Melhores momentos

AULA 10

Listas

Ilustração de uma lista encadeada "sem cabeça"

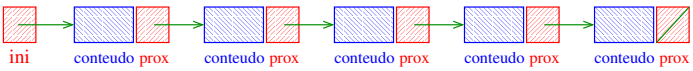
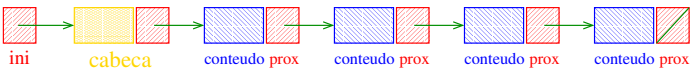


Ilustração de uma lista encadeada "com cabeça"

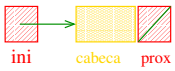


Estrutura de uma lista

```
struct celula {
   int conteudo;
   struct celula *prox;
 };
typedef struct celula Celula;

Celula *ini;
/* inicialmente a lista esta vazia */
ini = NULL;
```

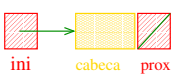
Estrutura de uma lista com cabeça

```
struct celula {
   int conteudo;
   struct celula *prox;
 };
typedef struct celula Celula;

Celula *ini, cabeca;
/* inicialmente a lista esta vazia */
cabeca.prox = NULL;
ini = &cabeca;
```



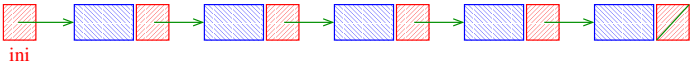
Estrutura de uma lista com cabeça

```
struct celula {
   int conteudo;
   struct celula *prox;
 };
typedef struct celula Celula;

Celula *ini;
/* inicialmente a lista esta vazia */
ini = malloc(sizeof(Celula));
ini->prox = NULL;
```



Imprime uma lista

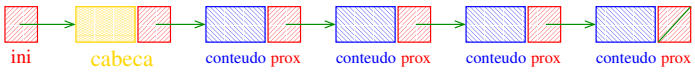
Esta função imprime o conteudo de cada célula de uma lista encadeada ini.




```
void imprima (Celula *ini)
{
   Celula *p;
   for (p=ini; p != NULL; p=p->prox)
       printf("%d\n", p->conteudo);
}
```

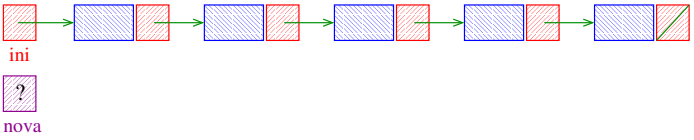
Imprime uma lista com cabeça

Esta função imprime o `conteudo` de cada célula de uma lista encadeada com cabeca `ini`.

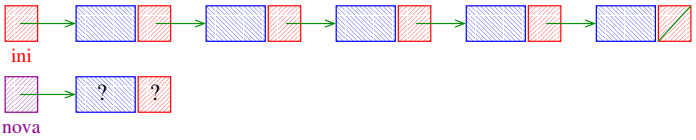



```
void imprima (Celula *ini)
{
   Celula *p;
   for (p=ini->prox; p != NULL; p=p->prox)
       printf("%d\n", p->conteudo);
}
```

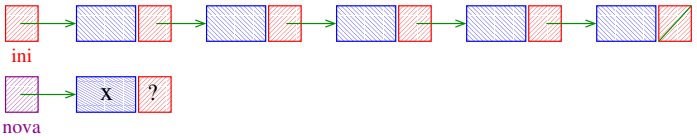
Inserção no início de uma lista



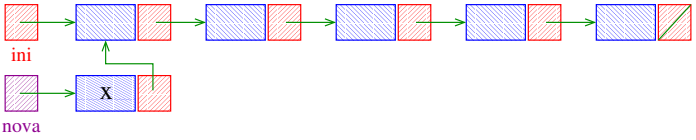
Inserção no início de uma lista



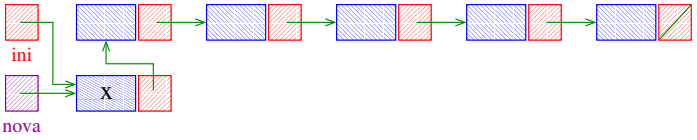
Inserção no início de uma lista



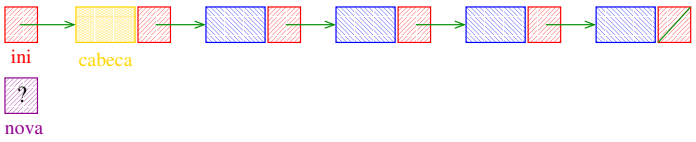
Inserção no início de uma lista



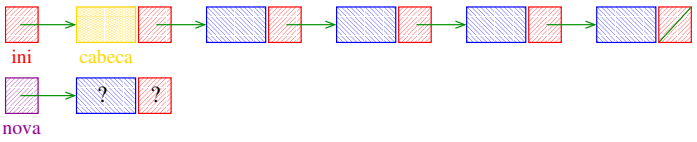
Inserção no início de uma lista



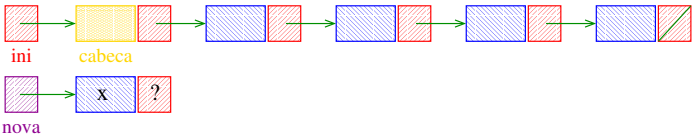
Inserção no início de uma lista com cabeça



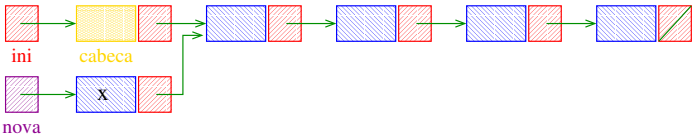
Inserção no início de uma lista com cabeça



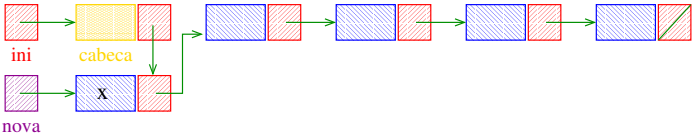
Inserção no início de uma lista com cabeça



Inserção no início de uma lista com cabeça



Inserção no início de uma lista com cabeça



AULA 11

Inversão de uma lista

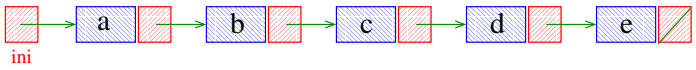
PF 4, S 3.3

http://www.ime.usp.br/~pf/algoritmos/aulas/lista.html

Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando **apenas os ponteiros** (!).

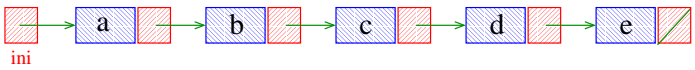
Lista antes da inversão:



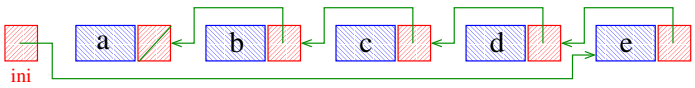
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando **apenas os ponteiros** (!).

Lista antes da inversão:



Lista depois da inversão:



Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.

```
Celula *inverta(Celula *ini) {
  Celula *p, *q, *r;
  p = NULL; q = ini;
  while (q != NULL) {
      r = q->prox;
      q->prox = p;
      p = q;
      q = r;
  }
  return p;
}
```

Exemplos de chamadas

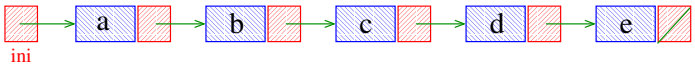
```
Celula *ini, *ini2;
ini = ini2 = NULL;

[...manipulação da lista ...]

ini = inverta(ini);
ini2 = inverta(ini2);
```

Exemplos de chamadas

```
Celula *ini, *ini2;
Celula cabeca;
ini = &cabeca
cabeca.prox = NULL;
ini2 = mallocSafe(sizeof(Celula));
ini2->prox = NULL;
[...manipulação das listas ...]
ini->prox= inverta(ini->prox);
cabeca.prox = inverta(cabeca.prox);
ini->prox = inverta(cabeca.prox);
cabeca.prox = inverta(ini->prox);
ini2->prox= inverta(ini2->prox);
```

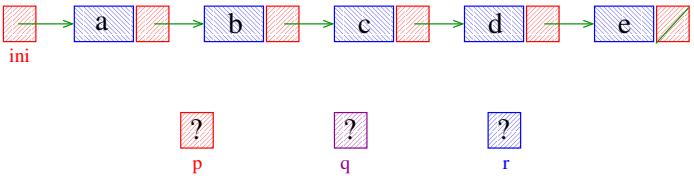
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



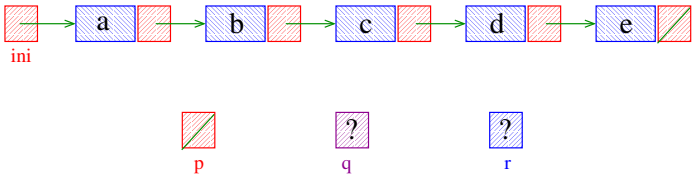
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



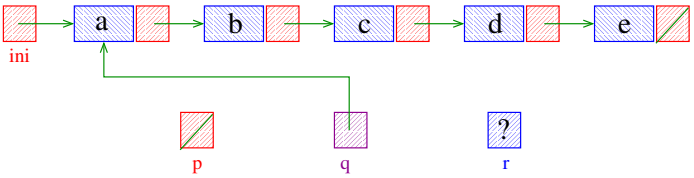
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



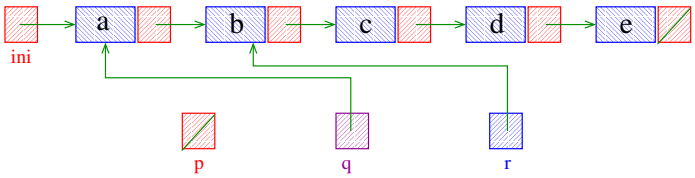
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



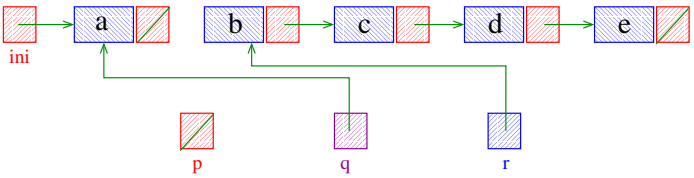
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



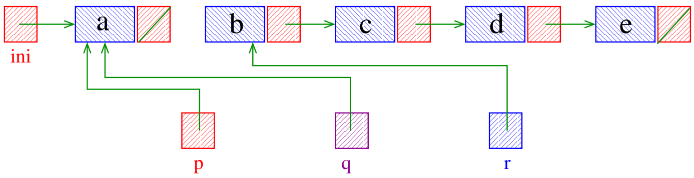
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



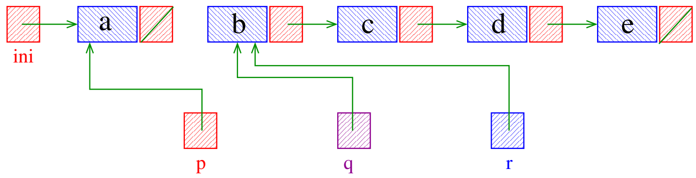
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



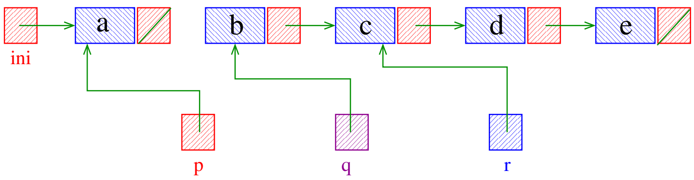
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



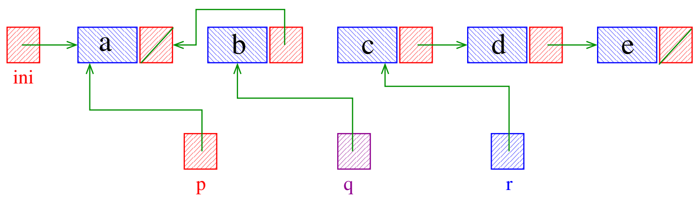
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



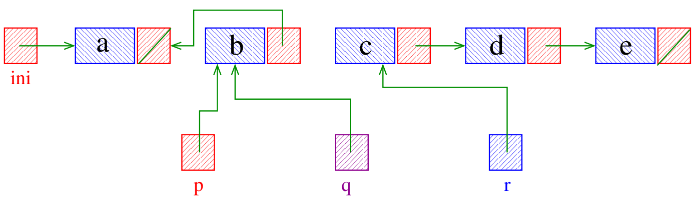
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



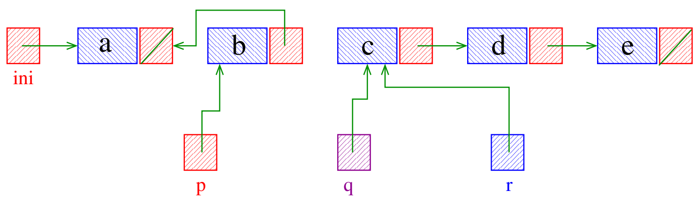
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



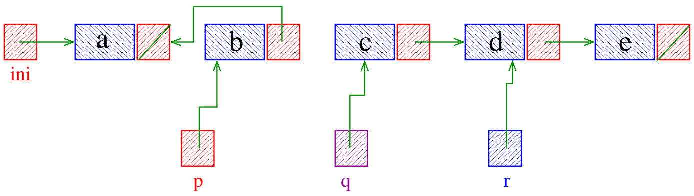
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



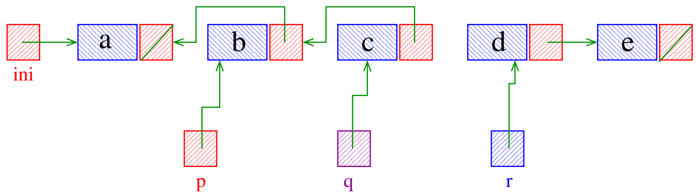
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



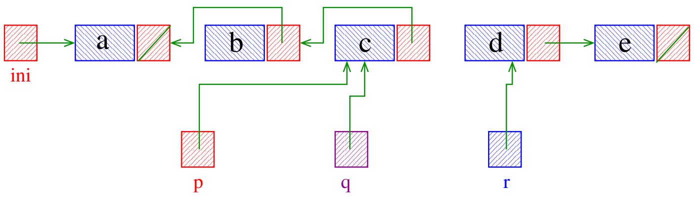
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



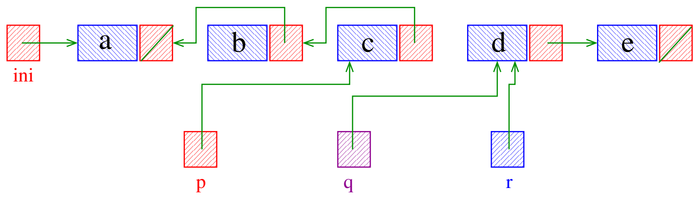
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



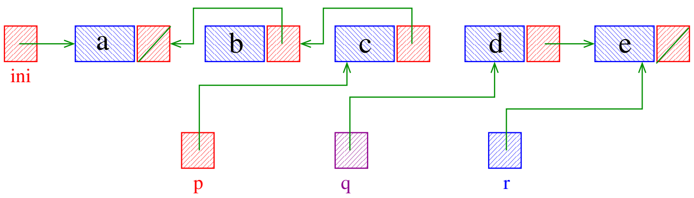
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



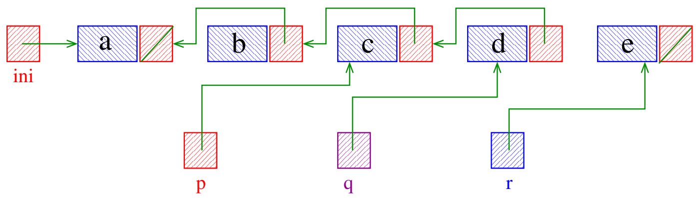
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



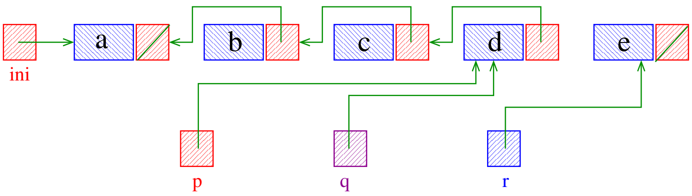
Inversão de uma lista

Recebe uma lista `ini` e inverte a ordem de suas células alterando apenas os ponteiros.



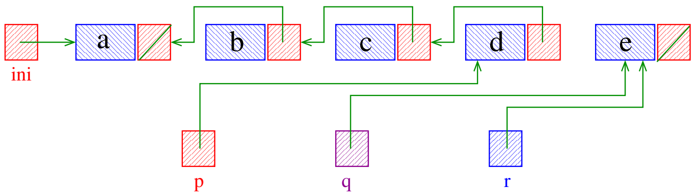
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



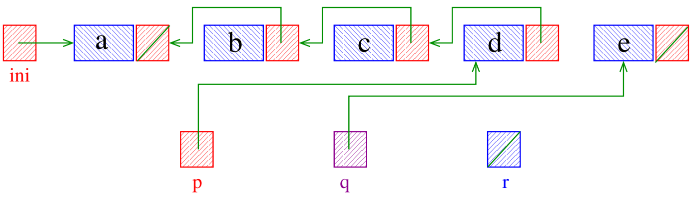
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



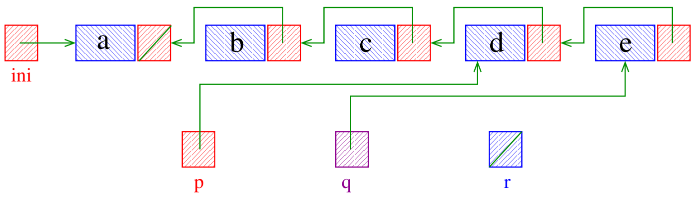
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



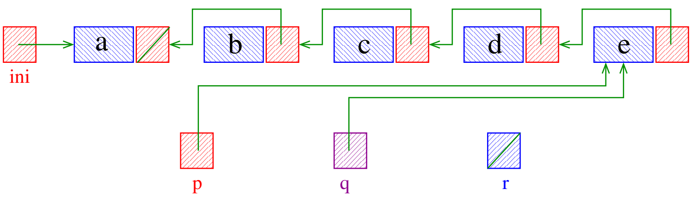
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



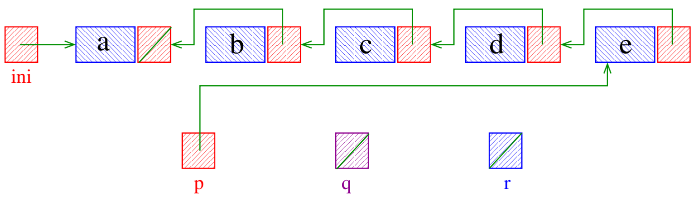
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



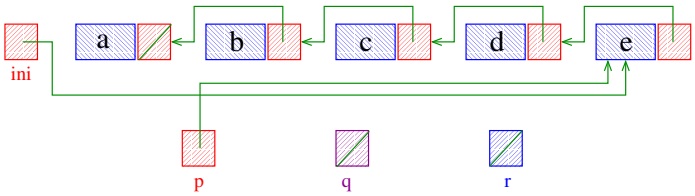
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



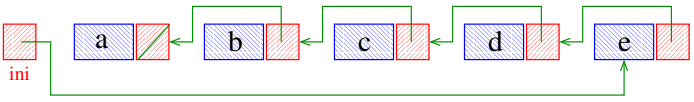
Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.



Inversão de uma lista

Recebe uma lista ini e inverte a ordem de suas células alterando apenas os ponteiros.

```
void inverta(Celula **ini) {
  Celula *p, *q, *r;
  p = NULL; q = *ini;
  while (q != NULL) {
      r = q->prox;
      q->prox = p;
      p = q;
      q = r;
  }
  *ini= p;
}
```

Exemplos de chamadas

```
Celula *ini, *ini2;
ini = ini2 = NULL;

[...manipulação da lista ...]

inverta(&ini);
inverta(&ini2);
```

Exemplos de chamadas

```
Celula *ini, *ini2;
Celula cabeca;
ini = &cabeca
cabeca.prox = NULL;
ini2 = mallocSafe(sizeof(Celula));
ini2->prox = NULL;

[...manipulação das listas ...]

inverta(&ini->prox);
inverta(&cabeca.prox);
inverta(&cabeca.prox);
inverta(&ini->prox);
inverta(&ini2->prox);
```

Pilhas

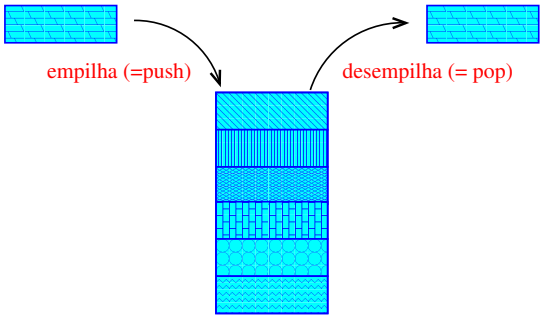
PF 6.1 e 6.3

http://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html

http://en.wikipedia.org/wiki/RPN_calculator

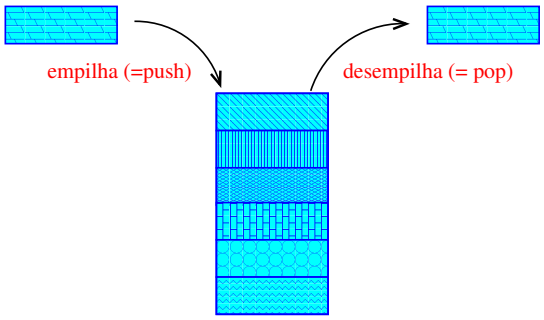
Pilhas

Uma **pilha** (=*stack*) é uma lista (=sequência) dinâmica em que todas as operações (inserções, remoções e consultas) são feitas em uma mesma extremidade chamada de **topo**.



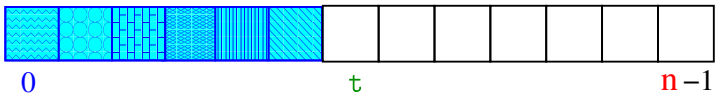
Pilhas

Assim, o primeiro objeto a ser removido de uma pilha é o último que foi inserido. Esta política de manipulação é conhecida pela sigla **LIFO** (=*Last In First Out*)



Implementação em um vetor

A pilha será armazenada em um vetor `s[0..n-1]`.



O índice `t` indica o topo (=*top*) da pilha.

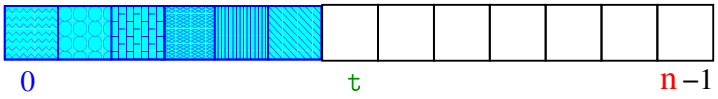
Esta é a primeira posição vaga da pilha.

A pilha está vazia se "`t == 0`".

A pilha está cheia se "`t == n`".

Implementação em um vetor

A pilha será armazenada em um vetor `s[0..n-1]`.



Para remover (=desempilhar=*pop*) um elemento faça

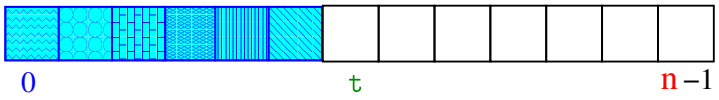
```
x = s[--t];
```

que é equivalente a

```
t -= 1;
x = s[t];
```

Implementação em um vetor

A pilha será armazenada em um vetor `s[0..n-1]`.



Para inserir (=empilhar=*push*) um elemento faça

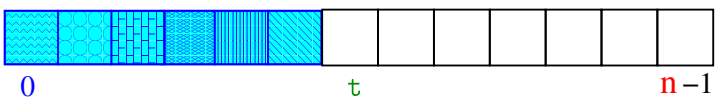
```
s[t++] = x;
```

que é equivalente a

```
s[t] = x;
t += 1;
```

Implementação em um vetor

A pilha será armazenada em um vetor `s[0..n-1]`.

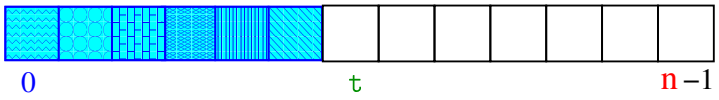


Para consultar um elemento, sem removê-lo, faça

```
x = s[t-1];
```

Implementação em um vetor

A pilha será armazenada em um vetor `s[0..n-1]`.



Tentar desempilhar de uma pilha que está vazia é um erro chamado *stack underflow*

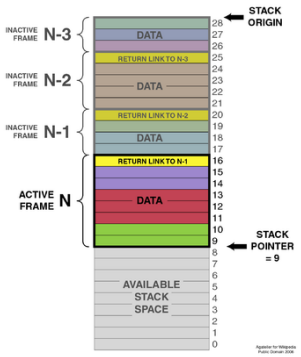
Tentar empilhar em uma pilha cheia é um erro chamado *stack overflow*

Pilha de execução

PF 6.5
http://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html
http://en.wikipedia.org/wiki/Call_stack

Pilha de execução

Para executar um programa, o computador utiliza uma **pilha de execução** (=*execution stack*=*call stack*).



Fonte: http://en.wikipedia.org/wiki/File:ProgramCallStack2.png

Pilha de execuções

Uma pilha de execução é usada para armazenar:

- variáveis locais das funções "ativas";
- parâmetros das funções "ativas";
- endereço de retorno para o ponto em que as funções foram chamadas;
- cálculo de expressões;

Exemplo

```
int main (void) {
   int i, j, k, y;
   i = 111; j = 222; k = 444;
   y = /*1*/ F (i, j, k) /*4*/;
   printf ("%d\n", y);
   return EXIT_SUCCESS;
}
int G (int a, int b) {
   int x;
   x = a + b;
   return x;
}
int F (int i, j, k) {
   int x;
   x = /*2*/ G (i, j) /*3*/;
   x = x + k;
   return x;
}
```

?
?
?
?
?
?
?
?
?
?
?
?
?
?

Saida

	?	
	?	
	?	
	?	
	?	
	?	
	?	
	?	
	?	
y	?	
k	?	
j	?	main
i	?	
end. retorno	010101010	

Saida

	?	
	?	
	?	
	?	
	?	
	?	
	?	
	?	
	?	
y	?	
k	444	
j	222	main
i	111	
end. retorno	010101010	

Saida

	?	
	?	
	?	
	?	
x	?	
k	444	
j	222	F
i	111	
end. retorno	/*4*/	
y	?	
k	444	
j	222	main
i	111	
end. retorno	010101010	

Saida

x	?	
b	222	G
a	111	
end. retorno	/*3*/	
x	?	
k	444	
j	222	F
i	111	
end. retorno	/*4*/	
y	?	
k	444	
j	222	main
i	111	
end. retorno	010101010	

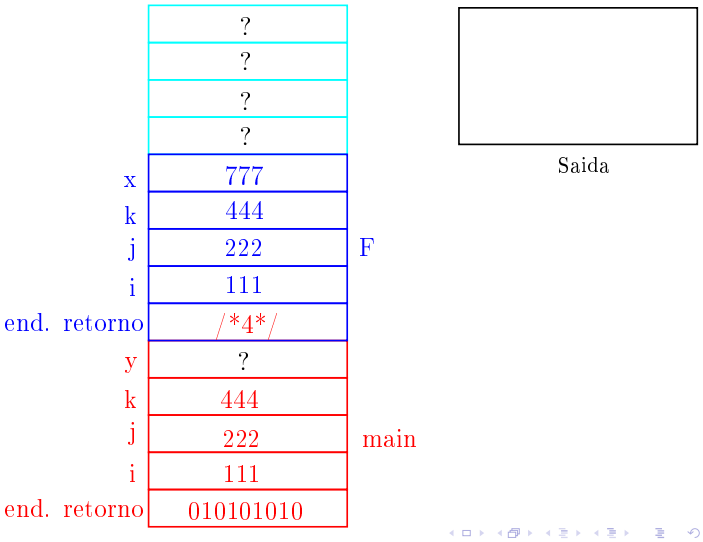
Saida

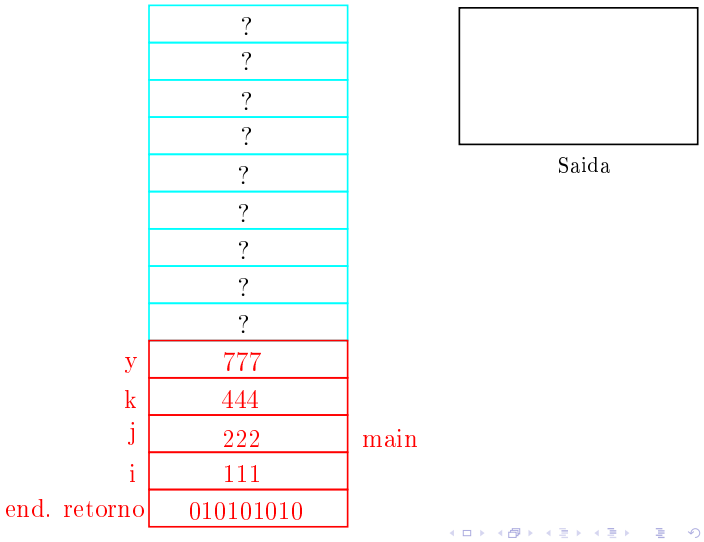
x	333	
b	222	G
a	111	
end. retorno	/*3*/	
x	?	
k	444	
j	222	F
i	111	
end. retorno	/*4*/	
y	?	
k	444	
j	222	main
i	111	
end. retorno	010101010	

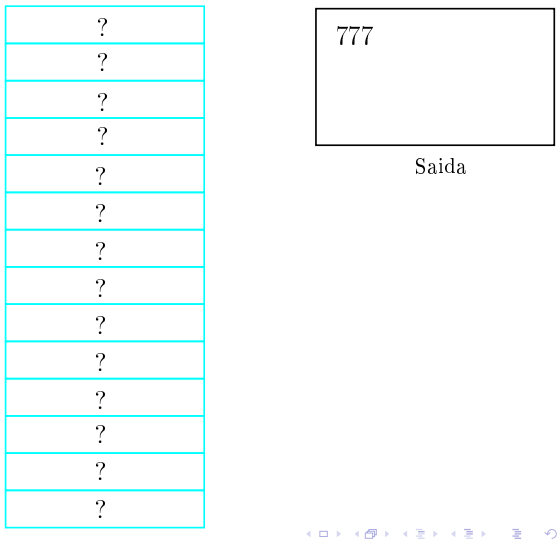
Saida

	?	
	?	
	?	
	?	
x	333	
k	444	
j	222	F
i	111	
end. retorno	/*4*/	
y	?	
k	444	
j	222	main
i	111	
end. retorno	010101010	

Saida







Exercício

Qual o **erro** na função a seguir?

```
CelPixel *
inserir(CelPixel *p, int a, int b)
{
  CelPixel novo;
  novo.x = a;
  novo.y = b;
  novo.prox = p;
  return &novo;
}
```

Aviso

```
meu_prompr> gcc -Wall -ansi -O2 -pedantic
...
In function 'inserir':
warning:  function returns address of
local variable [enabled by default]
```