Màxim d'una pila — X49526_ca

Preliminars
En aquest exercici extendrem la classe `Stack` suposant que el tipus `T` dels elements de la pila té definits els operadors de comparació `<`, `<=`, `==`, `>`, `>=`, `!=`, és a dir, que dues variables de tipus `T` es poden comparar sempre. (des d'un punt de vista algebraic, direm que hi ha un **ordre total** en els conjunt de valors de tipus `T`). També suposem que una variable `x` de tipus `T` té definit l'operador d'assignació =.
Com que en aquest exercici **només** instanciarem `Stack` de tipus `int`, totes dues coses ja estan garantides. Per tant, no us ha d'amoïnar.
Exercici
Implementeu un nou mètode de la classe Stack que retorni el **màxim** de tots els elements continguts a la pila.
D'entre els fitxers que s'adjunten en aquest exercici, trobareu `stack.old.hpp`, a on hi ha una implementació de la classe genèrica `Stack`. En primer lloc, haureu de fer:

```
cp stack.old.hpp stack.hpp
```

A continuació, haureu de buscar dins `stack.hpp` les següents línies:

```
/*
 * Pre:  Sigui [a1,...,an] el contingut actual de la pila des
 * del fons fins al top.
 * Post: Retorna el màxim de a1+...+an.
 * Descomenteu les següents tres línies i implementeu el mètode:
 */

// T maxim()
// {
// }
```

Descomenteu les tes línies que s'indiquen i implementeu el mètode. Potser necessitareu modificar més coses de la classe depenent de quin enfocament seguiu. Aquí us en recomanem dos:

- **Enfocament senzill i ineficient:** Una implementació senzilla consisteix en fer un mètode que cada vegada que el cridem recorri tota la pila i en calculi i retorni el màxim. Això hauria de ser suficient per a superar els jocs de proves públics, però no els privats.
- **Enfocament eficient:** Consisteix en crear una variable que contingui el màxim de la pila, i que cada vegada que fem un `push` i un `pop` se'n modifiqui el contingut (si calgués).

  Cal tenir en compte el fet que a la pila hi poden haver repetits, i potser l'element màxim també està repetit. Per tant, possiblement, apart de tenir una variable que indiqui quin és l'element màxim, també caldrà tenir una variable que indiqui quantes vegades és a la pila.

Finalment, en algun cas, quan feu un pop, inevitablement caldrà fer un recorregut per tota la pila per a saber quin és el màxim i quantes vegades hi apareix, però globalment, seran menys vegades que amb la primera estratègia. Especialment, penseu que si en una pila no hi fem cap `pop`, aleshores aquesta estratègia serà molt més eficiant que l'anterior.

D'entre els fitxers que s'adjunten a l'exercici també hi ha `program.cpp` (programa principal) i `Makefile` per a compilar. Per a pujar la vostra solució, heu de crear el fitxer `solution.tar` així:

```
tar cf solution.tar stack.hpp
```

Entrada

L'entrada del programa és una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre la pila:

```
push x (x és de tipus int)
pop
top
print
maxim
```

Se suposa que la seqüència d'entrada serà correcta (sense `pop` ni `top` ni `maxim` sobre pila buida).
El programa principal que us oferim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe pila. Només cal que implementeu el mètode abans esmentat.

Sortida

Per a cada instrucció `top`, s'escriurà el top actual de la pila, per a cada instrucció `print`, s'escriurà el contingut de la pila, i per a cada instrucció `maxim`, s'escriurà el màxim dels elements de la pila. El programa que us oferim ja fa això. Només cal que implementeu el mètode `maxim`.

Exemple d'entrada 1

```
push 10
top
print
maxim
push 20
top
print
maxim
push 30
top
print
maxim
pop
top
print
maxim
push 31
top
print
maxim
push -40
top
print
maxim
pop
pop
maxim
pop
push 10
top
print
maxim
push 20
```

```
top
print
maxim
push 30
top
print
maxim
pop
pop
maxim
```

Exemple de sortida 1

```
top: 10
print: 10
màxim: 10
top: 20
print: 10 20
màxim: 20
top: 30
print: 10 20 30
màxim: 30
top: 20
print: 10 20
màxim: 20
top: 31
print: 10 20 31
màxim: 31
top: -40
print: 10 20 31 -40
màxim: 31
màxim: 20
top: 10
print: 10 10
màxim: 10
top: 20
print: 10 10 20
màxim: 20
top: 30
print: 10 10 20 30
màxim: 30
màxim: 10
```

Exemple d'entrada 2

```
push 3
push -10
push 15
push 17
push 20
print
maxim
push 18
top
pop
pop
print
maxim
push 12
push -7
pop
top
pop
push 19
print
maxim
push -16
print
maxim
push 0
print
maxim
```

```
push -4
push 12
push -19
pop
pop
top
push -4
print
maxim
top
pop
push 10
push 5
top
print
maxim
```

Exemple de sortida 2

```
print: 3 -10 15 17 20
màxim: 20
top: 18
print: 3 -10 15 17
màxim: 17
top: 12
print: 3 -10 15 17 19
màxim: 19
print: 3 -10 15 17 19 -16
màxim: 19
print: 3 -10 15 17 19 -16 0
màxim: 19
top: -4
print: 3 -10 15 17 19 -16 0 -4 -4
màxim: 19
top: -4
top: 5
print: 3 -10 15 17 19 -16 0 -4 10 5
màxim: 19
```

Exemple d'entrada 3

```
push -1
push -1
push -1
push -1
push 0
push 0
push 1
push 1
print
maxim
pop
print
maxim
pop
print
maxim
pop
print
maxim
pop
print
maxim
pop
print
maxim
pop
print
maxim
pop
print
maxim
```

Exemple de sortida 3

```
print: -1 -1 -1 -1 0 0 1 1
màxim: 1
print: -1 -1 -1 -1 0 0 1
màxim: 1
print: -1 -1 -1 -1 0 0
màxim: 0
print: -1 -1 -1 -1 0
màxim: 0
print: -1 -1 -1 -1
màxim: -1
print: -1 -1 -1
màxim: -1
print: -1 -1
màxim: -1
print: -1
màxim: -1
```

Observació

Avaluació sobre 10 punts: (Afegiu comentaris si el vostre codi no és prou clar)

- Solució lenta: 4 punts.
- solució ràpida: 10 punts.

Entenem com a solució lenta una que és correcta i capaç de superar els jocs de proves públics. Entenem com a solució ràpida una que és correcta i capaç de superar els jocs de proves públics i privats.
IMPORTANT: Òbviament, a dins del mètode (o mètodes) que implementeu, no podeu fer servir els mètodes que ja té la pila ja té implementats. A més, us haureu d'assegurar que

els mètodes que ja té implementats funcionen correctament després dels canvis que haureu aplicat a la classe `Stack`.

Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:14:29.662Z