

Tipos abstratos de dados

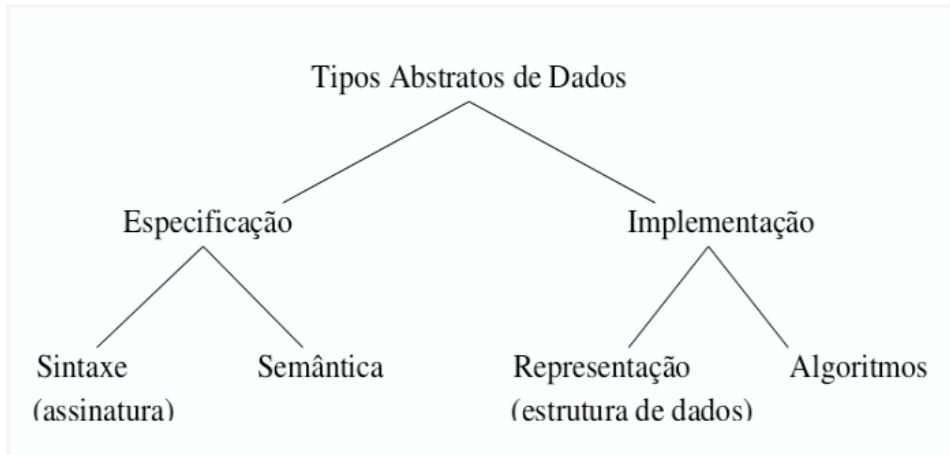
Abstração

- capacidade de concentrar-se nos aspectos essenciais de um contexto qualquer, ignorando características de menor importância ou acidentais
- ao definir um Tipo Abstrato de Dados, são considerados os aspectos essenciais do tipo de dado necessário e das operações que serão realizadas sobre ele

Tipo de Dado

- Quando o conceito de "tipo de dado" é dissociado dos recursos do hardware do computador, um número ilimitado de tipos de dados pode ser considerado.
- Um tipo de dado é um conceito abstrato, definido por um conjunto de propriedades lógicas.
- Assim que um tipo de dado abstrato é definido e as operações válidas envolvendo esse tipo são especificadas, podemos implementar esse tipo de dado (ou uma aproximação).

Especificação x implementação



separação entre especificação e implementação



uso do TAD sem conhecer nada sobre a sua implementação



um TAD pode ter mais que uma implementação

TIPOS DE DADOS ABSTRATOS

- Fundamentalmente, um tipo de dado significa um conjunto de valores e uma sequência de operações sobre esses valores.
- Esse conjunto e essas operações formam uma construção matemática que pode ser implementada usando determinada estrutura de dados do hardware ou do software.

TIPOS DE DADOS ABSTRATOS

- Ao definir um tipo de dado abstrato como um conceito matemático, não nos preocupamos com a eficiência de tempo e espaço.
 - Essas são questões de implementação.
- Na realidade, a definição de um TDA não se relaciona com nenhum detalhe da implementação.
- É possível até que não se consiga implementar determinado TDA num determinado hardware ou usando determinado sistema de software.
- Por exemplo, já constatamos que o TDA *inteiro* **não é universalmente implementado**. Apesar disso, especificando-se as propriedades matemáticas e lógicas de um tipo de dado ou estrutura, o TDA será uma diretriz útil para implementadores e uma ferramenta de apoio para os programadores que queiram usar o tipo de dado corretamente.

Pilhas




implementação sobre vetores

Prof. Andreia Machion

Definição

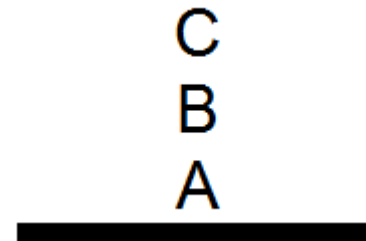
- Pilha é uma estrutura para armazenar um conjunto de elementos, que funciona da seguinte forma:
 - Novos elementos entram no conjunto, exclusivamente, no topo;
 - O único elemento que posso retirar da pilha em um dado momento, é o elemento do topo.
- Do Inglês: *Stack, LIFO*
 - Uma Pilha (em Inglês: *Stack*) é uma estrutura que obedece o critério L.I.F.O.: *Last In, First Out*. Ou seja, o último elemento que entrou no conjunto será o primeiro a sair.

Quer ver?

Pilha de Potes	Pilha de Livros	Pilha de Pratos	Pilha de Cartas
			
Imagens: Yew Tree Gallery (potes), iStockPhoto (livros), Darren Maurer (pratos), e Barbosa, Miyoshi, e Gomes (cartas)			

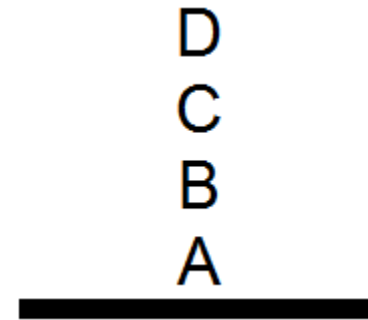
Idéias de operação

- Uma pilha é um conjunto ordenado de elementos, ou seja, a ordem dos elementos no conjunto é importante. Se eu tenho três elementos em uma pilha, A, B e C, e se eles entraram na pilha nessa ordem, o elemento que estará no topo da pilha será o elemento C. E se eu quiser retirar um elemento nesse momento, o único elemento que poderei retirar da pilha será exatamente o elemento C



Idéias de operação

- Considerando ainda o exemplo da figura anterior, se quisermos inserir um elemento D na pilha nesse momento, este elemento passaria a ser o elemento do topo da pilha, conforme mostra a figura ao lado



Para que serve uma pilha?

Exemplo da chamada de subprogramas

- Ao elaborar um programa de computador você já se deparou com um erro de execução chamado *stack overflow*? Sabe o que significa *stack overflow*? Sabe como ocorre? É um erro bastante comum. Se ainda não aconteceu com você, provavelmente ainda vai acontecer.
- Um computador está executando um trecho de programa A, e durante a execução de A encontra o comando *call* B, ou seja, uma chamada a um subprograma B. O computador precisa parar de executar A, executar B, e retornar ao programa A no ponto onde parou. Como o computador faz para lembrar a posição exata para onde deve retornar? Essa questão pode se complicar se tivermos várias chamadas sucessivas.

Na prática...

- Considere o exemplo da tabela abaixo. Temos um programa principal, A, e 3 subprogramas: B, C e D.
- Ao iniciarmos a execução de A, na linha 1 temos o comando *print* A, que imprime a letra A. Depois temos o comando *call* C, ou seja, uma chamada ao subprograma C.
- Nesse ponto temos que interromper a execução de A e iniciar a execução do subprograma C. Ao finalizar a execução do subprograma C (comando *return*, subprograma C linha 3), precisamos retomar a execução de A na linha 3, porque as linhas 1 e 2 de A já foram executadas.
- Vamos simular!

Programa A (principal)	Subprograma B	Subprograma C	Subprograma D
1 print A	1 call C	1 print C	1 print D
2 call C	2 print B	2 call D	2 return
3 call B	3 call D	3 return	
4 call D	4 call C		
5 return	5 return		

Videoaulas de apoio

- Professor André Backes
 - Aula 01 - TAD (Tipo Abstrato de Dado)
 - https://www.youtube.com/watch?v=bryesHll0vY&list=PL8iN9FQ7_jt6H5m4Gm0H89sybzR9yaaka
 - Aula 38 - Pilha: Definição
 - https://www.youtube.com/watch?v=2RCrd7gOUMM&list=PL8iN9FQ7_jt6H5m4Gm0H89sybzR9yaaka&index=38
 - Aula 39 - Pilha Estática
 - https://www.youtube.com/watch?v=OIQJL-K2SUI&list=PL8iN9FQ7_jt6H5m4Gm0H89sybzR9yaaka&index=39
- UNIVESP
 - Estrutura de Dados - Aula 8 - Pilha - implementação estática
 - <https://www.youtube.com/watch?v=ruOzUIA4rbs&list=PLxl8Can9yAHf8k8LrUePyj0y3LLpigGcl&index=9&t=0s>