SBVCONC: Construção de Compiladores

Aula 04: A Linguagem de Programação CPRL

Bacharelado em Ciência da Computação
Prof. Dr. David Buzatto

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Compiler PRoject Language (CPRL)

- Linguagem de programação pequena, mas completa, com construções similares às encontradas em Ada, Java, C++ e Pascal;
- Projetada para ser adequada ao ser usada como uma linguagem de programação para o projeto de um compilador em disciplinas avançadas do nível de graduação ou introdutória em nível de pós-graduação;
- Os recursos ilustram muitas das técnicas básicas e problemas associados à tradução de linguagens de programação.

Considerações Gerais Sobre o Léxico

- Sensível à caixa (*case sensitive*), ou seja, letras maiúsculas e minúsculas são consideradas diferentes em todos os tokens, inclusive as palavras reservadas;
- Caracteres de espaço em branco separam os tokens, caso contrário, são ignorados;
- Nenhum token pode se estender depois do fim de uma linha;
- Espaços não podem aparecer em nenhum token, com exceção de caracteres e literais de Strings;
- Comentários começam com duas barras (//) e vão até o fim da linha.

Identificadores

- Identificadores iniciam com uma letra e podem conter letras (sem acentos) e dígitos;
- Um identificador precisa caber em uma única linha e todos os caracteres dos identificadores são significativos;

```
identifier = letter ( letter | digit )* .
letter = [A-Za-z] .
digit  = [0-9] .
```

Palavras Reservadas

and	array	begin	Boolean	Char
class	const	declare	else	elsif
end	exit	false	for	function
if	in	is	Integer	loop
mod	not	of	or	private
procedure	program	protected	public	read
readln	return	String	then	true
type	var	when	while	write
writeln				

Algumas palavras-chave não são usadas atualmente na CPRL, mas são reservadas para um possível uso futuro.

Literais

- Literal inteiro: um ou mais dígitos
  - Exemplos: `0, 1, 1842`
- Literal booleano: `true` ou `false`
- Literal de caractere: um único caractere cercado por aspas simples
  - Diferentes de um literal de String de comprimento um
  - Exemplos: `'A', 'x', '\'`
  - Contrabarra (\) denota sequências de escape em literais de caracteres e Strings: `\t, \n, \" , \'`
- Literal de Strings: um ou mais caracteres imprimíveis cercados por um par de aspas duplas
  - O tipo String não é totalmente suportado na CPRL, por questões de simplicidade.

Outros Tokens (delimitadores e operadores)

```
:   ;   ,   .   (   )   [   ]     // um caractere
+   -   *   /   <   =   >

:=   !=   >=   <=                   // dois caracteres
```

Declarações, Instruções (*Statements*) e Expressões (*Expressions*)

- As três maiores categorias sintáticas na CPRL são as declarações, as instruções e as expressões;
- As declarações introduzem novos nomes (identificadores definidos pelos usuários) ao programa:
  - Exemplos incluem declaração de variáveis, tipo de arrays e subprogramas;
  - Uma declaração deve ser feita antes do nome introduzido poder ser usado;
- As instruções executam ações básicas ou controlam o fluxo de execução:
  - Exemplos incluem instruções de atribuição, de laço (*loop*) e condicionais (*if*).
- As expressões são entidades sintáticas que têm valores e esses valores possuem um tipo como Integer ou Boolean.
  - Exemplos:
    - Literais como `100` ou `false`
    - Expressões de variáveis, também chamas da valores nomeados (*named values*) como `i` ou `x`
    - Expressões compostas envolvendo operadores e operandos como `x + 7` ou `i < 100`.

Expressões versus Instruções

- Ao contrário de muitas linguagens de programação, a CPRL faz uma forte distinção entre expressões e instruções;
- Por exemplo, em C, qualquer expressão seguida por um ponto e vírgula é considerada uma instrução:

```
x >= 5;   // instrução válida em C, mas não em CPRL
```

- Em Java podemos chamar uma função (método) que retorna uma valor sem de fato usar o valor retornado;
  - Em CPRL, todas as chamadas de funções retornam valores e todas essas chamadas são consideradas expressões. O valor retornado por uma chamada de função não pode ser ignorado;
  - Uma chamada de procedimento (construção análoga aos métodos `void` em Java) em CPRL não retorna um valor, sendo considerada uma instrução.

Tipagem na CPRL

- A CPRL é uma linguagem com tipagem estática;
  - Todas as variáveis ou constantes na linguagem pertencem à exatamente um tipo;
  - O tipo é uma propriedade estática e pode ser determinado pelo compilador, permitindo assim detecção de erros em tempo de compilação.

Tipos

- Tipos Escalares Padrão (pré-definidos):
  - `Boolean`
  - `Integer`
  - `Char`
- Tipo Array:
  - Arrays de uma dimensão, sendo permitida a declaração de arrays de arrays;
  - O número de elementos do array precisa ser fornecido, além do tipo de cada elemento;
  - Exemplos:

```
type T1 = array[10] of Boolean;

type T2 = array[10] of Integer;

type T3 = array[10] of T2;
```

  - Os índices são inteiros variando de `0` a `n-1`, onde `n` é o número de elementos do array.

Constantes e Variáveis

- As constantes e as variáveis precisam ser declaradas antes de serem referenciadas;
- Constantes:
  - Exemplo: `const maxIndex := 100;`
  - O tipo do identificador da constante é inferido à partir do tipo do literal do valor;
- Variáveis:
  - Exemplos:

```
var x1, x2 : Integer;
var found : Boolean;
type IntArray = array[100] of Integer;
var table : IntArray;
```

Operadores

- Os operadores, em ordem de precedência, são os seguintes:
  1. Negação booleana: `not`
  2. Operadores de multiplicação: `*`, `/`, e `mod`
  3. Operadores unários de adição: `+` e `-`
  4. Operadores binários de adição: `+` e `-`
  5. Operadores relacionais: `=`, `!=`, `<`, `<=`, `>` e `>=`
  6. Operadores lógicos: `and` e `or`

Expressões

- Para os operadores binários, ambos os operandos devem ser do mesmo tipo;
- Similarmente, para compatibilidade na atribuição, tanto o lado esquerdo quanto direito devem ser do mesmo tipo;
- Expressões lógicas, ou seja, que envolvem os operadores lógicos **and** ou **or** usam avaliação em curto-circuito.

Equivalência de Tipos

- Objetos são considerados do mesmo tipo somente se possuem o mesmo nome de tipo:
  - "Equivalência nomeada" de tipos;
- **Exemplo:**

```
type T1 = array[10] of Integer;
type T2 = array[10] of Integer;
var x : T1;
var y : T1;
var z : T2;
```

- No exemplo acima, x e y tem o mesmo tipo, mas x e z não.

Instrução de Atribuição

- O operador de atribuição é o **:=**
- Uma instrução de atribuição tem a seguinte forma:

```
variable := expression;
```

- **Exemplo:**

```
i := 2 * i + 5;
```

Instrução **if**

- Inicia com a palavra-chave **if** e termina com as palavras chave **end if**, seguidas de um ponto e vírgula;
- Pode conter zero ou muitas cláusulas **elsif** (atenção ao nome!) e uma cláusula **else** opcional;
- **Exemplos:**

```
if x > 0 then
    sign := 1;
elsif x < 0 then
    sign := -1;
else
    sign := 0;
end if;
```

```
if a[i] = searchValue then
   found := true;
end if;
```

Instruções de Repetição/Laço (**loop**) e de Saída (**exit**)

- Uma instrução de repetição pode ser precedida por uma cláusula **while** opcional, mas o corpo da instrução de repetição é cercada pelas palavras-chave **loop** e **end loop**;
- Uma instrução **exit** pode ser usada para sair do laço mais interno que a contém;
- **Exemplos:**

```
while i < n loop
    sum := sum + a[i];
    i := i + 1;
end loop;
```

```
loop
   read x;
   exit when x = SIGNAL;
   process(x);
end loop;
```

Instruções de Entrada/Saída (*Input/Output*)

- A CPRL define somente E/S sequencial de texto, para dois fluxos (*streams*) de caracteres:
  - Entrada padrão (*standard input*);
  - Saída padrão (*standard output*)
- As instruções **`write`** e **`writeln`** podem possuir múltiplas expressões separadas por vírgulas;
- A entrada é suportada apenas para inteiros e caracteres;
- **Exemplos:**

```
read x;
writeln "The answer is ", 2*x + 1;
```

Programas

- Um programa tem uma seção declarativa seguida por uma seção de instruções:
  - A seção declarativa consiste em uma lista de declarações iniciais, que pode ser vazia, seguida por uma lista de declarações de subprogramas, que também pode ser vazia;
  - A seção de instruções é cercada pelas palavras reservadas **begin** e **end**;
  - Um ponto indica o fim do código de um programa;
- **Exemplos:**

```
begin
    writeln "Hello, world.";
end.
```

```
var x : Integer;
begin
   read x;
   writeln "x = ", x;
end.
```

Subprogramas

- A CPRL provê duas formas separadas de subprogramas:
  - Procedimento (*procedure*):
    - Similar à uma função **`void`** nas linguagens C, C++ e Java;
    - Não retorna valor;
    - Invocada através de uma instrução de chamada de procedimento;
  - Função (*function*):
    - Deve retornar um valor;
    - Invocada como parte de uma expressão;
  - Invocações recursivas de subprogramas são permitidas!
  - Todos os subprogramas devem ser declarados antes de serem invocados;
  - Todos os nomes dos subprogramas devem ser diferentes;
  - O nome de um subprograma deve ser repetido após o **`end`** de fechamento da declaração do mesmo.

Procedimentos

- Similares aos procedimentos da linguagem Pascal, com exceção de que instruções `return` explícitas são permitidas na seção de instruções. Nos procedimentos, a instrução `return` não pode ser seguida de uma expressão;
- Procedimentos são invocados simplesmente ao se fornecer seus nomes, seguidos por uma lista separada por vírgulas de argumentos (*actual parameters*/parâmetros reais) cercados por parênteses e com um ponto e vírgula;
  - Se não houver argumentos, somente o nome do procedimento é requerido;
- Chamadas à procedimentos são consideradas instruções:

```
procedureCallStmt = procId ( actualParameters )? ";"

actualParameters = "(" expressions ")" .
```

Exemplo de Procedimento

```
procedure sort(var a : A) is
    var i, j, save : Integer;
begin
    i := 1;
    while i < arraySize loop
        save := a[i];
        j := i - 1;

        while j >= 0 and save < a[j] loop
            a[j + 1] := a[j];
            j := j - 1;
        end loop;

        a[j + 1] := save;
        i := i + 1;
    end loop;
end sort;
```

Assuma as seguintes declarações globais:

```
const arraySize := 10;
type A = array[arraySize] of Integer;
```

Funções

- As funções são similares aos procedimentos, mas ao contrário deles, devem retornar valores;
- As chamadas à funções são expressões;
- Uma função retorna um valor ao executar a instrução `return` na forma:

```
return <expression>;
```

Exemplo de Função

```
function max(x, y : Integer) return Integer is
begin
    if x >= y then
        return x;
    else
        return y;
    end if;
end max;
```

Parâmetros

- Na CPRL existem dois tipos de modos de parâmetros: parâmetros de valor (*value parameters*) e parâmetros variáveis (*variable parameters*);
- Parâmetros de valor são passador por valor (cópia) e são o padrão;
- Parâmetros variáveis são passados por referência e precisam, obrigatoriamente, ser declarados usando a palavra chave **var** como no exemplo abaixo:

```
procedure inc(var x : Integer) is
begin
    x := x + 1;
end inc;
```

- Parâmetros variáveis não são permitidos em funções.

Instruções de Retorno

- Uma instrução de retorno termina a execução de um subprograma e retorna o controle de volta ao ponto onde o subprograma foi invocado;
- Uma instrução de retorno dentro de uma função deve ser seguida de uma expressão cujo valor avaliado é do tipo de retorno da função;
- Uma instrução de retorno dentro de um procedimento não pode ser seguida por uma expressão. Ela simplesmente retorna o controle à instrução que se segue à invocação do procedimento.
- Um procedimento possui uma instrução de retorno implícita como última instrução, sendo assim, a maioria dos procedimentos não terão uma instrução de retorno explícita;
- Uma função requere uma ou mais instruções de retorno para retornar o valor da função. Não é instrução de retorno implícita no fim de uma função.

O Subconjunto CPRL/0 da CPRL

- É a parte da linguagem não relacionada a arrays e subprogramas;
- Inclui:
  - Programas;
  - Tipos pré-definidos;
  - Declaração de constantes;
  - Declaração de variáveis;
  - A maioria das instruções permitidas (condicionais, laços etc);
- Exclui:
  - Declaração de tipo de array;
  - Declaração de subprogramas;
  - Instruções de chamadas de procedimentos;
  - Expressões de chamadas de funções;
  - Instruções de retorno.

Definição Completa da Linguagem CPRL

- A definição completa da linguagem CPRL pode ser encontrada no Apêndice C, fornecido no material da disciplina, cedido gentilmente pelo Professor Moore, autor da obra Introduction to Compiler Design: an Object Oriented Approach Using Java. 2. ed.

Bibliografia

MOORE JR., J. I. **Introduction to Compiler Design: an Object Oriented Approach Using Java**. 2. ed. [s.l.]:SoftMoore Consulting, 2020. 284 p.

AHO, A. V.; LAM, M. S.; SETHI, R. ULLMAN, J. D. **Compiladores: Princípios, Técnicas e Ferramentas**. 2. ed. São Paulo: Pearson, 2008. 634 p.

COOPER, K. D.; TORCZON, L. **Construindo Compiladores**. 2. ed. Rio de Janeiro: Campus Elsevier, 2014. 656 p.

JOSÉ NETO, J. **Introdução à Compilação**. São Paulo: Elsevier, 2016. 307 p.

SANTOS, P. R.; LANGOLOIS, T. **Compiladores: da teoria à prática**. Rio de Janeiro: LTC, 2018. 341 p.