As linguagens de programação abrigam geralmente mais de um paradigma, mas possuem um modelo principal que influenciará mais fortemente o processo de criação de programas, colocando questões importantes.

Linguagens Imperativas

- As linguagens imperativas são orientadas a ações, onde a computação é vista como uma seqüência de instruções que manipulam valores de variáveis (leitura e atribuição).
- Foram criadas principalmente pela influência da arquitetura de computadores preponderante *Von Neumann*, onde programas e dados são armazenados na mesma memória. Instruções e dados são transmitidos da CPU para a memória, e vice-versa. Resultados das operações executadas na CPU são retornadas para a memória.
- Seus conceitos principais são as **variáveis**, que representam simbolicamente células (ou posições) de memória, **declarações de atribuição**
  (Exemplo do Pascal: *x := 10*), baseadas nas operações de transmissão de dados, e a forma iterativa de **declarações de repetição** (Exemplo do C: *while*).
- Operandos em expressões são enviados da memória para a CPU, e o resultado da avaliação dessas expressões é transferido da CPU para a memória, representada pela variáveis do lado esquerdo de uma declaração de atribuição.
- Repetição é feita por iteração, porque as instruções em uma arquitetura *Von Neumann* são armazenadas em posições adjacentes de memória, tornando a iteração mais eficiente, mas desencorajando o uso de recursão para repetição.
- Exemplos de linguagens imperativas: FORTRAN, BASIC, COBOL, Pascal, C, Python, ALGOL, Modula.

- A formulação do algoritmo é fundamental e a descrição de dados é incidental
  .
  - Consistem de alterações de valores através de operações baseadas em atribuições e um fluxo de controle seqüencial, condicional ou iterativo

Linguagens Imperativas Não Estruturadas

- FORTRAN foi uma das primeiras linguagem de alto nível imperativas, destinada
  inicialmente ao desenvolvimento de aplicações científicas. As versões originais de FORTRAN possuíam a instrução *goto* necessária para determinar a repetição e a seleção de execução de instruções, o que dificultava bastante a leitura e o acompanhamento da execução de um programa escrito em FORTRAN. Assembly e BASIC são outros exemplos de linguagens imperativas que possuem o conceito de *goto*. Uso de goto geralmente leva ao que chamam na literatura de código *spaguetti*.
- Exemplo de código com o uso de instrução *goto*, baseado na sintaxe do Pascal:

```
read(x);
2: if x = 0 then goto 8;
writeln(x);
4: read(next);
if next = x then goto 4;
x := next;
goto 2;
8: ...;
```

Linguagens Imperativas Estruturadas

- Linguagens estruturadas surgiram com o objetivo de facilitar a leitura e acompanhamento da execução de algoritmos.
- Normalmente linguagens estruturadas não fazem uso de comando *goto*.
- Instruções são agrupadas em blocos, os quais podem ser considerados como unidades de programa, abstraindo-se das suas estruturas internas.
- Blocos de instruções podem ser selecionados para execução através de declarações de seleção como *if...else*, ou repetidamente executados através de declarações de repetição como *while*.
- Linguagens estruturadas **procedurais** permitem a criação de procedimentos (e funções), que são blocos de instruções, que formam os elementos básicos de construção de programas. Procedimentos criam um nível de abstração, onde não é necessário conhecer todos os passos de execução de um procedimento, apenas qual a sua função e quais os seus pré-requisitos para que execute de acordo com o esperado.

- Linguagens estruturadas **modulares** criam um outro mecanismo de abstração, o módulo, que normalmente é composto de definições de variáveis e procedimentos, que são agrupados de acordo com critérios específicos.
- Exemplo de código estruturado, baseado na sintaxe do Pascal, correspondente ao mesmo exemplo da seção anterior:

```
read(x);
while x <> 0 do begin
writeln(x);
repeat
read(next);
until next <> x;
x := next;
end;
```

1.3.2. Linguagens Funcionais

- Utilizam como bloco básico de construção de sistema o conceito de **funções**.
- Sistemas são construídos através da definição, composição e definição de funções.
- Linguagens funcionais são baseadas em funções matemáticas e a notação funcional Lambda de Church.
- Formas funcionais – recebem como parâmetros funções e produzem funções como resultado.

- Funções devem ser identificadas e reunidas, de forma que a sua composição, em algum momento, forneça a resposta desejada
- Consistem de substituições de parâmetros em funções que podem ser aplicadas sobre funções
- Estaticamente tipadas, tipagem forte
- uso intensivo de listas
- uso intensivo de recursividade para processamento repetitivo

- A linguagem LISP introduziu o paradigma.
- LISP começou como puramente funcional, depois incorporou algumas características de linguagens mperativas para aumentar a sua eficiência.
- Algumas linguagens funcionais:
- *Scheme* – derivada de LISP
- *Common LISP* – buscando padronização de LISP
- *CLOS (Common LISP Object System)* – dialeto de LISP que incorpora elementos de linguagens orientadas a objetos
- *ML* – linguagem funcional fortemente tipada
- *Miranda* – baseada em *ML*, mas é puramente funcional
- Funções matemáticas?
- *cube*(*x*) ” *x* * *x* * *x*
- **Expressão Lambda (Church) -** definição de funções sem nome. Exemplo: *l (x) x* * *x** *x* – usa-se: (*l (x) x* * *x** *x*)(*2*)
-
- **Composição de funções**: operador º.
- Exemplo de composição: *h ” f º g*. Se *f(x) ” x + 2* e *g(x) ” 3 * x*, então *h(x) ” f(g(x))* ou *h(x) ” (3 * x) + 2*
- **Construção**: aplica um mesmo argumento a uma lista de funções, produzindo uma lista de resultados. A lista de funções é delimitada por colchetes. Considerando as funções: *g(x) ” x * x, h(x) ” 2 * x, i(x) ” x/2*, então [*g, h, i*](*4*) produz (*16, 8, 2*)
- **Aplicar a Todos (□)**: recebe como argumento uma única função. Se aplicada a uma lista de valores, aplica a função argumento aos valores da lista, criando uma lista com os resultados coletados. Considerando *h(x) ” x * x*, então □(*h*, (*2, 3, 4*)) produz (*4, 9, 16*)
- LISP e outras linguagens funcionais baseiam-se no processamento de listas. Internamente, estas listas são normalmente implementadas como estruturas de listas ligadas. Exemplo de listas: *(pai mãe (filho1 filho2)), (10 (101 (1011 1012)) 20)* – Ilustração.Exemplo de chamada de função em LISP: *(+ 5 7), (- 24 (* 4 3))*. + e

– são funções pré-definidas. Exemplo de definição de função em LISP: (DEFINE (square number) (* number number)).

1.3.3. Linguagens Lógicas

- Linguagens de programação lógicas (ou declarativas) usam notações formais lógicas para a especificação de processos computacionais.
- Cálculo de predicado é a notação usada atualmente em linguagens de programação lógicas.
- Programação não é procedural. Computação é baseadas em fatos, que podem ser relações (associações) entre coisas, e regras, que produzem fatos deduzidos a partir de outros.
- Prolog é talvez a única linguagem lógica mais difundida e usada popularmente.
- Prolog foi desenvolvida em 1972 para processamento de linguagem natural, na França. O nome vem de *PROgrammation en LOGique*.
- Programas Prolog consistem em dois tipos de declaração, que são **fatos** e **regras**.

- Fatos conhecidos a respeito de um determinado problema devem ser reunidos e estabelecidas regras que permitam derivar novos fatos

- Consistem de unificações de átomos e termos em cláusulas e mecanismos de retrocesso
- dinamicamente tipadas, tipagem fraca
- uso intensivo de listas
- uso intensivo de recursividade para processamento repetitivo

Exemplos de fatos:
parent (joanne, jake)
parent (vern, joanne)
Exemplo de regra:
grandparent (X, Z) :- parent (X, Y), parent (Y, Z)
Pode-se aplicar ao interpretador Prolog perguntas:
? parent (vern, joanne) – irá resultar em verdadeiro
? grandparent (vern, jake) – irá resultar em verdadeiro

1.3.4. Linguagens Orientadas a Objetos

- O paradigma da orientação a objetos considera **objetos** e **classes** como blocos básicos de construção de um sistema. Sistemas são vistos como coleções de objetos que se comunicam, enviando mensagens, colaborando para dar o comportamento global dos sistemas.
- Uma classe determina o comportamento e a estrutura de objetos similares (Célula, Mamífero, Réptil, Homo Sapiens).
- Pode-se dizer que a idéia da orientação a objetos foi importada a partir da observação do comportamento de sistemas complexos do mundo real (animais, plantas, máquinas, ou até mesmo empresas), onde cada objeto possui um determinado conjunto de responsabilidades dentro de um sistema, que normalmente estão relacionadas com a manutenção de conhecimento e com ações que devem executar.
- Por exemplo, pode-se definir um objeto relógio com as responsabilidades de manter horário atual e horário de despertar (conhecimento), como também de despertar, permitir acertar hora, etc.. Estas responsabilidades podem ser executadas através do envio de mensagens, que é o mecanismo básicos de comunicação entre objetos.
- Sistemas orientados a objetos apresentam hierarquias de classes, onde classes mais genéricas são especializadas em classes mais específicas (ex.: mamíferos -> felinos, primatas, canídeos, ...), e hierarquias de objetos, onde objetos mais genéricos são compostos por objetos mais específicos (ex.: planta é composta de órgãos, como o galho, que é composto de células, ..., que são compostas de átomos, que são compostos de prótons, neutrons, ..., que são compostos de quarks...).
- De uma maneira mais simplista e aproximando de linguagens imperativas, pode-se dizer que objetos encapsulam dados, assim como estruturas de dados em linguagens imperativas, que mantém conhecimento, e que também encapsulam procedimentos, que são as ações executadas na troca de mensagens.

os objetos da computação são o foco principal e as ações descrevem o comportamento desejado destes objetos .
projeto de classes, seus relacionamentos, objetos e suas interações.
reutilização: bibliotecas compostas de classes/interfaces que podem ser adaptadas por extensão de propriedades.

- SIMULA 67 foi a primeira linguagem a incorporar estes conceitos, desenvolvida especialmente para a criação de aplicações de simulação...
- Outra linguagem OO importante e pioneira é Smalltalk, originada em uma dissertação de doutorado de Alan Kay em 1969.
- Outras linguagens OO combinam características de linguagens imperativas e orientadas a objetos: C++, Eiffel, Java, Ada...

1.3.5. Paralelismo

A agenda de tarefas que contribuem para o resultado final deve ser cuidadosamente planejada

.

Modelos de Linguagens

imperativo

- Linguagens expressam seqüências de comandos que realizam transformações sobre dados.
- base: máquina de von Neumann
- orientadas a procedimentos /

orientadas a objetos

declarativo

- Linguagens que não possuem o conceito de seqüências de comandos
- linguagens funcionais, baseadas em funções
- linguagens lógicas, baseada em axiomas lógicos