Programação Estruturada Estruturas de controle

Paradigmas de Linguagens de Programação

Profa. Heloisa – 1º. Sem. 2019

Estruturas de Controle

Sebesta, R.W. *Concepts of Programming Languages*. 9a.edição/ Addison-Weley, 2009. Capítulo 9.

- Estruturas que determinam o fluxo de execução (que comando é executado depois do outro).

- Estruturas de controle podem ser:
  - em nível de comando ou
  - em nível de unidades.

Estruturas de Controle em Nível de Unidades

- Mecanismos que permitem fazer chamadas de unidades.
- **Chamadas de unidades explícitas**: funções, procedimentos.
- **Chamadas de unidades implícitas**: tratadores de exceção, corrotinas, unidades concorrentes.

Unidades Subordinadas Chamadas Explicitamente

Inclui os subprogramas: subrotinas, funções, procedimentos

- Função – abstrai uma expressão a ser avaliada
  - Funções retornam valores.
  - Exemplo: cálculo de fatorial de um dado número:
  - **fatorial(n)** deve retornar n!
  - Efeito secundário: quando os parâmetros da função retornam valores.
- Procedimento – abstrai um comando a ser executado
  - Modifica variáveis
  - Exemplo: ordenação de um vetor de números.
  - **ordena(v)** deve ordena o vetor v.

PASCAL:

```
function <nome da função> ( <lista de argumentos> ) : <tipo>;
< corpo da função>
end
procedure <nome> (<parâmetros>);
<corpo>
End
```

C:

```
<tipo do resultado> <nome> ( <declaração de parâmetros
   formais. )
{
<lista de declarações>
<lista de comandos>
}
```


Parâmetros

- Permitem a aplicação de subprogramas a dados diferentes.
- Melhora o reuso de código
- Sem parâmetros, a utilidade dos subprogramas se restringiria a segmentação do código.

- Parâmetro formal – identificadores usados no cabeçalho do subprograma (definição do subprograma)

- Parâmetro real – identificadores, expressões ou valores usados na chamada do subprograma.

- Argumento – usado como sinônimo de parâmetro real ou para referir o valor passado do parâmetro real para o formal.

Correspondência entre Parâmetros Formais e Reais

- A maioria das linguagens usa um **critério posicional** para amarração de argumentos e parâmetros:
- Definição do procedimento – p_i são parâmetros:

```
procedure S( p1; p2; ....; pn);
..
end;
```

- Chamada do procedimento – a_i são argumentos:

```
S( a1; a2; .....; an);
```

p_i corresponde a a_i para i=1,...n.

Convenções para passagem de parâmetros

- Passagem por Referência
  - (Call by Reference) ou
  - Compartilhamento (Sharing)
- Unidade chamadora passa para a unidade chamada o **endereço** do argumento.
- A variável usada como argumento é **compartilhada** e pode ser modificada.

Convenções para passagem de parâmetros

- Passagem por Cópia – Os parâmetros se comportam como variáveis locais. Pode ter 3 tipos:
  - Passagem de Valor – argumentos são usados para inicializar parâmetros, que funcionam como variáveis locais. Valores não podem retornar por esses parâmetros.
  - Passagem de Resultado – parâmetros não recebem valores na chamada, funcionam como variáveis locais mas retornam valores na saída
  - Passagem de Valor-Resultado – engloba os dois anteriores

Convenções para passagem de parâmetros

- Passagem de nome – a amarração do parâmetro à posição não é feita na hora da chamada, mas a cada vez que ele é usado na unidade chamada.
- Portanto, atribuições ao mesmo parâmetro podem ser feitas a posições diferentes a cada ocorrência.

Exemplos:
Passagem por referência X passagem por valor

PASCAL:

```
var a, b: integer;
procedure P (x: integer; var y: integer);
   begin x := x + 1 ; y := y + 1 ;
   writeln (x , y)
   end;

begin a := 0 ; b := 0 ;
         P (a,b);
         writeln (a,b)
end.
```

x é passado por valor
y é passado por referência

Saída: 1 1
0 1

C

```
void troca1(int x, int y)
     {int z;
     z =x; x = y; y = z;}
.......
a = 10; b = 5;
troca1(a,b);
```

x e y são passados por valor
os valores de a e b não **são trocados**

```
void troca(int *px, int *py)
   { int z;
   z = *px; *px = *py;
   *py = z;}
.......
a = 10; b = 5;
troca(&a,&b);
```

x e y são passados por referência
os valores de a e b são trocados

Exemplos – Passagem por valor-resultado

- Semântica idêntica a de passagem por referência, exceto quanto os parâmetros formais e reais são sinônimos.
- Função troca3 em sintaxe similar a ADA:

```
procedure troca3 (a : in out Integer, b : in out Integer) is
// a e b são passados por valor-resultado
      temp : Integer;
      begin
      temp := a;
      a := b;
      b := temp;
      end troca3;
```

- Chamada: troca3(c, d);

Exemplos – Passagem por valor-resultado

- Chamada: troca3(c, d);
- As ações da chamada de troca 3 assumindo passagem por valor-resultado são:
  - addr_c = &c %salva o endereço do parâmetro real c
  - addr_d = &d %salva o endereço do parâmetro real d
  - a = *addr_c %copia o valor de c em a
  - b = *addr_d %copia o valor de d em b
  - (executa as instruções de troca3)
  - *addr_c = a %copia o valor de a na posição ocupada por c
  - *addr_d = b %copia o valor de b na posição ocupada por d
- Os valores de c e d são de fato trocados.

Passagem por valor-resultado

Deve ter o mesmo efeito da passagem por referência. Problemas que devem ser evitados:

```
program
.....
    procedure exemplo(x, y);
    begin  i := y end;
...........
begin
        i := 2; A[i] := 99;
        exemplo( i, A[i]);
end.
```

- i pode ser alterado diretamente pela atribuição, ou indiretamente pela cópia do resultado
- Na chamada de *exemplo (i, A[i])*:
  - endereços de i e A[i] são armazenados
  - valores de i e A[i] são copiados em x e y
  - valor de i é modificado em i := y
  - valores de x e y são copiados de volta em i e A[i], logo **valor antigo de i é restaurado**

Escolha de Mecanismos:

- Parâmetros que devem retornar valores, devem ser passados por referência
- Parâmetros que não retornam valores podem ser passados por valor, por segurança;
- Eficiência da implementação:
  - passagem por referência é cara em termos de processamento pois faz acesso indireto;
  - passagem por cópia pode custar em termos de memória se os objetos forem grandes.

Passagem por nome

- Parâmetro real substitui textualmente o parâmetro formal correspondente em todas as suas ocorrências.
- O parâmetro formal é amarrado ao **método de acesso** no momento da chamada mas a amarração a valor ou endereço é feita só na hora que o parâmetro é atribuído ou referenciado.
- Adia o cálculo de um argumento até que ele seja realmente utilizado no procedimento.
- Algol introduziu a passagem por nome.
- Objetivo: flexibilidade

Exemplo – passagem de parâmetro por nome

```
procedure BIGSUB;
  integer GLOBAL;
  integer array LIST [1:2];
  procedure SUB (PARAM);
       integer PARAM;
       begin
       PARAM := 3;
       GLOBAL := GLOBAL + 1;
       PARAM := 5
       end;
  begin
  LIST [1] := 2;
  LIST[2] := 2;
  GLOBAL := 1;
  SUB(LIST [GLOBAL])
  end;
```

PARAM é passado por nome

Exemplo – passagem de parâmetro por nome

- No final LIST tem valores 3 e 5, atribuídos em SUB.
- Vantagem: flexibilidade
- Desvantagens:
  - –mais lento
  - –algumas operações simples não podem ser implementadas como trocar os parâmetros

Exemplo – passagem de parâmetro por nome

- Cálculo de $\sum_{k=l}^{u} a_k$

```
real procedure Sum(k, l, u, ak)
    value l, u;
    integer k, l, u;
    real ak;
    comment k and ak are passed by name;
  begin
    real s;
    s := 0;
    for k := l step 1 until u do
      s := s + ak;
    Sum := s
  end;
```

- A variável índice (k) e o termo de soma (ak) são passados por nome.
- Chamada por nome permite que o procedimento mude o valor da variável índice durante a execução do for.
- Faz com que o argumento ak seja recalculado durante cada iteração do loop.
- Tipicamente ak vai depender das mudanças (por efeito secundário) de k.

- Por exemplo, para calcular a soma dos 100 primeiros termos de um array de reais V[]:

Sum(i, 1, 100, V[i]).

- Durante a execução de Sum, o valor de i vai ser incrementado a cada iteração do for.
- Cada avaliação de ak vai usar o valor corrente de I para acessar os elementos do array V[i].

- Uma soma dupla pode ser feita com:
- Sum(i, l, m, Sum(j, l, n, A[i,j]))
- Uma soma de inteiros pode ser calculada por:

Sum(i,1,100,i)

- Soma de quadrados de inteiros:

Sum(i,1,100,i*i)