9- Sequenciadores

– Um sequenciador é uma construção que altera o fluxo de controle normal (uma única entrada e uma única saída)

⇓

Permite a programação de fluxos de controle mais genéricos

– Exemplos

- Saltos (*jumps*)
- Escapes (*escapes*)
- Exceções (*exceptions*)

9- Sequenciadores...

- Saltos (*Jumps*)
  - Comandos simples, seqüenciais, condicionais e iterativos possuem uma única entrada e uma única saída

⇓

Qualquer comando formado pela composição de comandos simples, seqüenciais, condicionais e iterativos também possui uma única entrada e uma única saída

  - Um salto (*jump*) é uma transferência explícita de controle de um ponto do programa para outro
  - A forma mais comum de saltos é: **goto L**, sendo que **L** é um label que denota um ponto do programa. Para amarrar um label a um ponto do programa escreve-se algo da forma: **L: C**

9- Sequenciadores...

– Exemplo

Formas de controle básicas	Seqüência de execução	Com *jump*
Composição (Seqüencial)	S0 S1 S2 S3	S0 goto L1 L2: S2 goto L3 L1: S1 goto L2 L3: S3
Alternativa	S0 S1 S3 ou S0 S2 S3	S0 if A = 0 then goto L1 S1 goto L2 L1: S2 L2: S3
Iteração	S0 S2 ou S0 S1 S2 ou S0 S1 S1 S2...	S0 L1: if A = 0 then goto L2 S1 goto L1 L2: S2

9- Sequenciadores...

- O uso irrestrito de saltos permite que um comando possua várias entradas e saídas
  - Vantagens
    - fácil de usar (em programas pequenos)
    - completamente genérico
    - eficiente
  - Desvantagem: falta de legibilidade (“programas espaguete”)

9- Sequenciadores...

- Muitas LPs impõem restrições à utilização de saltos: em geral, o escopo de um label é o bloco mais próximo que contém sua ocorrência de amarração
- Pascal impõe as seguintes restrições a saltos: um comando **goto** pode ser usado dentro de um bloco ou para saltar de um bloco interno para o externo, mas não pode ser usado para saltar de um bloco externo para um interno. Em particular, apesar de permitir saltar para fora de um procedimento, não é permitido saltar para dentro

< VISUALIZAR EXEMPLO NO QUADRO >

9- Sequenciadores...

- Sequenciadores alteram significativamente a semântica de uma LP. Por exemplo, o comando sequencial C1; C2; é executado através da execução de C1, seguida da execução de C2. Mas se houver um sequenciador em C1, não se sabe se C2 será executado
- Problema relacionado ao salto para fora de um bloco: eliminar variáveis locais ao bloco; abandonar uma ativação de procedimento
- E se o procedimento for recursivo? Qual(is) das ativações deverá(ao) ser abandonada(s)?

$$\Downarrow$$

Labels não podem denotar simplesmente um ponto do programa, mas um ponto do programa de uma determinada ativação do procedimento no qual se encontra

- Saltos, que parecem ser simples, introduzem uma complexidade indesejável à semântica de uma LP

9- Sequenciadores...

- Escapes (*Escapes*)
  - Um escape é um sequenciador que termina a execução de um comando textualmente fechado. Em outras palavras, é uma transferência direta de controle para a saída do comando

⇓

Permite a programação de fluxos de controle com uma única entrada e múltiplas saídas

9- Sequenciadores...

- Exemplo: C

```
while (x <= 100) {
  if (x < 0) {
    cout << “Erro: valor negativo”;
    break;
  }
  ... cin >> x;
}

do {
  cin >> x;
  if (x < 0) {
    cout << “Erro: valor negativo”;
    continue;
  }
  ...
} while (x <= 100);
```

9- Sequenciadores...

- Exemplo: Ada

  Por default, **exit** finaliza o loop mais próximo (interno). No caso de querer terminar um loop mais externo, é necessário dar um nome a ele

  < VISUALIZAR EXEMPLO NO QUADRO >

- Um tipo importante de escape é o **return**. Em geral, LPs impedem que escapes transfiram controle para fora de procedimentos. Isso evita que escapes sejam capazes de terminar ativações de procedimentos
- O tipo escape mais drástico é o **halt**, que termina o programa inteiro

9- Sequenciadores...

- Exceções (*Exceptions*)
  - O que acontece quando uma operação aritmética causa um *overflow*? E uma divisão por zero? E uma operação de entrada/saída não puder ser realizada?
    - Observação: quando essas ***condições excepcionais*** ocorrem, o programa não pode continuar normalmente
    - Alternativa 1: o programa interrompe sua execução $\Rightarrow$ inflexibilidade que compromete a modularidade e a falta de robustez
    - Alternativa 2: o controle é transferido para um tratador de exceções, uma parte do programa que especifica o que deve ser feito em condições excepcionais $\Rightarrow$ robustez

9- Sequenciadores...

- Uma técnica comum para tratar condições de exceção é dar a cada abstração P, um código de resultado, que pode ser uma variável global, um parâmetro de resultado ou um resultado de função. Quando chamado, P seta seu código de resultado para indicar se tudo deu certo ou, caso contrário, qual condição excepcional foi detectada
  - Desvantagens:
    - o programa pode ficar confuso, devido ao número de testes
    - o programador pode omitir um teste de resultado

9- Sequenciadores...

- Um outra técnica é associar um tratador a cada exceção específica
- Uma exceção é uma indicação (sinal) de que surgiu uma condição excepcional
- PL/I foi a primeira LP a incorporar alguma forma geral de tratamento de exceções $\rightarrow$ muito confuso
- Ada incorporou esse conceito na linguagem com mais sucesso:
  - um tratador para qualquer exceção e é associado a um comando C. Se C (direta ou indiretamente) “dispara” e, a execução de C é abandonada e o controle é transferido para o tratador
- Pode-se associar diferentes tratadores para a mesma exceção de diferentes comandos
- Pode-se associar diferentes tratadores para diferentes exceções de um mesmo comando

9- Sequenciadores...

– Exemplo em Java:

```
try {
    // código a ser executado
}
catch (ClasseDeExceção instânciaDaExceção) {
    // tratamento da exceção
}
finally {
    // código a ser executado mesmo que uma exceção seja
    lançada
}
```

9- Sequenciadores...

```
01.public class ExemploDeExcecao {
02.     public static void main(String[] args) {
03.             String var = "ABC";
04.             try {
05.                     Integer i = new Integer(var);
06.                     System.out.println("A variável i vale " + i);
07.             } catch (NumberFormatException nfe) {
08.                     System.out.println("Não é possível atribuir a
09. string " + var + " a um Objeto Inteiro.\n" + "A seguinte
10. mensagem foi retornada:\n\n" + nfe.getMessage());
11.             }
12.     }
13.}
```

- O código acima apresentará algo como:

  Não é possível atribuir a string ABC a um Objeto Inteiro.
  A seguinte mensagem foi retornada:

  For input string: "ABC"
- Perceba que a linha *System.out.println("A variável i vale " + i)* não foi executada, pois houve um erro na linha anterior. Portanto, apenas a mensagem de tratamento do erro NumberFormatException foi impressa na tela.

Retirado de http://www.tiexpert.net/programacao/java/try-catch-finally.php