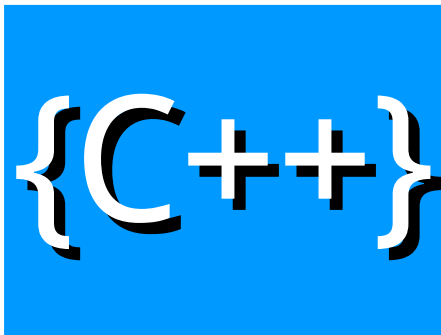




Exception

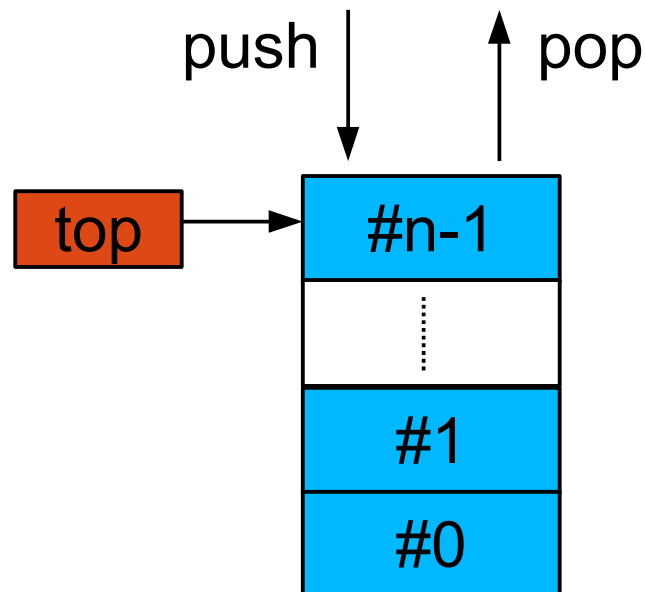
Week 12



Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw

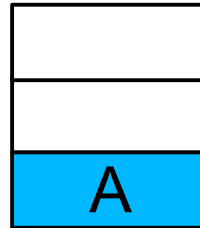
堆疊 (Stack)

- 加入 (push) 與刪除 (pop) 於同一端
- 具有後進先出 (LIFO, Last-in-First-out) 或先進後出 (FILO, First-in-Last-out) 性質的有序串列

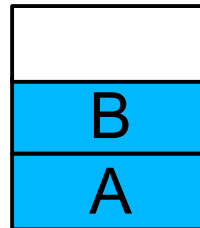


堆疊範例

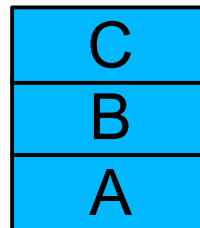
Push(A)



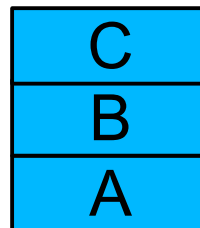
Push(B)



Push(C)

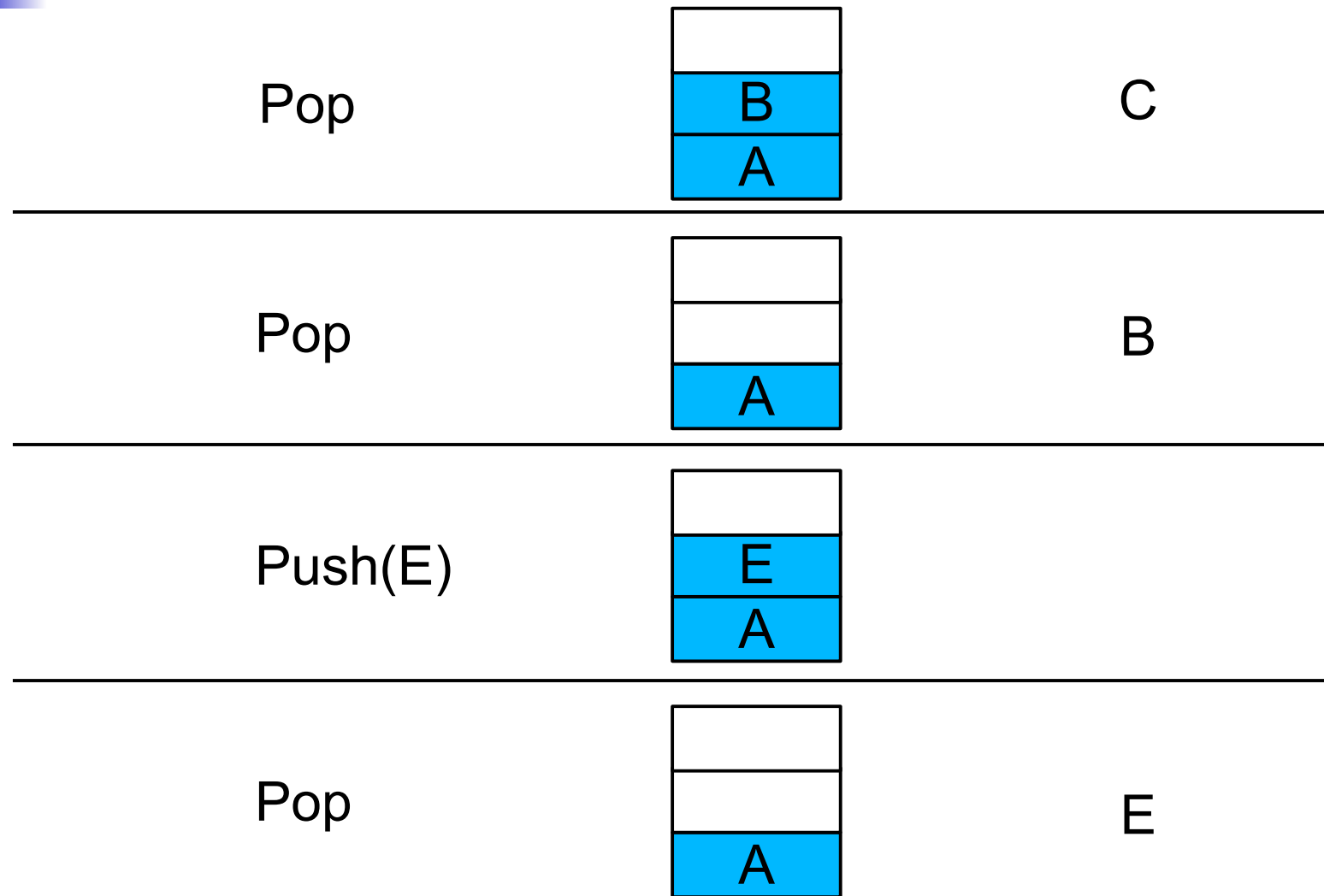


Push(D)



FULL

堆疊範例





Assignment 12

- 實作 class StackType, 需要俱備以下操作
 - bool IsEmpty() 決定堆疊是否為空
 - bool IsFull() 決定堆疊是否爆滿
 - void Push(char item) 將字元放入堆疊
 - void Pop() 移除堆疊中最上方的資料
 - char Top() 傳回堆疊中最上方的資料
- Push(), Pop(), Top() 需處理例外狀況
 - 使用標準函式庫提供的例外基礎類別來撰寫自己的例外類別
- 堆疊用於儲存 char 型別的資料



Assignment 12

- 撰寫主程式測試字串中括號是否對稱
 - 使用堆疊 (Stack) 來實作
 - 用以下範例作為測試資料
 - 第七個範例會造成堆疊爆滿，需要做例外處理，顯示異常訊息

#	字串	預期結果
1	(x)	Well formed
2	(x{})	Well formed
3	(x{x[]})	Well formed
4	(x[x])	Not well formed
5	(x(x[]x	Not well formed
6	(xx)]	Not well formed
7	(x((((Not well formed



括號是否對稱

■ 定義

- Opening Symbol:characters "(", "{", or "["
- Closing Symbol:characters ")", "}", or "]"

■ 演算法

- 字串狀態初始設定為平衡
- 讀入一個字元
- 如果目前字元是 opening symbol，將字元放入 (Push) 堆疊
- 或者如果目前字元是 closing symbol 然後：
 - 檢查堆疊是否為空，如果是空，然後因為沒成對的 opening symbol 所以字串狀態設為不平衡，迴圈結束
 - 或者，
 - 取得 (Top) 堆疊最上方的字元
 - 彈出 (Pop) 堆疊最上方的字元
 - 比較目前字元與最上方的字元，如果兩者成對，則表示字串狀態仍然是平衡的，繼續執行迴圈（讀入下一個字元），如果不成對，則表示字串狀態不平衡，結束迴圈
- 當讀入字元為 '\n'，則迴圈結束，如果字串狀態仍是平衡而且堆疊是空，然後印出 "Well formed"
- 若字串狀態不是平衡的則印出 "Not well formed"



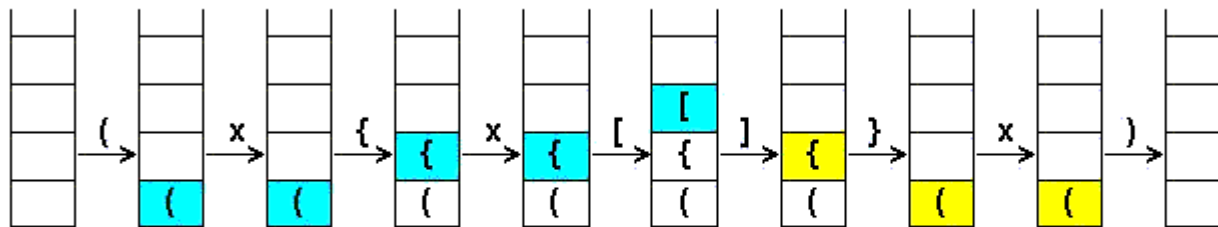
案例

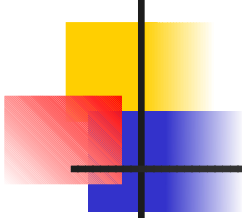
■ 字符串 (x{x[] }x)

```
'('      : Push (  
'x'      : Ignore  
'{'      : Push {  
'x'      : Ignore  
'['      : Push [  
' ]'     : Get top of stack, openSymbol='['  
          Pop  
          Compare if '[' matches ' ]'  
' }'     : Get top of stack, openSymbol='{'  
          Pop  
          Compare if '{' matches ' }'  
'x'      : Ignore  
' )'     : Get top of stack, openSymbol='(  
          Pop  
          Compare if '(' matches ' )'  
'\n'     : Print expression is well-formed
```


案例

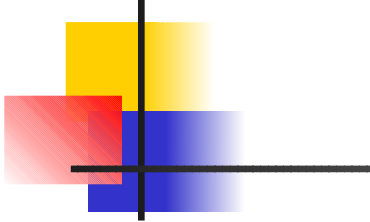
■ 字符串 $(x\{x[]\}x)$



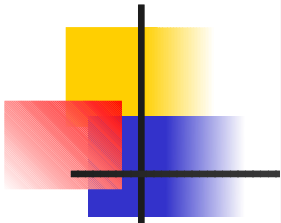


使用標準函式庫提供的例外基礎類別來撰寫自己的例外類別

```
1 // Fig. 27.1: DivideByZeroException.h
2 // Class DivideByZeroException definition.
3 #include <stdexcept> // stdexcept header file contains runtime_error
4 using std::runtime_error; // standard C++ library class runtime_error
5
6 // DivideByZeroException objects should be thrown by functions
7 // upon detecting division-by-zero exceptions
8 class DivideByZeroException : public runtime_error
9 {
10 public:
11     // constructor specifies default error message
12     DivideByZeroException::DivideByZeroException()
13         : runtime_error( "attempted to divide by zero" ) {}
14 }; // end class DivideByZeroException
```



```
1 // Fig. 27.2: Fig27_02.cpp
2 // A simple exception-handling example that checks for
3 // divide-by-zero exceptions.
4 #include <iostream>
5 using std::cin;
6 using std::cout;
7 using std::endl;
8
9 #include "DivideByZeroException.h" // DivideByZeroException class
10
11 // perform division and throw DivideByZeroException object if
12 // divide-by-zero exception occurs
13 double quotient( int numerator, int denominator )
14 {
15     // throw DivideByZeroException if trying to divide by zero
16     if ( denominator == 0 )
17         throw DivideByZeroException(); // terminate function
18
19     // return division result
20     return static_cast< double >( numerator ) / denominator;
21 } // end function quotient
22
23 int main()
24 {
25     int number1; // user-specified numerator
26     int number2; // user-specified denominator
27     double result; // result of division
28
29     cout << "Enter two integers (end-of-file to end): ";
```



```
30
31 // enable user to enter two integers to divide
32 while ( cin >> number1 >> number2 )
33 {
34     // try block contains code that might throw exception
35     // and code that should not execute if an exception occurs
36     try
37     {
38         result = quotient( number1, number2 );
39         cout << "The quotient is: " << result << endl;
40     } // end try
41
42     // exception handler handles a divide-by-zero exception
43     catch ( DivideByZeroException &divideByZeroException )
44     {
45         cout << "Exception occurred: "
46             << divideByZeroException.what() << endl;
47     } // end catch
48
49     cout << "\nEnter two integers (end-of-file to end): ";
50 } // end while
51
52 cout << endl;
53 return 0; // terminate normally
54 } // end main
```