Linguagens de Programação

Textos de apoio

- Capítulos 15 do livro "Informática - Conceitos e Aplicações" do Marcelo Marçula;

O que é Linguagem de Programação

É um conjunto de palavras (vocabulário) e um conjunto de regras gramaticais (para relacionar essas palavras) que serve para instruir o sistema de computação a realizar terefas específicas, e com isso, criar **Programas**.

Cada linguagem tem o seu conjunto de palavras-chave sintaxe.

Programas

Para se criar programas em primeiro lugar é necessário entender o problema que se quer resolver.

Em seguida deve-se criar a lógica dessa solução passo a passo.

Depois deve-se escrever de forma organizada essa lógica numa linguagem humana (pseudo código).

Programas

- Após escrito o pseudo-código o programador deve escrever os comandos na linguagem de programação , criando o código fonte.
- O codigo-fonte, que é em linguagem humana, deve ser traduzido pelo Compilador para uma linguagem compreensível pela CPU, gerando a linguagem de máquina.

Essa transformação pode ocorrer por dois meios: Compilação ou Interpretação.

Compilação

- Portanto, a compilação é um processo que transforma código-fonte em programa executável pela CPU, de modo a criar programas autônomos, ou seja, que não necessitem da linguagem de programação para serem executados.
- Para isso são necessárias algumas etapas:
- O programador escreve os comandos que realizarão as tarefas desejadas na LP, usando palavras-chaves e sintaxes próprias;

Compilação

- Esses comandos, de uma forma geral, são escritos em uma linguagem mais próxima da linguagem humana do que da linguagem de máquina - denominada Linguagem de Alto Nível.
- Esse conjunto de comandos denomina-se Código-Fonte.
- Esse código-fonte passa por um processo denominado Compilação, onde a LP utiliza um de seus módulos chamado Compilador para transformar os comandos do código-fonte em comandos de linguagem de máquina, criando o Código-Objeto.

Compilação

- Durante a compilação a LP realiza análise léxica (letras), sintática (organização da palavras) e semântica (coerencia de tipos de dados, multiplicar a por 5!!) para detectar se os comandos escritos pelo programador utilizaram as palavras-chave válidas e se essas palavras estão dispostas na ordem correta.

Compilação

- O código-fonte de um programa pode ser formado por apenas um conjunto de comandos, mas a maioria dos programas é formado por vários desses conjuntos interligados (rotinas).
- Isso possibilita que um programa com uma quantidade de funções muito grande possa ser desenvolvido por um grupo de pessoas, cada um desenvolvendo uma parte dele.
- E por fim todos são integrados, gerando o produto final.

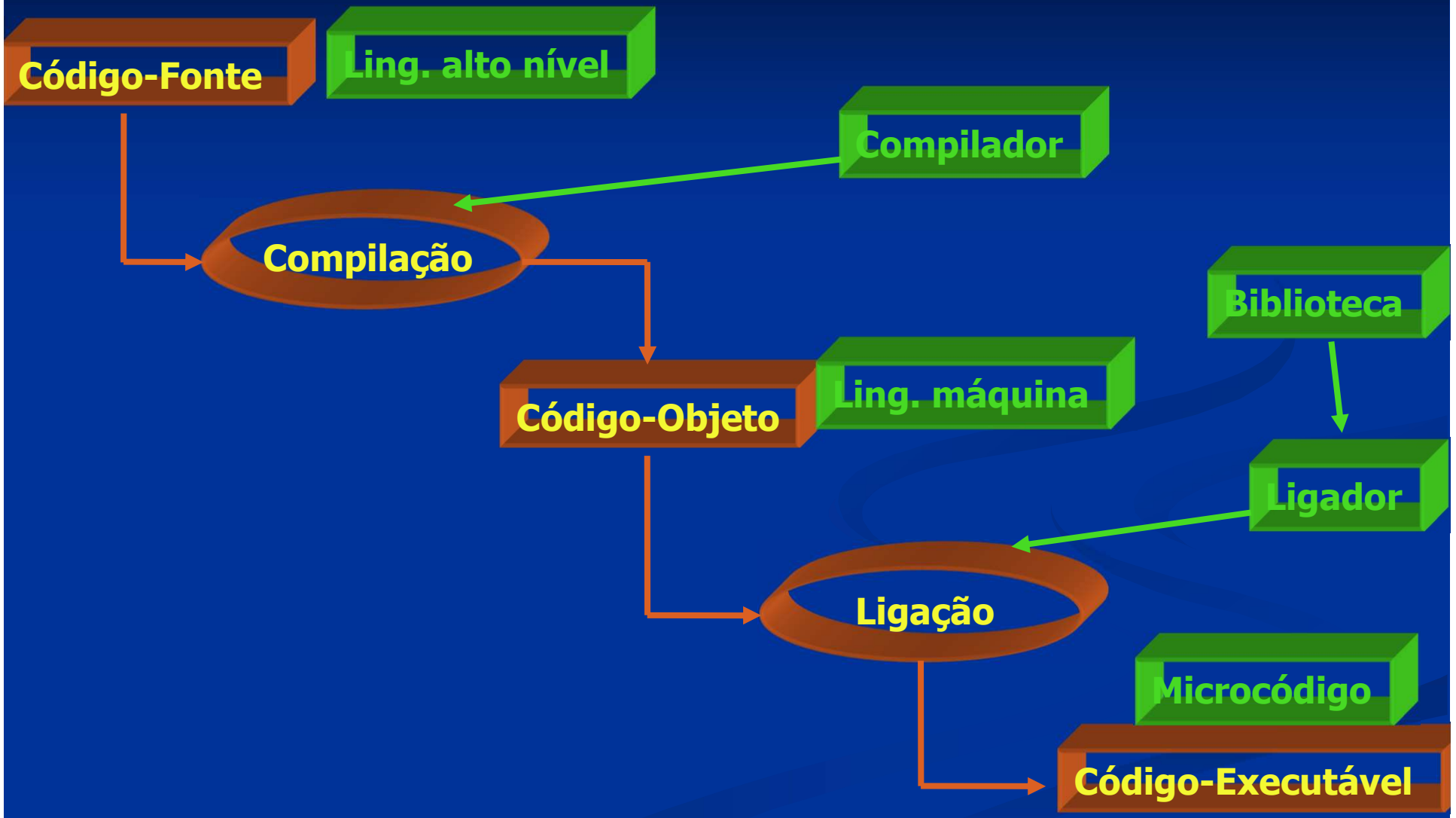
Compilação

- Também pode ocorrer, e ocorre na maioria das vezes, que programas utilizem códigos ou dados previamente desenvolvidos e armazenados em arquivos conhecidos como Bibliotecas.
- Quando isso ocorre o processo de compilação deve agrupar todas essas partes, ligando-as de forma adequada. Isso é feito por um módulo da LP denominado de Linkeditor (ligador).

Compilação

- O produto final do processo de Linkedição ou ligação é o Código-Executável, um programa autônomo que pode ser portável e executado em outros computadores sem a presença da LP, mas com sistema operacional compatível.
- Exemplos de LP Compiladas: FORTRAN, Delphi, Pascal, Visual Basic, C, C++, etc

Processo de geração de programas



Interpretação

Consiste em executar o código-fonte diretamente por meio de um módulo da LP denominado Interpretador.

Nesse processo não existe código-objeto nem código-executável.

Neste caso há a necessidade da presença do ambiente da LP no sistema de computação onde o programa será executado. Os programas não são autônomos.

Interpretação

Um dos problemas da Interpretação é que em determinados comandos, por exemplo, comandos de repetição (While, Repeat, For) a LP vai interpretar o comando quantas vezes ele for repetido, tronando-a mais lenta em relação as LPs compiladas.

O processo de Interpretação é utilizado em algumas LPs em conjunto com a compilação durante o desenvolvimento, quando o programador deseja executar o código-fonte pra detectar problemas.

LPs Interpretadas: LISP, PHP, Python, Javascript

Escolha da LP

A escolha da LP depende de alguns fatores como:

- Área de aplicação do programa (cada LP é voltada para criação de determinados tipos de aplicação);
- Complexidade do programa a ser criado e da estrutura dos dados que serão gerados por ele (a LP pode ajudar a diminuir a complexidade);
- Tipo de sistema de computação no qual o programa será executado (interação entre código gerado e CPU);
- Desempenho desejado (depende da LP);
- Conhecimento da equipe de programadores;
- Disponibilidade da LP.

Gerações de Linguagens

As LPs são divididas em gerações, mas não apenas em relação a época em que foram criadas, mas também com a proximidade que a LP tem com a linguagem natural humana.

Essa proximidade é conhecida como abstração, pois cada vez mais o programador não precisa se preocupar com detalhes ligados ao sistema (0 e 1).

As LPs que necessitam de comandos escritos mais próximos da linguagem de máquina, são conhecidas como Linguagem de Baixo Nível e LPs que utilizam palavras em idioma normal são conhecidas como Linguagem de Alto Nível.

Para facilitar o processo de programação, a cada novo lançamento, as empresas que criam LPs procuram aproximar a linguagem de programação da linguagem natural humana.

Gerações de Linguagens

Linguagens de 1ª Geração
Linguagens de 2ª Geração
Linguagens de 3ª Geração (Estruturadas)
Linguagens de 4ª Geração

Linguagens de 1ª Geração

Linguagem de mais baixo de nível, o programador necessita escrever comandos praticamente no nível da máquina, cada comando tem correspondência direta com um comando em micro código da CPU.

É usada quando outras linguagens não conseguem cumprir requisitos de velocidade de execução ou utilização de memória.

Ex. Assembly

Exemplo

```
.MODEL SMALL ; modelo de memória
.STACK ; espaço de memória para instruções do programa na
         pilha
.CODE ; as linhas seguintes são instruções do programa
    mov ah,01h ; move o valor 01h para o registrador ah
    mov cx,07h ; move o valor 07h para o registrador cx
    int 10h ; interrupção 10h
    mov ah,4ch ; move o valor 4ch para o registrador ah
    int 21h ; interrupção 21h
END ; finaliza o código do programa
```

Exemplo: comparação entre duas variáveis com o objetivo de saber qual é a maior

```
1 – movwf var_1 ; move o valor de var_1 para W
2 –    subwf var_2, 0 ; subtrai o valor de var_2 de W
3 –    btfss STATUS, C ; verifica o estado do bit CARRY
4 –    goto v1_maior ; vai para valor menor
5 – v2_maior: ; var_2 > var_1
6 –       return ; sai da rotina
7 – v1_maior: ;  trata o caso de var_2 < var_1
```

Linguagens de 2ª Geração

São LPs que apresentam um avanço em relação ao Assembly. Os comandos são dados através de palavras utilizadas no dia-a-dia, normalmente verbos (em inglês: Read, Write, Do, If, While, etc).

FORTRAN (FORmula TRANslation): primeira LP. Ideal para aplicações matemáticas e de engenharia, não adequada para criação de SW básico, tempo real ou embarcados.

COBOL (COmmon Business Oriented Language): primeira LP voltada para aplicações comerciais. Ainda muito usada, principalmente em ambientes comerciais onde se utilizam computadores de grande porte;

Linguagens de 2ª Geração

BASIC (Beginners All-Purpose Symbolic Instruction Code): LP desenvolvida para o ensino de programação, renasceu nos anos 70 com os computadores pessoais

ALGOL (ALGOrithmic Language): LP voltada para aplicações cientificas, muito usada na Europa, compiladores caros, precursora das LP de 3ª geração.

Exemplo 2ª geração - FORTRAN

```
PROGRAM BASKHARA
REAL A,B,C, DELTA, X1,X2, RE, IM
PRINT *, "Este programa resolve uma equação de 2o.grau"
PRINT *, "no formato: a*x**2 + b*x + c = 0"
PRINT 10, "Digite a,b,c: "
10 FORMAT(A,1X,$)
20 READ(*,*,ERR=20) A,B,C
DELTA=B*B-4.*A*C
IF (DELTA.GT.0) THEN ! (DUAS RAIZES REAIS)
   X1=(-B-SQRT(DELTA))/(2.*A)
   X2=(-B+SQRT(DELTA))/(2.*A)
   PRINT *, "RAIZES: X1=",X1
   PRINT *, " X2=",X2
```

Continuação Baskhara

```
ELSE IF (DELTA.EQ.0) THEN ! (DUAS RAIZES REAIS
   IGUAIS)
   X1=-B/(2.*A)
   X2=X1
   PRINT *, "RAIZES: X1=X2=",X1
 ELSE ! (DUAS RAIZES COMPLEXAS)
   RE=-B/(2.*A) IM=SQRT(-DELTA)/(2.*A)
   PRINT *, "RAIZES COMPLEXAS: X1=",RE," -",IM,"i"
   PRINT *, " X2=",RE," +",IM,"i"
ENDIF
END
```

Linguagens de 3ª Geração (Estruturada)

As LPs de terceira geração seguem de alguma forma as técnicas de programação estruturada e dividem-se em:

- Linguagem de alto nível de uso geral;
- Linguagens orientadas a objeto;
- Linguagens especializadas.

Linguagem de alto nível de uso geral

São LPs baseadas no ALGOL e bastante utilizadas, como: Pascal, PL/1, C, ADA, etc

Pascal: primeira LP estruturada, foi criada pra ensinar essa técnica. Descendente direta do Algol, utilizada para criar aplicações científicas, de engenharia e SW básico.

PL/1: criada pela IBM, foi a primeira LP verdadeiramente de amplo espectro, utilizada tanto para aplicações científicas como comerciais.

ADA: LP criada pelo DoD USA para criação de sistemas de tempo real para uso militar (embarcados em armamentos). Semelhante a Pascal, também usada em aplicações não militares.

Linguagem orientada a objeto

São LPs que suportam os modelos de análise e projeto orientados a objeto. Usam todos os conceitos da orientação a objeto (classe, objeto, herança, associação, etc.).

Smalltalk: criada para explicar os conceitos de orientação a objeto;

C++: LP derivada da linguagem C é utilizada para uma ampla gama de aplicações;

Java: criada pela Sun Microsystems, pode ser executada em qualquer sistema de computação, não importando a CPU que utilize, característica muito importante para aplicações voltadas para Internet

Linguagem orientada a objeto

Linguagens de Script (Internet): São LPs que melhoram a funcionalidade das páginas de Internet. Ex. Javascript (Netscape), VBscript e JScript (Microsoft)

Linguagens Internet: LPs que na maioria das vezes são usadas para criação de páginas Internet dinâmicas. Ex. PHP, ASP (Active Server Page) da Microsoft; JavaServer Pages da Sun; PERL (Practical Extraction and Report Language); e Python.

Linguagens especializadas

São LPs que apresentam formas de sintaxe bastante incomuns e são criadas para aplicações especiais, utilizadas principalmente em Inteligência Artificial (IA) para desenvolvimento de sistemas especialistas.

Ex. LISP: usada quase que exclusivamente para aplicações de IA e sistemas especialistas;

PROLOG: é orientada a objeto e também utilizada em IA para desenvolvimento de sistemas especialistas;

FORTH: utilizada para desenvolvimento de SW para microprocessadores.

Linguagens de 4ª Geração (4GL)

São as LP que apresentam o maior nível de abstração (mais alto nível). Isso é conseguido automatizando um grande número das tarefas do programador (quando não todas). Classificam-se em três categorias:

- Linguagem de consulta;
- Geradores de programa;
- Linguagem de prototipação.

Linguagem de consulta

São LPs declarativas criadas para trabalhar em conjunto com banco de dados, permitindo que esses dados sejam manipulados. Os comandos devem ser utilizados dentro do código gerado por outras linguagens que "hospedam" esses comandos.

Ex. SQL (Structured Query Language).

O SQL é uma LP padronizada pela ANSI, mas praticamente todos fabricantes de BD criam extensões transformando-a em SQL proprietária.

Geradores de programa

São LPs que permitem que os programadores criem programas inteiros, usando apenas declarações baseadas nos modelos gerados pelo projeto do sistema. São LPs de nível muito elevado, onde o trabalho de programação é muito reduzido. Algumas ferramentas CASE (Computer-Aided System Engineering) que automatizam o desenvolvimento de sistemas usam esse tipo de LP.

Ex. Rational Rose (IBM); System Arquitect (Popkin); ERWin (CA - Computer Associates).

Linguagem de prototipação

São LPs que facilitam a criação de interface com o usuário, conhecidas como Linguagens Visuais. Possibilitam construir um protótipo do sistema para manipulação pelo usuário objetivando identificar funcionalidades, estética, completude, etc. Depois disso o programador se preocupa em programar o que deve ser executado quando o usuário interagir com esses componentes de interface. Por isso essas LPs são conhecidas como Linguagens Orientadas a Eventos.

Ex. DELPHI, Visual BASIC, C#.

That´s all!

Thanks