Relazione progetto C++ Febbraio 2023

Nome : Niccolò
Cognome: Balzarotti
Matricola : 852003
Mail : n.balzarotti2@campus.unimib.it

INTRODUZIONE

Dopo aver valutato il problema ho deciso di implementare la classe stack utilizzando i vettori.

Il vettore contiene elementi generici del tipo della classe.

TIPI DI DATI

Gli elementi contenuti nel vettore sono elementi generici del tipo dello stack.

IMPLEMENTAZIONE E METODI IMPLEMENTATI

La classe stack dispone di questi attributi:
Un puntatore generico al vettore che contiene gli elementi dello stack, il puntatore all'ultimo elemento inserito nello stack (_top) e la dimensione dello stack.

All'interno della classe stack ho implementato 5 metodi fondamentali:
Il costruttore di default, l'operatore di assegnamento, il costruttore secondario, il copy constructor e infine il distruttore.

Gli ultimi tre metodi utilizzano una funzione chiamata clear() che permette di svuotare completamente lo stack

Basandomi sulle richieste della traccia di esame ho deciso di implementare altri metodi:

Il metodo clear() che viene utilizzato all'interno dei due costruttori secondari, copy constructor, distruttore e del metodo fill permette di svuotare lo stack eliminando tutti i valori presenti al suo interno e ripristinando gli attributi della classe permettendo una corretta gestione della memoria.

Il metodo size() che restituisce la dimensione di uno stack.

Il metodo is_empty() che restituisce true se uno stack è vuoto, false altrimenti.

Il metodo is_full() che restituisce true se uno stack è pieno,false altrimenti.
Per questi due metodi esiste un caso particolare, se ho uno stack di dimensione (size) 0 questo risulterà sia empty perchè non contiene elementi e sia full perchè la sua dimensione non permette di aggiungere altri elementi.

Il metodo push che permette di inserire un elemento in cima allo stack, nel caso in cui si provasse ad inserire un elemento in uno stack pieno viene lanciata un eccezione (stack overflow).

Il metodo pop che permette di rimuovere un elemento dalla cima dello stack, nel caso in cui si provasse a rimuovere un elemento da uno stack vuoto viene lanciata un eccezione (stack underflow).

Il metodo top che restituisce l'elemento in testa allo stack senza però rimuoverlo e anche in questo caso se si prova a restituire il primo elemento di uno stack vuoto viene lanciata un eccezione (stack underflow).

Un costruttore secondario per stack che data una coppia di iteratori begin e end che puntano rispettivamente all'inizio e alla fine di una sequenza di elementi viene creato uno stack e riempito con gli elementi di questa sequenza.

Il metodo stored_elements() che permette di restituire il numero di elementi presenti nello stack.

Il metodo fill che data una iteratori di puntatori begin e end che puntano rispettivamente all'inizio e alla fine di una sequenza di elementi svuota lo stack da tutti gli elementi che contiene e lo riempie con gli elementi della sequenza fino al completamento della size.

Un altro metodo usato è quello per la ridefinizione dell'operatore di stream << per lo stack (operator<<), questo metodo permette la stampa dello stack.

Infine la classe stack implementa un const iterator di tipo forward.

L'operator++ è stato ridefinito per poter leggere il vettore al contrario (il puntatore viene decrementato) essendo che l'ultimo elemento inserito è quello in cima allo stack.

MAIN

Ho utilizzato il main per effettuare tutti i test sullo stack, in particolare ho effettuato test sullo stack utilizzando tipi interi, stringhe e infine tipi custom.

Il primo test effettuato è stato quello per lo stack di tipi custom, in particolare ho deciso di testare i metodi su stack composti da Punti (punto(x,y)).

Ho implementato nel main la struct punto dopo di che ho testato tutti i metodi principali e il const iterator tramite delle assert, successivamente ho eseguito gli stessi test per il tipo intero e stringa.

Inoltre ho anche testato la constness dei metodi const.