FICR
Faculdade Católica Imaculada Conceição

Estruturas de Dados

Mayrton Dias
mayrton.queiroz@p.ficr.edu.br

1. Pilhas
2. Implementação de pilha
3. Fila

Pilhas

Tipos Particulares de Listas

Pilha

- Inserções e remoções ocorrem somente em um extremo da lista
- **LIFO: Last In First Out**

Fila

- Inserções são realizadas em um extremo e remoções em outro extremo da lista.
- **FIFO: First In First Out**

Definição de Pilha

Pilha

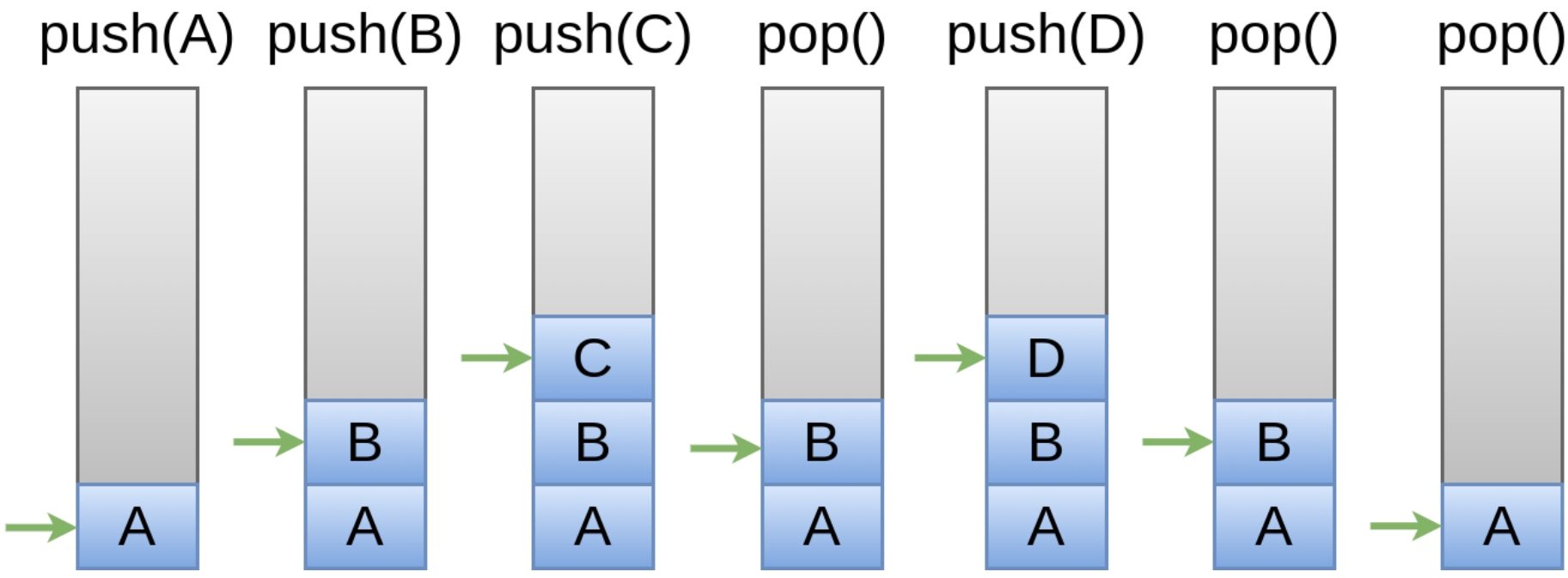
- **Ideia fundamental**: Acesso a elementos da pilha é sempre feito através do seu **topo**.
- Quando um **elemento novo** é **introduzido** na pilha, ele passa a ser o elemento do **topo**.
- O único elemento que pode ser **removido da pilha** é o do **topo**.
- Exemplo: Metáfora da pilha de pratos

Definição de Pilha

FICR
Faculdade Católica Imaculada Conceição

Pilha

- Inserção e remoção acontecem no topo da pilha.



Aplicações de Pilha

Estrutura de dados mais utilizada em programação

- Implementada diretamente pelo Hardware da maioria das máquinas modernas.
- Implementação de compiladores
  - Pilha de execução de funções
  - Avaliação de expressões
- Navegadores web
  - Usam pilhas para armazenar os endereços mais recentemente visitados
- Mecanismo de reversão de operações (“undo”) dos editores de texto
  - Armazena as alterações em uma pilha

Implementação de pilha

Pilha

Faculdade Católica Imaculada Conceição

Estrutura da pilha

```
typedef struct pilha Pilha;
typedef struct pilhaNo PilhaNo;

struct pilha{
    PilhaNo *topo;
};

struct pilhaNo{
    int valor;
    PilhaNo *prox;
};
```

Pilha

Criar pilha

```
Pilha* criarPilha(){
    Pilha *nova = (Pilha*)malloc(sizeof(Pilha));

    if(nova == NULL){
        printf("Sem espaco\n");
        return NULL;
    }

    nova->topo = NULL;
    return nova;
}
```

Pilha

Inserir o elemento no topo da pilha (push)

```
int push(Pilha *pilha, int valor){
    PilhaNo *nova = (PilhaNo*) malloc(sizeof(PilhaNo));

    if(nova == NULL){
        printf("Sem espaco\n");
        return 0;
    }

    nova->valor = valor;
    nova->prox = pilha->topo;
    pilha->topo = nova;
    return 1;
}
```

Pilha

Remover o elemento do topo da pilha (pop)

```
PilhaNo* pop(Pilha *pilha){
    PilhaNo *aux;

    if(pilha == NULL){
        printf("A pilha nao foi criada\n");
        return NULL;
    }
    if(pilha->topo == NULL){
        printf("A pilha esta vazia\n");
        return NULL;
    }
    aux = pilha->topo;
    pilha->topo = pilha->topo->prox;
    return aux;
}
```

Pilha

Visualizar o topo da pilha

```
int verTopo(Pilha *pilha){
    PilhaNo *aux;
    if(pilha == NULL){
        printf("A Pilha nao foi criada\n");
        return -1;
    }
    if(pilha->topo == NULL){
        printf("A pilha esta vazia\n");
        return -1;
    }
    return pilha->topo->valor;
}
```

Pilha

Verificar se a pilha é vazia

```
int ehVazia(Pilha* pilha){
    if(pilha == NULL){
        printf("A pilha nao foi criada\n");
        return 1;
    }
    if(pilha->topo == NULL){
        printf("A pilha esta vazia\n");
        return 1;
    }
    return 0;
}
```

Pilha

Excluir a pilha

```
Pilha* excluirPilha(Pilha* pilha){
    PilhaNo *aux;

    while(pilha->topo != NULL){
        aux = pilha->topo;
        pilha->topo = pilha->topo->prox;
        free(aux);
    }

    free(pilha);
    return NULL;
}
```

Fila

Fila com vetores

Definição da fila

```
typedef struct{
    int inicio;
    int fim;
    int *elementos;
} Fila;
```

Fila com vetores

Criar a fila

```
Fila* criarFila(){
    Fila *nova = (Fila*)malloc(sizeof(Fila));

    if(nova == NULL){
        printf("Nao tem espaco\n");
        return NULL;
    }

    nova->inicio = 0;
    nova->fim = 0;
    nova->elementos = (int*)malloc(TAM*sizeof(int));

    if(nova->elementos == NULL){
        printf("Nao tem espaco\n");
        free(nova);
        return NULL;
    }
    return nova;
}
```

Fila com vetores

Enfileirar um elemento na fila (enqueue)

```c
int enqueue(Fila *fila, int valor){

    if(fila == NULL){
        printf("A Fila nao foi criada\n");
        return 0;
    }

    if((fila->fim + 1) % TAM != fila->inicio){
        fila->elementos[fila->fim] = valor;
        fila->fim = (fila->fim + 1 ) % TAM;
    }else{
        printf("Espaco esgotado\n");
        return 0;
    }

    return 1;
}
```

Fila com vetores

Desenfileirar um elemento na fila (dequeue)

```
int dequeue(Fila *fila){
    int valor;

    if(fila == NULL){
        printf("A fila nao foi criada\n");
        return -1;
    }

    if(((fila->inicio + 1) % TAM) != fila->fim){
        valor = fila->elementos[fila->inicio];
        fila->inicio =  (fila->inicio + 1) % TAM;
        return valor;
    }
    return -1;
}
```

Fila com apontadores

Definição da fila

```
typedef struct fila Fila;
typedef struct filaNo FilaNo;

struct fila{
    FilaNo *inicio;
    FilaNo *fim;
};

struct filaNo{
    int valor;
    FilaNo *prox;
};
```

Fila com apontadores

Criar a fila

```
Fila* criarFila(){
    Fila *nova = (Fila*)malloc(sizeof(Fila));

    if(nova == NULL){
        printf("Sem espaco\n");
        return NULL;
    }

    nova->inicio = NULL;
    nova->fim = NULL;

    return nova;
}
```

Fila com apontadores

Enfileirar um elemento na fila (enqueue)

```
int enqueue(Fila *fila, int valor){
    FilaNo *nova = (FilaNo*) malloc(sizeof(FilaNo));
    if(nova == NULL){
        printf("Sem espaco\n");
        return 0;
    }

    nova->valor = valor;
    nova->prox = NULL;

    if(fila->inicio == NULL){
        fila->inicio = nova;
        fila->fim = nova;
        return 1;
    }
    fila->fim->prox = nova;
    fila->fim = nova;
    return 1;
}
```

Fila com apontadores

Desenfileirar um elemento na fila (dequeue)

```
FilaNo* dequeue(Fila *fila){
    FilaNo *aux;

    if(fila == NULL){
        printf("A fila nao foi criada\n");
        return NULL;
    }
    if(fila->inicio == NULL){
        printf("A fila esta vazia\n");
        return NULL;
    }
    if(fila->inicio == fila->fim){
        aux = fila->inicio;
        fila->inicio = NULL;
        fila->fim = NULL;
        return aux;
    }
    aux = fila->inicio;
    fila->inicio = fila->inicio->prox;
    return aux;
}
```

Fila com apontadores

Excluir a fila

```
Fila* excluirFila(Fila* fila){
    FilaNo *aux;

    while(fila->inicio != NULL){
        aux = fila->inicio;
        fila->inicio = fila->inicio->prox;
        free(aux);
    }

    free(fila);
    return NULL;
}
```

FICR
Faculdade Católica Imaculada Conceição

Estruturas de Dados

Mayrton Dias
mayrton.queiroz@p.ficr.edu.br