Chương 4.
NGĂN XẾP

ThS. Nguyễn Chí Hiếu

2017

NỘI DUNG

Giới thiệu ngăn xếp

Cài đặt ngăn xếp

Ứng dụng của ngăn xếp

Ngăn xếp (Stack)

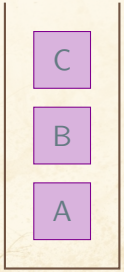
- Thực hiện theo cơ chế *LIFO (Last In, First Out) vào sau ra trước*.
- Dùng để lưu trữ các phần tử có thứ tự truy xuất *ngược* với thứ tự lưu trữ.

Giới thiệu ngăn xếp

Các thao tác cơ bản

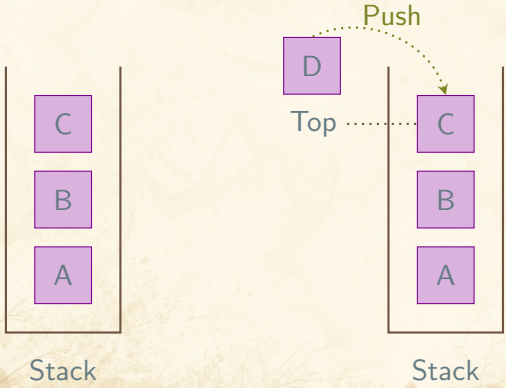
- Push: *thêm* phần tử vào *đỉnh* ngăn xếp.
- Pop: *xóa* phần tử tại *đỉnh* ngăn xếp.
- GetTop: *lấy* phần tử tại *đỉnh* ngăn xếp.
- Kiểm tra ngăn xếp rỗng, đầy.

Giới thiệu ngăn xếp

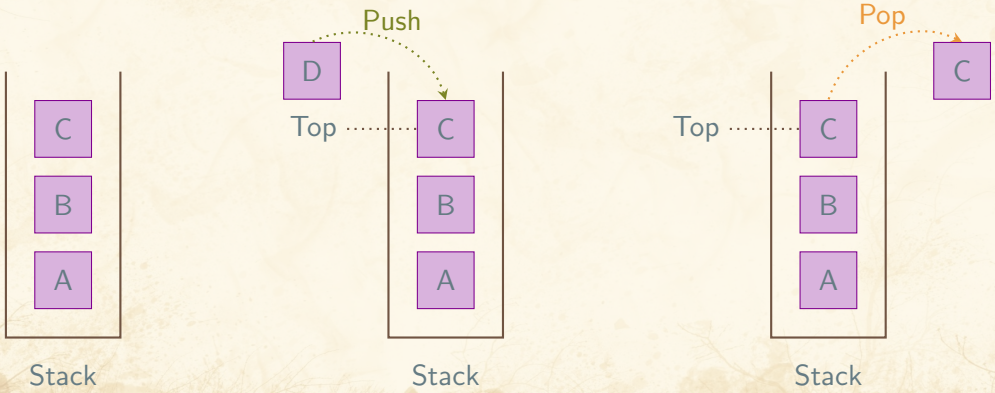


Stack

Giới thiệu ngăn xếp



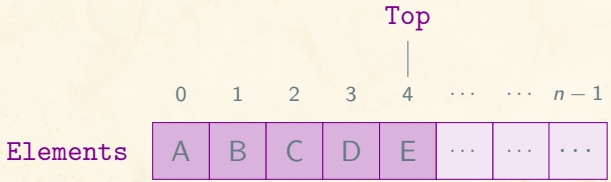
Giới thiệu ngăn xếp



Cài đặt ngăn xếp bằng mảng

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

```
1  struct Stack
2  {
3      Data  Elements[MAX_SIZE];
4      int   Top;
5  };
```



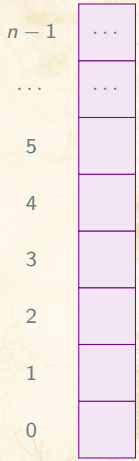
Cài đặt ngăn xếp bằng mảng

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

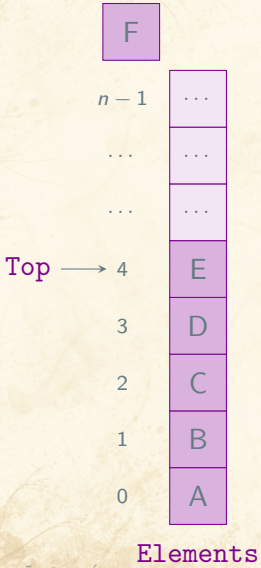
```
1    if Top = -1
2        return true
3    return false
```

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

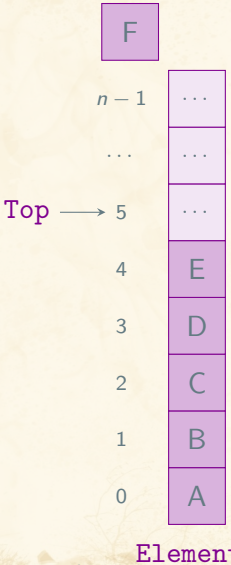
```
1    if Top = n - 1
2        return true
3    return false
```



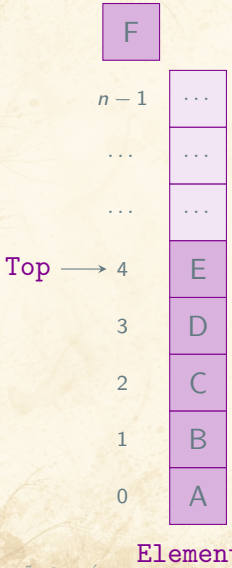
Cài đặt ngăn xếp bằng mảng

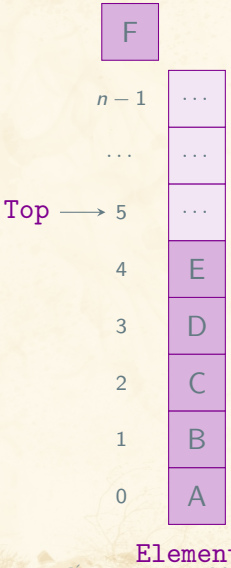


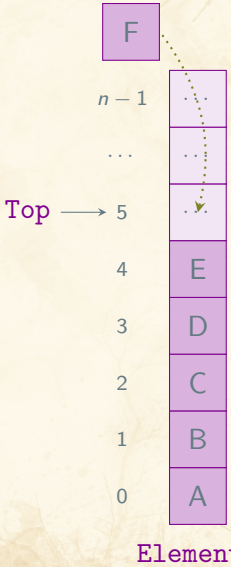
Cài đặt ngăn xếp bằng mảng



Cài đặt ngăn xếp bằng mảng







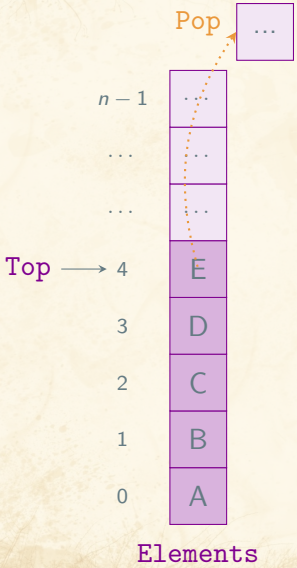
Cài đặt ngăn xếp bằng mảng

Thuật toán 3: Push(s, x)

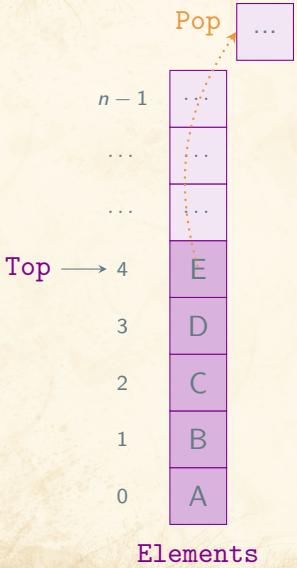
- Đầu vào: ngăn xếp s và phần tử x cần thêm.
- Đầu ra: ngăn xếp s sau khi thêm x.

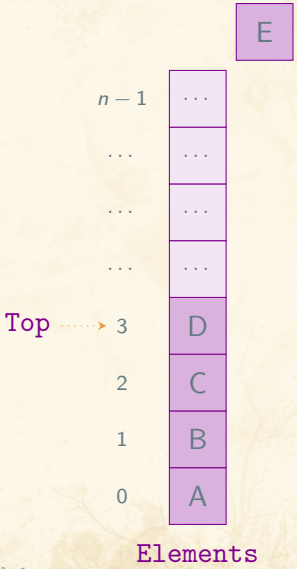
```
1    if ngăn xếp chưa đầy
2        Top ← Top + 1
3        Elements[Top] ← x
```

Cài đặt ngăn xếp bằng mảng



Cài đặt ngăn xếp bằng mảng





Cài đặt ngăn xếp bằng mảng

Thuật toán 4: Pop(s)

- Đầu vào: ngăn xếp s.
- Đầu ra: phần tử đỉnh ngăn xếp hay NULL_DATA (ngăn xếp rỗng).

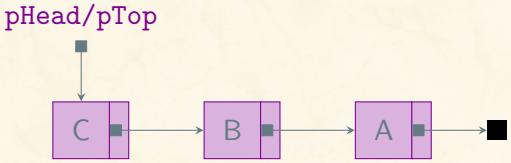
```
1    if ngăn xếp khác rỗng
2        x ← Elements[Top]
3        Top ← Top - 1 // Pop(s) ≠ GetTop(s)
4        return x
5    return NULL_DATA
```

Thuật toán 5: GetTop(s)

```
1    if ngăn xếp khác rỗng
2        x ← Elements[Top]
3        return x
4    return NULL_DATA
```

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.

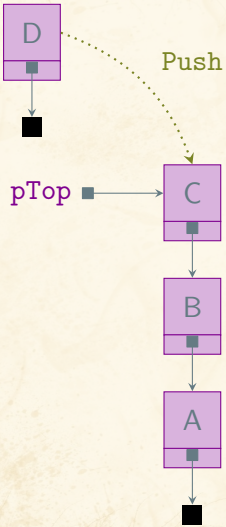


Cài đặt ngăn xếp bằng danh sách liên kết

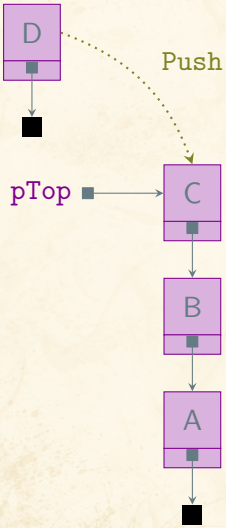
```
1  typedef int Data;
2  struct Node
3  {
4     Data Info;
5     Node *pNext;
6  };
```

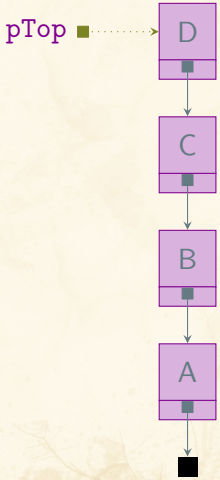
```
1  struct Stack
2  {
3     int   Count; // Dem so phan tu trong ngan xep
4     Node  *pTop; // pHead
5  };
```

Thao tác thêm phần tử



Thao tác thêm phần tử

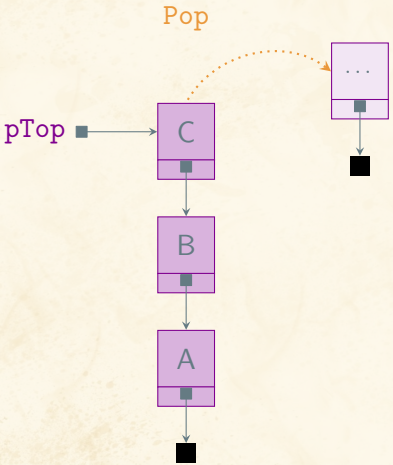




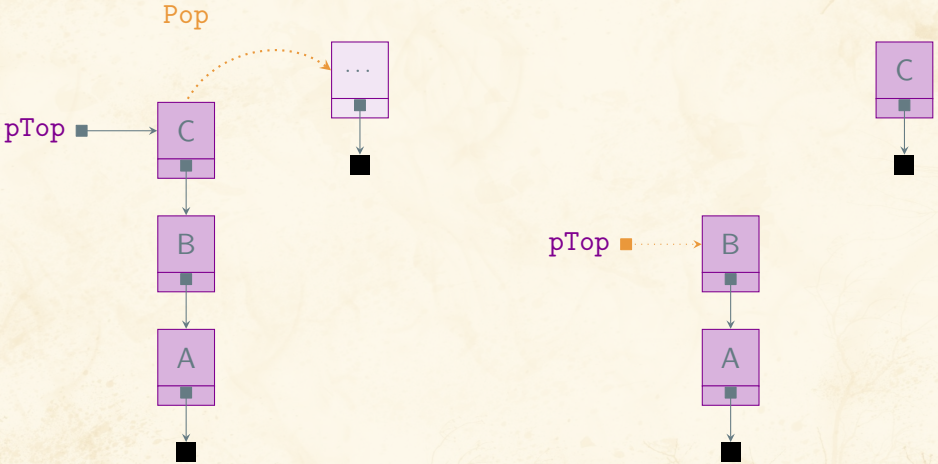
Thao tác thêm phần tử

```
1  // Push phần tử vào đỉnh ngăn xếp
2  // tương tự thao tác thêm đầu danh sách liên kết (hàm InsertHead)
3  void Push(Stack *s, Data x)
4  {
5      Node *p = InitNode(x);
6      if (s->pTop == NULL) // TH1. Ngan xep rong
7          s->pTop = p;
8      else // TH2. Ngan xep khac rong
9      {
10         p->pNext = s->pTop;
11         s->pTop = p;
12     }
13     s->Count++;
14 }
```

Thao tác lấy phần tử



Thao tác lấy phần tử



Thao tác lấy phần tử

```
1  // Pop phần tử từ đỉnh ngăn xếp
2  // tương tự thao tác xóa đầu danh sách liên kết (RemoveHead)
3  Data Pop(Stack *s)
4  {
5      Node *p = new Node();
6      if (s->pTop == NULL) // TH1. Ngan xep rong
7          return NULL_DATA;
8      // TH2. Ngan xep khac rong
9      p = s->pTop;
10     s->pTop = s->pTop->pNext;
11     s->Count--;
12     Data x = p->Info;
13     delete p;
14     return x;
15 }
```

Cài đặt ngăn xếp bằng danh sách liên kết

```
1  Data GetTop(Stack *s)
2  {
3      Node *p = new Node();
4      if (s->pTop == NULL) // TH1. Ngan xep rong
5          return NULL_DATA;
6      // TH2. Ngan xep khac rong
7      p = s->pTop;
8
9      return p->Info;
10 }
```

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)
12
```

Chuyển từ trung tố sang hậu tố

```
14      // TH2: dau ')'
15      else if x = ')'
16          y ← Pop(stack)
17          // Pop den khi gap dau '('
18          while y ≠ '('
19              write y to postfix
20              y ← Pop(stack)
21      
```

Chuyển từ trung tố sang hậu tố

```
21      // TH3: toan tu (+, -, *, /, ...)
22      else if x = operator
23          // Xet do uu tien tat ca toan tu trong stack
24          while GetPriority(GetTop(stack)) ≥ GetPriority(x)
25              y ← Pop(stack)
26              write y to postfix
27          Push(stack, x)
28
```

Chuyển từ trung tố sang hậu tố

```
30      // TH4: toan hang (a, B, 1, ...)
31      else // x = operand
32          write x to postfix
33
34  // Lay ra tat ca ky tu trong stack
35  while stack ≠ ∅
36      y ← Pop(stack)
37      write y to postfix
```

Chuyển từ trung tố sang hậu tố

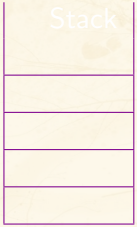
Ví dụ 2

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

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

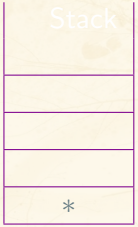
Chuyển từ trung tố sang hậu tố

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



Chuyển từ trung tố sang hậu tố

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



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

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

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

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

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
9      // TH1: toan hang
10     if x = operand
11         Push(stack, x)
```

Tính biểu thức hậu tố I

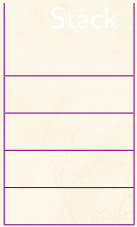
```
10      // TH2: toan tu (+, -, *, /, ...)
11      else
12          x2 ← Pop(stack)
13          x1 ← Pop(stack)
14          // x1, x2: operand; x: operator
15          y ←Calculate(x1, x2, x) // y ← x1 x x2
16          Push(stack, y)
17  return Pop(stack)
```

Tính biểu thức hậu tố

Ví dụ 4

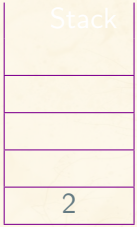
Cho biểu thức 2 7 3 $+$ $*$ 8 $-$. Tính biểu thức dạng hậu tố.

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



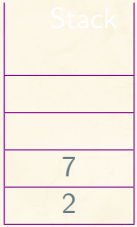
Tính biểu thức hậu tố

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 −



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. Cài đặt ngăn xếp sử dụng mảng và danh sách liên kết.
2. Sử dụng ngăn xếp để thực hiện thao tác chuyển một số hệ thập phân *(cơ số 10)* sang hệ nhị phân *(cơ số 2)*.
3. Tính toán giá trị biểu thức bằng ký pháp Ba Lan ngược.

Tài liệu tham khảo

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.

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

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

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