1. PARADIGMA IMPERATIVO

A arquitetura dos computadores teve um efeito crucial no design das linguagens de programação. A maioria das linguagens populares nos últimos 35 anos foram projetadas em torno da arquitetura de computadores conhecida como arquitetura de von Neumann. Essas linguagens são chamadas de imperativas. Em um computador com essa arquitetura, tanto os dados quanto os programas são armazenados na mesma memória, e a CPU (Unidade Central de Processamento), que realmente executa as instruções, é separada da memória. Consequentemente, instruções e dados devem ser transportados da memória para a CPU, e os resultados das operações realizadas na CPU devem ser devolvidos à memória.

Devido à arquitetura de von Neumann, as características centrais das linguagens imperativas são: variáveis, que modelam as células de memória; comandos de atribuição, que são baseados nas operações de transferência de dados e instruções; a execução sequencial de instruções; e a forma iterativa de repetição, que é o método mais eficiente nessa arquitetura. Os operandos das expressões são passados da memória para a CPU, e o resultado da expressão é passado de volta para a célula de memória representada pelo lado esquerdo do comando de atribuição. A iteração é rápida em computadores com esse tipo de arquitetura porque as instruções são armazenadas em células adjacentes da memória. Essa eficiência desencoraja o uso da recursão para repetição.

É interessante notar que, às vezes, as linguagens de programação imperativas também são chamadas de procedurais, mas isso não tem relação direta com o conceito de procedimento. Neste capítulo, o paradigma imperativo de linguagens de programação, encontrado em linguagens como Fortran, Cobol, Basic, Pascal, Modula-2, C e Ada, será apresentado. Inicialmente, na seção 1.1, será feita uma introdução ao assunto, bem como exemplos de linguagens que utilizam esse paradigma, serão descritas.

1.1 Introdução

As linguagens imperativas são caracterizadas por três conceitos fundamentais: variáveis, atribuições e sequência. O estado de um programa imperativo é mantido em variáveis associadas a localizações de memória específicas, que correspondem a um endereço e um valor de armazenamento. O valor de uma variável pode ser acessado direta ou indiretamente e alterado por meio de um comando de atribuição. O comando de atribuição introduz uma dependência de ordem no programa: o valor de uma variável muda antes e depois de um comando de atribuição. Além disso, o significado (efeito) de um programa depende da ordem na qual os comandos são escritos e executados.

As funções em linguagens de programação imperativas são descritas como algoritmos que especificam como processar um intervalo de valores a partir de um valor de domínio, seguindo uma série de passos prescritos. A repetição, ou laço, é usada extensivamente para processar os valores desejados. Laços são utilizados para percorrer uma sequência de localizações de memória, como vetores, ou para acumular um valor em uma variável específica. Assim, devido às suas características, as linguagens imperativas são frequentemente chamadas de "baseadas em comandos" ou "orientadas a atribuições" [GHE 97].

Através de exemplos, são apresentadas as principais características das linguagens de programação imperativas. As linguagens escolhidas para realização de uma análise comparativa são:

- Fortran,
- Pascal,
- C,
- Ada.

Em cada uma destas linguagens são abordados os seguintes tópicos:

- Valores e tipo;
- Expressões;
- Comandos e seqüências (= estruturas de controle);
- Declarações;
- Procedimentos e funções.

	FORTRAN	**PASCAL**
- valores e tipos - expressões	- tipos: integer, real, double precision, complex, logical; vetor: dimension <nome> (dim1, dim2, dim3), real, integer - constantes lógicas: .true., .false. - operadores: **, *, /, +, -, .ge., .gt., .le., .lt., .eq., .ne., .not., .and., .or.	- tipos: simples: boolean, integer, char, real; estruturados: array, file, record, set - expressões: boolean: and, or, not, xor integer: +, -, *, div, mod, =, >=, <>. abs, sqr, trunc, round string: var a: string; a = 'abc'; file: type arq = file of integer;

- comandos
e seqüências
	if (<exp>) then
	...
	end if
	if (<exp>)
	...
	else if ( ) then
	...
	else
	...
	end if
	- comandos I/O: open, close, read, write, print, rewind, endfile
	- goto, continue, pause, stop
		- comandos simples: write, writeln, read, clrscr, gotoxy, delay, readkey, upcase
		if-then, if-then-else
		case <exp> of
		case <op1>:
		case <op2>:
		else
		end
		for x := <inic> to|downto <fim> do
		while <cond> do
		begin
		...
		end;

- declarações - procedimentos e funções	- declaração var.: <tipo> <id> - funções: function <id> (<parâm.>) - proc.: procedure <id> (<p.>)	- declaração var.: var <id>: <tipo>; - procedimentos e funções: procedure <id> (<parâm.>); function <id> (<parâm.>): <tipo>;
	versões anteriores a FORTRAN 90: somente letras maiúsculas; outras versões traduzem para maiúsculas durante a compilação	não é case sensitive

	C	ADA
- valores e tipos - expressões	- tipos: char, int, float, double, struct, union - operadores: -, +, *, /x, %, ++, >, >=, =, !=, &&, ||, !, & bit a bit: &, |, ^, ~, >>, <<	- tipos: array: <id> array(x..y) of <tipo> integer, natural, positive, character, boolean, string: type string is array(x..y) of character; float - operadores: =, /=, >, >=, +, -, abs, **, and, or, xor, not

- comandos
e seqüências
	if (<exp.>) <comandos>
	else <comandos>
	for (inic; cond; incremento)
	
	switch (<exp>){
	case <op1>: ...; break;
	case <op2>: ...; break;
	default: ...;
	}
	while (<cond.>){ ....;}
	do
	....
	while (<cond.>);
	return <exp>, goto <tótulo>, break
		if <cond> then case <exp> is
		elsif <cont> then <alt. 1>
		else <alt. 2>
		end if; end case;
		
		when <lista escolha> => <com.>
		when <others> => <comandos>
		
		while <cond.> loop
		<comandos>
		end loop;
		for <id> in <interv> loop
		<comandos>
		end loop

- declarações - procedimentos e funções	- declaração var.: <tipo> <lista variáveis>; - constantes: const <tipo> <id> = <valor>; - funções: <tipo> <id> (<parâm>)	- declaração var.: <id>: <tipo>; - procedimentos: procedure <id> (<parâm>) is - funções: function <id> (<p.>) return <tipo>
	é case sensitive	