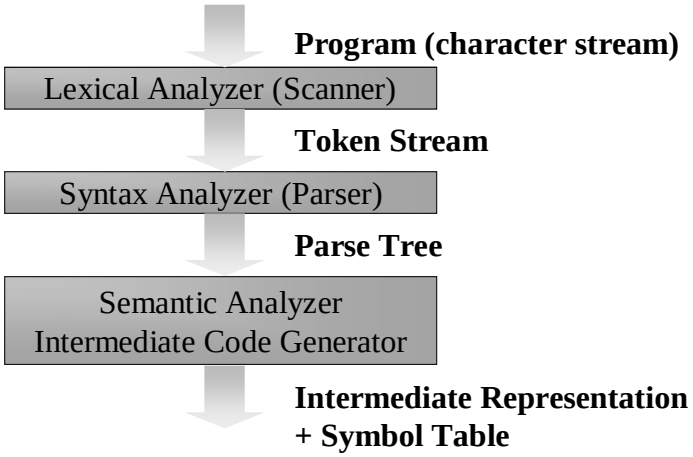
Análise Semântica e Representação Intermédia

Compiladores, Aula Nº 20

João M. P. Cardoso

Localização actual nas etapas de compilação



O que é a semântica de um programa?

- Sintaxe
  - Como o programa é constituído
  - Representação textual ou estrutura
- Semântica
  - Qual é o significado do programa?

Qual o motivo da análise semântica?

- Certifica-se de que o programa está de acordo com as definições da linguagem de programação
- Reportar, sempre que haja erros semânticos, mensagens de erro que sejam úteis para o utilizador
- Não é preciso muito trabalho adicional se for incorporada durante a criação da representação intermédia

Erros na Análise Semântica

```
boolean sum(int A[], int N) {
  Int i, sum;
  For(i=0; i<N; i++) {
    sum1 = sum + A[i];
  }
  return sum;
}
...
Int s = sum(A);
```

Não foi atribuído valor inicial a sum

Variável não declarada

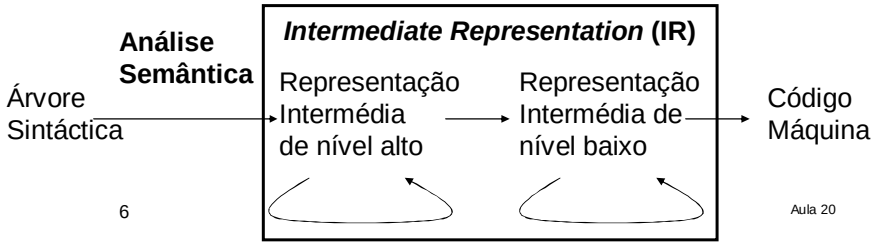
Tipo da variável retornada não confere com a declaração do cabeçalho da função

Falta um argumento

Inconsistência entre tipos no RHS e LHS

Objectivo das representações intermédias do programa

- Permitir análises e transformações
  - Optimizações
- Estruturar tradução para código máquina
  - Sequência de passos



Representação Intermédia de Nível Alto

- Preserva o fluxo de controlo estruturado
- Útil para optimizações ao nível do ciclo
  - Desenrolamento de ciclos, Fusão de ciclos, etc.
- Preserva estrutura ao nível dos objectos
- Útil para optimizações em programas orientados por objectos

Representação Intermédia de Nível Baixo

- Passa do modelo de dados abstracto para o espaço planar de endereçamento
- Elimina o fluxo de controlo estruturado
- Útil para tarefas de compilação de nível baixo
  - Afectação de registos
  - Selecção de instruções

Alternativas

- Há muitas alternativas possíveis
  - Árvores de instruções e de expressões
  - Grafos direccionados acíclicos (DAGs)
  - Código de 3 endereços (C3E)
  - E muitas outras...
- Mais ou menos especificas à própria linguagem
- Estas lições apresentam uma possibilidade de árvores de instruções e de expressões (mais tarde falaremos também de código de 3 endereços)

Tarefas de Compilação

- Determina formato das estruturas na memória
  - Determinar formato de arrays e de objectos na memória
  - Determinar formato da pilha de chamadas na memória
- Gerar código
  - para ler valores
    - parâmetros, elementos de arrays, campos de objectos
  - para avaliar expressões e computar valores novos
  - para escrever valores
  - para estruturas de controlo
- Enumera funções e cria a tabela de funções
  - Invocação de funções acede à entrada correspondente na tabela de funções
- Gera código para funções
  - variáveis locais, e acesso a parâmetros
  - Invocações de funções

Tabelas de símbolos (*symbol tables*)

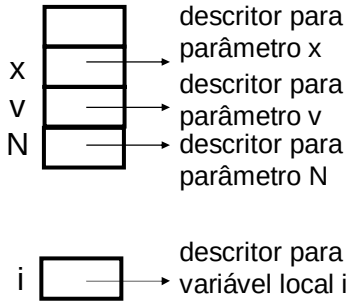
- Conceito chave na compilação
  - enquanto a declaração de tipos, variáveis e funções são processadas vamos atribuir significados a esses identificadores utilizando tabelas de símbolos
- Compiladores utilizam tabelas de símbolos para produzirem
  - Layout das estruturas na memória
  - Tabelas de funções
  - Código para aceder a campos, variáveis locais, parâmetros, etc.

Tabelas de Símbolos

- Durante a tradução de árvores sintácticas para representação intermédia
  - Tabelas de símbolos mapeiam identificadores (strings) em descritores (informação acerca dos identificadores)
  - Operação básica: Lookup
    - Dada uma string, encontrar o seu descritor
    - Implementação típica: Hash Table (contentor associativo)
- Exemplo
  - Dado o nome de uma variável, encontrar descritor
    - Descritor local, descritor de parâmetro, descritor global

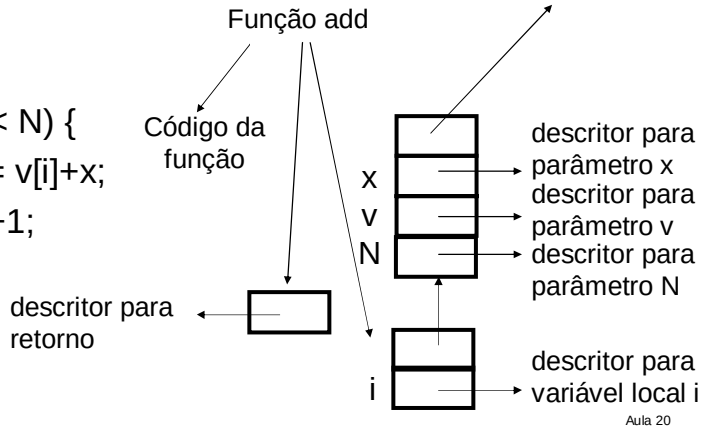
Exemplo

```
void add(int x, int[] v, int N)
   {
   int i;
   i = 0;
   while (i < N) {
        v[i] = v[i]+x;
        i = i+1;
   }
}
```



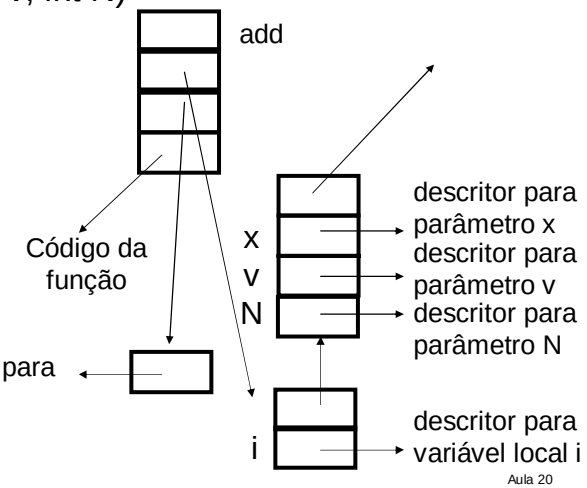
Exemplo

```
void add(int x, int[] v, int N)
   {
   int i;
   i = 0;
   while (i < N) {
        v[i] = v[i]+x;
        i = i+1;
   }
}
```



Exemplo

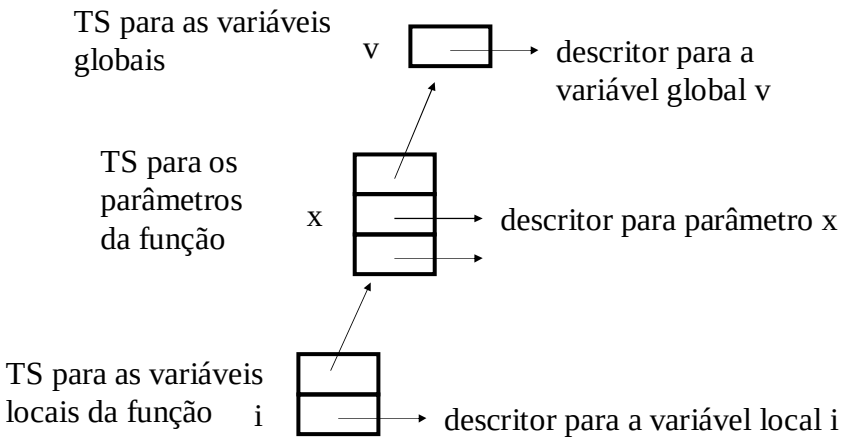
```
void add(int x, int[] v, int N)
   {
   int i;
   i = 0;
   while (i < N) {
         v[i] = v[i]+x;
         i = i+1;
   }
}
```



Hierarquia em tabelas de símbolos

- Alcance/Escopo (*scope*)
  - O mesmo nome para uma variável pode ter significados diferentes em locais diferentes
  - É necessária uma tabela de símbolos por cada escopo
- A hierarquia deriva de
  - Encadeamento de escopos
- Hierarquia na tabela de símbolos reflecte esta hierarquia
- *Lookup* atravessa de modo ascendente a hierarquia até que o descritor seja encontrado

Lookup i num exemplo



Lookup I num exemplo

- v[i] = v[i]+x;
- 1º vai procurar na TS das variáveis locais e só se não encontrar é que sobe na hierarquia das TS