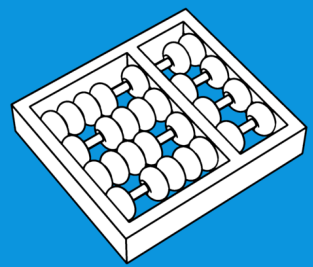
Apresentação da disciplina

MC102 - Algoritmos e Programação de Computadores

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

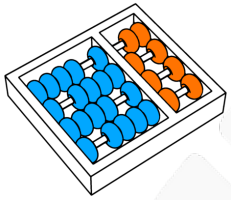
0

UNICAMP

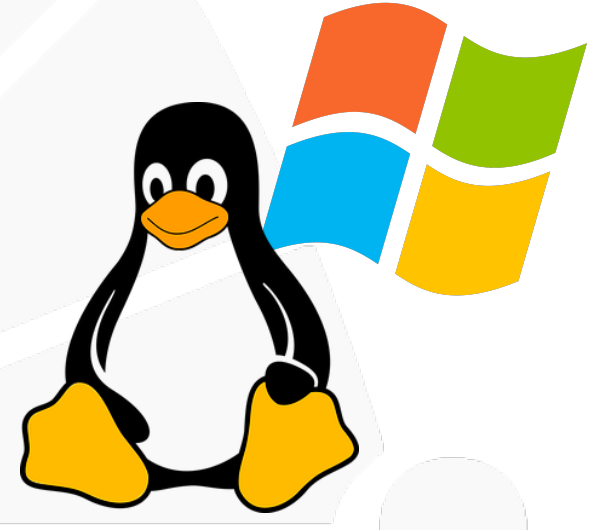


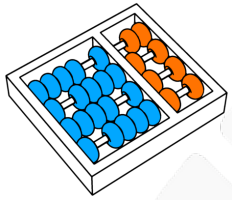
"Ciência da computação tem tanto a ver com o computador como a astronomia com o telescópio, a biologia com o microscópio, ou a química com os tubos de ensaio. A Ciência não estuda ferramentas, mas o que fazemos e o que descobrimos com elas."

Atribuída a Edsger W. Dijkstra.



Áreas de aplicação da computação

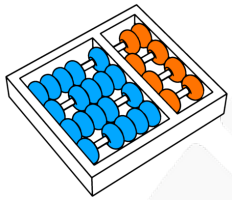




O que faz um computador?

- Executa programas?
- O que são programas?
- Como os programas são criados?

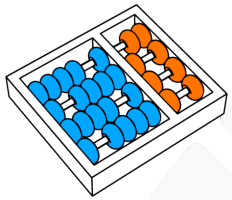
Algoritmos



O que é um algoritmo?

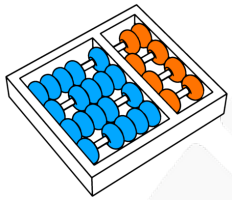
Um **ALGORITMO** pode ser descrito, informalmente, como uma **SEQUÊNCIA LÓGICA, FINITA E DEFINIDA DE AÇÕES** capazes de resolver um problema.

O conceito de algoritmo vai além da programação.



Algoritmos em tarefas comuns

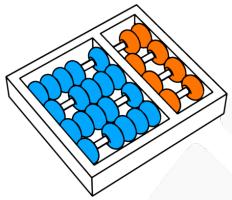
- Mascar chiclete.
- Fritar um ovo.
- Trocar pneu furado.
- Trocar uma lâmpada queimada.
- Calcular a média de duas notas.



Algoritmos em tarefas comuns. Sugestões

Mascar chiclete:

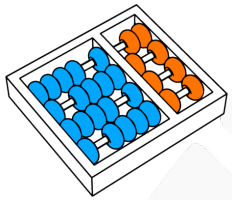
1. Pegar o chiclete.
2. Retirar o papel.
3. Jogar o papel no lixo.
4. Colocar o chiclete na boca.
5. Enquanto o chiclete tiver sabor:
   Mastigar o chiclete.
6. Jogar o chiclete no lixo.



Algoritmos em tarefas comuns. Sugestões

Calcular a média de duas notas:

1. Ler a primeira nota.
2. Ler a segunda nota.
3. Somar as duas notas.
4. Dividir a soma por 2.
5. Imprimir o resultado da divisão.

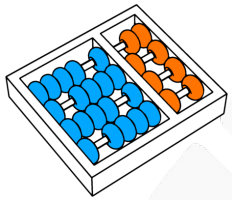


O que é imprescindível na proposta de um algoritmo?

- Compreender os requisitos do problema.
- Identificar as entradas.
- Identificar as saídas.
- Definir uma sequência lógica e finita de ações para solucionar o problema.
- Garantir clareza e precisão dessas ações.
- Garantir que as saídas sejam corretas.

O que acontece quando não verificamos esses itens?

O algoritmo pode ser escrito também em pseudocódigo (mais comum) ou com um fluxograma.

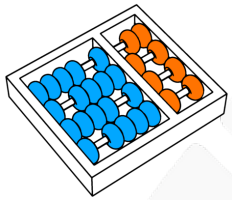


Pseudocódigo

Um pseudocódigo é uma forma abstrata de escrever o programa

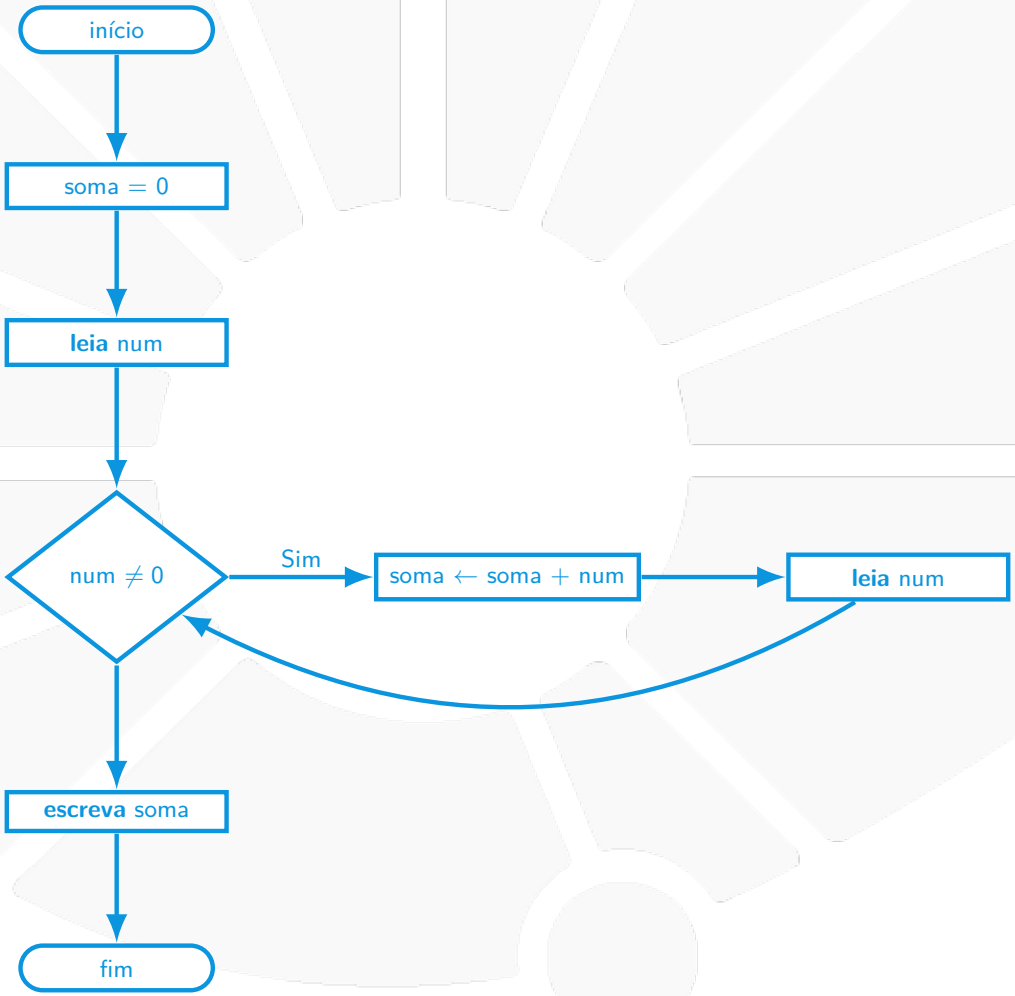
- Não estamos preocupados com a linguagem.
- Mas sim com a solução do problema.
- Também com a clareza.

```
1 soma ← 0
2 leia num
3 enquanto num ≠ 0
4 |   soma ← soma + num
5 |   leia num
6 escreva soma
```

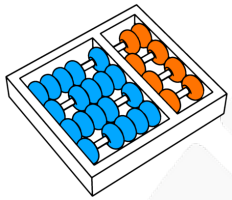


Fluxograma

A ideia é representar as instruções graficamente:



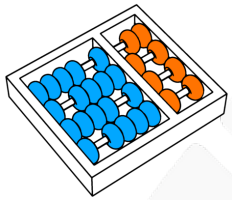
Programação



Programa

Um **PROGRAMA** é uma sequência de instruções que especificam como executar uma computação

- Ex: calcular a soma de números até que o zero seja digitado.
- É escrito em uma determinada linguagem:
  - Chamada de linguagem de programação.
  - Ex: Python, C, C++, Java, etc. . . .
- O texto do programa é chamado de **CÓDIGO FONTE**.



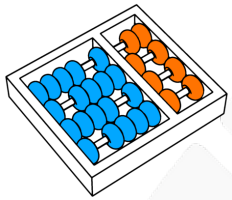
O que é uma linguagem de programação?

- Meio de comunicação:

Indivíduo que deseja solucionar um problema

Computador

- Deve **LIGAR** o **PENSAMENTO HUMANO** com a **PRECISÃO** requerida para o **PROCESSAMENTO DA MÁQUINA**.



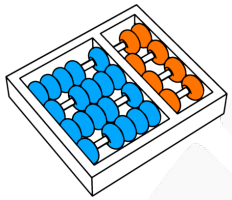
Execução de programas

Código escrito em uma LP

(Programa)

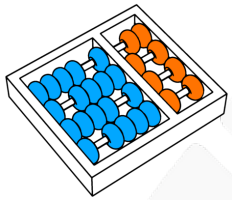
Código de máquina

(Executável)



Como classificar LPs?

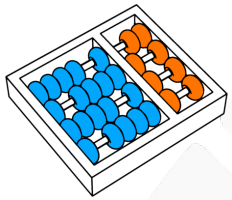
- Em relação ao **PARADIGMA**:
  - Imperativo, funcional, lógico, orientado a objetos etc.
- Em relação ao **NÍVEL**:
  - Baixo, Médio ou Alto.



O que é o paradigma?

MODELO INTERPRETATIVO ou conceitualização de uma **REALIDADE**.

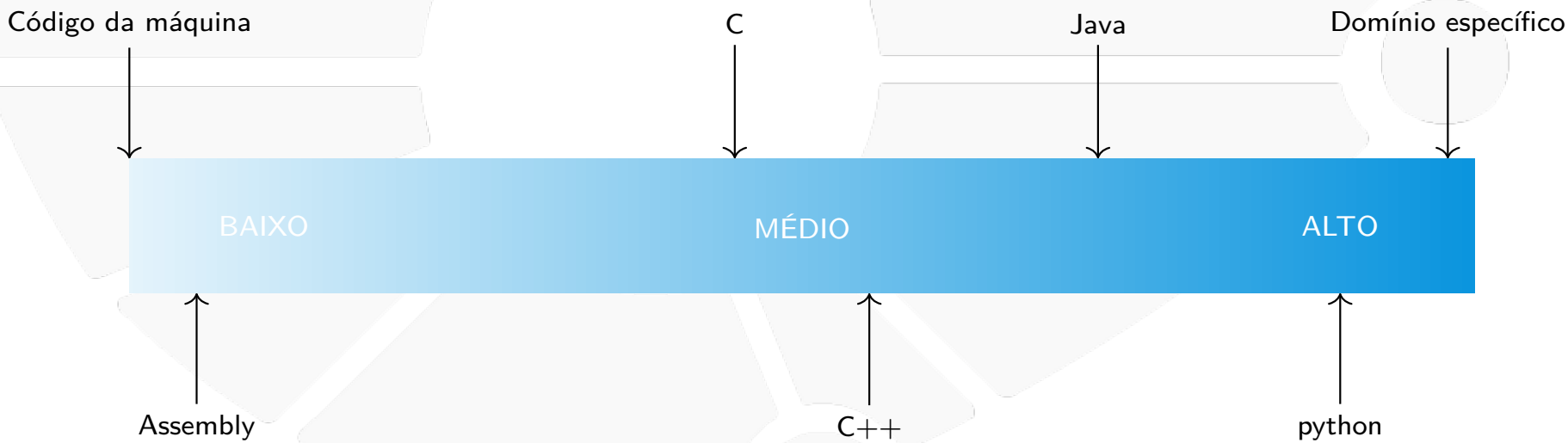
Um **PONTO DE VISTA** que determina como uma **REALIDADE** é entendida.

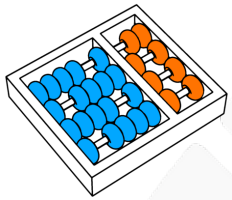


O que é o nível?

O **NÍVEL** de uma LP é um indicativo da **CAPACIDADE DE ABSTRAÇÃO** em relação ao hardware e o código da máquina que a linguagem oferece ao programador.

A classificação por nível pode ser considerada como um espectro:





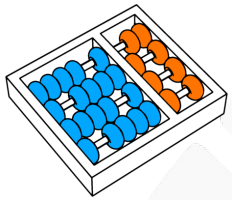
Exemplo de duas linguagens diferentes

Python

```python
soma = 0
num = int(input())
while(num != 0):
    soma = soma + num
    num = int(input())
print(soma)
```

C

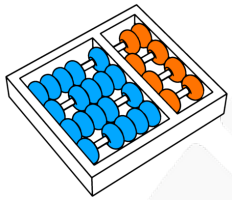
```c
#include <stdlib.h>
int main() {
    int soma = 0, num;
    scanf("%d", &num);
    while(num != 0){
        soma = soma + num;
        scanf("%d", &num);
    }
    printf("%d", soma);
    return 0;
}
```



Fases para elaborar um programa

- Coleta, análise e especificação de requisitos.
- Algoritmo.
- Implementação.
- Teste.
- Manutenção.

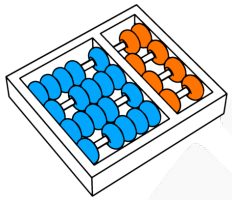
A disciplina



Objetivos

Introduzir o uso do computador na resolução de problemas através de desenvolvimento de programas em linguagens de alto nível.
Para isto faremos:

- Análise do problema.
- Escolha de uma solução.
- Projeção de um algoritmo para a solução escolhida.
- Implementação do algoritmo na linguagem de programação **Python**.
- Teste e execução da solução implementada.



Objetivos

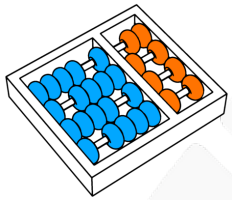
Por que Python?

- Linguagem de programação de alto nível, muito intuitiva.
- Popular e com constante crescimento na comunidade:
  - 1^{ra} no índice TIOBE.
  - 2^{da} no GitHub.

Estamos interessados não apenas em aprender Python:

- Mas aprender a **PROJETAR ALGORITMOS**!
- Você ainda será um programador se Python ficar obsoleto!

O algoritmo não depende da linguagem de programação! Pode ser usado e reutilizado nas mais variadas linguagens!



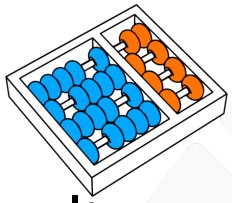
Carga horária

6 horas semanais divididas em três encontros de 2 horas cada:

- 1 encontro com aulas teóricas nas terças, das 21h às 23h.
- 1 encontro com aulas teóricas nas quintas, das 19h às 21h.
- 1 encontro com aulas práticas nas sextas, das 21h às 23h.

IMPORTANTE:

- A aprovação depende de 75% de presença!!!
- Dedique mais tempo, 6 horas semanais não é suficiente!!
- Não se conforme com entregar o mínimo solicitado!



Avaliação. Cálculo da média final

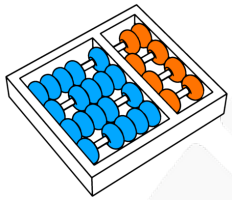
Itens que serão avaliados:

- (L) Laboratórios.
- (P) Projetos.
- (T) Testes.
- (D) Dúvidas de aula.

Cálculo da média final:

$$\mathbf{M} = \begin{cases} \min(\mathbf{L}, \mathbf{P}, \mathbf{T}), & \text{se } \mathbf{L} < 5,\ \mathbf{P} < 5 \text{ ou } \mathbf{T} < 10 \\ \frac{6{,}0\mathbf{L} + 2{,}5\mathbf{P} + 1{,}0\mathbf{T} + 0{,}6\mathbf{D}}{10}, & \text{caso contrário.} \end{cases}$$

Alunos com $0 \leq \mathbf{M} < 5$ e pelo menos 75% de frequência poderão fazer entregas atrasadas de laboratórios e testes com desconto de 50%.



Avaliação. Laboratórios

Laboratório (lab) é como chamamos os exercícios de programação:

- Vários durante o semestre (≈ 1 por semana).
- Individuais e ser entregues até uma data estipulada.

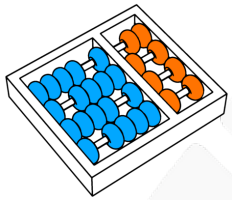
A nota é proporcional ao número de casos de teste resolvidos:

- Mas pode sofrer descontos.
- Pela qualidade do programa, por não cumprir critérios do enunciado,...

O aluno tem uma **segunda chance** para entregar os laboratórios:

- Se você **não entregou** até a data estipulada, ou se entregou com nota $\leq 4{,}9$, a sua segunda chance terá 20% de desconto (nota máxima 8).
- Se você entregou com nota ≥ 5, a sua segunda chance não terá desconto e poderá aumentar a nota até 10.

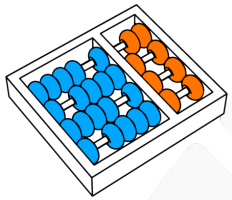
Não deixem os laboratórios acumularem!!!



Avaliação. Testes

Teremos vários testes de entendimento durante o semestre:

- Serão feitos no **Google Sala de Aula**.
- Tem prazo de entrega.
- A correção é automatizada.
- Você pode entregar várias vezes para melhorar a nota.
  - Faça isso! É a ideia!



Avaliação. Maior dúvida

Você deverá responder qual foi a maior dúvida que teve:

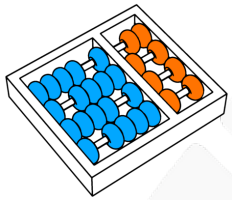
- Após cada aula.
- No Google Sala de Aula.
- Vou responder dúvidas selecionadas na aula seguinte

E se eu não tiver nenhuma dúvida?

- Nenhuma mesmo? Nenhuminha? Nem pequenininha?
- Então, você diz o que mais gostou de aprender na aula.
- Ou pode comentar sobre algo que poderia melhorar na aula.

Não é para pensar muito para fazer a pergunta.

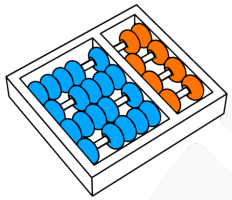
- Nem deixar de perguntar na aula...



Avaliação. Projetos

Projetos são atividades de programação:

- Orientadas e entregues no **Google Sala de Aula**.
- Tem prazo de entrega maior.
- São feitos em duplas (ocasionalmente equipes maiores).



Fraudes

Qualquer tentativa de fraude nos testes, laboratórios ou projetos:

- Implicará em nota final **0,0** para todos os envolvidos.

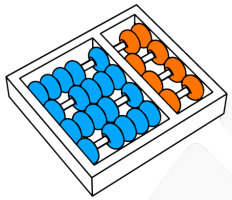
Exemplos de fraudes são:

- Compartilhar trechos de código de qualquer forma.
- Utilizar trechos de códigos da internet ou de outras fontes.
- Copiar, comprar ou vender um laboratório.
- Usar ChatGPT, Copilot, e outras ferramentas do gênero.
- Disponibilizar soluções online antes do término do semestre letivo.

Serei bem inflexível em relação a fraudes:

- Acusados de fraude poderão pedir a formação de uma comissão para avaliar o caso na secretaria.

É melhor não entregar do que ser pego por fraude!



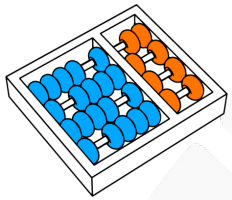
Se arrependendo da fraude. . .

Fraudei e me arrependi, o que eu faço?

- Entre em contato explicando o que ocorreu e os envolvidos.
- Você terá nota zero nas atividades envolvidas na fraude.
- Mas apenas se comunicar antes de eu acusar a fraude.
- Você não fica imune a ser reprovado por outras fraudes.
- Outro participantes da fraude que não se manifestarem serão enquadradas pela regra da nota final zero.

Você sempre pode me perguntar se algo é fraude ou não.

- Inclusive de maneira anônima.



Equipe

Professor: **Santiago** Valdés Ravelo

- E-mail: santiago@ic.unicamp.br ou ravelo@unicamp.br.

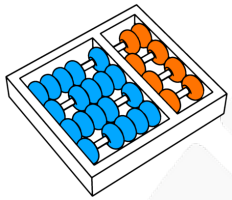
Monitores **PED** (Programa de Estágio Docente).

- São alunos de Mestrado e Doutorado.

Monitores **PAD** (Programa de Apoio Didático).

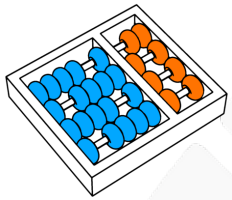
- São alunos de Graduação.

Inclusive PEDs e PADs da turma QR (da EC).



Ferramentas online

- **Google Sala de Aula**.
- **Página da disciplina**.
- **beecrowd** (chave: **mWUlOoi**).
- **Discord**.



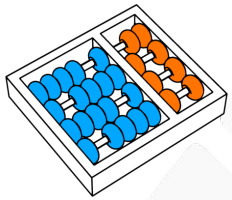
Referências

Não vamos seguir um livro específico, mas

- **How to Think Like a Computer Scientist**: Interactive Edition de Brad Miller e David Ranum
  - Inglês: `https://runestone.academy/runestone/static/thinkcspy/index.html`
  - Português: `https://panda.ime.usp.br/pensepy/static/pensepy/index.html`
- **Dive into Python 3** de Mark Pilgrim: `https://diveintopython3.net`
- Páginas oficiais da linguagem Python:
  - `www.python.org`
  - `www.python.org.br`

Também existe muito material sobre Python na internet

- Mas cuidado, vamos usar **Python 3** não Python 2!



Dúvidas comuns

Preciso saber programar previamente?

- Não!
- Mas se espere que tenham curiosidade de aprender além do que é ensinado em aula.

Preciso trazer um notebook para a aula teórica?

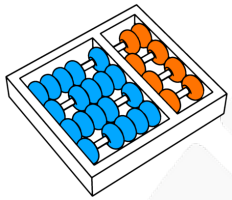
- Não! Mas pode trazer se tiver e quiser.

Posso levar um notebook para a aula no laboratório?

- Sim, mas não é obrigatório.
- A aula é em um laboratório com computadores.

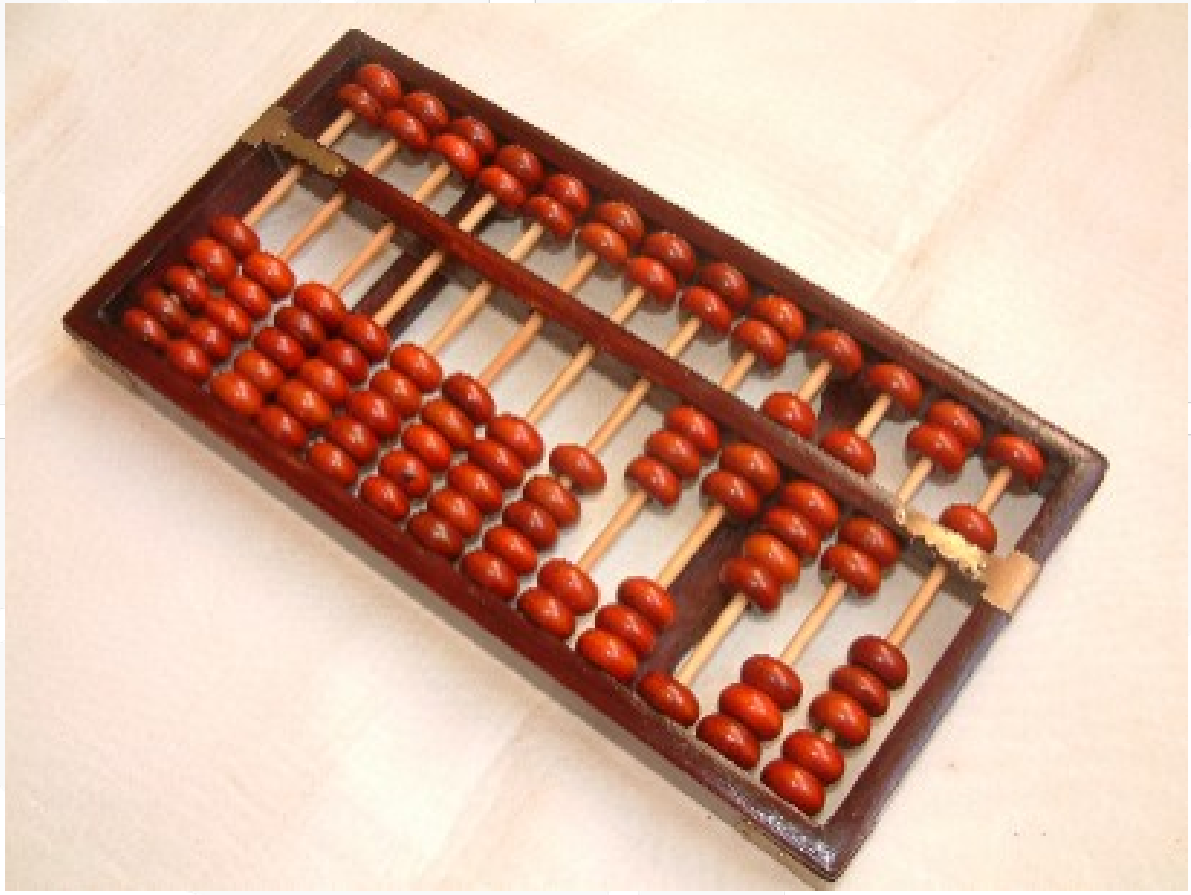
Outras dúvidas?

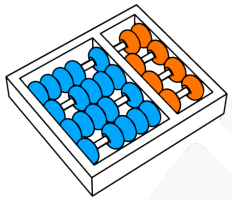
História da Computação



Ábaco (753 AEC, provavelmente bem antes)

O Ábaco é uma das primeiras formas de calcular rapidamente.

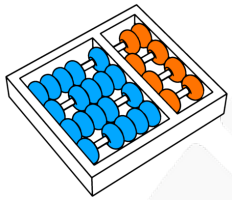




Algoritmo de Euclides (300 AEC)

Euclides publicou um algoritmo para resolver o **máximo divisor comum** entre dois números no seu livro *Elementos*.

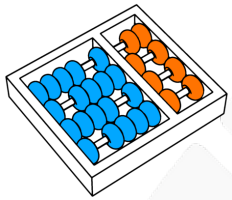
- Mas acredita-se que o algoritmo já era conhecido antes.
- Esse é um dos primeiros algoritmos da história.



Al-Khwarizmi (780 - 850)

Abu Abdullah Muhammad ibn Musa **Al-Khwarizmi** foi um matemático persa.

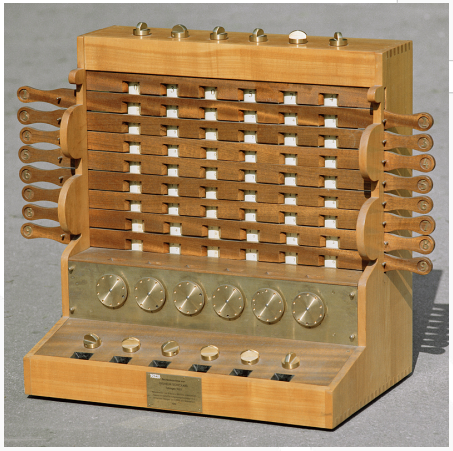
- É considerado o **Pai da Álgebra**.
- Criou soluções sistemáticas para resolver equações lineares e quadráticas.
- Ambos os termos ***algoritmo*** e ***algarismo*** vêm do seu nome.



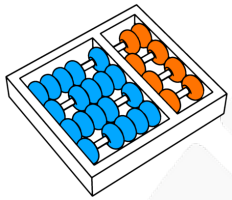
Máquina de Schickard (1623)

Wilhelm Schickard construiu a **1ª máquina de calcular mecânica**.

- Capaz de realizar as operações básicas de **adição e subtração**,
- para números de **seis dígitos**.



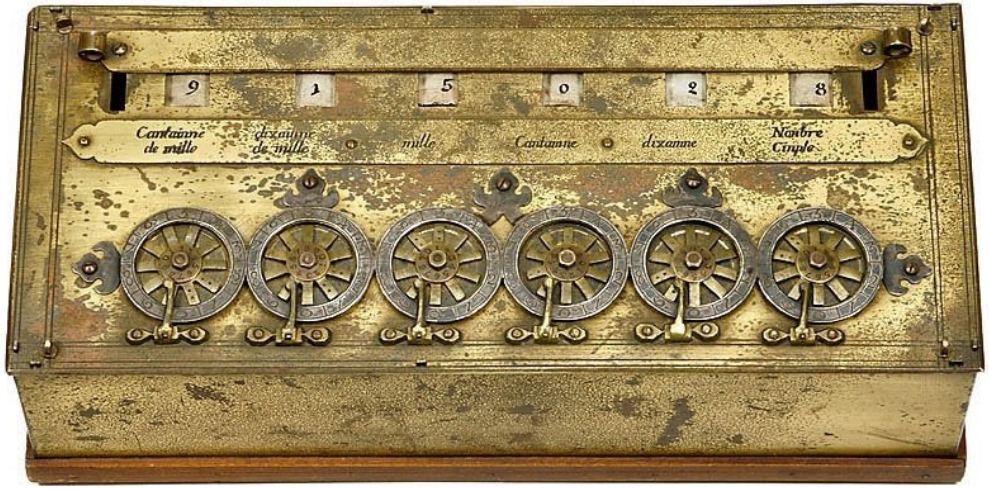
Réplica



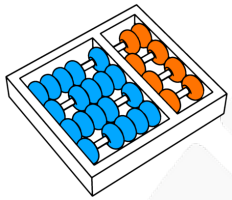
Pascaline (1642)

Blaise Pascal inventou a **calculadora mecânica** chamada ***Pascaline***.

- Realizava operações básicas de **adição** e **subtração**,
- para números de **oito dígitos**.



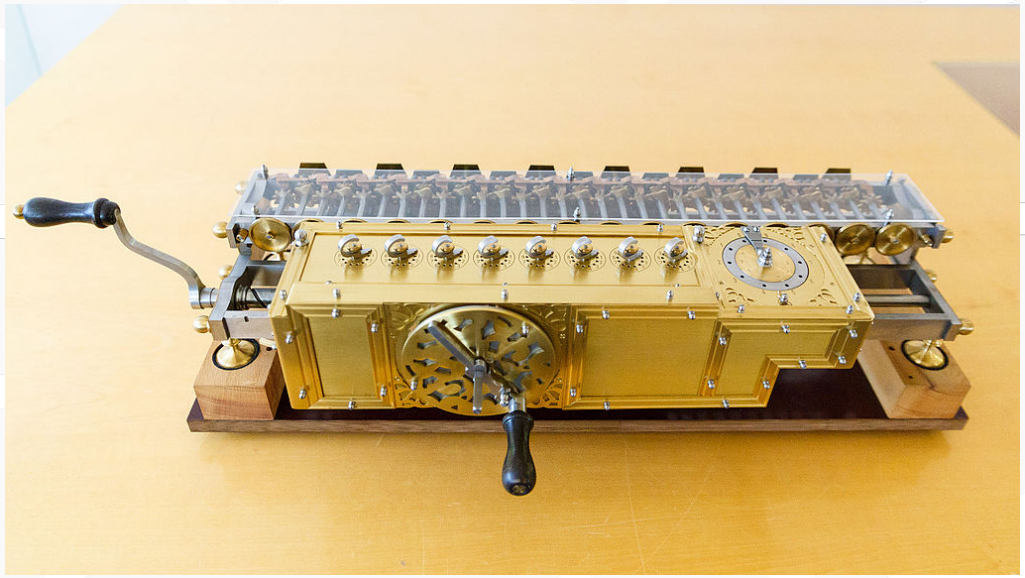
Réplica



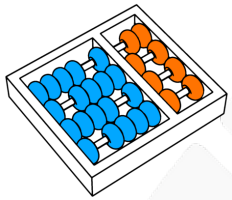
Roda de Leibniz (1673)

Gottfried Leibniz aperfeiçoou a máquina de Pascal.

- Criou a ***Roda de Leibniz***, uma **calculadora mecânica**.
- Realizava **adição**, **subtração**, **multiplicação** e **divisão**.



Réplica

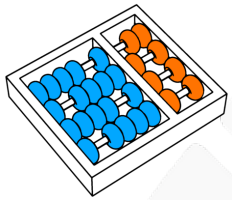


Máquina de Jacquard (1801)

Joseph-Marie Jacquard inventou um tear mecânico controlado por cartões perfurados.

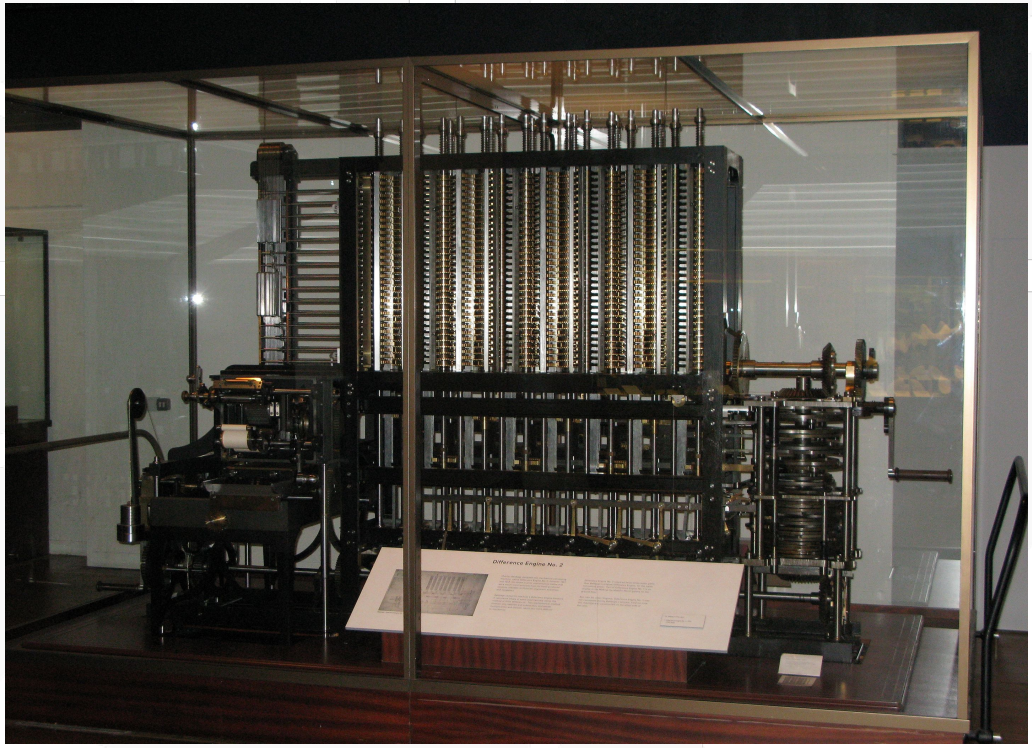
- É a primeira máquina programável da história.
- Os cartões forneciam os comandos necessários para a tecelagem dos padrões nos tecidos.

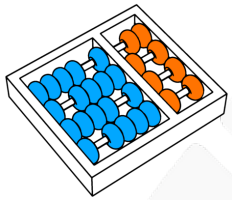
Réplica



Máquina Diferencial (1822)

Charles Babbage projetou a **Máquina Diferencial** para cálculos com polinômios.

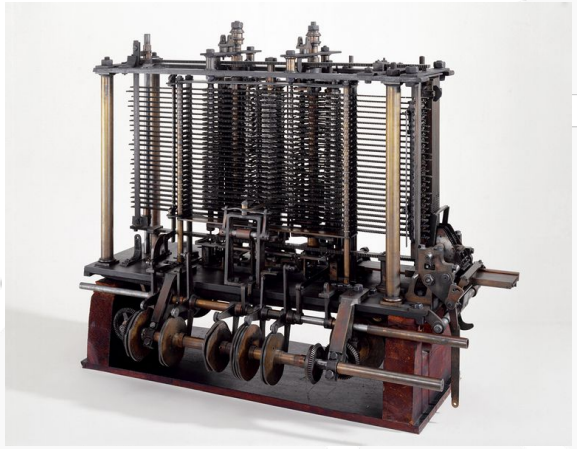




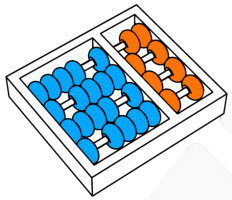
Máquina Analítica (1835)

Charles Babbage projetou a **Máquina Analítica**.

- Um **computador** mecânico **programável** de uso geral.
- Empregava cartões perfurados para a entrada de dados
- e uma máquina a vapor para fornecimento de energia.



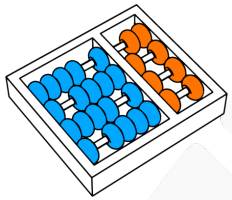
Réplica



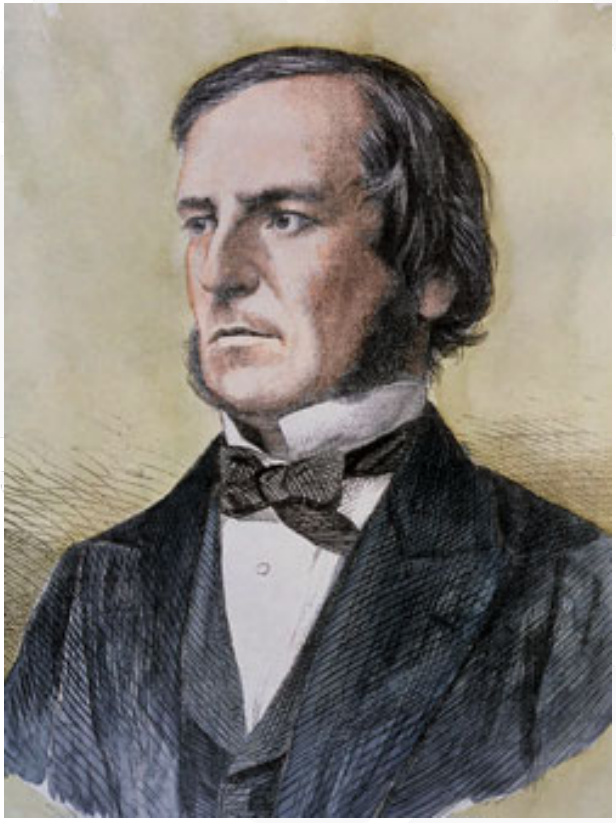
Ada Lovelace (1815 - 1852)

Augusta Ada King, Condessa de Lovelace:

- Publicou o **primeiro algoritmo**,
- para ser executado na máquina analítica de Babbage.
- **É a primeira pessoa a programar na história!**



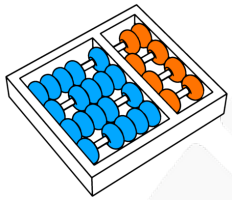
Álgebra Booleana (1847)



George Boole criou a chamada **Álgebra Booleana**, que é a base da **computação lógica**.

A **Álgebra Booleana** é similar a **Álgebra**, mas:

- Ao invés de **números**, temos **Verdadeiro** e **Falso**.
- Ao invés de +, -, *, /, temos **e**, **ou** e **não**.

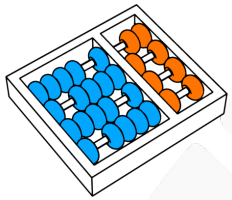


Máquina de Hollerith (1890)

Herman Hollerith, um dos fundadores da **IBM**:

- Construiu uma máquina programável capaz de processar dados
- armazenados em cartões perfurados.
- Foi utilizada para auxiliar o censo de 1890 dos EUA.

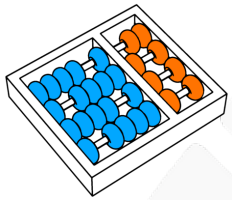
Réplica



Máquina Universal (1936)

Alan Turing criou um modelo teórico de um computador:

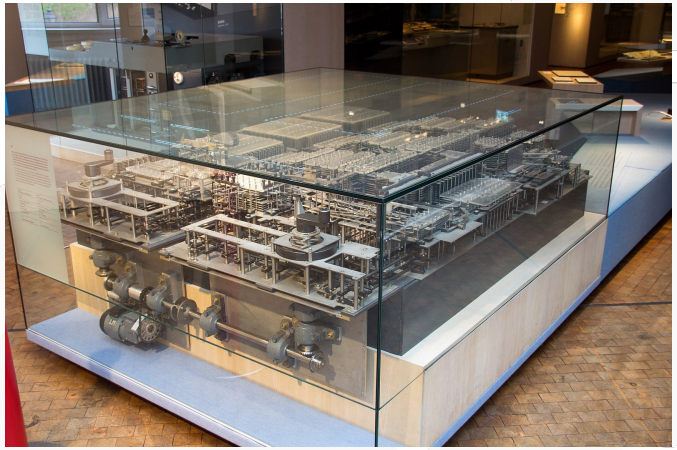
- Chamado de **Máquina Universal**.
- Dele surgiu a ideia da **computabilidade**.
  - Quais problemas podem ser resolvidos por computador?
- Turing é conhecido como o **Pai da Ciência da Computação**.



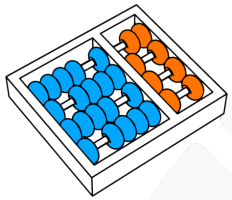
Z1 (1938)

Konrad Zuse construiu o **primeiro computador eletromecânico** completamente funcional, conhecido como Z1.

- A máquina usava relés que executavam os cálculos.
- Dados eram lidos em fitas perfuradas.
- Utilizava o sistema binário de numeração.



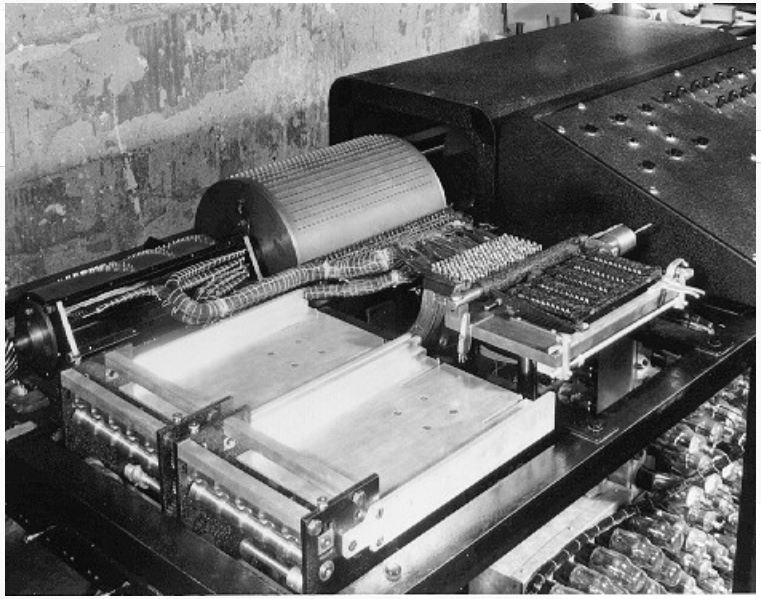
Réplica

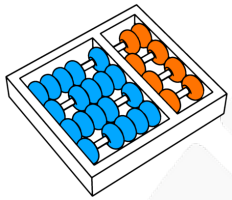


ABC (1942)

John Atanasoff e **Clifford Berry** construíram o **primeiro computador eletrônico digital**.

- Conhecido como **ABC** (Atanasoff-Berry Computer).
- Projetado para resolver sistemas de equações lineares

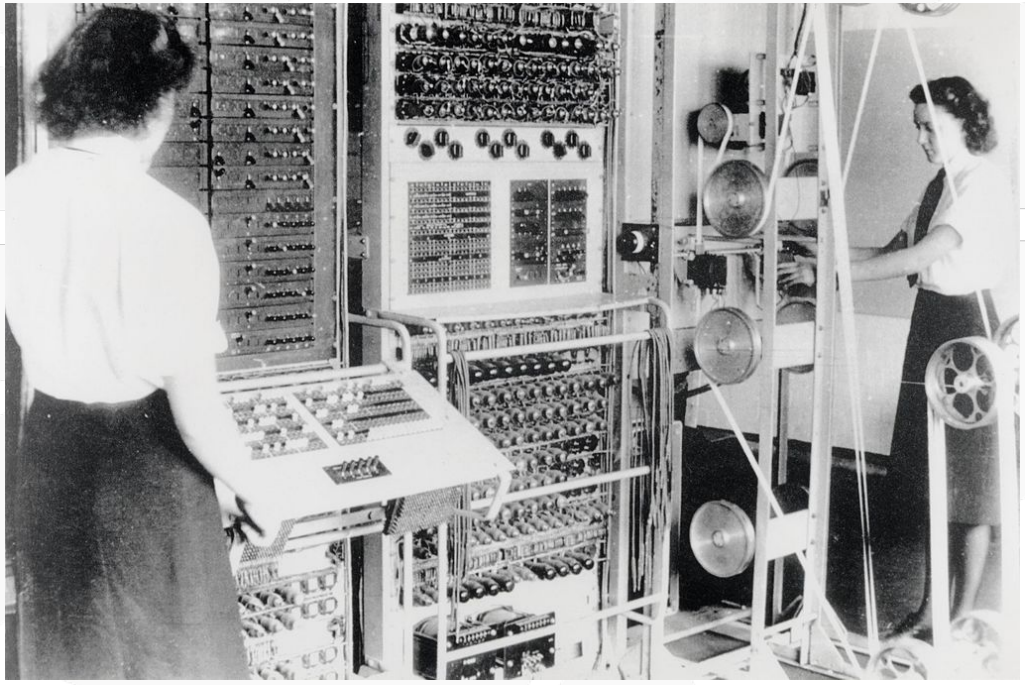


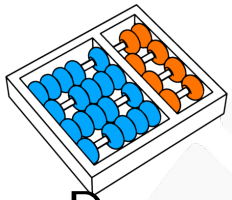


Colossus (1944)

Allan Turing ajudou a construir o computador **Colossus**.

- Para **decifrar códigos** durante a 2ª guerra mundial,
- criados pela máquina alemã **Enigma**.

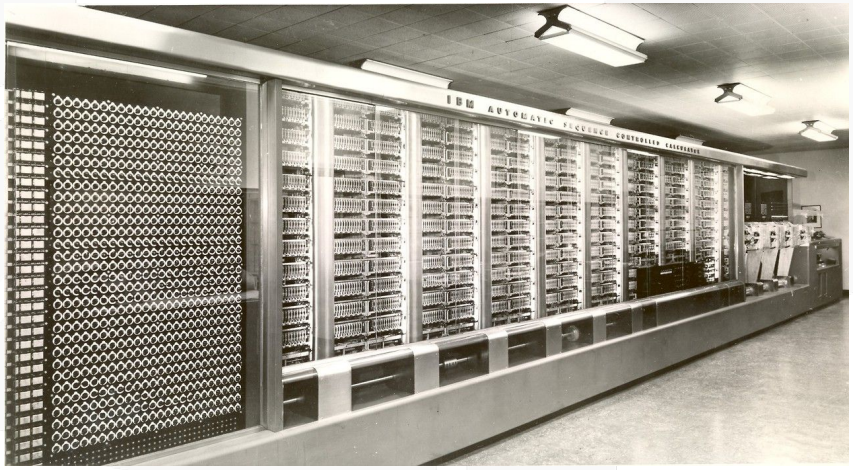


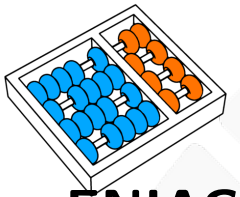


Mark I (1944)

Desenvolvido pela **Marinha** dos Estados Unidos, a Universidade de **Harvard** e a **IBM**, com base na máquina analítica de Babbage.

- Utilizava componentes **elétricos** e **mecânicos**, funcionava com relés e era programado por fita de papel.
- Possuía **10m** de comprimento, **2m** de largura e pesava **70** toneladas.
- Projetado para calcular **trajetórias balísticas** de canhões de longo alcance.

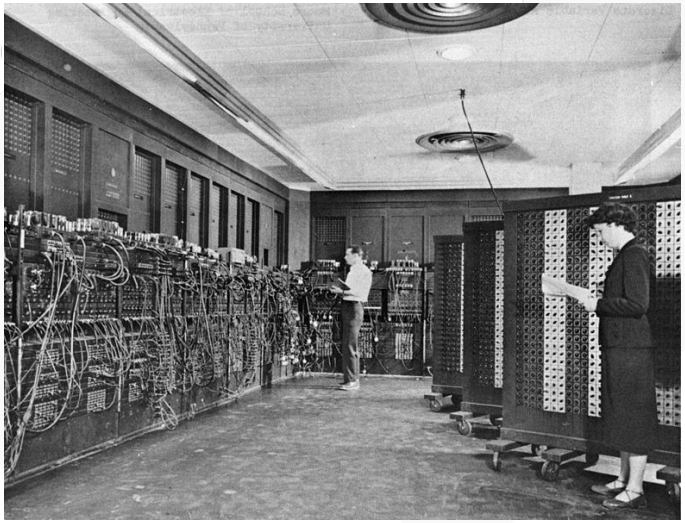


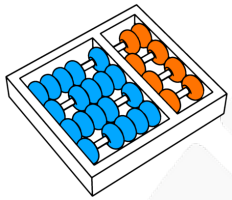


ENIAC (1946)

ENIAC - *Electronic Numeric Integrator And Calculator*:

- Desenvolvido pelo **Exército** dos Estados Unidos.
- **18000** válvulas.
- **30m** de comprimento, **3m** de largura e pesava **30** toneladas.
- Projetado para calcular **trajetórias balísticas** de mísseis.
- Era necessário conectar um grande número de fios, relés e sequências de chaves para definir códigos a serem executados.

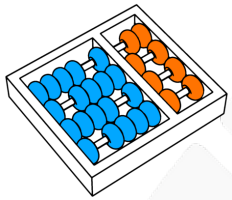




Arquitetura de von Neumann (1946)

John von Neumann propôs que o programa fosse armazenado em um computador da mesma forma que os dados.

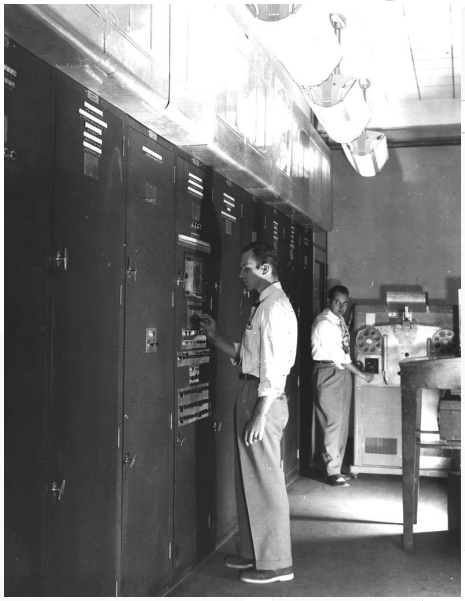
- Esta proposta, chamada de "**Arquitetura de von Neumann**".
- É a **base para os computadores programáveis modernos**
- e é composta por **3 características principais**:
  - Codificação das instruções de modo a serem armazenada na **memória do computador**.
  - **Armazenamento em memória** das instruções e de toda e qualquer informação necessária na execução da tarefa.
  - Busca das instruções, a cada passo do processamento, **diretamente na memória**, e não nos então utilizados cartões perfurados.

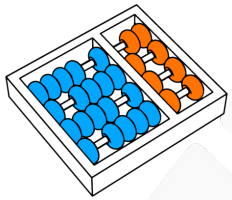


EDVAC (1947)

John von Neuman, **John Eckert** e **John Mauchly** começaram a trabalhar em uma **versão melhorada do ENIAC**.

- Denominado *Electronic Discrete Variable Automatic Computer*.
- Incorporou o conceito de armazenamento de programas em memória.

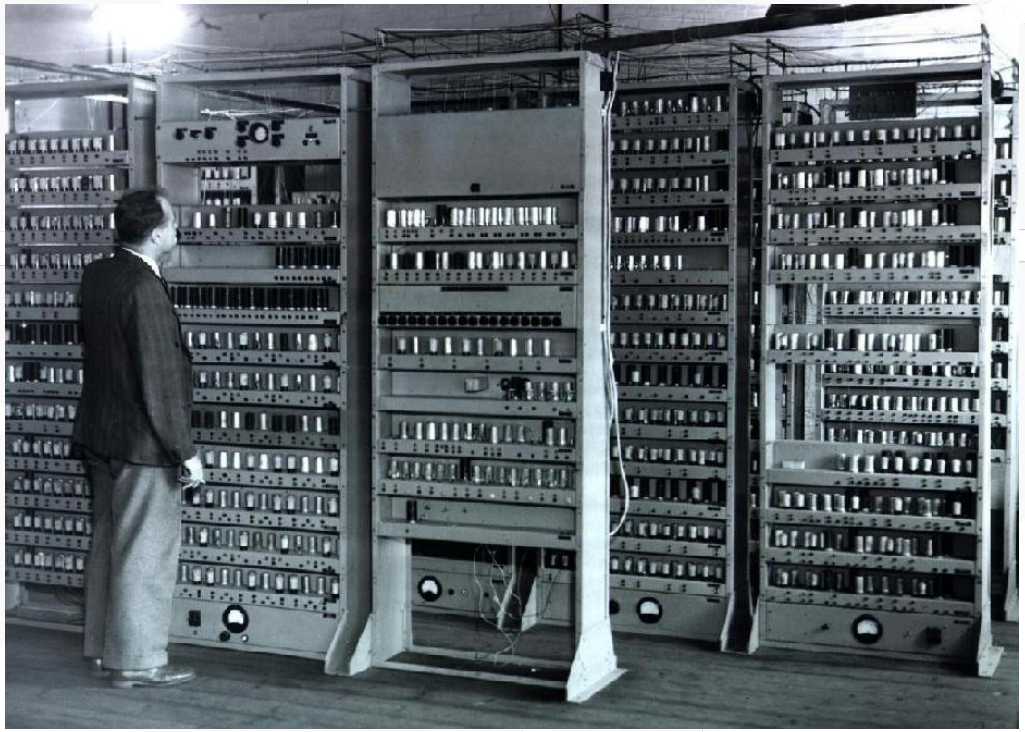


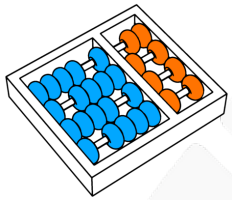


EDSAC (1949)

Maurice Wilkes construiu o EDSAC.

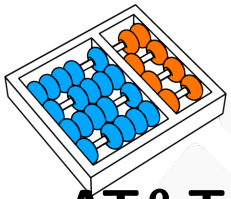
- *Electronic Delay Storage Automatic Calculator*.
- Outro computador que armazenava programas em memória.





COBOL (1953)

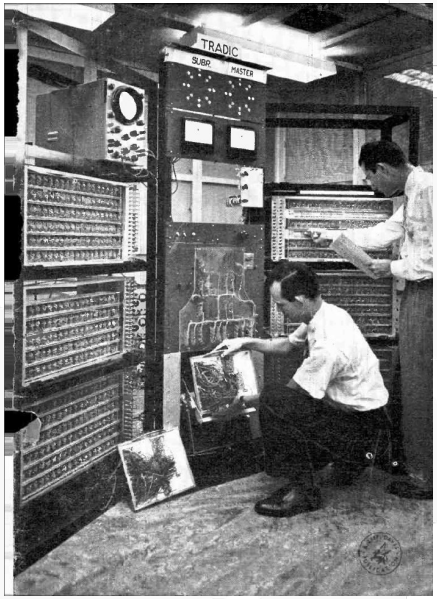
Grace Hopper desenvolve a primeira linguagem de programação.

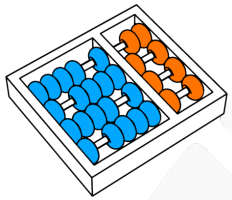


TRADIC (1955)

AT&T Bell construíram o **TRADIC**.

- *Transistorized Airborne Digital Computer*.
- O **primeiro** computador totalmente **transistorizado**.
- Com aproximadamente 800 transistores ao invés de válvulas.
- Permitia trabalhar com menos de 100W de energia.



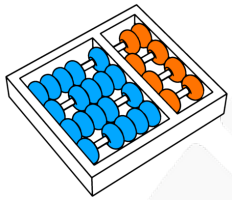


Outros avanços

1958 **Jack Kilby** desenvolveu um dos primeiros **circuitos integrados**, contendo 5 componentes em uma peça de germânio com meia polegada de comprimento.

Esses circuitos são um conjunto de transistores, resistores e capacitores construídos sobre uma base de silício (material semicondutor).

1965 **Hartmanis e Stearns** publicam o artigo "***On the Computational Complexity of Algorithms***" que define os conceitos de **consumo de tempo e memória** dos algoritmos.

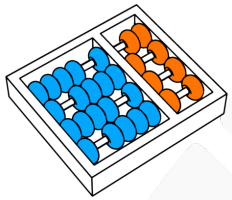


Outros avanços

1969 A agência americana **ARPA** (*Advanced Research and Projects Agency*) desenvolveu a rede **ARPANET**, cujo objetivo era interligar as bases militares e os departamentos de pesquisa do governo americano.

Esta rede iniciou dentro do Pentágono e foi a precursora da **Internet**.

1969 Foi lançado o **Kenbak-1**, considerado o primeiro microcomputador (computador pessoal).

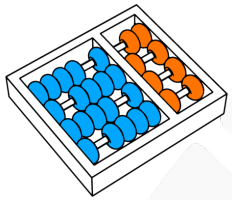


Outros avanços

1971 (1971-1973) **Cook**, **Levin** e **Karp** mostram que vários problemas computacionais são **NP-completos**, um forte indício de que não podem ser resolvidos rapidamente por um computador.

1971 **Ray Tomlinson** implementou um sistema de correio eletrônico (e-mail) na ARPANET.

1972 **Alan Kay** descreveu uma proposta de um dispositivo portátil (chamado "**Dynabook**"), precursor dos atuais *notebooks* ou *laptops*.

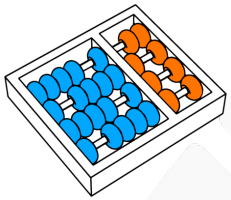


Outros avanços

1973 **Robert Metcalfe** criou o sistema de conectividade **Ethernet** para interligação de computadores em redes locais no centro de pesquisa da Xerox Corporation, em Palo Alto (EUA).

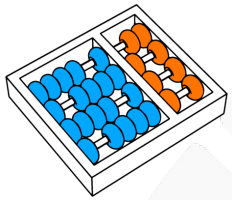
1975 **Bill Gates** e **Paul Allen** fundaram a **Microsoft Corporation**.

1976 **Steve Jobs**, **Steve Wozniak** e **Ronald Wayne** fundaram a **Apple Computer, Inc**.



Apple II (1977)

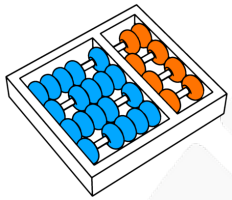
A **Apple** lançou o microcomputador **Apple II**.



IBM PC (1981)

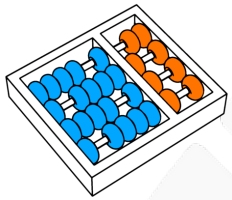
A **IBM** lançou o microcomputador **IBM PC** (*Personal Computer*) **5150**, que se tornou o padrão de computador pessoal.

- O computador possuía processador Intel 8088 de **4,77 MHz**, **64 Kbytes RAM**, uma unidade de disquetes de 5 1/4" (de até **720 Kbytes**), sem disco rígido.
- A **Microsoft** foi contratada para desenvolver o sistema operacional **MS-DOS** (*Microsoft Disk Operating System*).



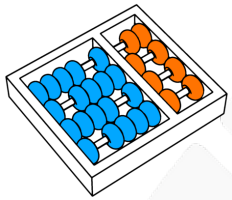
Macintosh (1984)

A **Apple** lançou o computador pessoal Macintosh (**Mac**).



Macintosh Portable (1989)

A **Apple** lançou o ***Macintosh Portable***, o primeiro computador com funcionamento por **bateria**.



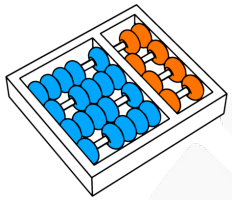
Uma rápida evolução

1993 A **Intel** batizou de ***Pentium*** a sua nova geração de processadores, os quais utilizavam registradores de 32 bits, com **3,1 milhões de transistores**.

1993 A **Apple** lançou o primeiro **PDA** (*Personal Digital Assistant*), o pioneiro dos **computadores de mão**.

1997 O termo telefone inteligente (***smartphone***) foi utilizado pela **Ericsson** para descrever seu aparelho GS 88 Penelope.

1998 **Larry Page** e **Sergey Brin**, dois estudantes de doutorado da Universidade de **Stanford**, criaram a **Google**.



Uma rápida evolução

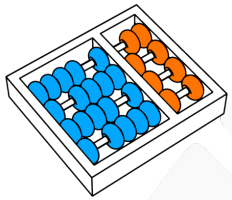
2001 A **Apple** lança o sistema operacional **Mac OS X** e o **iPod**.

2001 Foi lançado o aparelho Kyocera 6035, da Palm, Inc., um dispositivo que combina um PDA com um telefone celular, sendo considerado **um dos primeiros *smartphones*** do mercado.

2003 A Research in Motion Limited (**RIM**) lançou o *smartphone* **BlackBerry**.

2003 A plataforma aberta **Android** foi lançada por **Andy Rubin**, um dos fundadores da empresa Android, Inc., que foi comprada pela **Google** em 2005.

2007 A **Apple** lançou o **iPhone**, um dos primeiros telefones celulares com interface baseada em tela sensível a múltiplos toques.



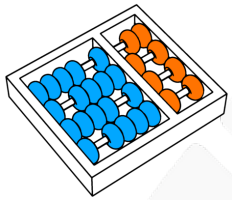
Uma rápida evolução

2010 A **Apple** lançou o **iPad**.

2012 O **Facebook** alcança **1 bilhão** de usuários.

2015 A **Apple** lançou o **Apple Watch**.

2016 A Universidade de **Maryland** construiu o primeiro **computador quântico reprogramável**.



Futuro

Existem muitos desafios e desenvolvimentos para o futuro:

- Depois de 50 anos, a mais importante pergunta da **computação** ainda não foi respondida (**P = NP**?)
  - **Podemos resolver diversos problemas computacionais importantes rapidamente ou não?**
- **Computação Quântica** ainda não é uma realidade.
- Estamos muito longe de uma **Inteligência Artificial Geral**.
- Como aproveitar a gigantesca quantidade de **dados** gerados?
- Como lidar com **segurança** e **privacidade** dos usuários?
- Como lidar com **questões de diversidade** no projeto de algoritmos?
- Entre muitas outras perguntas das várias áreas da computação!

Apresentação da disciplina

MC102 - Algoritmos e Programação de Computadores

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

0

UNICAMP