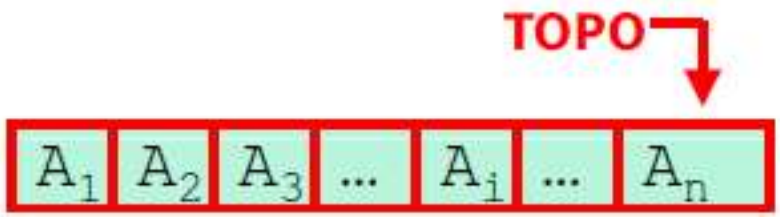
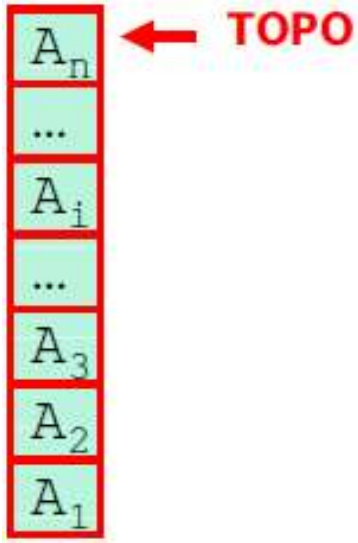
Pilha (Stack)

- Uma pilha é uma coleção de objetos que possuem uma relação de ordem entre si, ou seja, existe o primeiro; segundo; etc.
- Na pilha os objetos obedecem ao princípio "Último a entrar - Primeiro a sair" - LIFO (Last-in - First Out)

$$a_1, a_2, a_3, \ldots, a_n \qquad n \geq 0$$

- Na pilha abaixo, o primeiro objeto a entrar foi o objeto $\mathbf{a_1}$ e o último o objeto an, porém somente é possível a remoção do objeto $\mathbf{a_n}$.
- Para remover o objeto $\mathbf{a_1}$ é necessário antes remover todos os outros objetos e finalmente será possível sua remoção
- O objeto $\mathbf{a_n}$ é conhecido como o **topo** da pilha



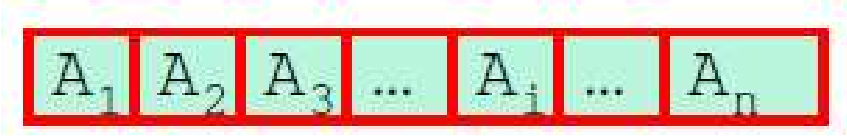


Pilha (Stack)

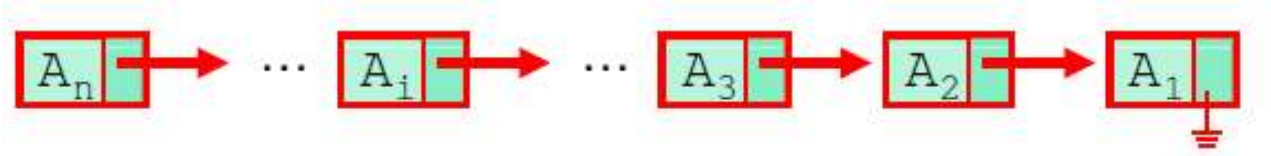
- Uma pilha pode ter 0 ou mais elementos
- Uma PILHA VAZIA não possui elementos.
- Uma pilha, pode ser implementada como uma lista onde todas as inserções e remoções são feitas somente em uma de suas extremidades.
- O topo da pilha pode ser o último ou o primeiro objeto da lista.

Pilha (Stack)
Estática x Dinâmica

- Assim como a lista a pilha pode ser representada de forma estática ou dinâmica
- Pilha Estática
  - Neste caso os objetos da pilha estarão contidos em um Vetor

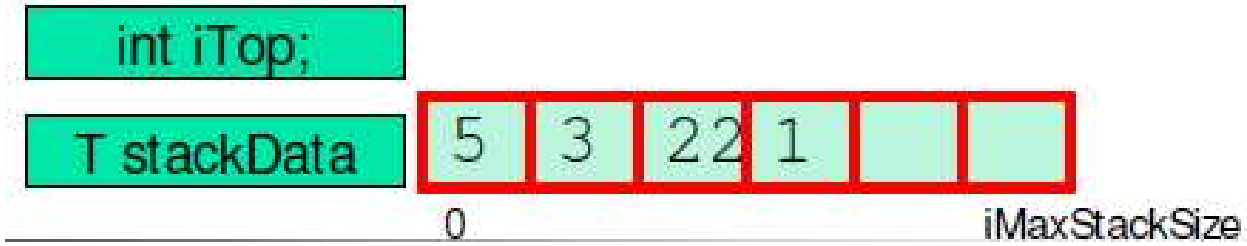


- Pilha Dinâmica
  - Neste caso os objetos da pilha são alocados de forma dinâmica
  - Um elemento qualquer deve conhecer a posição do seu sucessor



Pilha (Stack) Estática Representação

- Para criar uma pilha estática a partir de um vetor necessitamos basicamente das seguintes informações:
  - Um vetor que irá conter os objetos da pilha (stackData).
  - Este vetor irá conter um número máximo de objetos (iMaxStackSize)
  - O vetor irá utilizar um tipo genérico de dados (T)
  - Um número inteiro que irá indicar o índice do TOPO da pilha (iTop).
  - Inicialmente o TOPO da pilha é igual a -1.
  - O TOPO deverá ser sempre menor ou igual ao tamanho máximo do vetor (iMaxStackSize)



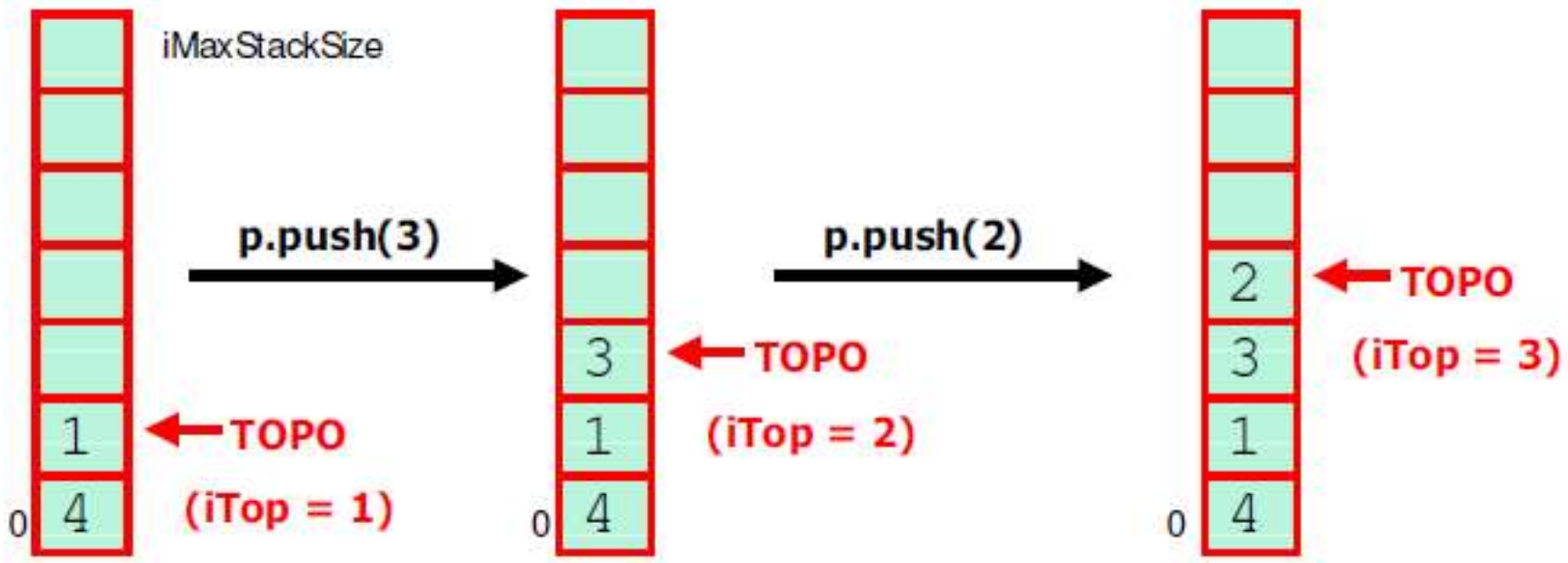
Pilha
Operações

- É possível a inserção (**push**) de novos objetos no TOPO da pilha
- É possível a remoção (**pop**) somente do objeto que se encontra no TOPO da pilha
- Somente o objeto que está no TOPO da pilha pode ser recuperado (**peek**)

Pilha – Operações
Inserção

- push
  - Insere um elemento no topo da pilha caso a mesma não esteja cheia
  - Retorna uma constante indicando sucesso ou falha

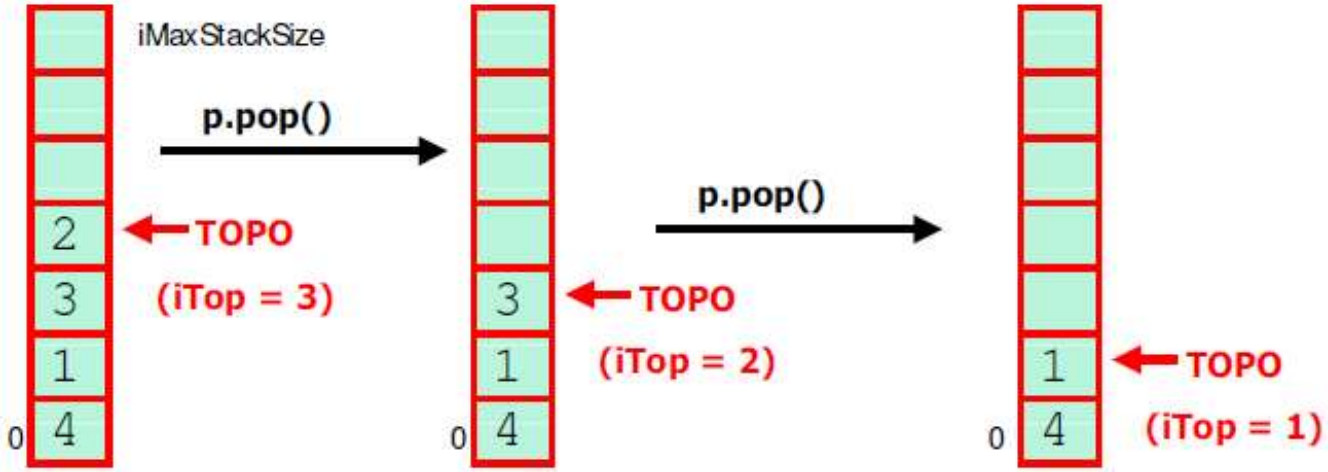
```
int push (const T& item)
```



Pilha – Operações Remoção

- pop
  - Remove o objeto que se encontra no topo da Pilha
  - Retorna o objeto removido e uma constante indicando sucesso ou falha

```
int pop(T& item) - C++
int pop(struct Stack* s, T* item) - C
```



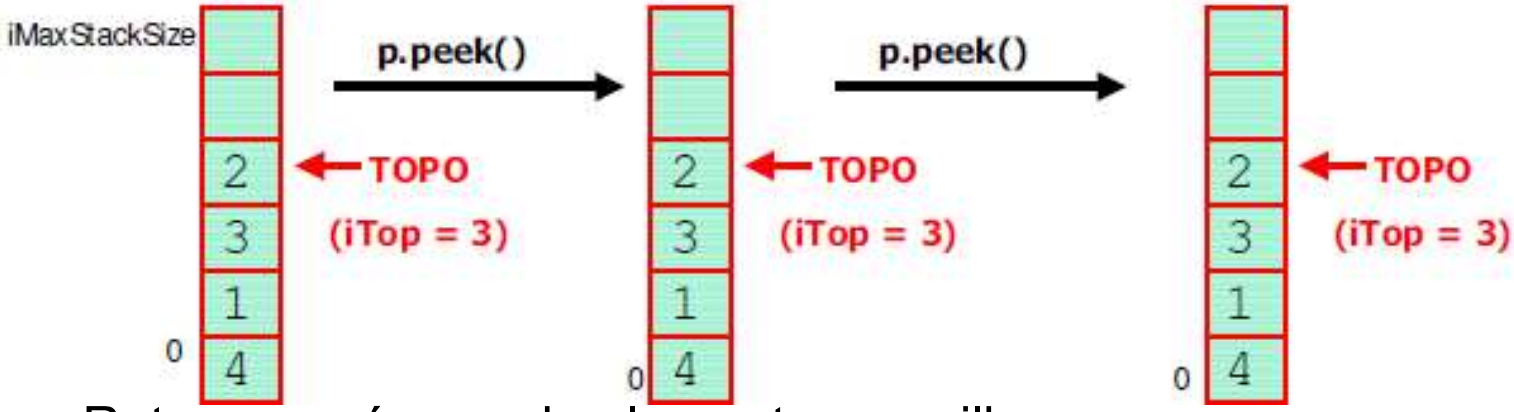
- removeAll
  - Remove todos os objetos que se encontram na Pilha

```
void removeAll()
```

Pilha – Operações Consulta e Recuperação

- peek
  - Recupera o elemento que se encontra no TOPO na pilha, sem no entanto retirá-lo
  - Retorna uma constante indicando sucesso ou falha que ocorre quando a pilha está vazia, por exemplo

```
int peek(T& item)
```



- getSize - Retorna o número de elementos na pilha

```
int getSize(void)
```

- find – Encontra a primeira ocorrência de um determinado objeto

```
bool find(T& item) const
```

Pilha – Operações
Métodos de Apoio

- isEmpty
  - Retorna um booleano indicando se a pilha está ou não vazia

```
bool isEmpty(void) const;
```

- isFull
  - Retorna um booleano indicando se a pilha está ou não cheia

```
bool isFull(void) const;
```

- print
  - Imprime o conteúdo da pilha

```
Void print() const;
```

Pilha
Definição - C

```
#define MAXSTACKDATA 10
typedef double T;

struct Stack {
  T stackData[MAXSTACKDATA];
  int iTop;
};
```

Pilha
Definição – C++

```
#define MAXSTACKDATA 10

template <class T> class Stack {
  private:
    T stackData[MAXSTACKDATA];
    int iTop;
  public:
    //operacoes
};
```

Pilha
Definição – Pilha.h

```
# ifndef STACK_CLASS
# define STACK_CLASS
# include <iostream>
# define iMaxStackSize 32
typedef int T;

class Stack {
   private:
     //Vetor que irá conter os elementos da pilha
     T stackData[ iMaxStackSize];
     //Inteiro que indica o índice do topo da pilha
     int iTop;
   public:
     //construtor
     Stack(void)
```

Pilha
Definição – Pilha.h

```
    //Operações de Modificação
    //Armazena um item na pilha no topo da pilha
    void push (const T&);
    //Remove o topo da pilha
    T pop (void);
    //Remove todos os objetos da pilha
    void removeAll(void);
    //Operações de Consulta
    // Obtém objeto que se encontra no topo da pilha sem modifica-lo
    T peek (void) const;
    int getSize(void);
    bool find(T& item) const;
    //Métodos de Apoio
    bool isEmpty(void) const;
    bool isFull(void) const;
   void print()const;
};
#endif
```

Pilha
Implementação – Pilha.cpp

```
// Construtor do objeto pilha
Stack::Stack (void){
 iTop = -1
}
//Armazena um item na pilha no topo da pilha
int Stack::push (const T& item){
   // Verifica se a pilha está cheia, retorna constante indicando erro
   if (isFull()) {
     cout << "A pilha está cheia!" << endl;
     return 1;
   }
   // incrementa o topo da pilha
   iTop++;
   //coloca o elemento no topo da pilha
   stackData[iTop] = item;
   return 0;
}
```

Pilha
Implementação – Pilha.cpp

```
//Remove o topo da pilha e o elemento que se encontra nesta posição
int Stack::pop (T& elem) {
   // Verifica se a pilha está vazia. Caso esteja retorna condição
   erro
   if (isEmpty()) {
     cout << "A pilha está vazia!" << endl;
     return 1;
   }
   // obtem o elemento que está no topo
   elem = stackData[ iTop] ;
   //decrementa o topo da pilha
   iTop--;
   return 0;
}
```

Pilha
Implementação – Pilha.cpp

```
// Obtem o objeto que se encontra no topo da pilha sem modificá-la
int Stack::peek (T& item) const {
   // Verifica se a pilha está vazia. Caso esteja o programa será
   finalizado
   if (isEmpty()) {
     cout << "A pilha está vazia!" << endl;
     return 1;
   }
   item = stackData[ iTop] ;
   return 0;
}

//Verifica se a pilha está vazia
bool Stack::isEmpty(void) const {
   // Retorna true ou false, dependendo do resultado da comparação
   return (iTop == -1);
}
```

Pilha
Implementação – Pilha.cpp

```
//Verifica se a pilha está cheia
bool Stack::isFull(void) const {
   // Retorna true ou false, dependendo do resultado da comparação
   return (iTop == iMaxStackSize-1);
}
// Limpa a pilha
void Stack::removeAll(void) {
   // Coloca o valor do topo como -1
   iTop = -1;
}
```

Pilha
Implementação – Pilha.cpp

```
void Stack::print()const {
   //Começa a partir do topo
   int ii(iTop);
   //Caso a pilha seja vazia retorna
   if (isEmpty()){
     cout << "Pilha Vazia!" << endl;
     return;
   }
   cout << "Imprimindo o conteudo da Pilha..." << endl;
   while (ii >= 0){
      cout << "Elemento: " << ii << " - " << stackData[ ii] << endl;
      ii--;
   }
}
# endif //STACK_CLASS
```

Pilha
Implementação – Pilha.cpp

```
//Verifica se a pilha está cheia
bool Stack::isFull(void) const {
   // Retorna true ou false, dependendo do resultado da comparação
   return (iTop == iMaxStackSize-1);
}
// Limpa a pilha
void Stack::removeAll(void) {
   // Coloca o valor do topo como -1
   iTop = -1;
}
```

Pilha
Implementação – Pilha.cpp

```
void Stack::print()const {
   //Começa a partir do topo
   int ii(iTop);
   //Caso a pilha seja vazia retorna
   if (isEmpty()){
     cout << "Pilha Vazia!" << endl;
     return;
   }
   cout << "Imprimindo o conteudo da Pilha..." << endl;
   while (ii >= 0){
      cout << "Elemento: " << ii << " - " << stackData[ ii] << endl;
      ii--;
   }
}
# endif //STACK_CLASS
```