UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br
Departamento de Ciências de Computação

SCC-120 - Capítulo 2
Introdução à Linguagem Algorítmica

João Luís Garcia Rosa[1]

[1] Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
http://www.icmc.usp.br/~joaoluis

2010

Sumário

Sumário

Acontecimento

- Seja o seguinte acontecimento:

  Um homem troca uma lâmpada queimada.

- Neste acontecimento nota-se que a *ação* praticada pelo homem foi a de “trocar uma lâmpada.” Nota-se ainda que o *efeito* do acontecimento foi a lâmpada trocada. E finalmente, tem-se que o *objeto* a sofrer a ação foi a “lâmpada.”
- Estas três conclusões são óbvias, mas servem como introdução aos conceitos de ação, efeito e objeto:

Sumário

Ação, efeito, objeto

- **Ação**: é qualquer acontecimento de duração finita, com um resultado previsto e bem definido. A ação deve ter lugar em um período definido de tempo: início em t_0 e término em t_1. Ou seja, o efeito final da ação pode ser descrito comparando o estado no momento t_0 com o estado no momento t_1.
- **Efeito**: é a mudança de estado após a ação. No exemplo, o efeito final é que a lâmpada, que no momento t_0 estava queimada, no momento t_1 estará trocada.
- **Objeto**: é o elemento necessário para que uma ação seja executada.

Sumário

Decomposição em sub-ações

- O acontecimento do homem trocando a lâmpada pode, no decorrer do tempo, ser dividido em sub-ações, mais detalhadas, ou seja:
  - “Posicione a escada debaixo da lâmpada queimada
  - Suba na escada até que a lâmpada possa ser alcançada
  - Gire a lâmpada queimada no sentido anti-horário até que se solte
  - Escolha uma nova lâmpada da mesma potência da queimada
  - Posicione a nova lâmpada no soquete
  - Gire-a no sentido horário até que ela se firme
  - Desça a escada.”

Sumário

Ação Primitiva

- *Processo* é a execução de uma ação, ou seqüência de sub-ações mais elementares, cujo efeito final cumulativo é igual ao da ação original especificada pelo acontecimento.
- *Processador* é o agente que executa o processo.

Tomando novamente o acontecimento anterior, nota-se que algumas das sub-ações descritas pode ser detalhada em outras sub-ações, e que estas, por sua vez, também pode ser dividida em outras sub-ações:

Ação Primitiva

1. Posicione a escada embaixo da lâmpada queimada
2. Selecione uma nova lâmpada para a substituição
   - Se a potência não for a mesma da queimada, repita o processo até encontrar uma que sirva
     - Descarte a lâmpada selecionada
     - Selecione uma nova
3. Repita até que a lâmpada possa ser alcançada
   - Suba um degrau da escada
4. Repita até que a lâmpada fique livre do soquete
   - Gire a lâmpada no sentido anti-horário
5. Posicione a nova lâmpada no soquete
6. Repita até que a lâmpada esteja firme no soquete
   - Gire a lâmpada no sentido horário
7. Repita até que se alcance o chão
   - Desça um degrau da escada

Ação Primitiva

- Como pode ser verificado, este processo de detalhamento das ações pode continuar quase que indefinidamente.
- Para estabelecer um limite, é que aparece o conceito de ação primitiva.
- *Ação primitiva* é uma ação que o processador é capaz de entender e executar sobre objetos bem determinados, sem a necessidade de que esta ação seja melhor explicada.

Sumário

Algoritmo

- *Algoritmo* é um conjunto finito ordenado, bem conhecido e bem definido de ações primitivas, que possam ser executadas sobre objetos bem definidos e que produza um efeito desejado.
- Um algoritmo se destina a resolver um problema: fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, para se atingir, como resultado final, a solução de um problema.
- Um *padrão de comportamento* descreve as seqüências de ações necessárias para se obter um certo acontecimento.
- A palavra “algoritmo” vem do nome de um matemático persa (825 d.C.), *Abu Ja’far Mohammed ibn Musa al Khowarizmi* [1].

Algoritmo

- Ao escrever um algoritmo, considera-se o acontecimento como um processo. Este processo é subdividido em outros processos chegando-se a um seqüência de sub-ações compostas apenas por ações primitivas.
- Faz-se necessária a definição de uma nomenclatura para a descrição de acontecimentos.

Nomenclatura

- *aspas* (“”): Usa-se aspas para indicar o início e o fim de um acontecimento.
- *ponto-e-vírgula* (;): Usa-se ponto-e-vírgula para separar cada frase.
  Ex.: Troca de um pneu

“trocou o pneu”	“ergueu o carro;	“ergueu o carro;
	trocou o pneu;	desparafusou a roda;
	abaixou o carro”	trocou o pneu;
		parafusou a roda;
		abaixou o carro”

- Nota-se que a seqüência da terceira coluna é a descrição mais detalhada do acontecimento.

Exemplos de Algoritmos

1. “pegar a panela;
pegar a sacola;
descascar as batatas;
colocar na panela”.
2. “se não quiser sujar o vestido então colocar o avental;
descascar as batatas”.
O resultado será:
dona de casa com avental | dona de casa sem avental
e as batatas descascadas | e as batatas descascadas.

Exemplos de Algoritmos

- Define-se os tipos de objetos usados:
  Tipo coisa = sacola, panela;
  Tipo fruta = banana, melão, abacaxi, melancia, abacate, pêra;

Table: Ações primitivas e objetos

ação	*objeto*
pegar	coisa, fruta
descascar	fruta
guardar	coisa, fruta

São algoritmos?

1. “pegar panela; descascar pêra”
2. “pegar panela; descascar maçã”
3. “pegar sacola; descascar panela”
4. “colocar avental; descascar batatas”

Constantes

- Uma *constante* é um dado valor fixo que não se modifica ao longo do tempo, durante a execução de um programa. Pode ser um número (como se conhece da Matemática), um valor lógico ou uma seqüência de caracteres quaisquer com algum significado para o problema em estudo. Conforme o seu tipo, a constante é definida como sendo:
  - *numérica*: 25; 3,14; $7,8 \times 10^3$.
  - *lógica*: *falso* ou *verdadeiro*.
  - *literal*: “José da Silva”; “X1Y2Z3”; “*A!B?-”; “1234”; “17/03/07”.

Atribuição e variável

- Sabe-se da Matemática que uma *variável* é a representação simbólica dos elementos de um certo conjunto.
- Nos algoritmos, destinados a resolver um problema no computador, a cada variável corresponde uma *posição de memória*, cujo conteúdo pode variar ao longo do tempo durante a execução de um programa.
- Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

Atribuição e variável

- “descascar pêra” | “descascar banana” | “descascar melão”
- “*x* é uma representante do tipo fruta;
  descascar *x*”
- “*x* é um representante do tipo fruta;
  x é uma banana;
  descascar *x*”.
- “tipo fruta x;
  x=banana;
  descascar x”

São algoritmos?

1. “coisa y;
   y=sacola;
   guardar y”
2. “fruta y;
   y=maçã;
   descascar y”
3. “fruta x;
   coisa y;
   x=pera;
   y=sacola;
   guardar x e y”

Comentários

- **Importante**: clareza do algoritmo.
- *Comentário*: texto, frase, que aparece sempre delimitado por barra e asterisco (/*comentário*/) ou iniciado por duas barras (//comentário).
- Podem ser colocados em qualquer ponto do algoritmo.
- Exemplo:

```
numérico
    MAT, /*número da matrícula do aluno*/
    NOTA, /*total de pontos obtidos no semestre letivo*/
    COD; /* código do curso */
literal
    NOME, // nome completo do aluno
    END, // endereço do aluno
    C; // conceito final
```

Sumário

Expressões Aritméticas

- Denomina-se *expressão aritmética* aquela cujos operadores são aritméticos e cujos operandos são constantes e/ou variáveis do tipo numérico.
- O conjunto de operações básicas adotado é o que se conhece da Matemática:
  - adição
  - subtração
  - multiplicação
  - divisão
- Exemplos:
  X + Y; TOTAL/N; X - Y; A x B + C; (SOMA*SOMA).

Expressões Lógicas

- Denomina-se *expressão lógica* aquela cujos operadores são lógicos e cujos operandos são relações, constantes e/ou variáveis do tipo lógico. Representa uma condição.
- *Relação*: comparação realizada entre dois valores de mesmo tipo básico.
- Estes valores são representados na relação através de constantes, variáveis ou expressões aritméticas, estas últimas para o caso de valores numéricos.
- Uma relação é escrita usando os operadores relacionais

Relações

Table: Operadores Relacionais

Operador	*Função*
= =	Igual
!=	Diferente
<	Menor
>	Maior
<=	Menor ou igual
>=	Maior ou igual

- O resultado obtido de uma relação é sempre um valor lógico.

Operadores Lógicos

- A Álgebra das Proposições define três conectivos usados na formação de novas proposições a partir de outras já conhecidas.
- Estes conectivos são os operadores nas expressões lógicas, a saber:

e: conjunção
ou: disjunção
não: negação

Expressões Literais

- Uma *expressão literal* é aquela formada por operadores literais e operandos que são constantes e/ou variáveis do tipo literal. Supondo que A e B são variáveis literais e que o símbolo "|" é um operador de concatenação de literais, a expressão

$$A \mid B$$

- fornece como resultado um único literal formado pelo conteúdo de A seguido do conteúdo de B.
- Ex.: Se A contém o literal "Ponte " e B contém o literal "Preta", o valor fornecido pela expressão A | B é o literal "Ponte Preta".

Algoritmos Numéricos

- Considere agora que o tipo entendido pelo processador, ou seja, o grupo de objetos permitidos sejam os números naturais.
- Tipo natural = 1,2,3,4,5,...
- Suponha que o processador seja capaz de entender e executar as operações aritméticas adição e subtração e seja também capaz de entender a atribuição e a definição de variáveis sobre este conjunto de números naturais como ações primitivas.

São algoritmos?

1. “natural x;
   x=4;
   x=3-1;
   x=9”
2. “natural x,y;
   x=4;
   x=3+y;
   y=4”

Sumário

Cláusula condicional

- Considere a operação Módulo de um valor. O módulo de um número é por definição:

$$f(x) = -x \text{ se } x < 0$$
$$f(x) = x \text{ se } x >= 0$$

- Portanto, para implementar o algoritmo desta operação, há a necessidade de se fazer um teste, para saber se x é maior, menor ou igual a 0. Pode-se fazer o seguinte teste:

Se x < 0 faz-se f(x) = -x
Se x >= 0 faz-se f(x) = x

- ou ainda:

Se x < 0 então f(x) = -x
senão f(x) = x

Cláusula condicional

- Isto é uma *cláusula condicional*, pois este comando testa uma condição, e sua estrutura segue sempre o seguinte formato:
  Se *expressão booleana* **então** executa ação 1
  senão executa ação 2
- O funcionamento é o seguinte:
  - A *expressão booleana* é uma afirmativa que no momento da execução poderá ter apenas dois valores: *verdadeiro* e *falso*.
  - Se a expressão booleana for *verdadeira*, executa-se a ação 1 logo a seguir da palavra **então**.
  - Se a expressão for *falsa*, executa-se a ação 2, correspondente à palavra **senão**.

Comando Composto

- Imagine o seguinte problema: escrever um algoritmo que dado um valor x, se x = 10, y valerá 15 e z valerá 20, em caso contrário, teremos ambos iguais a 0.
- Para resolver este problema, veja o seguinte algoritmo:

```
“inteiro x, y, z;
imprima (“Digite x ”);
leia (x);
se x = = 10
então
      y = 15
senão
      y = 0;
se x = = 10
então
      z = 20
senão
      z = 0
imprima (“Os valores de y e z são: ”, y, z)”
```

Comando Composto

- Este algoritmo resolve o problema mas são necessários dois testes idênticos. Daí aparece o conceito de *comando composto*: é o comando que consegue agrupar vários comandos em seu interior, sendo que o mesmo é tratado como se fosse um comando só.
- A sintaxe do comando composto é:

```
início
    comando 1;
    comando 2;
    .
    .
    comando n;
fim.
```

Comando Composto

- Resolvendo o problema anterior com este novo conceito, veja como simplifica:

```
“inteiro x, y, z;
imprima (“Digite x: ”);
leia (x);
se x = = 10
então
    início
        y = 15;
        z = 20;
    fim
senão
    início
        y = 0;
        z = 0;
    fim;
imprima (“Os valores de y e z são: ”, y, z)”
```

Sumário

Comandos de Entrada

- Nos algoritmos vistos até aqui, os valores iniciais de uma variável eram sempre colocados como comandos de atribuição dentro do próprio programa:
  “inteiro a, b, x;
  a = 10;
  b = 5;
  x = a+b”
- Isto acarreta o seguinte problema: para cada valor de *a* desejado, deve-se apagar o valor anterior e escrever o novo valor. O ideal seria se a cada expressão do algoritmo, pudesse-se “entrar” com um valor diferente. O *comando de entrada* de dados é LEIA, cuja sintaxe é:

  LEIA (lista de variáveis)

- onde lista de variáveis é uma seqüência de nomes de variáveis separadas por vírgula.

Comandos de Entrada

- Exemplo:
  “inteiro x;
  leia (a,b);
  x = a+b”
- Quando o comando LEIA é executado, as variáveis dentro da lista assumem neste instante um valor determinado, que pode ser qualquer um.
- É equivalente ao comando de atribuição de variáveis, portanto destrói o valor já existente da variável, quando é executado:
  “inteiro a;
  a = 10;
  leia (a);
  a = 30 + a;
  leia (a)”

Comandos de Saída

- Outro problema ocorre quando se quer expor algum resultado, ou seja, deseja-se informar qual o resultado obtido. Para a exposição dos resultados tem-se um comando de impressão IMPRIMA, cuja sintaxe é:

  IMPRIMA (lista de variáveis ou valores)

- onde *lista de variáveis* é igual à do comando LEIA e *lista de valores* pode ser quaisquer valores inteiros ou seqüência de caracteres entre aspas:

  a, 10, “Meu nome é”, “Resultado da soma = ”

Sumário

Comando Repetitivo

- A estrutura de repetição permite que uma seqüência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.
- Esta estrutura é delimitada pelo comando *faça* e pela expressão *até condição* e a interrupção é feita quando a condição for satisfeita, e pelo comando *enquanto condição faça*, onde a interrupção é feita quando a condição deixar de ser satisfeita. A condição de interrupção que deve ser satisfeita é representada por uma expressão lógica.
- Há também a possibilidade de repetição automática, onde as codições iniciais e a condição são estabelecidas no próprio comando.

Comando Repetitivo

- Existem duas formas do comando repetitivo. 1^a forma:
  1. interrupção de início:

     enquanto condição faça
     seqüência de comandos;

     Nesta estrutura, a seqüência de comandos será repetida enquanto a condição for satisfeita. Quando a condição deixar de ser satisfeita, a repetição é interrompida e a seqüência de comandos que vier logo após a seqüência de comandos passa a ser executada.

Comando Repetitivo

- Ex.: Algoritmo que gera os números pares e efetua a soma deles:
  “inteiro par, soma;
  soma = 0;
  par = 100;
  enquanto par <= 200
  início
  soma = soma + par;
  par = par + 2;
  fim
  imprima (soma)“

Comando Repetitivo

- 2^a forma:
  2. interrupção no fim:

faça
seqüência de comandos
enquanto condição

Nesta estrutura, a seqüência de comandos será repetida enquanto a condição for satisfeita. Quando a condição não mais for satisfeita, a repetição é interrompida e a seqüência de comandos que vier após a expressão enquanto condição passa a ser executada.

Comando Repetitivo

- Ex.: o mesmo algoritmo anterior:
  “inteiro par, soma;
  soma = 0;
  par = 100;
  faça
  soma = soma + par;
  par = par + 2;
  enquanto par <= 200;
  imprima (soma)”

Bibliografia Básica I

- Horowitz, E., Sahni, S., and Rajasekaran, S.
  Computer Algorithms.
  Computer Science Press, 1998.
- Ferrer, H.
  Algoritmos Estruturados.
  Editora LTC, 3ª. edição, 1999.
- Tremblay, J.P. e Bunt, R.B.
  Ciência dos Computadores - Uma Abordagem Algorítmica.
  Editora McGraw-Hill, 1983.