AULA 08
ESTRUTURA DE DADOS

Pilha - implementação estática

Norton T. Roman & Luciano A. Digiampietri

Pilha

Pilha é uma estrutura linear na qual:

Pilha

Pilha é uma estrutura linear na qual:

- As inserções ocorrem no topo da pilha;

Pilha

Pilha é uma estrutura linear na qual:

- **As inserções ocorrem no topo da pilha;**
- **As exclusões ocorrem no topo da pilha.**

Pilha

Pilha é uma estrutura linear na qual:

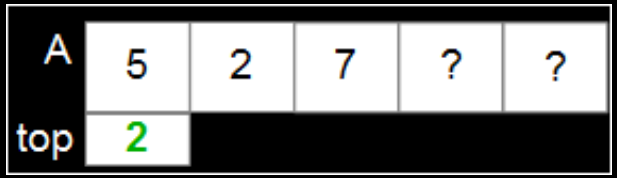
- As inserções ocorrem no topo da pilha;
- As exclusões ocorrem no topo da pilha.
- Utiliza a mesma lógica de uma pilha de papéis.

Pilha - implementação estática

Utilizaremos um arranjo de elementos de tamanho predefinido;

Controlaremos a posição do elemento que está no topo da pilha.

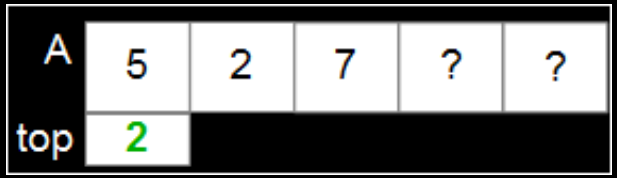
Ideia



Temos um arranjo de tamanho predefinido

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

Ideia

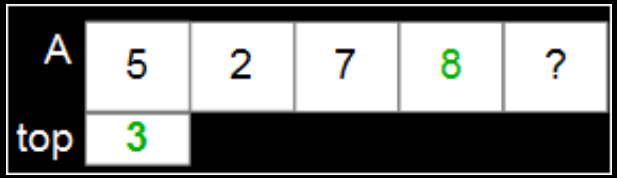


Temos um arranjo de tamanho predefinido

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

Como inserimos o elemento 8?

Ideia

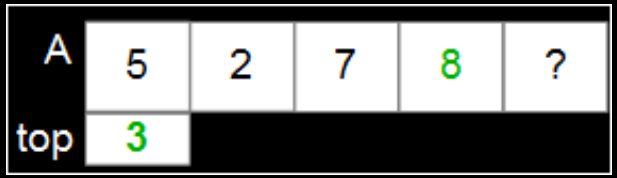


Temos um arranjo de tamanho predefinido

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

Como inserimos o elemento 8?

Ideia



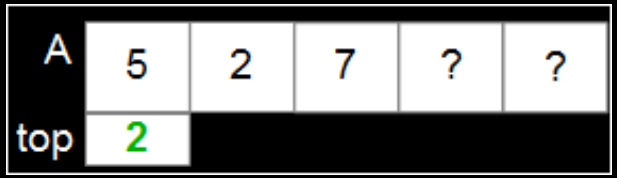
Temos um arranjo de tamanho predefinido

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

Como inserimos o elemento 8?

Como excluímos um elemento?

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?

Como excluímos um elemento?

Modelagem

```
#include <stdio.h>
#define MAX 50

#define true 1
#define false 0
typedef int bool;

typedef int TIPOCHAVE;
```

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

typedef struct {
  REGISTRO A[MAX];
  int topo;
} PILHA;
```

Modelagem

```
#include <stdio.h>
#define MAX 50

#define true 1
#define false 0
typedef int bool;

typedef int TIPOCHAVE;
```

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

typedef struct {
  REGISTRO A[MAX];
  int topo;
} PILHA;
```

Modelagem

```
#include <stdio.h>
#define MAX 50

#define true 1
#define false 0
typedef int bool;

typedef int TIPOCHAVE;
```

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

typedef struct {
  REGISTRO A[MAX];
  int topo;
} PILHA;
```

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 do campo *topo*.

Inicialização

Para inicializar uma pilha já criada pelo usuário, precisamos apenas acertar o valor do campo ***topo***.

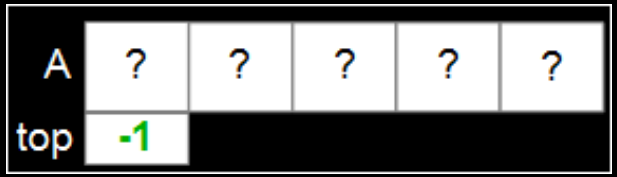
Já que o topo indicará a posição no arranjo do elemento que está no topo da pilha e a pilha está vazia, iniciaremos esse campo com valor -1.

Inicialização

```
void inicializarPilha(PILHA* p) {
  p->topo = -1;
}
```

Inicialização

```
void inicializarPilha(PILHA* p) {
  p->topo = -1;
}
```



Retornar número de elementos

Retornar número de elementos

Já que o campo *topo* contém a posição no arranjo do elemento no topo da pilha, o número de elementos é igual a: *topo* + 1

Retornar número de elementos

Já que o campo *topo* contém a posição no arranjo do elemento no topo da pilha, o número de elementos é igual a: ***topo* + 1**

Notem que para a **pilha vazia** isto também funciona.

Retornar número de elementos

```
int tamanhoPilha(PILHA* p) {
  return p->topo + 1;
}
```

Exibição/Impressão

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

Exibição/Impressão

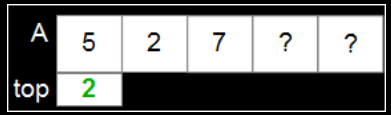
```
void exibirPilha(PILHA* p) {
  printf("Pilha: \" ");
  int i;
  for (i=p->topo;i>=0;i--) {
    printf("%i ", p->A[i].chave);
  }
  printf("\"\n");
}
```

Exibição/Impressão

```
void exibirPilha(PILHA* p) {
  printf("Pilha: \" ");
  int i;
  for (i=p->topo;i>=0;i--) {
    printf("%i ", p->A[i].chave);
  }
  printf("\"\n");
}
```

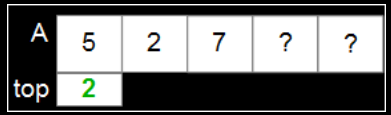
Exibição/Impressão

```
void exibirPilha(PILHA* p) {
  printf("Pilha: \" ");
  int i;
  for (i=p->topo;i>=0;i--) {
    printf("%i ", p->A[i].chave);
  }
  printf("\"\n");
}
```



Exibição/Impressão

```
void exibirPilha(PILHA* p) {
  printf("Pilha: \" ");
  int i;
  for (i=p->topo;i>=0;i--) {
    printf("%i ", p->A[i].chave);
  }
  printf("\"\n");
}
```



Saída:

```
$ Pilha: " 7 2 5 "
```

Inserção de um elemento (*push*)

O usuário passa como parâmetro um registro a ser inserido na pilha

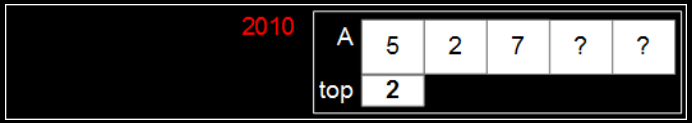
Inserção de um elemento (*push*)

O usuário passa como parâmetro um registro a ser inserido na pilha

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

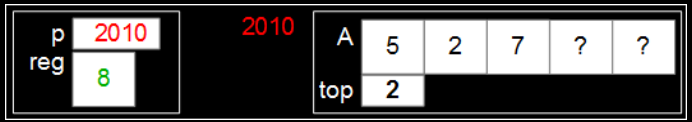
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHA* p, REGISTRO reg) {
     if (p->topo >= MAX-1) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
}
```



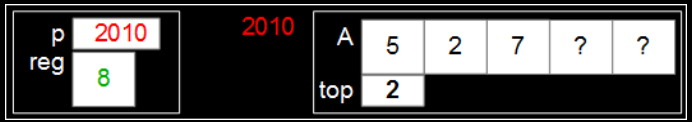
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHA* p, REGISTRO reg) {
     if (p->topo >= MAX-1) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
}
```



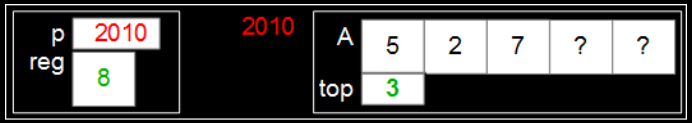
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHA* p, REGISTRO reg) {
     if (p->topo >= MAX-1) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
}
```



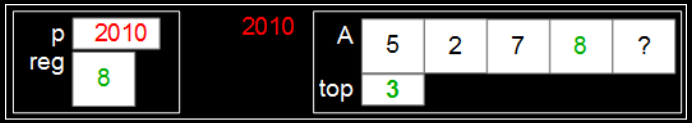
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHA* p, REGISTRO reg) {
     if (p->topo >= MAX-1) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
}
```



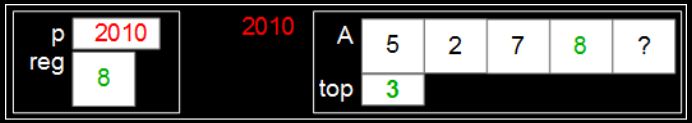
Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHA* p, REGISTRO reg) {
     if (p->topo >= MAX-1) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
}
```



Inserção de um elemento (*push*)

```
bool inserirElementoPilha(PILHA* p, REGISTRO reg) {
     if (p->topo >= MAX-1) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
}
```



Exclusão de um elemento (*pop*)

O usuário solicita a exclusão do elemento do topo da pilha:

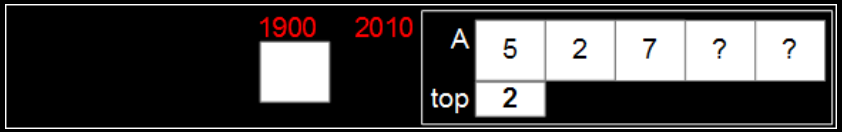
Exclusão de um elemento (*pop*)

O usuário solicita a exclusão do elemento do topo da pilha:

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

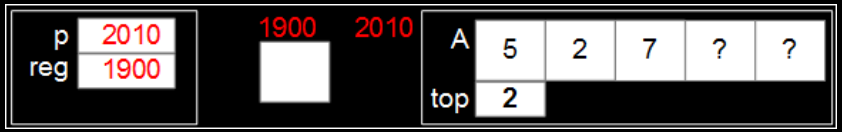
Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



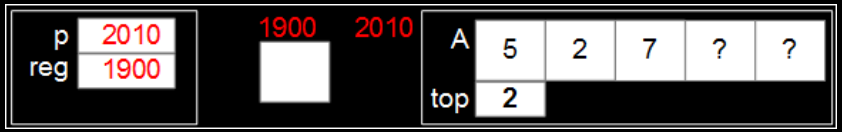
Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



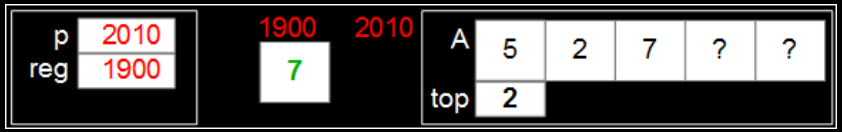
Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



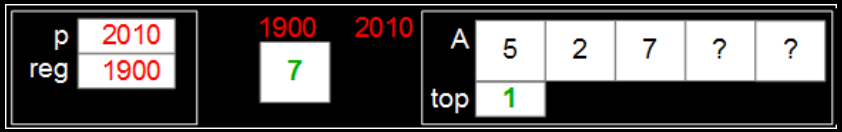
Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



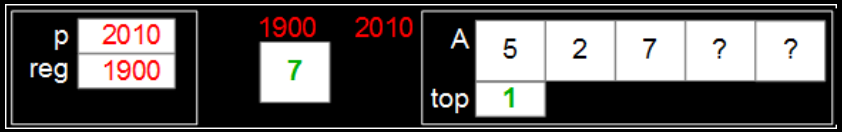
Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



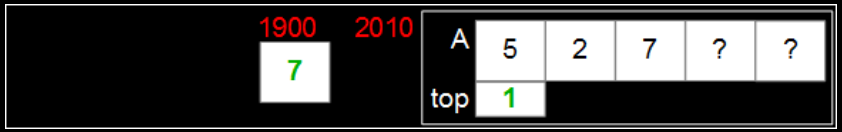
Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



Exclusão de um elemento (*pop*)

```
bool excluirElementoPilha(PILHA* p, REGISTRO* reg) {
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
}
```



Reinicialização da pilha

Reinicialização da pilha

Para esta estrutura, para reinicializar a pilha basta colocar -1 no campo *topo*

Reinicialização da pilha

```
void reinicializarPilha(PILHA* p) {
  p->topo = -1;
}
```

AULA 08
ESTRUTURA DE DADOS

Pilha - implementação estática

Norton T. Roman & Luciano A. Digiampietri