Algumas notas sobre PASCAL

PARTE II - Procedimentos e Funções

Anjolina Grisi de Oliveira

1 Introdução

Os procedimentos e funções são úteis na programação uma vez que permitem que um programa possa ser dividido em *subprogramas*. Quando um programa é grande, fica mais fácil depurá-lo se ele está dividido em subprogramas. A linguagem PASCAL possui dois tipos de subprogramas: procedimentos (*procedure*) e funções (*function*).

2 Procedimentos

Em PASCAL os procedimentos são definidos após a declaração das variáveis do programa principal. O procedimento é ativado quando é chamado pelo programa principal. Eles podem ou não ter parâmetros. A sua forma mais simples é definida como a seguir, sem a inclusão de parâmetros.

2.1 Procedimentos sem parâmetros

Os procedimentos são definidos da seguinte forma:

```
Procedure <nome do procedimento>;
  declaracao de var.
Begin
  comandos
End;
```

Exemplo 1 *O seguinte programa recebe dois números e, de acordo com um menu de opções faz diferentes operações com esse número.*

```
Program Ex1;
uses crt;
var
  num1,num2 : integer;
  op:char;

procedure MostraMenu;

begin
    clrscr;
    writeln('Digite a opcao desejada');
    writeln(' (1) Soma e produto');
```

```
    writeln(' (2) Produto');
    writeln(' (3) Resto da div. por x');
end;

procedure SomaProduto;

begin
   writeln('Soma = ', num1+num2);
   writeln('Produto = ', num1*num2);
end;

procedure Resto;
var x: integer;

begin
   writeln('Entre com o valor de x');
   readln(x);
   if x<=num1 then
     writeln('resto de ',num1,' por ', x, ' = ',num1 MOD x)
   else
     writeln(x, 'maior que ', num1);
   if x<=num2 then
     writeln('resto de ',num2, ' por ', x, ' = ', num2 MOD x)
   else
     writeln(x, 'maior que ', num2);
end;

Begin
  writeln('Entre com dois numeros inteiros');
  readln(num1,num2);
  MostraMenu;
  op:=readkey;
  case op of
    '1': SomaProduto;
    '2': Resto;
    else
       writeln('opcao invalida');
  end;
  readln;
End.
```

Algumas observações:

- Os subprogramas podem incluir definições locais, tanto de variáveis como de constantes ou tipos. No exemplo 1, o procedimento *Resto* possui uma **variável local** (x). As definições locais só existem durante a execução do subprograma. Por esse motivo, elas somente são acessíveis pelo subprograma que as contem.

- As variáveis do programa principal são chamadas de **variáveis globais** porque são acessíveis por todos os subprogramas, além do programa principal. No exemplo 1, as variáveis *num1* e *num2* são globais.
- O **escopo** de um identificador define a porção do programa em que ele atua. Portanto, o escopo das variáveis locais é o subprograma onde elas foram definidas e os subprogramas contidos no subprograma onde elas foram definidas. O escopo de uma variável global é todo o programa. A figura 1 ([CooCla85]) ilustra esse conceito.
  - no programa *A*, os seus identificadores possuem como escopo os blocos: *A, B, C, D, E*;
  - no procedimento *B*, escopo: *B,D*;
  - no procedimento *C*, escopo: *C, E, F*;
  - na função *D*, escopo: *D*;
  - no procedimento *E*, escopo: *E*;
  - no procedimento *F*, escopo: *F*;

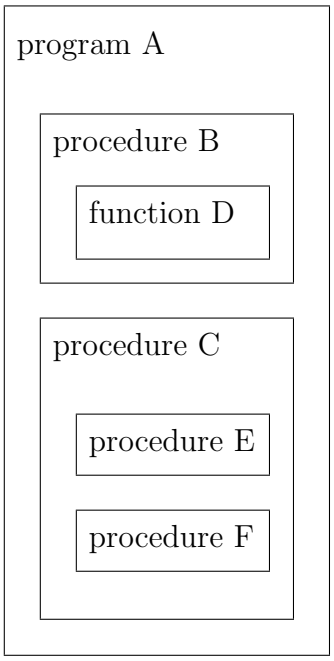

Figure 1: Escopo de identificadores

2.2 Procedimentos com parâmetros

Os parâmetros ou argumentos são usados para comunicação entre o programa principal e os subprogramas. Na realidade, nós já usamos essa comunicação com os procedimentos pré-definidos da linguagem, como por exemplo o *readln* e o *writeln*. Quando temos um trecho de programa como a seguir:

```
readln(x);
writeln(x);
```

Os procedimentos *readln* e *writeln* precisam de parâmetros para executar sua tarefa. No caso de procedimentos definidos pelo usuário, a declaração da *procedure* fica da seguinte forma:

```
procedure <identificador> (lista de argumentos);
 declaracao dos identificadores locais
begin
  comandos;
end;
```

A passagem de parâmetros entre o programa principal e o procedimento se dá de duas formas:

1. *passagem por valor*: nesse caso, o programa principal passa um valor para o procedimento. Esse valor pode ser passado explicitamente, ou então passa-se o valor de uma variável. Por exemplo, no programa seguinte a segunda chamada do procedimento *writeln* passa o valor de x como argumento.

   ```
   writeln('A');
   x:='A';
   writeln(x);
   ```

2. *passagem por referência (ou por variável)*: nesse caso, na chamada do subprograma o programa principal passa o *endereço* da variável. A variável correspondente no subprograma é um nome alternativo da variável da chamada do subprograma. Mudanças nessa variável afetam a variável do programa principal.

Definição dos parâmetros

Quando a passagem de parâmetros se dá por variável, o grupo de identificadores possui a palavra **var** na sua frente. Caso contrário, a passagem de parâmetros é por valor.

Exemplo 2 *No procedimento $P1$ os identificadores x e y são parâmetros cuja passagem se dá por variável, ao passo que z é um parâmetro com a passagem por valor. No caso do procedimento $P2$, o argumento w1 a passagem é por valor e w2 é por referência.*

```
procedure P1(var x,y : integer; z:integer);

procedure P2(w1:char; var w2: char);
```

Exemplo 3 *Aqui mostramos o exemplo 1 modificado. O exemplo mostrado aqui é mais recomendado em programação, pois não devemos alterar o valor de uma variável global dentro dos subprogramas. Esse tipo de ação pode gerar erros difíceis de serem depurados.*

```
Program Ex1;
uses crt;
var
  num1,num2 : integer;

procedure Menu(var op:char);

begin
    clrscr;
    writeln('Digite a opcao desejada');
    writeln(' (1) Soma e produto');
    writeln(' (2) Produto');
    writeln(' (3) Resto da div. por x');
    op:=readkey;
end;

procedure SomaProduto(x1,x2:integer);

begin
   writeln('Soma = ', x1+x2);
   writeln('Produto = ', x1*x2);
end;

procedure Resto(n1,n2:integer);
var x: integer;

begin
   writeln('Entre com o valor de x');
   readln(x);
   if x<=num1 then
     writeln('resto de ',n1,' por ', x, ' = ',n1 MOD x)
   else
     writeln(x, 'maior que ', n1);
   if x<=num2 then
     writeln('resto de ',n2, ' por ', x, ' = ', n2 MOD x)
   else
     writeln(x, 'maior que ', n2);
end;

Begin
  writeln('Entre com dois numeros inteiros');
```

```
  readln(num1,num2);
  Menu(op);
  case op of
    '1': SomaProduto(num1,num2);
    '2': Resto(num1,num2);
    else
       writeln('opcao invalida');
  end;
  readln;
End.
```

3 Funções

Funções são subprogramas que retornam um único valor para o programa principal. Esse valor é retornado no nome da função. A sua chamada pelo programa principal se dá da mesma forma da chamada de uma função pré-definida da linguagem.

Declarando funções

```
function <nome da funcao> : tipo do dado retornado;
   declaracoes locais
begin
   comandos
end;
```

```
function <nome da funcao> (lista de argumentos) : tipo do dado retornado;
    declaracoes locais
begin
    comandos
end;
```

Exemplo 4 *Podemos ainda modificar o nosso exemplo 1. O procedimento MostraMenu pode ser transformado numa função, uma vez que pode retornar para o programa principal a opção teclada pelo usuário.*

```
function RecebeOp :char;

begin
    clrscr;
    writeln('Digite a opcao desejada');
    writeln(' (1) Soma e produto');
    writeln(' (2) Produto');
```

```
    writeln(' (3) Resto da div. por x');
    RecebeOp:=readkey;
end;

Begin
  writeln('Entre com dois numeros inteiros');
  readln(num1,num2);
  case RecebeOp of
    '1': SomaProduto(num1,num2);
    '2': Resto(num1,num2);
    else
       writeln('opcao invalida');
  end;
  readln;
End.
```

Alternativamente, poderiamos colocar o comando op := RecebeOp*, nesse caso o* case *fica como antes.*

Exemplo 5

```
Program Calc;

var num,dig: integer;

function digito(n:integer) : integer;
begin
   if n > 1000 then
      digito:= n mod 9
   else
      digito:= (n mod 5) + (n mod 3);
end;

begin
  readln(num,dig);
  if dig <>  digito(num) then
     writeln('digito invalido');
  readln;
end.
```

4 Arranjos como parâmetros

O tipo dos identificadores na declaração dos subprogramas deve ser básico: (a) inteiro, booleano, caracter ou real; (b) definido pelo programador. Os seguintes cabeçalhos de procedimentos [TenAug86] são inválidos:

```
procedure proc(i:1..10);
procedure proc2(a:array[1..10] of integer);
```

Podemos apresentar as seguintes declarações para passarmos " arrays" como parâmetros:

```
type indice = 1..10;
     tipoa = array[indice] of integer;

procedure proc(i:indice);
procedure proc2(a:tipoa);
```

References

[Asc99] Ana Fernanda Gomes Ascencio. Lógica de programação com Pascal. Makron Books Ltda, São Paulo, Brasil. 1999.

[CooCla85] Doug Cooper and Michael Clancy. Oh! Pascal! W.W. Norton and Company, New York and London. Second Edition, 1985.

[TenAug86] Aaron M. Tenenbaum and Moshe J. Augenstein. Data Structures using Pascal. Segunda edição, Prentice-Hall International Editions, EUA, 1986.

[Ziv99] Nivio Ziviani. Projeto de Algoritmos com implementações em Pascal e C. Pioneira Informática, São Paulo, Brasil. Quarta edição, 1999.