1 - Lógica de Programação

**Algoritmo**: Sequência de passos que resolve um problema.

**Pseudocódigo**: é uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples (em português)

**Fluxograma**: representa graficamente o algoritmo.

**Variáveis**: um objeto (localizada na memória).

**Constante**: valores imutáveis

**Expressões** literais: pode receber valores variáveis ou constantes.

**Linguagem de baixo nível**

Se aproxima mais da linguagem de máquina (precisa ter o conhecimento direto da arquitetura do computador)

(Assembly)

**Linguagem de alto nível**

Sintaxe se aproxima mais da nossa linguagem e se distanciam da linguagem de máquina.

(C, C++, PHP, Javascritp, Pyhton)

**Compiladas**

Executada diretamente pelo sistema operacional ou processador

(C++, C#, )

**Interpretadas**

Código fonte é executava por um programa.

(JavaScript, PHP, Python)

2 – Estruturas de dados

# Conceitos iniciais sobre estrutura de dados, arrays e registro

## Estrutura de dados:

Estrutura organizada de dados na memória de um computador ou em qualquer dispositivo de armazenamento.

Devemos saber como realizar um determinado conjunto de operações:

* Inserir dados;
* Excluir dados;
* Localizar um elemento;
* Percorrer todos os itens constituintes da estrutura para visualização;
* Classificar.

Principais estruturas de dados:

* Vatores e matrizes;
* Registro;
* Lista;
* Pilha;
* Fila;
* Arvore;
* Tabela hash;
* Grafos.

## Vetores e matrizes

São estruturas de dados simples que podem auxiliar quando há muitas variáveis do mesmo tipo.

Permite armazenar vários dados de um único tipo de dados.

**Vetor ou Array**: é uma estrutura de dados indexada, que pode armazenar uma determinada quantidade de valores do mesmo tipo. (ex. vetor [0], vetor [1])

**Matriz ou Array multidimensional**: É um vetor que possui duas ou mais dimensões. (ex. matriz [0] [0] = 8

|  |  |  |
| --- | --- | --- |
| 8 | 30 | 2 |
| 25 | 17 | 18 |
| 5 | 12 | 21 |

## Registro

Permite armazenar mais de um tipo de dado. (ex. em Python: dicionários)

Toda estrutura de registro tem um nome.

# Listas, Pilhas e Filas

## Listas

Armazena dados de um determinado tipo em uma ordem específica.

Listas possuem tamanho ajustável, diferente de arrays, que possuem tamanho fixo.

**Lista ligada**: cada um dos nós conhece o valor que está sendo armazenado em seu interior além de conhecer o elemento posterior a ele. Os nós são amarrados com a indicação de qual é o próximo elemento.

**Lista duplamente ligada**: Elas são bidirecionais. Os nós sabem quem é o próximo elemento e também quem é o elemento anterior, o que permite a navegação reversa.

## Pilha

Estrutura de dados que serve como uma coleção de elementos. Permite o acesso a somente um item de dados armazenados.

O acesso aos itens da pilha é restrito, somente um item pode ser removido ou lido por vez.

**Pilha LIFO/UEPS**: Última que Entra Primeira que Sai. O último elemento a ser retirado é o último que tiver sido inserido.

**Pilha FIFO/PEPS**: Primeiro que Entra Primeiro que sai. O primeiro elemento a ser retirado é o primeiro que foi inserido.

## Filas

O elemento mais antigo é que será o primeiro a ser removido.

# Estrutura de dados do tipo árvores, tabela hash e grafos

## Árvore

Organiza os elementos de forma hierárquica, onde existem um elemento que fica no topo da árvore (raiz) e os elementos subordinados a ele(nós/folhas)

## Tabelas hash

É uma estrutura de dados especiais, que associa chaves de pesquisa a valores.

Generalização da ideia de array utilizando uma função denominada hashing para espalhar os elementos, fazendo que fiquem de forma não ordenada dentro do ‘array’ que define a tabela.

A tabela hash permite a associação de “valores” a “chaves”.

**Valores**: é a posição ou índice onde o elemento se encontra.

**Chave**: parte da informação que compõe o elemento a ser manipulado.

## Grafos

Estruturas que permitem programar a relação entre objetos.

Objetos são vértices ou “nós” do grafo.

Os relacionamentos são arestas.

Introdução ao Git e ao GitHub

# Navegação via command line interface e instalação

Alguns comandos podem se diferenciar do Windows para o Unix

|  |  |  |
| --- | --- | --- |
|  | Windows | Unix |
| Entrar em uma pasta  Ex.: cd windows | cd | cd |
| Vai pra pasta raiz | cd / | cd / |
| Voltar 1 nível da pasta | cd .. | cd .. |
| Lista de diretórios da pasta atual | Dir | ls |
| Lista de diretórios da pasta atual ocultos |  | Ls -a |
| Lista o diretório completo |  | pwd |
| Cria uma pasta | Mkdir | Mkdir |
| Deleta apenas arquivos | Del | Rm |
| Deleta pastas e todos os arquivos | Rmdir nomedapasta /s | Rm -rf nomedapastas |
| Limpar o terminal | cls | clear |
| Criar um arquivo txt | Echo hello > hello.txt |  |

O Windows pode auto completar uma frase, apertar a tecla “tab”.

/s -- Exclui uma árvore de diretórios (o diretório especificado e todos os seus subdiretórios, incluindo todos os arquivos).

/q -- Especifica o modo silencioso. Não solicita confirmação ao excluir uma árvore de diretórios. O parâmetro **/q** só funcionará se **/s** também for especificado.

Link para o mardown: https://markdownguide.org/basic-syntax

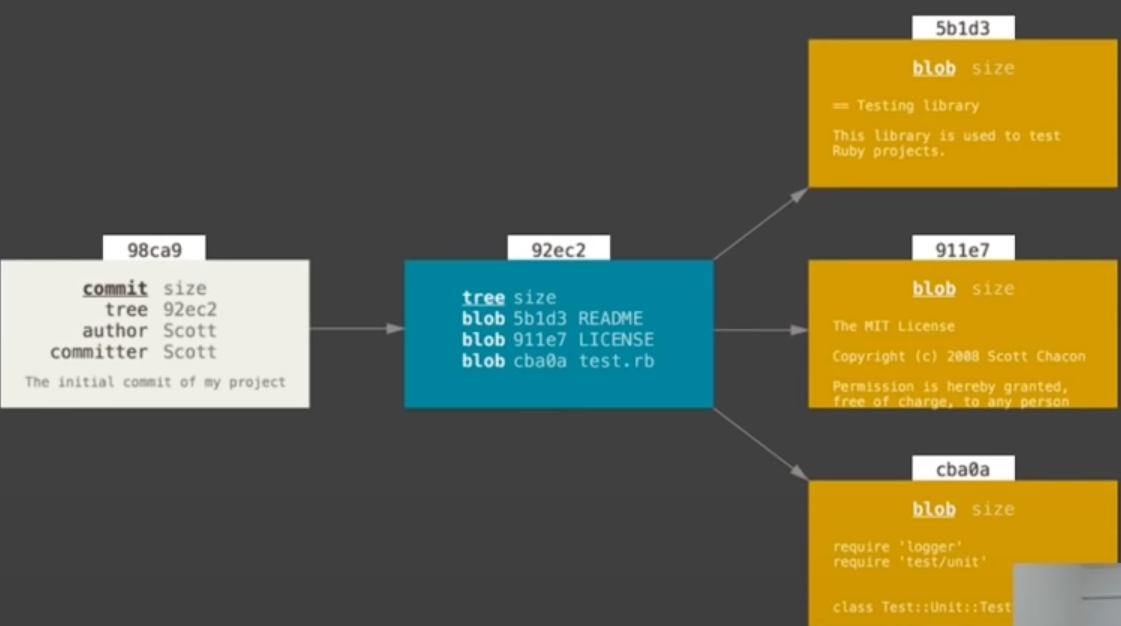
# Entendendo como o Git funciona

SHA (Secure Hash Algorithm): é um conjunto de funções hash criptográficas projetadas pela Agência de Segurança Nacional dos EUA.

A encriptação gera conjunto de caracteres identificadores de 40 dígitos.

## Objetos internos do git

* **Blobs**: objetos binários. Não registra sua data de criação, seu nome ou nada além de seu conteúdo.
* **Trees**: Uma lista de diretórios, referindo-se a **blobs** e também a outras **árvores**
* **Commits**: Único para cada autor. Um instantâneo da árvore de trabalho



## Chaves SSH e Tokens

Chave SSH:

Para iniciar no GitHub

**Criando a chave pública**

Para criar uma ssh para o GitHub do meu PC, digitar o código abaixo no Git Bash

$ ssh-keygen -t ed25519 -C jaimejaysi@gmail.com

Vai pedir uma senha e assim será gerado a chave na pasta: C/users/JJ/.ssh/id\_ed25519

Para acessar o conteúdo da chave pública criada, digitar o código abaixo no GitBash

Cd / C/users/JJ/.ssh/

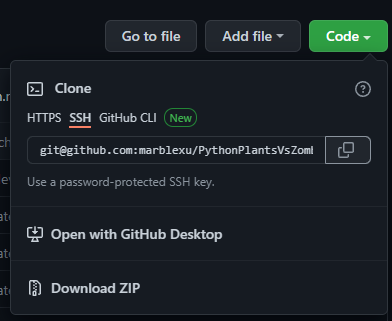
Cat id\_ed25519.pub

A minha chave pública criada foi: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIL0QuofGe9tN6fRDfRmCeSrxJq9kQsjIWRra/62F4aRT jaimejaysi@gmail.com

**Criando a chave privada e repassando para o PC**

Aval $(ssh-agent -s) -- vai ser dado um “pid” de um novo projeto

Ssh-add id\_ed25519 -- chave privada vai ser adicionada na pasta

**Para clonar um código do GitHub com o SSH**

Copie o link do código que você quer digitar.

Entre no diretório de onde você quer adicionar o código e digite:

Git clone “aqui ficará o código copiado do git”

Ele vai pedir para continuar, você vai confirmar com YES

**Para clonar um código do GitHub com Token**

Primeiro você irá precisar gerar o token no GitHub em Settings, Developer Settings, Personal acess tokens, Generate new token.

Escolha uma data que ficará ativo o seu token e o nome. Após ative a caixa “repo”, depois é só clicar em Generate Token e copiar o link do token.

Vá no seu código que você quer copiar, copie o link HTTPS.

Entre no diretório de onde você quer adicionar o código no Gitbash e digite:

Git clone “aqui ficará o código copiado do git”

Vai aparecer uma tela para colar o seu token, depois é só clicar em “sign in”. Pronto.

## Comandos do git

|  |  |
| --- | --- |
| Git init | Cria um repositório vazio “.git” dentro de uma pasta. |
| Git add | Adiciona um arquivo |
| Git commit | Adiciona um nome |

**Adicionar um arquivo no GitHub**

Dentro do diretório que você quer, escreva o comando:

Git init - Cria um repositório vazio “.git” dentro de uma pasta.

É preciso fazer uma configuração antes:

Git config –global users.email “jaimejaysi@gmail.com”

Git config –global user.name JaysiJJ

Coloque o arquivo que você quer adicionar no GitHub dentro da pasta.

Git add \*

Git commit -m “primeiro commit”

## Ciclo de vida dos arquivos no git

|  |  |
| --- | --- |
| Git status | Informações de status do repositório da pasta atual |
| mv documento.md ./receitas | Mover o arquivo “documento.md” para a pastas receitas |
|  |  |
|  |  |

**Para mover um arquivo para uma outra pasta**. Ex.: c/workspace/livre-receitas

mv documento.md ./receitas

git add strogonoff.md - é preciso, para que não ocorre erro ao ver o status do arquivo

git commit -m “cria uma pasta receitas, move arquivo para receitas” - assim o status ficará limpo.

Toda vez que houver uma modificação de um arquivo no repositório do seu pc, será necessário fazer esses passos:

Git add\* - irá adicionar todos os arquivos do repositório

Git commit -m “descrição da alteração feita”

## Trabalhando com o GitHub

Git config --list

Se o seu nome e email não estiverem escritos, será necessário adicionar

Git config --global user.name “JaysiJJ”

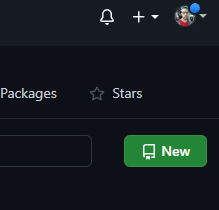
Git config --global user.email “jaimejaysi@gmail.com”

## Criando um repositório no GitHub

Clique em “your repositories”



Clique em “new”

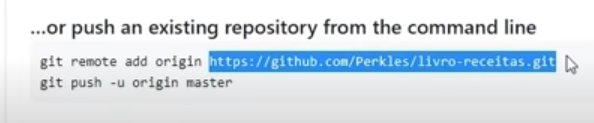


Digite o nome do repositório e uma descrição.

Se o seu repositório tiver um markdown, você não precisa selecionar a caixa “readme”.

Depois, é só clicar em “create repository”

Copie o link gerado.



Escrever os comandos:

Git remote add origin linkcopiadoacima

Git remote -v - listar as listas de repositórios cadastrados

Git status - para verificar o status

Git push origin master - irá pedir as suas credenciais / após isso ele envia para o repositório

## Adicionar alterações de um arquivo que já está dentro do GitHub

Git add\* - irá adicionar todos os arquivos do repositório

Git commit -m “descrição da alteração feita”

Git push origin master - irá pedir as suas credenciais / após isso ele envia para o repositório

(alguns repositórios estão como main e não master)

Caso alguém já tenha feito uma alteração no seu arquivo, você precisa baixar o arquivo antes de fazer a modificação:

Git pull origin master - vai baixar o arquivo modificado

Git add\* - irá adicionar todos os arquivos do repositório

Git commit -m “descrição da alteração feita”

Git push origin master - irá pedir as suas credenciais / após isso ele envia para o repositório