O que são?

- Implementações e funcionalidades -> você sabe o que faz e não como faz;
- Ou até, ter especificações de o quais são a funcionalidades que precisam ser implementadas e, então, cada um faz elas como quer;

De forma geral, o TAD visa dividir o código em especificações (o que ele faz) e a implementação (como faz);

Visa desvincular o **tipo de dado** (**estrutura de dados** e **operações** que as manipulam) de sua **implementação**;

- Ideia parecida com Encapsulamento em Orientação a Objetos; -> Escondemos os dados e detalhes do usuário, fornecendo apenas uma **interface pública** para manipulá-los.

Exemplo
sintaxe: int Soma (int a, int b);
semântica: Documentação:: Comentários ou lugares que explicam como as funções são usadas, etc;
Algoritmos:

```
int Soma(int a, int b){
        return a+b;
}
```

Representação:

- **Arquivo de cabeçalho (.h) -> especificação**
  seu_tad.h
- **Arquivo fonte (.c) -> implementação**
  seu_tas,c

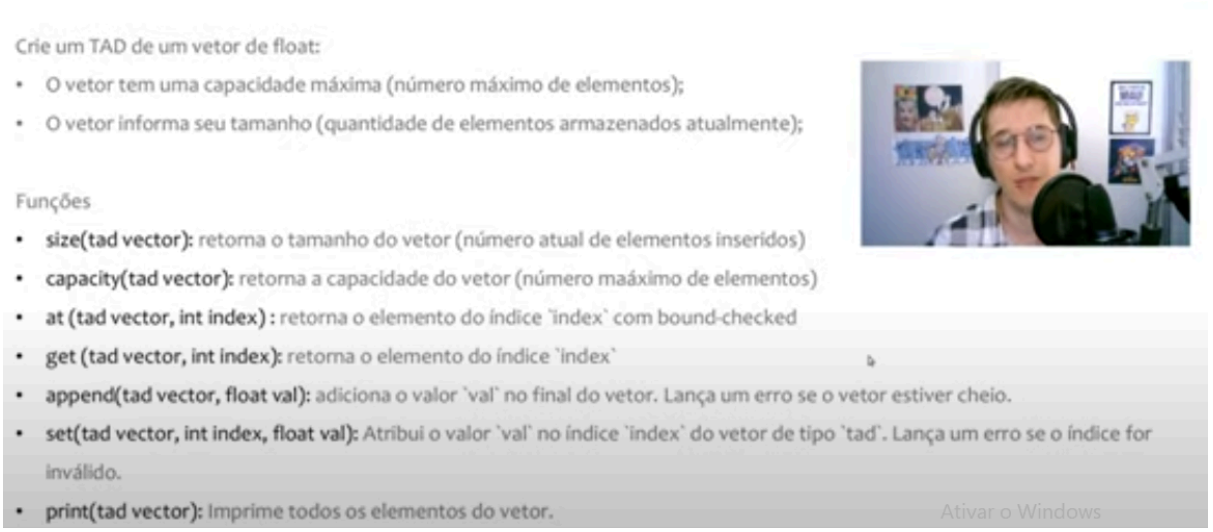
Os programas ou outras TADs que utiliuzam seuTAD devem incluir sua especificação:
#include "seu_tad.h" -> Ou seja, inclui somente as especificações

Vantagens

- reutilização:

- facilidade de manutenção: mudanças na implementação do TAD não afetam o código fonte dos programas que o utilizam
- corretude: códigos testados em diferentes contextos
- Não é possível criar uma função que tenha outra função de mesmo nome em alguma biblioteca que esteja definida no código; -> por isso, é bom colocar (nome do projeto_nome da função)

Implementação:



Encapsulamento em C

Como esconder certas coisas do usuário e provê pra ele outras coisas?
- > Tudo o que você definir no .c não estará disponível para o usuário;
-> As funções em .h que retornam os valores de .c servem justamente para se ter acesso a esses valores sem ter acesso direto a como eles são feitos, estruturados, etc;

- É possível fazer funções “privadas”, ou seja, só existirão no .c. Dessa forma, ela só será utilizada pelo seu .c -> Elas são úteis quando você precisa fazer uma função para lidar com algumas características do seus algorítmos e você não vê sentido em deixar essa função disponível para uso, já que ela é mais de uso interno. -> Colocar um ‘_’ antes do nome para nós, desenvolvedores, podermos diferenciar das outras.

  O que o usuário pode ver (.h) -> Interface pública
  O que o usuário não pode ver (.c) -> Interface Privada

Evitando Múltiplos Includes (Include Guards)

SEMPRE USAR NO INÍCIO DOS SEUS TADS NO .H!

Colocar no código:

```
#ifndef (ver melhor no vídeo)
#define (ver melhor no vídeo)
        struct
        {
        }


#endif //Se ela já foi definida, ele irá “cagar” para tudo o que foi pra cima
```