



## Principais estruturas aprendidas em sala usadas em Detecção de Plantas

### VETOR

```
cpp Verify Open In Editor    
  
1 const char* categorias_plantas[NUM_PLANTAS] = {  
2     "Palmeira",  
3     "Roseira",  
4     "Arruda",  
5     "Mosquitinho"  
6 };
```

A função do **vetor** é importante porque **possibilita armazenar os dados das plantas de forma organizada**, facilitando o acesso, processamento e manipulação dessas informações no sistema. No contexto de um projeto de detecção de pragas e reconhecimento de plantas, o vetor desempenha os seguintes papéis essenciais:

#### Organização dos Dados:

Cada elemento do vetor representa uma planta ou categoria, permitindo que todas as informações estejam centralizadas e sejam acessadas facilmente através de índices. Isso evita a criação de múltiplas variáveis, simplificando o código.

#### Facilidade de Acesso:

O vetor permite acessar qualquer dado armazenado de forma direta usando seu índice. Por exemplo, o nome de uma planta ou o valor de confiança pode ser recuperado com uma única instrução.

#### Iteração Eficiente:

Com loops, é possível percorrer todo o vetor para realizar operações em massa, como preenchimento, processamento de resultados ou exibição de dados.

#### Escalabilidade:

Novas plantas ou categorias podem ser adicionadas ao vetor sem alterar significativamente o código, tornando o sistema modular e fácil de atualizar.

### Integração com Outras Partes do Sistema:

Os dados armazenados no vetor podem ser utilizados para alimentar modelos de inferência, exibir resultados em interfaces ou enviar informações para servidores remotos.

O vetor pode armazenar os nomes das plantas que o sistema reconhece e os resultados das detecções realizadas, como:

```
const char* categorias_plantas[] = { "Palmeira", "Roseira", "Arruda",  
"Mosquitinho" };  
float confianca[] = { 0.85, 0.72, 0.95, 0.40 }; // Valores de  
confiança
```

Com isso, o sistema pode iterar sobre o vetor para exibir os resultados

```
for (int i = 0; i < 4; i++) {  
    Serial.printf("Planta:   %s   -   Confiança:   %.2f\n",  
categorias_plantas[i], confianca[i]);  
}
```

Saída no monitor serial:



```
Planta: Palmeira - Confiança: 0.85
```

```
Planta: Roseira - Confiança: 0.72
```

```
Planta: Arruda - Confiança: 0.95
```

```
Planta: Mosquitinho - Confiança: 0.40
```

## REPETIÇÃO

```
cpp Verify Open In Editor    
  
1 for (size_t i = 0; i < NUM_PLANTAS; i++) {  
2     resultado->classificacao[i].valor = random(0, 100) / 100.0; // V  
3 }
```

Esse é o **loop** principal do Projeto, pois permite a execução contínua das tarefas, captura de imagens e processamento de inferências.

Nesse caso em específico o que ele está fazendo é enquanto a câmera estiver ligada ele vai identificar as plantas e imprimir no monitor serial.

### **Benefícios da Repetição no Projeto**

<b>Benefício</b>	<b>Descrição</b>
<b>Automação de Tarefas Repetitivas</b>	Processa automaticamente todas as plantas ou resultados, sem necessidade de código manual redundante.
<b>Execução Contínua</b>	Permite que o sistema monitore e processe dados em tempo real sem intervenção manual.
<b>Adaptação a Mudanças</b>	O número de categorias ou sensores pode ser aumentado sem necessidade de grandes ajustes no código.
<b>Redução de Complexidade</b>	Loops eliminam a repetição de blocos de código, tornando o programa mais limpo e eficiente.