AULA 13
ESTRUTURA DE DADOS

Duas pilhas - implementação estática

Norton T. Roman & Luciano A. Digiampietri

Pilha

Pilha é uma estrutura linear na qual:

- As inserções ocorrem no topo da pilha;
- As exclusões ocorrem no topo da pilha.
- Utiliza a mesma lógica de uma pilha de papéis.

Duas pilhas

Imagine que estamos organizando um conjunto de provas e temos que separá-las em dois subconjuntos

Duas pilhas

Imagine que estamos organizando um conjunto de provas e temos que separá-las em dois subconjuntos

Poderíamos criar uma pilha para cada conjunto.

Duas pilhas

Imagine que estamos organizando um conjunto de provas e temos que separá-las em dois subconjuntos

Poderíamos criar uma pilha para cada conjunto.

Porém, isto poderia ser um desperdício de recursos (no caso da implementação estática).

Duas pilhas - implementação estática

Utilizaremos um único arranjo de elementos de tamanho predefinido;

Duas pilhas - implementação estática

Utilizaremos um único arranjo de elementos de tamanho predefinido;

Colocaremos “uma pilha” em cada extremidade do arranjo;

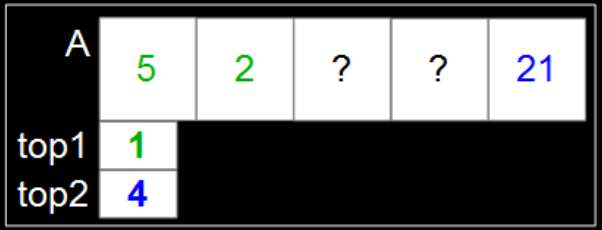
Duas pilhas - implementação estática

Utilizaremos um **único arranjo** de elementos de tamanho predefinido;

Colocaremos “uma pilha” **em cada extremidade do arranjo**;

Controlaremos a posição do elemento que está no **topo** de cada uma das “pilhas”.

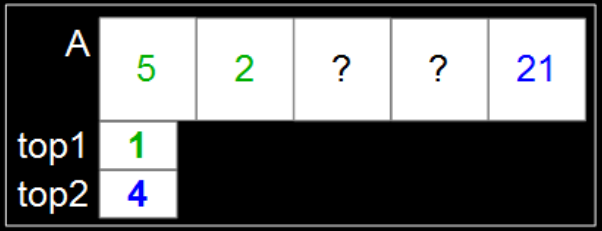
Ideia



Temos um arranjo de tamanho predefinido

Temos um campo para indicar a posição do elemento que está no topo

Ideia

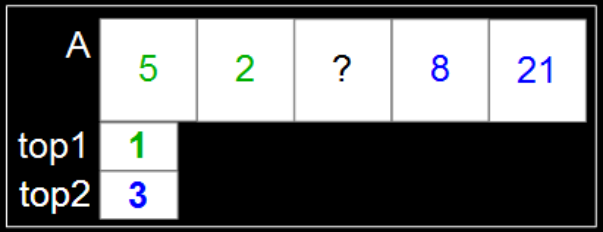


Temos um arranjo de tamanho predefinido

Temos um campo para indicar a posição do elemento que está no topo

Como inserimos o elemento 8 na pilha 2?

Ideia

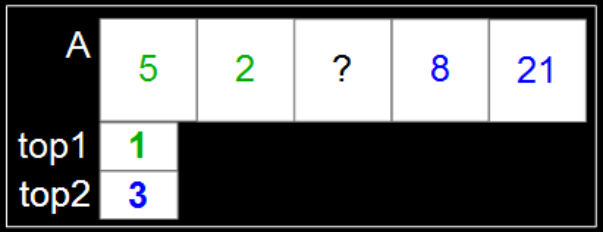


Temos um arranjo de tamanho predefinido

Temos um campo para indicar a posição do elemento que está no topo

Como inserimos o elemento 8 na pilha 2?

Ideia



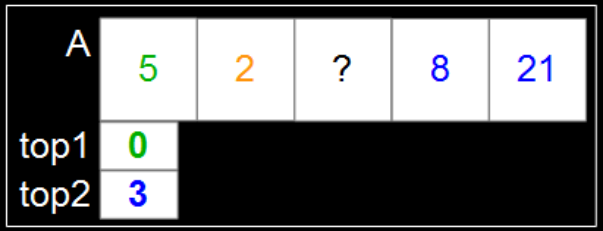
Temos um arranjo de tamanho predefinido

Temos um campo para indicar a posição do elemento que está no topo

Como inserimos o elemento 8 na pilha 2?

Como excluímos um elemento da pilha 1?

Ideia



Temos um arranjo de tamanho predefinido

Temos um campo para indicar a posição do elemento que está no topo

Como inserimos o elemento 8 na pilha 2?

Como excluímos um elemento da pilha 1?

Modelagem

```
#include <stdio.h>

#define MAX 50

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
} REGISTRO;
```

```
typedef struct {
  REGISTRO A[MAX];
  int topo1;
  int topo2;
} PILHADUPLA;
```

Modelagem

```
#include <stdio.h>

#define MAX 50

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
} REGISTRO;
```

```
typedef struct {
  REGISTRO A[MAX];
  int topo1;
  int topo2;
} PILHADUPLA;
```

Modelagem

```
#include <stdio.h>

#define MAX 50

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
} REGISTRO;
```

```
typedef struct {
  REGISTRO A[MAX];
  int topo1;
  int topo2;
} PILHADUPLA;
```

Funções de gerenciamento

Implementaremos funções para:

- **Inicializar a estrutura**
- **Retornar a quantidade de elementos válidos**
- **Exibir os elementos da estrutura**
- **Inserir elementos na estrutura (*push*)**
- **Excluir elementos da estrutura (*pop*)**
- **Reinicializar a estrutura**

Inicialização

Para inicializar uma pilha já criada pelo usuário, precisamos apenas acertar o valor dos campos *topo1* e *topo2*.

Inicialização

Para inicializar uma pilha já criada pelo usuário, precisamos apenas acertar o valor dos campos *topo1* e *topo2*.

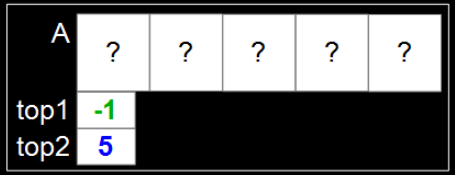
Já que o topo indicará a posição no arranjo do elemento que está no topo das pilhas e as pilhas estão vazias, iniciaremos com os valores topo1=-1 e topo2=MAX.

Inicialização

```
void inicializarPilhaDupla(PILHADUPLA* p) {
  p->topo1 = -1;
  p->topo2 = MAX;
}
```

Inicialização

```
void inicializarPilhaDupla(PILHADUPLA* p) {
  p->topo1 = -1;
  p->topo2 = MAX;
}
```



Retornar número de elementos

Retornar número de elementos

Podemos utilizar os valores dos campos *topo1* e *topo2* para calcular o número de elementos:

Retornar número de elementos

Podemos utilizar os valores dos campos *topo1* e *topo2* para calcular o número de elementos:

Na “pilha 1” há *topo1 + 1* elementos válidos;

Retornar número de elementos

Podemos utilizar os valores dos campos *topo1* e *topo2* para calcular o número de elementos:

Na “pilha 1” há *topo1 + 1* elementos válidos;

Na “pilha 2” há *MAX - topo2* elementos válidos;

Retornar número de elementos

```
int tamanhoPilhaDupla(PILHADUPLA* p) {
  return p->topo1 + 1 + MAX - p->topo2;
}
```

Exibição/Impressão

Para exibir os elementos da estrutura precisaremos iterar pelos elementos válidos e, por exemplo, imprimir suas chaves.

Um dos parâmetros da função de impressão irá nos dizer qual das pilhas será exibida.

Exibição/Impressão

```
bool exibirUmaPilha(PILHADUPLA* p, int pilha) {




}
```

Exibição/Impressão

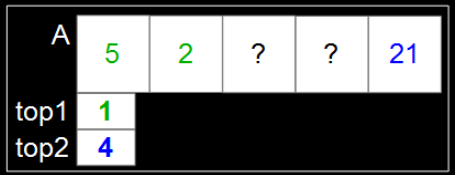
```
bool exibirUmaPilha(PILHADUPLA* p, int pilha) {
   if (pilha < 1 || pilha > 2) return false;




   return true;
}
```

Exibição/Impressão

```
bool exibirUmaPilha(PILHADUPLA* p, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   printf("Pilha %i: \" ", pilha);



   printf("\"\n");
   return true;
}
```

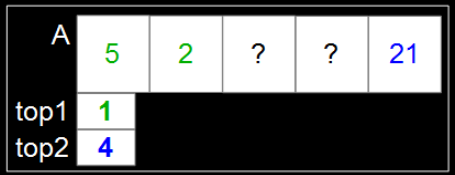
Exibição/Impressão

```
bool exibirUmaPilha(PILHADUPLA* p, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   printf("Pilha %i: \" ", pilha);
   int i;
   if (pilha == 1) for (i=p->topo1;i>=0;i--) printf("%i ", p->A[i].chave);

   printf("\"\n");
   return true;
}
```



Exibição/Impressão

```
bool exibirUmaPilha(PILHADUPLA* p, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   printf("Pilha %i: \" ", pilha);
   int i;
   if (pilha == 1) for (i=p->topo1;i>=0;i--) printf("%i ", p->A[i].chave);
   else for (i=p->topo2;i<MAX;i++) printf("%i ", p->A[i].chave);
   printf("\"\n");
   return true;
}
```



Inserção de um elemento (*push*)

O usuário passa como parâmetro um registro a ser inserido na pilha e diz em qual pilha deseja inserir

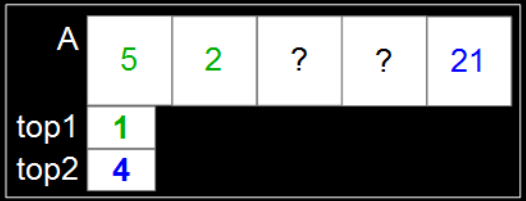
Inserção de um elemento (*push*)

O usuário passa como parâmetro um registro a ser inserido na pilha e diz em qual pilha deseja inserir

- Se a estrutura não estiver cheia, o elemento será inserido no topo da respectiva pilha, ou melhor, "acima" do elemento que está no topo da pilha.

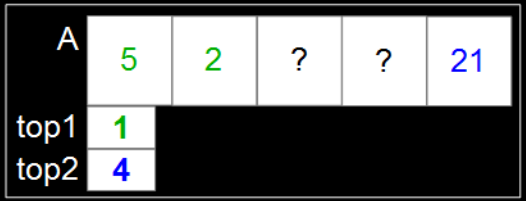
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHADUPLA* p, REGISTRO reg, int pilha) {



}
```



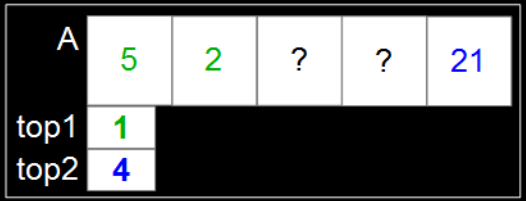
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHADUPLA* p, REGISTRO reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;




}
```



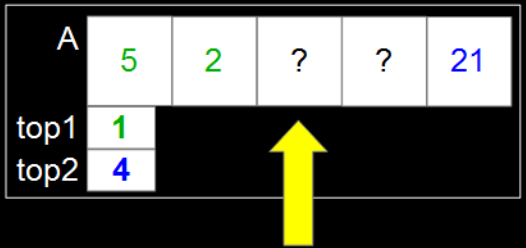
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHADUPLA* p, REGISTRO reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (p->topo1 + 1 == p->topo2) return false;




}
```



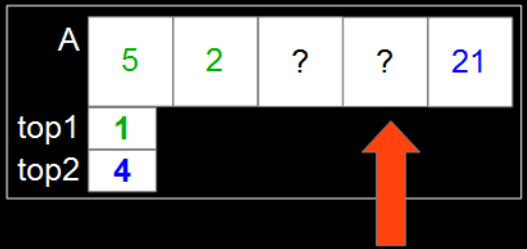
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHADUPLA* p, REGISTRO reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (p->topo1 + 1 == p->topo2) return false;
   if (pilha == 1) {
      p->topo1 = p->topo1+1;
      p->A[p->topo1] = reg;
   }




}
```



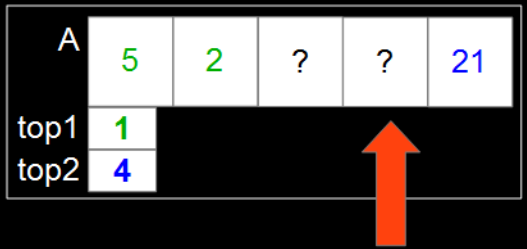
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHADUPLA* p, REGISTRO reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (p->topo1 + 1 == p->topo2) return false;
   if (pilha == 1) {
      p->topo1 = p->topo1+1;
      p->A[p->topo1] = reg;
   } else{
      p->topo2 = p->topo2-1;
      p->A[p->topo2] = reg;
   }

}
```



Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHADUPLA* p, REGISTRO reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (p->topo1 + 1 == p->topo2) return false;
   if (pilha == 1) {
      p->topo1 = p->topo1+1;
      p->A[p->topo1] = reg;
   } else{
      p->topo2 = p->topo2-1;
      p->A[p->topo2] = reg;
   }
   return true;
}
```



Exclusão de um elemento (*pop*)

O usuário solicita a exclusão do elemento do topo de uma das pilhas:

Exclusão de um elemento (*pop*)

O usuário solicita a exclusão do elemento do topo de uma das pilhas:

Se a pilha não estiver vazia, além de excluir esse elemento da pilha iremos copiá-lo para um local indicado pelo usuário.

Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHADUPLA* p, REGISTRO* reg, int pilha) {









}
```

Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHADUPLA* p, REGISTRO* reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;









}
```

Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHADUPLA* p, REGISTRO* reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (pilha == 1) {
     if (p->topo1 == -1) return false;
     *reg = p->A[p->topo1];
     p->topo1 = p->topo1 - 1;
   }




}
```

Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHADUPLA* p, REGISTRO* reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (pilha == 1) {
     if (p->topo1 == -1) return false;
     *reg = p->A[p->topo1];
     p->topo1 = p->topo1 - 1;
   }else{
     if (p->topo2 == MAX) return false;
     *reg = p->A[p->topo2];
     p->topo2 = p->topo2 + 1;
   }

}
```

Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHADUPLA* p, REGISTRO* reg, int pilha) {
   if (pilha < 1 || pilha > 2) return false;
   if (pilha == 1) {
     if (p->topo1 == -1) return false;
     *reg = p->A[p->topo1];
     p->topo1 = p->topo1 - 1;
   }else{
     if (p->topo2 == MAX) return false;
     *reg = p->A[p->topo2];
     p->topo2 = p->topo2 + 1;
   }
   return true;
}
```

Reinicialização da pilha

Reinicialização da pilha

Para esta estrutura, para reinicializar as pilhas basta acertar os valores de *topo1* e *topo2* ou chamarmos a função *inicializarPilhaDupla*.

Reinicialização da pilha

```
void reinicializarPilha(PILHADUPLA* p) {
  p->topo1 = -1;
  p->topo2 = MAX;
}

void reinicializarPilha2(PILHADUPLA* p) {
  inicializarPilhaDupla(p);
}
```

AULA 13
ESTRUTURA DE DADOS

Duas pilhas - implementação estática

Norton T. Roman & Luciano A. Digiampietri