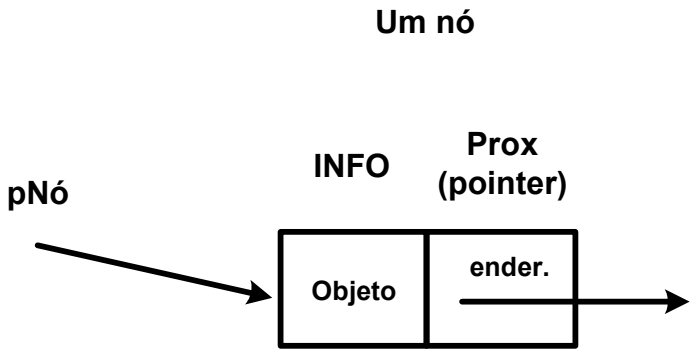
Pilha (stack) encadeada (linked)

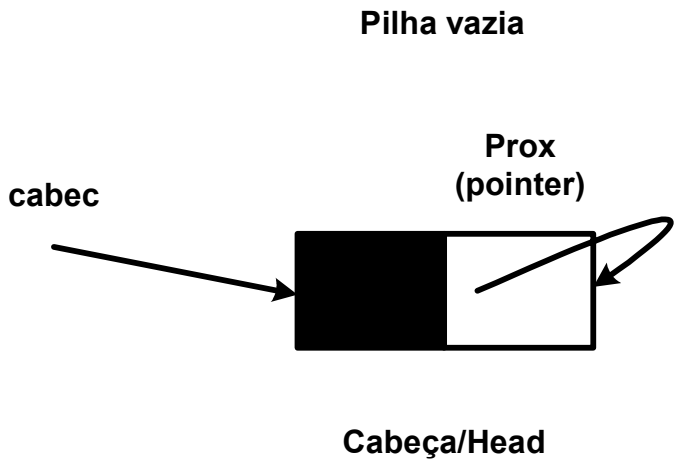
Uma pilha é representada por uma sequência de nós.



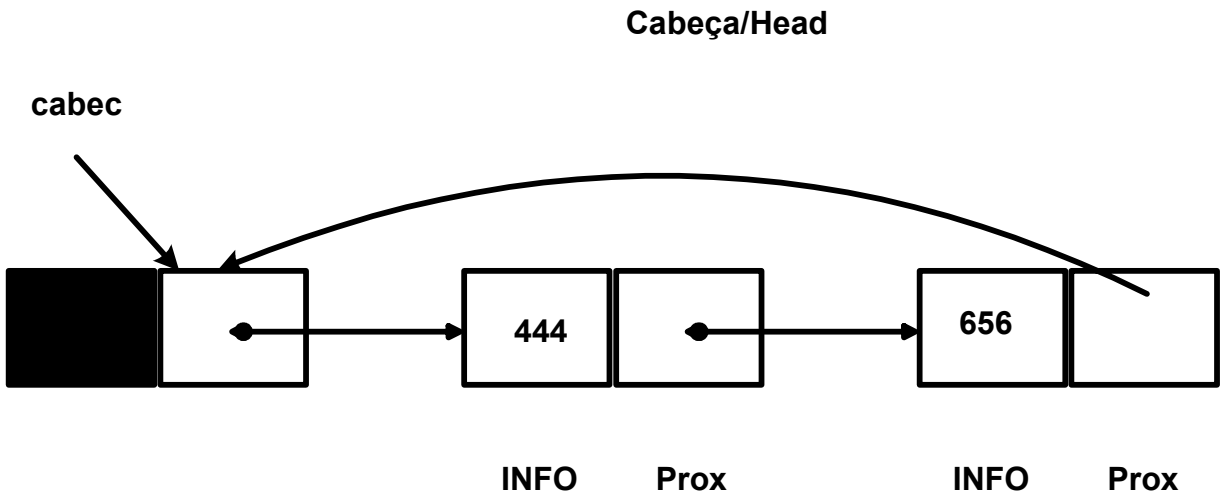
Cada nó (na memória) possui no mínimo dois campos:

1. Um campo para armazenar um objeto, INFO;
2. Outro campo para armazenar o endereço (link) do próximo nó na pilha, Prox.

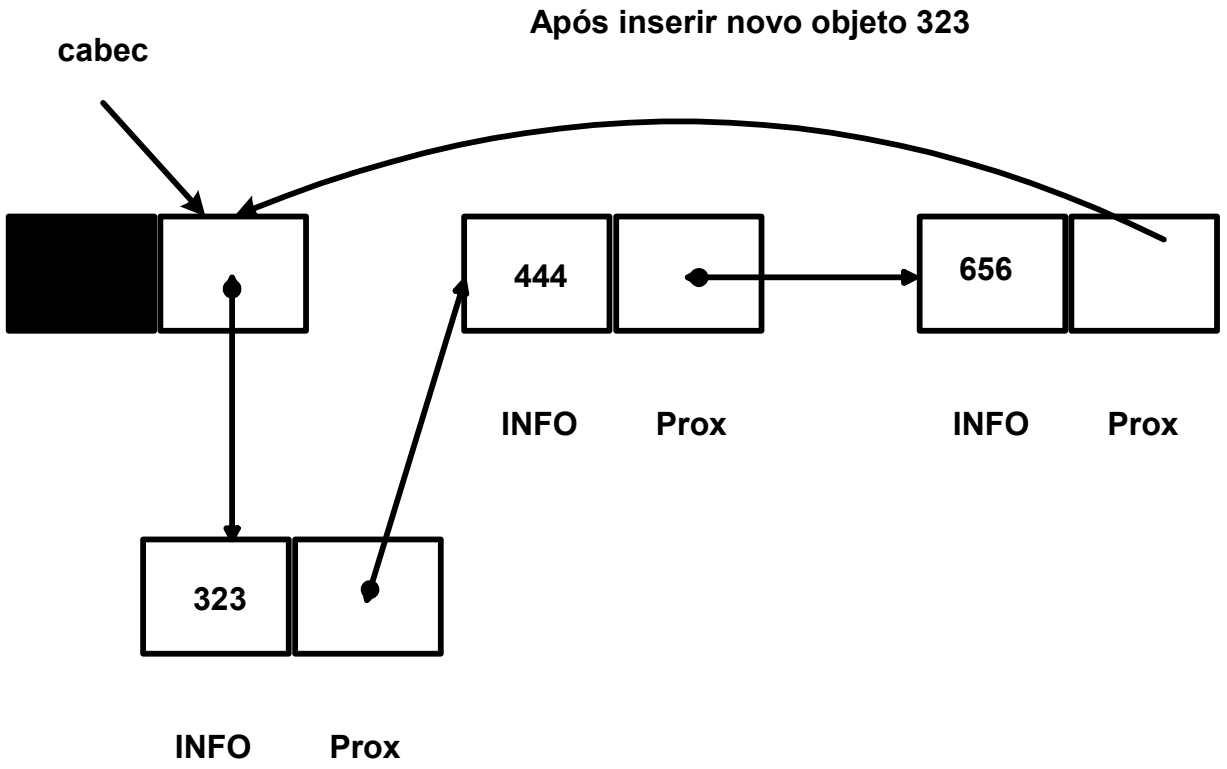
Convenção: pilha vazia é representada por um nó especial, chamado "cabeça" (ou "head") cujo campo Prox armazena o endereço do próprio nó, conforme ilustração a seguir. Deve haver um apontador para este nó, que na ilustração chama-se cabec.



Após empilhar dois objetos (no caso, dois inteiros), primeiro o 656, e depois o 444, a estrutura resultante é ilustrada a seguir.


Cabeça/Head
cabec
444
656
INFO
Prox
INFO
Prox

Após empilhar um objeto novo, o 323, a pilha fica como abaixo.



Observação: não é necessário estimar previamente o número máximo de objetos a serem empilhados

Algoritmo PilhaVazia?
Entrada: apontador cabec
Saída: mensagem "está vazia" (código -1) ou "não está vazia" (código 1)
se cabec.Prox = cabec então mensagem "está vazia";
senão mensagem "não está vazia";

Algoritmo Empilha (push-down)
Entrada: apontador cabec, objeto O a ser empilhado
Saída: mensagem "Overflow", ou mensagem "objeto foi empilhado com sucesso"
observ.: Overflow significa que não há memória disponível para um nó novo para armazenar objeto novo
pNovo = "apontador para memória do tamanho de um nó, da ListaLivre";
se pNovo ≠ NULL então {
pNovo.INFO = O; pNovo.Prox=cabec.Prox;
cabec.Prox = pNovo;
mensagem "objeto foi empilhado com sucesso";
}
senão {mensagem "Overflow"; }

```
Algoritmo Desempilha (pop-up)
Entrada: apontador cabec
Saída: objeto O, ou mensagem "Underflow"
observ.: Underflow significa que não há objeto a ser desempilhado
se cabec.Prox ≠ cabec então {
    pNó = cabec.Prox;  // apontador para topo
   O = pNó.INFO;
    cabec.Prox = pNó.Prox;
   "devolve nó apontado por pNó para a ListaLivre";
   }
senão { mensagem "Underflow";}
```

Exercício:

(1) Escrever algoritmo Último que calcula apontador para o nó na base da pilha.

(2) Escrever algoritmo Tamanho que calcula quantos objetos ocorrem na Pilha.

```
// programa de stack/pilha de
// objetos encadeados em struct (Routo Terada 2010)
// Funcionando com EmpilhaTopo e DesempilhaTopo
#include <stdio.h>
#include <stdlib.h>
// a seguir typedef global
typedef struct nodo{ // typedef aqui com nome
   int INFO;
   struct nodo *Prox; // aponta para proximo
   } NO; // aqui vai nome do tipo que e' NO
//
int PilhaVazia(NO *cabec){ // pilha vazia ?
   if(cabec->Prox == cabec){ // Cabeca da pilha
```

```
   printf("Pilha estaa vazia\n");
   return -1;
   }
   else {printf("Pilha nao estaa vazia\n"); return 1;}
}
```

```
NO * EmpilhaTopo(NO *cabec, int O){ // objeto O
   NO *pNovo; // a seguir, reserva memo da Lista Livre
   pNovo = (NO *) malloc(sizeof(NO));
   if(pNovo != NULL){
   pNovo -> INFO = O;
   pNovo -> Prox = cabec->Prox;
   cabec -> Prox = pNovo;
   printf("Dentro de EmpilhaTopo, INFO = %d\n",
   pNovo -> INFO);
   return cabec;
   }
   else {return NULL;} // Lista Livre esgotada
}
```

```
int DesempilhaTopo(NO *cabec){ // Desempilha objeto O
   NO *pTopo; int O;
   if(cabec -> Prox != cabec){
   pTopo = cabec->Prox;
   O = pTopo -> INFO;
   cabec -> Prox = pTopo -> Prox;
   printf("Dentro de DesempilhaTopo, INFO = %d\n", O);
   free(pTopo); // devolve memo para a Lista Livre
   return O;
   }
   else {printf("Tentou desempilhar pilha vazia\n");
   return -999;}
}
```

```
void ImprPilha(NO *cabec){ // exibe todos os objetos
   NO *pNo; int O;
   pNo = cabec -> Prox; // aponta para o objeto no topo
   if(pNo == cabec) printf("Pilha estaa vazia\n");
   else printf("Objetos na Pilha, do topo ate a base:\n");
   while(pNo != cabec){ // enqto não atinge a base da pilha
   O = pNo -> INFO;
   printf(" INFO = %d\n", O);
   pNo = pNo -> Prox; // um no abaixo, na pilha
   }
}
```

```
int main(){
NO *pNovo, *cabec; int Obj;
// definir cabeca da pilha
cabec = (NO *) malloc(sizeof(NO)); // da Lista Livre
cabec -> INFO = -9999;
cabec -> Prox = cabec;
printf("INFO = %d\n", cabec -> INFO);
if(PilhaVazia(cabec) == -1)
   printf("Verificou que a pilha estaa vazia\n");
cabec = EmpilhaTopo(cabec, 656);
ImprPilha(cabec);
cabec = EmpilhaTopo(cabec, 444);
ImprPilha(cabec);
```

```
cabec = EmpilhaTopo(cabec, 323 );
ImprPilha(cabec);
Obj = DesempilhaTopo(cabec);
printf("Desempilhou %d\n", Obj);
ImprPilha(cabec);
system("pause");
return 0;
}
```

Saída a seguir:
INFO = -9999
Pilha estaa vazia
Verificou que a pilha estaa vazia
Dentro de EmpilhaTopo, INFO = 656
Objetos na Pilha, do topo ate a base:
 INFO = 656
Dentro de EmpilhaTopo, INFO = 444
Objetos na Pilha, do topo ate a base:
 INFO = 444
 INFO = 656
Dentro de EmpilhaTopo, INFO = 323
Objetos na Pilha, do topo ate a base:

INFO = 323
INFO = 444
INFO = 656
Dentro de DesempilhaTopo, INFO = 323
Desempilhou 323
Objetos na Pilha, do topo ate a base:
INFO = 444
INFO = 656