Modifiche Progetto Programmazione per la Fisica

Vanessa Lenzotti, Lorenzo Mensi, Stef Gaudenzi Lootens, Fabio Ehrenhofer

Settembre 2021

Introduzione

Questo documento ha lo scopo di illustrare brevemente le modifiche apportate al progetto finale nella forma in cui è stato consegnato allo scorso appello (è quindi un appendice alla relazione precedente, che resta valida).
Il link alla nostra repository github è https://github.com/FabioEhr/final-project.

Modifiche comuni a tutti i programmi

2.1 Modifiche a *usuful_func.hpp*

Completa rivisitazione della gestione del sistema di input: sostituzione delle funzioni *string_to_char, string_to int e string_to_decimal* con *validate_int, validate_double e validate_char* per garantire un funzionamento corretto grazie all'utilizzo della standard library. Tali cambiamenti sono contenuti nei file denominati *"useful_func.hpp"* contenuti nelle cartelle *sir_grid, sir_yae/source, sir_base* e *grid_base/source*.

Modifiche a *sir_base*

Nel *sir_base* è stata modificata l'implementazione di fondo dello *struct Condition*, infatti, ora l'utente non dovrà inserire la percentuale di infetti/suscettibili/guariti ma il valore assoluto. Dunque, anche tutta la gestione di input/output e la *exception handling* è stata modificata.

3.1 *main.cpp*

In questo file è stato trasformato il *for* che chiama la funzione *Print* in un algoritmo di tipo *for_each*.

3.2 *sir.hpp*

Tutto l'*exception handling* è stato spostato dal costruttore della classe Pandemic a quello di *Condition e Virus* in modo tale che la loro chiamata sia più completa. Inoltre è stato aggiunto *const* nelle *getter function*.

3.3 *Parser.cpp e parser.hpp*

In *createVirus()* sono state apportate tutte le modifiche necessarie al cambio di implementazione, per passare ai numeri assoluti della popolazione. Inoltre Print è stata trasformata in una *function object*.

3.4 Grafica

Implementazione del *grid_base* (celle immobili)

4.1 Contenuto di grid_base

Questo programma si ispira al gioco della vita, infatti è composto da una griglia quadrata in cui ogni quadratino rappresenta una persona. Le persone possono assummere uno di questi 3 stati: suscettibili, contagiate e guarite. Una persona contagiata può infettare con una certa probabilità(pari alla *contagiousness*) le celle che le stanno attorno, ma ha un tempo limitato perchè ad ogni turno c'è una certa probabilità(pari al *recovery_rate*) che guarisca.

4.2 Come compilare

Essendo il codice composto da numerose *translation units*, abbiamo utilizzato *Cmake* per facilitare la fase di compilazione. All'interno della cartella *grid_base* digitare i comandi:

1. cmake -S ./source -B ./build
2. cd build
3. make

che produrranno due eseguibili nella cartella *build* . Questi servono per: avviare la simulazione(*./simulate*) e verificare il corretto funzionamento del programma tramite *Doctest*, (*grid_test*). Equivalentemente è possibile compilare i 2 programmi in questo modo:

- g++ main.cpp parser.cpp worl.cpp -Wall -Wextra -fsanitize=address -std=c++17
  presets.cpp- Wall - Wextra –fsanitize=address
- g++ main.cpp parser.cpp world.cpp -Wall –Wextra –fsanitize=address -std=c++17

4.3 Contenuto di world.hpp

Inizialmente è presente l'*enum Cell* rappresentante gli stati che una persona della *grid* può assumere (suscettibile, infetta, guarita).
La *struct Virus* descrive le caratteristiche del morbo che andrà a colpire la griglia.
Contagiousness rappresenta la probabilità che un' infenzione avvenga fra una

persona contagiata e una suscettibile.
Recovery_rate invece è la probabilità che una persona infetta in questo turno guarisca.
La classe *World* contiene la griglia di persone *m_grid*, con i rispettivi metodi per accedere o settare la griglia intera o una specifica cella. I metodi per accedere ad una cella sono duplicati, perchè in un caso si vuole permettere di accedere alla persona esprimendo le coordinate della griglia, in un altro scrivendo la coordinata nel vettore unico in cui sono contenute le persone.

4.4 *world.cpp*

Il file inizia con la definizione del metodo *evolve* che permette di far evolvere la griglia di uno step. Un dettaglio degno di attenzione è che all'interno di essa per decidere se una persona si infetta o guarisce viene estratto a sorte un numero reale.

4.5 *parser.hpp*

In questo file sono contenute tutte le dichiarazioni delle funzioni che poi verranno definite nel corrispondente file*.cpp*.

4.6 *parser.cpp*

La prima funzione che incontriamo è *createVirus()* che permette di inizializzare un virus.
Successivamente, si trova *insertGrid()* che consente di avere una griglia con il numero di infetti e guariti desiderato, sparsi in modo casuale per essa.
initialize_grid() serve per inserire tutti i valori che inizializzano una *grid*.
Infine, troviamo *printGrid()* che permette di stampare una griglia che indica gli stati delle persone con i simboli I=infected, S=suscettibles e R=recovered. Per i colori ho incluso *"termcolor.hpp"* prodotto da Ikalnyskyi.

4.7 *main.cpp*

Tutto il codice si trova in un *"try catch"* e stampa in continuazione la griglia fino a quando l'utente non chiude il programma con crtl+c.

4.8 *grid.test.cpp*

Con questi test si controlla che le varie funzioni del programma operino correttamente, in particolare *evolve()* e *hinsertGrid()*.

Modifiche al sir_grid (celle mobili)

5.1 Generazione di numeri casuali

Gli oggetti di tipo std::random_device e std::default_random_engine, utilizzati per la generazione di numeri casuali, sono stati inseriti all'interno di static

member function delle classi Person e Grid, in modo che non ne vengano creati di superflui, migliorando così l'efficienza.

5.2 Output sul terminale

La stampa su terminale è stata resa più espressiva mediante l'utilizzo della libreria header-only "termcolor" (si veda https://github.com/ikalnytskyi/termcolor), che permette di cambiare il colore del testo e/o dello sfondo.

5.3 Altre modifiche

-Correzione di alcune sviste (rimozione di ';' dove non necessari, return-type delle getter functions resi *const*).
-Sostituzione di for loop con range-for loop dove possibile.
-Rimossa la necessità di utilizzare *this*.
-Correzione dei comandi del preprocessore (*#include* mancanti)
-Sostituzione di "cascate" di *if-else* (dove la condizione è espressa da un *char* o un *enum*) con *switch* statement.
-Le funzioni *get_ map* e *draw_ map* sono state rese dei metodi della classe *Grid*, per semplificarne la gestione e migliorare la coerenza del codice.

Modifiche al *sir_ yae*

6.1 *Const e reference*

Aggiunta di *const* e reference (&) ove necessario per seguire le direttive illustrate a lezione.

6.2 *Switch case*

Sostituzione di *if* "a cascata" con *Switch statements* ove possibile per facilitare la lettura del codice e sfruttare l'ottimizzazione intrinseca.

6.3 Altre modifiche

Alcuni cambiamenti nella gestione dell'input-output (che includono l'utilizzo delle funzioni *validate()* menzionate prima), per evitare il più possibile comportamenti anomali e rendere la stampa più leggibile.