Paradigma Imperativo

Francesco Antonello Ferraro

April 26, 2017

Abstract

whatever

Abstract

Esse artigo visa demostrar a especificidades do paradigma imperativo, além de exibir uma análise de como o mesmo se contrapõe com o paradigma funcional. Para que a visualização e entendimento sejam claros serão utilizadas tabelas contrastando essas diferenças.

1 Relação com Engenharia de Software

Uma linguagem de programação é a padronização de métodos que tem como objetivo comunicar instruções um computador mantendo a mesma estrutura sintática. Não obstante, um paradigma de programação é a visão que o indivíduo que digita o código tem sobre a estrutura e execuçjão de um programa. O que nos leva a concluir que diferentes linguagem fazem escolhas diferentes quanto a forma e compilação dos mesmos, dependendo da visão que o seu autor vê como o mais ajustado para enfrentar os seus desafios e problemas que a mesma tenta resolver. Portanto, como enegenheiros de software é imprescindível que possamos entender as características básicas que diferenciam as principais linguagens de programação do mercado, permitindo a escolha da linguagem mais apropriada para o desenvolvimento de um software para um possível contratante.

2 Contextualização

Mas o que torna uma linguagem imperativa? Basicamente um programa encrito de maneira imperativa possui duas característica básicas:

- Um estado do programa;
- Instruções que alteram o estado do programa.

Por intruções pode se inferir o sentido lexical da palavra de mandar, ou imperar que uma atividade seja executada de forma autoritária. Uma frase que é usualmente usada na literatura para decrever esse paradigma é

> Primeiro faça isso, depois faça aquilo.

Indicando que o prgramador é diretamente responsável por preconceber todos os caminhos em que o usuário vai poder percorrer. O problema começa a aparecer quando o tamanho do programa cresce. Maiores programas oferecem mais "caminhos" aos usuário, que exponecialmente aumentam a combinações de rotas possíveis em que o usuário teóricamente tem acesso. Com isso situações atípicas podem intruduzir bugs de difícil reconhecimento tornando o código de difícil manutenção.

3 Modelo Computacional

O paradigma imperativo também é conhecido por paradigma procedural, é o modelo de programação mais antigo. Na década de 40, John von Neumann e outros cientistas da computação se dão conta de que assim como o programa, seus dados podem ser guardados na memória principal da aplicação. Touring também trabalhaca na mesma linha de pensamento à época e é esse conceito que fundamenta o paradigma imperativo. A atribuição de novos valores às variáveis que residem em memória é o cerne do conceito. Essas declarações atribuem locais de memória e nome para cada variável e associam tipos aos valores armazenados. A execução do programa é basicamente síncrona; isto é, eles são executados na mesma ordem em que aparecem na memória. Entretanto, tanto variações condicionais quando incondicionais podem alterar o fluxo normal de execução. Os fatos de que um programa alocar dinamicamente variaveis e seus valores em memória , bem como, possuir comandos de ramificação e/ou condicionais que permitem que um determinado comando deixe de ser executado ou seja repetidamente executado tornam uma linguagem completa quanto a Touring. Essa medida, é importante por que apresenta um conjunto mínimo de recusos que uma linguagem de programação deve possuir ; a fim de ser capaz de expressar qualquer algoritmo a ser cocebido. Segundo TUCKER(2010):

> Linguagens imperativas que contem variaveis e valores inteiros, operacoes aritmeticas basicas, comandos de atribuicao, sequenciamento de comandos baseados em memorias, condicoes e comandos de ramificacao sao "completas quanto a Turing"

4 Características

A fim de ser considerada imperativa uma linguagem tem de obrigatoriamente ser completa quanto a Touring e suportar características mais elaboradas que se tornaram mais necessárias com a evolução da ciência como um todo. São eles

- Estruturas de controle.
- Entrada/saída.
- Manipulação de exceções e erros.
- Abstração procedural.
- Expressões e atribuição.
- Suporte de biblioteca para estruturas de dados.

Enquanto os três primeiros ítems são inerentes à Touring, os três ultimos fazem referência a essas características mais contemporâneas do paradigma. São esse pontos em que o artigo tenta focar, por serem as características que variam dependendo da linguagem a ser observada.

4.1 Abstração Procedural

É a capacidade de extrair pequenas unidades lógicas que permitem o programador abstrair a implementação de um algarismo, possibilitando um maior foco no valor da algoritmo do que sua implementação em sí. Considere esse simples programa escrito em Golang. Dado um array de inteiros chamado list, ele impreme na tela verdadeiro ou falso, dependendo do valor da variável check.

```
fmt.Println("Current Time:", time.Now())

var list = []int{2, 3, 3}
var check = 3
```

```
for _, a := range list {
if a == check {
log.Println(true)
}
}
log.Println(false)
```

Já no exemplo a seguir, o resultado é o mesmo, entretanto as instruções ultilizadas para gerar o resultado do algoritmo que indentifica se a variével check está em _ list_ foram abstraidos para uma função chamada contains. Nesse caso, a implmentação é pequena, apenas seis linhas, mas é fácil perceber como esse processo permite que grandes algoritmos não atrapalhem a legibilidade do código escrito. Uma vez escrito, um algoritmo seja abstraído em um porção que código que pode ser reultilizada em diferentes partes do programa,

```
package main

import "log"

func main() {
var list = []int{1, 2, 3}
var check = 3
log.Println(contains(list, check))
}

func contains(s []int, e int) bool {
for _, a := range s {
if a == e {
return true
}
}
return false
}
```

4.2 Expressões e Atribuições

4.3 Suporte de Bibliotecas para estrutura de Dados

4.4 Exemplos de Linguagens Imperativas

Ada,ALGOL,Basic,C,PHP,Java,Cobol,Fortran,Pascal,Python,Lua e Mathematica

kk

5 Bibliografia

Tucker, Allen B. Linguagens de programação - Princípios e paradigmas Allen B. Tucker, Robert E. Noonan Porto Alegre : AMGH, 2010.