Criando Programas de Computador

Mário Leite

...

A criação de um programa de computador envolve várias etapas, antes de chegar ao usuário final. O programador pode criar o programa (código-fonte) em qualquer linguagem que ele domine; entretanto, ANTES de o computador receber essas ordens, elas devem ser traduzidas para a sua própria linguagem; a única que "ele" entende: a linguagem de máquina (linguagem binária), como mostrado no esquema da **figura 4.1**.

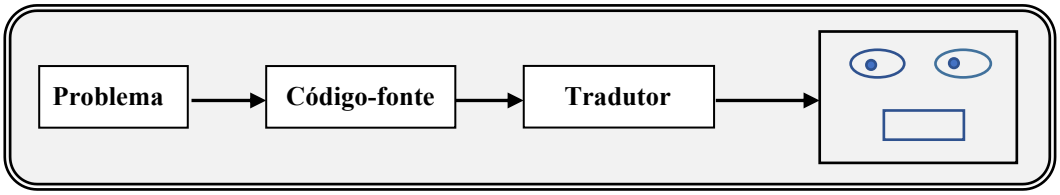

Figura 4.1 - Tradução básica do código-fonte

A tradução do código-fonte para a linguagem de máquina pode ser observada no esquema mostrado na **figura 4.2** onde são mostrados os tipos básicos de tradução.

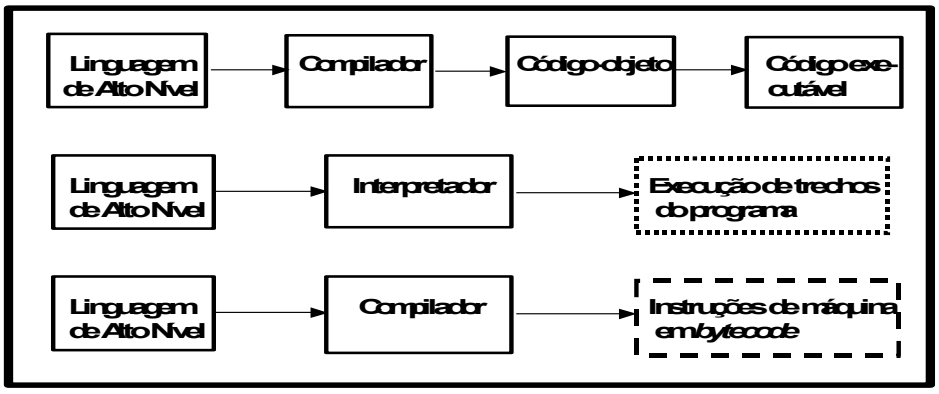

Figura 4.2 - Tradução de um *programa-fonte* para linguagem de máquina

O **Compilador** é um programa TRADUTOR que gera outro programa em Linguagem de baixo nível a partir da tradução integral do *programa-fonte* (programa original escrito em Linguagem de Alto Nível). O compilador traduz o código-fonte escrito em formato de textos, e a partir dele é gerado um código objeto (ou Pcode em *bytecodes*), criando um segundo arquivo (**.obj**) que após compilado deve ser juntado a um *runtime* (executor) através de um *linkEditor* (vinculador); nesta etapa intermediária (que em alguns ambientes de desenvolvimento é transparente ao usuário) são agregadas algumas funções que estão em bibliotecas, produzindo finalmente o arquivo executável do programa com extensão (***.exe)***, se não houver nenhum erro de sintaxe no código-fonte. O resultado é um arquivo (programa executável) que pode ser carregado diretamente do *prompt* do sistema operacional ou de uma janela, como no caso do Windows®. O ambiente MSDOS®, Windows® ou Linux é que, ao executar o programa, o traduz para o código binário da máquina e faz a conexão com os periféricos.

O **Interpretador** faz a leitura/tradução/execução (nesta ordem) de cada linha do *programa-fonte*, permitindo ao programador saber, de imediato, se determinada instrução é válida ou não, na composição de uma linha de código. A **figura 4.2** mostra os dois tipos principais de traduções: *compilação* e *interpretação*. Portanto, existem linguagens compiladas e outras interpretadas; e existem, ainda, aquelas que suportam os dois tipos de tradução. Ainda na **figura 4.2** pode ser notado

que existe um terceiro tipo de tradução do *código-fonte* que não gera diretamente código de máquina; em vez disto, o que é gerado é um arquivo intermediário composto por *bytecodes*. Esses *bytecodes* são produtos resultantes da compilação, mas que ainda serão interpretados por uma máquina virtual para, finalmente, executar o programa. Um exemplo clássico desse tipo de tradução é o utilizado pela linguagem Java, onde os arquivos contendo os *bytecodes* têm extensão **class.** A **tabela 4.1** mostra as vantagens e desvantagens do compilador e do interpretador; e a pergunta que pode ser feita é a seguinte: "*qual dos dois tipos é melhor*?" O ideal é poder utilizar uma linguagem que ofereça esses dois recursos simultaneamente. Por exemplo, o Visual Basic (até a versão 6) possui os dois tradutores; na etapa de desenvolvimento a aplicação pode ser testada com o interpretador, e depois de feitas todas as correções o compilador pode ser usado para gerar o *código-executável* da aplicação. O Delphi, concorrente direto do Visual Basic.Net, é apenas compilado; durante a etapa de desenvolvimento, para cada teste é gerado em disco o arquivo executável de sintaxe. O mesmo acontece com C, que são apenas compiladas; ao passo que o Python, por exemplo, só possui o interpretador.

Tradutor	Vantagens	Desvantagens
Compilador	* Permite estruturas de programas mais complexas, otimizando o código. * Gera arquivo executável, permitindo maior autonomia e segurança do código-fonte. * Execução mais rápida.	* Correção de erros mais difícil. * Não permite correções de modo dinâmico. * Necessita de várias etapas de tradução do código-fonte. * Consome muita memória.
Interpretador	* Consome pouca memória. * Permite estruturas dinâmicas de programação. * Tradução em uma única etapa.	* Execução lenta. * Não gera arquivo executável, o que diminui a segurança do código-fonte.

Tabela 4.1 - Comparação entre Compilação e Interpretação

Abaixo, estão alguns exemplos de códigos-fonte para verificar se um determinado número é primo, em algumas linguagens de programação.

---

1. VISUALG

```
Algoritmo "primo"
var
   n, i, cont : inteiro
inicio
   Escreva("Digite um número: ")
   leia(n)
   cont <- 0
   Para i De 1 Ate n Faca
      Se(n % i = 0) Entao
         cont <- cont + 1
      Fimse
   fimpara
   Se(cont = 2) Entao
      Escreva("É primo")
   Senao
      escreva("Não é primo")
   fimse
FimAlgoritmo
```

---

2. FORTRAN 77

```
PROGRAM PRIMO
   INTEGER N, I, CONT
   CONT = 0

   PRINT *, 'Digite um número:'
   READ *, N

   DO 10 I = 1, N
      IF (MOD(N, I) .EQ. 0) CONT = CONT + 1
10    CONTINUE

   IF (CONT .EQ. 2) THEN
      PRINT *, 'É primo'
   ELSE
      PRINT *, 'Não é primo'
   ENDIF
END
```

---

3. BASIC (QBASIC / FreeBASIC estilo antigo)

```
DIM n, i, cont
cont = 0

PRINT "Digite um número:"
INPUT n

FOR i = 1 TO n
    IF n MOD i = 0 THEN cont = cont + 1
NEXT i

IF cont = 2 THEN
    PRINT "É primo"
ELSE
    PRINT "Não é primo"
END IF
```

---

4. PASCAL

```
program Primo;
var
  n, i, cont: integer;
begin
  cont := 0;
  writeln('Digite um número:');
  readln(n);

  for i := 1 to n do
    if (n mod i = 0) then
      cont := cont + 1;

  if (cont = 2) then
    writeln('É primo')
  else
    writeln('Não é primo');
end.
```

---

5. PYTHON

```python
n = int(input("Digite um número: "))
cont = 0

for i in range(1, n+1):
    if(n % i == 0):
        cont += 1

if(cont == 2):
    print("É primo")
else:
    print("Não é primo")
```

---

6. C

```c
#include <stdio.h>

int main() {
    int n, i, cont = 0;

    printf("Digite um número: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        if (n % i == 0) cont++;
    }

    if(cont == 2)
        printf("É primo");
    else
        printf("Não é primo");

    return 0;
}
```

---

7. VISUAL BASIC 6

```
Dim n As Integer, i As Integer, cont As Integer
cont = 0

n = Val(InputBox("Digite um número: "))

For i = 1 To n
    If n Mod i = 0 Then cont = cont + 1
Next i

If cont = 2 Then
    MsgBox "É primo"
Else
    MsgBox "Não é primo"
End If
```

---

8. DELPHI (modo Console)

```
program Primo;
var
  n, i, cont: Integer;
begin
  cont := 0;
  Write('Digite um número: ');
  Readln(n);

  for i := 1 to n do
    if(n mod i = 0) then
      cont := cont + 1;

  if(cont = 2) then
    Writeln('É primo')
  else
    Writeln('Não é primo');
end.
```

---

9. VB.NET

```
Module Module1
  Sub Main()
      Dim n, i, cont As Integer

      Console.Write("Digite um número: ")
      n = Console.ReadLine()

      cont = 0
      For i = 1 To n
          If(n Mod i = 0) Then cont += 1
      Next

      If(cont = 2) Then
          Console.WriteLine("É primo")
      Else
          Console.WriteLine("Não é primo")
      End If
    End Sub
End Module
```

---

10. PHP

```
<?php
$n = intval(readline("Digite um número: "));
$cont = 0;

for ($i = 1; $i <= $n; $i++) {
    if ($n % $i == 0) $cont++;
}

if ($cont == 2)
    echo "É primo";
else
    echo "Não é primo";
?>
```

---

11. JAVA

```java
import java.util.Scanner;

class Primo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n, i, cont = 0;

        System.out.print("Digite um número: ");
        n = sc.nextInt();

        for (i = 1; i <= n; i++) {
            if (n % i == 0) cont++;
        }

        if (cont == 2)
            System.out.println("É primo");
        else
            System.out.println("Não é primo");
    }
}
```

---

12. JAVASCRIPT (Node.js ou browser)

```javascript
let n = parseInt(prompt("Digite um número:"));
let cont = 0;

for (let i = 1; i <= n; i++) {
    if (n % i === 0) cont++;
}

if (cont === 2)
    alert("É primo");
else
    alert("Não é primo");
```

Nota1: Postagem baseada no livro: "*Curso Básico de Programação: Teoria e Prática*".
Publicado pelo autor na "Amazon", "Clube de Autores" e "Editora Ciência Moderna".

https://www.amazon.com.br/Curso-B%C3%A1sico-Programa%C3%A7%C3%A3o-Teoria-Pr%C3%A1tica/dp/8539908700

Nota2: Acesse o *link* abaixo para ver meus mais recentes livros de Python publicado pelo "Clube de Autores", no formato impresso, da coleção "***1001 Programas em Python Para Você Aprender Praticando***":
Volume1: Nível Básico (500 programas)
Volume2: Nível Intermediário (300 programas)
Volume3: Nível Avançado (201 programas)

https://clubedeautores.com.br/livros/autores/mario-leite

Para adquirir PDF dos livros: marleite@gmail.com