FONDAMENTI DI PROGRAMMAZIONE

PROVA PRATICA 29 GENNAIO 2020

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Un PuzzleBobble rappresenta una versione semplificata del celebre videogioco. Lo schema di gioco è composto da una matrice di 6 colonne e 10 righe di caselle. Ogni casella può essere libera oppure occupata da una bolla, che a sua volta può essere di vari colori. Implementare le seguenti operazioni che possono essere effettuate su un PuzzleBobble:

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

✓ PuzzleBobble pb;

Costruttore di default che inizializza un PuzzleBobble inizialmente vuoto di bolle.

✓ cout << pb;</pre>

Operatore di uscita per il tipo PuzzleBobble. Lo schema viene stampato secondo il seguente formato:

I bordi dello schema sono rappresentati da caratteri '=' e '|'. Una casella vuota è rappresentata da uno spazio, mentre una bolla è rappresentata dalla lettera maiuscola corrispondente al suo colore: 'R' per rosso, 'G' per verde, 'B' per blu, 'Y' per giallo.

✓ pb.fire(i,color);

Funzione che spara una bolla di colore color dal bordo inferiore dello schema lungo la colonna di indice i. Il colore è specificato da uno dei caratteri maiuscoli 'R', 'G', 'B', 'Y'. L'indice i della colonna è un numero da 0 (prima colonna a sinistra) a 5 (ultima colonna a destra). La bolla scorre dal basso verso l'alto e si ferma "attaccandosi" sotto la prima bolla che trova su quella colonna, oppure sotto al limite superiore dello schema se non trova bolle. Per esempio, se nello schema visualizzato sopra viene chiamata la funzione pb.fire(2,'Y'), lo schema risultante sarà il seguente:

Le bolle non possono attaccarsi fuori dello schema. Quindi una chiamata che dovesse attaccare una bolla sotto il bordo inferiore dello schema non ha effetto. La funzione deve essere concatenabile, quindi deve essere possibile scrivere pb.fire(2,'Y').fire(1,'B'). Se uno degli input non è valido, la funzione non ha effetto.

✓ (int)pb;

Operatore di conversione a intero per il tipo PuzzleBobble. Restituisce l'altezza del blocco di bolle nello schema, cioè il numero di righe che vanno dal bordo superiore fino all'ultima riga in basso che contiene almeno una bolla. Per esempio, nello schema visualizzato sopra l'operatore restituirà 3.

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

✓ pb.fire(i,color); (versione avanzata della precedente versione)

Modificare il metodo pb. fire () di modo che, se la bolla appena attaccata forma file di bolle dello stesso colore orizzontali e/o verticali il cui numero complessivo di bolle (considerando sia l'eventuale fila verticale che l'eventuale fila orizzontale) è maggiore o uguale a 3, tali bolle scoppiano e spariscono dallo schema. (Nota che questo potrebbe rendere alcune bolle non più attaccate a niente. Tali bolle rimangono nello schema e NON devono sparire.) Per esempio, se nello schema visualizzato precedentemente viene chiamata la funzione pb. fire (1, 'R'), lo schema risultante sarà il seguente:

Nota che 5 bolle rosse sono scoppiate e sparite dallo schema, mentre due bolle gialle sono rimaste non attaccate a niente. La bolla rossa in alto a sinistra non è scoppiata in quanto non faceva parte né della fila verticale né di quella orizzontale formate dalla bolla appena attaccata.

✓ pb.scroll();

Metodo che fa scorrere il bordo superiore dello schema di una riga verso il basso, inserendo nella riga più alta una barra rappresentata da caratteri '='. Di conseguenza, il numero di righe disponibili per le bolle diminuisce di uno. Per esempio, se nello schema visualizzato sopra viene chiamata la funzione pb.scroll(), lo schema risultante sarà il seguente:

```
|====== |
|R BY G|
| G|
|Y Y |
| |
| |
```

Le bolle non possono uscire dal limite inferiore dello schema. Quindi una chiamata che dovesse fare uscire alcune bolle non ha effetto. La funzione deve essere concatenabile.

✓ pb.compact();

Metodo che compatta ogni colonna di bolle verso l'alto, eliminando le caselle vuote tra bolla e bolla. Per esempio, se nello schema visualizzato sopra viene chiamata pb.compact (), lo schema risultante sarà:



Mediante il linguaggio C++, realizzare il tipo di dato astratto **PuzzleBobble**, definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo string, il tipo vector, il tipo list, ecc. **Gestire le eventuali situazioni di errore**.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Test costruttore
Test funzione fire
|RRYY
|B Y
Test operatore int
Altezza: 3
--- SECONDA PARTE ---
Test funzionalita' scoppio bolle
|RR
R Y
   G
    G
Test funzione scroll
|RR
|R Y
Test funzione compact
|RR Y
|R
   G
```

Note per la consegna:

Affinché l'elaborato venga considerato valido, il programma deve produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato solo se lo studente avrà completato l'autocorrezione del proprio elaborato.

In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.