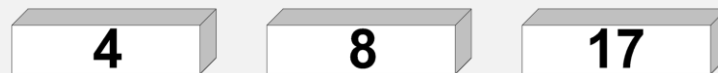


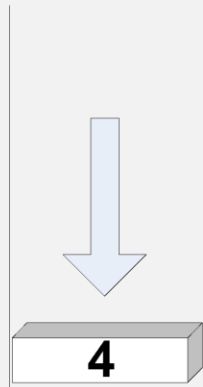
## 3.5 Stack

A stack is a linear list with the restriction that insertions and deletions can be performed in only one position, namely, the end of the list, called the top. Stack is LIFO (last in, first out) data structure

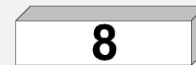
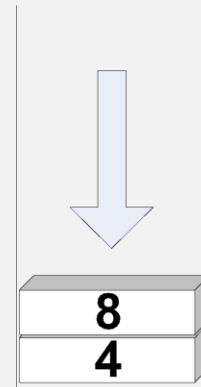
จัดเรียงข้อมูล  
ตามลำดับการ push



Push

Push

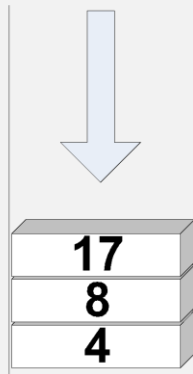
### 3.5.1 The fundamental operations on a stack

**push** : is equivalent to an insert, add an item at the top of the stack. problem **\*\*stack overflow**

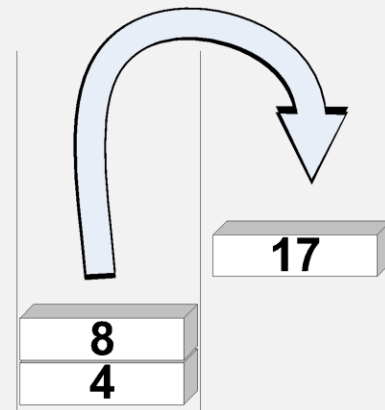
**pop** : deletes the most recently inserted element. problem **\*\*stack underflow**

**top** : return top value **\*\*stack underflow**

Push



Pop



The general model is that there is some element that is at the top of the stack, and it is only element that is visible

### **3.5.2 Implementation of stacks.**

- Linked List
- Array

```
1. #include <string.h>
2. #include <stdio.h>
3. #include <iostream>
4. using namespace std;
5. struct Node;
6. typedef struct Node *Stack;
7. struct Node
8. {   int value;
9.     struct Node *Next;
10. };
```

```
11. int IsEmpty(Stack S);
12. Stack CreateStack(void);
13. void MakeEmpty(Stack
    S);
14. void Push(int X, Stack S);
15. void Pop(Stack S);
```



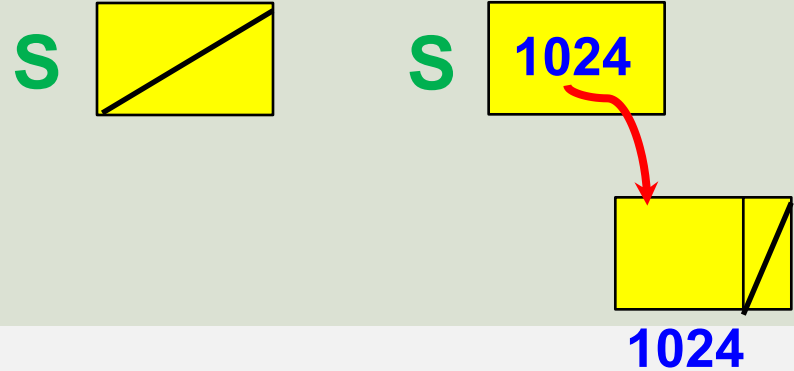
## โครง Stack

CreateStack สร้างตัวชี้ stack

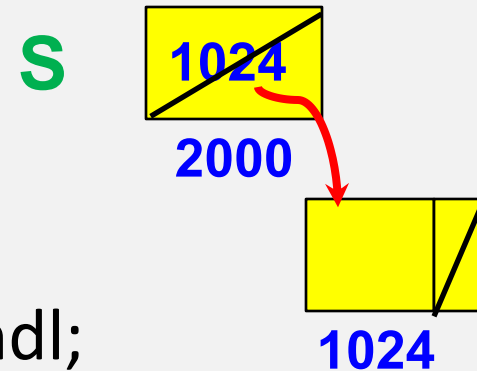
การ insert จะใช้การ push

การ delete จะใช้การ pop

```
int main()
{
    Stack S=NULL;
    S=CreateStack();
}
```



```
Stack CreateStack(void)
{
    Stack S;
    S = new struct Node;
    if (S== NULL)
        cout <<"Out of space!!!"<<endl;
    S->Next=NULL;
    return S;
}
```



```
int main()
{
    Stack S=NULL;
    S=CreateStack();
    Push(5,S);
}
```

S

1024

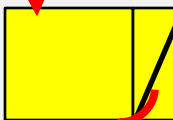
9000

```
void Push(int X,Stack S)
{
    Stack TmpCell;
    TmpCell = new struct Node;
    if(TmpCell == NULL)
        cout << "Out of space!!!";
    else
    {
        TmpCell->value = X;
        TmpCell->Next = S->Next;
        S->Next = TmpCell;
    }
}
```

S

1024

2000



1024

TmpCell

1050

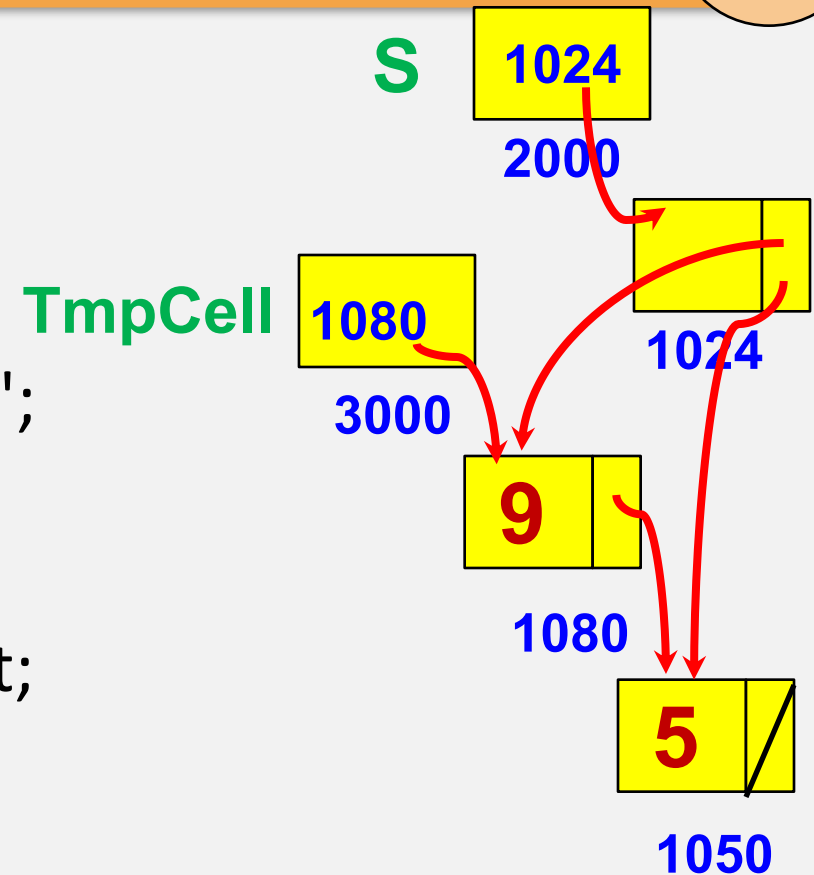
3000



1050

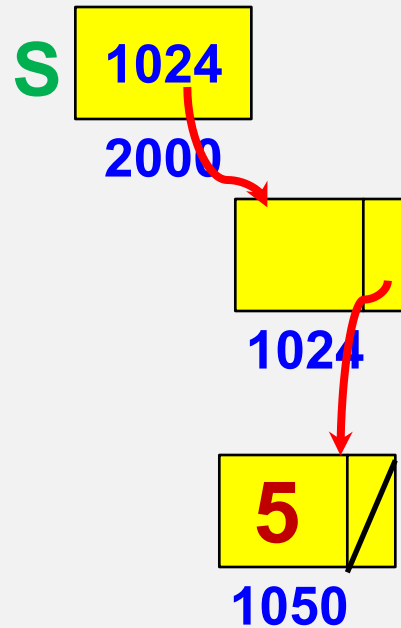
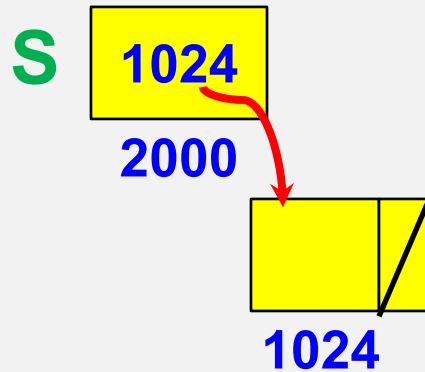


```
void Push(int X,Stack S)
{
    Stack TmpCell;
    TmpCell = new struct Node;
    if(TmpCell == NULL)
        cout << "Out of space!!!";
    else
    {
        TmpCell->value = X;
        TmpCell->Next = S->Next;
        S->Next = TmpCell;
    }
}
```





```
void MakeEmpty(Stack S)
{   if (S== NULL)
        cout << "Must use CreateStack first" << endl;
    else
        while(!IsEmpty(S))
            Pop(S);
}
```

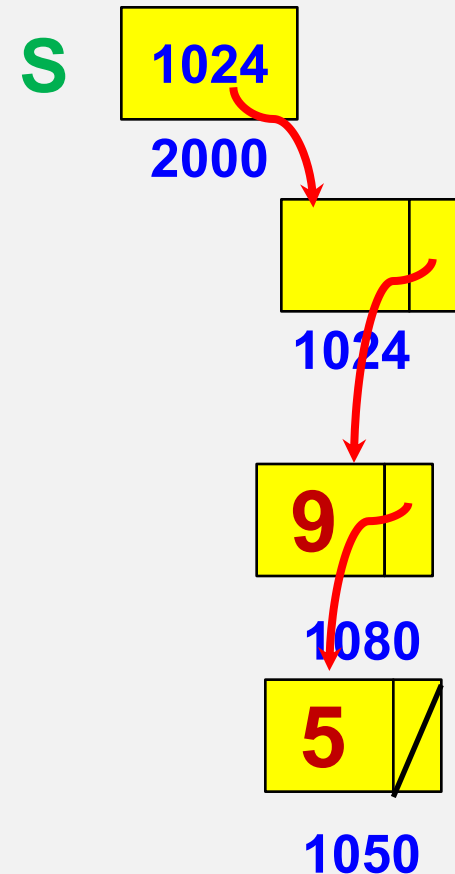
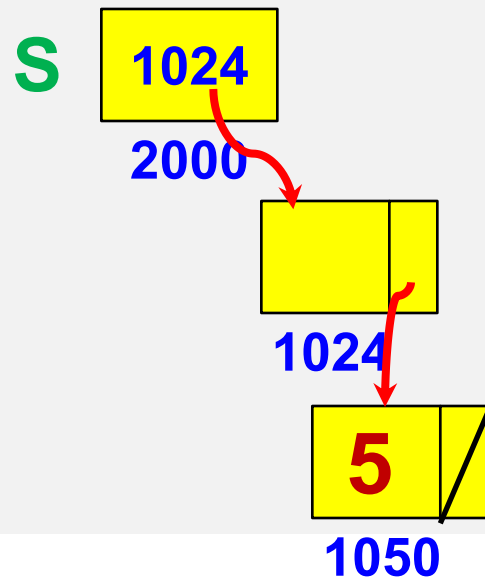


```

int IsEmpty(Stack S)
{
    return S->Next == NULL;
}
    
```

```

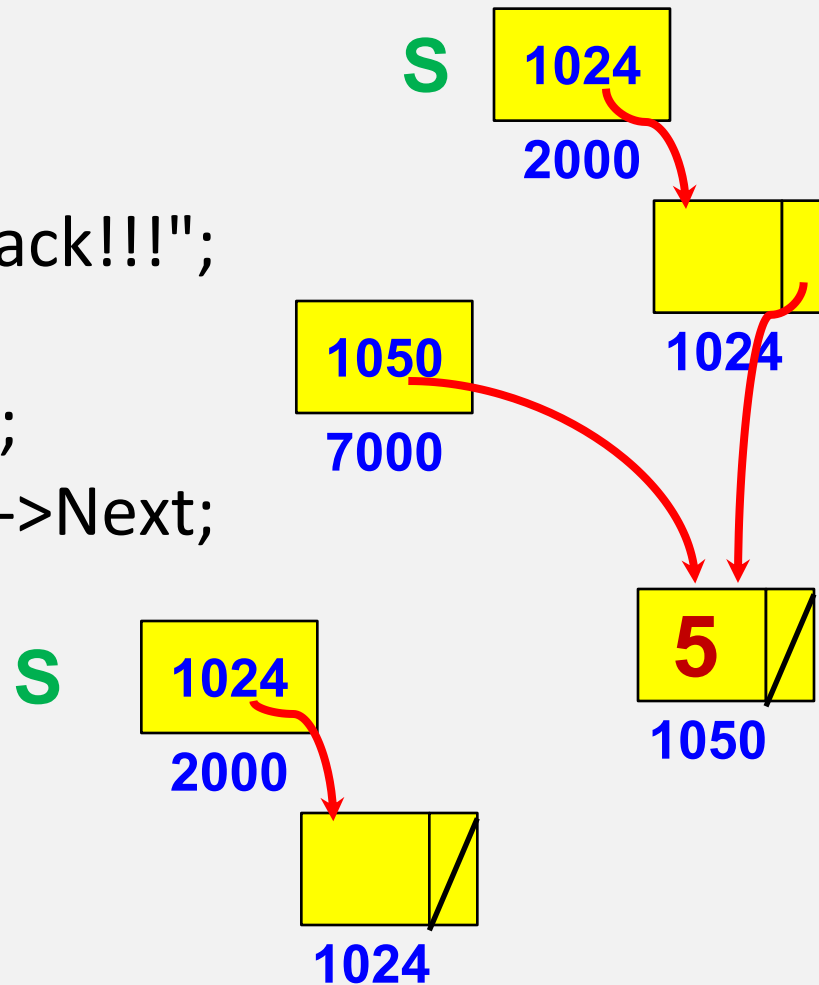
int Top(Stack S)
{
    if(!IsEmpty(S))
        return S->Next->value;
    cout << "Empty Stack!!!"<< endl;
    return 0;
}
    
```



```

void Pop(Stack S)
{
    Stack FirstCell;
    if(IsEmpty(S))
        cout << "Empty Stack!!!";
    else
    {
        FirstCell = S->Next;
        S->Next = S->Next->Next;
        delete(FirstCell);
    }
}

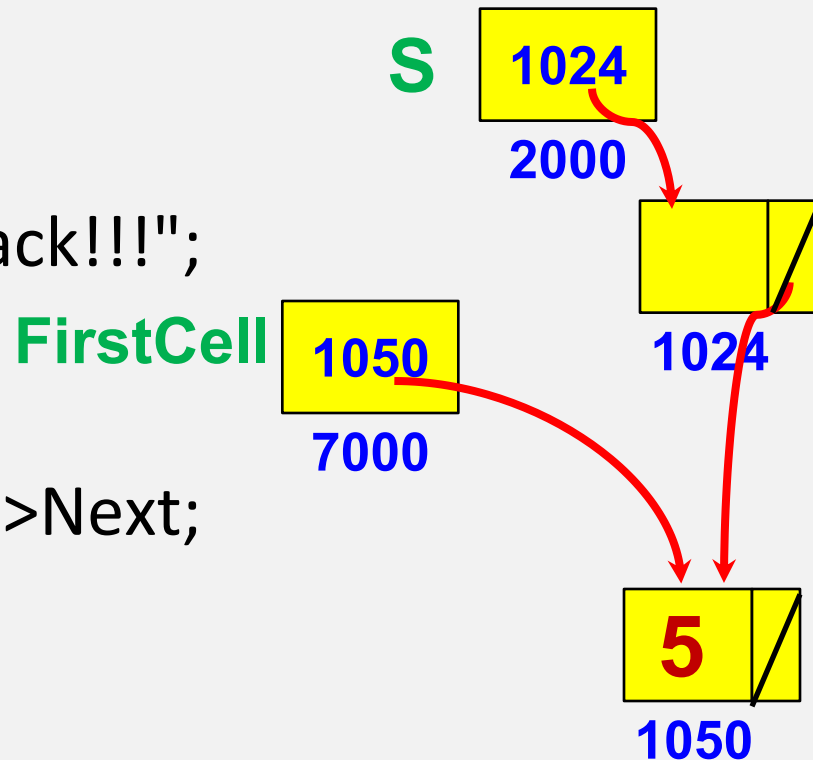
```



```

void Pop(Stack S)
{
    Stack FirstCell;
    if(IsEmpty(S))
        cout << "Empty Stack!!!";
    else
        {
            FirstCell = S->Next;
            S->Next = S->Next->Next;
            delete(FirstCell);
        }
}

```



## การบ้าน

### 2. stack

จงเขียนโปรแกรมทำการ push pop หรือ top ข้อมูลด้วย stack

=====

Menu

=====

1) Push

2) Pop

3) Top

4) Exit

Please choose >

ถ้าเลือกข้อ 1

Push : 10

Top = 10 //เมื่อ push เสร็จแล้ว ให้ทำการ top ข้อมูลขึ้นมาแสดง  
จากนั้นกลับไปยังเมนู

ถ้าเลือกข้อ 1 อีก

Push : 8

Top = 8

ให้ทำการ push ข้อมูลทั้งหมดดังนี้ 10 8 15 22 3

ถ้าเลือกข้อ 2

Top = 3 //แสดงข้อมูลของ top ก่อนจะ pop ลบข้อมูล

Pop success! //จากนั้น pop ข้อมูลทิ้ง

ถ้าเลือกข้อ 3

Top = 22

ให้ทดลอง pop ข้อมูลจนหมด

ถ้าข้อมูลหมดแล้ว ยังมีการ pop ข้อมูล ให้แสดงข้อความ stack underflow

### 3.5.3 Application

#### 1) Balancing Symbols

```
int main()
{
    cout << "hello";
}
```

1. Make an empty stack.
2. Read characters until end of file.
3. If the character is an opening symbol, push it onto the stack.
4. If it is a - closing symbol,
  - then if the stack is empty report an error.
  - Otherwise, pop the stack.
5. If the symbol popped is not the corresponding opening symbol, the report an error.
6. At end of file, if the stack not empty report an error.



```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main()
{   stringstream ss;
    string str="";
    int num;
    while(str!=".")
    {       cin >> str;
            if(str==".")
                break;
            if(str=="+")
                cout << "Push or Pop"<< endl;
            else
```

```
else    //ตัวเลข
{
    ss << str;
    ss >> num;    // num ไปใช้งานได้แล้ว
    ss.clear();    //ต้อง clear
    cout << ++num << endl;
}
}
}
```

## 2) Infix and Postfix

Infix       $4 * 2$

Postfix     $42 *$

$4 * 2$        $= 42 *$

$4 * 2 + 3$      $= 42 * 3 +$

$4 + 2 * 3$      $= 423 * +$

$4 + 2 + 3$      $= 42 + 3 +$

$4 * 2 + 5 + 6 * 3 =$

### 3) Infix to Postfix Conversion

#### Example

Operator + , \* , ( , )

- parentheses

$$a + b * c + (d * e + f) * g = a b c * + d e * f + g * +$$

$$a * b - c + d$$

$$a / b + c * d$$

$$a - b * c / d$$

$$a - b * c + d$$

## Example

- Stack

$a*b-c+d$

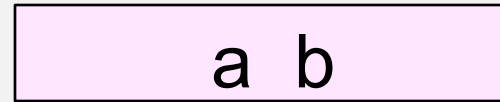
$a/b+c*d$

$a-b*c/d$

$a-b*c+d$



stack



output

$a + b * c + ( d * e + f ) * g$

$a + b * c + ( d * e + f ) * g$

$a \ b \ c \ * \ + \ d \ e \ * \ f \ + \ g \ * \ +$

## เงื่อนไข

1. ถ้า input เป็น operand ให้ print ที่จอภาพ
2. ถ้า input เป็น operator
  - 2.1 ถ้าเป็น operator ให้เปรียบเทียบ operator ใหม่กับค่าที่อยู่ top ของ stack
    - ถ้าค่าใหม่มี precedence มากกว่า ให้ push ข้อมูลลงใน stack ได้เลย
    - ถ้าค่าใหม่มี precedence น้อยกว่าหรือเท่ากับ ให้ pop ข้อมูลมาพิมพ์จนกว่า precedence จะน้อยกว่าค่าใหม่จะน้อยกว่าค่าใน stack หรือ stack empty แล้ว push ค่าใหม่ลงใน stack
    - ถ้าค่าใหม่เป็นวงเล็บเปิด ( ให้ push ลง stack ได้เลย และถือว่า precedence มีค่าน้อยที่สุด
    - ถ้าค่าใหม่เป็น วงเล็บปิด ) ให้ pop ข้อมูลขึ้นมาพิมพ์จนกว่าจะเจอเครื่องหมาย (

## 4) Postfix Expressions

$$42^* = 8$$

$$42^*3+ = 11$$

$$423^*+ = 10$$

$$42+3+ = 9$$

Infix :  $4 * 2 + 5 + 6 * 3$

Postfix :  $4 2 * 5 + 6 3 * +$   
 $= 31$



$42^*$



#### 4) Postfix Expressions

Implementation : Stack

Input number : push onto the stack

Input operator : applied to the two numbers that are popped from the stack.

$$42^* = 8$$

$$42^*3+ = 11$$

$$423^*+ = 10$$

$$42+3+ = 9$$

$$\text{Infix} : 4 * 2 + 5 + 6 * 3$$

$$\begin{aligned} \text{Postfix} : 4 2 * 5 + 6 3 * + \\ = 31 \end{aligned}$$