CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Pilhas

Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

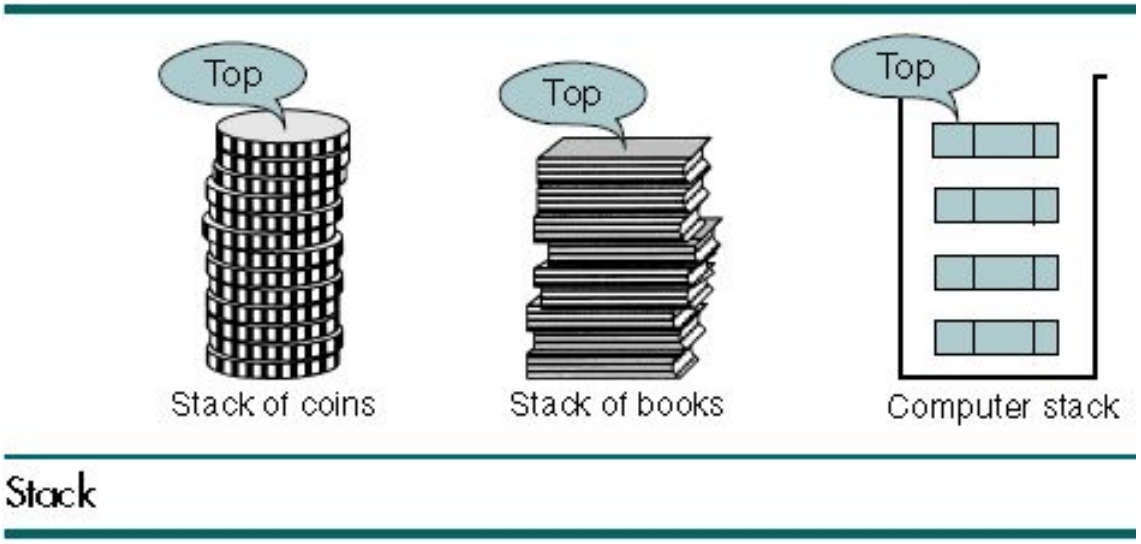
Pilhas (Stacks)
Definição

- Uma pilha (stack) é uma lista linear na qual todas as inserções e remoções ficam limitadas a apenas uma extremidade, chamada de topo (top).
- A inserir uma sequência de dados em uma pilha e depois remover essa sequência, a ordem desses dados será invertida.
- Exemplo: Os dados inseridos na sequência {5, 10, 15, 20} serão removidos como {20, 15, 10, 5}.
- O atributo (ou caraterística) de inversão é motivo pelo qual a pilha é conhecida como uma estrutura de dados LIFO (*Last in – First Out*), último a entrar, primeiro a sair.

- Uma pilha (stack) é uma estrutura de dados LIFO (*Last in – First Out*) em que todas as inserções e remoções estão restritas a um extremo chamado topo.

Pilhas (Stacks)
Exemplos

- O conceito de pilha é familiar. Utilizamos diferentes tipos de pilhas no nosso dia a dia. Por exemplo, falamos de pilhas de pratos, pilhas de livros, pilhas de moedas.
- Qualquer situação em que somente podemos inserir ou remover um objeto no topo é uma pilha.
- Se quisermos remover um objeto diferente do topo, deveremos remover todos os outros objetos acima dele.

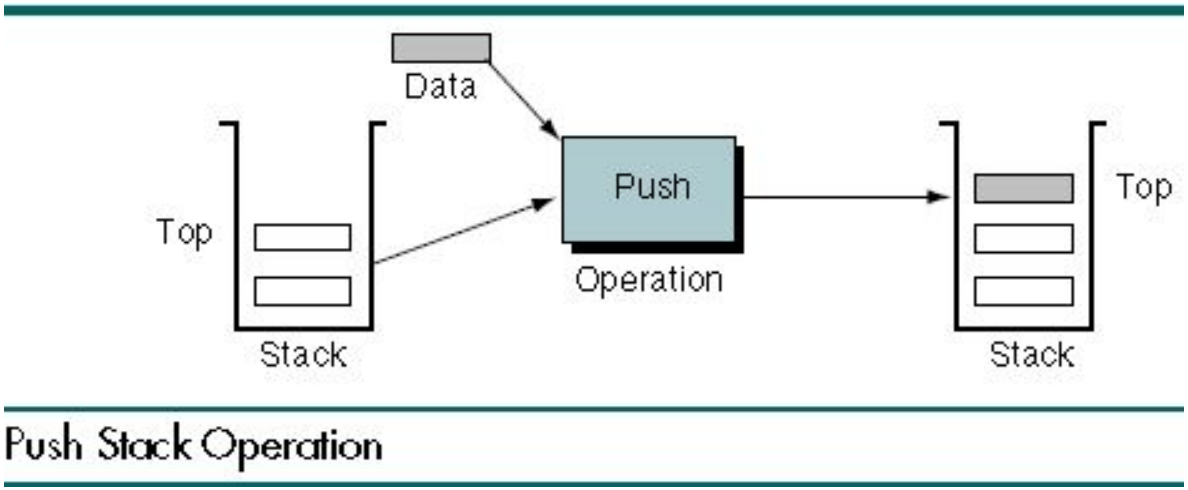


Pilhas (Stacks)
Operações Básicas

- As três operações básicas são:
  - Inserir Pilha (push)
  - Remover Pilha (pop)
  - Topo da Pilha (top)

Pilhas (Stacks)
Operação Inserir Pilha (PushStack)

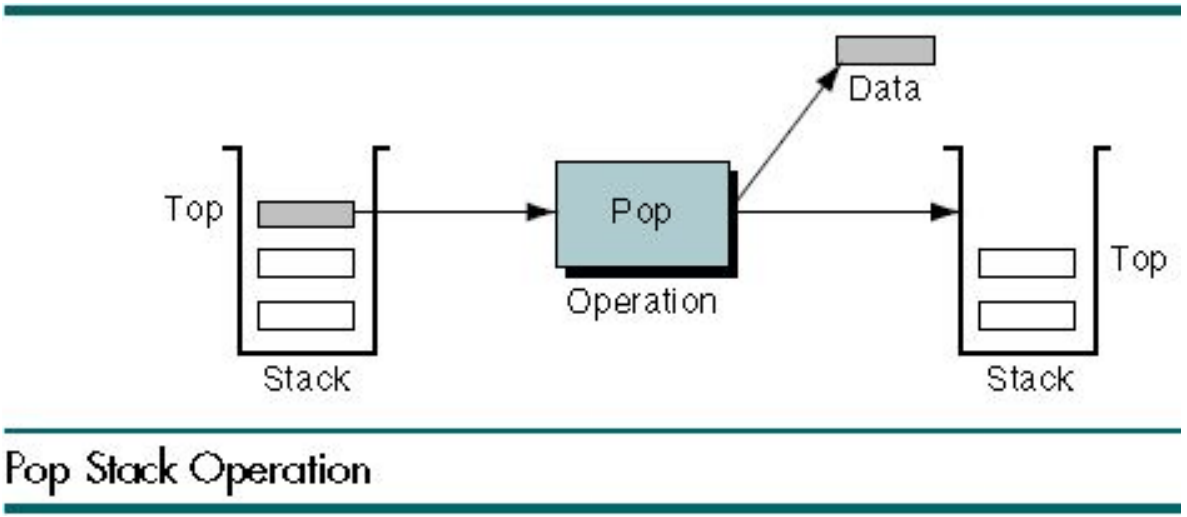
- A operação PushStack:



Push Stack Operation

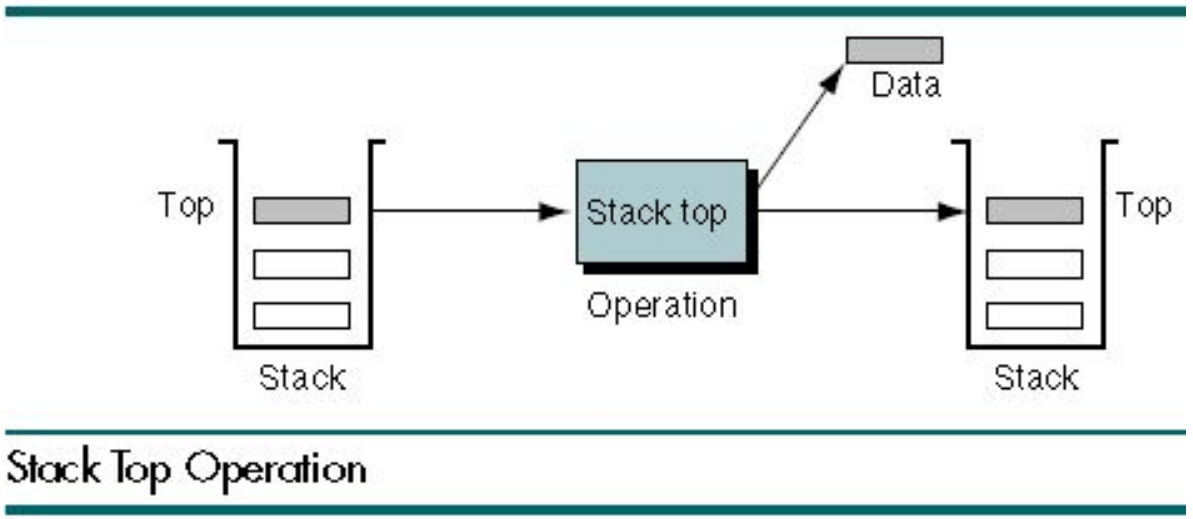
Pilhas (Stacks)
Operação Remover Pilha (PopStack)

- A operação PopStack:

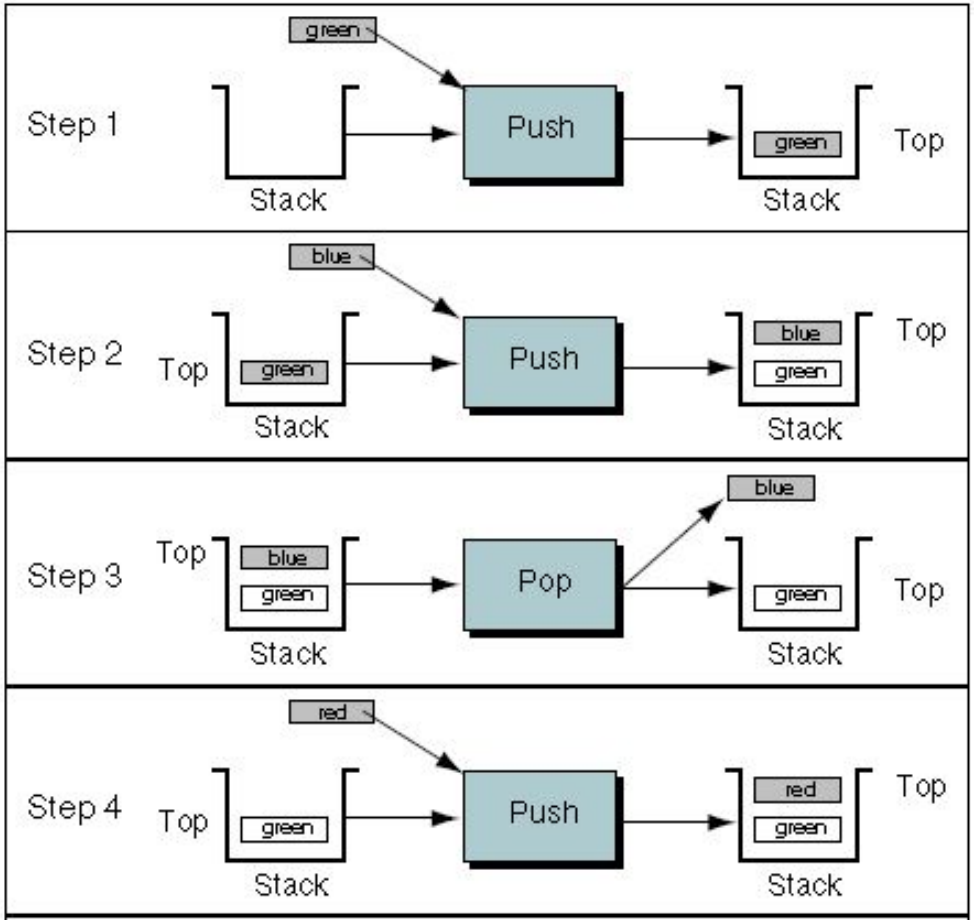


Pilhas (Stacks)
Operação Topo Pilha (StackTop)

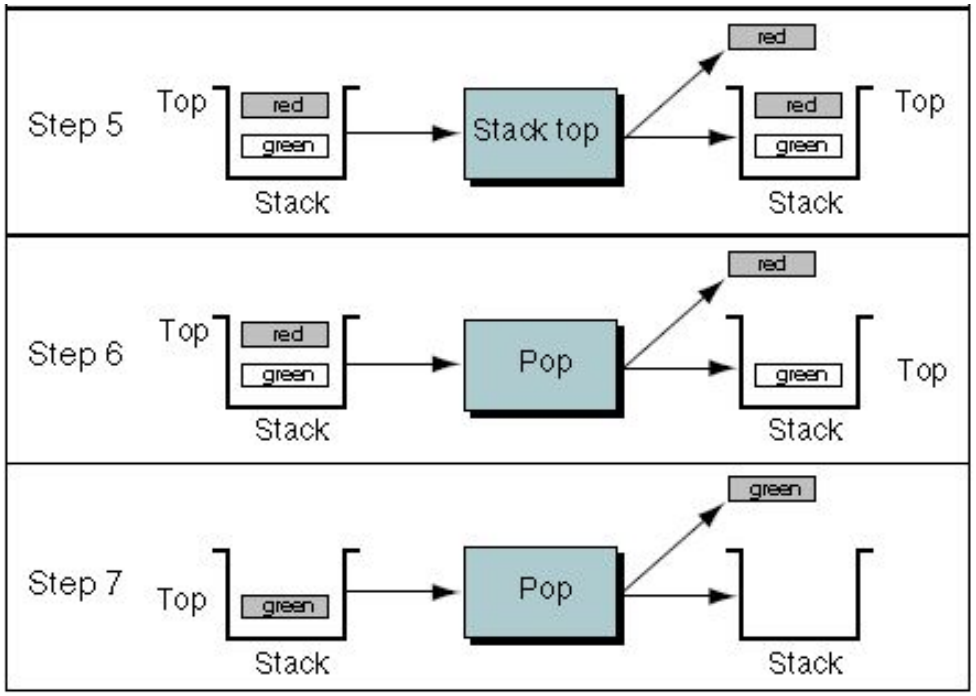
- A operação StackTop:



Pilhas (Stacks)
Exemplo de Operação

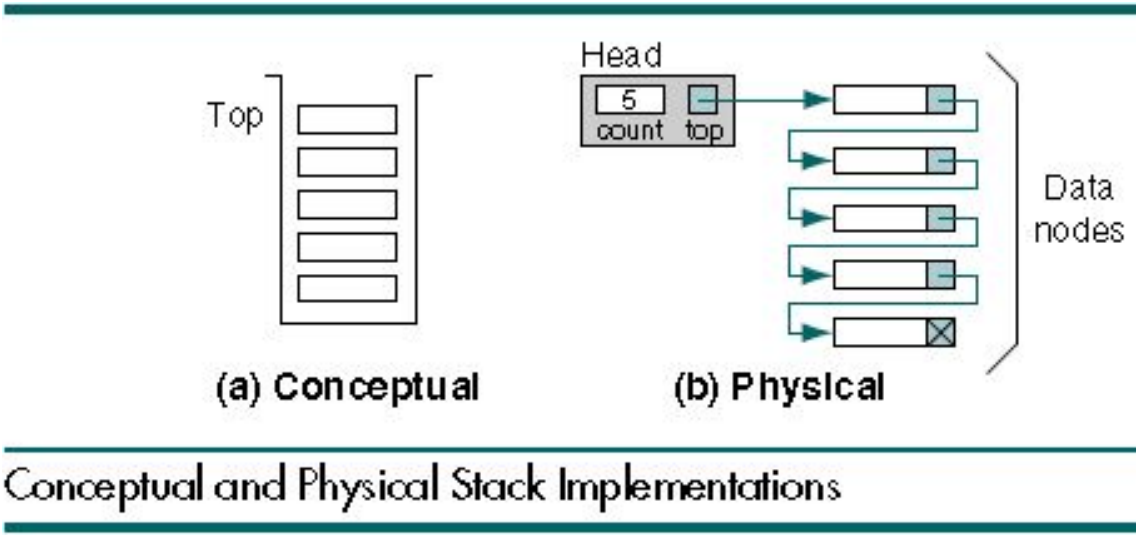


Pilhas (Stacks)
Exemplo de Operação



Pilhas (Stacks)
Implementação como Listas Encadeadas

- Estrutura conceitual e Estrutura Física como Lista Encadeada.

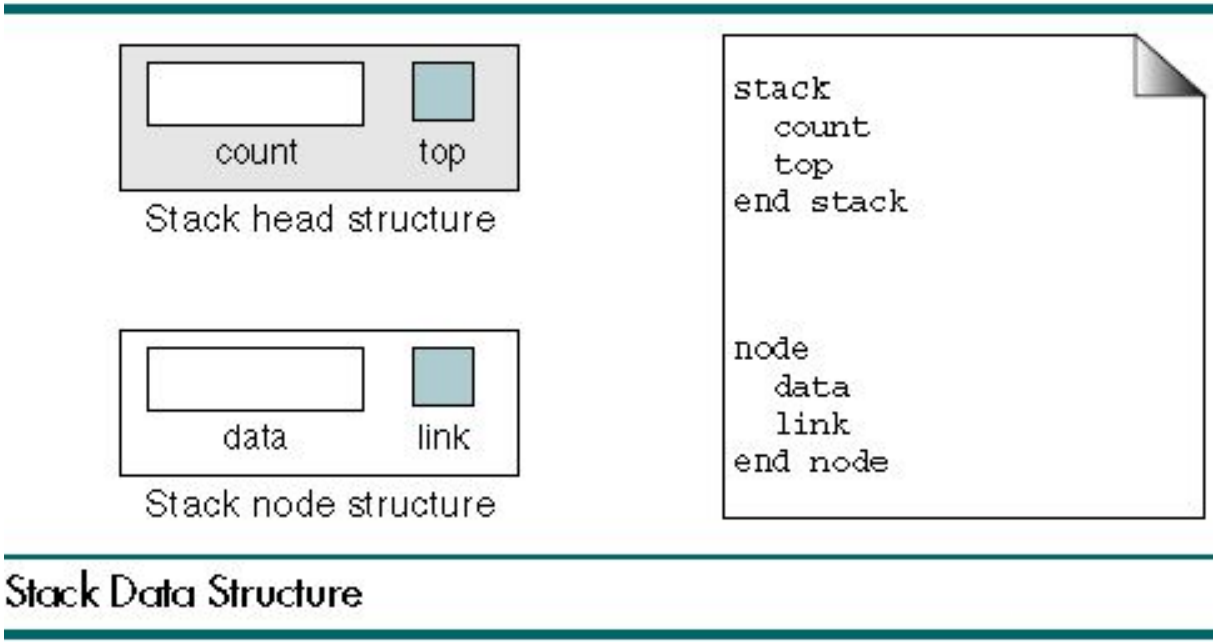


Conceptual and Physical Stack Implementations

Pilhas (Stacks)

Implementação como Listas Encadeadas - Estruturas

- Estruturas de dados de uma Pilha:
  - Cabeçalho da Pilha (Stack Head)
  - Nó da Pilha (Stack node)



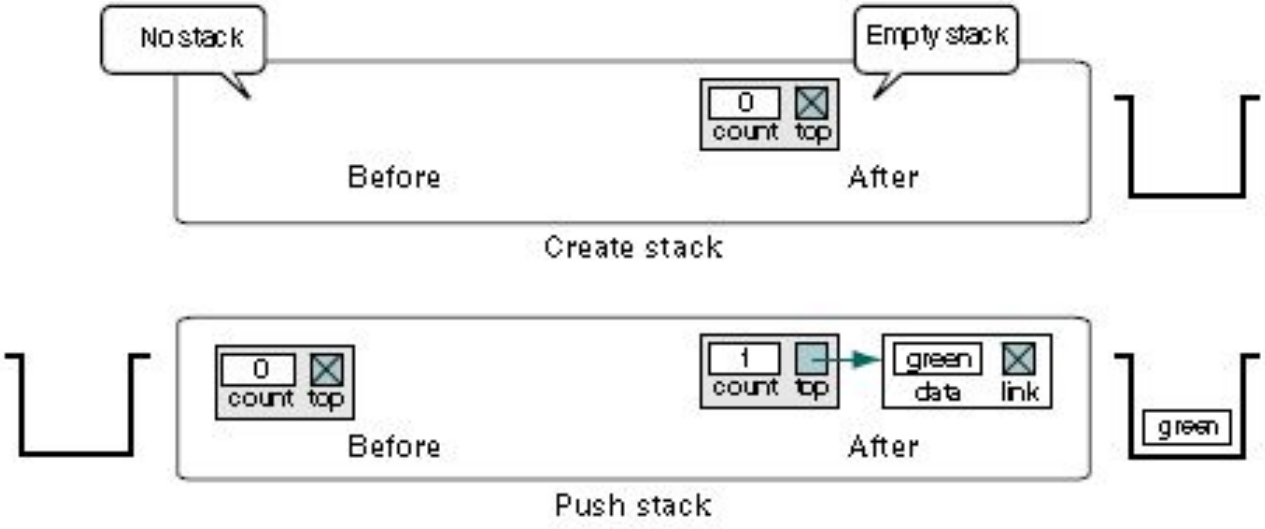
Pilhas (Stacks)

Implementação como Listas Encadeadas - Operações

- As seguinte 8 operações são suficientes para resolver qualquer problema básico com pilhas:
  - Criar Pilha (Create Stack);
  - Inserir Pilha (Push Stack);
  - Remover Pilha (Pop Stack);
  - Topo Pilha (Stack Top);
  - Pilha Vazia (Empty Stack);
  - Pilha Cheia (Full Stack);
  - Contador Pilha (Stack Count);
  - Destruir Pilha (Destroy Stack).

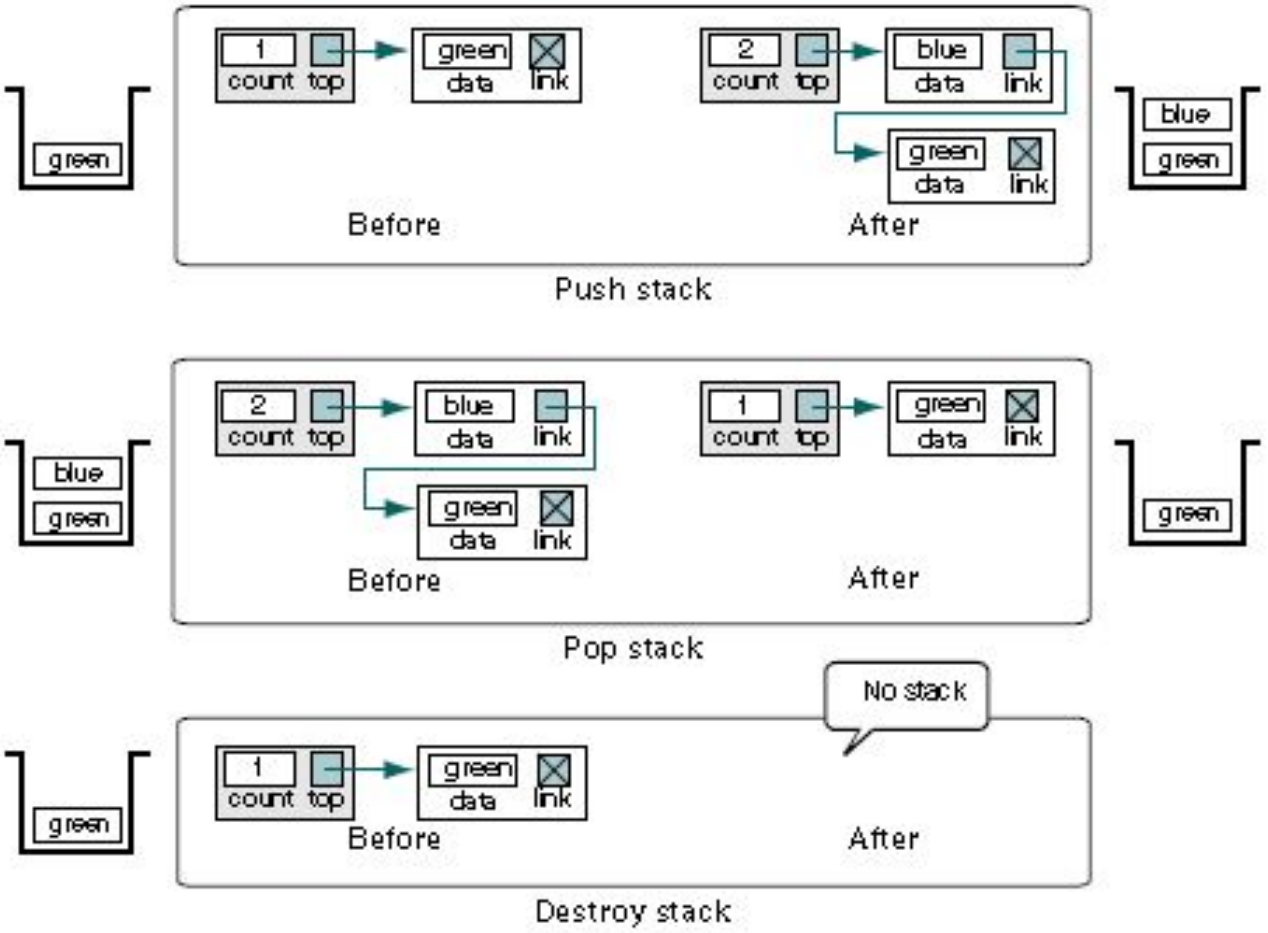
Pilhas (Stacks)
Implementação como Listas Encadeadas - Operações

- As figuras ilustram as 4 operações de pilha mais comuns:
  - Criar Pilha (Create Stack);
  - Inserir Pilha (Push Stack);
  - Remover Pilha (Pop Stack);
  - Destruir Pilha (Destroy Stack).



Pilhas (Stacks)
Implementação como Listas Encadeadas - Operações

- As figuras ilustram as 4 operações de pilha mais comuns:

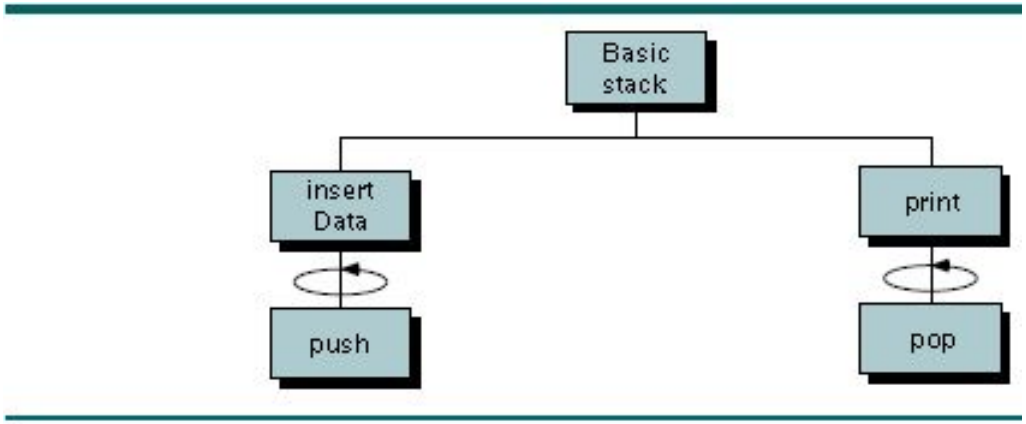


Pilhas (Stacks)
Implementações em Linguagem C

- Como mencionado anteriormente, existem duas abordagens para implementar pilhas.
- Por um lado podemos escrever programas específicos em C, que são mais simples de escrever, porém não serão reutilizáveis.
- Por outro lado, podemos criar um tipo abstrato de dado (TAD) para a pilha, TAD Pilha (Stack ADT), que poderá ser reutilizado com qualquer aplicação de pilha. Teremos o trabalho extra de escrever a aplicação em separado.

Pilhas (Stacks)
Implementação direta – Programa Principal

- O programa mostra como implementar uma pilha que utiliza caracteres em maiúsculas (caixa alta) como elementos.
- Os caracteres são gerados de maneira aleatória e inseridos utilizando a operação **push()**.
- Quando a inserção termina os elementos são impressos para isso eles são removidos um a um, utilizando a operação **pop()**.
- O design do programa básico de pilha é mostrado na figura.



Design for Basic Stack Program

Pilhas (Stacks)

Implementação direta – Programa Principal

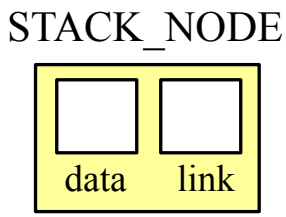
P3-01.c

```
1  /* This program is a test driver to demonstrate the
2     basic operation of the stack push and pop functions.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <stdbool.h>
9
10 // Structure Declarations
11 typedef struct node
12    {
13     char         data;
14     struct node* link;
15    } STACK_NODE;
16
17 // Prototype Declarations
18 void insertData (STACK_NODE** pStackTop);
19 void print      (STACK_NODE** pStackTop);
20
21 bool push       (STACK_NODE** pList, char  dataIn);
22 bool pop        (STACK_NODE** pList, char* dataOut);
```

// Define o tipo STACK_NODE
// Que define a estrutura do nó
Com dado caracter;
Um ponteiro de ligação

Representação de uma Pilha

Implementação direta – Estrutura




```
typedef struct node
{
      char          data;
      struct node*  link;
} STACK_NODE;
```

Pilhas (Stacks)

Implementação direta – Programa Principal

P3-01.c (Continuação...)

```
23
24  int main (void)
25  {
26  // Local Definitions
27  STACK_NODE* pStackTop;
28
29  // Statements
30     printf("Beginning Simple Stack Program\n\n");
31
32     pStackTop = NULL;
33     insertData  (&pStackTop);
34     print       (&pStackTop);
35
36     printf("\n\nEnd Simple Stack Program\n");
37     return 0;
38  }  // main
```

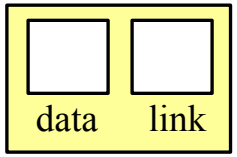
// Define pStackTop como ponteiro
// ao nó da pilha

// Repassa o endereço de pStackTop
// Repassa o endereço de pStackTop

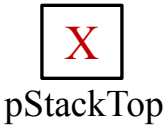
Representação de uma Pilha
Implementação direta – Ponteiro a Pilha

Programa Principal:

STACK_NODE




```
STACK_NODE*
pStackTop;
...
pStackTop=NULL;
```




```
...
insertdata(&pStackTop);
print(&pStackTop);
```

Pilhas (Stacks)
Implementação direta – Resultado

- Esta implementação produz a seguinte saída:

```
Results:
   Beginning Simple Stack Program

   Creating characters: QMZRHLAJOE
   Stack contained:     EOJALHRZMQ

   End Simple Stack Program
```

- O programa gera caracteres de maneira pseudo-aleatória. Cada caráter gerado é impresso e inserido em uma pilha. Os caracteres são apresentados na ordem de criação.
- Posteriormente, cada caracter é extraído da pilha e impresso, obtendo-se assim uma sequência em ordem inversa a ordem de criação.

Pilhas (Stacks)

Implementação direta – Inserir Dados (InsertData)

P3-02.h

```
1  /* ================= insertData =================
2     This program creates random character data and
3     inserts them into a linked list stack.
4        Pre  pStackTop is a pointer to first node
5        Post Stack has been created
6  */
7  void insertData (STACK_NODE** pStackTop)
8  {
9  // Local Definitions
10    char  charIn;
11    bool  success;
12
13 // Statements
14    printf("Creating characters: ");
15    for (int nodeCount = 0; nodeCount < 10; nodeCount++)
16        {
17         // Generate uppercase character
18         charIn  = rand() % 26 + 'A';
19         printf("%c", charIn);
20         success = push(pStackTop, charIn);
21         if (!success)
22            {
23             printf("Error 100: Out of Memory\n");
24             exit (100);
25            } // if
26        } // for
27    printf("\n");
28    return;
29 } // insertData
```

- A função insertData() gera 10 caracteres aleatórios e os insere em uma pilha, passada como parâmetro.

// Define um caracter

// Gera um caracter aleatório

// Chama a operação Push

Pilhas (Stacks)

Implementação direta – Insere (Push)

P3-03.h

```
1  /* ================== push ====================
2     Inserts node into linked list stack.
3        Pre     pStackTop is pointer to valid stack
4        Post    charIn inserted
5        Return  true  if successful
6                false if underflow
7  */
8  bool push (STACK_NODE** pStackTop, char charIn)
9  {
10 // Local Definitions
11    STACK_NODE* pNew;
12    bool        success;
13
14 // Statements
15    pNew = (STACK_NODE*)malloc(sizeof (STACK_NODE));
16    if (!pNew)
17        success = false;
18    else
19       {
20        pNew->data =  charIn;
21        pNew->link = *pStackTop;
22        *pStackTop =  pNew;
23        success = true;
24       } // else
25    return success;
26 } // push
```

A função push() cria um novo nó e o insere na pilha especificada como parâmetro da função. O nó contém um novo caráter.

// Define um ponteiro a um nó da pilha

// Aloca memória para um nó da pilha
// retorna um ponteiro a esse nó

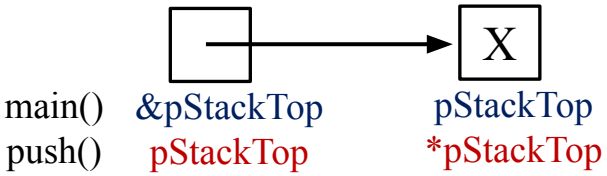
// Copia o caracter no novo nó
// Refaz as ligações

Representação de uma Pilha

Implementação direta – push

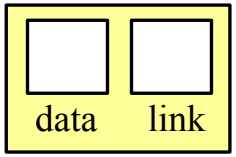
Chamada:

push (pStackTop, charIn);



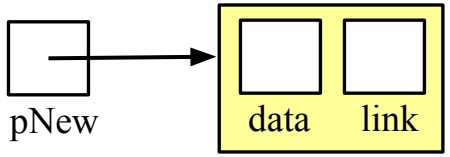
Função push:

STACK_NODE

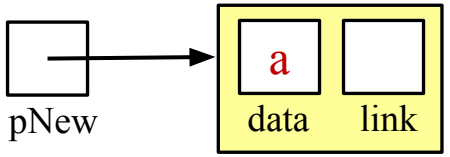


1. STACK_NODE* pNew;

2. pNew=(STACK_NODE*)malloc(sizeof(STACK_NODE));



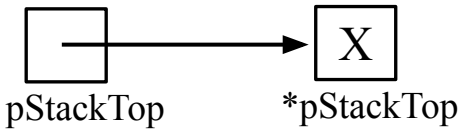
3. pNew->data=charIn;



Representação de uma Pilha

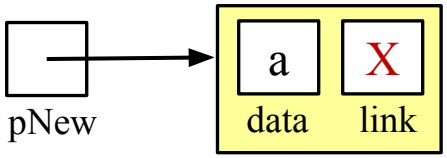
Implementação direta – push

Função push:

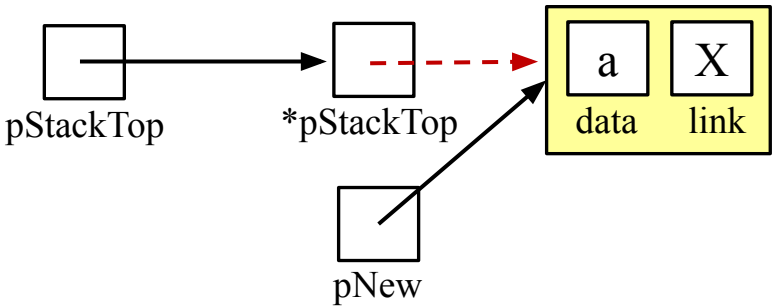


Caso não existem elementos na pilha.

4. pNew->link=*pStackTop;



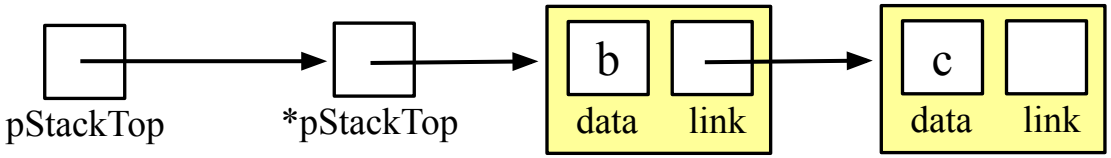
5. *pStackTop=pNew;



Representação de uma Pilha

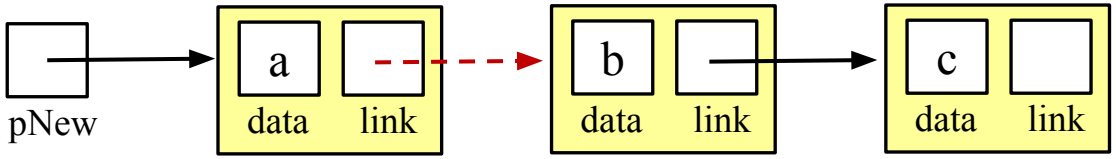
Implementação direta – push

Função push:

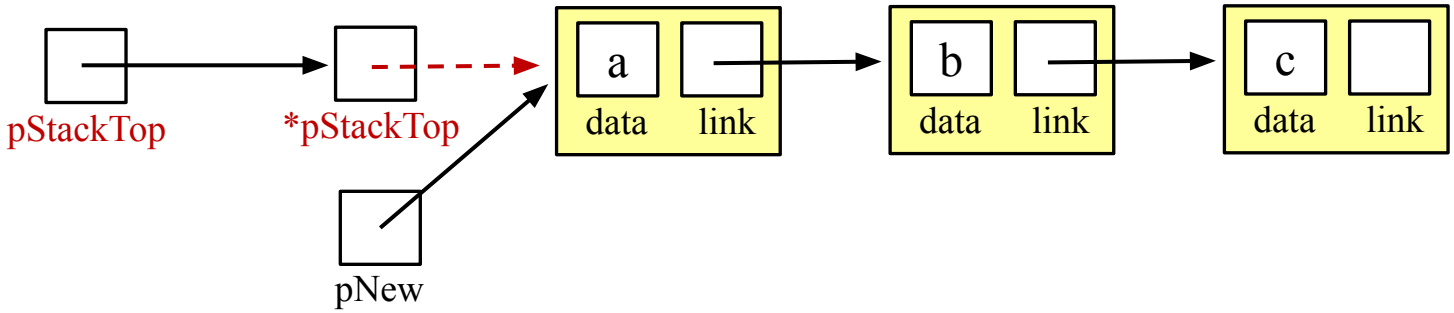


Caso existem elementos na pilha.

4 pNew->link=*pStackTop;



5 *pStackTop=pNew;



Pilhas (Stacks)
Implementação direta – Imprime (Print)

- A função de impressão print() imprime o conteúdo de uma pilha. Ela recebe como parâmetro o endereço de um ponteiro à pilha.
- Consiste de um loop que se repete enquanto for possível extrair o elemento no topo da pilha. Assim, que for extraído o elemento é impresso.

P3-04.h

```
1  /* ===================== print ====================
2     This function prints a singly linked stack.
3        Pre     pStackTop is pointer to valid stack
4        Post    data in stack printed
5  */
6  void print (STACK_NODE** pStackTop)
7  {
8  // Local Definitions
9     char printData;
10
11 // Statements
12    printf("Stack contained:     ");
13    while (pop(pStackTop, &printData))
14        printf("%c", printData);
15    return;
16 }  // print
```

// Define um caracter

// Chama a operação pop
// recupera o caracter removido

Pilhas (Stacks)

Implementação direta – Remove (Pop)

P3-05.h

```
1  /* ================== pop ====================
2     Delete node from linked list stack.
3        Pre  pStackTop is pointer to valid stack
4        Post charOut contains deleted data
5        Return  true  if successful
6                false if underflow
7  */
8  bool pop (STACK_NODE** pStackTop, char* charOut)
9  {
10 // Local Definitions
11    STACK_NODE* pDlt;
12    bool        success;
13
14 // Statements
15    if (*pStackTop)
16       {
17        success    = true;
18        *charOut   = (*pStackTop)->data;
19        pDlt       = *pStackTop;
20        *pStackTop = (*pStackTop)->link;
21        free (pDlt);
22       } // else
23    else
24       success = false;
25    return success;
26 } // pop
```

- A função pop() tenta extrair o caráter contido no primeiro nó da pilha, caso exista. Esse nó é removido da pilha.

// Define um ponteiro a um nó

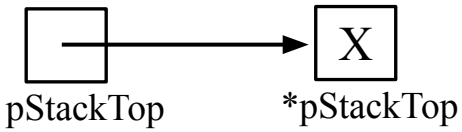
// Se o valor apontado não é nulo

// Salva o carater no topo
// Salva o ponteiro ao topo
// Atualiza o ponteiro ao topo com o seguinte
// Libera memória do nó apontado

Representação de uma Pilha

Implementação direta – pop

Função pop:



Caso não existem elementos na pilha.

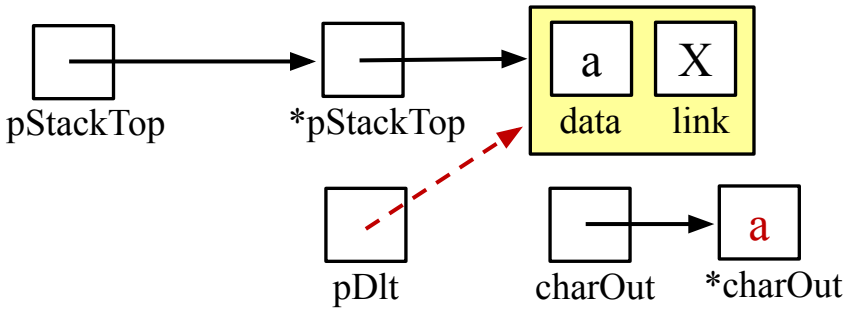
1. sucess=false;

Representação de uma Pilha

Implementação direta – pop

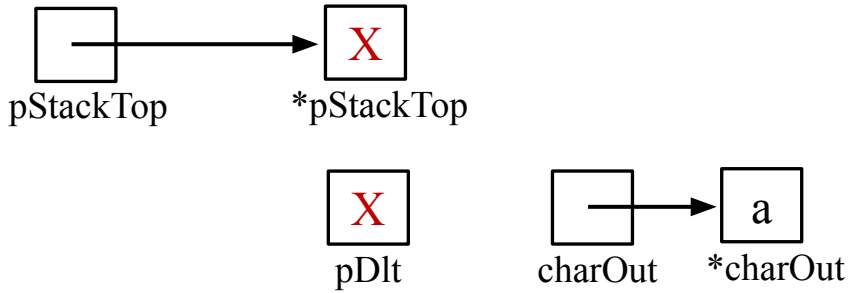
Função pop:

1. sucess=true;
   *charOut=(*pStackTop)->data;
   pDlt=*pStackTop;



Caso exista um elemento na pilha.

2. *pStackTop= (*pStackTop)->link;
   free(pDlt);



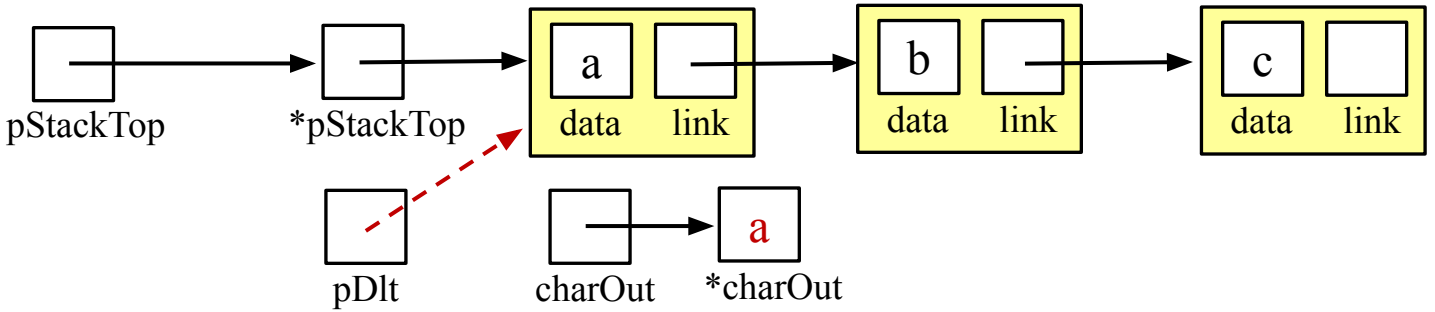
Representação de uma Pilha

Implementação direta – pop

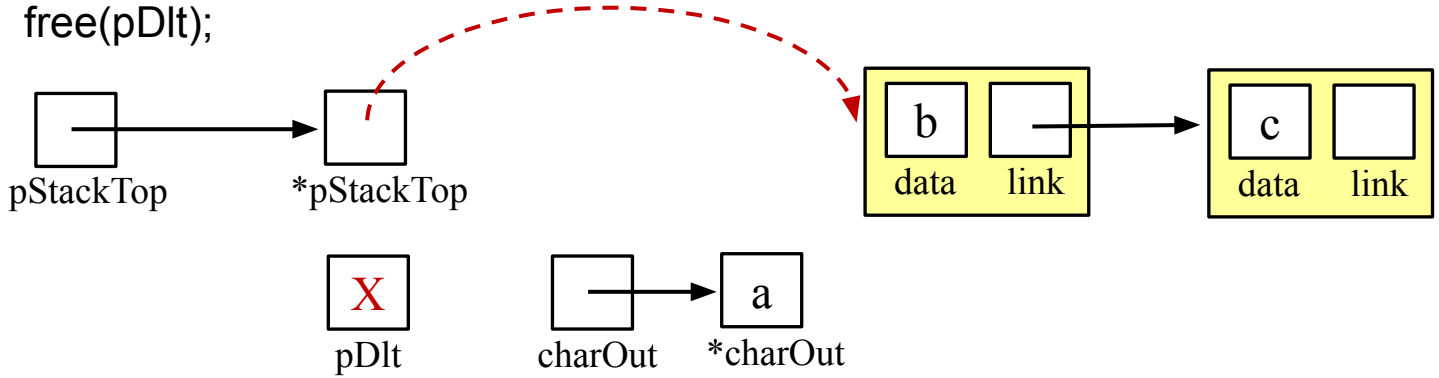
Função pop:

Caso existam vários elementos na pilha.

1. sucess=true;
   *charOut=(*pStackTop)->data;
   pDlt=*pStackTop;



2. *pStackTop= (*pStackTop)->link;
   free(pDlt);



Referências

- Gilberg, R.F. e Forouzan, B.A. Data Structures_A Pseudocode Approach with C. Capítulo 3. Stacks. Segunda Edição. Editora Cengage, Thomson Learning, 2005.