Introdução à Programação em Linguagem Pascal

Parte 1

Fundamentos, Expressões e Comandos de Entrada e Saída

Faculdades Integradas Stella Maris – FISMA
Curso de Tecnologia em Análise e Projeto de Sistemas
Prof. Márcio A. S. Torrente

Andradina – SP

Sumário:

Introdução:

Abordaremos aqui a utilização dos conceitos da programação Pascal por meio de exemplos, enfocando os termos fundamentais da linguagem, a sintaxe dos principais comandos, a lógica envolvida em cada um deles e a aplicação de estruturas de dados que organizam o armazenamento e aperfeiçoam a recuperação rápida das informações.

Estrutura de um programa Pascal:

A estrutura de codificação de um programa em Pascal se apresenta da seguinte forma:

```
Cabeçalho  { Program <identificador-do-programa>;
Área de    { [Uses <lista-de-bibliotecas>;]
Declarações  [Type <declaração-de-tipos-estruturados>;]
             [Const <declaração-de-constantes>;]
             [Var <declaração-de-variáveis>;]
             [Procedure <cabeçalho-do-procedimento>; Begin [<comandos> End;]
             [Function <cabeçalho-da-função>; Begin [<comandos> End;]
Bloco      { Begin
Principal de    <comandos>;
Comandos   { End.
```

Exemplo:

```
Program MenorPrograma;
Begin
   Write('Oi!')
End.
```

ou

```
Program MenorPrograma; Begin Write('Oi!') End.
```

Delimitadores de Comandos:

Como se observa na estrutura do programa e no exemplo acima se usa no Pascal um ponto-e-vírgula (;) para delimitar o término de cada linha de comando, permitindo ao compilador identificar onde se inicia e onde se encerra uma instrução de programa. Então é possível em uma única linha de texto do código-fonte, delimitarmos várias linhas de comando. É usado também pelo Pascal um ponto (.) para marcar o término do Bloco Principal de Comandos, determinando o término do programa.

Palavras Reservadas:

As palavras em negrito nas notações sintáticas são palavras reservadas ou palavras chaves da linguagem. Elas definem um comando, através de um verbo ou substantivo, compondo a sintaxe de uma instrução de programa reconhecida pelo compilador. Veja as principais palavras reservadas do Pascal:

and, asm, array, begin, case, const, constructor, destructor, div, do, downto, else, end, exports, file, for, function, goto, if, implementation, in, inline, interface, label, library, mod, nil, not, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor, entre outras...

Identificador:

Um identificador é uma única palavra que identifica qualquer dado ou rotina de programa. Para criarmos um identificador, devem ser respeitadas as seguintes regras:

1º) Não pode ser iniciado com um dígito numérico;

2º) Não pode conter espaços em branco nem símbolos especiais, exceto o sublinha (underline);

3º) Não pode ser uma palavra reservada;

Não há limite quanto ao tamanho de um identificador, porém, o compilador Pascal considera até 255 caracteres, desprezando os demais. No entanto, não se justifica criar um identificador tão grande que não seja um termo sugestivo ao dado a ser armazenado ou a rotina a ser executada.

Exemplo:

```
Program Potenciacao;
Var potencia, base, expoente : Real;
```

Onde *Potenciacao* é o identificador do programa, e *potencia*, *base* e *expoente* são identificadores das variáveis a serem utilizadas no programa.

Linhas de comentários:

Os comentários são textos explicativos escritos no código-fonte, delimitados pelos caracteres de { e }, ou pelos caracteres (* e *). Os textos de comentário são ignorados pelo compilador na fase de tradução do programa, úteis somente no sentido de documentar uma rotina de instruções dentro do programa, para torná-lo mais legível e fácil de entender, quando retomado para correção ou alteração.

Tipos de Comandos e Blocos de Comandos:

Um Comando Simples é aquele formado por uma única palavra reservada, seguido ou não de parâmetros necessários para sua funcionalidade e delimitado pelo ponto-e-vírgula (;). Um Comando Composto é aquele formado por mais de uma palavra reservada e que contém em sua estrutura um ou mais Comandos Simples ou outros Comandos Compostos. Quando um Comando Composto inclui mais de um comando qualquer, esses devem ser embutidos em sub-blocos de comandos, delimitados entre seu início e fim pelos termos BEGIN e END.

Esquema de estruturas de blocos em Pascal:

```
Begin
   <comando-simples1>;
   <comando-simples2>;
   <comando-composto1>
      Begin
         <comando-simples3>;
         <comando-composto2>
            Begin
               <comando-simples4>;
               <comando-simples5>;
            End;
         <comando-simples6>;
      End;
   <comando-simples7>;
   <comando-composto3>
      Begin
         <comando-simples8>;
         <comando-simples9>;
      End;
   <comando-simples10>;
End.
```

Exemplo de um programa estruturado:

```
Program Fibonacci;
{Programa para imprimir os 30 primeiros
 números da seqüência de Fibonacci:
 (1, 1, 2, 3, 5, 8, 13, 21, ... , 1346269)}
Uses WinCrt;
Const max = 30;
Var atual, anterior, proximo : LongInt;
    n : Byte;
Begin
   Writeln('Seqüência de Fibonacci:');
   Write('{');
   atual := 1;
   anterior := 0;
   For n := 1 to max - 1 do
      Begin
         proximo := atual + anterior;
         anterior := atual;
         atual := proximo;
         Write(atual, ', ');
      End;
   atual := atual + anterior;
   Write(atual, '}');
End.
```

Observe a endentação dos comandos dentro de cada bloco delimitado pelos termos BEGIN e END. Esse alinhamento de grupos de comandos não é considerado pelo compilador Pascal, no entanto, facilita extremamente a compreensão dos níveis de hierarquia dos comandos e a consecutiva assimilação da lógica do programa.

Tipos de dados predefinidos da linguagem Pascal:

Para cada tipo de dado, o Pascal é capaz de reservar um determinado número de bytes na memória. Para que ele possa reter corretamente uma determinada informação é necessário que ela seja adequada a um desses tipos. São eles:

Tipos de dados números inteiros:

Integer (Número inteiro de 2 bytes entre –32.768 e +32.767)
ShortInt (Número inteiro de 1 byte entre –128 e +127)
LongInt (Número inteiro de 4 bytes entre –2.147.483.648 e +2.147.483.647)
Byte (Número inteiro positivo de 1 byte entre 0 e 255)
Word (Número inteiro positivo de 2 bytes entre 0 e 65.535)

Tipos de dados números reais:

Real (Número real de 6 bytes variando de 2,9 E-39 a 1,7 E+38 incluindo o zero, podendo ser negativo ou positivo com 11 a 12 dígitos significativos)
Single (Número real de 4 bytes variando de 1,5 E-45 a 3,4 E+38 incluindo o zero, podendo ser negativo ou positivo com 7 a 8 dígitos significativos)
Double (Número real de 8 bytes variando de 5,0 E-324 a 1,7 E+308 incluindo o zero, podendo ser negativo ou positivo com 15 a 16 dígitos significativos)
Extended (Número real de 10 bytes variando de 3,4 E-4932 a 1,1 E+4932 incluindo o zero, podendo ser negativo ou positivo com 19 a 20 dígitos significativos)

Tipos de dados alfanuméricos:

Char (Caracter ASCII de 1 byte)
String[n] (Cadeia de caracteres ASCII de 2 a 256 bytes, sendo **n** um valor numérico determinando o comprimento da cadeia. Na omissão do valor **n**, o compilador do Pascal assume o comprimento máximo)

Tipos de dados lógicos:

Boolean (Valor lógico de 1 byte que representa apenas: TRUE ou FALSE)

Área de Declarações:

Localizada após o cabeçalho do programa e antes do bloco principal de comandos, a área de declarações é a parte do programa destinada à locação de memória para os dados que serão utilizados no decorrer do programa. O montante da memória locada está relacionado ao tipo de dado a ser utilizado. Podem ser declaradas aqui regiões de memória que podem ter seu conteúdo alterado (as Variáveis), regiões de memória onde a alteração dos dados não é permitida (as Constantes), outros tipos além dos predefinidos da linguagem e também rotinas de programas denominadas Subprogramas (as Procedures e Functions).

Declaração de Bibliotecas de Rotinas:

As Bibliotecas de Rotinas, chamadas no Pascal de UNITs (ou Unidades), são arquivos contendo um conjunto de pequenas rotinas, denominadas Procedures e Functions, reutilizáveis em diversos outros programas, podendo ser predefinidas da linguagem ou definidas pelo usuário. Essas UNITs são compiladas separadamente e inseridas no programa principal através da linkedição. Um programador pode simplesmente associar as Units predefinidas que necessitar ou ter sua própria biblioteca de rotinas criando uma Unit e associando-a a cada novo programa que fizer. Desta forma, não é necessário reescrever rotinas e sim, chamá-las através de seu identificador. Para associar uma UNIT predefinida ou definida pelo usuário a um programa, utiliza-se o comando USES na área de declarações.

Sintaxe do comando Uses:

```
Uses
   <lista-de-units>;
```

Veja as principais Units predefinidas do Pascal e os identificadores de suas rotinas predefinidas:

CRT ou **WINCRT** (Unit com rotinas de tratamento de vídeo e som):

ClrEol	InsLine	Sound	WhereY
ClrScr	KeyPressed	TextBackGround	Window
DelLine	LowVideo	TextColor	
GotoXY	NormVideo	TextMode	
HighVideo	NoSound	WhereX	

DOS ou **WINDOS** (Unit com rotinas que envolvem acesso ao Sistema Operacional):

DiskFree	FindNext	GetTime	SetFTime
DiskSize	FSearch	GetVerify	SetIntVec
DosExitCode	Fslip	Intr	SetTime
DosVersion	GetCBreak	Keep	SetVerify
EnvCount	GetDate	MsDos	SwapVectors
EnvStr	GetEnv	PackTime	UnpackTime
Exec	GetFatTr	SetCBreak	
FExpand	GetFTime	SetDate	
FindFirst	GetIntVec	SetFatTr	

SYSTEM (Unit default do compilador Pascal, sem necessidade de declaração):

Abs	Exp	Move	Seek
Addr	FilePos	New	SeekEof
Append	FileSize	Odd	SeekEolm
ArcTan	FillChar	Ofs	Seg
Assign	Flush	Ord	SetTextBuf
BlockRead	Frac	ParamCount	Sin
BlockWrite	FreeMem	ParamStr	SizeOf
Chr	GetDir	Pi	SPtr
Close	GetMem	Pos	Sqr
Concat	Halt	Pred	Sqrt
Copy	Hi	Ptr	SSeg
Cos	Inc	Random	Str
CSeg	Insert	Randomize	Succ
Dec	Int	Read	Swap
Delete	IOResult	Readln	Trunc
Dispose	Length	Rename	Truncate
DSeg	Ln	Reset	UpCase
Eof	Lo	Rewrite	Val
Eoln	MaxAvail	Round	Write
Erase	MemAvail	RunError	Writeln
Exit	MkDir	ScrollTo	

Exemplo:

```
Uses Crt, Dos, MinhaUnit;
```

Declaração de Constantes:

Constantes são entidades na memória RAM cujos seus conteúdos não se alteram durante a execução do programa e devem ser declaradas pelo comando CONST na área de declarações juntamente com seu valor constante, podendo ou não ser tipadas.

Sintaxe do comando Const:

```
Const
   <identificador1> [: <tipo>] = <valor>;
   <identificadorN> [: <tipo>] = <valor>;
```

Exemplo:

```
Const
   num_pi = 3.1415;
   ano : Word = 1969;
```

Declaração de Tipos Estruturados:

Além dos tipos predefinidos da linguagem Pascal, existe a possibilidade de se definir outros tipos com estrutura própria, chamados de Tipos Estruturados, a partir do comando TYPE na área de declarações, utilizando alguns dos tipos estruturados predefinidos no Pascal como: Array, Record, File, Set e Pointer, ou seja, Matrizes, Registros, Arquivos, Conjuntos e Ponteiros.

Sintaxe do comando Type:

```
Type
   <identificador1> = <estrutra-do-tipo1>;
   <identificadorN> = <estrutra-do-tipoN>;
```

Exemplo:

```
Type
   indices = 1..6;
   Sala    = Array[indices] of String;
   Aluno   = Record
                nome : String[30];
                nota : Real;
             End;
   Turma   = File of Aluno;
   Vogais  = (a, e, i, o, u);
   Vogal   = Set of Vogais;
Var
   predio  : Sala;
   registro: Aluno;
   classe  : Turma;
   tecla   : Vogal;
```

Declaração de Variáveis:

Variáveis são entidades que assumem valores temporários na memória RAM, distinguidas entre elas por identificadores declarados e associados a um tipo de dado predefinido ou definido pelo usuário, através do comando VAR na área de declarações:

Sintaxe do comando Var:

```
Var
   <lista-de-identificadores1> : <tipo1>;
   <lista-de-identificadoresN> : <tipoN>;
```

Exemplo: Observe que cada item de uma lista de identificadores de mesmo tipo, deve ser separado por vírgula (,).

```
Var
   nota, media     : Real;
   i, x1, x2       : Integer;
   codigo, idade   : Byte;
   flag, Sim       : Boolean;
   letra1, letra2  : Char;
   nome            : String[30];
   e_mail          : String;
```

Expressões Pascal:

Uma Expressão é similar a uma equação matemática, na qual um conjunto de operandos, que podem ser variáveis e/ou constantes, interagem com operadores resultando num único valor. Do ponto de vista do Pascal, uma expressão pode resultar um dado de qualquer tipo predefinido; portanto, se uma expressão resulta um valor numérico, é denominada Expressão Aritmética; se resulta um valor lógico é denominada Expressão Lógica e, se manipula ou resulta uma string ou caracter é denominada Expressão Alfanumérica.

Exemplo de uma fórmula matemática e sua equivalente Expressão Aritmética Pascal, onde `m`, `x` e `y` são variáveis numéricas; `2`, `3` e `5` são constantes e; `*`, `+` e `/` são operadores:

$$m = \frac{2x + 3y}{5} \quad \rightarrow \quad \texttt{m := (2 * x + 3 * y) / 5;}$$

Veja o exemplo de uma Expressão Lógica Pascal, onde `digito_certo` é uma variável lógica, `n` uma variável numérica e; os símbolos `>=`, `<` e `and` são operadores:

```
digito_certo := (n >= 0) and (n < 10);
```

Agora veja o exemplo de uma Expressão Alfanumérica Pascal, onde `nomecompleto`, `nome`, `prenome` e `sobrenome`, são variáveis declaradas como String; o espaço em branco `' '` é uma constante e; `+` é um operador.

```
nomecompleto := nome + ' ' + prenome + ' ' + sobrenome;
```

Neste caso, essa operação é chamada de Concatenação, ou seja, a junção de várias strings em uma só; a partir de uma Expressão Alfanumérica. Dizemos que existe, neste caso, uma sobrecarga do operador, pois a partir do tipo do operando Numérico ou Alfanumérico, o Pascal decide se efetua uma adição ou uma concatenação, como nos seguintes exemplos, supondo n1, n2 e n; variáveis do tipo Integer; s1, s2 e s; variáveis do tipo String:

```
n1 := 5;
n2 := 7;
s1 := 'SÃO';
s2 := 'PAULO';
n := n1 + n2;        {n = 12}
s := s1 + s2;        {s = 'SÃOPAULO'}
```

Operador de Atribuição:

O operador de atribuição altera dentro da memória RAM, o conteúdo de uma variável, atribuindo a partir daquela linha de programa o seu valor. Este é um comando simples compilado pelo Pascal através da junção de dois símbolos especiais (: e =). O identificador (do lado esquerdo do operador) deve sempre ser uma variável, considerada nesse caso como variável receptora; e o valor a ser atribuído (do lado direito do operador) pode ser uma variável, uma constante ou uma expressão de qualquer tipo. Obviamente a variável receptora e o valor atribuído devem ser do mesmo tipo, exceto no caso em que a variável é do tipo real e o

valor atribuído é do tipo inteiro, quando nesse caso o valor inteiro da expressão é automaticamente transformado em real.

Sintaxe de uma atribuição Pascal: (:=)

```
<variável> := <constante>|<variável>|<expressão>;
```

Exemplo:

```
Var
   x, y, z, soma : Integer;
   media : Real;
   cor : String[10];
   dentro, fora : Boolean;
Begin
   x := 5;
   y := 2 * x + 1;
   soma := x + y;
   z := soma;
   media := (x + y + z) / 3;
   cor := 'verde';
   dentro := cor = 'vermelho';
   fora := (media < 0) or (media >= 10);
   z := y / x; {Atribuição inválida pois a divisão resulta um dado Real e z é Inteiro}
   cor := fora; {Atribuição inválida pois a cor é String e fora é Boolean}
End.
```

Operadores de Aritméticos:

A tabela seguinte mostra os Operadores Aritméticos fundamentais da linguagem Pascal com suas prioridades de execução, quando agrupados em uma expressão aritmética. Seus operandos devem ser obrigatoriamente numéricos.

Operador	Operação	Prioridade
*	Multiplicação	1ª
/	Divisão real	1ª
div	Divisão inteira	1ª
mod	Resto da divisão inteira	1ª
+	Adição	2ª
-	Subtração	2ª

Exemplos: (para a e b sendo variáveis numéricas inteiras ou reais)

`r1 := a / b;` => Resulta o valor real do quociente de $a \div b$.
`r2 := a div b;` => Resulta o valor inteiro do quociente de $a \div b$.
`r3 := a mod b;` => Resulta o valor inteiro do resto da divisão $a \div b$.
`r3 := 2*a+b;` => Resulta o valor inteiro ou real da equação $2.a+b$.
`r4 := Sqr(a)+b;` => Resulta o valor inteiro ou real da equação a^2+b. Onde Sqr é uma função predefinida do Pascal que retorna o quadrado de um número.
`r5 := a+Sqrt(b);` => Resulta o valor real da equação $a+\sqrt{b}$. Onde Sqrt é uma função predefinida do Pascal que retorna a raiz-quadrada de um número.

Atenção: Quando em uma única expressão existir mais de um operador de mesma prioridade, o Pascal executa-os em pares de operandos da esquerda para direita, e pode-se quebrar a prioridade natural das operações com a utilização de parênteses.

Exemplos: (Para a = 7, b = 2, c = 4; neste caso n1, n2 e n8 devem ser variáveis do tipo Real)

```
n1 := a*b/c*b;    {n1 = 7.0}
n2 := a+b/c+b;    {n2 = 9.5}
n3 := a+b*c;      {n3 = 15}
n4 := (a+b)*c;    {n4 = 36}
```

```
n5 := (3*c+b) div c;    {n5 = 3}
n6 := (3*c+b) mod c;    {n6 = 2}
n7 := Sqr(b+c)-a;       {n7 = 29}
n8 := c-Sqrt(a+b);      {n8 = 1.0}
```

Observe: Note que o Pascal não oferece operadores de potenciação do tipo x^n que não seja x^2, nem de radiciação que não seja raiz-quadrada (√) através de funções predefinidas. No entanto utiliza-se uma alternativa matemática na qual nos leva ao mesmo resultado. Veja a seguir:

$p = b^n$ → `p := Exp(n * Ln(b));`

$r = \sqrt[n]{b} = b^{1/n}$ → `r := Exp(1/n * Ln(b));`

Onde Exp e Ln são funções predefinidas do Pascal que retornam respectivamente o Exponencial e o Logaritmo Natural de um número. No caso da potência n, para calcular o Exponencial é necessário primeiro calcular o produto entre o expoente n e o Logaritmo Natural da base.

Exemplos:

$p = 2^8$ → `p := Exp(8 * Ln(2));` → *p = 256*

$r = \sqrt[3]{125}$ → `r := Exp(1/3 * Ln(125));` → *r = 5*

Operadores de Relacionais:

Uma relação é uma comparação realizada entre dois operandos de mesmo tipo que resulta sempre em um valor lógico, predefinido em Pascal pelos termos TRUE (verdadeiro) ou FALSE (falso). Estes operandos são representados por constantes, variáveis ou expressões. A natureza da comparação é indicada pelos Operadores Relacionais representados em Pascal pelos seguintes símbolos, sem prioridades entre eles:

Operador	Operação
=	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
<>	Diferente de
In	Contido em (conjunto)

Exemplos:

```
a <> b                  => o conteúdo da variável a é diferente do conteúdo da variável b?
b >= 3                  => o conteúdo da variável b é maior ou igual ao número 3?
nome = 'Maria'          => o conteúdo da variável nome é igual à string Maria?
2 * x + 5 * y < 0       => o resultado da expressão 2x+5y é menor do que zero?
letra in ['a','e','i','o','u']   => o conteúdo da variável letra está contido no conjunto
                                    de caracteres vogais?
```

Operadores Lógicos:

A Negação (não), a Conjunção (e) e a Disjunção (ou) são consideradas as três operações fundamentais da lógica computacional que o Pascal implementa através dos três operadores: NOT, AND e OR com as seguintes prioridades entre eles:

Operador	Operação	Prioridade
`Not`	Negação (não)	1ª
`And`	Conjunção (e)	2ª
`Or`	Disjunção (ou)	3ª

A partir de uma tabela de entradas e saídas lógicas finitas para cada um desses operadores, denominada Tabela-Verdade, se obtém o valor resultante também lógico, ou seja, verdadeiro (TRUE) ou falso (FALSE), como a seguir:

Tabela-Verdade do Operador NOT:

Entradas	Operação
A	not A
False	True
True	False

Este operador unário atua como uma chave inversora, alterando a entrada falsa para verdadeira e vice-versa.

Tabela-Verdade do Operador AND:

Entradas		Operação
A	B	A and B
False	False	False
False	True	False
True	False	False
True	True	True

Neste operador de duas entradas, observa-se que basta uma entrada falsa para que o resultado seja falso.

Tabela-Verdade do Operador OR:

Entradas		Operação
A	B	A or B
False	False	False
False	True	True
True	False	True
True	True	True

Neste operador também de duas entradas, observa-se que, ao contrário do anterior, basta uma entrada verdadeira para que o resultado seja verdadeiro.

Funções Predefinidas:

Uma Função é parte de um programa, como uma rotina de instruções, que executa uma tarefa e retorna ao programa um determinado valor. Uma função é predefinida quando já vem pré-programada a partir do distribuidor da linguagem, reconhecida pelo compilador e ligada ao seu programa pelo linkeditor. Dessa forma as funções são ferramentas essenciais para o programador.

Em Pascal existem diversos tipos de Funções Predefinidas: **Numéricas, Lógicas e Alfanuméricas**. São Numéricas quando retornam valores numéricos resultantes de cálculos matemáticos úteis para o processamento de expressões aritméticas mais complexas, como uma calculadora científica; Lógicas quando retornam um valor lógico TRUE ou FALSE e Alfanuméricas quando retornam valores úteis para o tratamento de cadeias de caracteres (strings). Para que as Funções Pascal retornem um resultado válido, é necessário enviar um dado de requisito específico para cada Função, que chamamos de Parâmetro. As Funções do Pascal têm sempre maior prioridade de execução sobre quaisquer outros operadores numa expressão.

Veja a tabela de Funções Numéricas e Lógica, onde **Z** é qualquer número inteiro; **R** é qualquer número real e **B** é qualquer valor booleano (lógico):

Função	Retorna	Parâmetro x	Retorno
`Abs(x)`	O valor absoluto de x.	Z ou R	Z ou R
`Arctan(x)`	O arco tangente de x.	R	R
`Cos(x)`	O coseno de x.	R	R

Dec(x,i)	O valor de x decrementado de i.	Z	Z
Exp(x)	O número *e* elevado a um expoente x.	R	R
Frac(x)	A parte fracionária de x.	R	Z
Inc(x,i)	O valor de x incrementado de i.	Z	Z
Ln(x)	O logaritmo natural de x.	R	R
Odd(x)	O valor TRUE se x for ímpar.	Z	B
PI	O valor constante de π (3,141592653).	---	R
Random(x)	Um valor aleatório entre 0 e x.	Z	Z
Round(x)	O valor arredondado de x.	R	Z
Sin(x)	O seno de x.	R	R
Sqr(x)	O quadrado de x.	Z ou R	Z ou R
Sqrt(x)	A raíz quadrada de x.	R	R
Trunc(x)	O valor truncado de x.	R	Z

Exemplos: (Expressões aritméticas com uso de funções e suas equivalentes equações matemáticas)

```
q := Sqr(x);
```
→ $q = x^2$

```
t := Arctan(-Sqrt(x)/x);
```
→ $t = \tan\left(\frac{-\sqrt{x}}{x}\right)$

```
z := Round(3*(Cos(5*Pi/6)+Sin(5*Pi/6)));
```
→ $z \cong 3\left[\cos\left(\frac{5\pi}{6}\right) + \sin\left(\frac{5\pi}{6}\right)\right]$

```
x1 := (-b+Sqrt(Sqr(b)-4*a*c))/(2*a);
```
→ $x1 = \frac{-b+\sqrt{b^2-4.a.c}}{2.a}$

```
y := 1/(d/dt*Ln(x));
```
→ $y = \frac{1}{\frac{d}{dt}\ln(x)}$

Veja a seguir, a tabela de Funções Alfanuméricas, onde **Z**, **P**, **Q** e **N** representam qualquer número inteiro; **S**, **S1**, **S2** e **Sn**, é qualquer string; e **C** é qualquer caracter:

Função	Retorna	Parâmetros	Retorno
Concat(S1,S2,...,Sn)	Uma única string formada pela junção das strings S1, S2 até Sn.	S	S
Copy(S,P,Q)	Uma substring da string S, a partir da posição P, de comprimento Q.	S e Z	S
Length(S)	O comprimento de uma string S.	S	Z
Pos(S1,S2)	A posição em que a string S1 se encontra dentro da string S2.	S	Z
Ord(C)	O código ASCII do caracter C.	C	Z
Chr(N)	O caracter equivalente ao código ASCII de N.	Z	C
Succ(C)	O caracter sucessor do caracter C.	C	C
Pred(C)	O caracter predecessor do caracter C.	C	C
UpCase(C)	O caracter maiúsculo equivalente ao caracter C.	C	C

Procedimentos Predefinidos:

Um Procedimento, semelhante a uma função, é como uma rotina de instruções à parte do programa, que apenas executa uma tarefa sem assumir um valor específico. Um procedimento é predefinido quando já vem pré-programado pelo distribuidor da linguagem. Um procedimento também pode necessitar de parâmetros e pode retornar ao programa com novos valores em seus parâmetros.

A tabela a seguir lista os Procedimentos Predefinidos do Pascal, onde **Z**, **P** e **Q** representam qualquer número inteiro; **R**, qualquer número real; **S**, **S1** e **S2**, é qualquer string:

Procedimento	Descrição
`Insert(S1,S2,P)`	Insere a string S1, na string S2, a partir da posição P.
`Delete(S,P,Q)`	Elimina da string S, a partir da posição P, uma substring de comprimento Q
`Str(R,S)`	Converte um número inteiro ou real N para uma string S do tipo Alfanumérico.
`Val(S,R,Z)`	Converte uma string S para um dado numérico N inteiro ou real de mesmo valor. Caso a string S não possua apenas números, ocorrerá um erro de conversão onde Z será diferente de 0.

Exemplos: (Para S1 = 'computador', S2 = 'cabeçalho' e L1 = 'a')

```
Delete(S2,7,3);
S3 := 'Estou '+Copy(S1,1,3)+' uma '+Copy(S1,8,3)+' de '+S2+'...';
t  := Length(S1);
p  := Pos('alho';S2);
c  := Ord(L1);
L2 := UpCase(L1);
L3 := Chr(77);
Insert('micro',S1,1);
```

Após executar estas linhas de programa, as variáveis conterão os seguintes valores, lembrando que *S1*, *S2* e *S3* devem ser do tipo String; *L1*, *L2* e *L3* do tipo Char; *t*, *p* e *c* do tipo Byte:

S2 = 'cabeça'
S3 = 'Estou com uma dor de cabeça...'
t = 10
p = 6
c = 97
L2 = 'A'
L3 = 'M'
S1 = 'microcomputador'

Comandos de Entrada e Saída:

Os comandos de maior importância dentro de um programa são os de entrada e saída de dados, que possibilitam a interface entre o seu programa e o usuário. É a maneira pela qual um usuário insere os dados através do teclado ao seu programa e o programa exibe na tela as informações resultantes do processamento.

O Comando READ:

Os termos READ e READLN são os comandos de entrada de dados usados em Pascal. Veja na sintaxe abaixo que lista-de-identificadores são os nomes das variáveis receptoras de qualquer tipo predefinido, cujos valores serão lidos dos dispositivos de entrada (teclado ou drives de disco). Quando mais do que um, os identificadores contidos na lista, devem estar separados por vírgula. A diferença entre estes os dois comandos de entrada é que o comando READ lê registros de arquivos em disco e o comando READLN lê informações enviadas do teclado. Os valores de entrada, ao serem digitados no teclado, deverão estar separados por espaços ou seguidos da tecla ENTER para que o READLN capture corretamente os dados e armazene-os nas respectivas variáveis.

Sintaxe do comando READ:

```
Read[ln](<lista-de-identicadores>);
```

O Comando WRITE:

Os termos WRITE e WRITELN são os comandos de saída de dados usados em Pascal. A lista-de-identificadores representa os nomes das variáveis de qualquer tipo predefinido, qualquer constante ou qualquer expressão. Os identificadores contidos na lista, devem estar separados por vírgula. Constantes e expressões de tipos distintos podem aparecer num mesmo comando de saída. A diferença entre estes dois comandos é que o comando WRITE imprime os valores sempre numa mesma linha de vídeo até o seu término. Ao contrário, o comando WRITELN imprime os valores e incrementa uma linha de vídeo, proporcionando a cada impressão, um CR e LF (Retorno do Cursor e Avanço de Linha). Esses comandos READ e WRITE são considerados Comandos Simples.

Sintaxe do comando WRITE:

```
Write[ln](<lista-de-identicadores>|<constantes>|<expressões>);
```

Exemplos:

```
Read(arq,reg);                  {Lê do arquivo arq, o registro reg}
Write('Entre com o nome: ');    {Exibe na tela o texto entre apóstrofos}
Readln(nome);                   {Lê do teclado um conteúdo e armazena-o na
                                 variável nome}
Writeln(nome);                  {Exibe na tela o conteúdo da variável nome}
Writeln('Código: ', cod);       {Exibe na tela um texto, seguido do conteúdo da
                                 variável cod}
Writeln(2*x+y);                 {Exibe o resultado da expressão aritmética}
```

Formatação da Saída de Dados:

A formatação é definida somente no comando Write através de dois pontos (:) após sua expressão.

```
Write(E:P1:P2);    ou    Writeln(E:P1:P2);
```

Onde o parâmetro E representa o valor a ser escrito na tela, demonstrado na sintaxe do comando Write como sendo uma constante, uma variável ou uma expressão de qualquer tipo de dado predefinido. O parâmetro P1 representa um valor inteiro positivo e indica o número mínimo de caracteres a ser escrito na tela. Se P1 for insuficiente para representar o valor escrito, então será locado mais espaço na tela. Se P1 for excessivo, o espaço excedente será preenchido com brancos. O parâmetro P2 só aplica-se somente se o parâmetro E for do tipo REAL. P2 é um valor inteiro positivo e especifica o número de casas decimais, ou

seja, a quantidade de dígitos que devem ser escritos na tela após o ponto decimal. Se E2 for omitido, então o valor real será escrito na forma de ponto flutuante ou exponencial.

Exemplo:

```
Program FormatandoDados;
Uses Crt;
Var
   n    : Byte;
   a, b : Real;
   s    : Char;
Begin
   n := 10;
   a := 4.1;
   b := 7.8;
   s := '*';
   Writeln('------------------------------------------------------------');
   Writeln(n, ' + ', a, ' + ', b, ' = ', n+a+b, s);
   Writeln(n:3, ' + ', a:3, ' + ', b:3, ' = ', n+a+b:3, s:1);
   Writeln(n:5:2, ' + ', a:5:2, ' + ', b:5:2, ' = ', n+a+b:5:2, s:2);
   Writeln(n:1:4, ' + ', a:1:4, ' + ', b:1:4, ' = ', n+a+b:1:4, s:5);
   Writeln('------------------------------------------------------------');
End.
```

Saída em vídeo:

```
------------------------------------------------------------
10 + 4.1000000000E+00 + 7.8000000000E+00 = 2.1900000000E+01*
10 + 4.1E+00 + 7.8E+00 = 2.2E+01*
10 + 4.10 + 7.80 = 21.90 *
10 + 4.1000 + 7.8000 = 11.9000 *
------------------------------------------------------------
```

Referências Bibliográficas:

REENG/FEC, "Apostila de Introdução ao Pascal"; UNICAMP, Campinas-SP, 2000.
RINALDI, Roberto; "Turbo Pascal 7.0 Comandos e Funções"; Editora Érica, São Paulo-SP, 1993.
ASCENCIO, A.F.G.. Lógica de programação com Pascal. São Paulo: Makron Books, 1999.
MANZANO, J.A.N.G., YAMATUMI, W.Y., Free Pascal: Programação de Computadores, ed.1, São Paulo: Érica, 2007.
SCHIMITZ, Eder e TELES, Antônio; "Pascal e Técnicas de Programação"; Editora LTC, Rio de Janeiro-RJ, 1999.
SALIBA, Walter; "Técnicas de Programação uma Abordagem Estruturada"; Ed. Makron Books, São Paulo-SP, 1992.
SWAN, Tom; "Mastering Turbo Pascal 5.0"; Editora Hayden Books, SanDiego – California, USA, 1990.