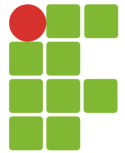
Pilhas

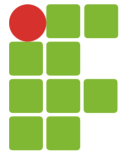
Estruturas de Dados e Algoritmos

Prof. Daniel Saad Nogueira Nunes

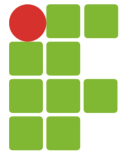
IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário



Sumário

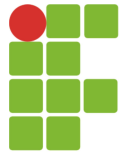


Introdução

Pilhas

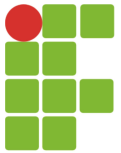
- Pilhas são um TAD em qual os elementos são mantidos em uma ordem específica. Esta ordem é a ordem LIFO (Last-in-First-Out)
- A ordem LIFO se caracteriza pelo fato dos últimos elementos a fazerem parte da estrutura, também serão os primeiros elementos a deixarem a estrutura.

Pilhas

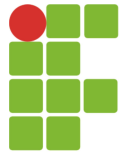


Operações sobre Pilhas

- Algumas das operações suportadas por uma pilha devem ser:
  - Empilhar elementos;
  - Desempilhar elementos;
  - Verificar o topo da pilha;
  - Verificar se a pilha está vazia.

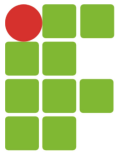


Sumário

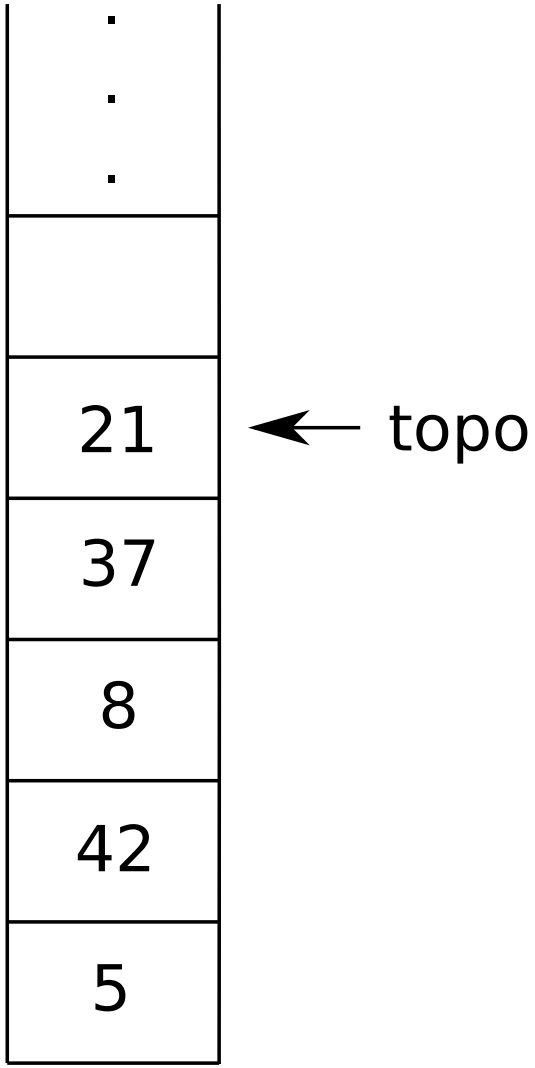


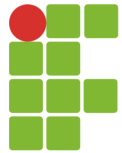
Implementação de Pilhas

- Pilhas podem ser implementadas através de vetores.
- Necessitamos de um índice que aponta para o topo da pilha.



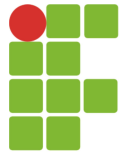
Implementação de Pilhas sobre Vetores





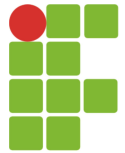
Implementação de Pilhas sobre Vetores

- Para verificar se a pilha está vazia, basta verificar se topo corresponde ao índice -1 do vetor ou se a variável associada ao tamanho da pilha é nula.
- Para acessar o topo da pilha, basta acessar o vetor na posição marcada pelo topo.
- Para empilhar um elemento no vetor, incrementamos a posição do topo e inserimos o elemento nesta posição.
- Para desempilhar um elemento do vetor, decrementamos a posição do topo.



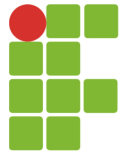
Representação de Pilhas sobre Vetores

- A representação de pilhas sobre vetores é bem rápida.
- Mas não funciona tão bem no cenário dinâmico.
- O vetor tem um limite, e portanto, a pilha possui um limite.



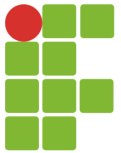
Implementação de Pilhas

- Pilhas podem ser implementadas por meio de estruturas auto-referenciadas.
- Uma das estruturas que podem prover as funcionalidades de uma pilha é uma lista Ligada.

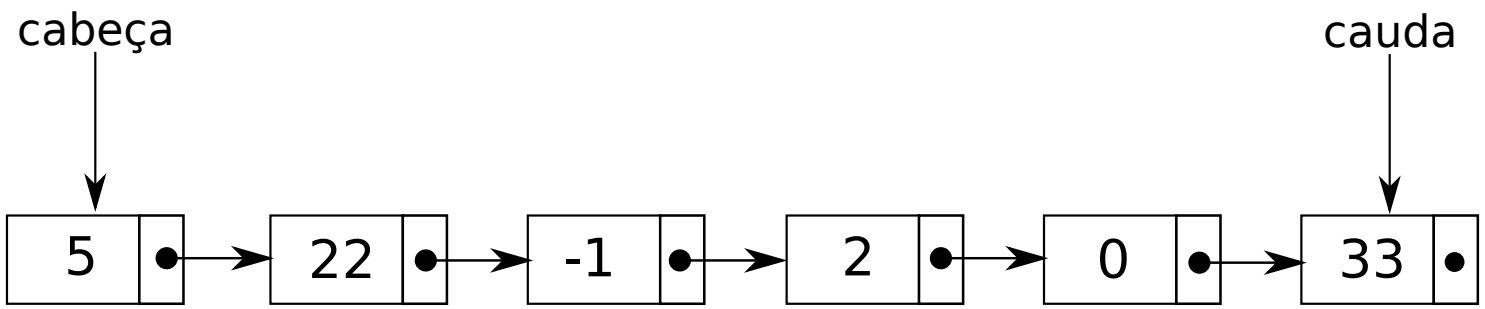


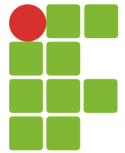
Implementação de Pilhas sobre Listas

- Utilizando listas, a operação de verificar se a pilha está vazia equivale à verificar se a lista está vazia.
- Para empilhar um elemento, insere-se um elemento na cabeça.
- Para desempilhar um elemento, retira-se da cabeça.
- Para acessar o topo da pilha, a cabeça deve ser acessada.

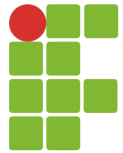


Implementação de Pilhas sobre Listas



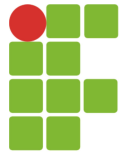


Sumário



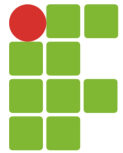
Pilhas: Definição

```
6  typedef void* (*stack_node_constructor_fn) (void*);
7  typedef void (*stack_node_destructor_fn)(void *);
```



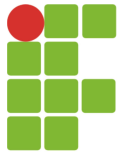
Pilhas: Definição

```
10 typedef struct stack_node_t{
11     void* data;
12     struct stack_node_t* next;
13 }stack_node_t;
```

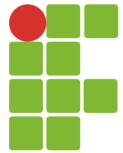


Pilhas: Definição

```
18 typedef struct stack_t{
19     stack_node_t* top;
20     stack_node_constructor_fn constructor;
21     stack_node_destructor_fn destructor;
22     size_t size;
23 }stack_t;
```

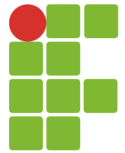


Sumário

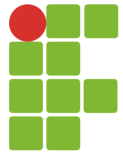


Pilhas: Inicialização

```
7  void stack_initialize(stack_t** s,stack_node_constructor_fn constructor,
8                        stack_node_destructor_fn destructor){
9      (*s) = mallocx(sizeof(stack_t));
10     (*s)->size = 0;
11     (*s)->top = NULL;
12     (*s)->constructor = constructor;
13     (*s)->destructor = destructor;
14 }
```

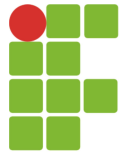


Sumário



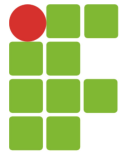
Pilhas: Funções Auxiliares

```
53  size_t stack_size(stack_t* s){
54      return(s->size);
55  }
```

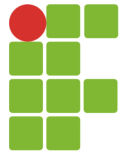


Pilhas: Funções Auxiliares

```
58  size_t stack_empty(stack_t* s){
59      return(s->size==0 ? 1 : 0);
60  }
```

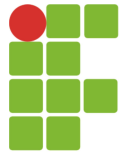


Sumário

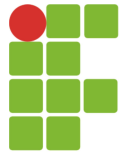


Pilhas: Empilhar

```
38  void stack_push(stack_t* s,void* data){
39      stack_node_t* new_node = mallocx(sizeof(stack_node_t));
40      new_node->data = s->constructor(data);
41      new_node->next = s->top;
42      s->top = new_node;
43      s->size++;
44  }
```

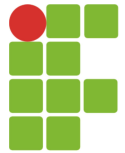


Sumário

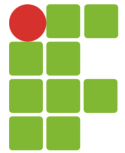


Pilhas: Obter o Topo

```
47  void* stack_top(stack_t* s){
48      assert(!stack_empty(s));
49      return(s->top->data);
50  }
```

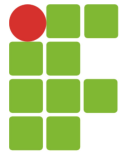


Sumário

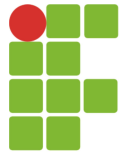


Pilhas: Desempilhar

```
28  void stack_pop(stack_t* s){
29      assert(!stack_empty(s));
30      stack_iterator_t it = s->top;
31      s->top = s->top->next;
32      s->destructor(it->data);
33      free(it);
34      s->size--;
35  }
```

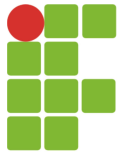


Sumário

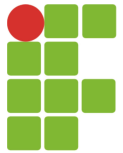


Pilhas: Limpeza

```
17  void stack_delete(stack_t** s){
18      while(!stack_empty(*s)){
19          stack_pop(*s);
20      }
21      free(*s);
22      (*s) = NULL;
23  }
```



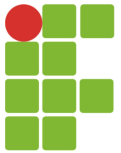
Sumário



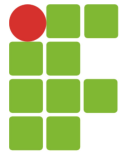
Pilhas

Complexidade das Operações

Operação	Complexidade
Empilhar	$\Theta(1)$
Desempilhar	$\Theta(1)$
Verificar topo	$\Theta(1)$

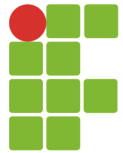


Sumário



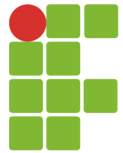
Exemplo de Utilização da Biblioteca

```
6  typedef struct pessoa{
7      char nome[30];
8      char cpf[20];
9      int idade;
10 }pessoa;
```



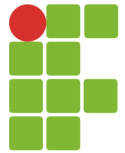
Exemplo de Utilização da Biblioteca

```
12  void* constructor_pessoa(void* data){
13      void* ptr = mallocx(sizeof(pessoa));
14      memcpy(ptr,data,sizeof(pessoa));
15      return ptr;
16  }
```



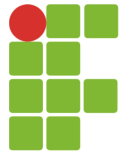
Exemplo de Utilização da Biblioteca

```
35  void destructor_pessoa(void* data){
36      free(data);
37  }
```



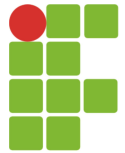
Exemplo de Utilização da Biblioteca

```
18  void my_getline(char* str,size_t size){
19      int i;
20      char c;
21      for(i=0;i<size-1;i++){
22          c = getchar();
23          if(c=='\n'){
24              str[i] = '\0';
25              break;
26          }
27          str[i] = c;
28      }
29      str[size-1]='\0';
30      while(c!='\n'){
31          c = getchar();
32      }
33  }
```



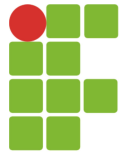
Exemplo de Utilização da Biblioteca

```
39  void cadastra_pessoa(pessoa* p){
40      printf("Nome: ");
41      my_getline(p->nome,30);
42      printf("CPF: ");
43      my_getline(p->cpf,20);
44      printf("Idade: ");
45      scanf("%d%*c",&p->idade);
46  }
```



Exemplo de Utilização da Biblioteca

```
48  void imprime_pessoa(const pessoa* p){
49      printf("Nome: ");
50      printf("%s\n",p->nome);
51      printf("CPF: ");
52      printf("%s\n",p->cpf);
53      printf("Idade: ");
54      printf("%d\n",p->idade);
55  }
```



Exemplo de Utilização da Biblioteca

```
57  int main(void){
58      int i;
59      stack_t* s;
60      pessoa p;
61      stack_initialize(&s,constructor_pessoa,destructor_pessoa);
62      for(i=0;i<5;i++){
63          printf("Cadastrando pessoa %d\n",i+1);
64          cadastra_pessoa(&p);
65          stack_push(s,&p);
66      }
67      while(!stack_empty(s)){
68          printf("\n**Imprimindo pessoa**\n");
69          p =  *(pessoa*) stack_top(s);
70          stack_pop(s);
71          imprime_pessoa(&p);
72
73          printf("\n");
74      }
75      stack_delete(&s);
76      return 0;
77  }
```