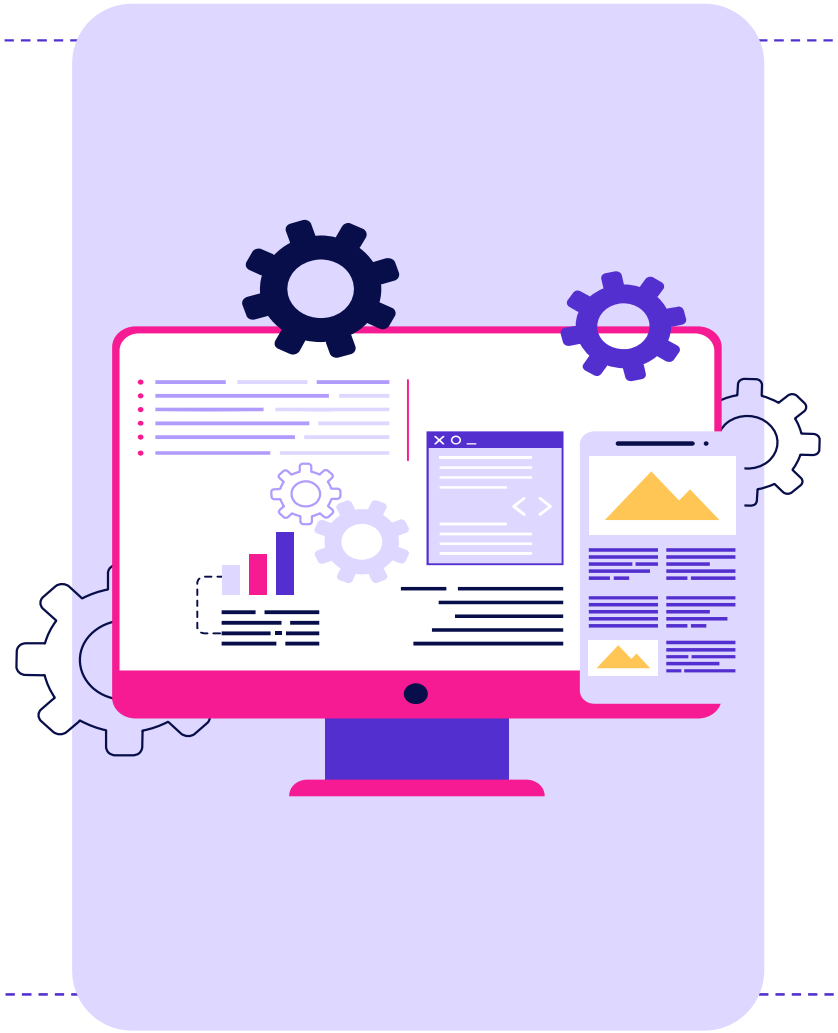
Estrutura de Dados 1

PILHA ESTÁTICA

PILHA DINÂMICA

Profª Juliana Franciscani



Roteiro

01 **PILHA Estática Conceito**

02 **PILHA Estática Código**

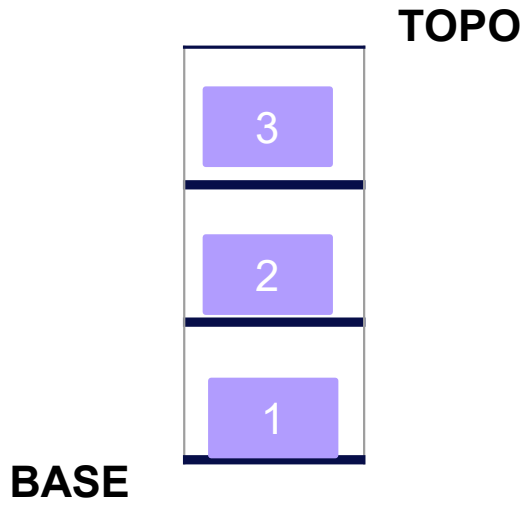
03 **PILHA Dinâmica Conceito**

04 **PILHA Dinâmica Código**

05 **Exercícios**

PILHA

- Sequência de elementos de um mesmo tipo
- Tipo especial de lista
- É uma estrutura em que os elementos são inseridos sempre no topo da pilha, e a remoção também é no topo.

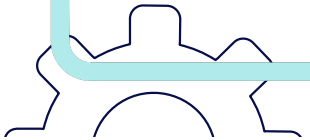


PILHA

FILO (First In Last Out) Primeiro elemento a entrar é o último elemento a sair.
Inserção sempre no TOPO da Pilha
Remoção sempre no TOPO da Pilha
Só é visível o Topo da Pilha, apenas o último elemento estará visível.

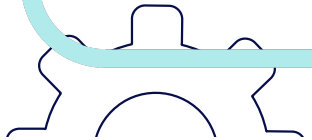
APLICAÇÃO DE PILHA

- Análise de expressão matemática
- Avaliação de expressão pós-fixa
- Conversão de expressão in para pós-fixa
- Conversão de decimal para binário



PILHA Estática

- Espaço de memória é definido no momento de **compilação**
- Deve-se definir o tamanho máximo MAX do vetor a ser utilizado
- **Acesso é sequencial:** elementos consecutivos na memória
- Deve-se verificar se a PILHA está cheia antes de cada inserção.



PILHA Dinâmica

- Espaço de memória é definido no momento de **execução**
- A PILHA cresce a cada elemento inserido e diminui a cada elemento removido.
- **Acesso é Encadeado:** cada elemento pode estar em uma área distinta da memória.
- Para acessar um elemento é preciso percorrer todos os seus antecessores

PILHA

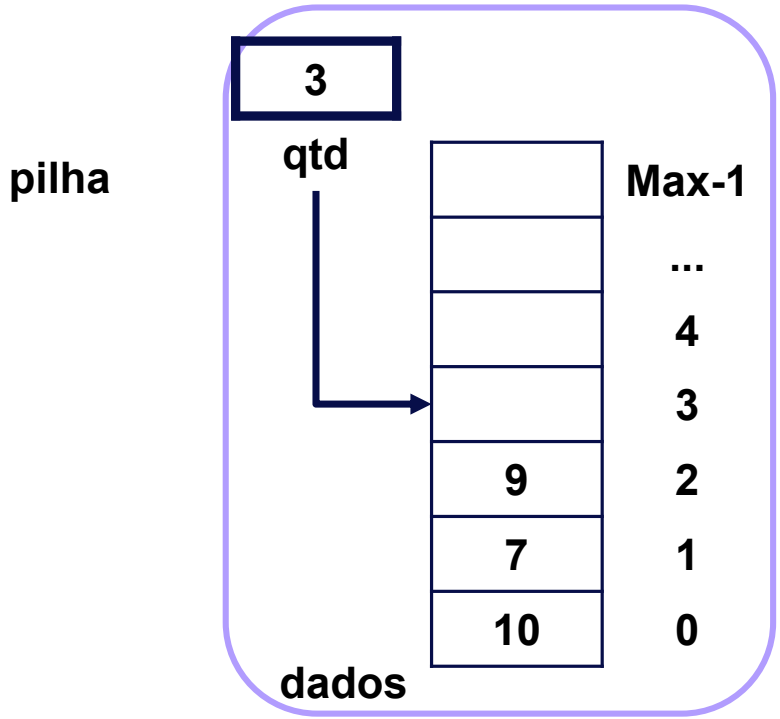
- Criar a pilha;
- Inserir item no topo da pilha;
- Acessar um elemento do topo da pilha;
- Remover o elemento do topo da pilha;
- Contar número de itens;
- Verificar se a pilha está vazia;
- Verificar se a pilha está cheia (estática)

PILHA ESTÁTICA

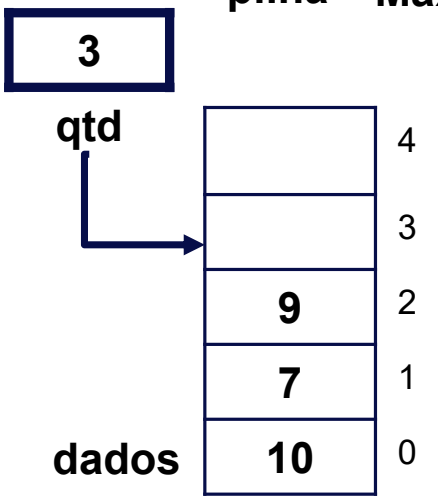
Explicação
Código em CPP

PILHA ESTÁTICA

- É composta por informações como: quantidade de elementos, além do vetor que armazenará os dados.
- Necessário informar o tamanho máximo (MAX) desse vetor.



pilha Max = 5

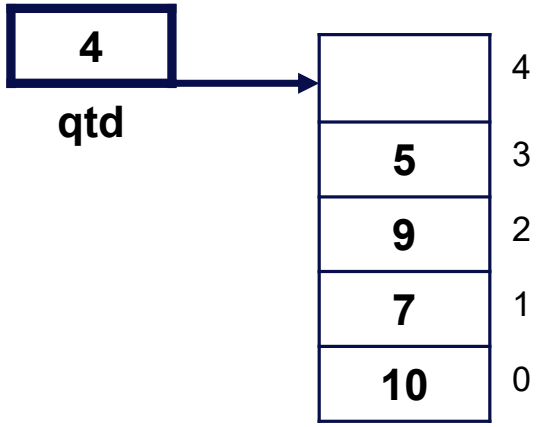


Inserir o 5 na pilha.

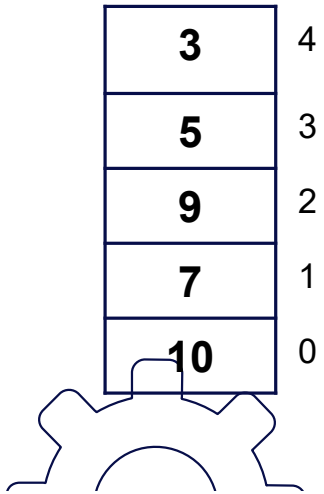
Pilha está cheia?

Insere no topo
dados[qtd]=5

Atualiza o qtd qtd++



Inserir o 3 na pilha

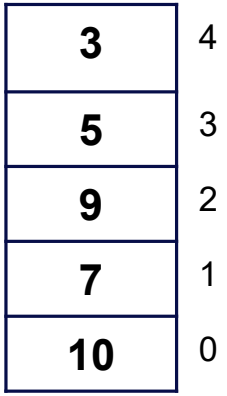


Pilha está cheia?

Insere no topo
dados[qtd]=3

Atualiza o qtd qtd++

Inserir o 20 na pilha



Pilha está cheia?
Qtd==Max?

Não é possível inserir

pilha Max = 5

3

qtd

	4
	3
9	2
7	1
10	0

dados

Remover

Pilha está vazia?

Remove do topo

Atualiza o qtd qtd--

2

qtd

	4
	3
9	2
7	1
10	0

Remover

Pilha está vazia?

Remover do topo

Atualiza o qtd qtd--

1

qtd

	4
	3
9	2
7	1
10	0

Remover

Pilha está vazia?

Remover do topo

Atualiza o qtd qtd--

Remover

0

qtd

	4
	3
9	2
7	1
10	0

Pilha está vazia?
Qtd==0?

Não é possível remover

main.cpp

```cpp
#include <iostream>
#include "pilhaEstatica.h"
using namespace std;

int main(){
    int opc;
    Pilha pilha;
    criarPilha(&pilha);
    cout << "\nPilha Estática!\n";
    do{
        cout << "Informe a opção desejada:\n"
                "1 - para inserir um elemento na pilha.\n"
                "2 - para exibir um elemento da pilha.\n"
                "3 - para remover um elemento da pilha.\n"
                "0 - para sair do menu.\n--> ";
        cin >> opc;
```

```
17          switch(opc){
18              case 1:
19                  inserirPilha(&pilha);
20              break;
21              case 2:
22                  exibirElemento(&pilha);
23              break;
24              case 3:
25                  removerElemento(&pilha);
26              break;
27              case 0:
28                  cout << "Saindo do menu!\n";
29              break;
30              default: cout << "Opção Inválida!\n";
31          }
32          cout << "\n---------------------\n\n";
33      }while(opc!=0);
34      return 0;
35  }
```

pilhaEstatica.h

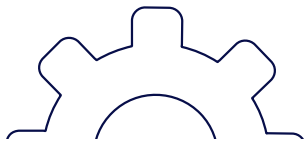
```
1  #ifndef PILHAESTATICA_H_INCLUDED
2  #define PILHAESTATICA_H_INCLUDED
3
4  struct Pilha{
5      int dados[5];
6      int topo;
7  };
8  void criarPilha(Pilha *pilha);
9  void inserirPilha(Pilha *pilha);
10 void exibirElemento(Pilha *pilha);
11 void removerElemento(Pilha *pilha);
12 #endif // PILHAESTATICA_H_INCLUDED
```

pilhaEstatica.cpp

```cpp
#include "pilhaEstatica.h"
#include <iostream>
using namespace std;

void criarPilha(Pilha *pilha){
    pilha->topo=0;
}

void inserirPilha(Pilha *pilha){
    if(pilha->topo < 5){
        cout << "Informe o elemento que deseja inserir na pilha: ";
        cin >> pilha->dados[pilha->topo];
        pilha->topo++;
    }
    else
        cout<<"Pilha está cheia, impossível inserir!\n";
}
```



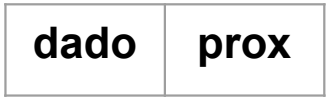
```
18
19  void exibirElemento(Pilha *pilha){
20      if(pilha->topo>0)
21          cout << "Ultimo elemento da pilha é: " << pilha->dados[pilha->topo-1];
22      else
23          cout << "Não há elementos na pilha.\n";
24  }
25
26  void removerElemento(Pilha *pilha){
27      if(pilha->topo>0)
28          pilha->topo--;
29      else
30          cout << "Não há elementos na pilha.\n";
31  }
```

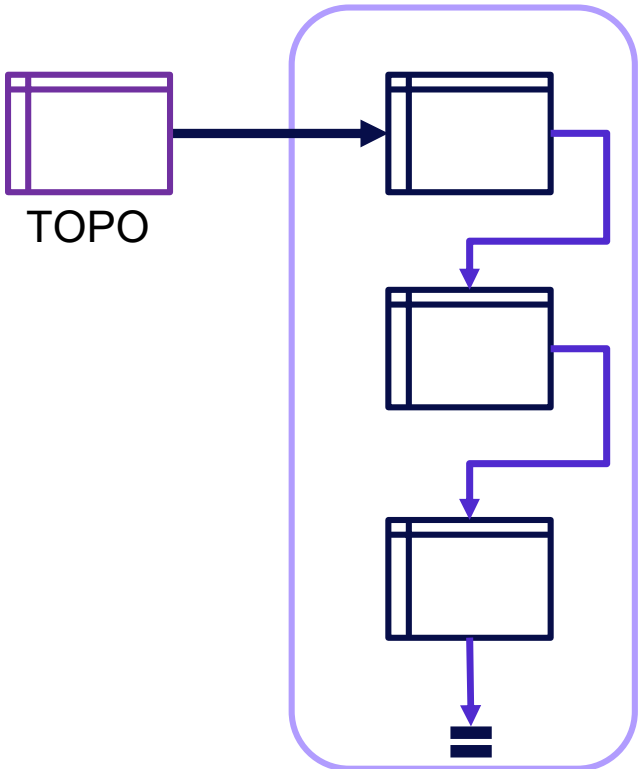
PILHA DINÂMICA

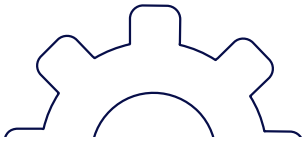
Explicação
Código em CPP

PILHA DINÂMICA

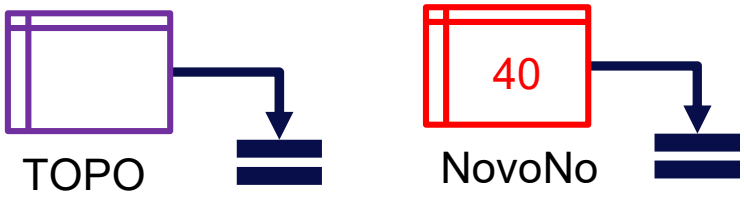
- Alocação dinâmica para cada elemento um NEW
- Composta pelo dado, um ponteiro para o próximo elemento e indicadores (ponteiros) para o topo da pilha







Inserção do elemento 40 em uma PILHA que não possui elementos

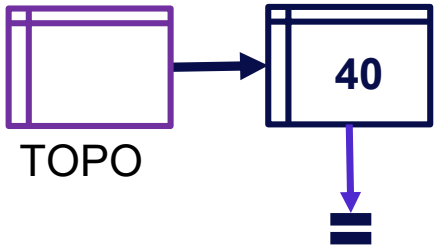


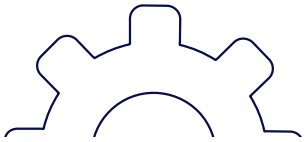
Pilha foi criada corretamente?

Pilha está vazia?
Topo aponta para nulo?

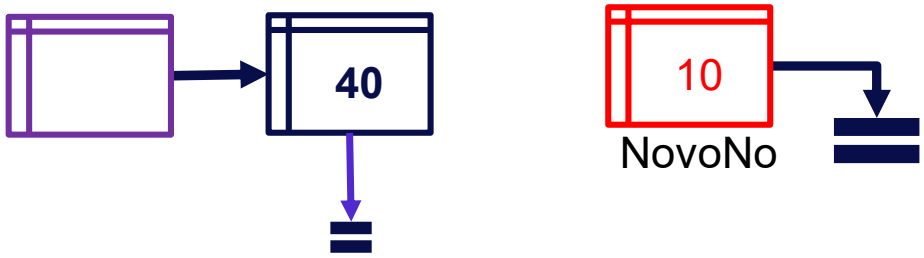
Como é o caso...
Aloca memória para novoNo
Atribui o valor a ele e prox para nulo

Topo aponta para novoNo





Inserção do elemento 10 em uma PILHA que não está vazia



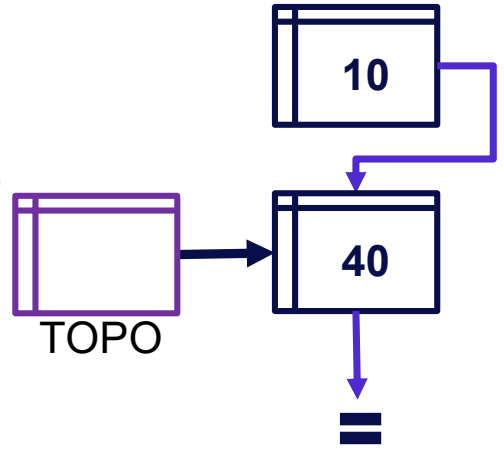
Pilha foi criada corretamente?

Pilha está vazia?
Topo aponta para nulo?

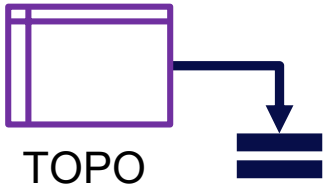
Como não é o caso...
Aloca memória para novoNo

novoNo->prox = *Topo

Atualiza o topo



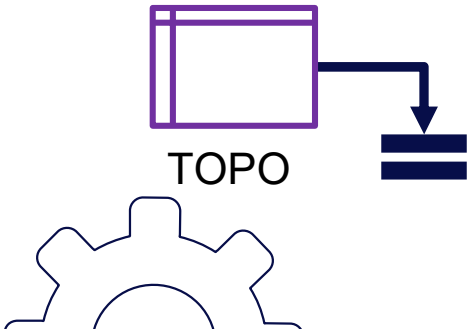
Remoção em uma Pilha vazia



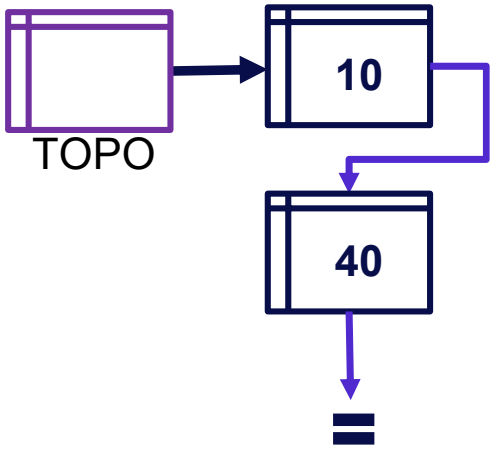
Pilha foi criada corretamente?

Pilha está vazia?
Topo aponta para nulo?

Como é o caso...
Mensagem que pilha está vazia
Não há elementos para remover



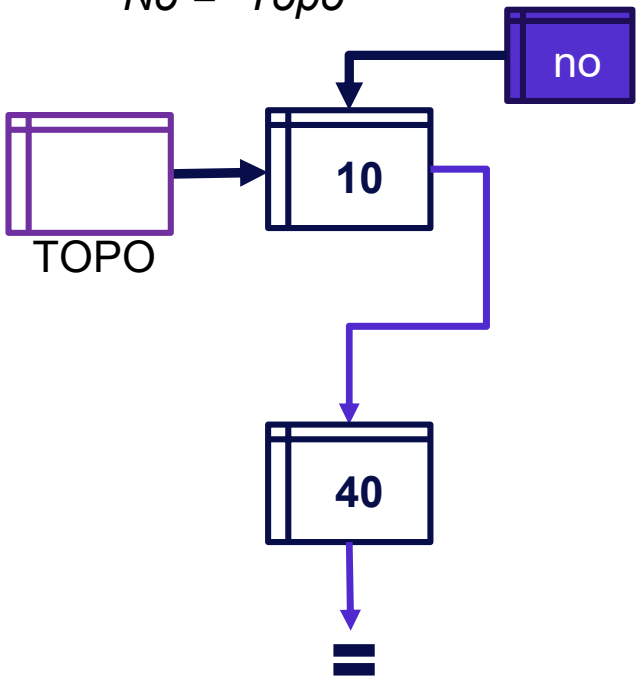
Remoção na PILHA com elementos



Pilha foi criada?

Pilha está vazia?
Topo aponta para nulo?

Como não é o caso...
Cria variável tipo no
*No = *Topo*



Atualiza o topo

Desaloca memória
Do no.

```cpp
#include <iostream>
#include <windows.h>
#include "pilhaDinamica.h"

using namespace std;
char menuInicial();
char menuSaida();

int main(){
    SetConsoleCP(1252);
    SetConsoleOutputCP(1252);
    Aluno alunoN;
    char menu;
    Pilha* topo = criarPilha();
    do{
        menu=menuInicial();
        switch(menu){
        case '1':
            cadastrarAluno(&alunoN);
            inserirPilha (topo, &alunoN);
            break;
        case '2':
            removerPilha(topo);
            break;
       case '3':
            consultarTopo(topo);
            break;
        default: cout<< "Opção inválida!";
        }
        menu = menuSaida();
        system("clear||cls");
    }while(menu!='S');
    liberarPilha(topo);
    return 0;
}
```

```
36  char menuInicial(){
37      char menu;
38      cout <<"\n --------- Menu --------\n"
39              "1 - para inserir aluno na pilha\n"
40              "2 - para remover um aluno da pilha\n"
41              "3 - exibir o topo da pilha\n"
42              "--> ";
43      fflush(stdin);
44      cin>>menu;
45      return menu;
46  }
47
48  char menuSaida(){
49      char menu;
50      cout<<"\nDeseja sair do programa? S para sim "
51              "e qualquer tecla para continuar...\n--> ";
52      cin >> menu;
53      menu = toupper(menu);
54      return menu;
55  }
```

```c
#ifndef PILHADINAMICA_H_INCLUDED
#define PILHADINAMICA_H_INCLUDED

struct ALUNO{
    int matricula;
    char nome[30];
    float nota;
};
typedef struct ALUNO Aluno;
struct NODE{
    Aluno dados;
    struct NODE *prox;
};
typedef struct NODE *Pilha;
typedef struct NODE No;

Pilha* criarPilha();
void liberarPilha(Pilha* topo);
int consultarTopo(Pilha* topo);
int inserirPilha(Pilha* topo, Aluno *alunoN);
int removerPilha(Pilha* topo);
void cadastrarAluno(Aluno *alunoN);
#endif // PILHADINAMICA_H_INCLUDED
```

```cpp
#include <iostream>
#include "pilhaDinamica.h" //inclui os Protótipos
using namespace std;

Pilha* criarPilha(){
    Pilha* topo = new Pilha;
    if(topo != nullptr){
        *topo=nullptr;
        cout << "Pilha Criada com sucesso!\n";
    }
    return topo;
}

void liberarPilha(Pilha *topo){
    if(topo != nullptr){
        No *noAux;
        while((*topo) != nullptr){
            noAux = *topo;
            *topo = noAux->prox;//*topo=top->prox;
            delete noAux;
        }
        delete topo;
    }
}
```

```
26  int consultarTopo(Pilha* topo){
27      if(topo == nullptr){
28          cout << "Pilha não existe!\n";
29          return 0;
30      }
31      if(*topo == nullptr){//pilha vazia
32          cout << "Pilha Vazia!\n";
33          return 0;
34      }
35      No *noAux = *topo;
36      cout << noAux->dados.nome << endl;
37      cout << noAux->dados.matricula << endl;
38      cout << noAux->dados.nota << endl;
39      return 1;
40  }
```

```
41  int removerPilha(Pilha* topo){
42       if(topo == nullptr){
43           cout << "Pilha não existe!\n";
44           return 0;
45       }
46       if(*topo == nullptr){//pilha vazia
47           cout << "Pilha Vazia!\n";
48           return 0;
49       }
50       No *noAux = *topo;
51       *topo = noAux->prox;
52       delete noAux;
53       cout << "Remoção realizada com sucesso!\n";
54       return 1;
55  }
```

```
57  int inserirPilha(Pilha* topo, Aluno *alunoN){
58      if(topo == nullptr){
59          cout << "Fila não existe!\n";
60          return 0;
61      }
62      No *novoNo = new No;
63      if(novoNo == nullptr){
64          cout << "Espaço de memória não alocado para o nó!\n";
65          return 0;
66      }
67      novoNo->dados = *alunoN;
68      novoNo->prox = nullptr;
69
70      novoNo->prox=*topo;
71      *topo=novoNo;
72      cout << "Inserção realizada com sucesso!\n";
73      return 1;
74  }
```

```
76  int exibirTopo(Pilha* topo){
77      if(topo == nullptr){
78          cout << "Pilha não existe!\n";
79          return 0;
80      }
81      if(*topo == nullptr){//pilha vazia
82          cout << "Não há elementos na Pilha!\n";
83          return 0;
84      }
85      No* noAux = *topo;
86      cout << "Nome:" << noAux->dados.nome<< "\n";
87      cout << "Matricula: " << noAux->dados.matricula << "\n";
88      cout <<"Nota: " <<noAux->dados.nota << "\n\n\n";
89      return 1;
90  }
91
```

No* exibirTopo::noAux

```
 92  void cadastrarAluno(Aluno *alunoN){
 93      cout << "Cadastro Aluno:\n";
 94      cout <<"Nome: ";
 95      fflush(stdin);
 96      cin.getline(alunoN->nome, sizeof(alunoN->nome));
 97      cout << "Matrícula: ";
 98      cin >> alunoN->matricula;
 99      cout << "Nota: ";
100      cin >> alunoN->nota;
101  }
```

Exercícios

1. A partir da pilha estática a seguir faça a inserção e a remoção na estrutura. Considere o tamanho máximo da pilha como 5. Na pilha, os números 3 e 5 já estão inseridos. Faça os seguintes comandos: Remover, Remover, Remover, Inserir (10), Inserir (3), Inserir (7), Remover.

2. Considerando a estrutura pilha dinâmica faça a inserção e remoção a partir de uma pilha vazia. Faça os seguintes comandos: Remover; Inserir (10); Inserir (15); Remover; Inserir(2); Remover.

Referências

EDELWEISS, N,; GALANTE, R.. Estruturas de dados. Porto Alegre: Bookman, 2009.

SZWARCFITER, J. L.; MARKENZON, L. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro: LTC, 2010.

ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C. 3. ed. rev. e ampl. São Paulo: Cengage Learning, 2011.

Aulas e vídeo aulas do professor André Backes: https://www.facom.ufu.br/~backes/