Estruturas de Dados I

Estruturas de Dados - Pilhas

Prof. Dr. Lidio Mauro Lima de Campos

limadecampos@gmail.com

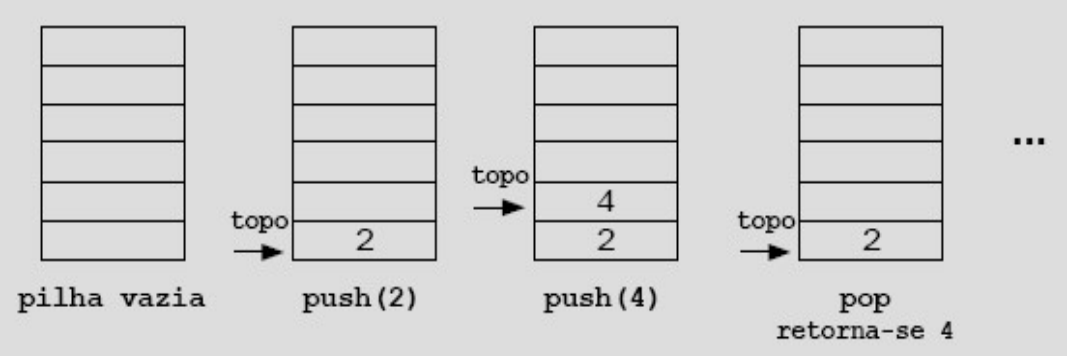
Universidade Federal do Pará – UFPA
ICEN
PPGCC

Agenda

- Pilhas
  - Implementação de pilha com Vetor.
  - Implementação de pilha com Lista.

Introdução

- Uma das estruturas de dados mais utilizadas é a pilha.
  - A inserção/acesso/remoção de elementos é sempre feita pelo **topo da pilha**.
  - O primeiro que sai é o último que entrou (***Last in, first out-LIFO***).
- As operações básicas são:
  - – **Empilha** (***push***).
  - – **Desempilhar** (***pop***).



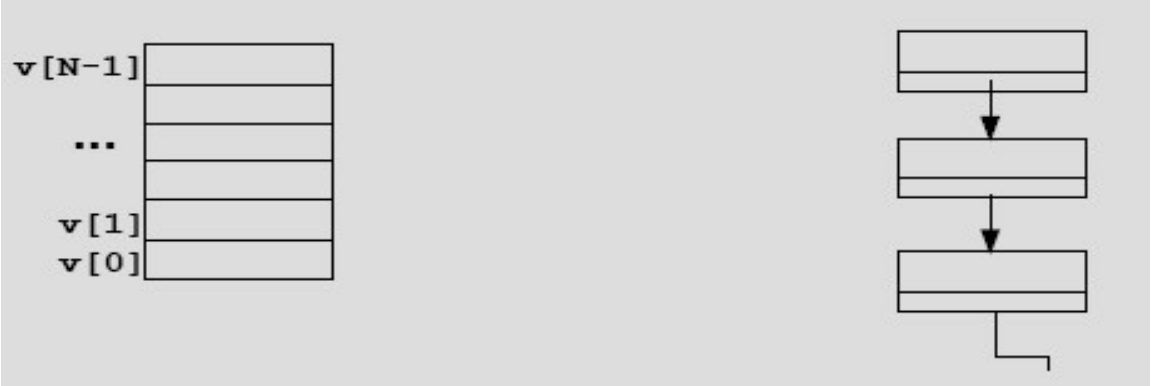
Funções de acesso Pilha

```
typedef struct pilha Pilha;
/*Função que cria uma pilha.*/
Pilha* pilha_cria(void);
/*Testa se uma pilha é vazia.*/
int pilha_vazia(Pilha *p);
/*Função que adiciona um elemento no topo de uma pilha.*/
void pilha_push(Pilha *p, int info);
/*Função que remove um elemento do topo de uma pilha.*/
int pilha_pop(Pilha *p);
/*Função que imprime os elementos de uma pilha;*/
void pilha_imprime(Pilha *p);
/*Libera o espaço alocado para uma pilha.*/
void pilha_libera(Pilha *p);
```

Implementações da Pilha

- Podemos implementar pilhas de duas maneiras:

Vetor, quando o número máximo de elementos é Conhecido.	Lista, quando não se sabe o número máximo de elementos.

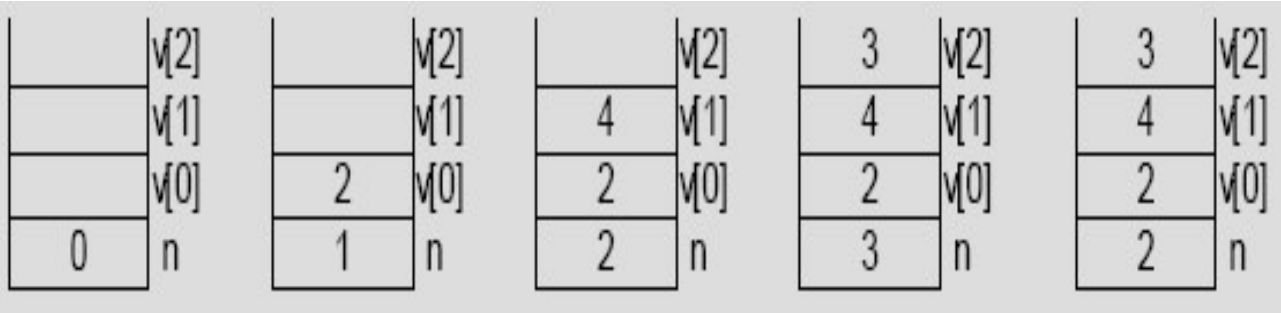


Implementação de Pilha com Vetor

- A estrutura que representa a o tipo pilha deve ser composto pelo vetor e pelo número de elementos armazenados.

```
#define N 3
typedef struct pilha
{
  int n;
  int v[N];
}Pilha;
```

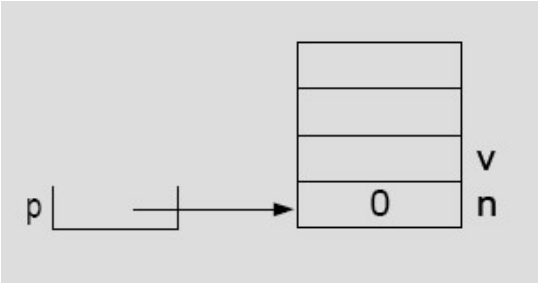
```
Pilha p1; p1.n=0;
p1.v[0] = 2; p1.n++;
p1.v[1] = 4; p1.n++;
p1.v[2] = 3; p1.n++;
p1.n--;
```



Implementação de Pilha com Vetor – Função Cria

```
Pilha* pilha_cria(void)
{
  Pilha *p = (Pilha*)malloc(sizeof(Pilha));
  if(p==NULL)
  {
   printf("Memoria insuficiente!!!\n");
   exit(1);
  }
  p->n = 0;
  return p;
}
```

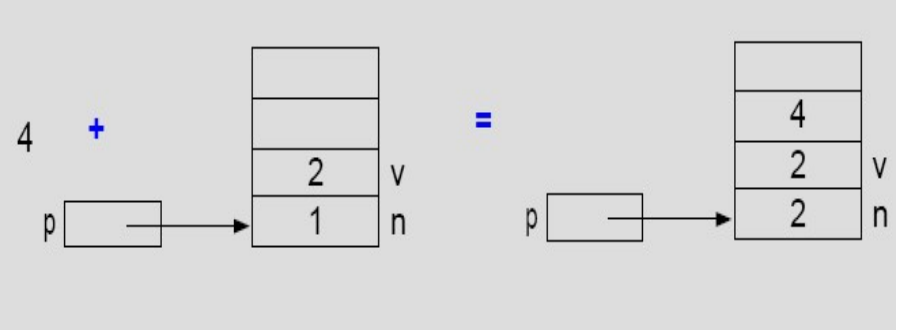
```
#define N 3
typedef struct pilha
{
  int n;
  int v[N];
}Pilha;
```



Implementação de Pilha com Vetor – Função Push

```
/*Função que adiciona um elemento no topo de uma
  pilha.*/
void pilha_push(Pilha *p, int info)
{
 if(p->n==N)
 {
  printf("Capacidade da Pilha Estorou!!!\n");
  exit(1);
 }
 p->v[p->n]= info;
 p->n = p->n + 1;
}
```

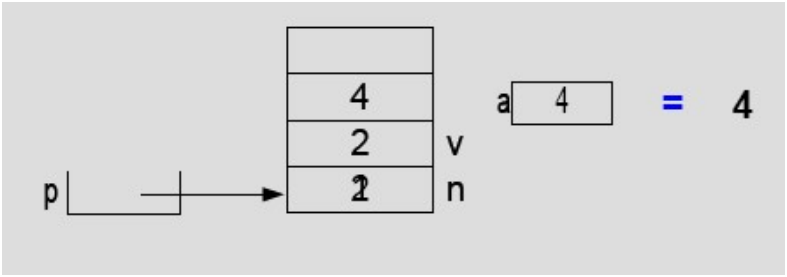
```
#define N 3
typedef struct pilha
{
  int n;
  int v[N];
}Pilha;
```



Implementação de Pilha com Vetor – Função Vazia e Pop

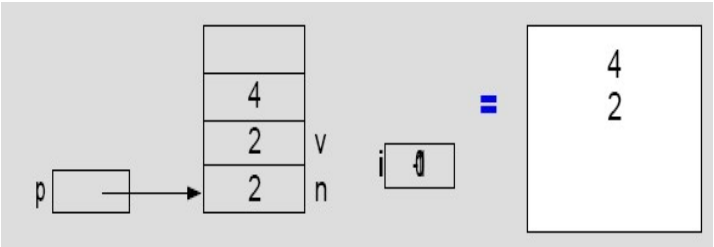
```
/*Testa se uma pilha é vazia.*/
int pilha_vazia(Pilha *p)
{  return p->n==0; }
/*Função que remove um elemento do topo de uma pilha.*/
int pilha_pop(Pilha *p)
{ int a;
  if(pilha_vazia(p))
 { printf("Pilha Vazia!!!\n");
   exit(1);
  }
  a = p->v[p->n-1] //p->n=2;
  p->n--;
  return a;
}
```

```
#define N 3
typedef struct pilha
{
  int n;
  int v[N];
}Pilha;
```



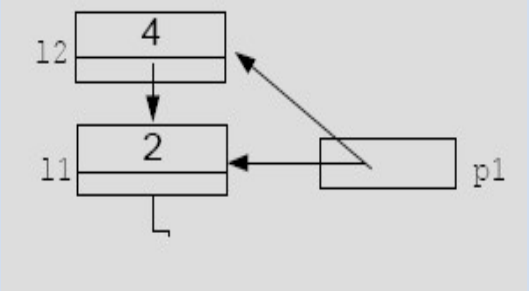
Implementação de Pilha com Vetor – Função Imprime e Libera

```
/*Função que imprime os elementos de uma pilha.*/
void pilha_imprime(Pilha *p)
{
  int i;
  for(i = p->n-1; i>=0;i--)
  {
   printf("%f\n",p->v[i]);
  }
}
/*Libera o espaço alocado para uma pilha.*/
void pilha_libera(Pilha *p)
{
  free(p);
}
```



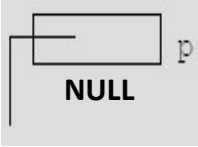
Implementação de Pilha com Lista Encadeada

- Os elementos são armazenados na lista, e a pilha pode ser representada por um ponteiro para o primeiro nó da Lista.

struct lista **{** **int info;** **Lista *prox;** **}; typedef struct lista Lista;**	**Pilha p1;** **Lista l1; l1.info=2; l1.prox=NULL** **p1.prim = l1;** **Lista l2; l2.info=4;** **l2.prox=l1; p1.prim=l2;**
struct pilha **{** **Lista *prim;** **}; typedef struct pilha Pilha;**	

Implementação de Pilha com Lista Encadeada

```
Pilha* pilha_cria(void)
{
  Pilha *p = (Pilha*)malloc(sizeof(Pilha));
  if(p==NULL)
  {
   printf("Memoria insuficiente!!!\n");
   exit(1);
  }
  p->prim = NULL;
  return p;
}
```

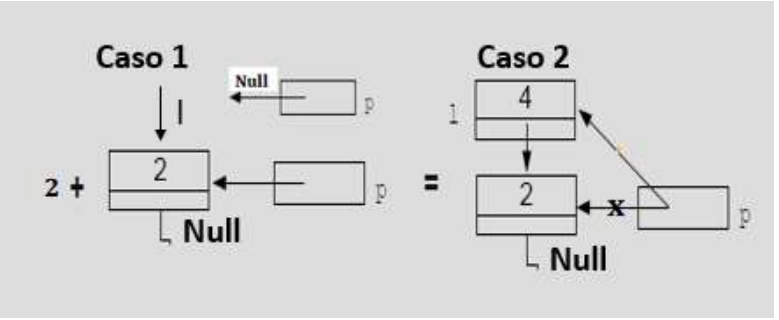


Implementação de Pilha com Lista Encadeada – Função Push

```
/*Função que adiciona um elemento no topo de uma pilha.*/
void pilha_push(Pilha *p, int info)
{
  Lista *l = (Lista*)malloc(sizeof(Lista));
  if(l==NULL)
  {
   printf("Memoria insuficiente!!!\n");
   exit(1);
  }
  l->info = info;
  l->prox = p->prim;
  p->prim = l;
}
```

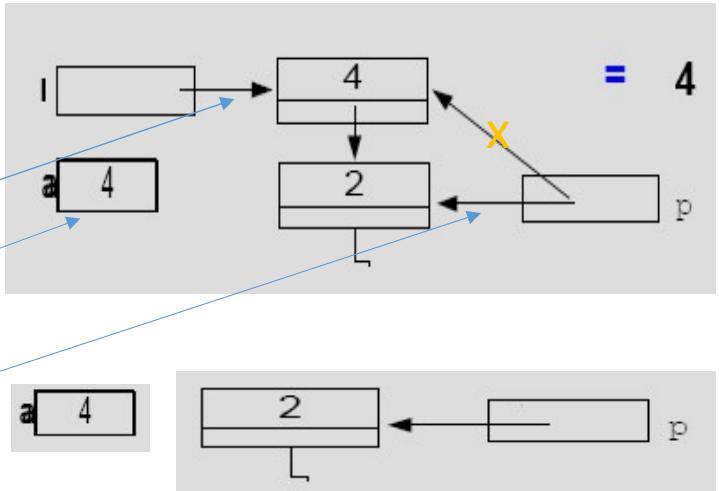
No Caso 1 – como a pilha estava vazia p->prim=NULL, logo l->prox=NULL e p->prim passa a apontar para o primeiro da lista.

No Caso 2 – a pilha não estava vazia , l->info=4, como p->prim aponta Para o primeiro nodo da lista , l->prox=p->prim (Nó com info=2), e p->prim=l Passa a apontar para o topo da lista com info=4.



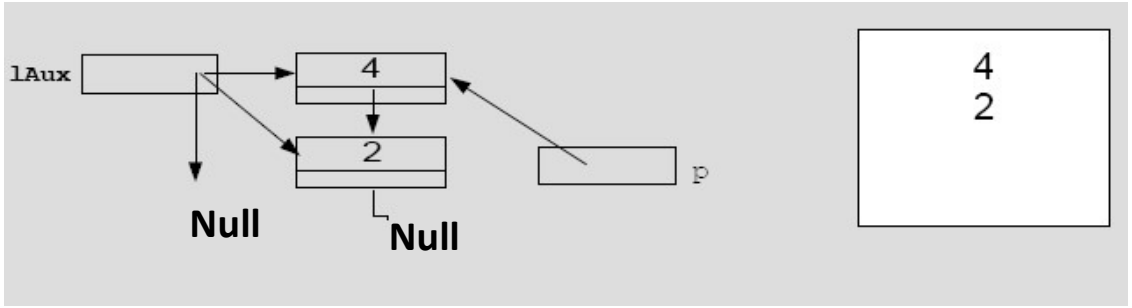
Implementação de Pilha com Lista Encadeada – Função Vazia e Pop

```
/*Testa se uma pilha é vazia.*/
int pilha_vazia(Pilha *p)
{ return p->prim==NULL; }
/*Função que remove um elemento do topo de uma pilha.*/
int pilha_pop(Pilha *p)
{ int a; Lista *l;
  if(pilha_vazia(p))
 { printf("Pilha Vazia!!!\n");
    exit(1);
  }
  l = p->prim;
  a = l->info;
  p->prim = l->prox;
  free(l);
  return a;}
```



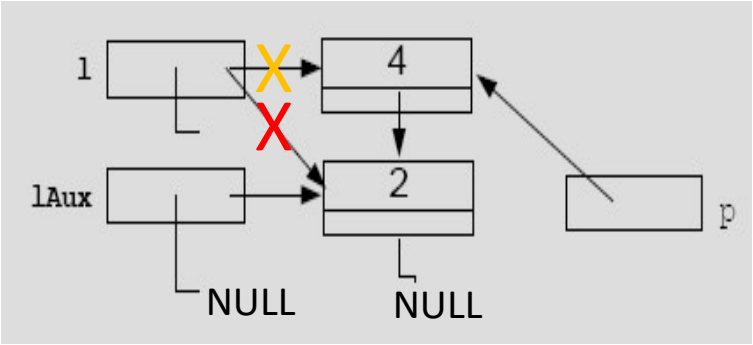
Implementação de Pilha com Lista Encadeada - Imprime

```
/*Função que imprime os elementos de uma pilha.*/
void pilha_imprime(Pilha *p)
{
  Lista *lAux = p->prim;
  while(lAux!=NULL)
  {
   printf("%f\n",lAux->info);
   lAux = lAux->prox;
  }
}
```



Implementação de Pilha com Lista Encadeada – Função Libera

```
/*Libera o espaço alocado para uma pilha.*/
void pilha_libera(Pilha *p)
{
  Lista* l = p->prim;
  Lista* lAux;
  while(l!=NULL)
  {
    lAux = l->prox;
    free(l);
    l = lAux;
  }
  free(p);
}
```



Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Algoritmos – Teoria e Prática, Tradução da Segunda Edição. Campus, 2016.
- Ziviani, N. Projeto de Algoritmos Com Implementações em Pascal e C, Pioneira Thomson Learning, 4ed. 2009.
- http://www2.hawaii.edu/~suthers/courses/ics311f20/Notes/Topic-07.html
- U. Manber. **Algorithms: A Creative Approach**. . Addison-Wesley. 1989.
- J. Kleinberg e E. Tardos. **Algorithm Design**. . Addison Wesley. 2005
- Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados, Editora Campus* (2004)