Insper

Sistemas Hardware-Software

Aula 12 - Tipos abstratos de dados

Engenharia
Fabio Lubacheski
Maciel C. Vidal
Igor Montagner
Fábio Ayres

Tipos de dados

Um **tipo de dados** (=*data type*) é um conjunto de **valores** acompanhado de um conjunto de **operações** que podem ser aplicadas a esses valores.

Um **tipo de dados** está ligado à **implementação concreta**, isto é, como o **compilador** e o **hardware** entendem aquele tipo, envolvendo a **representação na memória** e as r**estrições de implementação**, por exemplo o **tamanho**.

Tipos de dados

Um exemplo de **tipo de dados** na linguagem C é o **`int`** para **valores inteiros**.

Na arquitetura **x86-64** tem **32** bits (**tamanho**) e com valores de -2^{31} **a** $2^{31}-1$ e são armazenados na memória usando a **representação Little endian**.

Para o tipo de dados int tempos as **operações** de **comparação** (<, <=, ==, >, >=) e as **aritméticas** (+, -, *, /).

Tipos Abstratos de Dados

Um **tipo abstrato de dados** (=*abstract data type*) é um **tipo de dados** define um conjunto de **valores** e um conjunto de **operações** possíveis sobre esses valores, mas sem indicar como esses valores são armazenados ou como as operações são implementadas.

O objetivo é descrever uma **interface** que diga o que cada **operação faz**, sem dizer como deve ser feito.

Exemplo o TAD – Pilha (=Stack)

No **TAD Pilha** temos as seguintes **operações**: **inserção** de um item (**Push**) e **remoção** de um item mais recente (**Pop**).

Os **itens** podem tidos de dados (`int`,`char`) e outros TADs

Comportamento: o **último** item a entrar é o **primeiro** a sair (**LIFO**).

A **abstração** está no fato que o **TAD Pilha** pode ser implementada com **vetor**, **lista encadeada**, ou até usando duas filas, não importa !

TAD – Pilha (=Stack) - Interface

Os programas que usam o TAD só tem acesso a **interface**. Na linguagem **C** a interface é representada por um arquivo-interface (**=header file**).

```c
// stack.h
typedef struct {
    int capacity; // capacidade máxima
    int size; // quantidade de elementos armazenados
    int *data; // vetor para armazenar itens do TAD
}stack_int;

stack_int * stack_int_new(int capacity);
void stack_int_delete(stack_int *_s);
int stack_int_empty(stack_int *s);
int stack_int_full(stack_int *s);
void stack_int_push(stack_int *s, int value);
int stack_int_pop(stack_int *s);
```

TAD – Pilha (=Stack) – Implementação

A implementamos das funções do **arquivo .h** ficam em um **arquivo .c**, por convenção de mesmo nome.

```c
#include "stack.h"

stack_int *stack_int_new(int capacity) {
    stack_int *s = malloc(sizeof(stack_int));
    if (s == NULL) {
        printf("Erro de alocação de memoria para estrutura!\n");
        exit(EXIT_FAILURE);
    }
    s->capacity = capacity;
    s->size = 0;
    s->data = (int *)malloc(capacity * sizeof(int));
    if (s->data == NULL) {
        free(s);
        printf("Erro de alocação de memoria para os dados!\n");
        exit(EXIT_FAILURE);
    }
    return s;
}
```

TAD – Pilha (=Stack) – Testando

Para testar a implementação podemos criar um programa com a função **`main()`** que utiliza o TAD.

```c
// test_stack.c
// gcc -Og -Wall -g test_stack.c stack.o -o teste_stack
#include <stdio.h>
#include "stack.h"

int main() {
    // Cria um stack com capacidade para 5 elementos
    stack_int *s = stack_int_new(5);

    stack_int_push(s, 10);
    stack_int_push(s, 20);
    stack_int_push(s, 30);
    // Remove o topo: 30
    printf("Valor removido: %d\n", stack_int_pop(s));
```

Tipos Abstratos de Dados

- Conjunto de dados e operações
  - arquivo ***.h***
- Criação de algoritmos com essas operações
  - Não depende de detalhes internos

Tipos Abstratos de Dados

- Vantagens:
  - Código mais expressivo
  - Diminui erros por repetição
  - Evita deixar struct em estado inconsistente

Tipos Abstratos de Dados

- Desvantagens:
  - Esconde todos os detalhes
  - Não permite usos mais avançados ou diferentes do original

Atividade prática

Implementação de Point2D (30 minutos)

1. Revisão de malloc
2. Compilação de programas com mais de um arquivo .c

Vetor dinâmico - Atividade para Entrega

O tipo de dados **vetor dinâmico** é implementado em diversas linguagens de alto nível.

- Python: `list`
- Java: `ArrayList`
- C++: `std::vector`

Vetor dinâmico

Suas principais operações são

- criação/destruição
- **at(i)** – devolve elemento na posição i
- **remove(i)** – remove o elemento na posição i, deslocando todos os outros para a esquerda
- **insert(i)** – insere um elemento na posição i, deslocando todos os elementos para a direita

Vetor dinâmico

As operações abaixo mudam o tamanho do vetor!

- **remove(i)** – remove o elemento na posição i, deslocando todos os outros para a esquerda
- **insert(i)** – insere um elemento na posição i, deslocando todos os elementos para a direita

Não é preciso declarar tamanho para o vetor dinâmico

Vetor dinâmico

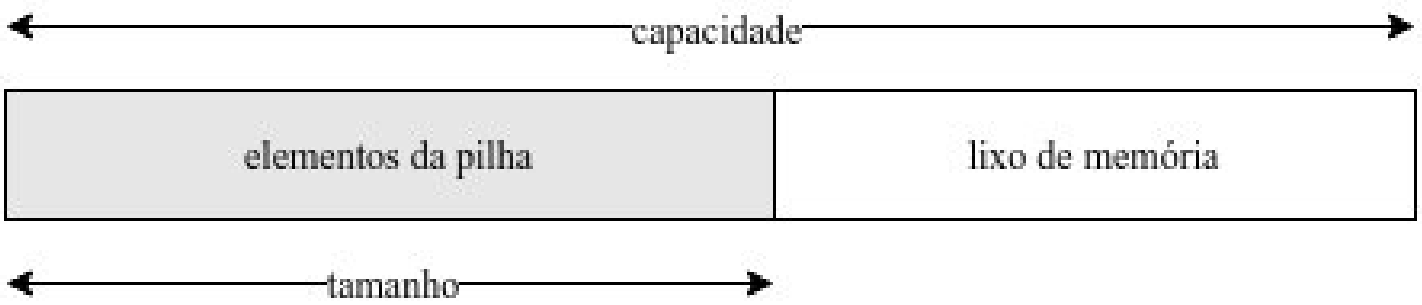
Capacidade

Relembrando *Desafios*

Supondo que soubéssemos o tamanho máximo que o vetor dinâmico assumiria, podemos aplicar esta técnica

Vetor dinâmico

E se `tamanho == capacidade`?



Bom, nesse caso precisamos de um espaço de memória maior para nosso vetor!

`realloc`

#include <stdlib.h>

void *realloc(void *ptr, size_t new_size)

Se bem sucedido: aloca um novo bloco de tamanho **new_size**, copia o conteúdo apontado por **`ptr`** para o novo bloco e retorna seu endereço. Antes de retornar chama **`free(ptr)`**.

Se falhou: retorna **`NULL`** e preenche **`errno`**

Vetor dinâmico

- **Quando encher**: dobrar capacidade
- Quando ficar com **menos de um quarto** da capacidade: **diminuir** a capacidade pela **metade**

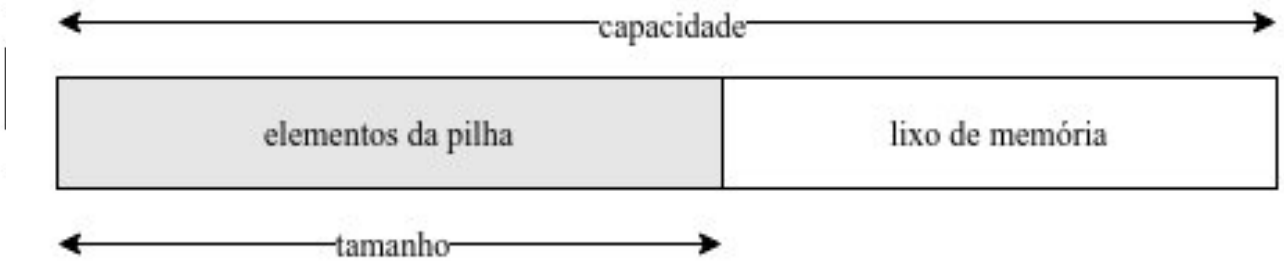
Vetor dinâmico

E se `tamanho == capacidade`?

1) Criamos um novo espaço de memória e copiamos o conteúdo para lá com realloc

2) Atualizamos a nova capacidade

3) Atualizamos o ponteiro para os novos dados



Insper

Atividade para entrega

Implementação de Vetor dinâmico (Entrega)

1. Revisão de malloc
2. Compilação de programas com mais de um arquivo .c
3. Entender uso de um TAD a partir de exemplos de uso

Insper

www.insper.edu.br