UNIP

UNIVERSIDADE PAULISTA

Paradigmas de Linguagens

Paradigma Imperativo

Professora Sheila Cáceres

Paradigma Imperativo – Um pouco de historia

- Paradigma fundamentado no conceito de máquina de Turing.
- A máquina de Turing é uma abstração matemática proposta por Alan Turing nos anos 30.
- John von Neumman aproximou o modelo de Turing a arquitetura de computadores.

Nota

Segundo alguns autores, este paradigma é sinônimo do paradigma estruturado. Segundo outros autores, existem vários paradigmas imperativos (estruturado, concorrente, orientado a objetos).

Nesta apresentação adotaremos a primeira abordagem (imperativo = estruturado).

Paradigma Imperativo

- Está fundamentado na idéia de computação como um processo que realiza mudanças de estado.
- Um estado representa uma configuração qualquer da memória do computador.
- Os programas de LPs incluídas nesse paradigma especificam como uma computação é realizada por uma sequência de alterações no estado da memória do computador.
- O foco deste paradigma é especificar como um processamento deve ser feito no computador.

Paradigma Imperativo

- A essência da programação imperativa (e também da máquina de Turing) se resume a três conceitos:
  1. Descrição de estados de uma máquina abstrata por valores de um conjunto de **variáveis**
  2. Reconhecedores desses estados, que são **expressões** compostas por relações u operações entre valores (podem ser variáveis)
  3. Comandos, que podem ser de dois tipos
     a) **Atribuição**
     b) **Controle**: determinam qual o próximo comando a ser executado.
- Os conceitos (negrita) descritos neste slide tem sido vistos em aulas passadas.

Paradigma Imperativo - Comandos

- Um programa imperativo é uma sequência ordenada de comandos.
- Cabe destacar que todos os programas no paradigma estruturado podem ser construidos utilizando apenas três estruturas essenciais:
  - seqüencial,
  - condicional

```
if (a==10)
        Console.WriteLine(“a é 10”)
else
        Console.WriteLine(“a é diferente 10”)
```

  - iterativa (repetição)

```
While (a>10){
        Console.WriteLine(“a é maior que 10”)
}
```

- Procura-se encontrar uma forma de quebrar um problema complexo em pequenas partes simples que trabalhadas conjuntamente, permitam solucioná-lo.

Paradigma imperativo - Exemplo

Programa em C

```
#include <stdio.h>
void main()
{
    int t;
    for(t = 0; t < 100; t++) {
        printf(“%d”,t);
        if(t == 10) break;
    }
}
```

Modularidade

- O paradigma imperativo permite a utilização de estruturas modulares.
- Uma estrutura modular permite separar o programa em **trechos menores** usando
  - Blocos:
  - **Funções**: permitem evitar a repetição de código.
- As bases deste paradigma são usadas por outros. Por exemplo, no paradigma orientado a objetos (que será visto proximamente), as funções do paradigma imperativo se tornam os métodos das classes do paradigma orientado a objetos.

Saltos incondicionais

- O comando de salto incondicional (goto) **não** representa uma boa prática de programação e este paradigma evita a sua utilização.

Paradigma Imperativo

- Programa e variáveis são armazenados juntos.
- O programa contém uma serie de comandos para executar cálculos, atribuir valores a variáveis, obter entradas, produzir saídas ou redirecionar o controle para outro ponto nessa série de comandos
- **Linguagens**: C, Ada, Perl, Cobol, Fortran.

Vantagens

- Eficiência (embute o modelo Von Neumann)
- Paradigma dominante e bem estabelecido
- Modelagem natural de aplicações do mundo real
- É fácil de se entender, sendo amplamente usada em cursos introdutórios de programação.

Desvantagens

- Possui difícil legibilidade e facilita introdução de erros em sua manutenção. Programas muito longos podem dificultam a programação
- Focaliza o "como" e não o "quê": como a tarefa deve ser feita e não em o que deve ser feito.
- Relacionamento indireto com a E/S (indução a erros/estados). Tende a gerar códigos confusos, onde tratamento dos dados são misturados com o comportamento do programa

Algumas linguagens Imperativas

- Ada
- ALGOL
- Basic
- C
- PHP
- Cobol
- Fortran
- Pascal
- Python

Fortran

- Fortran (FORmula TRANslation) foi a primeira linguagem de alto nível para ganhar ampla aceitação.
- Ele foi projetado para aplicações científicas e contou com uma notação algébrica, tipos, subprogramas, e entrada/saída formatada.
- Foi implementado em 1956 por John Backus na IBM especificamente para a máquina IBM 704. Execução eficiente foi uma grande preocupação, consequentemente, sua estrutura e comandos têm muito em comum com linguagens de montagem.
- FORTRAN ganhou ampla aceitação.

Algol

- ALGOL 60 (ALGorithmic Oriented Language) foi concebido em 1960 por um comitê internacional para uso em resolução de problemas científicos.
- Ao contrário FORTRAN foi concebido de forma independente de uma aplicação, uma escolha que conduzem a uma linguagem elegante.
- A descrição do ALGOL 60 introduziu a notação BNF para a definição de sintaxe e é um modelo de clareza e completude.
- Embora ALGOL 60 não conseguiu ganhar grande aceitação, introduziu estrutura de bloco, instruções de controle estruturados e procedimentos recursivos no paradigma de programação imperativa.

Pascal

- Desenvolvida pelo Prof. Niklaus Wirth e foi apresentada em 1970.
- Foi desenvolvida tendo em vista o ensino em programação.
- Tem obtido muito sucesso.
- Em Pascal, as funções devem ser declaradas antes de serem utilizadas.

C

- C foi projetado originalmente para ser implementado no sistema operacional UNIX por Dennis Ritchie no AT&T Bell Labs entre 1969 e 1973.
- A linguagem foi chamada "C", porque suas características foram obtidas a partir de uma linguagem anteriormente chamado de " B", que de acordo com a Ken Thompson era versão reduzida da linguagem de programação BCPL.
- C é uma das linguagens de programação mais populares.
- C tem influenciado muitas outras linguagens de programação, mais notavelmente C++, que originalmente começou como uma extensão para C.

Referências Bibliográficas

O material para a realização desta apresentação foi extraído de:

* Tucker, Linguagens de Programação, principios e paradigmas.

* Principios de Linguagens de programação, Ana Cristina Vieira de Melo, Flavio Soares