Aula 02 - Introdução à Lógica de Programação

A partir desse ponto, iremos todos juntos começar a fazer programa para ganhar dinheiro. A pergunta é:

Você sabe o que é um programa? Sabe mesmo?

Vamos testar seus conhecimentos sobre programas à prova com uma breve atividade. Logo abaixo se encontra uma lista de opções. Sua tarefa é marcar todas as opções que você considera ser um programa, e deixar desmarcado as opções que não correspondem à programas. Sinta-se à vontade, e leve o tempo que precisar:

- [ ] Windows
- [ ] Word
- [ ] Chrome
- [ ] Firefox
- [ ] Soneca da tarde
- [ ] Lanche da noite
- [ ] Preparar um delicioso bolo para as visitas
- [ ] Pelada com os amigos no Domingo
- [ ] Cineminha do fim de semana
- [ ] Os Embalos de Sábado à Noite
- [ ] Churrasco com a família
- [ ] Clube com a galera
- [ ] Viagem para um resort em Orlando
- [ ] Jornal Nacional
- [ ] Domingo Legal
- [ ] Fantástico
- [ ] Novela das 9
- [ ] Programa Eleitoral
- [ ] Bolsa Família
- [ ] Sair com a paquera

Vamos ao gabarito

Se você deixou de marcar qualquer uma dessas opções, você errou, pois todas essas opções são **programas**. Sim, é isso mesmo, e todas no seu sentido mais literal da palavra. **-"Ah, mas tem programa aí que não é programa de computador..."**, você diz. E eu digo: **-"Estava me referindo à programas. Não necessariamente tinha que ser de computador."** Para aprender a programar, antes é necessário aprender o que é um programa, em seu sentido mais básico da palavra.

Conceito de programa

Um programa consiste em basicamente num conjunto de instruções que visam atingir um determinado objetivo final, ou resolver um determinado problema. Cada uma dessas instruções são basicamente uma sequência de comandos que devem ser executados em uma determinada ordem. Essas instruções são chamadas de **Algoritmos**. Em outras palavras, um programa é basicamente um **roteiro**, ou ***script*** a ser seguido, e executar um programa é basicamente ler e seguir esse roteiro à risca.

É exatamente isso que você faz todos os dias quando vai trabalhar, ou quando vai estudar, ou quando vai passear, ou quando vai comer algo, quando vai viajar, ou até mesmo quando vai dormir: você segue um passo a passo. O mesmo acontece quando seus órgãos funcionam: seguem uma programação definida pela natureza para que seu corpo funcione corretamente, e você possa viver perfeitamente. Quando um desses orgãos não segue sua programação original, você fica doente.

No caso do PC, quando o usuário executa um programa de computador, o que ele faz é basicamente pedir para que o computador leia e siga um "manual de instruções" para que possa atender suas necessidades. Esse suposto manual de instruções foi criado e repassado para o computador por um profissional chamado de **Programador**, também conhecido como **Desenvolvedor**, ou ainda **Developer**, ou **Dev**. Em outras palavras: o que o desenvolvedor faz é criar uma espécie de **"Receita de bolo"** para que o computador possa seguir.

-"Muito legal. Agora me diz por que ele não executa a instrução abaixo?"

In [ ]:
```
'''
Bom dia computador.
Por favor, eu gostaria que o senhor criasse para mim um prrograma de computador.
Que tenha telas bonitas e interativas.
Que eu possa clicar nos botões e digitar textos.
Salve tudo e me avise quando terminar.
Bjs
'''
```

O motivo disso é bem simples de entender, e a explicação se encontra em um curso que serve como pré-requisito para este, o Curso de Operador de Micro. A explicação é que, conforme visto em Informática básica, o computador não é capaz de entender o idioma do ser humano. Na verdade, a linguagem que o computador entende é o **binário**: um código numérico composto por apenas 2 números, o 0 e o 1. Sempre que o usuário digita um texto, ou envia uma imagem para o computador, tudo o que ele recebe é uma sequência de números binários que representa a informação que o usuário enviou. Ele processa essa informação exatamente em vários 0 e 1, e ao terminar, envia uma resposta para o usuário também em sequências de 0 e 1, mas que no final o usuário irá ver como a resposta desejada, seja ela um texto, um áudio, uma imagem ou um vídeo. O computador sabe que aquela era a resposta desejada, mas não sabe o que aquilo quer

dizer, já que ele não fala o idioma dos humanos, mas simula entender. Uma boa analogia a isso é a **Teoria do Quarto Chinês**.

O Quarto Chinês

O Quarto Chinês foi um argumento proposto pelo filósofo norte-americano **John Searle** em 1980, que pretendia com essa teoria contrapor o Teste de Turing, que objetiva verificar se uma máquina pode ou não ser considerada inteligente. O argumento de Searle é baseado no fato de que os computadores podem entender somente a sintaxe mas não a semântica. Em outras palavras, ele não teria uma inteligência de fato, apenas simularia ter uma.

O teste

Imagine-se preso em um quarto sem janelas, com apenas uma porta opaca com uma única abertura para o mundo exterior por onde é possível passar apenas um pedaço de folha, como uma carta, por exemplo. Do outro lado da porta, fora da sala, há uma pessoa de origem chinesa, que não entende o nosso idioma, e nem nós entendemos o dela. Essa pessoa deseja se comunicar com nós, sem fazer ideia do fato de que não sabemos falar nem escrever no idioma dela, e para isso, ela repassa cartas pela abertura da porta, e obviamente elas estão escritas no idioma chinês, e por esse motivo, você não faz a menor ideia do que está escrito na carta.

Entretanto, dentro do quarto existe um livro, que funciona como um manual de instruções. Dentro desse manual, há vários símbolos em chinês idênticos aos que são encontrados nas cartas. Porém, o livro não ensina o que está escrito nos caracteres chineses. Ele apenas orienta, em nosso idioma, como proceder para entregar uma resposta para a pessoa do outro lado da porta.

Caso siga corretamente as instruções, você conseguirá (até certo ponto) se comunicar com a pessoa do outro lado, já que ela entenderá a mensagem repassada por você

perfeitamente, e acreditará piamente que a pessoa de dentro do quarto se trata também de outro chinês, e não de uma pessoa que só está seguindo ordens.

Segue abaixo um vídeo do youtube explicando rapidamente como funciona a teoria:

Link do vídeo: https://youtu.be/1_h76lh1wzE?si=3dzrqrC5MhLj5P7n

Da mesma forma como acontece no Quarto Chinês, o computador precisa de um intermediário para transcrever os dados que o usuário repassa para o computador para a linguagem de máquina, e que também possa fazer o sentido inverso da informação, transcrevendo a linguagem de máquina para a linguagem humana. Quem faz isso são as **Linguagens de Programação**.

Linguagens de Programação

Funcionam como idiomas que podem ser lidos tanto por seres humanos quanto por computadores, e atuam como intermédios entre a máquina e o homem. Atualmente, há mais de 2 mil linguagens de programação no mundo, sendo a principal delas o Python.

As linguagens mais próximas do que uma máquina pode entender, chamammos de **Linguagens de Baixo Nível**, enquanto que as linguagens mais próximas ao que o ser humano pode entender, chamamos de **Linguagens de Alto Nível**. O Python, por exemplo, é uma linguagem de alto nível. Já o binário é a linguagem mais baixo nível que existe.

História das Linguagens de Programação

- **1843: Máquina Analítica.** Ada Lovelace inventa o primeiro algoritmo de máquina para a máquina de diferenças de Charles Babbage, que estabelece as bases para todas as linguagens de programação.
- **1944-45: Plankalkül.** Em algum lugar entre 1944-45, Konrad Zuse desenvolveu a primeira linguagem de programação 'real' chamada Plankalkül (Plan Calculus). A linguagem do Zeus (entre outras coisas) permitia a criação de procedimentos, que armazenavam pedaços de código que podiam ser apresentados repetidamente para executar operações de rotina.
- **1949: Linguagem Assembly.** A linguagem Assembly foi usada na calculadora automática de armazenamento de atraso eletrônico (EDSAC). A linguagem Assembly era um tipo de linguagem de programação de baixo nível que simplificava a linguagem do código de máquina. Em outras palavras, as instruções específicas necessárias para operar um computador.
- **1949: Shortcode.** Shortcode (ou código de ordem curta), foi a primeira linguagem de alto nível (HLL) sugerida por John McCauley em 1949. No entanto, foi William Schmitt quem a implementou para o computador BINAC no mesmo ano e para o UNIVAC em 1950.

- **1952: Autocode.** Autocode era um termo geral usado para uma família de linguagens de programação. Desenvolvido pela primeira vez por Alick Glennie para o computador Mark 1 na Universidade de Manchester, o Autocode foi a primeira linguagem compilada a ser implementada, o que significa que pode ser traduzida diretamente em código de máquina usando um programa chamado compilador. O Autocode foi usado nas primeiras máquinas de computação Ferranti Pegasus e Sirius, além do Mark 1.

- **1957: FORTRAN.** FORmula TRANslation ou FORTRAN foi criada por John Backus é considerada a linguagem de programação mais antiga em uso atualmente. A linguagem de programação foi criada para cálculos científicos, matemáticos e estatísticos de alto nível. O FORTRAN ainda está em uso hoje em alguns dos supercomputadores mais avançados do mundo.

- **1958: ALGOL (Linguagem algorítmica).** A linguagem algorítmica ou ALGOL foi criada por um comitê conjunto de cientistas da computação americanos e europeus. ALGOL serviu como ponto de partida para o desenvolvimento de algumas das linguagens de programação mais importantes, incluindo Pascal, C, C++ e Java.

- **1958: LISP (processador de lista).** O processador de lista ou LISP foi inventado por John McCarthy no Instituto de Tecnologia de Massachusetts (MIT). Originalmente projetado para inteligência artificial, o LISP é uma das linguagens de programação mais antigas ainda em uso hoje e pode ser usado no lugar de Ruby ou Python. Empresas como Acceleration, Boeing e Genworks ainda estão usando LISP em suas pilhas de tecnologia.

- **1959: COBOL (Common Business Oriented Language).** Common Business Oriented Language (COBOL), é a linguagem de programação por trás de muitos processadores de cartão de crédito, caixas eletrônicos, chamadas telefônicas e celulares, sinais hospitalares e sistemas de sinais de trânsito (só para citar alguns). O desenvolvimento da linguagem foi liderado pela Dra. Grace Murray Hopper e foi projetada para rodar em todas as marcas e tipos de computadores. O COBOL ainda é usado até hoje principalmente para sistemas bancários e de gamificação .

- **1964: BASIC (Código de instrução simbólica para todos os fins do iniciante).** Beginners All-Purpose Symbolic Instruction Code ou BASIC foi desenvolvido por um grupo de estudantes do Dartmouth College. A linguagem foi escrita para alunos que não tinham um forte conhecimento de matemática ou computadores. A linguagem foi desenvolvida pelos fundadores da Microsoft, Bill Gates e Paul Allen, e se tornou o primeiro produto comercializável da empresa.

- **1970: PASCAL.** Nomeado após o matemático francês Blaise Pascal, Niklaus Wirth desenvolveu a linguagem de programação em sua homenagem. Foi desenvolvido como uma ferramenta de aprendizado para programação de computadores, o que significava que era fácil de aprender. Foi o favorito da Apple nos primeiros dias da empresa, devido à sua facilidade de uso e potência.

- **1972: Smalltalk.** Desenvolvido no Xerox Palo Alto Research Center por Alan Kay, Adele Goldberg e Dan Ingalls, o Smalltalk permitiu que os programadores de computador modificassem o código em tempo real. Ele introduziu uma variedade de aspectos da linguagem de programação que são linguagens visíveis de hoje, como Python, Java e Ruby. Empresas como Leafly, Logitech e CrowdStrike afirmam que usam Smalltalk em suas pilhas de tecnologia.

- **1972: C.** Desenvolvido por Dennis Ritchie no Bell Telephone Laboratories para uso com o sistema operacional Unix. Foi chamado de C porque foi baseado em uma linguagem anterior chamada 'B'. Muitas das linguagens líderes atuais são derivadas de C, incluindo; C#, Java, JavaScript, Perl, PHP e Python. Ele também ainda está sendo usado por grandes empresas como Google, Facebook e Apple.

- **1972: SQL (SEQUEL na época).** O SQL foi desenvolvido pela primeira vez pelos pesquisadores da IBM Raymond Boyce e Donald Chamberlain. SEQUEL (como era chamado na época), é usado para visualizar e alterar informações que estão armazenadas em bancos de dados. Hoje em dia a linguagem é um acrônimo – SQL, que significa Linguagem de Consulta Estruturada. Há uma infinidade de empresas que usam SQL e algumas delas incluem Microsoft e Accenture.

- **1980/81: Ada.** Ada foi originalmente projetada por uma equipe liderada por Jean Ichbiah da CUU Honeywell Bull sob contrato com o Departamento de Defesa dos Estados Unidos. Nomeada em homenagem à matemática de meados do século XIX Ada Lovelace, Ada é uma linguagem de programação de alto nível estruturada, estaticamente tipada, imperativa, de amplo espectro e orientada a objetos. Ada foi criada a partir de outras linguagens de programação populares na época, como Pascal. Ada é usado para sistemas de gerenciamento de tráfego aéreo em países como Austrália, Bélgica e Alemanha, bem como em uma série de outros projetos de transporte e espaço.

- **1983: C++.** Bjarne Stroustrup modificou a linguagem C no Bell Labs, C + + é uma extensão de C com aprimoramentos como classes, funções virtuais e modelos. Ele está listado entre as 10 principais linguagens de programação desde 1986 e recebeu o status de Hall da Fama em 2003. C + + é usado no MS Office, Adobe Photoshop, mecanismos de jogos e outros softwares de alto desempenho.

- **1983: Objetive-C.** Desenvolvido por Brad Cox e Tom Love, Objective-C é a principal linguagem de programação usada para escrever software para macOS e iOS, os sistemas operacionais da Apple.

- **1987: Perl.** Perl foi criado por Larry Wall e é uma linguagem de programação de alto nível e de uso geral. Ele foi originalmente projetado como uma linguagem de script projetada para edição de texto, mas hoje em dia é amplamente usado para muitos propósitos, como CGI, aplicativos de banco de dados, administração de sistemas, programação de rede e programação gráfica.

- **1990: Haskell.** Haskell é uma linguagem de programação de uso geral nomeada em homenagem ao lógico e matemático americano Haskell Brooks Curry. É uma

linguagem de programação puramente funcional, o que significa que é principalmente matemática. Ele é usado em vários setores, especialmente aqueles que lidam com cálculos complicados, registros e processamento de números. Como muitas outras linguagens de programação desta época, não é muito comum ver Haskell em uso para aplicativos conhecidos. Com isso dito, a linguagem de programação foi usada para escrever vários jogos, um dos quais é Nikki and the Robots .

- **1991: Python.** Com o nome da banda de comédia britânica 'Monty Python', o Python foi desenvolvido por Guido Van Rossum. É uma linguagem de programação de alto nível de uso geral criada para oferecer suporte a uma variedade de estilos de programação e ser divertida de usar. Python é, até hoje, uma das linguagens de programação mais populares do mundo, usada por empresas como Google, Yahoo e Spotify.

Há muitas outras linguagens de programação populares que foram criadas após esse período, como o Java, o C#, o PHP e o JavaScrpt, mas como o objetivo do curso é o Python, encerraremos a história por aqui.

Fonte

- https://www.atrainformatica.com.br/2023/04/05/historia-das-linguagens-de-programacao/

Tipos de Algoritmos

Há basicamente 3 formas de expressar Algoritmos fora das linguagens de programação:

- **Linguagem Natural ou Descrição Narrativa:** é a forma de se expressar naturalmente uma solução, utilizando o nosso próprio idioma ou a nossa fala para criar ou executar o algoritmo. Um excelente exemplo de um algoritmo em Descrição Narrativa é a famosa **Receita de Bolo**. Ela contém exatamente tudo o que um programa precisa ter. Sempre que escrever uma instrução passo a passo, ela será considerada um algoritmo em linguagem natural ou descrição narrativa.

- **Fluxograma:** é a forma visual de se expressar uma solução, utilizando diagramas descritivos. Normalmente são utilizados para descrever processos.

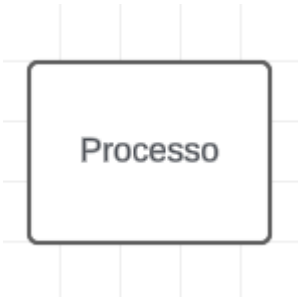
- **Pseudo-código:** é a forma mais aproximada de um código-fonte de uma linguagem de programação. A diferença é que a pseudo-linguagem simula um ambiente de programação, mas a linguagem utiliza de comandos em português para a execução dos algoritmos no computador. Exemplos de pseudo-código são o **VisualG** e o **Portugol Studio**. **Obs:** o Python é uma linguagem de programação tão fácil de aprender que no caso dessa linguagem específica não é vantagem ensinar lógica de programação utilizando psuedo-código, já que a própria linguagem já é simples o suficiente para ser utilizada diretamente no ensino de algoritmos, e portanto, não utilizaremos pseudo-código nesse curso.

Fluxogramas

Como o objetivo do curso é focar em Python, não utilizaremos tanto no decorrer do curso os fluxogramas, até porque existem outras formas melhores de se descrever um programa utilizando diagramas, como a UML, por exemmplo. Entretanto, é interessante sabermos um pouco sobre fluxogramas nesse início de curso, e portanto, segue abaixo uma descrição dos principais elementos de um fluxograma que você precisa saber:

Processo:

Também conhecido como "Símbolo de ação", esta forma representa um processo, ação ou função. É o símbolo mais amplamente usado em fluxogramas.



Exterminador:

Também conhecido como "Símbolo de terminação", este símbolo representa os pontos iniciais, finais e resultados potenciais de um caminho. Muitas vezes contém "Início" ou "Fim" dentro da forma.

Decisão:

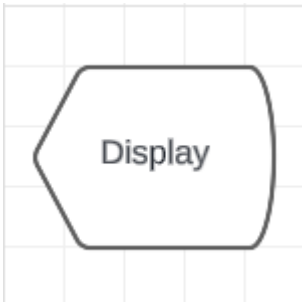
Indica uma questão a ser respondida, geralmente com sim/não ou verdadeiro/falso. O caminho do fluxograma pode se dividir em diferentes ramificações dependendo da resposta ou das consequências em seguida.

Entrada manual:

Representa a entrada manual de dados em um campo ou passo de um processo, geralmente por meio de um teclado ou dispositivo. Um exemplo de cenário inclui o passo em um processo de login em que o usuário é solicitado a inserir dados manualmente.

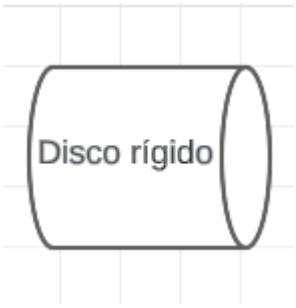
Exibição:

Esta forma é útil para indicar onde informações serão exibidas dentro de um fluxo de processo.



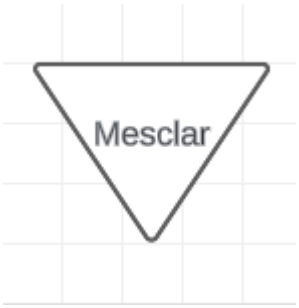
Disco rígido:

Indica onde dados são armazenados dentro de um disco rígido, também conhecido como armazenamento de acesso direto.



Mesclagem:

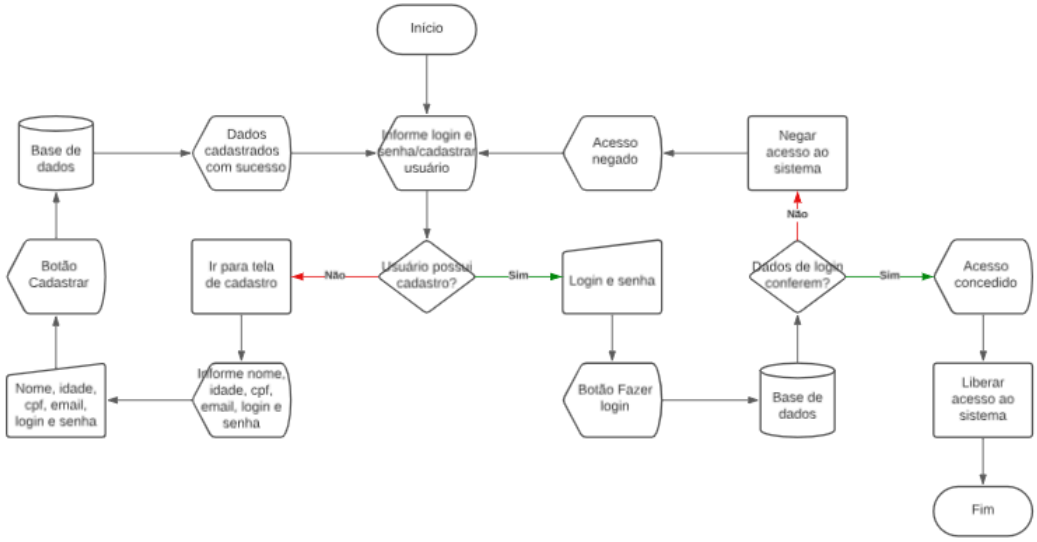
Combina vários caminhos para virar um só.



Banco de dados:

Representa os dados hospedados em um serviço de armazenamento que provavelmente permitirá a pesquisa e filtragem por parte de usuários.

Exemplo de fluxograma:



Fonte:

- https://www.lucidchart.com/pages/pt/fluxograma-simbologia?authuser=1

Exercícios

---

1. Qual o nome que se dá ao passo a passo para a solução de um problema?

- ◯ Cálculo
- ◯ Solução
- ◯ Algoritmo
- ◯ Receita de Bolo

2. Qual o nome do tipo de Algoritmo em que se utilizam figuras para se explicar a solução de um problema?

- ◯ Python
- ◯ Programa
- ◯ Linguagem Natural
- ◯ Fluxograma

3. Qual o nome do(a) primeiro(a) programador(a) da história?

- ◯ Condessa de Ada Lovelace
- ◯ Bill Gates
- ◯ Steve Jobs
- ◯ Guido Van Rossum

4. Do que se trata o Quarto Chinês?

- ◯ Um quarto onde se encontram chineses dispostos a fazerem programa por um preço
- ◯ Uma teoria criada para refutar a ideia de que o computador tem inteligência
- ◯ O nome de uma sala onde são desenvolvidos sistemas de Inteligência Artificial
- ◯ Uma sala onde não entendo nada do que dizem, parece que estão falando chinês

5. Para que serve o símbolo **Exterminador** do fluxograma?

- ◯ Para marcar o início e o fim de um programa no fluxograma
- ◯ Para exterminar a raça humana e dar a Skynet a vitória da Guerra das Máquinas
- ◯ Para vir do futuro para matar John Connor e a mãe dele, Sarah Connor
- ◯ Para marcar o fim de um processo no fluxograma