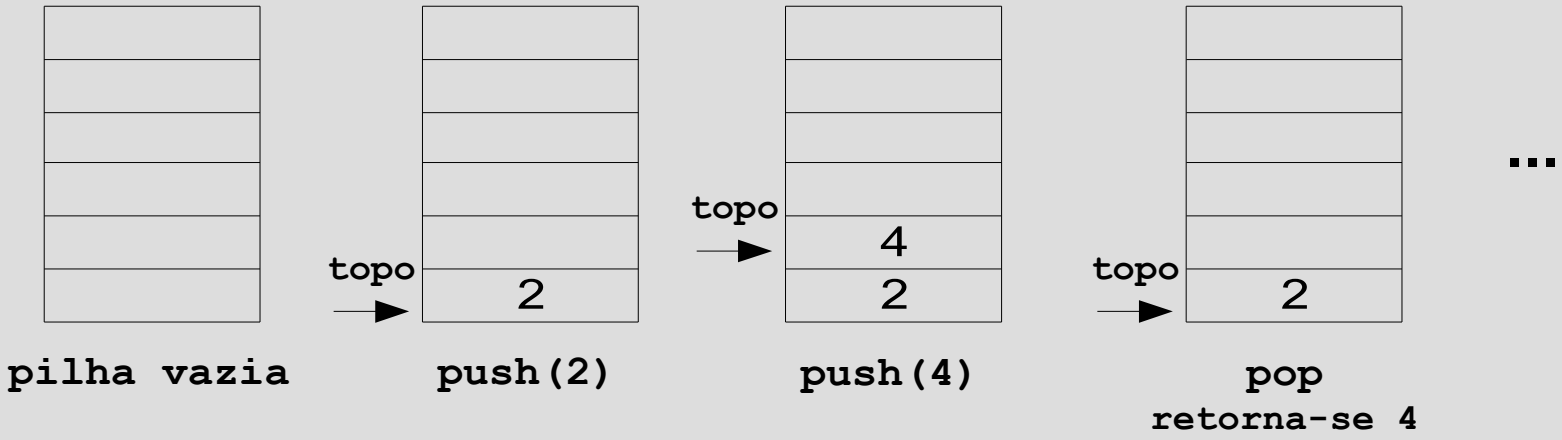
Estruturas de Dados

Pilha

Pilha

- 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)*
- As operações básicas são:
  - Empilha (*push*)
  - Desempilhar (*pop*)



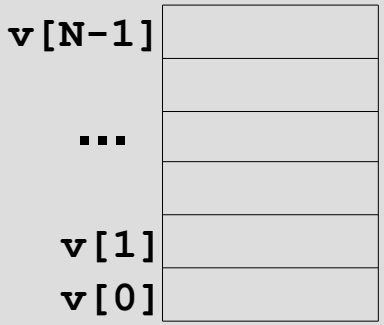
Tipo Abstrato de Dado Pilha

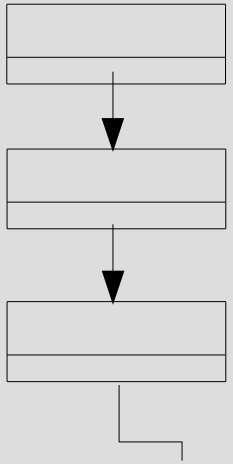
Podemos criar um TAD Pilha de Inteiros. Para tanto, devemos criar o arquivo pilha.h com o nome do tipo e os protótipos.

```
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);
```

Implementação do TAD Pilha

- Podemos implementar a interface pilha 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

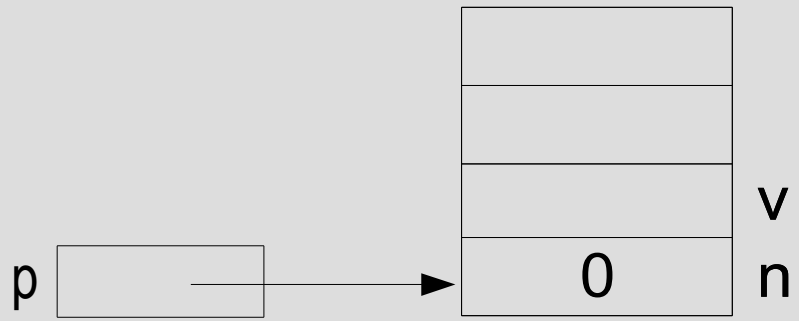
- A estrutura que representa o tipo pilha deve ser composto pelo vetor e pelo número de elementos armazenados

```
#define MAX 3

typedef struct pilha{
   int n;
   int v[MAX];
}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--;
```

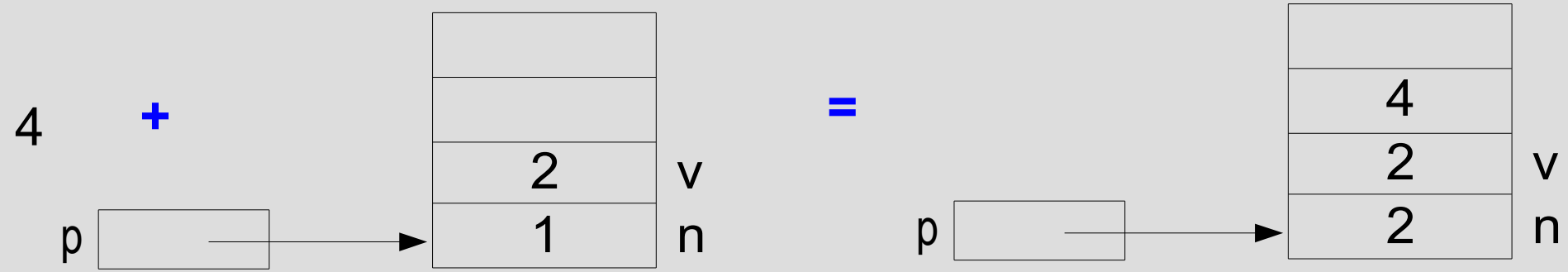
Implementação de Pilha com Vetor – Função Cria Pilha

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



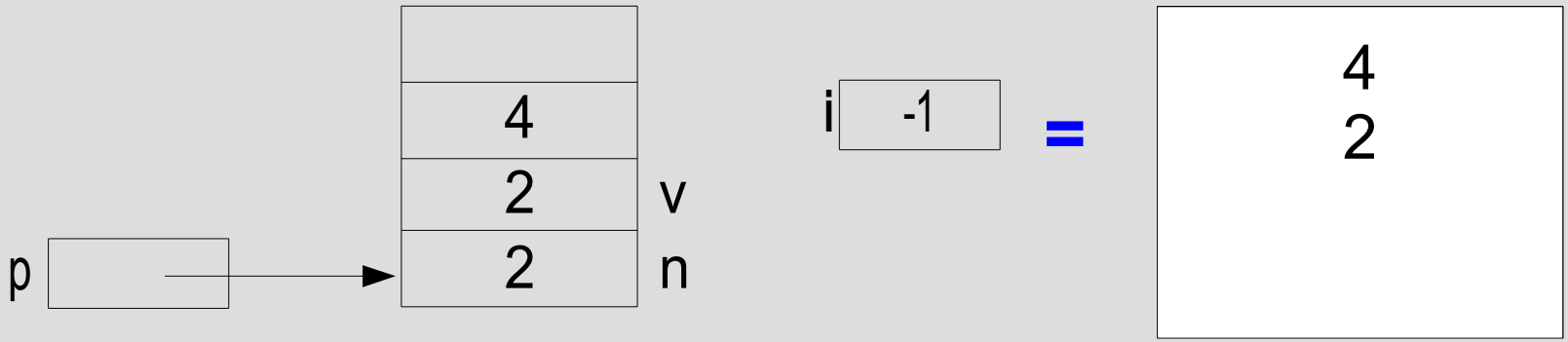
Implementação de Pilha com Vetor – Função *Push*

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



Implementação de Pilha com Vetor Função Imprime e Libera

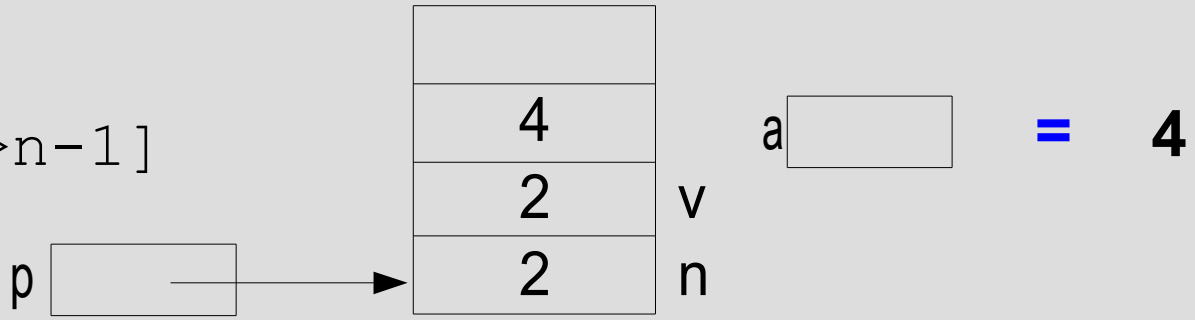
```
/*Função que imprime os elementos de uma pilha.*/
void pilha_imprime(Pilha *p){
    int i;
    for(i = p->n-1; i>=0;i--){
       printf("%f\n",p->v[i]);
    }
}
```




```
/*Libera o espaço alocado para uma pilha.*/
void pilha_libera(Pilha *p){
    free(p);
}
```

Implementação de Pilha com 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--;
    return a;
}
```

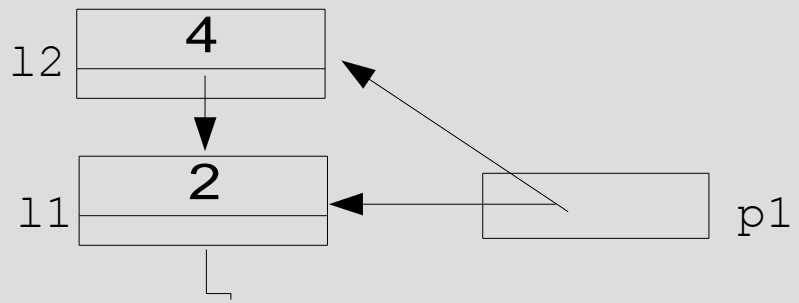


Implementação de Pilha com Lista Encadeada

- A estrutura que representa o tipo pilha é o primeiro elemento de uma lista.

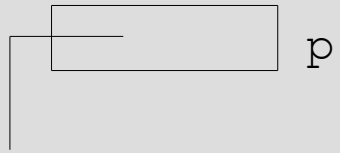
```
typedef struct lista Lista;
typedef struct pilha Pilha;

struct lista{
   int info;
   Lista *prox;
};

struct pilha{
   Lista *prim;
};
```

```
Pilha p1;

Lista l1; l1.info=2; l1.prox=NULL
p1.prim = &l1;

Lista l2; l2.info=4;
l2.prox=&l1; p1.prim=&l2;
```



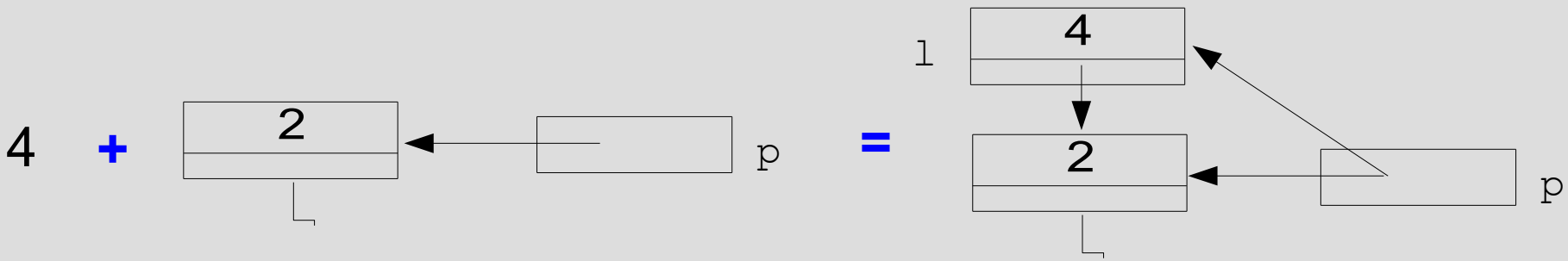
Implementação de Pilha com Lista – Função Cria Pilha

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



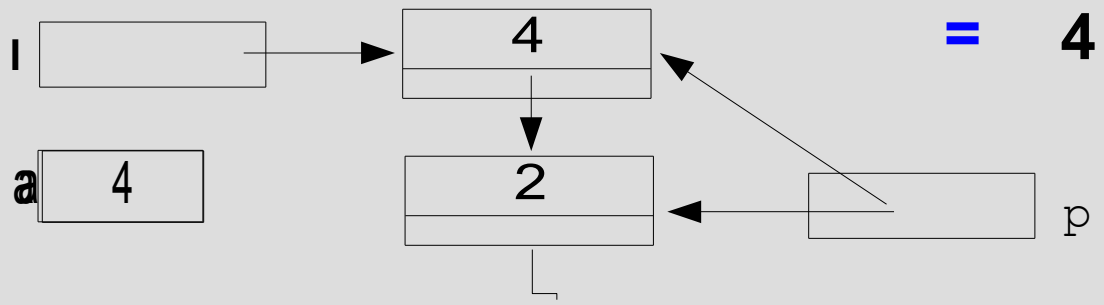
Implementação de Pilha com Lista – Função *Push*

```
/*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;
}
```



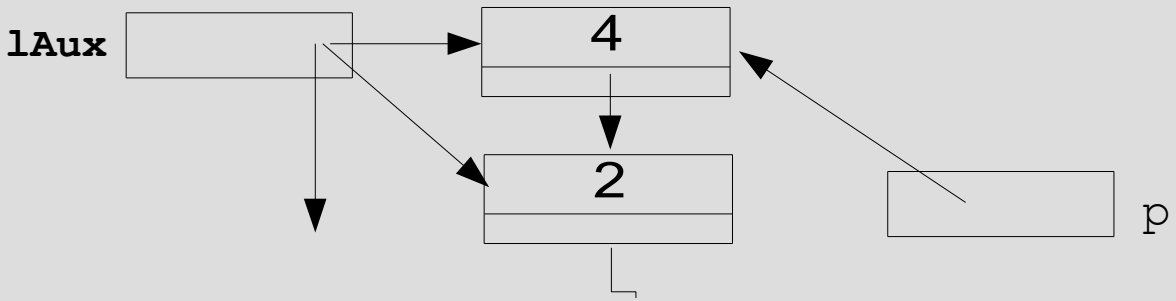
Implementação de Pilha com Lista – Função Vazia e *Pop*

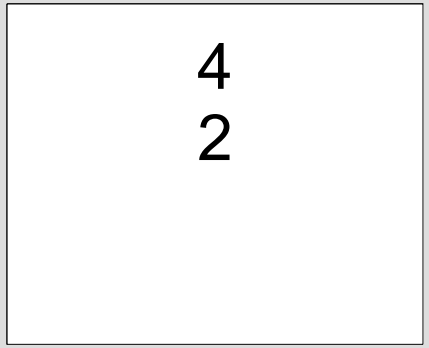
```
/*Testa se uma pilha é vazia.*/
int pilha_vazia(Pilha *p){
    return p->prim==NULL;
}

/*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
Função 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
Função Imprime

```
/*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);
}
```