Programmazione C++ appello Gennaio A.A. 2024/2025

Andrea Vasciminno

Università degli studi di Milano Bicocca
a.vasciminno@campus.unimib.it
Matricola:904899

1 Introduzione

Per l'implementazione del progetto è richiesto lo sviluppo progettazione e realizzazione di una classe generica che implementa uno stack di elementi di tipo T. Il quale non può essere sviluppato utilizzando una struttura dati a lista puntata

2 Scelte d'implementazione

2.1 Variabili interne alla classe

La struttura dati scelta per l'implementazione dello stack, non potendo utilizzare una lista, è ricaduta sull'utilizzo di un array **_stack** contenente elementi templati di tipo T. Per una corretta gestione della struttura a pila sono poi state utilizzate ulteriori due variabili.
La prima è la variabile **_top** che viene utilizzata per mantenere il controllo sull'elemento in cima allo stack, in modo da poter effettuare nel minor tempo possibile le operazioni di accesso alla cima dello stack. Se _top ha valore -1 allora lo stack è vuoto.
La seconda variabile **_size** viene utilizzata per conoscere il numero di celle di memoria allocate per lo stack, in modo da poter gestire possibili casi di overflow.

2.2 Acquisizione ed inserimento nella classe

L'acquisizione e l'inserimento dei dati dall'esterno della classe è permessa solo grazie ai metodi pubblici **push()** e **pop()**.
Il metodo push, in caso di spazio disponibile, inserisce un nuovo elemento in cima allo stack e aggiorna la variabile _top aumentandola di 1, in caso contrario genera eccezione.
Il metodo pop, in caso di elementi già allocati, preleva l'elemento in cima allo stack e aggiorna la variabile _top decrementandola di 1, in caso contrario genera eccezione.

2.3 Stampa della classe

Per stampare la classe bisogna invocare il metodo pubblico **print().** Il metodo non utilizza gli iteratori per stampare, poiché quest'ultimi, come da specifiche richieste da progetto, sono sviluppati per scorrere lo stack dal fondo fino alla cima; quindi, garantirebbero una visualizzazione inversa da quello che è l'ordine consueto di stampa.
Per questa motivazione la stampa è stata sviluppata utilizzando uno stack di appoggio da cui si effettua una pop per ogni elemento, andando così a mantenere ordine logico anche nella stampa a schermo.

2.4 Iteratori e costruttori

Nel costruttore fondamentale la scelta è ricaduta sull'utilizzo di un parametro di default per la size. Quindi, al momento dell'inizializzazione non viene passato nessun parametro la size d'inizializzazione dello stack sarà uguale a 10 elementi.
Gli iteratori sono di due tipi, iterator e const_iterator, per entrambi sono definiti due metodi, begin ed end per gli iteratori, cbegin e cend per i const iterator.

2.5 Funzioni aggiuntive

Per la funzione riempiStack viene ciclato lo stack da riempire per capire la quantità di elementi. Una volta compresa, se > 0, lo stack viene svuotato e solo dopo riempito nuovamente

La funzione filter_out viene templata per avere un predicato generico su cui effettuare il controllo. Lo stack viene svuotato e riempito due volte per mantenere l'ordine precedente degli elementi.

3 QT

Per quanto riguarda la parte di QT, la ricerca della directory da monitorare funziona tramite esplora risorse all'avvio del programma. Il file di log, se già esistente, viene cercato all'interno della directory selezionata; il quale non viene visualizzato, per scelta progettuale, all'interno del programma.
Nel caso non venga inserita nessuna directory il programma termina l'esecuzione
È possibile effettuare il salvataggio del log della sessione tramite apposito pulsante, ma in caso questo non venga fatto il programma effettua comunque un salvataggio automatico a chiusura.
L'ambiente di sviluppo è MacOS