

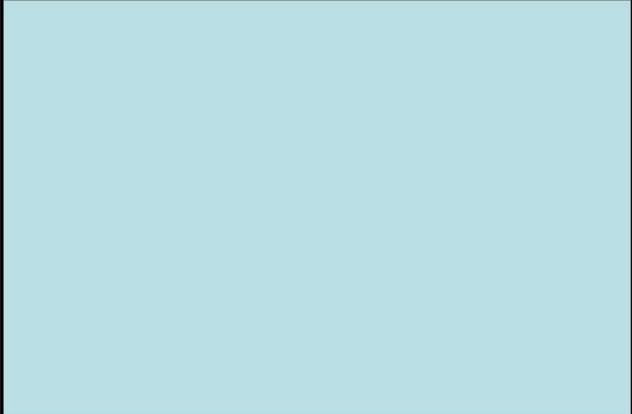
Pilha - Alocação Sequencial

Utilizando-se dos TAD's `FILA_SEQ` e `PILHA_SEQ`, vistos anteriormente, implemente a seguinte operação:

```
void inverte_fila (FILA_SEQ *f);
```

a qual recebe uma referência para uma fila sequencial de inteiros e inverte a ordem de seus elementos, utilizando-se para isto de uma pilha sequencial.





```
1 void invert_e_fila (FILHA_SEQ *f)
2 {
3     PILHA_SEQ p;
4     cria_pilha(&p);
5     while (!eh_vazia(f))
6         push(&p, cons_ret(f));
7     while (!eh_vazia_pilha(&p))
8         ins(f, top_pop(&p));
9 }
```

Pilhas

Alocação Encadeada

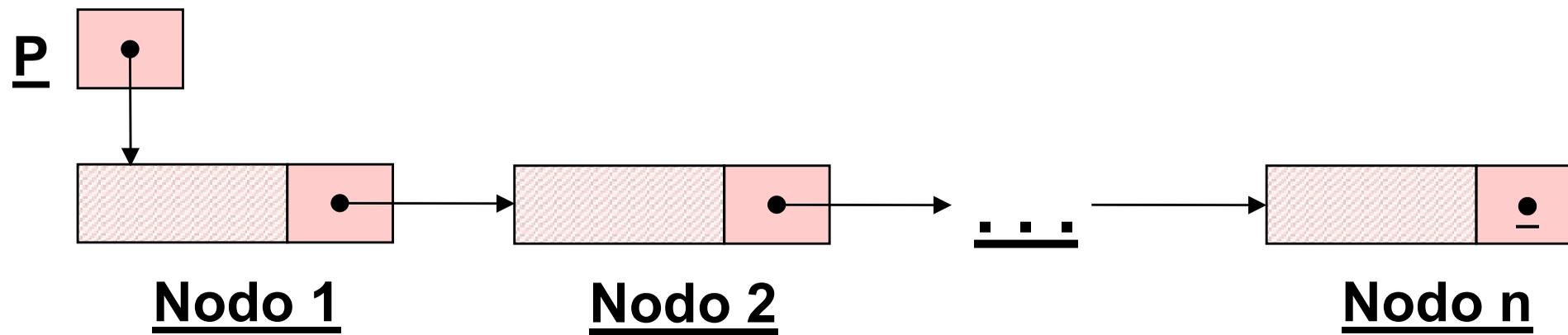
Pilha - Alocação Encadeada

Como já discutimos, a alocação sequencial apresenta algumas desvantagens.

Em virtude disso, podemos nós utilizar de uma lista encadeada para armazenarmos uma pilha, assim como fizemos com as filas.

Como todas as operações ocorrem numa das extremidades da lista, a representação da pilha se reduz a um único ponteiro para o primeiro nodo da lista.

Pilha - Alocação Encadeada



A implementação das operações é trivial.

Desta forma, definiremos e implementaremos, agora, o TAD `PILHA_ENC` (de valores inteiros).

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODE;
6  typedef NODE * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação create() que compõem o TAD

PILHA_ENC.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação `is_empty()` que compõem o TAD `PILHA_ENC`.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação push() que compõem o TAD PILHA_ENC.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação top() que compõem o TAD PILHA_ENC.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação pop() que compõem o TAD PILHA_ENC.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação `top_pop()` que compõem o TAD `PILHA_ENC`.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```

Pilha - Alocação Encadeada

Com base no que foi visto implemente a operação `destroy()` que compõem o TAD `PILHA_ENC`.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * next;
5  }NODO;
6  typedef NODO * PILHA_ENC;
7  void create (PILHA_ENC *);
8  int is_empty (PILHA_ENC);
9  void push (PILHA_ENC *, int);
10 int top (PILHA_ENC);
11 void pop (PILHA_ENC *);
12 int top_pop (PILHA_ENC *);
13 void destroy (PILHA_ENC);
```