

Estrutura de Dados

Tipos de dados abstratos e
estruturas de dados

Tipos de Dados Abstratos (TDA)

- É um conjunto de valores e uma coleção de operações que atuam sobre esses valores.
- As operações devem ser consistentes com os tipos de valores.
- O **TDA** possibilita ao programador a separação “**do que fazer**” de “**como fazer**”.

Tipos de Dados Abstratos (TDA)

- Exemplo:
 - Um programador necessita gravar um registro em um arquivo.
 - Ele só precisa saber “**o que deve ser feito**”.
 - É irrelevante para ele saber **como** ocorrem todas as operações que efetivamente realizam o processo de gravação do registro.

Tipo de Dados Abstratos (TDA)

- Ao **programador** basta saber que é preciso usar um comando (**write** – por exemplo) e o processo de gravação do registro ocorrerá de forma adequada.
- Esse “**write**” representa um **TDA**.
- Nos **primórdios da computação**, se um **programador** precisasse gravar um registro, ele teria de especificar todos os passos necessários.

Tipos de Dados Abstratos (TDA)

- Um TDA geralmente envolve:
 - Uma Estrutura de Dados (ex: Pilha);
 - Operações ou funções (ex:
 - criar pilha vazia,
 - testar se pilha está vazia;
 - Empilhar um novo elemento no topo da pilha;
 - Desempilhar um elemento do topo da pilha (desde que não esteja vazia).
 -).

A Estrutura de Dados

- Todo trabalho realizado por um **computador** é baseado na manipulação das **informações** contidas na **memória principal**.
- Essas **informações** podem ser classificadas em dois tipos:
 - **Instruções** – Comandam o funcionamento da máquina e determinam como devem ser tratados os dados.
 - **Dados** – Correspondem à porção das informações a serem processadas pelo computador.

Estruturas de Dados

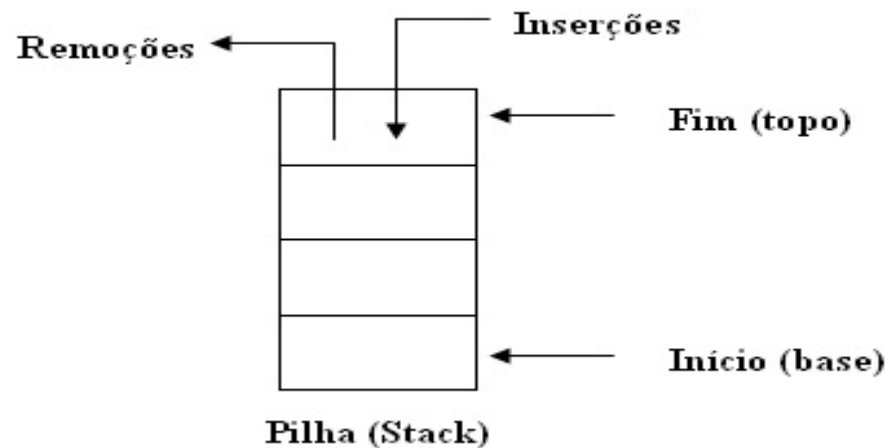
- Definição de estrutura de dados:
 - A organização e a representação das informações.
 - Algumas estruturas clássicas:
 - Pilhas;
 - Filas;
 - Listas Encadeadas;
 - Vetores;
 - Árvores;

Estrutura de Dados

Pilhas (em Linguagem C)

Pilhas (stacks)

- Uma Pilha é uma lista linear na qual todos os acessos (inserção, remoção ou consulta) são realizados em uma só extremidade, denominada TOPO.



Pilhas (stacks)

- Existe uma disciplina de acesso – um padrão de comportamento.
- O último elemento a ser inserido (**empilhado**) é o primeiro elemento a ser removido (**desempilhado**).
- As **pilhas** são uma estrutura **LIFO** (“**Last In, First Out**”).

Pilhas (stacks)

- **Analogia:** uma pilha de pratos.
- Uma pilha de pratos aumenta ou diminui por um único lado: seu **topo**.



Pilhas (stacks)

- Outra analogia: um estacionamento de trens (E. W. Dijkstra).
- Possibilidade de realização de permutações das composições de trens.
- As pilhas são utilizadas de forma semelhante, em situações de reversão da ordem de entrada de dados.

Pilhas – Estacionamento de Trens

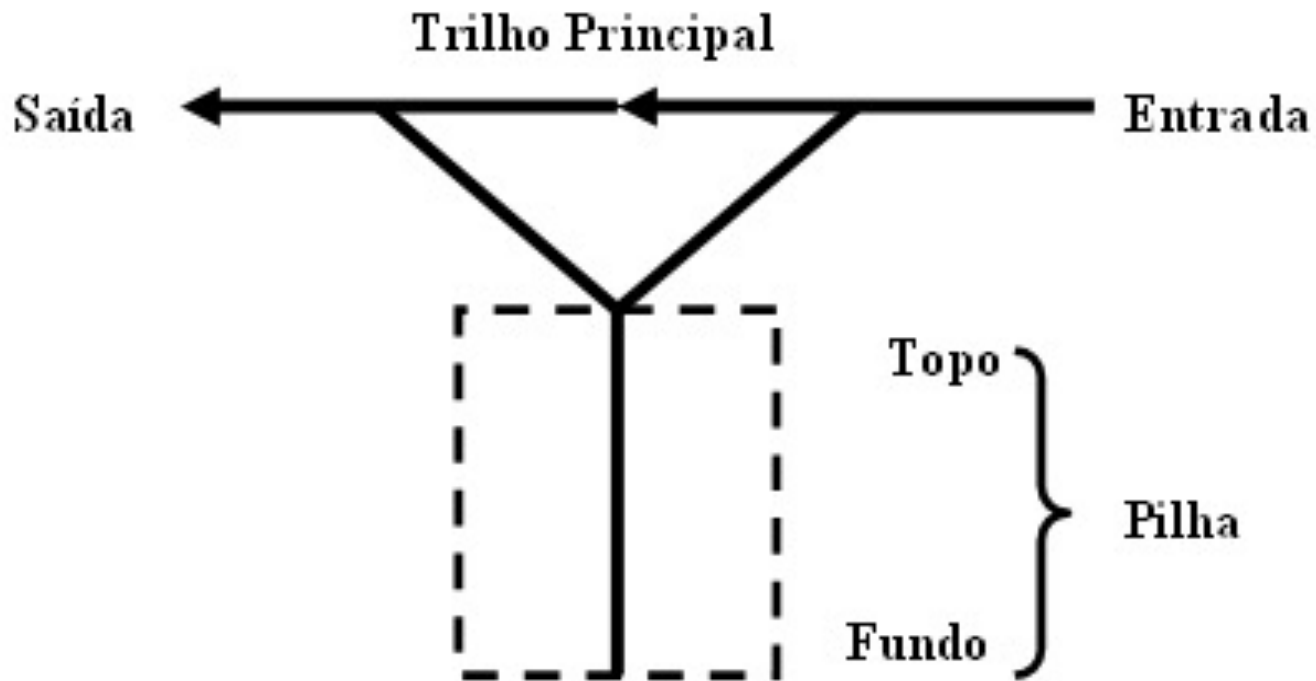
- Os **trens** somente podem entrar ou sair do **estacionamento** apenas por um lado.
- Esse lado funciona como o **topo** de uma **pilha**.
- Toda vez que um **trem** está entrando ou saindo, está sendo realizado o **empilhamento** ou **desempilhamento** de elementos em uma **pilha**.

Pilha – Estacionamento de Trens



<http://vfco.com.br/>

Pilhas – Estacionamento de Trens



Pilhas – Estacionamento de Trens

- Suponha que existam 3 trens na entrada, numerados na sequência de 1, 2 e 3, e nessa respectiva ordem.
- Os três podem sair na ordem 2, 1 e 3?
- E na ordem 3, 1 e 2?
- E na ordem 2, 3 e 1?

Operações Primitivas que Manipulam as Pilhas

- **InicPilha(s)**
 - Faz a Pilha s ficar vazia.
- **PilhaVazia(s)**
 - Retorna V se a Pilha está vazia.
- **PilhaCheia(s)**
 - Retorna V se a Pilha está cheia.
- **Empilha(s,x) ou push(s,x)**
 - Insere o elemento x no topo da pilha s.

Operações Primitivas que Manipulam as Pilhas

- Desempilha(s) ou pop(s)
 - Remove o elemento do topo da pilha s,
 - Retornando o elemento como valor da função.
- ElemTopo(s)
 - Acessa um elemento do topo da pilha s,
 - Sem, contudo, remover o elemento.

Representação de Pilhas com Vetor

- Pode-se representar uma pilha usando um vetor.
- O **vetor** apresenta uma estrutura que pressupõe **a definição prévia da quantidade máxima de elementos** que existirá em seu interior; portanto, ele possui **um número fixo de elementos**.

Representando Pilhas com Vetores

- A **Pilha**, ao contrário, é um **objeto dinâmico**, porque o **número de elementos aumenta ou diminui** à medida que **empilhamos** ou **desempilhamos** elementos.
- Quando um **vetor** é empregado para representar uma **pilha**, surge a possibilidade de ocorrência de **overflow**.

Representando Pilhas com Vetores

- A possibilidade de um **overflow** (*tentativa de inserir elementos em uma pilha cheia*) é evitada quando utilizamos uma **alocação dinâmica de memória (ponteiros)**.
- O **overflow** somente ocorre na **alocação dinâmica de memória** quando a **pilha encontra limitação de memória disponível no computador**.

Representando Pilha com Vetor

- Para representar uma **pilha** utilizando-se da **linguagem C**, podemos construir uma **estrutura (struct)** chamada **stack**.

```
#include <stdio.h>
#include <stdlib.h>

#define MAXPILHA 100

struct stack {
    int topo;
    int item[MAXPILHA];
};
```

Representando Pilhas com Vetores

- O campo **topo** é utilizado para sabermos qual é o **elemento** do topo da pilha.
- É o topo da pilha que se movimenta conforme são empilhados ou desempilhados os elementos.

Representando Pilhas com Vetores

```
void inicPilha(struct stack *ps){
    ps->topo = -1;
}

int pilhaVazia(struct stack *ps){
    if (ps->topo == -1)
        return (1); //Em C qualquer valor diferente de zero é verdadeiro
    else
        return (0); //Em C o valor zero é falso
}

int pilhaCheia(struct stack *ps){
    if (ps->topo == MAXPILHA - 1)
        return (1);
    else
        return (0);
}
```


Representando Pilhas com Vetores

```
int push(struct stack *ps, int x) {  
    //Empilhando Elemento na Pilha  
    if (pilhaCheia(ps) == 1){  
        printf("%s", "Ocorreu OVERFLOW na Pilha!\n");  
        exit(1);  
    } else {  
        return (ps->item[++(ps->topo)] = x);  
    }  
}  
  
int pop(struct stack *ps) {  
    //Desempilhando elemento da Pilha  
    if (pilhaVazia(ps) == 1) {  
        printf("%s", "Ocorreu UNDERFLOW na Pilha!\n");  
        exit(1);  
    } else {  
        return (ps->item[ps->topo --]);  
    }  
}
```

Representando Pilhas com Vetores

```
int elemTopo( struct stack *ps) {
    if (pilhaVazia(ps) == 1){
        printf("Ocorreu UNDERFLOW na Pilha!\n");
        exit(1);
    } else {
        return (ps->item[ps->topo]);
    }
}

int main(){
    struct stack s;
    inicPilha(&s);

    push(&s, 55);
    push(&s, 33);
    push(&s, 12);
    push(&s, 45);
    push(&s, 88);

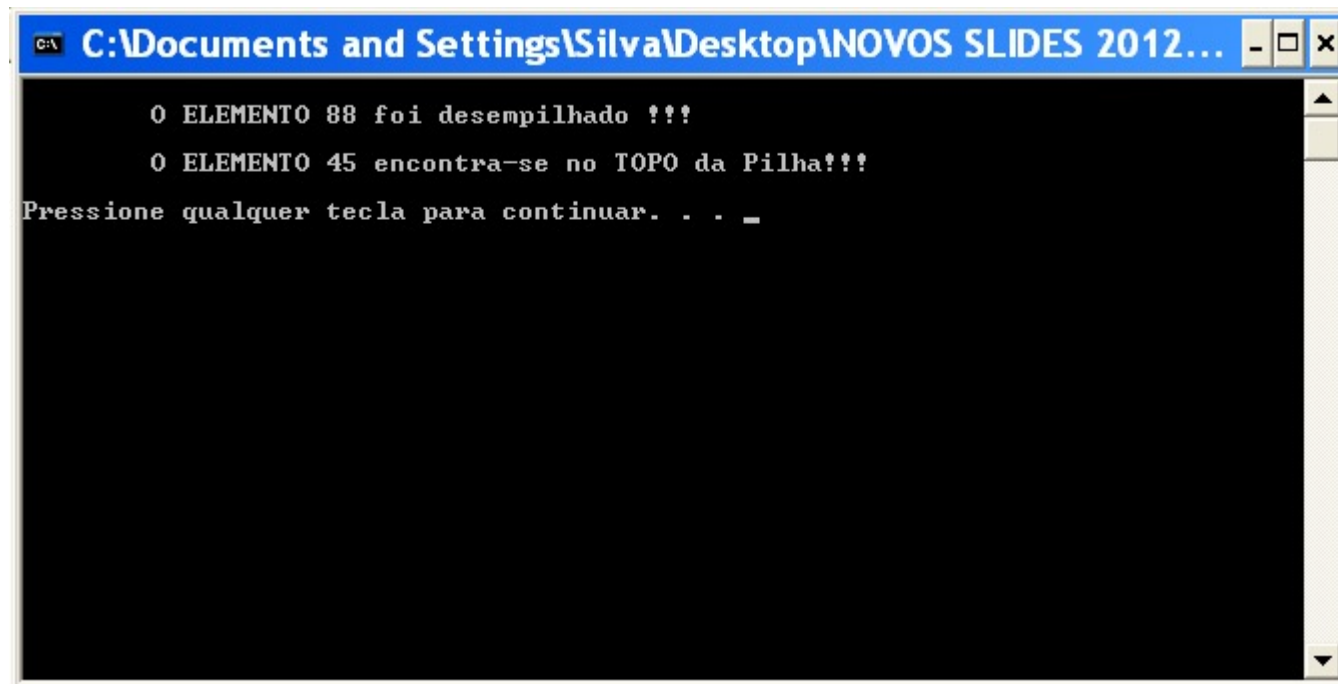
    int elemento = pop(&s);
```

Representando Pilhas com Vetores

```
printf("\n\tO ELEMENTO %d foi desempilhado !!!\n\n", elemento);  
  
printf("\tO ELEMENTO %d encontra-se no TOPO da Pilha!!!\n\n", elemTopo(&s));  
  
system("PAUSE");  
  
}
```

Representando Pilhas com Vetores

- Eis a execução de nosso programa:



A screenshot of a Windows command prompt window. The title bar at the top is blue and contains the text "C:\Documents and Settings\Silva\Desktop\NOVOS SLIDES 2012..." followed by standard window control buttons (minimize, maximize, close). The command prompt area has a black background with white text. The output of the program is as follows:

```
O ELEMENTO 88 foi desempilhado !!!  
O ELEMENTO 45 encontra-se no TOPO da Pilha!!!  
Pressione qualquer tecla para continuar. . . _
```

Uso Prático das Pilhas

- Uma **aplicação imediata das pilhas** é na **reordenação** de um **conjunto de dados**, tal que o primeiro e o último elemento devem ser trocados de posição e todos os elementos entre o primeiro e o último devem ser relativamente trocados.

Pilhas em Reordenação de Valores

- Alteremos o corpo principal do programa (main) anterior para:

```
int main(){
    struct stack s;
    inicPilha(&s);
    int numero;

    printf("\n\tInforme NUMERO inteiro (999 interrompe entrada de dados): ");
    scanf("%d", &numero);

    while ((numero != 999) && (pilhaCheia(&s) == 0)){

        push(&s, numero);

        printf("\n\tInforme NUMERO inteiro (999 interrompe entrada de dados): ");
        scanf("%d", &numero);
    }

    printf("\n\nO INVERSO da ENTRADA de DADOS:\n\n");
```

Pilhas em Reordenação de Valores

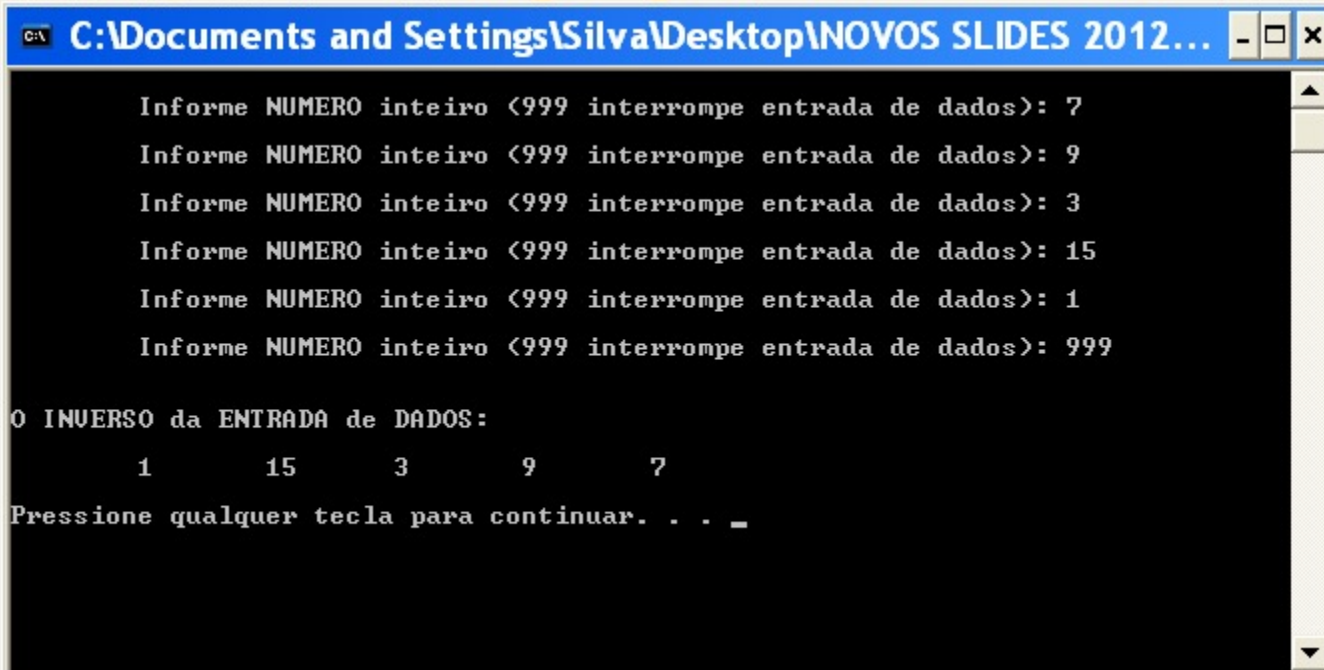
```
while (pilhaVazia(&s) == 0) {  
    numero = pop(&s);  
    printf("\t%d", numero);  
}
```

```
printf("\n\n");  
system("PAUSE");
```

```
}
```

Pilhas em Reordenação de Valores

- Exemplo de execução desse código:



```
C:\Documents and Settings\Silva\Desktop\NOVOS SLIDES 2012... - [ ] X

Informe NUMERO inteiro <999 interrompe entrada de dados>: 7
Informe NUMERO inteiro <999 interrompe entrada de dados>: 9
Informe NUMERO inteiro <999 interrompe entrada de dados>: 3
Informe NUMERO inteiro <999 interrompe entrada de dados>: 15
Informe NUMERO inteiro <999 interrompe entrada de dados>: 1
Informe NUMERO inteiro <999 interrompe entrada de dados>: 999

O INVERSO da ENTRADA de DADOS:
      1      15      3      9      7
Pressione qualquer tecla para continuar. . . _
```


Uso Prático das Pilhas

- Essa **reversão de dados** pode ser usada na resolução de um problema clássico como o que faz uma **conversão de um número na base decimal para a sua base binária**.
- **Exercício Proposto 1:** Adapte o nosso programa exemplo para obter essa solução.

Solução do Exercício Proposto 1

- 35 na base 10 = ? Na base 2
- $35 : 2 = 17$ resto 1
- $17 : 2 = 8$ resto 1
- $8 : 2 = 4$ resto 0
- $4 : 2 = 2$ resto 0
- $2 : 2 = 1$ resto 0
- $1 : 2 = 0$ resto 1
- 35 na base 10 = 100011 na base 2

Solução do Exerc. Prop. 1

```
int main(){
    struct stack s;
    inicPilha(&s);
    int numero, quociente, resto;

    printf("\n\tInforme NUMERO inteiro e positivo na BASE 10: ");
    scanf("%d",&numero);

    quociente = numero;

    while (quociente > 0){
        resto = (quociente % 2);
        quociente = (quociente / 2);

        push(&s, resto);
    }//while

    printf("\n\n\tO correspondente de %d na BASE 10 = ", numero);

    while (pilhaVazia(&s) == 0){
        resto = pop(&s);
        printf("%d", resto);
    }//while

    printf("\n\n");
    system("PAUSE");
}
```

Atividade

- Implementar uma biblioteca de pilha
 - Bibliotecas em C:
 - Arquivo Header (.h)
 - Contém definição das estruturas (structs) e assinaturas dos métodos
 - Arquivo de implementação (.c)
 - Traz as implementações dos métodos
 - Deve incluir o header (`#include xxx.h`)
 - Utilizando a biblioteca implementada, criar um programa que leia um número na base decimal e transforme para binário.