CHƯƠNG 4. NGĂN XẾP

ThS. Nguyễn Chí Hiếu

2021

NỘI DUNG

Giới thiệu ngăn xếp

Ngăn xếp (Stack)

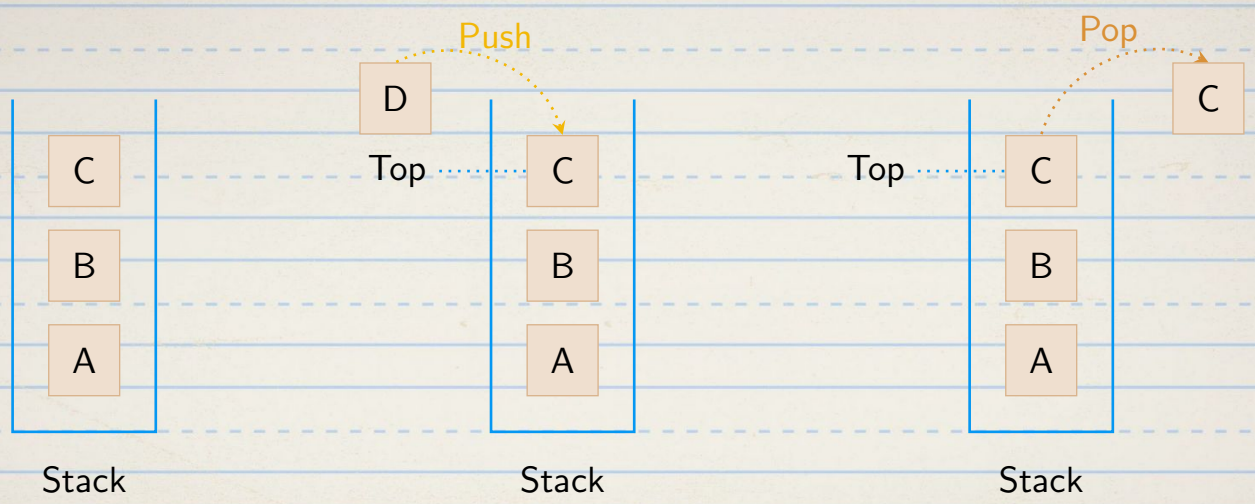
- Thực hiện theo cơ chế *LIFO (Last In, First Out) vào sau ra trước*.
- Dùng để lưu trữ các phần tử có thứ tự truy xuất *ngược* với thứ tự lưu trữ.

Giới thiệu ngăn xếp

Các thao tác cơ bản

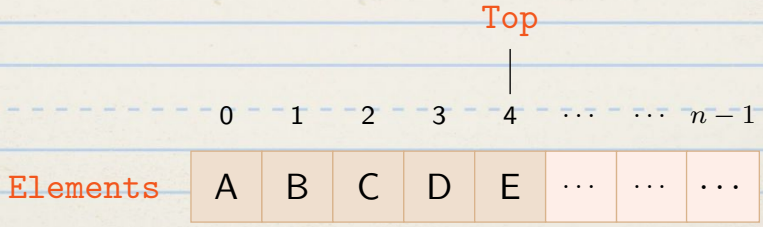
- Push: *thêm* phần tử vào *đỉnh* ngăn xếp.
- Pop: *xóa* phần tử tại *đỉnh* ngăn xếp.
- GetTop: *lấy* phần tử tại *đỉnh* ngăn xếp.
- Kiểm tra ngăn xếp rỗng, đầy.

Giới thiệu ngăn xếp



Cài đặt ngăn xếp bằng mảng

- Một mảng 1 chiều kích thước n: lưu trữ phần tử từ vị trí $[0, ..., n-1]$.
- Một biến top kiểu số nguyên: cho biết vị trí đỉnh ngăn xếp. *Mặc định, ngăn xếp vừa khởi tạo* `top = -1`.



Cài đặt ngăn xếp bằng mảng

```
1  public class Stack
2  {
3      public int[] elements;
4      public int top;
5
6      public void InitStack()
7      {
8          elements = new int[MAX_SIZE];
9          top = -1;
10     }
11 }
```

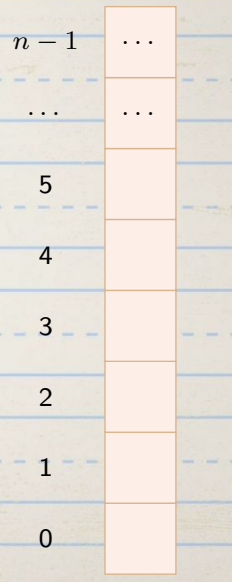

Cài đặt ngăn xếp bằng mảng

Thuật toán 1: IsEmpty(s)
- Đầu vào: ngăn xếp s.
- Đầu ra: true/false.

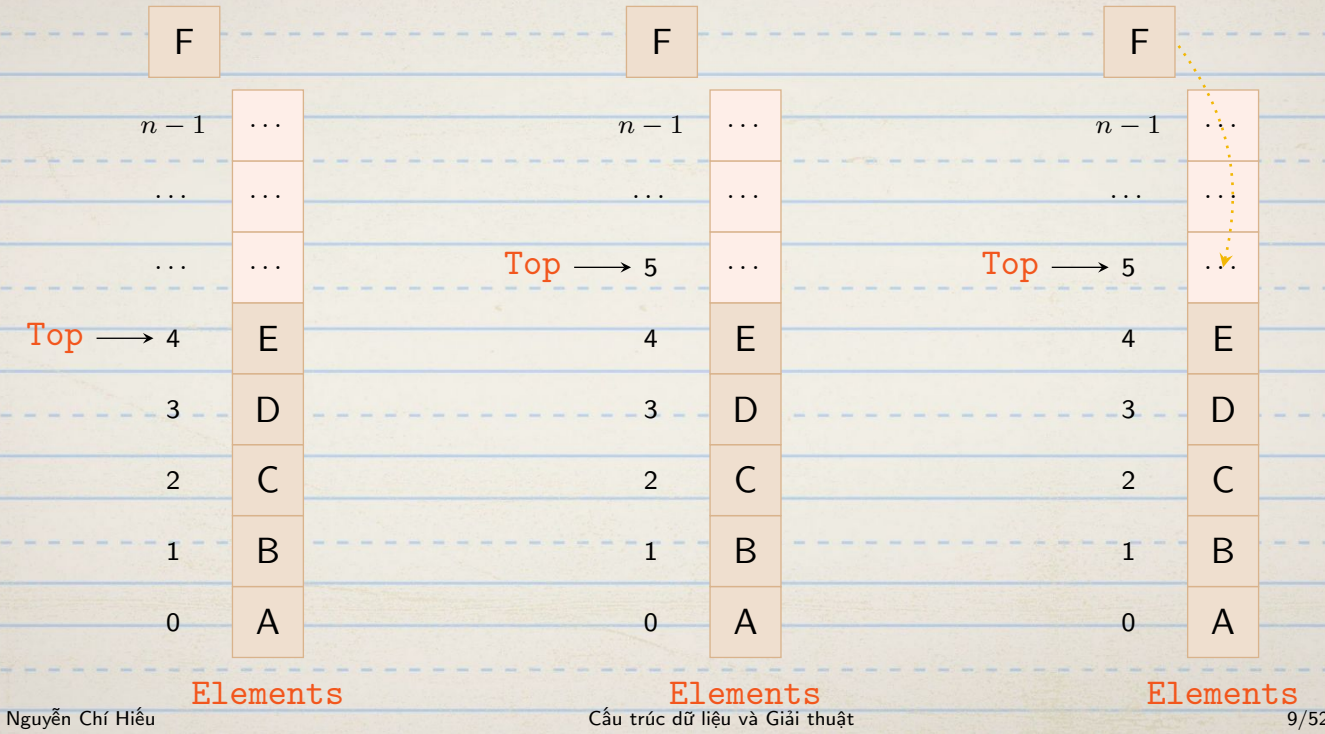
```
1 if top ≠ -1
2     return false
3 return true
```

Thuật toán 2: IsFull(s)
- Đầu vào: ngăn xếp s.
- Đầu ra: true/false.

```
1 if top ≠ MAX_SIZE - 1
2     return false
3 return true
```



Cài đặt ngăn xếp bằng mảng



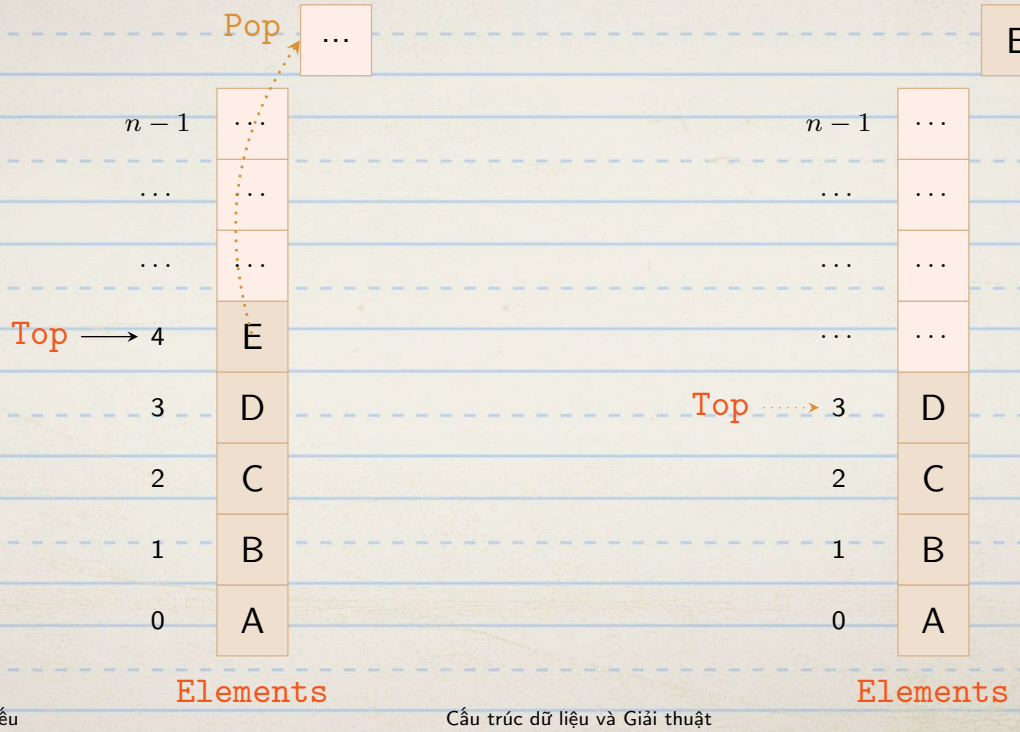
Cài đặt ngăn xếp bằng mảng

Thuật toán 3: `Push(s, x)`

- Đầu vào: ngăn xếp `s` và phần tử `x` cần thêm.
- Đầu ra: ngăn xếp `s` sau khi thêm `x`.

```
1  if ngăn xếp chưa đầy
2      top ← top + 1
3      elements[top] ← x
```

Cài đặt ngăn xếp bằng mảng



Cài đặt ngăn xếp bằng mảng

Thuật toán 4: Pop(s)

- Đầu vào: ngăn xếp s.
- Đầu ra: lấy ra và xóa phần tử khỏi đỉnh ngăn xếp

```
1 if ngăn xếp khác rỗng
2     x ← elements[top]
3     top ← top - 1 // Pop(s) ≠ GetTop(s)
4     return x
5 return STACK_EMPTY
```

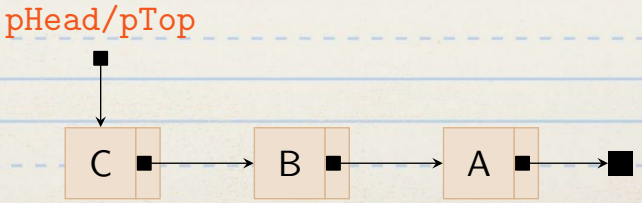
Thuật toán 5: GetTop(s)

- Đầu vào: ngăn xếp s.
- Đầu ra: lấy ra phần tử đỉnh ngăn xếp

```
1 if ngăn xếp khác rỗng
2     x ← elements[top]
3     return x
4 return STACK_EMPTY
```

Cài đặt ngăn xếp bằng danh sách liên kết

- Cấu trúc dữ liệu một phần tử của ngăn xếp chứa thành phần dữ liệu và thành phần liên kết *(tương tự danh sách liên kết)*.
- Cấu trúc dữ liệu ngăn xếp chứa một con trỏ pHead/pTop trỏ đến phần tử *đầu/đỉnh* của ngăn xếp.
- Các thao tác thêm, xóa phần tử thực hiện ở *đầu/đỉnh* ngăn xếp.



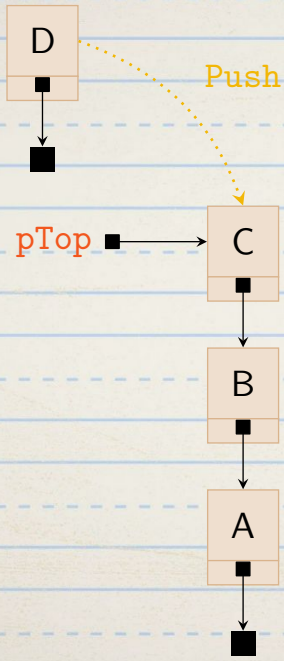
Cài đặt ngăn xếp bằng danh sách liên kết

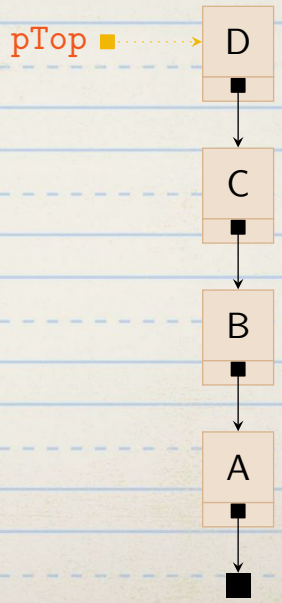
```
1  public class Node
2  {
3      public int info;
4      public Node pNext;
5
6      public void InitNode(int x)
7      {
8          info = x;
9          pNext = null;
10     }
11 }
```

Cài đặt ngăn xếp bằng danh sách liên kết

```
1  public class Stack
2  {
3      public Node pTop;
4      public int size;
5
6      public void InitStack()
7      {
8          pTop = null;
9          size = 0;
10     }
11 }
```


Thao tác thêm phần tử

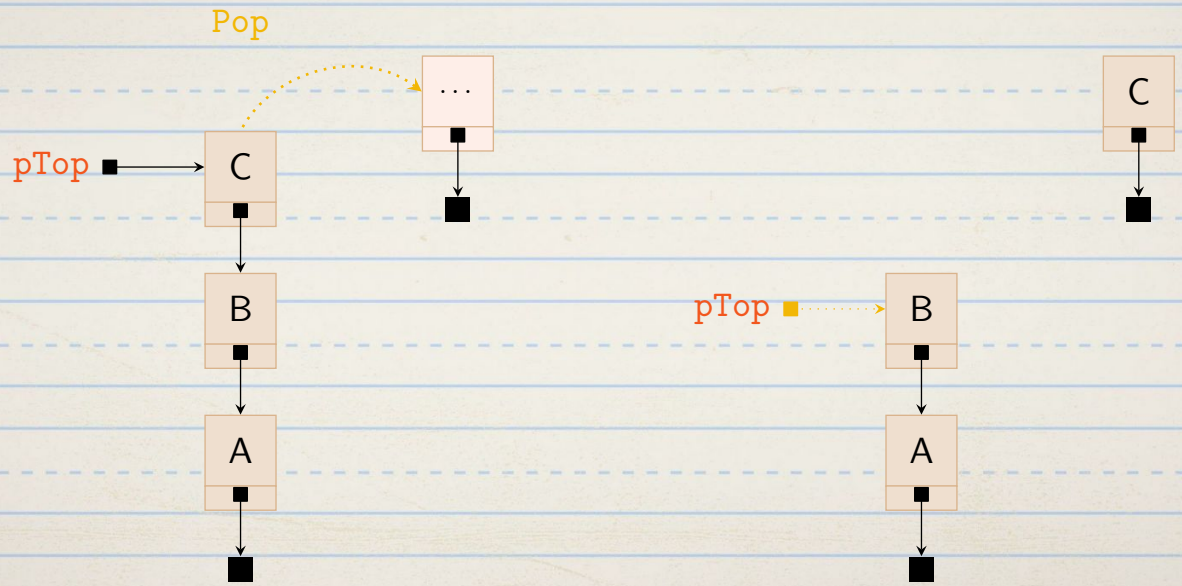




Thao tác thêm phần tử

```
1  // Push phần tử vào đỉnh ngăn xếp
2  // tương tự thao tác thêm đầu danh sách liên kết (hàm InsertHead)
3  public void Push(Node p)
4  {
5      if (p == null)
6          return;
7      p.pNext = pTop;
8      pTop = p;
9      size++;
10 }
```


Thao tác lấy phần tử



Thao tác lấy phần tử

```
1  // Pop lấy ra phần tử từ đỉnh ngăn xếp và xóa khỏi ngăn xếp
2  // tương tự thao tác xóa đầu danh sách liên kết (RemoveHead)
3  public Node Pop()
4  {
5      if (pTop == null)
6          return STACK_EMPTY;
7      Node p = pTop;
8      int x = p.info;
9      pTop = pTop.pNext;
10     size--;
11     p = null; // delete p
12     return x;
13 }
```


Thao tác lấy phần tử tại đỉnh ngăn xếp

```
1  // GetTop lấy phần tử tại đỉnh ngăn xếp, nhưng không xóa khỏi ngăn xếp
2  public int GetTop()
3  {
4      if (pTop == null)
5          return STACK_EMPTY;
6      return pTop.info;
7  }
```

Ứng dụng của ngăn xếp

Một số ứng dụng của ngăn xếp

- Sử dụng để khử đệ quy.
- Trong trình biên dịch, ngăn xếp được dùng để lưu trữ các thủ tục, biến, ...
- Tính giá trị biểu thức *(sử dụng ký pháp Ba Lan ngược)*.
- ...

Biểu diễn biểu thức

Phụ thuộc vào thứ tự toán tử *(operator)* đối với các toán hạng *(operand)*

- Trung tố *(Infix)*
  Cú pháp:

  `toán hạng toán tử toán hạng`
- Hậu tố *(Postfix)* hay ký pháp Ba Lan ngược *(RPN-Reverse Polish notation)*
  Cú pháp:

  `toán hạng toán hạng toán tử`

Biểu diễn biểu thức

Ví dụ 1

Xét biểu thức $x + y - z$

- Biểu diễn dạng trung tố
  - $x + (y - z)$
  - $(x + y) - z$
- Biểu diễn dạng hậu tố
  - $x\ y\ z - +$ $\Leftrightarrow$ $x + (y - z)$
  - $x\ y + z -$ $\Leftrightarrow$ $(x + y) - z$

Tính giá trị biểu thức

Hai bước thực hiện

1. Chuyển từ trung tố sang hậu tố.
2. Tính giá trị biểu thức hậu tố.

Chuyển từ trung tố sang hậu tố

Thuật toán 6: ConvertRPN(infix)

- Đầu vào: infix (biểu thức trung tố)
- Đầu ra: postfix (biểu thức hậu tố)

```
1  // Khoi tao stack rong
2  stack ← ∅
3
4  // Duyet infix: trai → phai
5  while infix ≠ ∅
6      // Doc 1 ky tu trong infix
7      read x from infix
8
9      // TH1: dau '('
10     if x = '('
11         Push(stack, x)
```


Chuyển từ trung tố sang hậu tố

```
12     // TH2: dau ')'
13     else if x = ')'
14         y ← Pop(stack)
15         // Pop den khi gap dau '('
16         while y ≠ '('
17             write y to postfix
18             y ← Pop(stack)
```

Chuyển từ trung tố sang hậu tố

```
19  ____// TH3: toan tu (+, -, *, /, ...)
20  ____else if x = operator
21  ____ ____// Xet do uu tien tat ca toan tu trong stack
22  ____ ____while GetPriority(GetTop(stack)) ≥ GetPriority(x)
23  ____ ____ ____y ← Pop(stack)
24  ____ ____ ____write y to postfix
25  ____ ____Push(stack, x)
```


Chuyển từ trung tố sang hậu tố

```
26  ____// TH4: toan hang (a, B, 1, ...)
27  ____else // x = operand
28  ____ ____write x to postfix
29  ____
30  ____// Lay ra tat ca ky tu trong stack
31  ____while stack ≠ ∅
32  ____ ____y ← Pop(stack)
33  ____ ____write y to postfix
```

Chuyển từ trung tố sang hậu tố

Ví dụ 2

Cho biểu thức $2 * (7 + 3) - 8$. Biểu diễn biểu thức dạng hậu tố.

Ký tự	Trường hợp	Infix	Postfix
		$2 * (7 + 3) - 8$	

Stack

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$* (7 + 3) - 8$	2

Stack

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\ (7+3)-8$	2
$*$	TH3	$(7+3)-8$	2

Stack
$*$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\ (7+3)-8$	2
$*$	TH3	$(7+3)-8$	2
$($	TH1	$7+3)-8$	2

Stack
$($
$*$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix	Stack
2	TH4	$*\ (7+3)-8$	2	
$*$	TH3	$(7+3)-8$	2	
$($	TH1	$7+3)-8$	2	$($
7	TH4	$+\ 3)-8$	2 7	$*$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix	Stack
2	TH4	$*\ (7+3)-8$	2	
$*$	TH3	$(7+3)-8$	2	
$($	TH1	$7+3)-8$	2	$+$
7	TH4	$+\ 3)-8$	2 7	$($
$+$	TH3	$3)-8$	2 7	$*$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\ (7+3)-8$	2
$*$	TH3	$(7+3)-8$	2
$($	TH1	$7+3)-8$	2
7	TH4	$+\ 3)-8$	2 7
$+$	TH3	$3)-8$	2 7
3	TH4	$)-8$	2 7 3

Stack
$+$
$($
$*$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\ (7+3)-8$	2
$*$	TH3	$(7+3)-8$	2
$($	TH1	$7+3)-8$	2
7	TH4	$+\ 3)-8$	2 7
$+$	TH3	$3)-8$	2 7
3	TH4	$)-8$	2 7 3
$)$	TH2	$-\ 8$	2 7 3 $+$

Stack
$*$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\ (7+3)-8$	2
$*$	TH3	$(7+3)-8$	2
(	TH1	$7+3)-8$	2
7	TH4	$+\ 3)-8$	2 7
$+$	TH3	$3)-8$	2 7
3	TH4	$)-8$	2 7 3
)	TH2	$-\ 8$	2 7 3 $+$
$-$	TH3	8	2 7 3 $+$ $*$

Stack
$-$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\ (7+3)-8$	2
$*$	TH3	$(7+3)-8$	2
(	TH1	$7+3)-8$	2
7	TH4	$+\ 3)-8$	2 7
$+$	TH3	$3)-8$	2 7
3	TH4	$)-8$	2 7 3
)	TH2	$-\ 8$	2 7 3 $+$
$-$	TH3	8	2 7 3 $+$ $*$
8	TH4		2 7 3 $+$ $*$ 8

Stack
$-$

Chuyển từ trung tố sang hậu tố

Ký tự	Trường hợp	Infix	Postfix
2	TH4	$*\,(7+3)-8$	2
$*$	TH3	$(7+3)-8$	2
$($	TH1	$7+3)-8$	2
7	TH4	$+\,3)-8$	$2\ 7$
$+$	TH3	$3)-8$	$2\ 7$
3	TH4	$)-8$	$2\ 7\ 3$
$)$	TH2	$-\,8$	$2\ 7\ 3\ +$
$-$	TH3	8	$2\ 7\ 3\ +\ *$
8	TH4		$2\ 7\ 3\ +\ *\ 8$
			$2\ 7\ 3\ +\ *\ 8\ -$

Stack

Chuyển từ trung tố sang hậu tố

Ví dụ 3

Xét các biểu thức sau

(a) $1+2*(9-3)/6$

(b) $3*((6+5)-9)$

Kết quả

(a)
- Stack: .....
- Biểu thức dạng trung tố: .....
- Biểu thức dạng hậu tố: .....
- ... 1 2 9 3 - * 6 / +

(b)
- Stack: .....
- Biểu thức dạng trung tố: .....
- Biểu thức dạng hậu tố: .....
- ... 3 6 5 + 9 - *

Tính biểu thức hậu tố

Thuật toán 7: ComputeRPN(postfix)
- Đầu vào: postfix (biểu thức hậu tố)
- Đầu ra: giá trị biểu thức

```
1  // Khoi tao stack rong
2  stack ← ∅
3
4  // Duyet postfix: trai → phai
5  while postfix ≠ ∅
6      // Doc 1 ky tu trong postfix
7      read x from postfix
8      // TH1: toan hang
9      if x = operand
10         Push(stack, x)
```

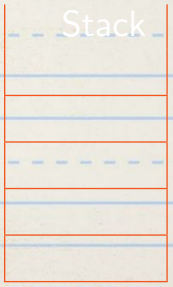

Tính biểu thức hậu tố I

```
11     // TH2: toan tu (+, -, *, /, ...)
12     else
13         x2 ← Pop(stack)
14         x1 ← Pop(stack)
15         // x1, x2: operand; x: operator
16         y ←Calculate(x1, x2, x) // y ← x1 x x2
17         Push(stack, y)
18 return Pop(stack)
```

Tính biểu thức hậu tố

Ví dụ 4

Cho biểu thức $2\ 7\ 3\ +\ *\ 8\ -$. Tính biểu thức dạng hậu tố.



Ký tự	Trường hợp	Postfix
		$2\ 7\ 3\ +\ *\ 8\ -$

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	$7\ 3\ +\ *\ 8\ -$

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	$7\ 3\ +\ *\ 8\ -$
7	TH1	$3\ +\ *\ 8\ -$

Stack
7
2

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	$7\ 3\ +\ *\ 8\ -$
7	TH1	$3\ +\ *\ 8\ -$
3	TH1	$+\ *\ 8\ -$

Stack
3
7
2

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	$7\ 3\ +\ *\ 8\ -$
7	TH1	$3\ +\ *\ 8\ -$
3	TH1	$+\ *\ 8\ -$
$+$	TH2	$*\ 8\ -$

Stack
10
2

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	$7\ 3\ +\ *\ 8\ -$
7	TH1	$3\ +\ *\ 8\ -$
3	TH1	$+\ *\ 8\ -$
$+$	TH2	$*\ 8\ -$
$*$	TH2	$8\ -$

Stack
20

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	7 3 + ∗ 8 −
7	TH1	3 + ∗ 8 −
3	TH1	+ ∗ 8 −
+	TH2	∗ 8 −
∗	TH2	8 −
8	TH1	−

Stack
8
20

Tính biểu thức hậu tố

Ký tự	Trường hợp	Postfix
2	TH1	7 3 + ∗ 8 −
7	TH1	3 + ∗ 8 −
3	TH1	+ ∗ 8 −
+	TH2	∗ 8 −
∗	TH2	8 −
8	TH1	−
−	TH2	

Stack
12

Bài tập

1. Cho một ngăn xếp rỗng, hãy lần lượt thực hiện các thao tác sau đây: Push(1), Push(5), Push(2), Push(7), Pop(), Pop(), Push(9). Hãy vẽ ngăn xếp tương ứng với các thao tác trên.
2. Cài đặt hàm thực hiện thao tác chuyển một số hệ thập phân *(cơ số 10)* sang hệ nhị phân *(cơ số 2)* có sử dụng ngăn xếp.
3. Xét bài toán dùng chuyển biểu thức từ trung tố sang hậu tố bằng cấu trúc ngăn xếp. Nếu sau khi đọc 5 ký tự, ngăn xếp chứa lần lượt các ký tự: - (+
   Hãy cho biết ký tự thứ 6 đọc được sẽ có kiểu là gì: toán tử, toán hạng, ngoặc đóng hay ngoặc mở? Giải thích.

Tài liệu tham khảo


Donald E. Knuth.
The Art of Computer Programming, Volume 3.
Addison-Wesley, 1998.


Dương Anh Đức, Trần Hạnh Nhi.
Nhập môn Cấu trúc dữ liệu và Thuật toán.
Đại học Khoa học tự nhiên TP Hồ Chí Minh, 2003.


Niklaus Wirth.
Algorithms + Data Structures = Programs.
Prentice-Hall, 1976.


Robert Sedgewick.
Algorithms in C.
Addison-Wesley, 1990.