Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

centro universitário unifacvest

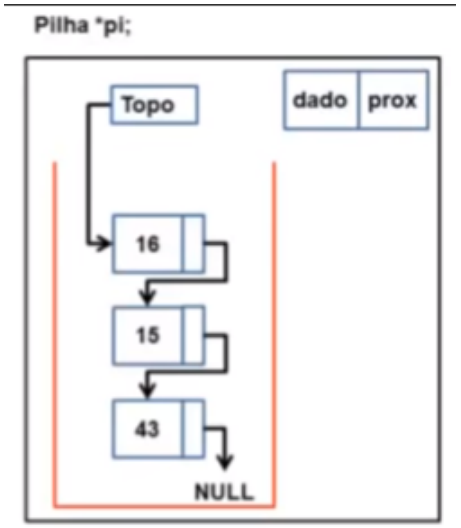
4. Pilha (Stack)

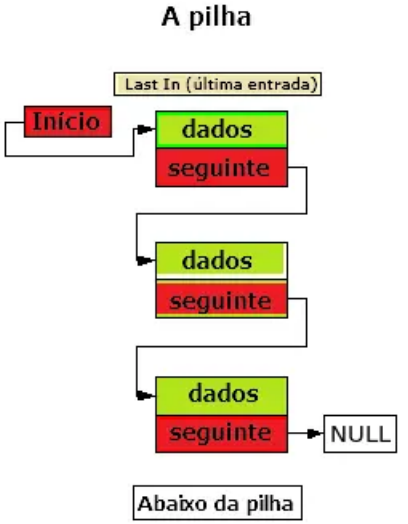
4.3. Alocação Dinâmica Encadeada;

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

- Pilha Dinâmica:

- Tipo de Pilha onde cada elemento aponta para o seu sucessor na Pilha;
- Usa um ponteiro especial (ponteiro para ponteiro) para o primeiro elemento da Pilha e uma indicação de final de Pilha.





4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando uma <u>"Pilha Dinâmica":</u>

- PilhaDin.h: definir

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

- PilhaDin.c: definir

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

- main.c: definir

- Fazer as chamadas das funções;

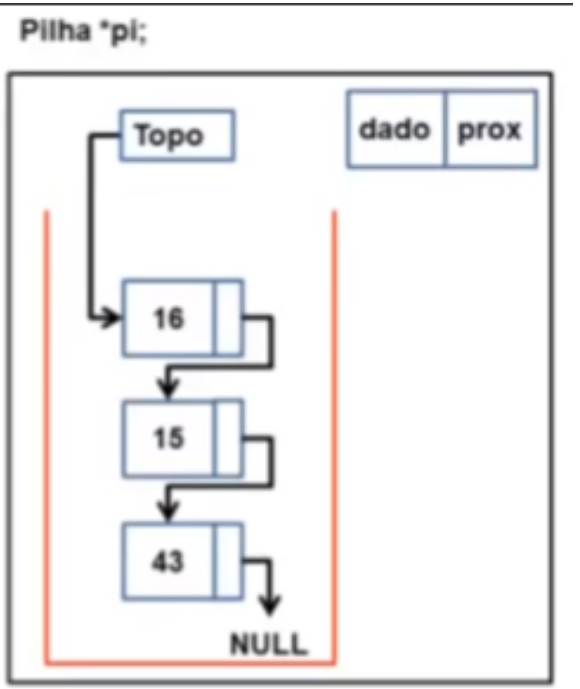
4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando uma "Pilha Dinâmica":

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h"
int main(){
    Pilha* pi;
}
```




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

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “cria_pilha”:

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h"
int main(){
    Pilha* pi;
    pi = cria_Pilha();
}

```

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
Pilha* cria_Pilha();

```

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

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;

Pilha* cria_Pilha(){
    Pilha* pi = (Pilha*) malloc(sizeof(Pilha));
    if(pi != NULL)
        *pi = NULL;
    return pi;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “libera_pilha”:

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h"
int main(){
    Pilha* pi;
    pi = cria_Pilha();
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
void libera_Pilha(Pilha* pi);
```

```c
//Arquivo PilhaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h" //inclui os Protótipo

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
void libera_Pilha(Pilha* pi){
    if(pi != NULL){
        Elem* no;
        while((*pi) != NULL){
            no = *pi;
            *pi = (*pi)->prox;
            free(no);
        }
        free(pi);
    }
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “tamanho_pilha”:

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h"
int main(){
    Pilha* pi;
    pi = cria_Pilha();
    int x = tamanho_Pilha(pi);
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
int tamanho_Pilha(Pilha* pi);
```

```c
//Arquivo PilhaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h" //inclui os Protót

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//----------------
int tamanho_Pilha(Pilha* pi){
    if(pi == NULL)
        return 0;
    int cont = 0;
    Elem* no = *pi;
    while(no != NULL){
        cont++;
        no = no->prox;
    }
    return cont;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “pilha_cheia”:

– Quando trabalhamos com estruturas dinâmicas não faz sentido verificar se ela está cheia. Demonstrado apenas para fins didáticos.

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h"
int main(){
    Pilha* pi;
    pi = cria_Pilha();
    int x = tamanho_Pilha(pi);
    x = Pilha_cheia(pi);
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
int Pilha_cheia(Pilha* pi);
```

```c
//Arquivo PilhaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h" //inclui os Protó

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//-----------------
int Pilha_cheia(Pilha* pi){
    return 0;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “pilha_vazia”:

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h"
int main(){
    Pilha* pi;
    pi = cria_Pilha();
    int x = tamanho_Pilha(pi);
    x = Pilha_vazia(pi);
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
int Pilha_vazia(Pilha* pi);
```

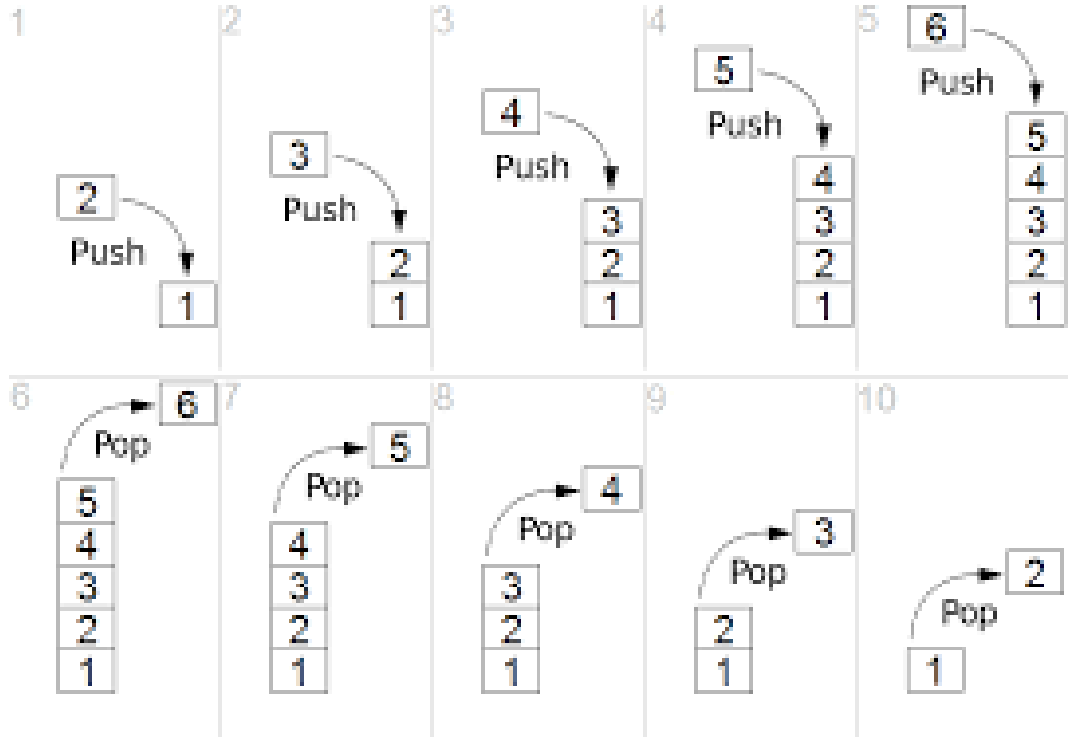
```c
//Arquivo PilhaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h" //inclui os Protótipos

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//----------------
int Pilha_vazia(Pilha* pi){
    if(pi == NULL)
        return 1;
    if(*pi == NULL)
        return 1;
    return 0;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “insere_Pilha”: (Push)

– Lembre-se que em uma Pilha a inserção é sempre feita no início, no topo;



4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “insere_Pilha”: (Push)

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

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

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

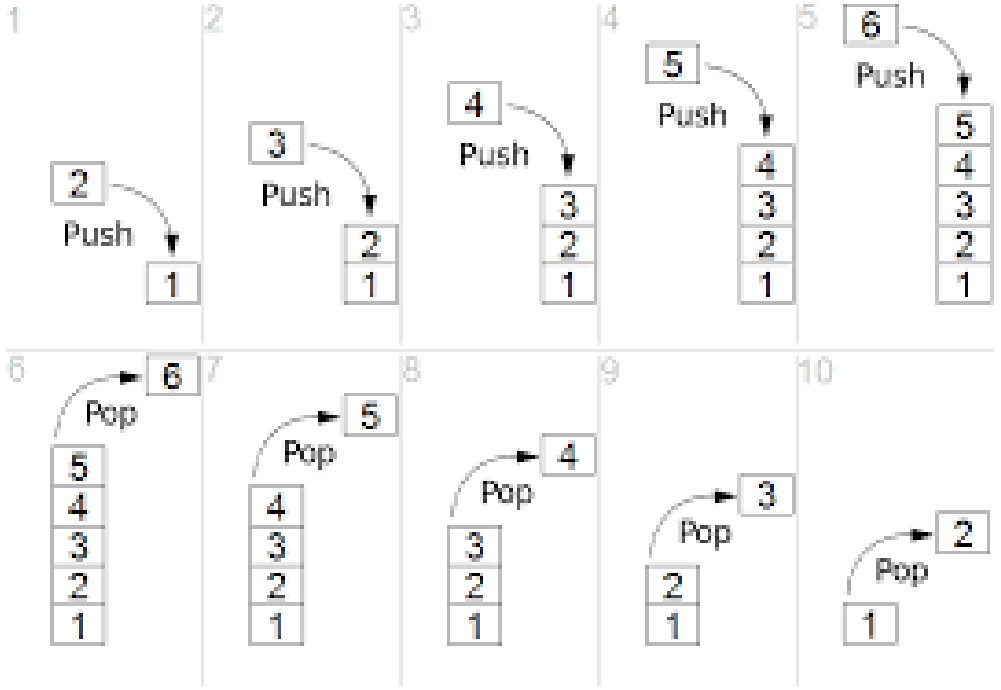
```c
//Arquivo PilhaDin.c
#include <stdio.h>
#include <stdlib.h>
#include "PilhaDin.h" //inclui os Protótipos

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//----------------
int insere_Pilha(Pilha* pi, struct aluno al){
    if(pi == NULL)
        return 0;
    Elem* no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

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)
4.3. Pilha com Alocação Dinâmica

Implementando a função “remove_Pilha”: (Pop)

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

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Pilha;
int remove_Pilha(Pilha* pi);
```

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

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//----------------
int remove_Pilha(Pilha* pi){
    if(pi == NULL)
        return 0;
    if((*pi) == NULL)
        return 0;
    Elem *no = *pi;
    *pi = no->prox;
    free(no);
    return 1;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

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 "PilhaDin.h"
int main(){
    Pilha* pi;
    pi = cria_Pilha();
    int x = tamanho_Pilha(pi);
    x = Pilha_vazia(pi);
    struct aluno dados_aluno;
    x = insere_Pilha(pi, dados_aluno);
    x = consulta_topo_Pilha(pi, &dados_aluno);
    x = remove_Pilha(pi);
    libera_Pilha(pi);
}
```

```c
//Arquivo PilhaDin.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

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

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

//Definição do tipo Pilha
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//-----------------
int consulta_topo_Pilha(Pilha* pi, struct aluno *al){
    if(pi == NULL)
        return 0;
    if((*pi) == NULL)
        return 0;
    *al = (*pi)->dados;
    return 1;
}
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

Implementando a função “imprime_Pilha”:

```
1   //Arquivo PilhaDin.h
2   struct aluno{
3       int matricula;
4       char nome[30];
5       float n1,n2,n3;
6   };
7
8   typedef struct elemento* Pilha;
9   void imprime_Pilha(Pilha* pi);
10
```

```
95  //Arquivo PilhaDin.c
96  void imprime_Pilha(Pilha* pi){
97      if(pi == NULL)
98          return;
99      Elem* no = *pi;
100     while(no != NULL){
101         printf("Matricula: %d\n",no->dados.matricula);
102         printf("Nome: %s\n",no->dados.nome);
103         printf("Notas: %f %f %f\n",no->dados.n1,
104                                    no->dados.n2,
105                                    no->dados.n3);
106         printf("-------------------------------\n");
107         no = no->prox;
108     }
109 }
```

```
66                  //Arquivo main.c
67                  imprime_Pilha(pi);
68                  msg_escolha_outra_opcao(); //mensagens.c
69                  break;
70              }
```

4. Pilha (Stack)
4.3. Pilha com Alocação Dinâmica

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!