Engenharia reversa de um compilador

linguagem fonte Formada por: cabeçalho (Bem simples, formado apenas pela palavra “algoritmo” seguida do nome do mesmo); Variáveis (As variáveis do algoritmo devem ser especificadas logo após o cabeçalho do algoritmo. Haverá apenas o tipo inteiro, que por ser único não precisa ser especificado) e Comandos, como se especifica a seguir.

LEIA

O comando leia ao ser compilado interromperá o fluxo do programa, gerará uma mensagem padrão ”LEITURA DA VARIÁVEL nnn”, onde nnn é o nome da variável da lista, aceitará o valor a ela fornecido via teclado, assinalará este valor a ela e continuará o processamento.

IMPRIMA

Aqui, ao ser traduzido este comando gerará uma impressão do programa, colocando antes dos valores a mensagem opcional e listando as variáveis cujos nomes fazem parte da lista. Não há interrupção no fluxo.

ASSINALAMENTO

O comando de assinalamento tem o formato **receptor <- expressão de tipo compatível com o receptor.** O símbolo “< -” pode ser substituído pelo símbolo ← se este estiver disponível.

SE

Comando condicional. Tem o formato **se condição lista-ações [senão lista-ações2] fim{se}** A condição é avaliada. Se ela for verdadeira, a lista-ações é executada. Se ela for .F. e houver a lista do senão (a lista-ações2) esta é executada. O comando sempre se encerra por um **fim{se}**.

ENQUANTO

Comando condicional. Tem o formato **enquanto condição lista-ações fim{enquanto}** A condição é avaliada. Se ela for verdadeira, a lista-ações é executada. Ao chegar ao final da lista, há um desvio ao começo do laço. Se ela for .F. há um desvio para depois do fim{enquanto}.

REPITA

Comando condicional. Tem o formato **repita lista-ações ate {condição}** A lista é executada. Ao se localizar a cláusula **ate**, a condição é avaliada. Se ela for verdadeira, o fluxo segue normalmente. Se ela for falsa, há um desvio ao início do laço.

PARA

Comando de repetição. Tem o formato **para variável de valor1 ate valor2 [passo valor3] lista-ações fim{para}** Controla laços finitos e previamente conhecidos.

Linguagem objeto

Haverá uma pilha cujo elemento tem 22 caracteres (6 de identificação, 8 de variável do para e 8 de incremento do para. Estes dois últimos campos só são usados no para) Haverá uma coleção de variáveis locais, com nomes V0000, V0001, V0002, etc. Elas são numéricas inteiras e servem tanto para efetuar cálculos, como para receber condições (0 é falso, 1 é verdadeiro), como para estabelecer pontos de chegada de desvios, o que em linguagem de programação se conhece como “label”, desde que apareçam sozinhas no código seguido por “:”.

Resumo dos desvios

Se simples SE simples: a condição é 0001. Se 0001 for falsa, desvie para 0002. Empilhe SE0002. Traduza normalmente. Ao achar fim{se}, desempilhe 0002 e aplique o rótulo no código.

Se composto SE composto: a condição é 0001. Se 0001 for falsa, desvie para 0002. Empilhe SE0002. Traduza normalmente. Ao achar senão, desempilhe 0002 guardando-o em SALVA. Empilhe SE0003 e desvie para ela. Aplique SALVA no código. Traduza normalmente. Ao achar fim{se}, desempilhe 0003 e aplique-o no código

Enquanto ENQUANTO: Empilhar EN0004, Aplicar 0004 no código. Empilhar EN0005. Calcular 0006 com a condição. Se 0006 é falso, desviar para 0005. Traduzir normalmente. Ao achar fim{enquanto}, desempilhar o 0005, guardando-o em SALVA, desempilhar o 0004, desviar para 0004, e aplicar SALVA no código.

Repita REPITA: aplicar 0007 no código e empilhar RE0007. Traduzir. Ao achar “ate”, empilhar RE0008. Colocar a condição em 0009. Se 0009 for verdadeira, desempilhe 0008 e desvie para 0008. Se 0009 for falsa, desempilhe e desvie para 0007. Aplique 008 no código.

Para PARA: Empilhar PA0010 e aplicar 0010 no código. Empilhar PA0011,variável, incremento. Fazer variável ← valor1. Colocar em 0012 a condição variável ≤ valor2. Se 0012 for falso, desviar para 0011. Traduzir normalmente. Ao achar o fim{para} consultar o topo, incrementar a variável, desempilhar 0011 guardar em SALVA, desempilhar 0010, desviar para lá. Aplicar SALVA no código.

Um exemplo

```
1: algoritmo TESTE
2: inteiro X,Y,Z,T
3: enquanto X > 10 faça
4:   se X = 14 então
5:     Z ← 10
6:     T ← Y
7:     T ← Z × 6
8:   fim se
9: fim enquanto
10: fimalgoritmo
```

Que foi convertido em

```
1: TESTE ;Z;T;V001;V002;V003;V004;V005;V006
2: V001:
3: V003 ← X > 10
4: → ( ~ V003)/V002
5: V004 ← X = 14
6: → ( ~ V004)/V005
7: Z ← 10
8: T ← Y
9: T ← Z × 6
10: V005:
11: → V001
12: V002:
```

Para você fazer

Exercício 1 A seguir, uma função já compilada. Faça a engenharia reversa e escreva o código fonte original.

```
1: TESTE-46495 ;T;X;V001;V002;V003;V004;V005;V006;V007
2: V001:
3: V003←Z<19
4: →( ~V003)/V002
5: V004:
6: V006←X=12
7: →( ~V006)/V005
8: T←X
9: T←Y
10: X←T*Z
11: →V004
12: V005:
13: →V001
14: V002:
```

Responda no verso

Exercício 2 Suponha a seguinte função fonte

```
1: algoritmo TESTE-19962
2: inteiro X,Y,Z,T
3: se Y ≠ 3 então
4:   para Y de 4 ate 6 passo 1 faça
5:     T ← 15
6:     T ← Z
7:     Z ← 19
8:   fim para
9: fim se
10: fimalgoritmo
```

Que foi compilada na seguinte função objeto

```
1: TESTE-19962 ;T;Z;V001;V002;V003;V004;V005;V006
2: =====
3: →( ~V001)/V002
4: Y ← 4
5: V003:
6: V005←Y≤6
7: →( ~V005)/V004
8: T←15
9: T←Z
10: Z←19
11: Y ← Y + 1
12: =====
13: V004:
14: V002:
```

Informe qual deveria ser o comando que aparece nas linhas duplamente sublinhadas.

1