INE5416
Paradigmas de Programação

Ricardo Azambuja Silveira
INE-CTC-UFSC
E-Mail: silveira@inf.ufsc.br
URL: www.inf.ufsc.br/~silveira

O paradigma imperativo

- Apoiado ńos conceitos de:
  - Identificadores
  - Tipos
  - Atribuição, operadores e expressões
  - Estruturas de controle
  - subprogramas

Nomes

- Um nome é uma cadeia de caracteres usada para identificar alguma entidade de um programa (variáveis, rótulos, subprogramas, parâmetros, etc)
- As primeiras linguagens usavam um único caractere
- Questões de projeto
  - Qual o tamanho máximo?
  - Caracteres de conexão são permitidos?
  - Há distinção entre maiúsculas e minúsculas?
  - As palavras especiais são palavras reservadas ou palavras-chave?

Nomes

- Tamanho
  - Fortran I: máximo 6
  - Cobol: máximo 30
  - Fortran 90 e ANSI C: máximo 31
  - ADA e Java: nenhum limite e os nomes são significativos
  - C++: nenhum limite, mas implementadores geralmente os impõem
- Conectores
  - Pascal, Modula-2 e Fortan 77 não permitem inserir caracteres especiais separando palavras

Nomes

- Distinção entre maiúsculas e minúsculas
  - Desvantagem: legibilidade (nomes que se parecem são diferentes)
  - Em C, C++, Java e Modula 2 os nomes são *case sensitive*
  - Em java os nomes predefinidos usam maiúsculas e minúsculas misturadas
- Palavras especiais
  - Uma palavra-chave é uma palavra que é especial apenas em algum contexto
    - Desvantagem: Dificulta a legibilidade
  - Uma palavra reservada é uma palavra especial que não pode ser utilizada como um nome definido pelo usuário

Variáveis

- É uma abstração de uma célula ou um conjunto de células de memória de um computador
- Pode ser caracterizada como um sêxtuplo de atributos:
  - Nome, endereço, valor, tipo, tempo de vida e escopo
- Nomes: conforme discutido anteriormente
- Endereço: A localização da célula de memória ao qual está associada
  - Uma variável pode ter diferentes endereços em diferentes momentos durante a execução de um programa (ex: recursão)
  - Uma variável pode ter diferentes endereços em diferentes pontos de um programa (ex: sub-rotinas)
  - Se dois nomes de variáveis podem ser utilizados para acessar a mesma localização de memórias, eles cão chamados apelidos ou *aliases*
  - Aliases são um problema para a legibilidade

Variáveis

- Como os apelidos podem ser criados
  - Ponteiros, reference variables, Pascal variant records, C and C++ unions, FORTRAN EQUIVALENCE e através de parâmetros
- Algumas justificativas para o uso de apelidos, como reuso de memória, não é mais válido. Em vez disso usa-se alocação dinâmica

Variáveis

- Tipos: determina a faixa de valores das variáveis, sua precisão e o conjunto de operações definidas
- Valores: o conteúdo da célula de memória associada à variável
- Célula de memória abstrata: a célula física ou um conjunto de células físicas formando uma unidade de armazenamento de tamanho adequado, associado a uma variável
  - O *l-value* de uma variável é o seu endereço
  - O *r-value* de uma variável é o seu valor

Vinculação

- Uma vinculação (*binding)* é uma associação, por exemplo, entre um atributo e uma entidadeou entre uma operação e um símbolo
- Tempo de vinculação (*binding time*) é o momento no qual a vinculação ocorre
- Possíveis tempos de vinculação:
  - **Tempo de projeto** da linguagem: vincular símbolos de operadores a operações
  - **Tempo de implementação** da linguagem: vincular um tipo de dados a sua forma de representação
  - **Tempo de compilação**: ligar uma variável de um programa a um tipo
  - **Tempo de carregamento**: uma chamada a um subprograma é vinculada ao seu código
  - **Tempo de carga**: uma variável estática é ligada a uma célula de memória
  - **Tempo de execução**: ligar uma variável dinâmica a uma célula de memória

Vinculação de atributos a variáveis

- Uma vinculação é **estática** se ocorre anteriormente ao tempo de execução e se mantêm imutável durante este período
- Uma vinculação é **dinâmica** se ocorre durante a execução do programa ou pode mudar durante este período
  - Vinculações e desvinculações executadas pelo sistema operacional ou pelo hardware (memória virtual ou *cache*) não estão incluídas neste conceito

Vinculação de tipos

- Como um tipo é especificado?
- Em que momento a vinculação ocorre?
- Vinculação estática:
  - Declaração explícita: Uma instrução de programa usada para declarar o tipo das variáveis
  - Declaração implícita: é um mecanismo de associação do tipo às variáveis por convenção padrão
    - FORTRAN, PL/I, BASIC, e Perl provêem declaração implícita
      - *vantagem:* facilidade de escrita
      - *Desvantagem:* legibilidade

Vinculação dinâmica de tipos

- Especificada através de um comando de atribuição (exemplo: APL, Snobol e JavaScript)

  LIST = [2, 4, 6, 8]

  LIST = 17.3

  - *Vantagem:* flexibilidade (unidades de programas genéricos)
  - *Desvantagens:*
    - Alto custo computacional (verificação dinâmica de tipos e interpretação)
    - Detecção de erros de digitação pelo compilador é difícil
- Inferência de tipos (ML, Miranda e Haskell)
  - Os tipos são determinados pelo contexto da referência, em vez de por comandos de atribuição

Vinculação de armazenamento e tempo de vida

- *Alocação* – Obtenção de uma célula de memória a partir de um conjunto de células para ser vinculada a uma variável
- *Desalocação* – devolução da célula vinculada a uma variável ao conjunto de células disponíveis
- O *tempo de vida* de uma variável é o tempo durante o qual ela está vinculada a uma célula da memória em particular
- Categorias das variáveis de acordo com o seu tempo de vida:
  - Estáticas
  - Dinâmicas na pilha
  - Dinâmicas no monte explícitas
  - Dinâmicas no monte implícitas

Vinculação de armazenamento e tempo de vida

- **Variáveis estáticas**
  - **Ligada a uma célula de memória antes do começo da execução e permanece ligada a mesma célula durante todo o ciclo de execução**
    - **Exemplos: Todas as variáveis do FORTRAN 77 e variáveis estáticas do C e Java**
    - **Vantagens: Eficiência (endereçamento direto), Suporte a subprogramas sensíveis a história**
    - **Desvantagens: falta de flexibilidade, não permite recursão**

- **Variáveis dinâmicas de pilha**
  - **Vinculações de armazenamento são feitas quanto os comandos de declaração são elaborados**
  - **Variáveis escalares têm todos os atributos, exceto o endereço, ligadas estáticamente**
    - **Exemplo: Variáveis locais em Pascal e em subprogramas em C**
    - **Vantagens: Permite recursáo, conserva memória**
    - **Desvantagens: sobrecarga de processamento para alocação e desalocação, subprogramas não podem ser sensíveis a história e as referencias são ineficientes (endereçamento indireto)**

Vinculação de armazenamento e tempo de vida

- Variáveis dinâmicas no monte explícitas
  - Alocada e desalocada através de diretivas explícitas, especificadas pelo programador, e que tem efeito durante a execução
  - Referenciadas apenas através de ponteiros ou referências
  - Exemplos: objetos dinâmicos em C++ (via new and delete) e todos os objetos em Java
  - Vantagens: provê recurso para gerenciamento de armazenamento dinâmico
  - Desvantagens: ineficiente e pouco confiável

Vinculação de armazenamento e tempo de vida

- Variáveis do monte implícitas
  - Alocada e desalocada através de comandos de atribuição
  - Exemplo: todas as variáveis em APL
  - *Vantagem:* flexibilidade
  - *Desvantagens:*
    - Ineficiente, porque todos os atributos são dinâmicos
    - Perda da detecção de erros

Verificação de tipos

- Generaliza-se o conceito de operandos e operadores para incluir subprogramas e comandos de atribuição
- *Verificação de tipos* é a atividade de assegurar que os operandos de um operador são de tipos compatíveis
- *Tipos compatíveis* são aqueles que são legais para o operador, ou é permitido, sob as regras da linguagem que sejam implicitamente convertidos pelo código gerado pelo compilador, para um tipo válido. Esta conversão automática é denominada *coerção*.
- Um *erro de tipo* é a a aplicação de um operador a um operando de tipo impróprio
  - Se todas as vinculações de tipos forem estáticas, a verificação quase sempre poderá ser feita estaticamente
  - Se todas as vinculações de tipos forem dinâmicas, a verificação deve sempre ser feita dinamicamente
- Uma linguagem de programação é *fortemente tipificada* se os erros de tipo são sempre detectados

Verificação de tipos

- *Vantagens da tipificação forte:* permite a detecção de usos equivocados das variáveis que resultam em erros de tipo
- Languages:
  1. FORTRAN 77 não é (não há verificação entre parâmetros reais e formais)
  2. Pascal é quase (variant records)
  3. Ada e Modula-3 são pouco mais que pascal (checagem de parâmetros pode ser desabilitada)
  4. C and C++ não é (checagem de parâmetros pode ser desabilitada; unions não tem verificação de tipos
  5. Ada e java são quase (conversão explícita e coerção)
- Regras de coerção afetam fortemente a tipagem — elas podem enfraquecê-la

Escopo

- O *escopo* de uma variável é a faixa de comandos dentro da qual ela é visível
- *Variáveis não-locais* de uma unidade de programa são aquelas que são visíveis mas não declaradas dentro da unidade
- As regras de escopo de uma linguagem determinam como as referências aos nomes são associadas com as variáveis

Escopo

- Escopo estático
  - Baseado no texto do programa
  - Para conectar um nome de referência a uma variável é preciso encontrar a declaração
  - *Processo de busca:* procura-se a declaração primeiramente no bloco local, depois vai incrementado o escopo até que o nome referenciado seja encontrado
  - O Encapsulamento de um escopo estático para um escopo específico é chamado seu *pai estático (static ancestors)*; os ancestrais mais próximos são chamados *ancestrais estáticos (static parent)*

- Escopo dinâmico
  - Baseado na chamada de seqüências das unidades de programas, não seu layout textual (temporal versus espacial)
  - Referências a variáveis são conectadas a declarações através da busca de volta através da cadeia de chamadas a subprogramas que forçaram a execução a este ponto

Constantes nomeadas

- Uma constante nomeada é uma variável que é vinculada a um valor somente no momento em que ela é vinculada a um armazenamento
  - *Vantagens:* legibilidade e facilidade de modificação
- A vinculação dos valores a constantes nomeadas pode ser estática (chamada constantes manifestas) ou dinamicas
- *Linguagens:*
  - *Pascal:* literals only
  - *Modula-2 and FORTRAN 90:* constant-valued
  - expressions
  - *Ada, C++, and Java*: expressions of any kind
- Inicialização de variáveis
  - A vinculação de uma variável a um valor no momento de sua vinculação ao armazenamento é denominado *initialização*
  - Initialização é frequentemente feito no comando de declaração. Ex: Ada

    SUM : FLOAT := 0.0;