Radaelli Simone | 902814 | s.radaelli37@campus.unimib.it

PROGETTO DI PROGRAMMAZIONE C++

Progetto C++

Il progetto richiede l'implementazione di uno stack con classe generica.

Per questo ho deciso di usare un file .hpp il quale facesse sia la dichiarazione che l'implementazione di ogni funzione.

Essendo, per consegna, impossibilitato ad utilizzare una lista dinamica ho deciso di utilizzare un array anch'esso dinamico. Questa scelta è dovuta principalmente dalla semplicità nell'accedere ai dati al suo interno e nell'altrettanta facilità nello scorrere dal primo all'ultimo elemento

Dati membro

I dati membro sono: -_stack: puntatore all'array dinamico.

-_size: rappresenta la dimensione fissa dell'array

-_top: è un valore intero che utilizzerò per rappresentare la cima del mio stack.

Per rendere ciò possibile, ho deciso di inizializzare questo valore a -1

e ad ogni push, prima dell'assegnamento, viene incrementato di uno in modo tale

che _stack[_top] possa puntare esattamente all'ultimo valore all'interno

dello stack.

set_zero

Nonostante non fosse stato esplicitamente richiesto, ho deciso di implementare un metodo chiamato set_zero() il quale potesse appunto "resettare" i miei dati membro.

push & pop

Dopo aver realizzato i costruttori assieme al distruttore, ho cercato di implementare le due funzioni principali di uno stack, cioè Push e Pop.

In entrambi i metodi, per prima cosa viene fatto un controllo sulla _size che lancia la sua rispettiva eccezione: nel caso della push, il controllo viene fatto per verificare se lo stack è pieno. Caso contrario invece quello della pop dove viene controllato se lo stack è vuoto e quindi non ha elementi da estrarre.

Ci tengo a specificare che nella funzione pop, l'elemento non viene fisicamente eliminato dallo stack ma semplicemente viene ritornato il valore e poi decrementata la variabile top. Così facendo, viene simulata la sua estrazione e nel caso ci fosse poi una push sullo stack, sovrascriverebbe il dato "estratto".

Load_from_range

Per quanto riguarda il metodo richiesto nel punto 3 che ho chiamato "load_from_range", vede come prima operazione la chiamata a clear. Ho deciso di chiamare questo metodo senza un controllo prima, cosi che a prescindere venga svuotato lo stack.

Per quanto riguarda il resto della funzione, ho fatto un controllo sulla dimensione dello stack e sulla quantità di dati che si desidera copiare all'interno dello stack stesso.

Se la quantità di dati dovesse superare la grandezza dello stack, verrà lanciata un'eccezione.

Print()

Riguardante la print, ho deciso di stampare lo stack in verticale piuttosto che come un classico array.

Iterator & Const_Iterator

Per l'implementazioni di iterator & const_iterator, ho utilizzato l'interfaccia dei Forward_iterator, essendo che il nostro stack debba essere visitato in un'unica direzione e non in entrambe oppure con posizioni casuali.

Oltre a questo, specifico che il begin sarà il primissimo elemento inserito all'interno dello stack mentre end rappresenterà la cima.