Linguagens de Programação – Parte III
Paradigmas de Programação

Linguagens de Programação

O processo de programação é norteado por regras que especificam como o programador estruturará sua solução para um dado problema.

Este conjunto de regras que serve como exemplo geral ou modelo é denominado paradigma.

Dentre os paradigmas de programação existentes podemos destacar os paradigmas: imperativo, orientado a objetos, funcional e lógico.

UNIVASF

Linguagens de Programação

O paradigma de programação imperativo está diretamente atrelado à arquitetura básica dos computadores sobre os quais os programas eram executados.

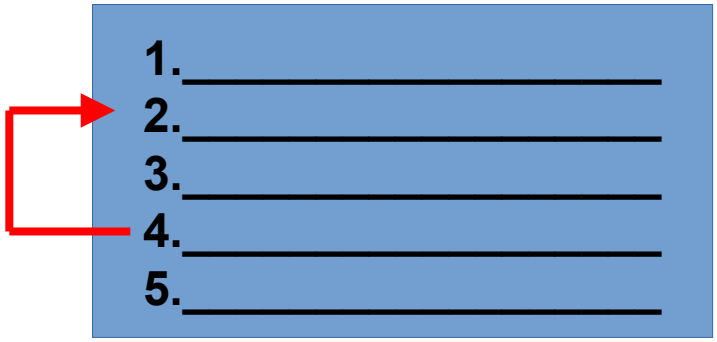
Boa parte dos computadores mais populares nos últimos anos foi projetada com base na arquitetura proposta por von Neumann.

UNIVASF

Linguagens de Programação

Devido à arquitetura de von Neumann os recursos centrais das linguagem imperativas são:

- as variáveis, as quais modelam as células de memória;
- as instruções de atribuição, baseadas na operações de canalização;
- e a forma iterativa de repetição, o método mais eficiente desta arquitetura.



Linguagens de Programação

Por fim, podemos associar o nome dado ao paradigma, ou seja, imperativo, como o tempo verbal imperativo.

Pois, o programador instruirá o computador da seguinte forma:

- faça isso;
- depois isso;
- depois aquilo…

Este paradigma destaca-se pela simplicidade, já que o ser humano, também baseia-se na ideia de ações e estados ao planejar as ações que irá executar.

Linguagens de Programação

Outro paradigma de programação que comentaremos é o funcional, onde o principal meio de fazer computações é aplicando funções a determinados parâmetros.

Ou seja, neste paradigma, basicamente trata-se a computação como uma avaliação de funções matemáticas.

Linguagens de Programação

Com a evolução da programação, o foco, anteriormente voltado para a arquitetura volta-se para os dados. Fomentando o surgimento do paradigma de programação orientado a objetos.

Onde não coloca-se em voga as funcionalidades do programa em desenvolvimento e sim as categorias de dados a serem manipulados (as classes que dão origem a objetos).

A metodologia orientada a objeto inicia-se com a abstração de dados.

UNIVASF

Linguagens de Programação

A abstração de dados trata-se de encapsular os dados por meio de procedimentos (métodos) que controlaram a acesso a estes.

Adiciona-se também o conceito de herança e outras características.

Herança é um conceito poderoso que potencializa a reutilização de código.

O desenvolvimento da programação orientada a objeto se deu em paralelo com a linguagem que suportou seus conceitos a Smaltalk. **(Hoje suportados em C++, Java, etc.)**

UNIVASF

Linguagens de Programação

O paradigma de programação lógico é um paradigma baseado em regras.

Em uma linguagem imperativa, um algoritmo é especificado com grandes detalhes, e a ordem de execução específica das instruções ou dos comandos deve ser incluída.

Em uma linguagem baseada em regras, estas são especificadas sem nenhuma ordem particular, e o sistema de implementação deve escolher uma ordem de execução que produza o resultado desejado.

UNIVASF

Linguagens de Programação

Essa abordagem ao desenvolvimento de software é radicalmente diferente daquelas utilizadas nos outros paradigmas vistos, e, evidentemente, exige um tipo de linguagem completamente diferente.

O Prolog é a mais popular linguagem de programação lógica utilizada.

UNIVASF

Linguagens de Programação – Parte IV
Especificação de Linguagens de Programação

MÉTODOS FORMAIS X INFORMAIS

O formalismo está associado à especificação de uma estrutura rígida e de um conjunto de regras coeso, que visam dotar o método formal da capacidade de nortear a construção de algoritmos.

Métodos informais por sua vez denotam a inexistência das características associadas ao formalismo. Um exemplo é linguagem natural.

Especificação de Linguagens de Programação

Associados aos métodos formais temos a sintaxe e a semântica.

Veremos agora alguns elemento relacionados com a sintaxe de uma linguagem de programação.

- conjunto de caracteres

  - são os constituintes mais simples de uma linguagem de programação.
  - a partir dele todas as sentenças – programas – são construídas, seguindo-se regras da gramática da linguagem. **Normalmente (caracteres alfabéticos, dígitos decimais e um conjunto variável de caracteres especiais).**

*** Os conceitos apresentados relacionados a especificação de linguagens de programação baseiam-se na apostila sobre Linguagens de Programação elaborada pelo prof. Celso Rodrigues da Fundação Universidade Federal do Rio Grande.**

UNIVASF

Especificação de Linguagens de Programação

- identificadores - sequências de caracteres usadas para se dar nomes aos objetos do usuário (ou mesmo pré-definidos) como variáveis, constantes, arquivos, tipos de dados, etc.
- operadores - símbolos especiais para certas operações (do tipo função).

Em geral recaem numa das classes:

- Aritméticos — Ex.: + - * /
- Relacionais — Ex.: > < = <= >= !=
- Lógicos — Ex.: && || ! (AND OR NOT)
- Não-numéricos — Ex.: ? ou ?:

UNIVASF

Especificação de Linguagens de Programação

- palavras-chaves ou palavras-reservadas - são as partes fixas de uma construção sintática qualquer, na forma de sequência de caracteres. Estas não podem ser usadas como identificadores.

  Exemplos: IF ELSE FOR

- delimitadores - elementos usados para marcar o início ou o fim de uma construção sintática. Podem ser palavras-chaves, mas em geral são caracteres especiais.

  Exemplos: ; {

UNIVASF

- chaves sintáticas - pares de delimitadores conjugados, ou seja: um sempre aparece acompanhado do outro.

Servem basicamente para facilitar o processo de análise sintática por parte do processador da linguagem, uma vez que marcam explicitamente o alcance de uma construção, diminuindo as possibilidades de ocorrência de ambiguidade no significado das sentenças que compões os programas.

Consequentemente, cada vez mais se reconhece sua influência positiva na legibilidade.

UNIVASF

Exemplos de chaves sintáticas:

- na linguagem C: { ... }
- em Pascal: begin ... end
  repeat ... until

- expressões - combinações de identificadores e operadores especificando uma operação mais complexa. Admitem formas diversas de construção, conforme sejam os operadores.

Dois aspectos importantes dos operadores influem no formato das expressões: aridade e o modo.

UNIVASF

Especificação de Linguagens de Programação

- a aridade - é o número de operandos (argumentos) de um operador.

Exemplo: + (soma) A1 + A2

\- (troca de sinal) -X

- modo ou posição relativa: os operadores podem estar num ou noutro modo, conforme sua posição em relação a seus operandos. Tais modos são:
  - infixado = operador entre os operandos
  - pré-fixado = operador antes dos operandos
  - pós-fixado = operador depois dos operandos

Nos exemplos abaixo, as expressões estão escritas nos três modos:

MODO	EXPRESSÃO 1	EXPRESSÃO 2
infixado	**A1 + A2**	**(A + B) * C**
pré-fixado	**+ A1 A2**	*** + A B C**
pós-fixado	**A1 A2 +**	**A B + C ***

Uma hipótese adicional é ter um operador **intermixado**. Nesse caso, o operador é composto de mais de um símbolo, devendo se misturar aos operandos. Um exemplo disto é o operador ?: da linguagem C.

• comandos - combinações de expressões e palavras-chaves.

Estas palavras-chaves representam operadores especiais que agem sobre o estado interno da máquina. (principalmente modificando valores de variáveis).

Especificação de Linguagens de Programação

Os comandos são as construções mais centrais das linguagens de programação tradicionais (imperativas).

Sua sintaxe influi na redigibilidade e na legibilidade da linguagem.

- declarações - combinações de palavras-chaves e identificadores.

As declarações visam estabelecer conjuntos de informações necessárias para a execução do programa.

Elas produzem o que se denomina amarração (ou associação) entre as diversas entidades do programa (variáveis, constantes, funções, tipos, etc.) e seus atributos, entre estes o seu nome (representado por um identificador).

Exemplo na linguagem C: **int x=0;**

Especificação de Linguagens de Programação

- comentários - trechos explicitamente marcados pelo programador para serem desconsiderados pelo processador da linguagem.

Os comentários são importantes para a documentação dos programas.

Exemplo de um comentário em C:

/* Observação */

Em geral comentários são delimitados por chaves sintáticas.

Linguagens de Programação - Parte V
Especificação de Linguagens de Programação (Gramática)

ESPECIFICAÇÃO SINTÁTICA: GRAMÁTICAS

Uma gramática é um conjunto de regras que especificam o formato das sentenças de uma linguagem, isto é, sua sintaxe.

Uma linguagem usada para se descrever outra linguagem é chamada metalinguagem.

Para se descrever linguagens de programação várias metalinguagens podem ser usadas, desde a linguagem natural até formalismos matemáticos rigorosos, passando por métodos gráficos.

Por exemplo, a regra

<atribuição> : := <receptor> := <expressão>

descreve o formato de um comando de atribuição em Pascal: à esquerda do símbolo ':=' deve estar uma construção caracterizada como 'receptor', e à direita uma 'expressão'.

O significado (semântica) disso pode ser dado informalmente, em linguagem natural: "substituir o valor do receptor pelo valor resultante da avaliação da expressão".

Especificação de Linguagens de Programação

Frisando, a especificação formal é necessária para se evitar ambiguidades na descrição, o que é imprescindível para o trabalho de implementação de uma linguagem.

BNF (Backus-Naur Form)

foi criada para a descrição sintática da linguagem ALGOL e tornou-se uma metalinguagem padrão. Com BNF pode-se especificar gramáticas de linguagens formais ou naturais (ainda que com restrições).

UNIVASF

Especificação de Linguagens de Programação

Uma gramática é um objeto formal G = (N, T, P, S), onde:

- **N** é o conjunto de símbolos não-terminais (ou metavariáveis). Estes representam classes ou categorias de construções sintáticas intermediárias na definição das sentenças da linguagem. Em BNF são escritos entre parêntese angulares, como por exemplo: <S>, <termo>, <comando>, etc.;

Especificação de Linguagens de Programação

Uma gramática é um objeto formal G = (N, T, P, S), onde

- **T** é o conjunto de símbolos terminais da linguagem, i.e., básicos, considerados atômicos, os quais podem se combinar para formar todas as construções válidas. Podem ser caracteres, ou sequências de caracteres, como, por exemplo: A, 1, for, +, [, procedure, etc.;

- **P** é o conjunto de produções. Uma produção é uma regra que define uma possível estrutura de um não-terminal.

UNIVASF

Especificação de Linguagens de Programação

Uma gramática é um objeto formal G = (N, T, P, S), onde

Em BNF, uma produção tem a seguinte forma:

<a> ::= <b>

onde <a> é um símbolo não-terminal e <b> é uma sequencia de símbolos terminais ou não-terminais, que vale por uma descrição da estrutura sintática da estrutura <a>;

- **S** é o chamado símbolo não-terminal inicial, que representa uma sentença.

UNIVASF

Especificação de Linguagens de Programação

Uma gramática pode ser usada para derivação ou redução de sentenças.

- **Derivação** é o processo pelo qual se obtém uma sentença.

Isto se faz através da substituição sucessiva de cada não-terminal pela sua definição (lado direito da produção correspondente), a partir da produção inicial (aquela que tem o símbolo **S** à esquerda), até se obter uma sequência que contenha apenas terminais, ou seja, uma sentença.

UNIVASF

Especificação de Linguagens de Programação

- **Redução** e o processo pelo qual se verifica a validade de uma sentença, i.e., se ela pertence à linguagem.

Consegue-se isso pela aplicação inversa das produções, escolhendo-se criteriosamente trechos da sentença a serem substituídos pelos correspondentes não- terminais, até se atingir o símbolo inicial (nesse caso dito símbolo objetivo). Se isto não for alcançado, a sequência dada não é uma sentença da linguagem.

UNIVASF

Especificação de Linguagens de Programação

Exemplo: uma gramática simples em BNF (especifica o formato dos identificadores em uma certa linguagem):

G1 = (N = { <id>, <letra>, <digito>, <resto> },

T = { A, B, C, 0, 1 },

P = { <id> ::= <letra> **"1"**

<id> ::= <letra><resto> **"2"**

<letra> ::= A **"3"**

<letra> ::= B **"4"**

<letra> ::= C **"5"**

<resto> ::= <letra> **"6"**

<resto> ::= <digito> **"7"**

<resto> ::= <letra><r **"8"**

<resto> ::= <digito>< **"9"**

<digito> ::= 0 | 1 }, **"10"**

S = <id>)

UNIVASF

Especificação de Linguagens de Programação

Obtendo (derivando) sentença dessa linguagem:

G1 = (N = { <id>, <letra>, <digito>, <resto> },
T = { A, B, C, 0, 1 },
P = { <id> ::= <letra> "1"
<id> ::= <letra><resto> "2"
<letra> ::= A "3"
<letra> ::= B "4"
<letra> ::= C "5"
<resto> ::= <letra> "6"
<resto> ::= <digito> "7"
<resto> ::= <letra><resto> "8"
<resto> ::= <digito><resto> "9"
<digito> ::= 0 | 1 }, "10"
S = <id>)

<id>

Regra 2: <id> → <letra><resto>

Regra 9: <letra><resto> → <letra><digito><resto>

Regra 7: <letra><digito><resto> → <letra><digito><digito>

Regra 3: <letra><digito><digito>→ A<digito><digito>

Regra 10: A<digito><digito>→ A1<digito>

Regra 10: A1<digito>→A10

Especificação de Linguagens de Programação

Exemplo: uma gramática em BNF de um subconjunto (imperfeito) do português

```
G2 = ( N = { <sentença>, <SN>, <SV>, <artigo>, <nome>, <verbo>,
             <adverbio>  },
     T = {  o, a, estudante, aluna, estuda, copia, materia, muito,
            todo, tempo, furiosamente, .  },
     P = {<sentença> ::= <SN> <SV> .
          <SN> ::= <artigo> <nome>
          <SV> ::= <verbo>
          <SV> ::= <verbo> <adverbio>
          <SV> ::= <verbo> <SN>
          <SV> ::= <verbo> <adverbio> <SN>
          <artigo> ::= o
          <artigo> ::= a
          <nome> ::= estudante
          <nome> ::= aluna
          <nome> ::= materia
          <verbo> ::= estuda
          <verbo> ::= copia
          <adverbio> ::= muito
          <adverbio> ::= todo o tempo
          <adverbio> ::= furiosamente                    },
     S = <sentença> )
```

A sentença: <u>O estudante estuda muito.</u> Pertence à linguagem acima? Demonstre...