Suplemento para
Bases Computacionais da Ciência

Estes *slides* são um resumo do professor para as aulas homônimas e não devem ser usados separadamente ou como fonte original.

Sugestões são bem-vindas: mande-as para `vinicius@ufabc.edu.br`

Referência da disciplina: MARIETTO, M. et al. *Bases Computacionais da Ciência.* Santo André: UFABC, 2013.

Cap. 1 — Fundamentos da Computação

Importância do raciocínio algorítmico

- Nascimento da civilização: receitas; procedimentos de cálculos; *algoritmos.*
- Ciência avançada e cotidiana, pouco estudada no ensino básico.
- Tese de Church–Turing: linguagens de programação capturam tudo que podemos calcular.

É fundamental:

- saber executar um algoritmo;
- saber expressar-se num;
- saber formular um a partir de uma metodologia intuitiva.

Inutilidade da força bruta: mesmo supercomputadores muito rápidos não dão conta do recado.

Conclusão: algoritmo esperto é melhor que exaustão.

Compare estes métodos — todos são exemplos de algoritmos!

Adição de números

- Agrupando pedrinhas, depois recontando todas a partir de 1...
- Usando algoritmo escolar:

$$\begin{array}{r} 455893476 \\ +230057911 \\ \hline =685951387 \end{array}$$

Multiplicação de números

- Somando o mesmo número repetidas vezes...
- Usando algoritmo escolar:

```
   394
  *121
------
   394
  +788
 +394
------
 =47674
```

Buscas sequencial e binária

Suponha que procuraremos nome em listagem de N pessoas.

- Sem ordenação, deve ser "um a um", até N verificações; com ordenação, também não queremos ir de um em um:

1º	2º	3º	4º	5º	*			

- Busca binária faz cerca de $\log_2 N$ verificações:

			1º	3º	*	2º		

Importância para Humanidades: TrueCrypt (maio de 2014)

Atores: Daniel Dantas, FBI, Edward Snowden, NSA...

Questões envolvidas:

- O que é *software* livre? Gratuito? Por quê?
- Descontinuidade de serviço (gratuito) — Ética;
- liberdade e propriedade autoral — Direito;

- sigilo e segurança particular;
- administração pública e defesa — Políticas Públicas;
- teorias de conspiração;
- hospedagem na Suíça? — Relações Internacionais;
- disponibilidade de peritagem?
- compreensão desse parecer?

História da computação

Algoritmos existem desde o início da matemática.

Para tratar deles, surgiram várias *formalizações* buscando refletir o mesmo conceito.

A mais utilizada é a *máquina de Turing.*

Máquina de Turing

Processador que lê e escreve em uma fita:

- Conforme lê dados, muda entre *estados* e escreve algo.
- É modelado como função matemática: do que lê e do estado em que está.

Qualquer algoritmo moderno pode ser representado em uma máquina de Turing (um pouco contraintuitivo).

Imensa maioria dos processadores reais também são projetados mudando entre estados.

Máquina de Turing universal

É uma máquina de Turing que:

- lê, de sua fita, a descrição de uma máquina de Turing em particular e uma fita para essa máquina;
- *executa* a máquina descrita com essa fita;
- produz o resultado que essa máquina produziria com essa fita.

Mostra que é possível o "computador programável".

Arquitetura de von Neumann

É a realização da máquina de Turing universal no *hardware:*

- trata programas e dados como entidades similares na mesma memória;
- implementa processamento, memória, entrada, saída como humanos visualizam.

Software moderno

O que é um sistema operacional?

Para o usuário, é um "programão" que trabalha direto com o processador e os componentes do computador, fazendo rodar os diversos programas.

Pode ser visto como a parte *soft* da máquina de Turing universal.

Exemplos: Windows, Linux, Mac OS etc.

Quais programas são utilizados na universidade nas disciplinas iniciais?

- navegadores (IE, Firefox, Chrome...) e recursos *web* (TIDIA);
- editores de texto (Word, LaTeX);
- planilhas (Excel, Calc);
- bancos de dados (Access, Oracle, Sophia...).

Exemplos de computação empresarial? Sistemas de companhias aéreas, transportadoras, supermercados etc.

Quando falarmos de Scilab e Robomind, utilizaremos *linguagens de programação*.

São veículos para transmitir e especificar etapas dos algoritmos ao computador.

Podem ser vistas como linguagens apropriadas para dar instruções, como as linguagens naturais (português etc.), mas sem sua ambiguidade.

TIDIA

Siga instruções da ProGrad para criar seu *login* (nome e senha de usuário).

Acesse `http://tidia-ae.ufabc.edu.br`

O ícone de login *está escondido no canto superior direito da tela, dentro do friso esverdeado.*

As turmas de que você participa(ou) estão nas diversas abas.

Passos para inserir uma nova disciplina ou turma:

- Seu professor deve informar qual é o *nome* que ele usou para sua disciplina e turma.
- No menu lateral, clique "Onde participo".
- Próximo às abas, veja "Sites que aceitam inscrição" e busque o nome dado. *Cuidado com muitos parônimos!*

Ferramentas importantes:

Exercícios Atividades Repositório

Mensagens Avisos Forum

Escaninho

Escaninho: avise o prof. se remeter algo por lá.

Cap. 3 — Noções de Estatística, Correlação e Regressão

Leia introdução (em casa), mas foco não são teoria e fórmulas.

Esses assuntos serão detalhados em "Introdução à Probabilidade e à Estatística" (BC 0406).

Ajuste de retas e curvas em "Funções de Várias Variáveis" (BC 0407).

Medidas de posição e dispersão

- Média, mediana e moda descrevem onde dados se aglomeram.
- Média é ponderada por frequências.
- Variância e desvio padrão descrevem afastamento dos dados.
- Desvio tem a mesma *dimensão* dos dados.

Notação: $a \vdash b$ indica os dados no intervalo $[a, b[$.

Em aula:

- Seção 3.4 faz duas atividades passo-a-passo.
- Planilha da Atividade 2 no *website* do docente.
- Histogramas e gráficos de frequência.
- Novidade: Inserir > Objetos > Gráficos.

- Dica: botões que aumentam/diminuem casas decimais.
- Sugestão para dados textuais: funções de contagem ou transformar em números (usual 0 ou 1).
- Seção 3.6 tem exercícios adicionais para aula e casa.

Mas tem próximo *slide!*

Correlação e regressão

Em aula também:

- Seção 3.3.7 faz gráfico de dispersão passo-a-passo.
- Requer duas séries paralelas de dados (duas colunas com mesmo número de dados).
- Reta de regressão ("linha de tendência"): tenha paciência para clicar corretamente no gráfico, tente botões esq. ou dir., $1\times$ ou $2\times$, *em cima dos pontos dos dados* (mudam de cor).
- Complete essa atividade!

Cap. 4 — Base de Dados

Note número de séries paralelas de dados e tipo/valores estritos de cada série.

Importância da consistência dos dados: planilha *versus* banco.

Em aula:

- Seção 4.6 faz uma atividade passo-a-passo.
- Seção 4.8, ex. 9: trabalho com base maior; se planilha travar com subtotais, experimente procedimento com planilha menor, do supermercado.
- Ex. 8 será detalhado em outra disciplina.

- Ambos arquivos de dados (csv) no *website* do docente: navegador abre direto em tela, então faça *download* (botão direito) para p. ex. área de trabalho e depois importe.
- Compare "autofiltro" e "filtro padrão" que permite selecionar várias opções em janela; opção "entre" é chamada "intervalo válido".

Sempre explore seu programa: a planilha mostra quantidade de células selecionadas, total do conteúdo selecionado, células dos "cantos" da tabela de dados...

Cap. 2 — Representação Gráfica de Funções

Dicas para o SciLab:

- No menu do Windows, abra "SciLab NNN", *não o console.*
- Comandos não podem ser modificados, devem ser refeitos (inclusive subsequentes).
- Para não digitar tudo novamente, *volte no histórico* utilizando a tecla "para cima" (análogo a terminais de comando).
- Use ponto decimal (não vírgula).
- Use ";" após cada comando, *especialmente gerando vetores* (*listas de números*), exceto se quiser ver resultado.

- Geração de vetor: `x = inicio : incremento : limite`
- Gráfico é feito calculando e ligando vários pontos.
- Então função deve ser calculada vetorialmente, isto é, repetida a cada valor para toda a lista de números, dito "ponto a ponto".
- São precisos muitos pontos: use `incremento` pequeno (0.01).

- Então expressões mais elaboradas precisam comandos explícitos: ".^" é exponenciação ponto a ponto (V.T. seção 2.3.3).
- Comando `plot(x,f)` trabalha com dois vetores paralelos, não com função "f(x)" (por isso digite só "f", ou chame "y" se for mais claro.

- Acompanhe atividades em aula (2.4) e exercícios (2.6).
- Para não sobrepor gráficos, feche a janela de gráfico entre exercícios.
- Porém, pode ser útil sobrepor gráficos: comparar valores, traçar eixos auxiliares, encontrar raízes, ex. 5...
- Ex. 4: note divisão invertida A\b, específica para matrizes; significa $A^{-1}b$ que é a solução correta para $Ax = b$ (matrizes não comutam).

Começamos a aprender programação!

Prefira digitar os códigos fornecidos, em vez de apenas copiar e colar, para melhor fixação, visualização de dúvidas e aprendizagem passo a passo.

Cap. 5 — Variáveis e Estruturas Sequenciais

Destaque para Scilab: visualizar conceitos algorítmicos de modo mais claro.

Trabalhar também com situações mais simples do Robomind.

Significado do "texto puro" para uso do código-fonte.

Conceito de instrução sequencial: usamos, de fato, desde a 1ª aula.

Sobre o Robomind

- Orientação do robô difere da orientação da tela: o robô anda em frente em relação à sua posição, não à da exibição no monitor.
- Compare os comandos andarFrente() e andarNorte().
- Seção 5.4: um exercício resolvido e alguns propostos.
- Ex. 2, 3 são rápidos.

- Ex. 4 é simples, mas não é rápido.
- Ex. 5 e 6 são bastante demorados, mas são necessariamente difíceis?
- Tente também os ex. 7 e 8.
- Robomind é retomado nos quatro exercícios da subseção 5.7.1.

Use o Bloco de Notas (ou o gedit em Linux) para prover os mapas para o robô.

Não precisa digitar linhas que começam com #: são apenas "comentários" não lidos pelo programa.

Seu próprio mapa: *documente-o* explicando funcionamento em comentários ao longo de todo o código-fonte.

Salve o texto finalizado com "Salvar Como" e extensão "map" (eg.: atividade.map).

Com o Scilab: Seção 5.5

Também usará código-fonte, inserido à parte, não no console.

Editor próprio de arquivos:

- Ícone do disquete para salvar arquivo já nomeado, depois de modificações.
- Ícone para "salvar e executar" nem sempre funciona.

Dê especial atenção aos exercícios em 5.7.2.

Comandos de entrada e saída de dados: para não editar código em cada execução (potencializaria eventuais erros).

`input` escreve sentença para informar usuário e retorna valor digitado à posição que ocupa no comando.

`printf` tem origem na linguagem C.

1º argumento é sequência de caracteres entre aspas, alguns especiais para identificar onde inserir informações:

- `%f` para escrever número real (*floating*);
- `%d` para escrever número inteiro (decimal);
- `\n` para iniciar uma nova linha.

Argumentos seguintes, separados com vírgulas, um para cada sinal `%f` ou `%d`, com variável contendo valor fracionário ou inteiro, resp.

Cap. 6 — Estruturas Condicionais

Duas novas instruções: “se” e “se–senão”.

Permitem selecionar grupos de instruções para executar com base em um teste lógico.

Blocos de comandos são delimitados por chaves (Robomind) ou palavras-chave (Scilab) para dizer ao programa o que separar do fluxo geral.

Condição lógica deve ser uma só, mas pode ser "composta": construída a partir de outras mais simples com *conectivos:*

- negação: `não`
- conjunção: `e`
- disjunção: `ou`

Disjunção é inclusiva: `P ou Q` é verdadeira sempre que uma ou ambas `P`, `Q` são verdadeiras; é falsa somente quando ambas `P`, `Q` são falsas.

“se” simples

Executa bloco somente se condição lógica for verdadeira; ignora tudo se for falsa.

No Robomind:

```
se (condição lógica)
    {
        comando 1
        comando 2
        ...
        último comando
    }
```


"se–senão"

Executa só 1º bloco se condição for verdadeira e só 2º se for falsa.

No Robomind:

```
se (condição lógica)
   {
      alguns comandos
   }
senão
   {
      outros comandos
   }
```

Confira na listagem 6.5.

Scilab

Compare construções com listagens 6.9 e 6.10: `if`, `else`, `end`.

Conectivos: `~`, `&`, `|`

Comparativos:

- igualdade: `==`
- diferença: `~=`
- desigualdades estritas: `<`, `>`
- desigualdades não estritas: `<=`, `>=`

Destaque para 2º exercício de 6.6.2.

No 6º exercício de 6.6.2, usa-se uma construção com vários testes seguidos.

Os testes podem ser *encadeados*, usando duas opções:

- `else` seguido de novo `if` (repita um `end` para cada `else`)
- comando especial `elseif`

```
if (condição lógica)
   alguns comandos
elseif (outra condição lógica)
   outros comandos
elseif (nova condição lógica)
   novos comandos
end
```

Cap. 7 — Estruturas de Repetição

Um bloco de comandos será repetido quantas vezes for necessário.

Nomes comuns: laço, ciclo, *loop*, repetição, iteração.

No Robomind:

```
repetir (número)
  {
    comando 1
    comando 2
    ...
    último comando
  }
```

Repetirá o bloco:

- um número dado de vezes,
- ou continuamente na forma `repetir ()`.

Repetição condicionada:

```
repetirEnquanto (condição lógica)
{
    comando 1
    comando 2
    ...
    último comando
}
```

(Em todos esses comandos, não esquecer as chaves!)


```
while (condição lógica)
   alguns comandos
end
```

(Não esqueça o `end`.)

Uma construção ubíqua:

```
for (variavel=inicio:passo:fim)
   alguns comandos
end
```

(Use `end`; pode omitir `:passo` se for 1.)

Corresponde a:

```
variavel=inicio;
while (variavel<=fim)
   mesmos comandos
   variavel=variavel+passo;
end
```


Exemplo: cálculo do fatorial $n! = 1.2.\ldots.(n-2)(n-1)n$.

```
resultado=1;
item=1;
while (item<=n)
   resultado=resultado*item;
   item=item+1;
end
```

ou ainda:

```
resultado=1;
for (item=1:n)
   resultado=resultado*item;
end
```

Em cada exemplo, a variável `resultado` armazena o produto *parcial* até o final da repetição.

Destaques

- Seção 7.2.
- Seção 7.4: ex. 1 e 2.
- Seção 7.5: experimente listagens 7.5, 7.7, 7.9 e 7.6 (nessa ordem).
- Subseção 7.7.2: ex. 3, 5, 6, 11.

Caps. 8 e 9 — Modelagem e Simulação Computacional

Destaques

- Uso das estruturas algorítmicas aprendidas.
- Oportunidade para revisá-las com contexto e significado.
- Fluxograma: veículo importante de comunicação. Compare exemplos com o código e a execução passo a passo pelo Robomind.

No Capítulo 8

- Bastante leitura informativa!
- Pág. 190 menciona derivação (não se preocupe; assunto de BM e FUV): $f'(x)$ é uma outra função, obtida a partir de $f(x)$.
- Experimente os *scripts* como prática das aulas anteriores.
- S. 8.5.1 sobre método de Monte Carlo e estimativa de π.
- S. 8.5.2 sobre resgate: compare código com fluxograma e com execução no computador (Robomind destaca o comando em execução).
- S. 8.7.1 sobre labirinto: *ditto;* note que algoritmo é simples.

No Capítulo 9

Estude a subseção 9.3.2 integralmente (leia o conteúdo e digite os comandos em vez de copiar e colar).

Na listagem 9.2

- Variável `tf` não é usada.
- Há conflito entre dois usos de `t`: chame o primeiro de `tt` (tempo total) e corrija linhas 9, 10, 11 e 17 (com `for t=0:dt:tt`).
- `cosd` e `sind` são cosseno e seno com ângulo em graus (*degrees*).
- `x=zeros(numero)` cria um vetor `x` com `numero` coordenadas, que são todas nulas inicialmente. Precisamos disso para, no laço abaixo, preencher cada coordenada `x(i)` com um valor específico.

Na listagem 9.3

- `xdel` apaga a janela de gráficos.
- `%f` e `%t` fora do `printf` são valores booleanos "falso" e "verdadeiro"
- Troque `=="s"w` por `=="s"`
- Troque `%.2s` por `%.2f␣s`