Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

centro universitário unifacvest

4. Pilha (Stack)

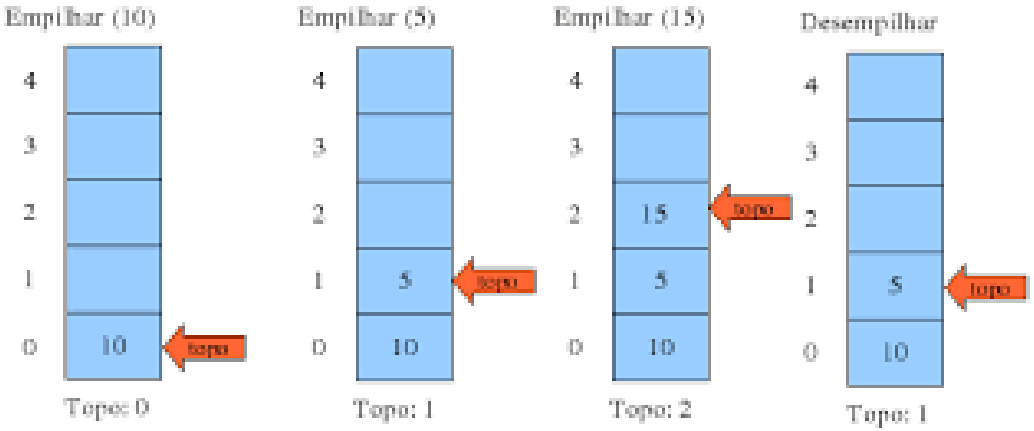
4.2. Alocação Estática;

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

- Pilha Estática:

– Tipo de Pilha onde o sucessor de um elemento ocupa a posição física seguinte do mesmo (uso de array);


© Can Stock Photo



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando uma "Pilha Estática":

- PilhaSequencial.h: definir

- Protótipos das funções;
- O tipo de dado armazenado na Pilha;
- O ponteiro da Pilha;
- Tamanho do vetor usado na pilha;

- PilhaSequencial.c: definir

- O tipo de dados pilha;
- Implementar as funções;

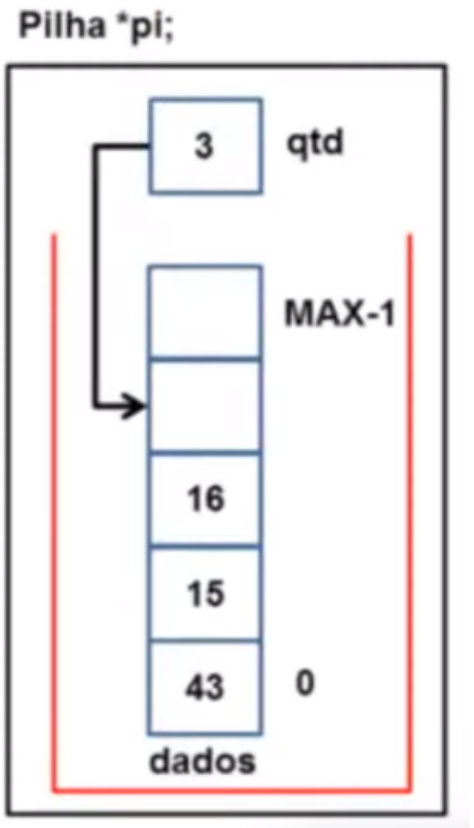
- main.c: definir

- Fazer as chamadas das funções;

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando uma "Pilha Estática":

```c
//Arquivo PilhaSequencial.h
#define MAX 100
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
typedef struct pilha Pilha;

//Arquivo PilhaSequencial.c
struct pilha{
    int qtd;
    struct aluno dados[MAX];
};

//programa principal
Pilha *pi;
```



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “cria_pilha”:

```
//programa principal
pi = cria_Pilha();

//Arquivo PilhaSequencial.h
Pilha* cria_Pilha();

//Arquivo PilhaSequencial.c
Pilha* cria_Pilha(){
    Pilha *pi;
    pi = (Pilha*) malloc(sizeof(struct pilha));
    if(pi != NULL)
        pi->qtd = 0;
    return pi;
}
```

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

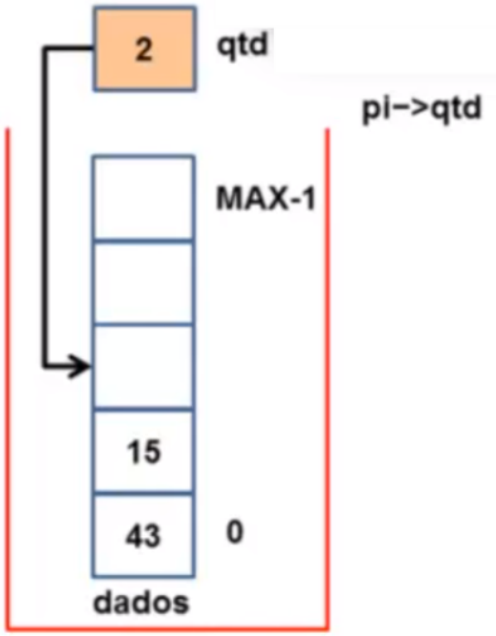
Implementando a função “libera_pilha”:

```
//programa principal
libera_Pilha(pi);

//Arquivo PilhaSequencial.h
void libera_Pilha(Pilha* pi);

//Arquivo PilhaSequencial.c
void libera_Pilha(Pilha* pi){
    free(pi);
}
```

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “tamanho_pilha”:

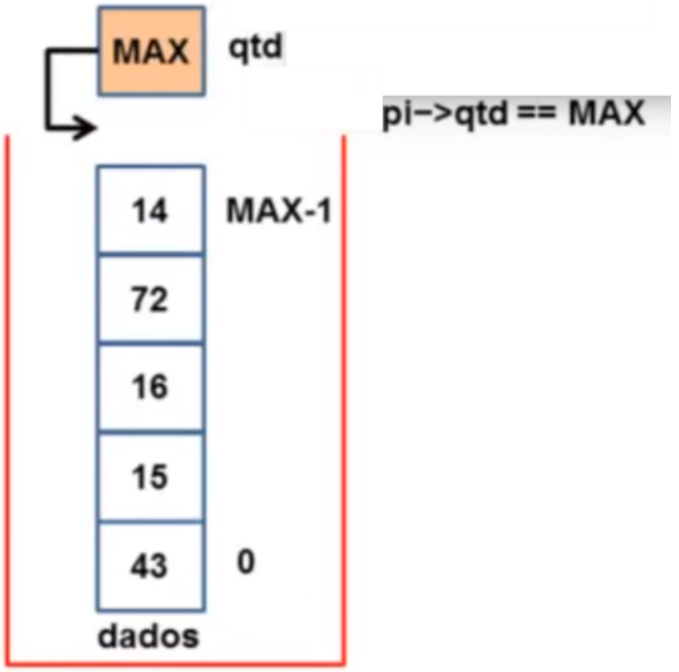
```
//programa principal
int x = tamanho_Pilha(pi);

//Arquivo PilhaSequencial.h
int tamanho_Pilha(Pilha* pi);

//Arquivo PilhaSequencial.c
int tamanho_Pilha(Pilha* pi){
    if(pi == NULL)
        return -1;
    else
        return pi->qtd;
}
```



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “pilha_cheia”:

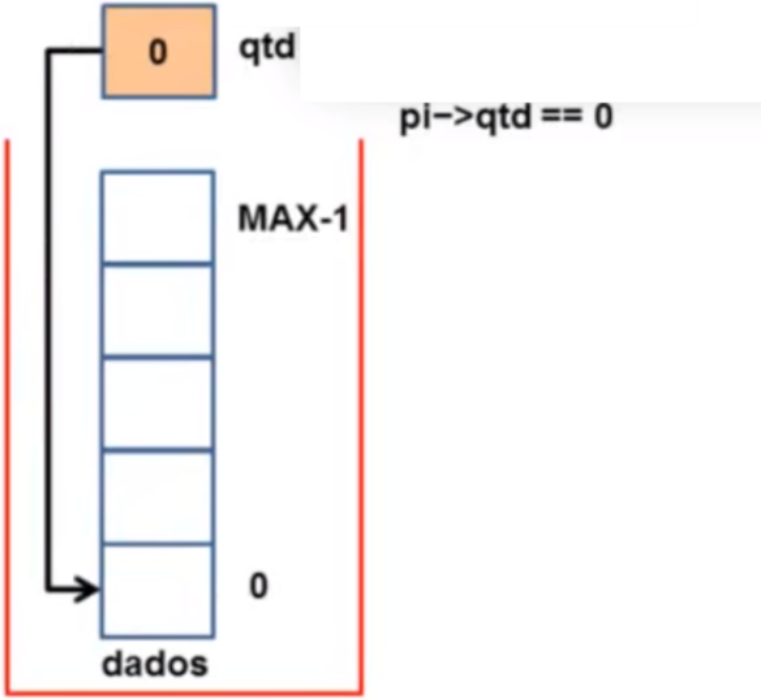
```
//programa principal
int x = Pilha_cheia(pi);
if(Pilha_cheia(pi))
//Arquivo PilhaSequencial.h
int Pilha_cheia(Pilha* pi);
//Arquivo PilhaSequencial.c
int Pilha_cheia(Pilha* pi){
    if(pi == NULL)
        return -1;
    return (pi->qtd == MAX);
}
```



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “pilha_vazia”:

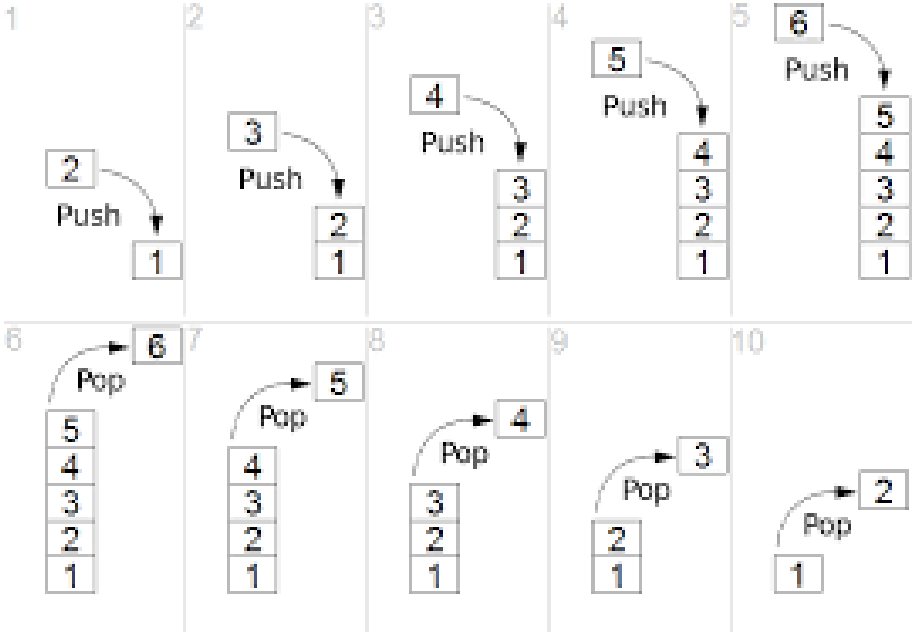
```c
//programa principal
int x = Pilha_vazia(pi);
if(Pilha_vazia(pi))
//Arquivo PilhaSequencial.h
int Pilha_vazia(Pilha* pi);
//Arquivo PilhaSequencial.c
int Pilha_vazia(Pilha* pi){
    if(pi == NULL)
        return -1;
    return (pi->qtd == 0);
}
```



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “insere_Pilha”: (Push)

- Lembre-se que em uma Pilha a inserção é sempre feita no início, no topo;
- Em uma Pilha estática não podemos inserir quando está cheia;



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “insere_Pilha”: (Push)

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaSequencial.h"
int main(){
    Pilha *pi;
    pi = cria_Pilha();
    struct aluno dados_aluno;
    x = insere_Pilha(pi, dados_aluno);
    libera_Pilha(pi);
}
```

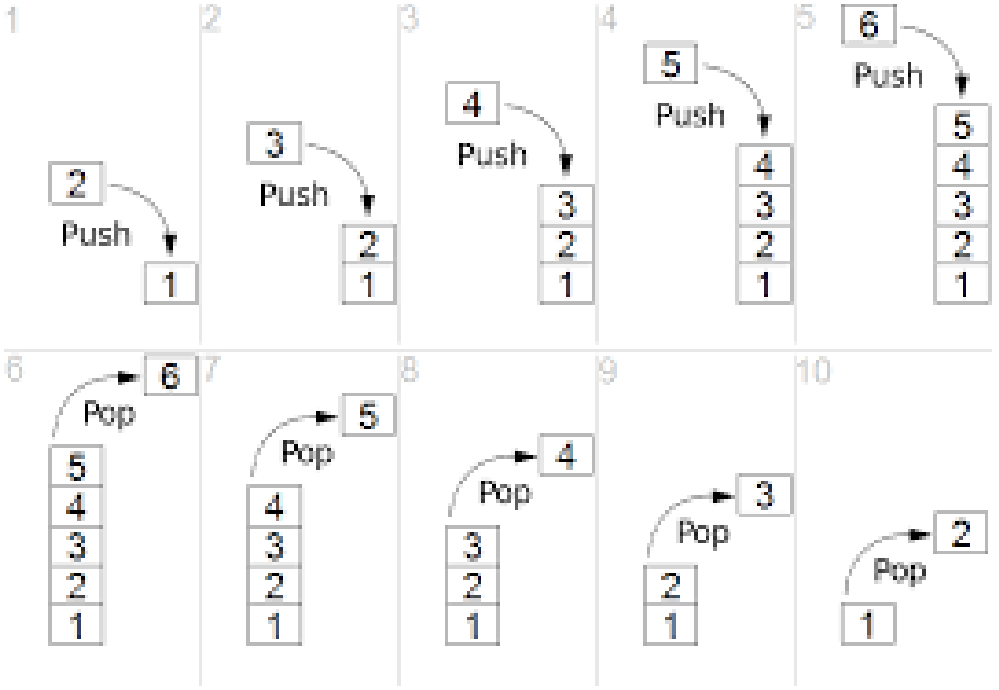
```c
//Arquivo PilhaSequencial.h
#define MAX 100
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct pilha Pilha;
//*********************************
int insere_Pilha(Pilha* pi, struct aluno al);
```

```c
//Arquivo PilhaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaSequencial.h" //inclui os Protótipos

//Definição do tipo Pilha
struct pilha{
    int qtd;
    struct aluno dados[MAX];
};
//*********************************
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    if(pi->qtd == MAX)//pilha cheia
        return 0;
    pi->dados[pi->qtd] = al;
    pi->qtd++;
    return 1;
}
```

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “remove_Pilha”: (Pop)

- Em uma Pilha a remoção é sempre no seu início;
- Não é possível remover de uma Pilha vazia;



4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “remove_Pilha”: (Pop)

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaSequencial.h"
int main(){
    Pilha *pi;
    pi = cria_Pilha();
    struct aluno dados_aluno;
    in x = insere_Pilha(pi, dados_aluno);
    x = remove_Pilha(pi);
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaSequencial.h
#define MAX 100
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct pilha Pilha;
//*********************************
int remove_Pilha(Pilha* pi);
```

```c
//Arquivo PilhaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaSequencial.h" //inclui

//Definição do tipo Pilha
struct pilha{
    int qtd;
    struct aluno dados[MAX];
};
//*********************************
int remove_Pilha(Pilha* pi){
    if(pi == NULL || pi->qtd == 0)
        return 0;
    pi->qtd--;
    return 1;
}
```

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “consulta_topo_Pilha”:

- Em uma Pilha a consulta se dá apenas ao elemento do Topo;

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaSequencial.h"
int main(){
    Pilha *pi;
    pi = cria_Pilha();
    struct aluno dados_aluno;
    in x = insere_Pilha(pi, dados_aluno);
    x = consulta_topo_Pilha(pi, &dados_aluno);
    x = remove_Pilha(pi);
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaSequencial.h
#define MAX 100
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct pilha Pilha;
//*********************************
int consulta_topo_Pilha(Pilha* pi, struct aluno *al);
```

```c
//Arquivo PilhaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaSequencial.h" //inclui os Protótipos

//Definição do tipo Pilha
struct pilha{
    int qtd;
    struct aluno dados[MAX];
};
//*********************************
int consulta_topo_Pilha(Pilha* pi, struct aluno *al){
    if(pi == NULL || pi->qtd == 0)
        return 0;
    *al = pi->dados[pi->qtd-1];
    return 1;
}
```

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

Implementando a função “imprime_Pilha”:

```c
//Arquivo PilhaSequencial.h
#define MAX 100
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct pilha Pilha;

void imprime_Pilha(Pilha* pi);
```

```c
//Arquivo PilhaSequencial.c
void imprime_Pilha(Pilha* pi){
    if(pi == NULL)
        return;
    int i;
    for(i=pi->qtd-1; i >=0; i--){
        printf("Matricula: %d\n",pi->dados[i].matricula);
        printf("Nome: %s\n",pi->dados[i].nome);
        printf("Notas: %f %f %f\n",pi->dados[i].n1,
                                   pi->dados[i].n2,
                                   pi->dados[i].n3);
        printf("--------------------------------\n");
    }
}
```

```c
        } else {
            //Arquivo main.c
            imprime_Pilha(pi);
            escolha_outra_opcao(); //mensagens.c
            break;
        }
```

4. Pilha (Stack)
4.2. Pilha com Alocação Estática

EXERCÍCIO:

- Fazer um menu para acessar as funções;

- Fazer as implementações descritas anteriormente;

- Popular a lista com informações do aluno na lista;

- Implementar mensagens para informar o usuário as ações que foram realizadas;

- Ao consultar pela posição ou matrícula, exibir as informações do elemento;

- Quando não encontrar um elemento exibir uma mensagem para o usuário;

- Quando não for possível realizar a operação, exibir a mensagem de lista vazia ou lista cheia conforme o caso.

- Quando inserir ou remover um elemento informar ao usuário se a operação ocorreu com sucesso ou não.

Referências

EDELWEISS, Nina; GALANTE, Renata. Estruturas de Dados. Porto Alegre, BOOKMAN, 2009.

HEINZLE, Roberto. Estruturas de Dados: implementações com C e Pascal. Blumenau, DIRETIVA, 2006.

TENENBAUM, Aron M. Estrutura de Dados usando C. São Paulo, Makron Books, 1995.

FORBELLONE, André Luiz Villar: EBERSPÄCHER, Henri Frederico. Lógica de Programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo, PRENTICE HALL, 2005.

KOFFMAN, Elliot B.; WOLFGANG, Paul A. T. Objetos, abstração, estruturas de dados e projeto usando C++. Rio de Janeiro, LTC, 2008.

PEREIRA, Silvio do lago. Estruturas de dados fundamentais: conceitos e aplicações. São Paulo, Érica, 1996.

VILLAS, Marcos Viana et al. Estruturas de dados – Conceitos e técnicas de implementação. Rio de Janeiro, Campus, 1993.

VELOSO, Paulo et al. Estrutura de dados. Rio de Janeiro, Campus, 1996.

Canal do Youtube: Linguagem C Programação Descomplicada

Site: https://programacaodescomplicada.wordpress.com/

Obrigado!