MC102 – Algoritmos e Programação de Computadores

1ª Aula – Introdução à Programação de Computadores

1. Objetivos

- Situar a atividade de programação de computadores
- Apresentar conceitos fundamentais relativos à programação de computadores

2. Motivação

A programação de computadores é uma atividade que leva à representação dos passos necessários à resolução de um problema em linguagem de programação. Para dar início ao aprendizado dessa atividade, é importante compreender seu contexto, seu propósito, os conceitos básicos subjacentes, bem como tomar contato com o ferramental necessário a sua realização.

3. Conceitos

3.1 O que é um computador? Para que serve?

(de sofisticada máquina de calcular à mídia)

◊ "Um computador é uma coleção de componentes que realizam operações lógicas e aritméticas sobre um grande volume de dados." (Miyazawa, 2001)

◊ Computador é ferramenta de trabalho (ex. editores de textos, planilhas, sistemas de informação, etc).

◊ Computador é mídia: serve como canal na comunicação humana (ex. FAX, *Web*, ICQ, apresentação multimídia, etc).

3.2 Como funciona um computador?

(organização de um computador)

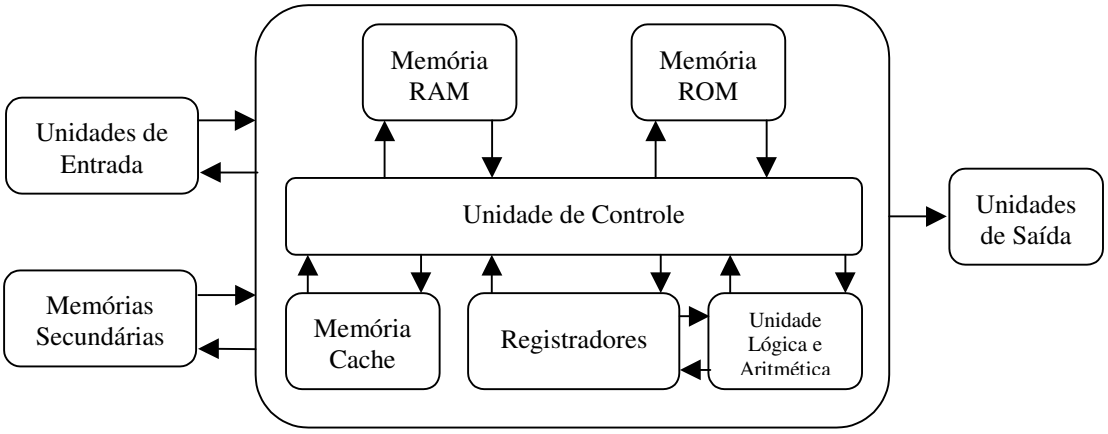

Figura 1: Organização Básica de um Computador Seqüencial (Miyazawa, 2001:1)

3.3 O que é algoritmo? Qual sua relação com programação de computadores?

(definição geral de algoritmo, sua origem, exemplos)

◊ "Um procedimento para resolver um problema matemático (ex. achar o máximo divisor comum) em um número finito de passos que freqüentemente envolve a repetição de uma operação; ou de forma mais abrangente: um procedimento passo-a-passo para resolver um problema ou realizar algum objetivo." (Manber, 1989:1)

Origem: matemático persa Mohammed al-Khowârzimî (em Latim: *Algorismus*)

Algoritmo mais antigo (400 a 300 AC): Algoritmo de Euclides, que calcula o máximo divisor comum (MDC) de dois números inteiros positivos. mdc (x, y) = mdc (y, x mod y); mdc (x, 0) = x.

Entrada: 2 valores inteiros positivos m e n (m > n)

Saída: máximo divisor comum de m e n.

Passo 1: Adote x = m e y = n;
Passo 2: Adote r = (resto de x dividido por y);
Passo 3: Adote novos valores x = y e y = r;
Passo 4: Se r é diferente de 0, volte ao passo 2; senão pare com a resposta x.

Figura 2: Algoritmo de Euclides (Miyazawa, 2001:3)

◊ O enfoque deste curso é em algoritmos computacionais, ou seja, algoritmos que "descrevem uma seqüência de ações que podem ser traduzidos para alguma linguagem de programação" (Miyazawa, 2001:2).

◊ Algoritmo correto: sempre termina e para qualquer instância de entrada produz uma saída correta.

◊ Programar consiste em representar/descrever um algoritmo em alguma linguagem de programação.

3.4 Quais são os ferramentais (básicos) necessários à programação de computadores?

(fluxograma, pseudo-linguagem, linguagem de programação, ambiente de programação)

Fluxograma: auxilia a explicar a seqüência de instruções em algoritmos e programas. Na Figura 2, a seguir, um retângulo representa um passo ou módulo do algoritmo, uma seta indica o próximo comando a ser executado, um losango indica uma condição que interfere no fluxo do algoritmo ou programa.

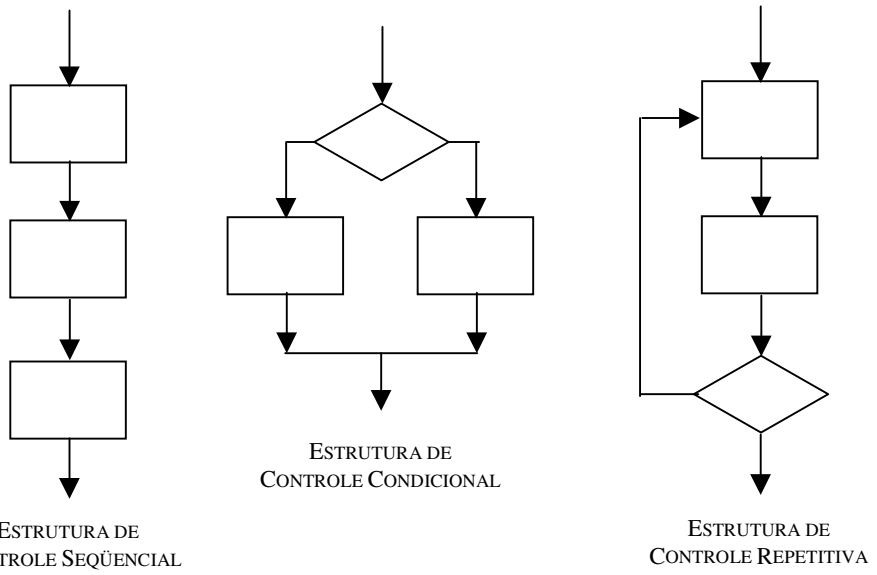

Figura 3: Exemplo de estruturas de controle usadas em programação estruturada (Miyazawa, 2001:9)

Pseudo-linguagem: notação que se assemelha a uma linguagem de programação, mas que também possibilita ao programador concentrar-se no problema a ser modelado sem "se prender" a uma linguagem de programação específica. Essa notação mistura definições formais sobre dados e estruturas de controle, com informações em estilo livre (ver Figura 2).

Linguagem de programação: uma linguagem desenvolvida para viabilizar a programação de computadores.

Ambiente de programação: conjunto de tecnologias que dá suporte à programação de computadores (ex. Sistema Operacional, editor de texto, compilador, etc).

4. Alguns Termos técnicos

Hardware: componentes mecânicos e eletro-eletrônicos que compõem o computador. Parte dura do computador.

Software: seqüência de instruções e comandos que fazem o computador realizar determinada tarefa, também chamados de *programas de computador*. Devem estar armazenados em algum tipo de memória.

Bit: menor unidade de informação de um computador (pode assumir os valores **0** ou **1**).

Bytes: conjunto de oito bits.

Periférico: é qualquer componente do computador (*hardware*) que não seja a CPU. Ex.: leitoras de disquete, monitores, teclados, vídeos, impressoras, etc.

Sistema Operacional: coleção de programas que gerencia e aloca recursos de *hardware* e de *software*. Exemplos de tarefas que um sistema operacional realiza são: leitura de dados pelo teclado, impressão de informações no vídeo, gerenciamento da execução de vários

programas pela CPU, gerenciamento da memória principal e da memória secundária para uso dos programas em execução, etc. Ex.: Linux, Unix, Windows XP, OS2, MS-DOS.

Linguagem de Máquina: conjunto de instruções que podem ser interpretados e executados diretamente pela CPU de um dado computador. É específica para cada computador.

Linguagem Assembler (Linguagem de Baixo Nível): Representação da linguagem de máquina através de códigos mnemônicos. Também é específica de cada máquina.

Linguagem de alto nível: linguagem que independe do conjunto de instruções da linguagem de máquina do computador. Cada instrução de alto nível eqüivale a várias instruções da linguagem de máquina, sendo assim mais produtiva. Ex.: Pascal, C, Algol, BASIC, Lisp, Prolog, etc.

Compilador: tradutor de programas escritos em uma linguagem de programação para programas em linguagem de máquina (ex. GCC). Uma vez que o programa foi convertido para código de máquina, este pode ser executado independente do compilador e do programa original.

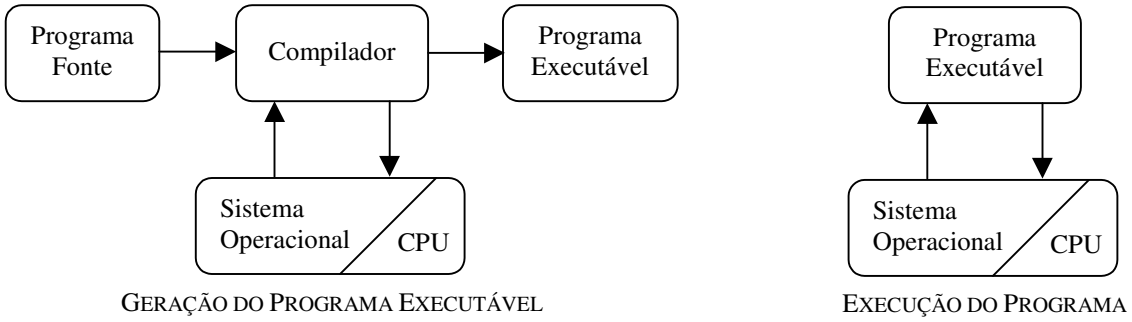

Figura 4: Etapas para execução de um programa compilado (Miyazawa, 2001:3)

Interpretador: é um programa que executa outros programas escritos em alguma linguagem de programação. A execução de um programa interpretado é em geral mais lenta que o programa compilado. Por outro lado, o uso de programas interpretados permite que trechos de código possam ser trocados por novos facilmente, fazendo com que o programa fonte possa mudar durante sua execução. Este é um dos grandes motivos de se usar programas interpretados em sistemas especialistas. Duas linguagens para as quais podemos encontrar interpretadores são Lisp e Prolog.

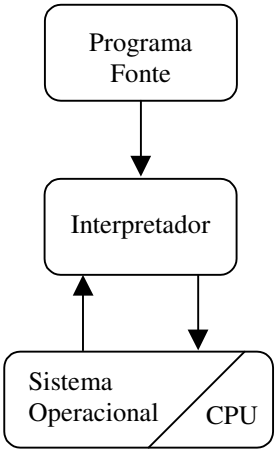

Figura 5: Execução de um programa interpretado (Miyazawa, 2001:3)

5. Exemplos

1) Algoritmo que indica qual dentre dois números é o maior[1].

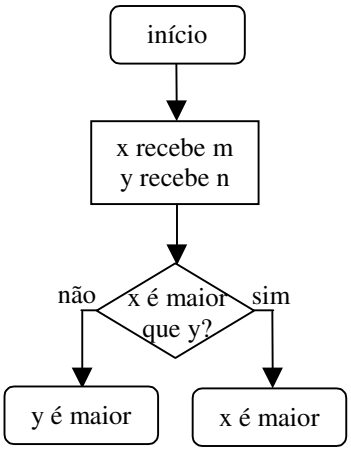

Figura 6: Representação do Algoritmo usando fluxograma

Passo 1: Adote x = m e y = n;
Passo 2: Se x maior que y, então resposta é x; senão resposta é y.

Figura 7: Representação do Algoritmo usando pseudo-linguagem

[1] Há um erro no algoritmo, uma vez que se x não é maior que y, então y pode ser maior ou igual a x. Sugestão: pedir para os alunos corrigirem o erro.

```c
#include <stdio.h>

void main(void) {
   int x, y;
   scanf ("%d, %d", &x, &y);
   if (x > y) {
      printf ("%d é maior.", x);
   }
   else {
      printf ("%d é maior.", y);
   }
}
```

Figura 8: Representação do Algoritmo em Linguagem de Programação C

2) Algoritmo de Euclides, que calcula o máximo divisor comum (MDC) de dois números inteiros positivos. mdc (x, y) = mdc (y, x mod y); mdc (x, 0) = x.

3) Ordenação de uma seqüência aleatória de números.

Referências

Hennessy, John L., Patterson, David A. *Organização e Projeto de Computadores – A Interface Hardware e Software*, 2ª edição, LTC editora.

Manber, Udi. *Introduction to Algorithms: A Creative Approach*, Addison-Wesley, 1989, 478p.

Miyazawa, Flávio K. *Notas de Aula de Algoritmos e Programação de Computadores*, Instituto de Computação, Unicamp, 2001, 169p.

Rubira, Cecília M. F. *Notas de Aula*, 1º semestre de 2000.

Tramontina, Gregório B. *Aula 1 – MC102 Turmas K e L*, 24 de Agosto 2004.