Programação 2 _ T5

Metodologias de desenvolvimento – programação modular e depuração

Estrutura lineares – filas e pilhas.

Rui Camacho
(slides por Luís Teixeira)
MIEEC 2020/2021

PROGRAMAÇÃO MODULAR

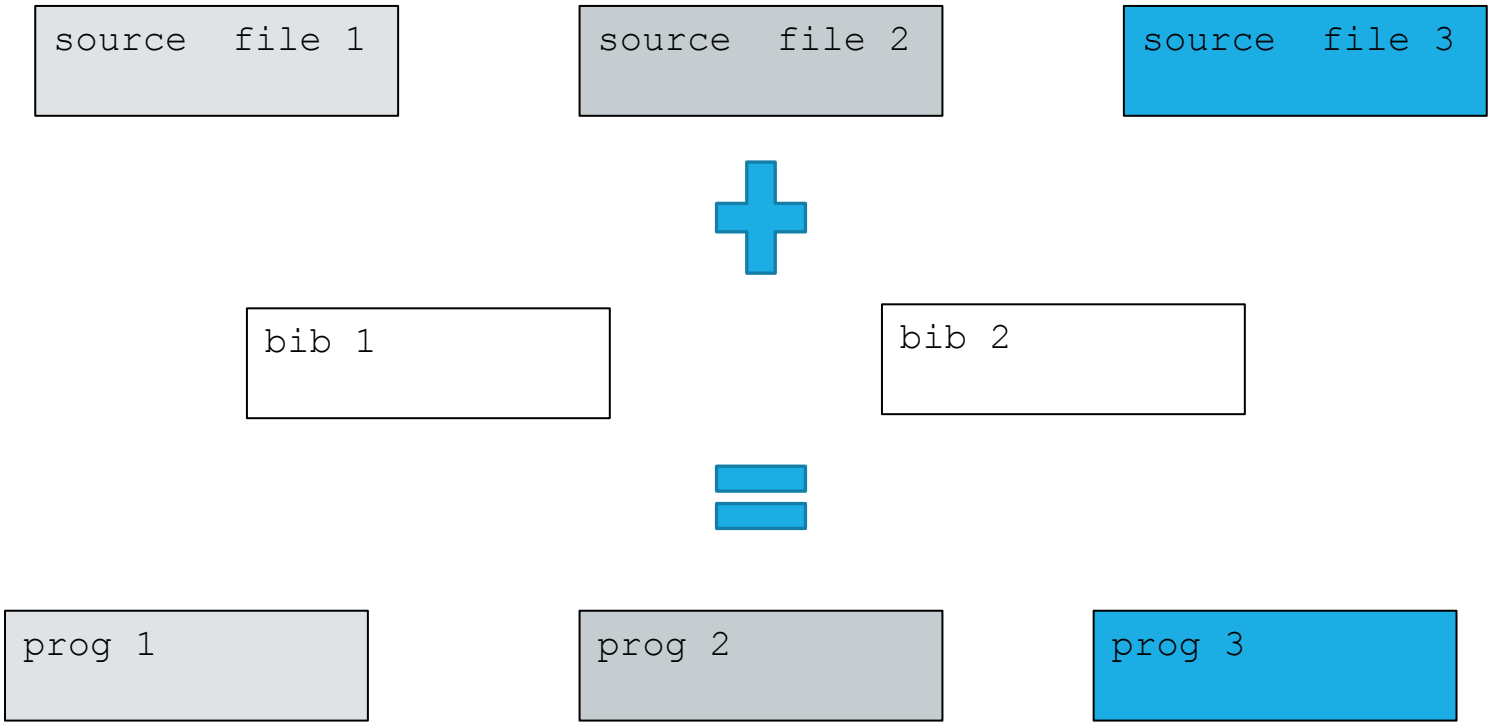
Código organizado em componentes separados que podem ser **reutilizados** → **módulos**

Manutenção do código é facilitada ao serem definidas fronteiras lógicas entre componentes

Cada módulo:

- Representa um conjunto de funcionalidades coerentes ou está associado a um **conceito** (por exemplo, um vetor)
- É utilizado através de **interfaces** conhecidas
- Tem uma implementação das funcionalidades que não necessita ser conhecida

BIBLIOTECAS DE FUNÇÕES: MODULARIDADE E REUTILIZAÇÃO EM C



BIBLIOTECA DE FUNÇÕES: EXEMPLO

quad.h

```
float sqr(float x);
// não esquecer o ';'
```

quad.c

```
float sqr(float x){
    return x*x;
}
```

testquad.c

```
#include <stdio.h>
#include "quad.h"

int main() {
    float a;
    printf("Escreva um real: ");
    scanf("%f", &a);
    printf("Quadrado = %f\n", sqr(a));
}
```

```
$ ls
quad.c quad.h testquad.c
$ clang -c quad.c
$ ls
quad.c quad.h quad.o testquad.c
$ clang testquad.c quad.o
$ ls
a.out quad.c quad.h quad.o testquad.c
$./a.out
```

compilação separada da biblioteca de funções

compilação do programa com a biblioteca de funções

GESTÃO PROCESSO DE COMPILAÇÃO

Programas ou bibliotecas mais complexas exigem a utilização de ferramentas de gestão do processo de compilação

- Make/Makefiles
  - Com base num ficheiro de configuração (Makefile) que compila automaticamente programas e bibliotecas
  - Gere dependências entre ficheiros com código fonte
- Integrated Development Environments (IDEs)
  - Para além da gestão do processo de compilação, incluem ferramentas de *debugging* (depuração) e *profiling* (análise do programa)

MAKEFILE

é obrigatório colocar um TAB

regra para compilar executável

regras para compilar ficheiros individuais .c

regra para apagar ficheiros compilados

Makefile

```
testquad: testquad.o quad.o
    clang testquad.o quad.o -o testquad

quad.o: quad.c quad.h
    clang -c quad.c

testquad.o: testquad.c quad.h
    clang -c testquad.c

clean:
    rm testquad quad.o testquad.o
```

```
$ make
clang -c testquad.c
clang -c quad.c
clang testquad.o quad.o -o testquad
$ ls
Makefile  quad.c quad.h quad.o testquad testquad.c testquad.o
$ make
make: `testquad' is up to date.
$ make clean
rm testquad quad.o testquad.o
$ ls
Makefile  quad.c quad.h testquad.c
```

IDE

Visual Studio

- Ferramenta principal de desenvolvimento para Windows, Windows Mobile, .NET, Silverlight, XNA
- Windows

Eclipse

- Open-source, implementado em Java
- Ferramenta principal de desenvolvimento para Android
- Windows, Linux, Mac OS X

Xcode

- Ferramenta principal de desenvolvimento para OSX e iOS
- Mac OS X

CodeBlocks

- Open-source, implementado em C++
- Mais simples mas mais limitado
- Windows, Linux, Mac OS X

DEPURAÇÃO (*DEBUGGING*)

Processo de procura e eliminação de erros ou defeitos em software e hardware

Passos principais

- 1 – Reproduzir problema
- 2 – Simplificação do teste que gera problema
- 3 – Análise do estado do programa

Técnicas de análise mais comuns

- ***Tracing*** – *prints* com indicações sobre valor de variáveis e fluxo do programa enquanto é executado
- ***Post-mortem*** – informação guardada após um *crash* programa, por exemplo: *memory dumps*
- ***Debugger*** – definição de pontos de paragem (*breakpoints*), execução passo-a-passo e inspeção directa de conteúdos de memória

DEBUGGERS

GDB (GNU Debugger)

- Suporta múltiplos sistemas, open-source
- Baseado em linha de comandos
- *Front-ends* incluem:
  - Eclipse, CodeBlocks, Xcode
- Windows, Linux, Mac OS X

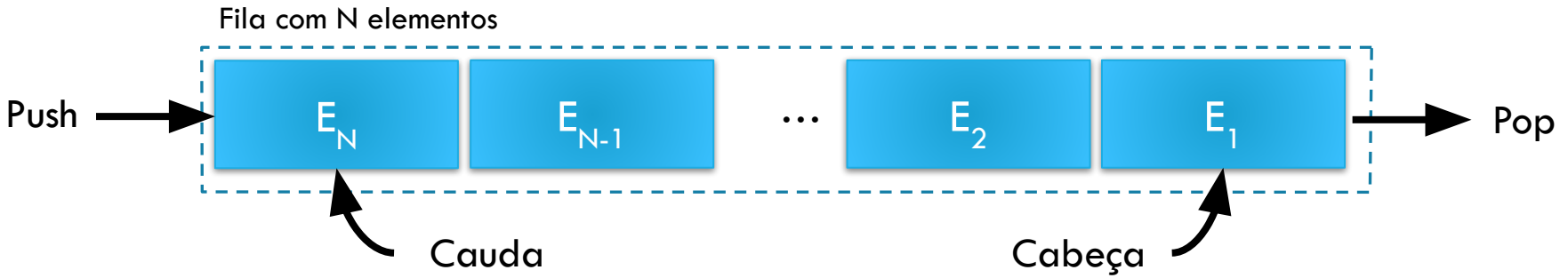
Visual Studio Debugger

- Integrado no Visual Studio
- Windows

FILA / QUEUE

É uma estrutura de dados linear, do tipo **FIFO (First-In-First-Out)**, em que a inserção de elementos se faz por uma extremidade, designada por **cauda**, e a remoção de elementos se faz pela extremidade oposta, designada por **cabeça da fila**.

Uma fila pode ser considerada como uma **restrição de lista**.



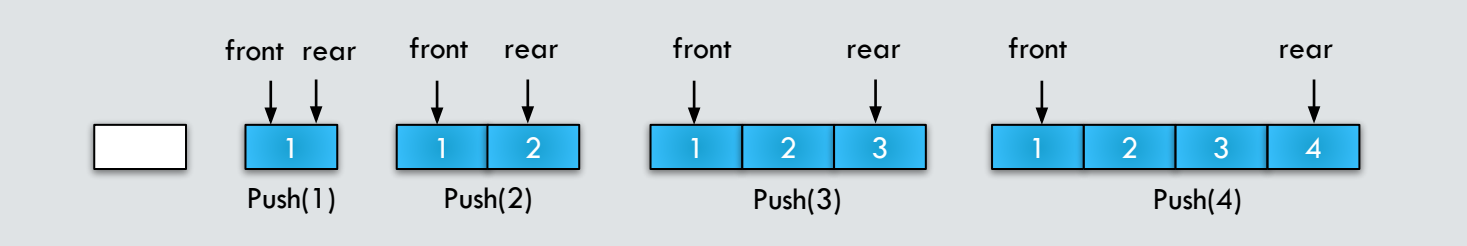
Exemplos:

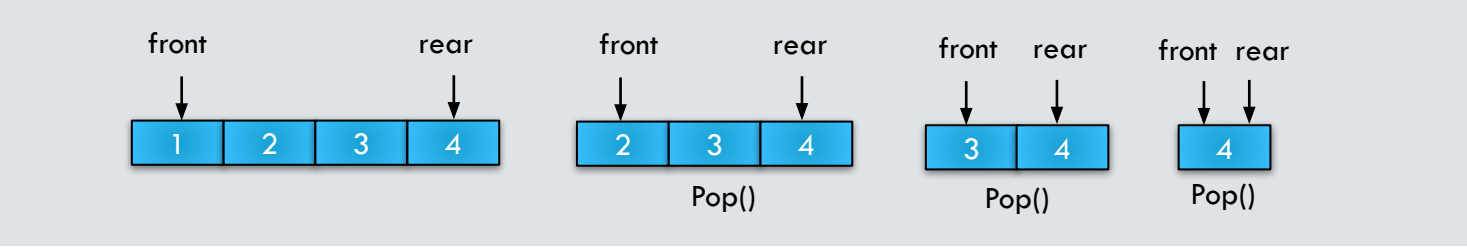
- fila de atendimento numa caixa de supermercado
- fila de impressão de uma impressora de rede

FILA / QUEUE

Operações comuns:

- criar uma fila vazia
- adicionar/remover um elemento a uma fila
- verificar qual o elemento da cabeça/cauda da fila (mais antigo/ recente)





IMPLEMENTAÇÃO DA FILA / QUEUE

A implementação da fila pode ser feita com:

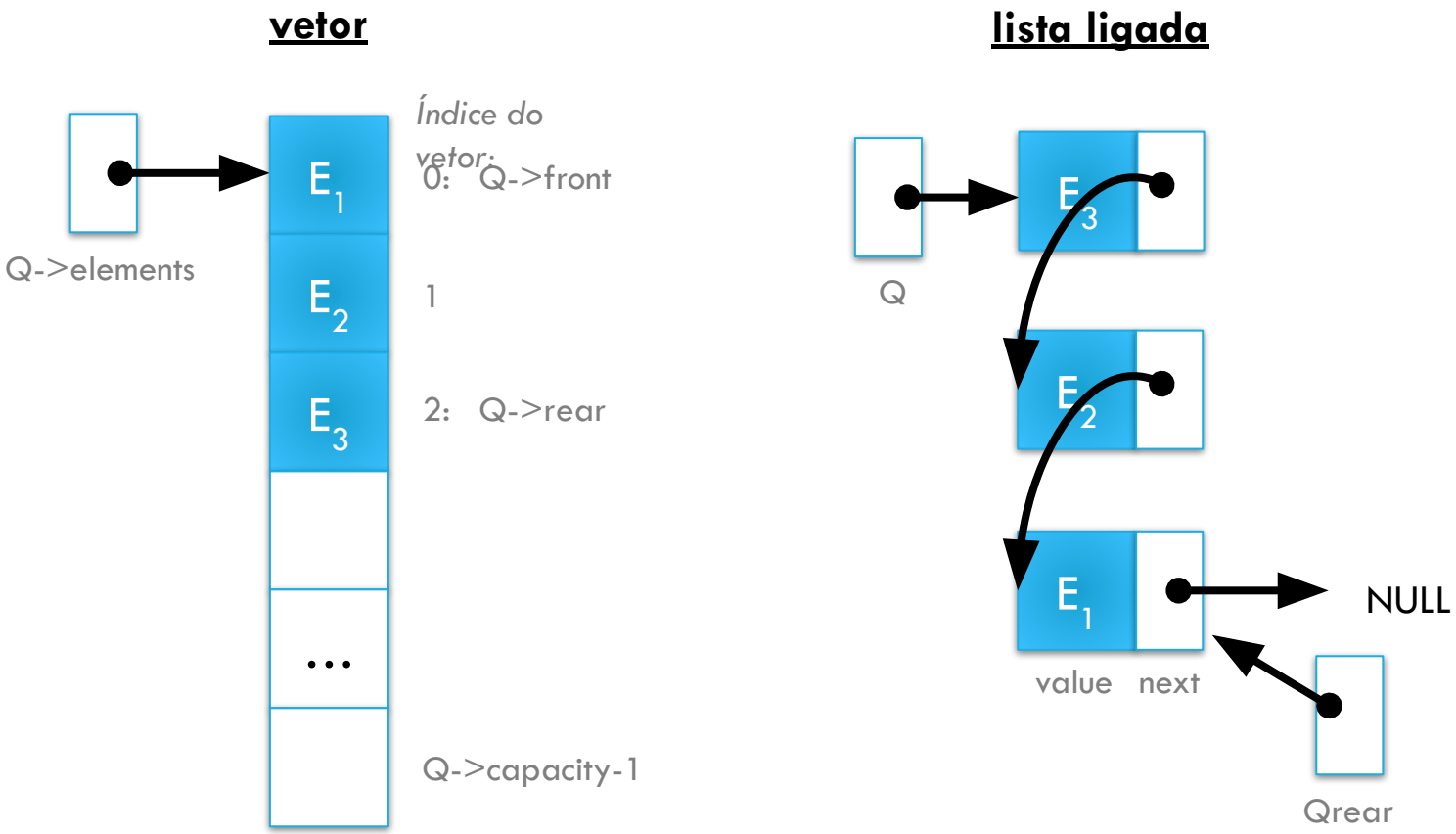
Um **vetor**

- Capacidade do vetor é pré-definida. É necessário saber o tamanho da fila e o índice dos elementos que estão na cauda e na cabeça da fila.
- **Push**: Se a fila não estiver cheia, insere novo elemento na cauda e incrementa o índice respectivo. O índice volta a zero se se exceder o tamanho do vector (fila circular).
- **Pop**: Se a fila não estiver vazia, remove o elemento da cabeça da fila e incrementa o índice respectivo. O índice volta a zero se se exceder o tamanho do vector (fila circular).

Uma **lista ligada**

- **Push**: Insere novo elemento no fim da lista.
- **Pop**: Remove o elemento do início da lista.

IMPLEMENTAÇÃO DA FILA / QUEUE



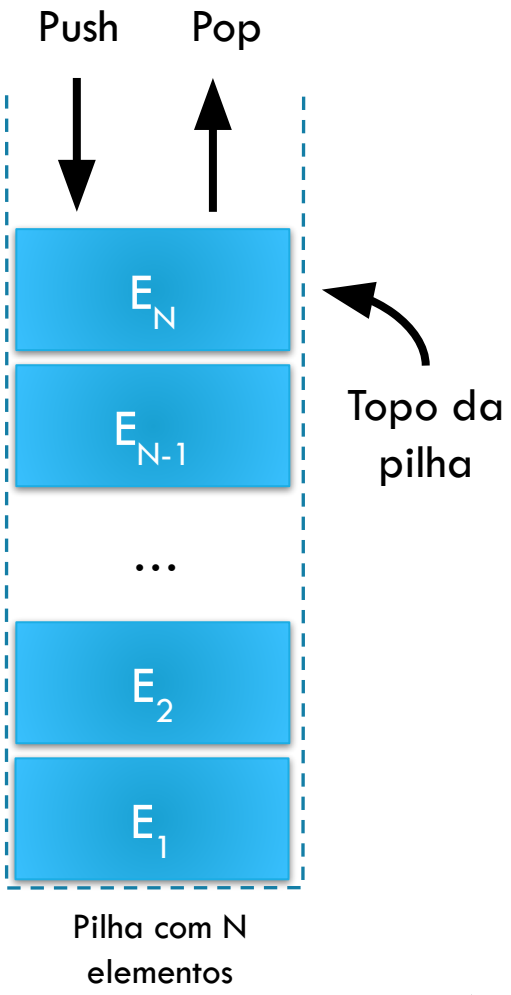
PILHA / STACK

É uma estrutura de dados linear, do tipo **LIFO (Last-In-First-Out)**, em que a inserção e a remoção de elementos se faz pela mesma extremidade, designada por **topo da pilha**.

Uma pilha pode ser considerada como uma **restrição de lista.** Porque é uma estrutura de dados mais simples que a lista, é possível obter implementações mais eficazes.

Exemplos :

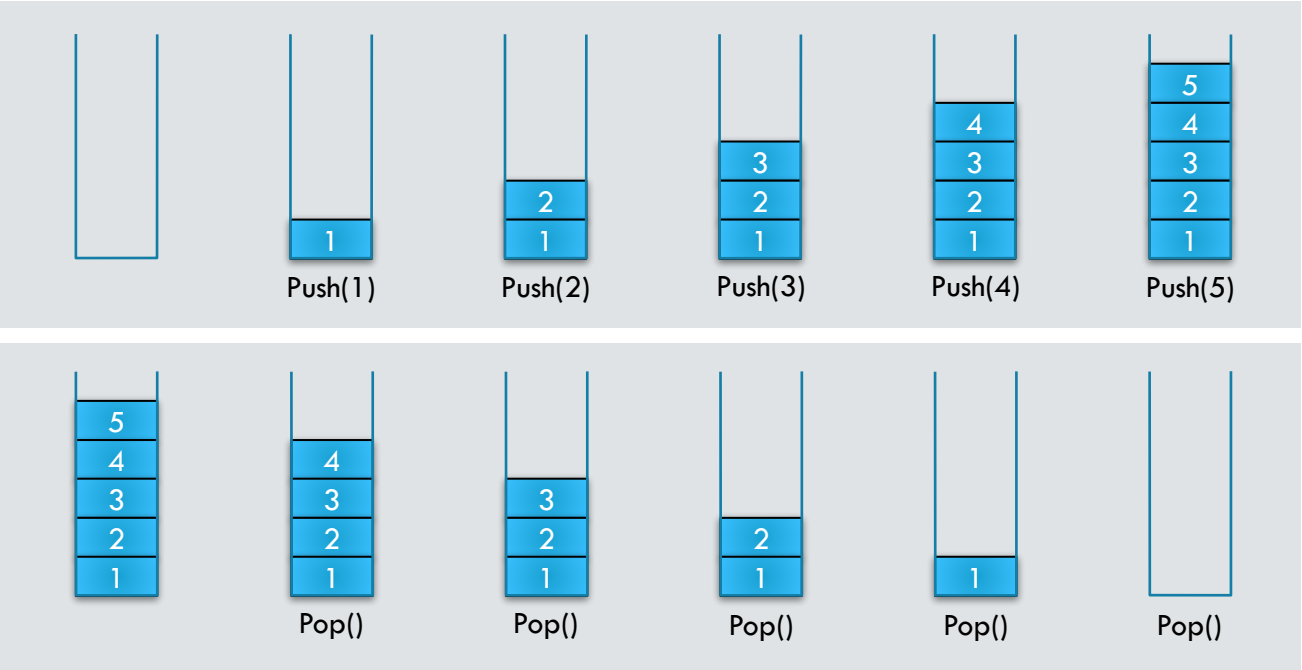
- tabuleiro de receção de serviço de um administrativo
- zona de memória auxiliar da execução de rotina de programação



PILHA / STACK

Operações comuns:

- criar uma pilha vazia
- adicionar/remover um elemento à pilha
- verificar qual o topo da pilha (último elemento adicionado)



IMPLEMENTAÇÃO DA PILHA / STACK

A implementação da pilha pode ser feita com:

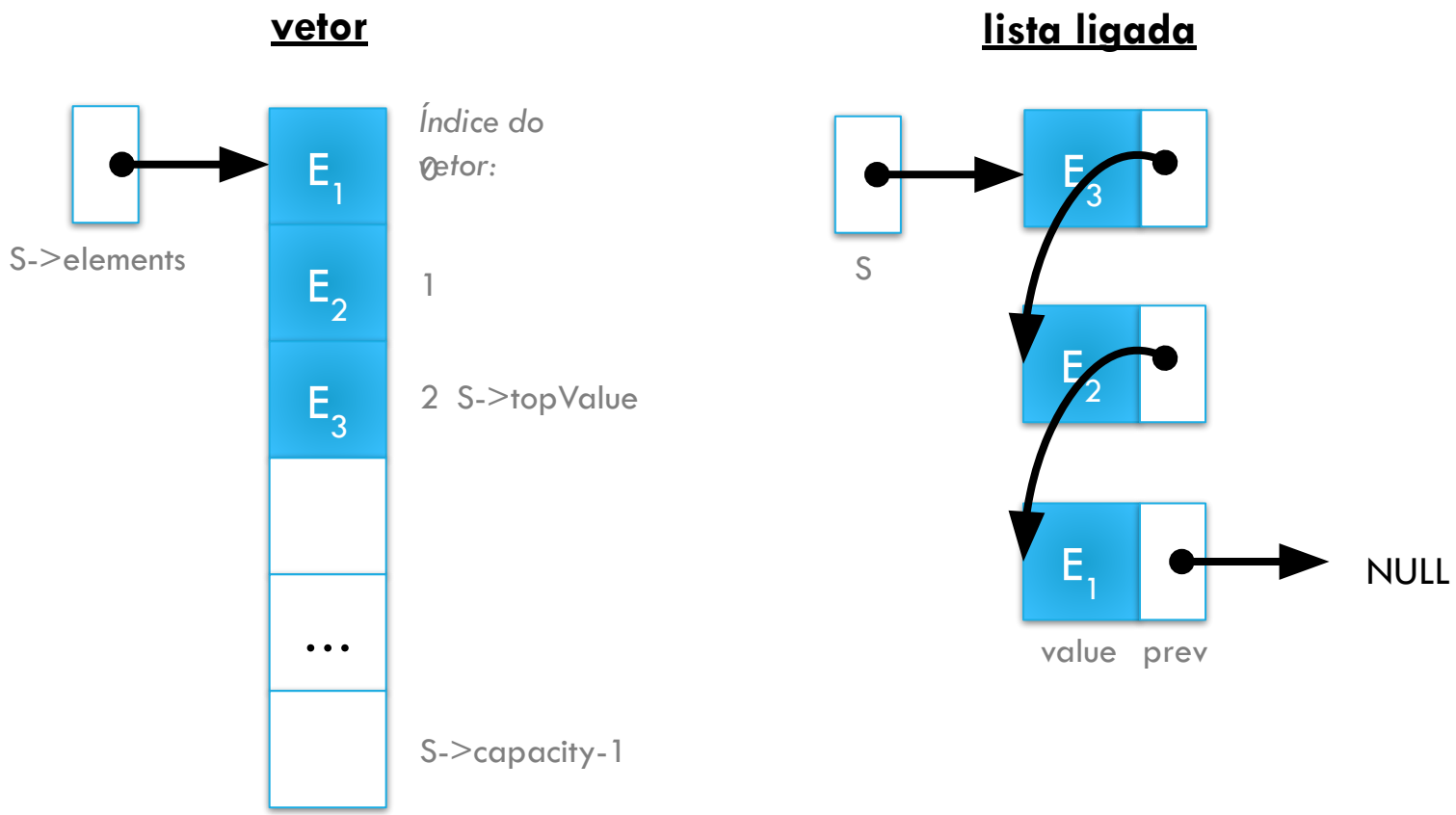
Um **vetor**

- Capacidade do vetor é pré-definida e é necessário saber o índice do elemento que está no topo da pilha.
- **Push**: Se o vetor não estiver cheio, insere novo elemento na primeira posição vazia e incrementa o índice do topo.
- **Pop**: Se o vetor não estiver vazio, remove o elemento que cujo índice corresponde ao topo da pilha e decrementa o índice.

Uma **lista ligada**

- **Push**: Insere novo elemento no início da lista.
- **Pop**: Remove o elemento do início da lista.

IMPLEMENTAÇÃO DA PILHA / STACK



PILHAS: EXEMPLO DE APLICAÇÃO

Notação RPN (Reverse Polish Notation)

- expressões matemáticas onde os operadores surgem a seguir aos operandos (notação *posfixa*)

  ex: **3 4 +**

- *vantagem*: não requer parênteses nem regras de precedência

Outras notações:

- **Notação infixa** (comum): operadores surgem entre os operandos ex: **3 + 4**

- **Notação prefixa**: operadores surgem antes dos operandos

  ex: **+ 3 4**

Exemplo:

Notação infixa: **2 * (4 + 5) / 3**

Notação RPN: **2 4 5 + * 3 /**

PILHAS: EXEMPLO DE APLICAÇÃO

Algoritmo para avaliação de expressões RPN

1. Processar sequencialmente os elementos da expressão. Para cada elemento:

    1.1 Se o elemento for um número (operando), colocá-lo na pilha (**push**)

    1.2 Se o elemento for um operador

    1.2.1 Retirar os dois elementos do topo da pilha (**pop**)

    1.2.2. Processar os elementos de acordo com o operador

    1.2.3. Colocar o resultado na pilha (**push**)

2. Retirar o (único) elemento da pilha. É o resultado.

EXERCÍCIO – FILAS/PILHAS

Qual seria o estado de uma fila após as seguintes instruções?

```
push(2); push(3); pop(); push(5); pop(); push(1);
push(4)
```

E de uma pilha?

- Como procederia para inverter os elementos de uma pilha?
- E de uma fila?