Progetto per programmazione e amministrazione di sistema (19/02/2017)

Studente: Emanuele Motto

Matricola: 806756

email: e.motto1@campus.unimib.it

Presentazione progetto

Il progetto cbuffer mira all'implementazione e al test di un buffer circolare generico di elementi di tipo T con capienza fissata a costruzione.

Le caratteristiche principali del progetto comprendono (oltre alla creazione del buffer circolare) la possibilità di aggiungere elementi alla fine del cbuffer e rimuoverli dalla coda.
Si sottolinea la necessità di sovrascrivere gli elementi del buffer a partire dal più vecchio nel caso la capacità massima venga raggiunta.

Vengono implementate inoltre varie funzioni di utilità generale, oltre alla possibilità di generare un buffer di una certa grandezza con due iteratori generici in input e una funzione "evaluate_if" che, presi un buffer e una funzione unaria, esamina la funzione su tutti gli elementi del buffer.

Implementazione

Il cbuffer viene implementato come un costrutto con 4 attributi:

- _buffer = puntatore al buffer degli elementi
- _size = capienza del cbuffer
- _inizio = indicatore di dove il buffer effettivamente inizia (differisce dall'indice 0)
- _fine = indicatore di dove il buffer effettivamente finisce (differisce da _size-1)

Essendo un buffer circolare che sovrascrive i suoi elementi più vecchi ovviamente gli indici di inizio e fine dovranno essere spostati ogniqualvolta avviene una sovrascrizione oppure quando il buffer si svuota completamente.
Considerando per esempio un buffer di capienza 3, i suoi indici saranno 0, 1 , 2 e si muoveranno nel modo seguente:

nessun elemento: _inizio = -1 _fine = -1

1 elemento: _inizio = 0 _fine = 0

2 elementi: _inizio = 0 _fine = 1

3 elementi _inizio = 0 _fine = 2

4 elementi _inizio = 1 _fine = 0

ecc…

Questa logica viene mantenuta in tutta l'implementazione e verrà applicata all'inserimento di nuovi elementi/nella rimozione. [effettuando chiamate a push_back(elemento)/pop()]

Sono implementati inoltre le classi iterator e const_iterator come forward iterator (++, ==, !=, *, →) che permettono l'accesso in lettura (const_iterator) e in lettura/scrittura(iterator) agli elementi della classe cbuffer.
Implementati i metodi standard begin() e end() che restituiscono due iteratori.
Il metodo begin() restituisce l'iteratore che "punta" all'elemento più vecchio presente nel cbuffer, mentre il metodo end() restituisce un iteratore all'elemento dopo l'ultimo elemento presente nel cbuffer.
l'iteratore ritornato della funzione end() non punta a nessun elemento del cbuffer e non è deferenziabile con l'operatore "*" (a differenza degli altri iteratori). Questo iteratore serve a denotare la fine logica della sequenza di elementi presenti nel cbuffer.

Composizione del progetto

Il progetto cbuffer è composto da 4 file principali: cbuffer.h, main.cpp, Doxyfile, MakeFile.

- cbuffer.h: In questo file troviamo l'implementazione della classe cbuffer con tutti I suoi metodi, le ridefinizioni degli operatori e gli iteratori.
- main.cpp: nel file main.cpp troviamo la funzione main necessaria a far partire il programma, che è essenzialmente composto da una serie di test atti a verificare il funzionamento di tutte le funzionalità della classe cbuffer.h
- Doxyfile: file per la generazione della documentazione doxygen con output in HTML
- MakeFile: file necessario alla compilazione grazie al comando make