Conceitos Básicos de Compiladores

Prof. Marcus Ramos

Revisado em 03/06/2022

Linguagem de alto-nível

Similaridade com a linguagem natural;
Noções da matemática;
Uso de abstrações;

- **Legibilidade;**
- **Produtividade;**
- **Portabilidade;**
- **Segurança.**

Java
C
C++
Fortran
Algol
Pascal
etc

Legibilidade

- Grau de facilidade para ler, entender e modificar um programa;
- Alta nas linguagens de alto-nível e baixa nas linguagens de baixo-nível.
- Garantido pela similaridade com a linguagem natural, proximidade com a matemática e também pelo uso de abstrações.

Produtividade

- O programador produz a mesma quantidade de linhas por unidade de tempo;
- Quanto mais expressiva a linguagem, maior a produtividade;
- Linguagens de alt~-nível são mais expressivas, linguagens de baixo-nível são menos expressivas
- Exemplo:

```
a = b * c + d          (uma linha)

LOAD    b
LOAD    c
MULT
LOAD    d
ADD
STORE   a              (seis linhas)
```

Portabilidade

- Possibilidade de usar o mesmo programa sem modificações em diversas máquinas e/ou arquiteturas;
- Maior em linguagens de alto-nível e menor em linguagens de baixo-nível;
- Garantida pelo uso de compladores e interpretadores.

Segurança

- Grau de confiança que o programador tem de que o seu programa não contém erros;
- Compilação sem erros e execução em erros;
- Maior em linguagens de alto-nível, menor em linguagens de baixo-nível;
- Garantida pela quantidade de regras que a linguagem tem;
- Muitas nas de alto-nível;
- Poucas nas de baixo-nível.

Linguagem de baixo-nível

- Dependência da máquina ou arquitetura;
- Pode ser classificada em:
  - Linguagem de montagem (assembly code), que usa mnemônicos; em inglês, assembly code;
  - Linguagem de máquina (binário ou hexadecimal); em inglês, machine code;
  - Assembler é o montador, que traduz de assembly code para machine code.

Linguagem fonte

- Linguagem usada para escrever os programas-fonte;
- Entrada do compilador;
- Pode ser alto ou baixo-nível.

Linguagem objeto

- Linguagem usada para escrever os programas-objeto;
- Saída do compilador;
- Pode ser alto ou baixo-nível.

Tipos de processadores

- Linguagem-fonte pode ser alto ou baixo-nível;
- Linguagem-objeto pode ser alto ou baixo-nível;
- Quatro tipos de processadores:

	Objeto: Alto-nível	Objeto: Baixo-nível
Fonte: Alto-nível	Tradutor ou filtro	Compilador
Fonte: Baixo-nível	“Descompilador”	Montador

Especificação de linguagens

- Linguagens são onipresentes na computação;
- Seu uso é viabilizado por meio dos processadores;
- É importante especificar as linguagens corretamente.
- Método formal ou informal.

Especificação de linguagens

- Método formal (desejável)
  - Uso da matemática;
  - Mais rigor e precisão;
  - Menos ambugüidade;
  - Possibilidade de automação;
- Método informal (não desejável)
  - Uso da linguagem natural;
  - Menos rigor e precisão
  - Mais ambigüidade
  - Impede automação

Especificação de linguagens

- Sintaxe
  - Forma, aparência, estrutura;
  - Quais símbolos ou palavras?
  - Em que ordem eles devem ser usado?
  - Palavras-chave x Palavras reservadas;
- Semântica:
  - Significado

Especificação de linguagens

- Sintaxe
  - Livre de contexto;
    (balancemanento de termos)
  - Dependente de contexto;
    (declaração e uso de variáveis, funções etc)
- Semântica

Especificação de linguagens

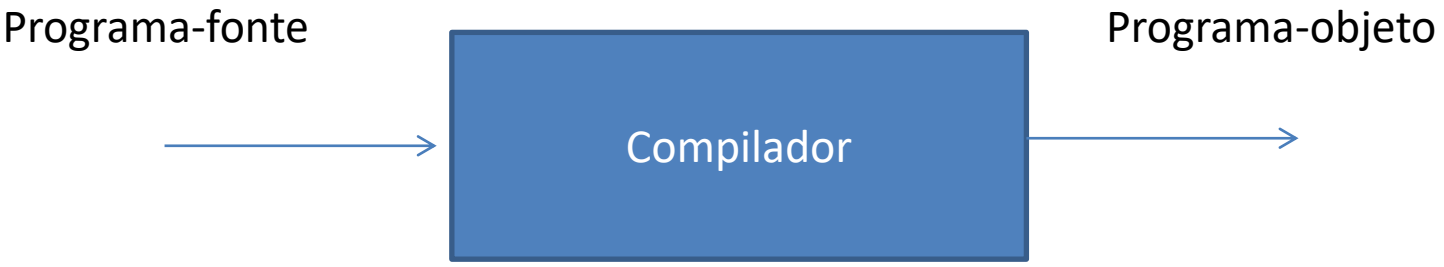
- Uso do método formal na prática:
  - Sintaxe livre de contexto – possível por meio de gramáticas livres de contexto e/ou autômatos de pilha;
  - Sintaxe dependente de contexto – difícil e/ou trabalhoso
  - Semântica - difícil e/ou trabalhoso.

Especificação de linguagens

- Solução:
  - Sintaxe livre de contexto – método formal (envolve aproximações e outras soluções);
  - Sintaxe dependente de contexto – método informal
  - Semântica – método informal.
- Conseqüências práticas.

Compilador

Programa que efetua uma transformação sintática preservando o conteúdo semântico.



Compilação

- Envolve a criação de um programa-objeto a partir do programa-fonte;
- Usada quando se deseja mais eficiência na implementação;
- Preserva o código-fonte.

Interpretação

- Envolve quando se deseja mais interatividade;
- Possui uma eficiência menor na implementação;
- Mais tempo e mais memória;
- Não preserva o código-fonte.

Tempos de execução

- Programa totalmente compilado: 1
- Programa totalmente interpretado: 100
- Programa inicialmente compilado e depois interpretado (como Java): 10

- Java:
  - Protege o código-fonte;
  - Garante portabilidade;
  - Não prejudica tanto a eficiência da execução.

Tempo para construir um compilador

- Extremo inferior:
  - Mini-Basic: 6 homens-mês;
- Extremo superior:
  - Algol 68: 30 anos-ano;
- Típico:
  - Pascal: 2 homens-ano.