MC-102 — Aula 01
Introdução à Programação de Computadores

Anderson Santos e Pedro Moura

Instituto de Computação – Unicamp

2017

Roteiro

O que vamos aprender neste curso

- Neste curso vocês aprenderão como usar um computador para resolver problemas.
- (a) Definiremos um problema a ser revolvido, (b) descreveremos uma solução imperativa e (c) implementaremos esta solução criando um programa.
- Uma solução imperativa descreve uma sequencia de comandos e passos que devem ser executados para se resolver um problema.
- Uma solução imperativa para um determinado problema é conhecida como **algoritmo** para se resolver o problema.

O que vamos aprender neste curso

- Suponha o problema de se encontrar a raiz quadrada de x.
- Abaixo temos uma solução imperativa (algoritmo) para encontrar $\sqrt{x}$ de forma aproximada:
  1. Comece com uma solução inicial i (exemplo $x/2$)
  2. Enquanto i^2 não for próximo o suficiente de x faça
     - ★ calcule novo i como $\frac{(i+x/i)}{2}$
- Este algoritmo usa o conhecido método de Newton e pode ser demonstrado matematicamente que converge para a raiz de x.

O que vamos aprender neste curso

Algoritmo:

1. Comece com uma solução inicial i (exemplo $x/2$)
2. Enquanto i^2 não for próximo o suficiente de x faça
   - calcule novo i como $\frac{(i+x/i)}{2}$

Definimos que i^2 está próximo de x quando $|i^2 - x| < 0.1$.

Exemplo: raiz quadrada de 9:

1. $i = 4.5$, mas $i^2 = 20.25$ não está próximo de x, então atualizamos $i = (i + x/i)/2 = 3.25$
2. $i = 3.25$, mas $i^2 = 10.56$ não está próximo de x, então atualizamos $i = (i + x/i)/2 = 3.0096$
3. $i = 3.0096$ e $i^2 = 9.057$ estando próximo de x, portanto finalizamos o processo.

Achamos a solução aproximada 3.0096.

O que vamos aprender neste curso

- Aprenderemos principalmente a criar algoritmos para resolver pequenos problemas.
- Um algoritmo pode ser descrito de várias formas, dentre elas, em português como vimos, ou em uma linguagem de programação.
- Neste curso também aprenderemos a **implementar** um algoritmo em uma linguagem de programação específica que é a linguagem Python.
- A vantagem de se implementar um algoritmo em uma linguagem de programação é que podemos a partir daí, criar um programa que usa o computador para resolver o problema.

Por que aprender algoritmos e programação?

- Neste curso vocês aprenderão a criar algoritmos e programas para resolver problemas.
- Criar algoritmos e programar é uma atividade básica de um cientista ou engenheiro da computação.

Por que aprender algoritmos e programação?

- *Eu não sou da computação !!!*
- Possíveis Respostas:
  - Porque é legal!
  - Posso ter algum retorno financeiro com isso!
  - Mais importante: conhecimento...

Por que aprender algoritmos e programação?

Eu sou das engenharias!
Alguns exemplos:

- Como engenheiro você deverá ser capaz de automatizar algum processo.
  - Você poderá criar algoritmos e programas para gerenciar e automatizar algum processo que hoje é manual.
- Como engenheiro você deverá ser capaz de desenvolver novas ferramentas ou protótipos.
  - Para criar ferramentas/protótipos você deverá fazer simulações para a realização de testes preliminares.
- Você poderá enxergar situações onde uma solução computacional pode trazer benefícios.
  - Mesmo que você não crie a solução você poderá propô-la e será capaz de "conversar"com o pessoal de TI para implementar a solução.

Por que aprender algoritmos e programação?

Eu sou das áreas científicas! Matemática, Física, Química etc.
Alguns exemplos:

- Como cientistas vocês devem propor uma hipótese e testá-la.
  - Em vários casos onde os sistemas podem ser " modelados matematicamente", são criados algoritmos que fazem a simulação do sistema para checagem de uma hipótese.
- Você deverá resolver sistemas complexos de equações que não necessariamente podem ser resolvidos por softwares padrões (como MatLab).
  - Vocês deverão implementar seus próprios resolvedores.
- Simulações.
  - Muitos dos modelos propostos para explicar algum fenômeno são simulados computacionalmente. Implementar os modelos é uma tarefa básica.

O que esperar deste curso

- Vocês aprenderão o básico para criar algoritmos e desenvolver programas.
- Utilizaremos a linguagem Python para descrição dos algoritmos.
- Vocês **NÃO** vão aprender a usar programas neste curso (como office, etc).
- Vocês **VÃO** ter porém, uma boa noção de como criar programas como o office, etc.

O que será necessário

- Você deverá ter acesso a um computador onde o Python 3 esteja instalado.
- Pode-se instalar Python baixando a versão para o seu sistema operacional favorito em `https://www.python.org/`.
- Para escrever um algoritmo em Python, utilizamos um **editor de texto** simples como *emacs, kyle etc*.
- Pode-se também instalar uma *IDE (Integrated Development Environment)* como *PyCharm* para escrever um algoritmo em Python.

O que será necessário

- Como um desenvolvedor será muito importante você encontrar soluções para problemas técnicos.

Technical Sophistication: A capacidade de resolver problemas técnicos.

- Uma parte importante da construção da sua *Sofisticação Técnica* é buscar por informações na Web sobre problemas técnicos, como exemplo, problemas de instalação do Python ou do PyCharm.
- Aprendam a usar o google!!

O que será necessário

Para ir bem neste curso:

- Faça todos os laboratórios.
- Faça e implemente as listas de exercícios.
- E finalmente faça e implemente as listas de exercícios.

O que será necessário

Para ir bem neste curso:

- Faça todos os laboratórios.
- Faça e implemente as listas de exercícios.
- E finalmente faça e implemente as listas de exercícios.

O que será necessário

Para ir bem neste curso:

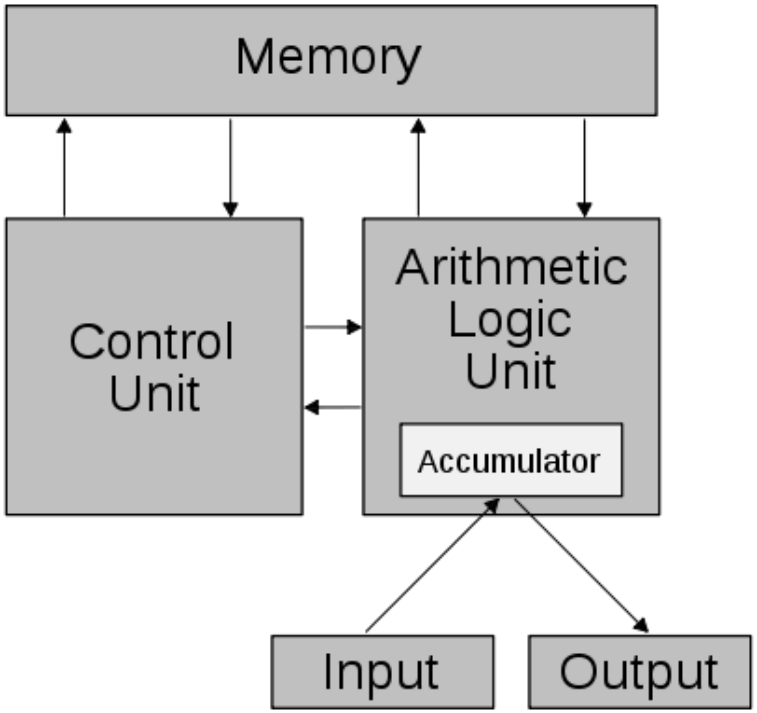
- Faça todos os laboratórios.
- Faça e implemente as listas de exercícios.
- E finalmente faça e implemente as listas de exercícios.

O que é um computador?

- Um computador é uma máquina que, a partir de uma entrada, realiza um número muito grande de cálculos matemáticos e lógicos, gerando uma saída.
- Os computadores fazem isto muito bem e muito rápido. Computadores modernos fazem centenas de milhares de cálculos por segundo.
- **Exemplo:** Enquanto leio esta frase um computador típico executou mais de 1 Bilhão de instruções.

Hardware e dispositivos

- Usualmente chamamos de **Hardware** todos os dispositivos físicos que compõem um computador, como CPU, Disco Rígido, Memória, etc.
- Estes dispositivos seguem uma organização básica como na figura (Arq. de Von Neumann).



Hardware e dispositivos

Todo o hardware opera com sinais digitais: sem energia e com energia. Normalmente usamos valores 0 e 1 para representar isto.

- Chamamos estes sinais de **Bit** $\rightarrow$ Valores 0 ou 1.
- Chamamos de **Byte** $\rightarrow$ um agrupamento de 8 bits.
- Todas as informações armazenadas no computador são representadas por números 0s e 1s. Informações como letras, símbolos, imagens, programas são todas vários 0s e 1s.

Software

Softwares são os programas que executam tarefas utilizando o hardware de um computador.

- Os softwares são compostos por um conjunto de instruções que operam o hardware.
- Temos abaixo, por exemplo, três instruções para um computador de 32 bits.
- Um software é composto por milhares de instruções deste tipo.

```
0100 0010 0011 0101 0101 0100 0011 0110
0100 1110 1100 1100 1001 0110 0110 1000
0000 0101 1111 1110 1101 0011 0000 1100
```

Organização básica de um ambiente computacional

- Um ambiente computacional é organizado como uma pilha, onde cada item da pilha realiza tarefas bem específicas.
- Items acima na pilha fazem uso de soluções propostas pelos items abaixo.

Programas de Aplicação
Compiladores/Interpretadores
Sistema operacional
Hardware

Organização básica de um ambiente computacional

Programas de Aplicação.

- Como usuários, interagimos com os programas de aplicação.
- Neste curso iremos descer nesta hierarquia, para construirmos novos programas de aplicação.
- Para construir novos programas podemos escrever diretamente códigos digitais que serão executados por um computador.
- Mas usaremos uma linguagem de programação específica e um interpretador para executar o nosso programa.

Programas de Aplicação
Compiladores/Interpretadores
Sistema operacional
Hardware

Organização básica de um ambiente computacional

Compiladores/Interpretadores e Linguagens de Programação.

- Uma **linguagem de programação** é um conjunto de comandos que são mais "próximos" da linguagem humana do que os sinais digitais.
- Neste curso usaremos a *linguagem de programação Python*.

Organização básica de um ambiente computacional

Compiladores/Interpretadores e Linguagens de Programação.

- Podemos classificar as linguagens de programação em dois tipos: **compiladas** e **interpretadas**.
- Uma linguagem compilada, como C, usa um **compilador** para transformar o código em C para um programa executável.
- O compilador realiza esta tarefa juntamente com um **assembler**.
- Somente depois de se compilar o programa e que pode-se executá-lo.

```
for(i=0; i< 10; i++)
    c = a+b;
```

```
loop:  add c, a, b
       add i, i, 1
       bnq i, 10, loop
```

```
0100 0010 0011 0101 0101 0100
0110 0110 0111 0101 0101 0100
1111 0000 0111 0101 0101 0100
```

Organização básica de um ambiente computacional

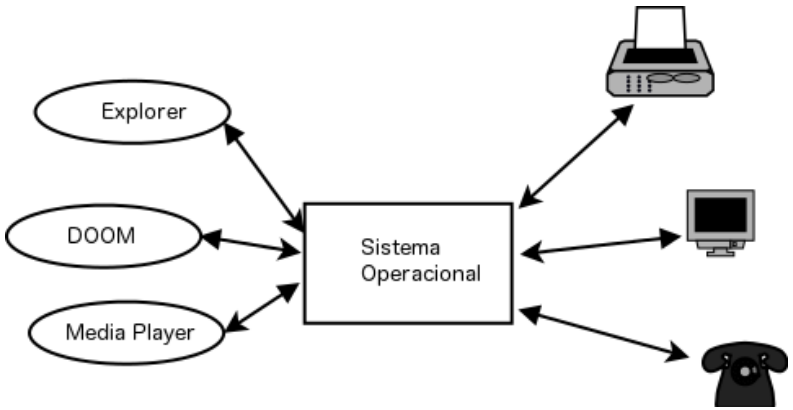
Compiladores/Interpretadores e Linguagens de Programação.

- Python é uma linguagem interpretada, e portanto um **interpretador** lê os comandos da linguagem e os executa diretamente transformando cada comando em linguagem de máquina.

Organização básica de um ambiente computacional

Sistema Operacional.

- Os programas possuem instruções que são executadas no hardware.
- Mas o acesso ao hardware, como disco rígido, memória, processador, é controlado por um software especial conhecido como **sistema operacional.**
- O sistema operacional é o responsável pelo controle do hardware, incluindo segurança, gerenciamento de memória, dentre outros.
- Exemplos de sistema operacionais: *Windows, OS X, Linux, Android, iOS.*



Algoritmos

Antes de criarmos um programa para resolver um determinado problema, devemos ser capazes de especificar um algoritmo para resolver o problema.

- Algoritmo: Seqüência de passos, precisos e bem definidos, para a realização de uma tarefa ou resolução de um problema.
- Algoritmos podem ser especificados de várias formas, inclusive em português.

Exemplo de algoritmo

Como ordenar as cartas de um baralho?

De algoritmos a programas

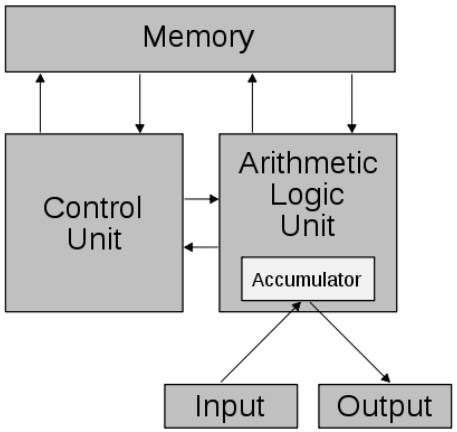
- Depois de criarmos um algoritmo, podemos especificar este em uma linguagem de programação.
- Usaremos a linguagem Python para descrever os algoritmos.
- O programa em python será executado diretamente invocando-se o interpretador python para executar o programa.

Um pouco de história

- Os primeiros computadores eram do tipo **programas-fixo**. Isto significa que cada computador era desenvolvido para resolver um problema específico, como cálculos de projéteis durante a Segunda Guerra.
- Desta forma, o programa era diretamente gravado como parte do hardware do computador e não podia ser alterado.

Um pouco de história

- A arquitetura de Von Neumann foi um marco importante pois trouxe a idea de um **computador com programa armazenado**, ou seja, a ideia de que os computadores deveriam ter um conjunto pré-determinado de instruções que o operavam, e que os programas fossem construídos utilizando-se tais instruções. Haveria uma memória para armazenar tanto o programa quanto os dados necessários.
- Com isso foi criado os computadores de uso geral, pois eles podem ser **programados** para resolver os mais diversos tipos de problemas.



Um pouco de história

- Bem antes do surgimento dos primeiros computadores de uso geral, Alan Turing descreveu matematicamente um computador hipotético chamado **Máquina de Turing**, que serviu para formalizar os conceitos de algoritmos e o que é computável.
- A **tese de Church-Turing** diz que se uma função é computável então existe uma Máquina de Turing que computa esta função
- Você pode pensar simplificadamente que uma função ser computável significa um problema admitir um algoritmo que o resolva.

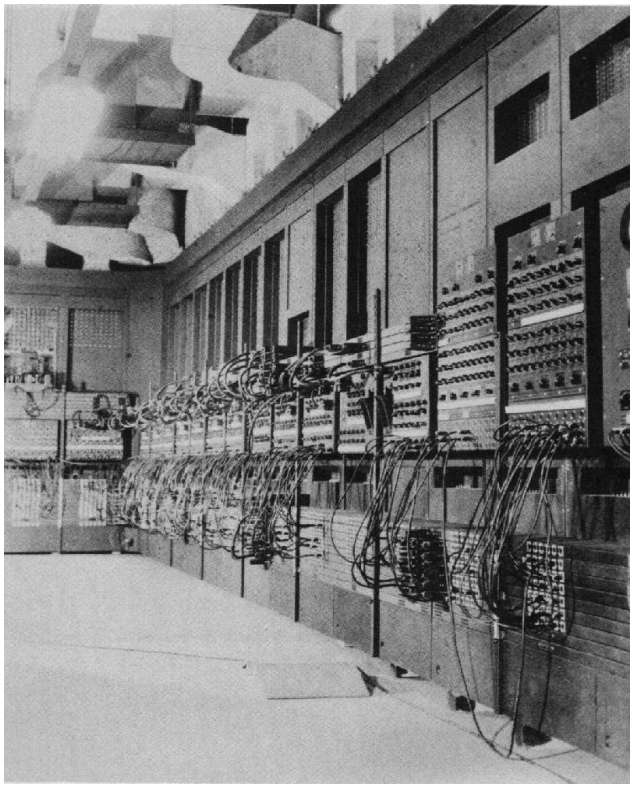
Um pouco de história

- A partir da tese de Church-Turing temos que qualquer problema que possa ser resolvido por um algoritmo (por um computador) pode ser resolvido com uma máquina de Turing.
- Apesar de computadores serem muito poderosos, Turing mostrou que existem problemas que não podem ser resolvidos por uma Máquina de Turing (e por consequência nenhum computador).
- Um destes problemas é o conhecido **problema da parada**: dado um programa de computador, você deve criar um algoritmo que decide se o programa de computador encerra ou nunca para.
- Turing mostrou ser impossível a criação de uma Máquina de Turing para este problema.

Um pouco de história

Os primórdios da programação: programação em código absoluto ou binário (apenas 0s e 1s).

ENIAC



Um pouco de história

Uma melhoria: A Linguagem Assembly.

- Cria-se uma linguagem de baixo nível (Linguagem Assembly) para representar as instruções em código binário.
- Um programa, chamado montador ou assembler, faz a transformação em código absoluto.

```
LOOP:   MOV A, 3
        INC A
        JMP LOOP
```

Um pouco de história

Uma brilhante idéia: Criação de linguagens de alto nível e compiladores/interpretadores para estas.

- Mais distantes da máquina e mais próximas de linguagens naturais (inglês, português, etc.).
- Mesmo mais compreensíveis, elas não são ambíguas.
- Um compilador/interpretador as transforma em código executável.

Exemplos de linguagens

- C
- Python
- Java

Primeiro programa em Python

Um programa em Python é um arquivo texto, contendo declarações e operações da linguagem. Isto é chamado de *código fonte*.

```
print("Ola turma de MC102!!")
```

Você pode salvar este arquivo como hello.py.

Como executar este programa

- Para executar este programa basta abrir um *terminal* e escrever:

```
$ python hello.py
Ola turma de MC102!
```

Relembrando

- Algoritmos.
- Implementação em uma Linguagem de Programação.
- Hardware e Software.
- Pilha de um ambiente computacional: Programas de Aplicações, Compilador/Interpretador, Sistema Operacional, Hardware.
- Código Binário, Assembly, Linguagem de Alto Nível.