else
    printf("Exclusão negada.\n");
  getch();
}
```

Conclusão

Quando escrevi este material não tive como objetivo esgotar o tema proposto, de modo que apresentei apenas uma introdução à lógica de programação, mostrando exemplos em duas linguagens de programação bastante conhecidas. Se você chegou até este ponto e compreendeu o conteúdo, recomendo que escolha uma linguagem de programação que se adeque às suas necessidades e pesquise por materiais que tratem especificamente da linguagem escolhida. Estou certo de que com dedicação e persistência, poderá alcançar o objetivo pretendido. Desejo sucesso nesta nova jornada.