Algoritmos e Lógica de Programação

Construção de Algoritmos

5

Sintaxe e Semântica
Instruções Primitivas
Estrutura seqüencial
Dicas para escrever bons algoritmos

Reinaldo Gomes
reinaldo@cefet-al.br

O que é Sintaxe e Semântica?

- Sintaxe diz respeito a forma de como as instruções devem ser escritas
  - Conjunto de regras formais que especificam a composição dos algoritmos a partir de letras, dígitos e outros símbolos

A violação da sintaxe de uma instrução impede que o algoritmo seja executado!

O que é Sintaxe e Semântica?

- Semântica diz respeito ao significado lógico das instruções que serão executadas pelo computador
  - Conjunto de regras que especificam o "significado" de qualquer programa, sintaticamente válido.

A violação da semântica de um algoritmo não impede que ele seja executado. Todavia, ele processará um resultado errado!

Sintaxe e Semântica

- Exemplo de erro sintático:
  - Inteiro : média;
  - Média ← 25/5 {pois toda divisão retorna um no Real}
  - "média" >= 7 {pois não se pode comparar tipos diferentes}
- Exemplo de erro semântico:
  - Real : média;
  - Se (média >= 7) então
    - Escreva ("REPROVADO");
  - Senão
    - Escreva ("APROVADO");

Sintaticamente a estrutura "SE-Então-Senão" esta correta, mas semanticamente NÃO.

Sintaxe e Semântica

- Os erro sintáticos são identificados pelos tradutores, enquanto que os erros semânticos não
- Por isso que os erros semânticos exigem mais atenção para corrigi-los

Instruções Primitivas

- São os instruções básicas que efetuam tarefas essenciais para o recebimento e apresentação de dados. Estas são:
  - Entrada
  - Saída

Instruções Primitivas

- A instrução de entrada de dados permite que informações dos usuários sejam transferidas para a memória do computador (variáveis)
- Sua sintaxe pode ser:
  - Leia (var1);
    - Ex: Leia (nome);
  - Leia (var1, ..., varN);
    - Ex: Leia (nome, sexo)
  - A semântica: os dados são fornecidos ao computador por meio de um dispositivo de entrada (ex: teclado e mouse) e armazenados nas posições de memória das variáveis

7

Instruções Primitivas

- Exemplo – Entrada

```
Algoritmo ExemploInstruçãoEntrada
Real: preçoUnit, preçoTot;
Inteiro: qtd ;
Início
   Leia(preçoUnit, qtd);
   preçoTot ← preçoUnit * qtd;
Fim
```

8

Instruções Primitivas

- Questão de Implementação I
  - Quando uma variável é declara, esta apenas reserva uma posição na RAM. Ou seja, o conteúdo dessa posição é vazio.
  - Assim, a cada variável criada deve-se ler ou atribuir um valor a ela

9

Instruções Primitivas

- Questão de Implementação II
  - Diferente do operador "←", que só atribui valores pré-definidos, a instrução LEIA permite qualquer entrada de dados válida
  - Ou seja, os valores das variáveis não são mais fixos

10

Instruções Primitivas

- A instrução de saída de dados é o meio pelo qual variáveis, constantes e expressões têm seus dados exibidos pelos dispositivos de saída de um computador
  - Escreva (var1);
    - Ex: Escreva (salário);
  - Escreva (Var1, ..., varN);
    - Ex: Escreva (nome, endereco, cidade);
  - Escreva ("texto");
    - Ex: Escreva ("Aula de programação");
  - Escreva ("texto", var1, ..., varN,"texto",var1, ..., varN);
    - Ex: Escreva ("nome e CPF = ",nome, cpf," fone = ",fone);

11

Instruções Primitivas

- Atenção:
- Usa-se "," (vírgula) para concatenar (juntar) o valor de uma variável com um texto explicativo
- Exemplo:
- média ← 15/2;
- Escreva ("Média = ",média);

- Média = 7,5

12

Instruções Primitivas

- Semântica da instrução escreva:
  - Enviar seus argumentos para um dispositivo de saída.
  - No caso de uma lista de variáveis, o conteúdo de cada uma delas é pesquisado na memória e enviado para um dispositivo de saída
  - No caso de argumentos constantes (ex: texto), estes são enviados diretamente para o dispositivo de saída.

13

Instruções Primitivas

- Exemplo

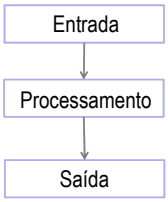
```
Algoritmo ExemploInstruçõesEntradaSaida
Real : preçoUnit, preçoTot;
Inteiro : qtd;
Início
   Leia(preçoUnit, qtd);
   preçoTot ← preçoUnit * qtd;
   Escreva(preçoTot);
Fim.
```

14

Instruções Primitivas

- Questão de Implementação III

```
Início
   Leia(preçoUnit, qtd);
   preçoTot ← preçoUnit * qtd;
   Escreva(preçoTot);
Fim
```



15

Instruções Primitivas

- Duas regras básicas para melhorar nossos algoritmos
  - Toda operação de leitura deve antes enviar uma mensagem ao usuário informando que dados ele deve entrar
  - Todo resultado enviado ao usuário, deve ser precedido de uma mensagem explicativa

16

Instruções Primitivas

```
Algoritmo ExemploInstruçõesEntradaSaida
Real : preçoUnit, preçoTot;
Inteiro : qtd;
Início
   Escreva("informe preço unitário e quantidade");
   Leia(preçoUnit, qtd);
   preçoTot ← preçoUnit * qtd;
   Escrever("Preço total = ", preçoTot);
Fim
```

17

Estrutura de Sequência

- Na estrutura de seqüência os comandos de um algoritmo são executados na ordem em que aparecem
- Uma estrutura de seqüência é delimitada pelas palavras reservadas Início e Fim e contém basicamente comandos de atribuição, de entrada e de saída
- Todos os algoritmos vistos até agora utilizam uma única estrutura seqüencial (Início-Fim)

18

10 dicas para escrever bons algoritmos

- Dica Geral: Todo algoritmo deve ser feito visando a sua eficiência, clareza e manutenção.

1. Pense de forma incremental (Refinamentos sucessivos) e detalhada no problema a ser resolvido
   - Quais são os dados de entrada?
   - Como estes dados devem ser processados?
   - Quais são os dados de saída?
2. Faça o algoritmo o mais simples possível, facilitando:
   - A leitura do algoritmo por outras pessoas
   - A correção de erros quando estes existem

10 dicas para escrever bons algoritmos

3. Escreva comentários claros e objetivos no momento em que estiver escrevendo o algoritmo
   - Comentários podem ocorrer em qualquer parte do algoritmo, eles devem estar entre chaves
     - Ex: {isso é um comentário}.

- Os comentários deverão acrescentar alguma coisa; não apenas frasear as instruções.
  - Instruções – dizem o que está sendo feito
  - Comentários – dizem o porquê de está sendo feito
  - EX: lucro ← venda – custo;
  - {Atribui a lucro o valor de venda menos o valor de custo}
  - {Calcula o lucro obtido}

10 dicas para escrever bons algoritmos

4. Use comentários no início do algoritmo
   - Descrição do que o algoritmo faz;
   - Como utilizá-lo;
   - Qual o significado das variáveis mais importantes;
   - As estruturas de dados e métodos especiais utilizados;
   - O autor;
   - A data de escrita.

10 dicas para escrever bons algoritmos

4. Use comentários no início do algoritmo

```
Algoritimo Soma
{esse algoritmo soma as variáveis num1 e num2.
Autor: Meu Nome; Data: 99/99/9999}
Inteiro : num1,num2,soma;
Início
Escreva (“Digite dois números inteiros”)
Leia(num1, num2);
soma ← num1 + num2;
Escreva(“A soma é:”,soma);
Fim
```


10 dicas para escrever bons algoritmos

5. Escolha nomes de variáveis que sejam significativos
   - Ex: lucro ← preçoVenda – preçoCusto;
6. Utilize espaços e/ou linhas em branco para melhorar a legibilidade do algoritmo.
7. Utilize parênteses para aumentar a legibilidade e prevenir erros.
   - Exemplos:
     - sem parênteses | com parênteses extras
     - 5 > 7 | 3 = 5 (5>7) e (3=5)
8. Escreva apenas um comando por linha. isto também facilita a legibilidade do algoritmo

10 dicas para escrever bons algoritmos

9. Alinhe os comandos de acordo com o nível a que pertençam, isto é, destaque a estrutura na qual estão contidos. Este alinhamento é chamado de identação.

```
Início
    comando do algoritmo no nivel 1;
        subcomando dentro do nível 2;
            subcomando dentro do nível 3;
    comando do algoritmo no nível 1
    Início
        subcomando dentro do nível 2;
        subcomando dentro do nível 2;
    Fim
Fim
```

10 dicas para escrever bons algoritmos

10. Toda vez que for feita uma modificação no algoritmo, os comentários associados devem ser alterados. É preferível não comentar do que deixar um comentário errado.

Construção de Algoritmos

5

Sintaxe e Semântica
Instruções Primitivas
Estrutura seqüencial
Dicas para escrever bons algoritmos

Reinaldo Gomes
reinaldo@cefet-al.br