4- Armazenamento

- Variáveis
  - Variável: objeto que contém valor
  - Modelam objetos do mundo real que possuem um estado
  - Em geral, variáveis são utilizadas através de atribuições e têm tempo de vida curto. Contraexemplo: arquivos
  - Variáveis atualizáveis ≠ variáveis da matemática.

    Ex.: Na matemática não é possível realizar x := x + 1
  - Seja ***n*** uma variável. “O valor de ***n***” quer dizer “o conteúdo da célula denotada por ***n***”
  - Modelo abstrato de armazenamento
    - Uma memória é uma coleção de células
    - Cada célula tem um status corrente: alocada ou livre
    - Cada célula alocada tem um conteúdo corrente: valor armazenável ou indefinido

4- Armazenamento...

- Variáveis Compostas
  - Uma variável de tipo composto consiste de componentes (que são variáveis). O conteúdo desses componentes pode ser inspecionado e atualizado seletivamente

Atualização total x atualização seletiva

  - Atualização seletiva = atualização de um componente de variável
  - Nem toda LP oferece esse recurso. Ex.: ML

4- Armazenamento...

Variáveis arranjo

- Motivação: “quando e como um conjunto de índices de um arranjo é determinado?”
  1. Arranjo estático = em tempo de compilação
  2. Arranjo dinâmico = no momento da criação da variável. Ex. (Algol-68): [m:n] int vetor
     - Obs.: quando esta declaração é encontrada em tempo de execução, o vetor é alocado conforme os valores correntes de m e n; preserva este tamanho até seu desaparecimento quando da saída do seu escopo
  3. Arranjo flexível = quando são feitas atribuições à variável (durante a execução). Ex. (Algol-68): flex [1:0] int a
     - Obs.: O objeto referenciado por a é declarado como um vetor contendo, inicialmente, nenhum inteiro. A instrução a := (2, 3 , 49) altera os limites para [1:3] e atribui valores a todos os seus elementos.

4- Armazenamento...

- Armazenáveis
  - Valores que podem ser armazenados em células simples, que não podem ser atualizados seletivamente. Ex.: PASCAL $\Rightarrow$ valores primitivos, conjuntos, apontadores
- Tempo de vida
  - É o intervalo de tempo entre a criação e a destruição de uma variável
  - Relaciona-se à economia de memória $\Rightarrow$ uma variável só precisa ocupar memória se estiver "*viva*"

4- Armazenamento...

Variáveis locais e globais

- Variável local = declarada dentro de um bloco para uso somente dentro do bloco
- Variável global = pode ser referenciada dentro de um bloco, mas não foi declarada localmente
- Uma ativação de um bloco é um intervalo de tempo durante o qual o bloco está sendo executado

4- Armazenamento...

- Ex.: considere a seguinte estrutura de programa (linguagem hipotética)

```
Bloco A
    Bloco B
        Bloco C
        …
        fim C
        Bloco D
        …
        fim D
    … C, D,... C, D,...
    fim B
 …B...
 fim A
```

Considere a seguinte sequência de chamadas, como disposto no pseudocódigo, originalmente a partir de A: qual o tempo de vida das variáveis deste programa?

4- Armazenamento...

- Em geral, uma variável local não retém o seu conteúdo em ativações sucessivas do bloco onde foi declarada. Contra-exemplo: variável estática, pois é local mas seu tempo de vida é igual ao tempo de execução do programa inteiro

- Exemplo: C – Qual será a saída do programa abaixo?

```
int a = 1;
void f ( ) {
  int b = 1; // inicializado a cada chamada de f
  static int c = a; // inicializado uma única vez
  cout<<“a = ”<<a++
      <<“b = ”<<b++
      <<“c = ”<<c++<<‘\n’;
  c = c +2;
}
void main ( ) { while (a < 4) f ( ); }
```

4- Armazenamento...

Variáveis de heap

- Podem ser criadas e destruídas a qualquer momento e, portanto, o tempo de vida não segue um padrão
- São criadas por comandos
- São anônimas
- São acessadas através de apontadores
- Alocador = cria uma variável de heap que é acessada indiretamente, geralmente por um ponteiro
- Liberador = destrói uma variável de heap

4- Armazenamento...

- Quase todas as LPs imperativas fornecem apontadores, apesar de constituir um conceito de baixo nível, ao invés de dar suporte a tipos recursivos diretamente
- Sejam px e py dois apontadores para uma lista. A atribuição px : = py faz com que as duas variáveis compartilhem a lista
- E se a LP implementar o tipo lista diretamente? Como interpretar o tipo de atribuição acima?
  1. Colocar em px uma referência para a lista referenciada por py:
     - Esta interpretação envolve compartilhamento
     - Pode ser inconsistente com a forma de atribuição de registros ou arranjos
     - Fácil de ser implementada
  2. Colocar em px uma cópia completa da lista referenciada por py:
     - É uma interpretação mais natural
     - Pode ser mais consistente com a forma de atribuição de registros ou arranjos
     - A cópia de listas é de implementação custosa

4- Armazenamento...

Variáveis persistentes

- Variável transiente = o tempo de vida está limitado à ativação do programa que a criou. Ex.: variáveis locais e de heap
- Variável persistente = o tempo de vida transcende uma ativação de um programa particular. Ex.: arquivos
- Arquivos são variáveis compostas que, em geral, são utilizadas para conter grande quantidade de dados com tempo de vida longo
- Muitas LPs restringem a forma de utilização e atualização de arquivos, por questões de eficiência. Por exemplo, proíbem a atribuição de um arquivo completo
- O Princípio da Completeza de Tipos sugere que todos os tipos da LP sejam permitidos para variáveis transientes e persistentes. Assim, não haveria tipos, comandos ou procedimentos especiais para operações de entrada-saída

4- Armazenamento...

Referências “penduradas” (dangling)

- Referência pendurada é um ponteiro que aponta para uma área de memória que foi desalocada
- Principais situações em que pode ocorrer:
  - Quando uma referência a uma variável local é atribuída a uma variável com tempo de vida maior. Exemplo (LP hipotética)

```
var r: ^Integer ;
procedure P;
  var v: Integer;
  begin
    r := &v;
  end;
begin
  P;
  r^ := 1
End
```

  - Quando uma LP possui um liberador, este poderá transformar todas as referências à variável liberada em referências penduradas

4- Armazenamento...

- Algumas soluções:
  - Não permitir que uma referência a uma variável local seja atribuída a uma variável com tempo de vida maior.

    Desvantagem: pode requerer verificações em tempo de execução
  - Tratar todas as variáveis como variáveis de heap $\Rightarrow$ ao fim de uma ativação de bloco as variáveis declaradas dentro dele continuam a existir enquanto houver alguma referência a elas

    Desvantagem: alocação de memória é menos eficiente

4- Armazenamento...

- Comandos
  - São frases de programa que serão executadas a fim de atualizar variáveis
  - Caracterizam as LPs imperativas
  - Tipos fundamentais de comandos:
    A. Saltos
    - Salto (*skip*) ou comando vazio (*dummy*) é a forma mais simples de comando, pois nada realiza
    - Exemplo: **if** *E* **then** *C* **else** *skip*;

4- Armazenamento...

B. Atribuições

- Em geral, possuem a forma V := E, onde V é uma acesso a variável, cujo valor passará a ser E
- O que significa acesso a uma variável? Em outras palavras, qual o significado de n, nos comandos read (n); n := n +1; write (n); ?
  - Pode ser uma referência a uma variável em alguns contextos e o conteúdo atual da variável em outros contextos; ou
  - Pode ser sempre uma referência a uma variável, mas em alguns contextos há uma operação implícita de de-referenciação. Em algumas LPs, a de-referenciação deve ser explícita. Exemplo (ML): n := !n + 1;
- Alguns tipos de atribuições
  - Múltipla: $V_1 := ... := V_n := E$
  - Simultânea: $V_1,..., V_n := E_1,..., E_n$
  - Combinada com operador binário: $V += E$

4- Armazenamento...

C. Chamadas de procedimento

- Aplica uma abstração de procedimento a alguns argumentos
- Em geral tem a forma $P(AP_1,...,AP_n)$, onde P determina a abstração de procedimento a ser aplicada e os parâmetros reais $AP_1,...,AP_n$ determinam os argumentos a serem passados
- Um argumento pode ser um valor (resultado de uma expressão) ou o acesso a uma variável

4- Armazenamento...

D. Comandos sequenciais

- Fluxo de controle mais comum
- Em geral têm a forma $C_1; C_2; \ldots ; C_n$, onde o comando C_1 é executado antes de C_2, e assim por diante
- Está disponível em toda LP imperativa

4- Armazenamento...

E. Comandos colaterais

- Comandos são executados sem uma ordem definida
- Em geral têm a forma $C_1, C_2..., C_n$
- Comandos colaterais são não-determinísticos:
  - Uma computação é determinística se for possível predizer com precisão a sequência de passos que será executada. Caso contrário, ela é não determinística
  - Exemplo: x := 5, x := x + 1;
- Mas um comando colateral pode ser *efetivamente determinístico* se nenhum subcomando utilizar uma variável atualizada por outro

4- Armazenamento...

F. Comandos condicionais

- Possui vários subcomandos, dentre os quais somente um é escolhido para ser executado
- O comando *if* é o mais simples e pode ser encontrado em toda LP imperativa
- O comando *if* não-determinístico pode existir e ser útil, por exemplo e principalmente, para programação concorrente
- Existem também comandos condicionais com escolha baseada em valores diferentes de lógicos

4- Armazenamento...

G. Comandos iterativos

- Um comando iterativo (*loop*) tem um subcomando (corpo do *loop*) que será executado repetidamente e, em geral, algum tipo de sentença que determina quando a iteração deverá parar
- Comandos iterativos podem ser:
  - Indefinidos: não se conhece o número de iterações. Exemplo: while
  - Definidos: é possível definir o número de iterações. Caracteriza-se pelo uso de variável de controle. O tratamento de variáveis de controle varia de linguagem para linguagem. Exemplo: for
- Questionamentos:
  1. Qual o valor da variável de controle após o término da repetição?
  2. Qual o valor da variável de controle após um salto para fora da repetição?
  3. O que acontece quando o corpo do *loop* possui uma atribuição para a variável de controle?

4- Armazenamento...

- Expressões com efeitos colaterais
  - São expressões que ao serem avaliadas causam o efeito colateral de atualizar variáveis

  Expressões de comandos
  - Permitem utilizar o poder da atribuição e da iteração no cálculo de resultados de funções

4- Armazenamento...

- Exemplo (LP hipotética)

```
var p: real;
      i: integer;

begin
  p := a[n];
  for i := n-1 downto 0 do
    p := p * x + a[i];
  yield p
end
```

- Em Pascal, por exemplo, o corpo de uma abstração de função é uma expressão de comando
- Qualquer tipo de expressão de comando introduz a possibilidade de efeitos colaterais

4- Armazenamento...

– Exemplo: uma função que lê um caractere de um arquivo e retorna o caractere lido, tem um efeito colateral sobre o arquivo

E	if getchar (f) = ‘F’ then
R	sexo := feminino
R	else if getchar (f) = ‘M’ then
O	sexo := masculino

4- Armazenamento...

Linguagens orientadas para expressões

- Um linguagem orientada para expressões é uma linguagem imperativa que elimina todas as distinções entre expressões e comandos. A avaliação de expressões retorna um valor e, em geral, possui o efeito colateral de atualizar o valor da variável
- Por exemplo, é possível escrever V' := (V := E)
- Exemplos de LPs: Algol-68 e C
- Vantagem: maior simplicidade e uniformidade
- Desvantagem: encoraja o uso de efeitos colaterais, o que pode gerar um estilo de programação crítico.

Exemplo: while (ch := getchar (f)) <> '*' do write (ch)