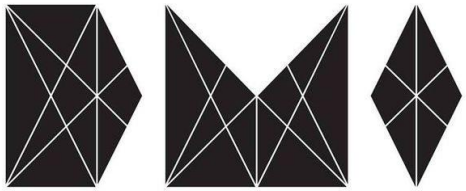


Università degli Studi di Catania
Dipartimento di Matematica e Informatica

"ARCHITETTURA DEGLI ELABORATORI", A.A. 2020/2021

DOCUMENTAZIONE PROGETTO

TITOLO PROGETTO

"SIMULAZIONE DEL FUNZIONAMENTO DI UNA MEMORIA CACHE"

REALIZZATO DA: SALVATORE ALFIO SAMBATARO

DESCRIZIONE DEL PROGETTO

Il progetto realizzato è un progetto didattico riguardante le memorie del calcolatore, in particolare la memoria Cache.

Esso è stato realizzato per facilitare la comprensione del funzionamento della memoria cache, e in particolare da una visione dettagliata del funzionamento dei diversi schemi di indirizzamento utilizzati dalle memorie cache.

Il programma infatti permette di caricare dei dati di tipo intero in "blocchi di memoria" e poi caricare questi ultimi in "posizioni di memoria cache", scegliendo quale tipo di indirizzamento simulare.

SPIEGAZIONE TEORICA

FUNZIONAMENTO DELLA CACHE

La cache è una memoria veloce ma piccola, interposta tra il processore e la memoria centrale. Il suo scopo è quello di velocizzare gli accessi alla memoria centrale, in quanto quest'ultima risulta essere troppo lenta rispetto al processore.

Il funzionamento della cache si basa sui cosiddetti "principi di località".

In particolare, distinguiamo i principi di località temporale e spaziale:

- Località temporale (istruzioni): *se una certa istruzione viene prelevata in un certo ciclo "i", con molta probabilità la stessa istruzione verrà ri-eseguita al ciclo "i+p", con p intero positivo relativamente piccolo.*

- Località spaziale (dati): *se un certo dato collocato ad un indirizzo "i" viene usato, con molta probabilità sarà usato anche il dato di indirizzo "i±p", con p intero positivo o negativo relativamente piccolo*.

  Poiché i dati e le istruzioni in memoria sono organizzati in blocchi, è anche possibile dire che "*se un blocco di parole di memoria (dati o istruzioni) viene usato dal processore, con molta probabilità in breve tempo e per più volte esso verrà usato nuovamente*" (principio di località spaziale-temporale)

SCHEMI DI INDIRIZZAMENTO

La dimensione della cache è molto ridotta rispetto alla dimensione della memoria centrale. Di conseguenza, non può esistere una corrispondenza biunivoca tra blocchi di memoria e posizioni di cache (ovviamente, ogni posizione di cache può ospitare un solo blocco di memoria).

Per questo motivo, è necessario introdurre i cosiddetti SCHEMI DI INDIRIZZAMENTO, utili ad assegnare a ciascun blocco di memoria centrale una (o in certi casi, più di una) posizione di cache.

Ne esistono di tre tipi:

1) ***INDIRIZZAMENTO DIRETTO***: ogni blocco di memoria è caricabile in una sola posizione di cache. In particolare, avendo una cache di N posizioni, il blocco di memoria *i-esimo* sarà caricabile esclusivamente nella posizione $i \ mod \ N$.

   Es. con una cache a 4 posizioni:

Blocco di memoria	Posizione cache
0	0 $(0 \ mod \ 4)$
2	2 $(2 \ mod \ 4)$
4	0 $(4 \ mod \ 4)$
9	1 $(9 \ mod \ 4)$

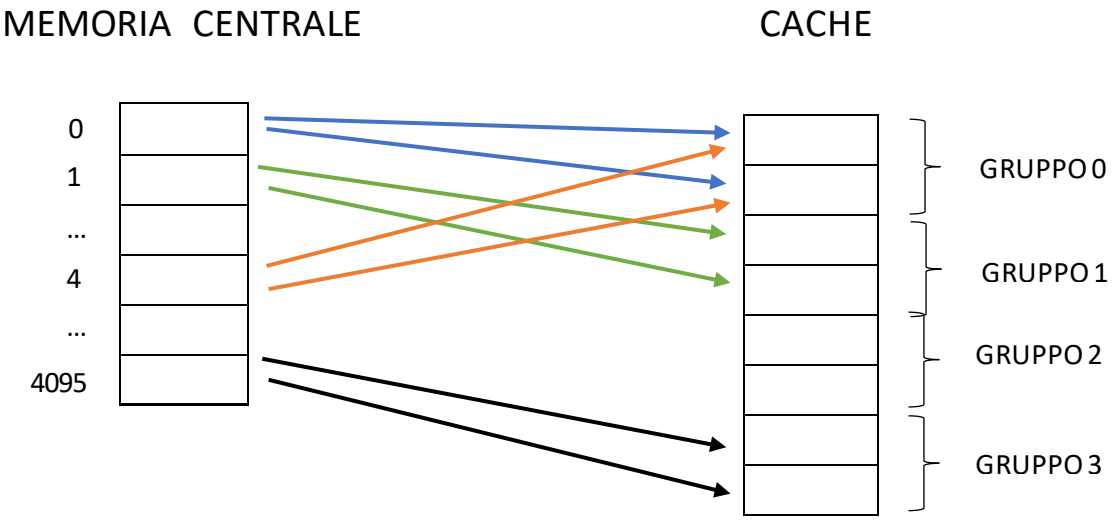
2) ***INDIRIZZAMENTO ASSOCIATIVO***: ogni blocco di memoria è caricabile in QUALSIASI posizione di cache.

   Es. con una cache a 8 posizioni:

Blocco di memoria	Posizione cache
0	0 , 1 , 2 , … , 7
3	0 , 1 , 2 , … , 7
…	0 , 1 , 2 , … , 7
i	0 , 1 , 2 , … , 7

3) ***INDIRIZZAMENTO ASSOCIATIVO A GRUPPI***: la cache viene divisa in GRUPPI, ognuno con lo stesso numero di posizioni. In particolare, avendo una cache divisa in K gruppi, ognuno dei quali è formato da N posizioni (in questo caso si parla di "*cache a N vie*"), il blocco di memoria i-esimo sarà caricabile in una qualunque delle N posizioni del gruppo $i \ mod \ K$.

   Es. cache con 8 posizioni divisa in 4 gruppi (ogni gruppo ha quindi 2 posizioni):



ALGORITMI DI SOSTITUZIONE

Poiché la memoria cache è molto più piccola della memoria centrale, accade spesso che, nel tentativo di caricare un certo blocco di memoria, le posizioni di cache in cui è possibile caricare quest'ultimo siano tutte occupate. In questo caso, vengono usati i cosiddetti ALGORITMI DI SOSTITUZIONE:

1) ***LRU*** (***L***east ***R***ecently ***U***sed): il blocco di memoria viene caricato nella posizione di cache USATA MENO RECENTEMENTE;

2) ***FIFO*** (***F***irst ***I***n ***F***irst ***O***ut): il blocco di memoria viene caricato nella posizione di cache CARICATA MENO RECENTEMENTE;

3) ***SOSTITUZIONE CASUALE***

N.B. Gli algoritmi di sostituzione sono utili nel caso in cui si usi l'indirizzamento associativo o associativo a gruppi, in quanto ogni blocco può essere caricato in più posizioni di cache;

STRUMENTI PER L'IMPLEMENTAZIONE

Per l'implementazione sono stati utilizzati:

- ***IDE "Dev C++",*** in cui sono inclusi:

  Editor per il codice
  Debugger
  Compilatore

- ***Librerie***:

  <iostream> : uso delle operazioni di input e output
  <cstdlib> e <ctime> : uso delle operazioni di generazione di numeri casuali
  <math.h> : uso di funzioni matematiche (es: $\log_2 n$)
  <windows.h> : cambio colore del testo della shell

IMPLEMENTAZIONE SOFTWARE

Per simulare il funzionamento della memoria centrale e della memoria cache del calcolatore, sono stati utilizzati:

- classe ADT “posCache”, in cui sono definiti attributi e metodi necessari alla rappresentazione astratta di una posizione della memoria cache.

  ATTRIBUTI:

  - **int*** dati: array di interi che rappresenta i valori contenuti in una generica posizione di cache
  - **bool** libera: valore booleano per indicare se la posizione è libera o meno;
  - **int** contFIFO: contatore utilizzato nell’implementazione dell’algoritmo di sostituzione “FIFO”;
  - **int** contLRU: contatore utilizzato nell’implementazione dell’algoritmo di sostituzione “LRU”;
  - **int** dim: numero di parole di una singola posizione di cache

  METODI:

  - posCache(…) : costruttore utile alla creazione di oggetti della classe “posCache”;

- int memoria[dimMemoria][dimBlocco]: matrice di interi che rappresenta la memoria centrale. È stato preferito l’uso di una matrice per permettere la memorizzazione di più parole all’interno di un unico blocco di memoria;

- posCache cache[dimCache]: array di oggetti di tipo “posCache”, che rappresenta la memoria cache;

- Funzioni di vario genere:
  - **void** colora (int color): permette di cambiare il colore del testo del terminale. Il nuovo colore è determinato dal parametro “color”;

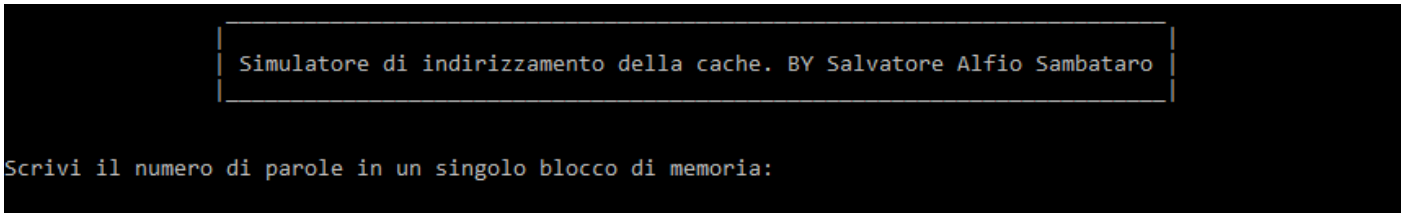
  - **bool** potenzaDiDue(int n): permette di verificare se il numero dato in input è una potenza di due; questa funzione è usata per “forzare” l’inserimento di una dimensione di memoria che sia, appunto, una potenza di due;

GUIDA UTENTE

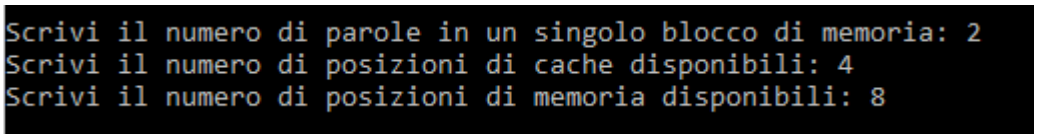
Per lanciare il programma **con sistema operativo Windows**, è sufficiente compilare il file “main.cpp” e lanciare il file con estensione “.exe” che sarà stato creato.

N.B. Potrebbero esserci problemi di compatibilità con altri sistemi operativi (es. Linux) a causa della presenza di alcune librerie compatibili esclusivamente con sistemi Windows, tra cui ad esempio <window.h>

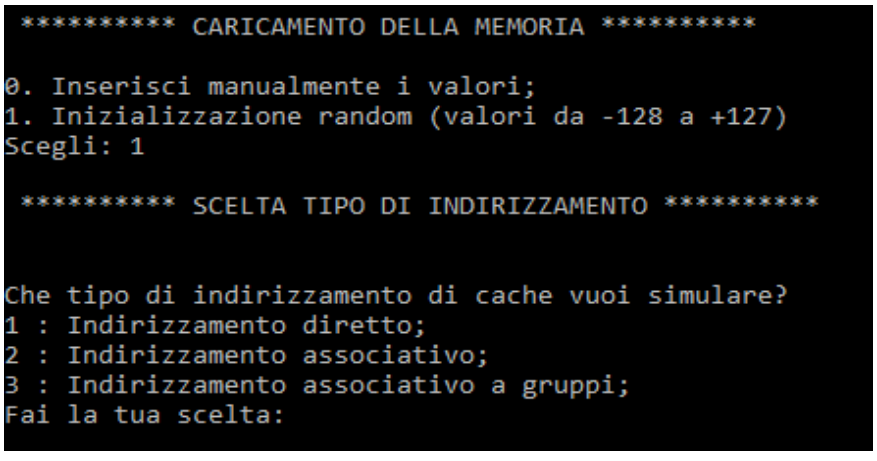
- All’avvio, il programma presenta la seguente schermata:



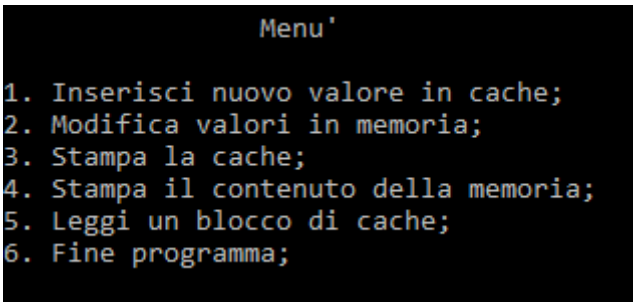
- A questo punto, sarà richiesto l’inserimento da tastiera delle dimensioni di un singolo blocco di memoria (numero di parole per blocco), numero di posizioni della memoria cache e numero di blocchi di memoria centrale. **È necessario inserire valori che siano potenze di 2.**



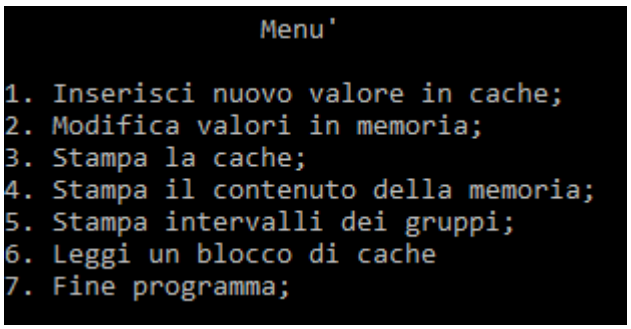
- Successivamente si devono caricare dei valori interi in memoria centrale. Per farlo, è possibile scrivere manualmente i valori o caricare valori casuali compresi tra -128 e +127. Una volta caricati i valori, si dovrà inoltre scegliere il tipo di indirizzamento di cache da simulare (se si sceglie l’indirizzamento associativo a gruppi, sarà richiesto il numero di gruppi da formare):



- Una volta scelto il tipo di indirizzamento, si apriranno tre diversi menù a seconda della scelta fatta:



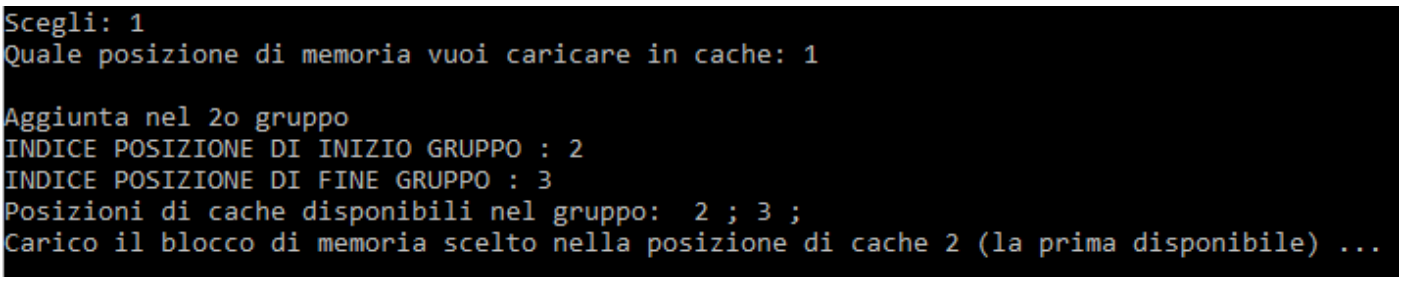
Diretto / Associativo



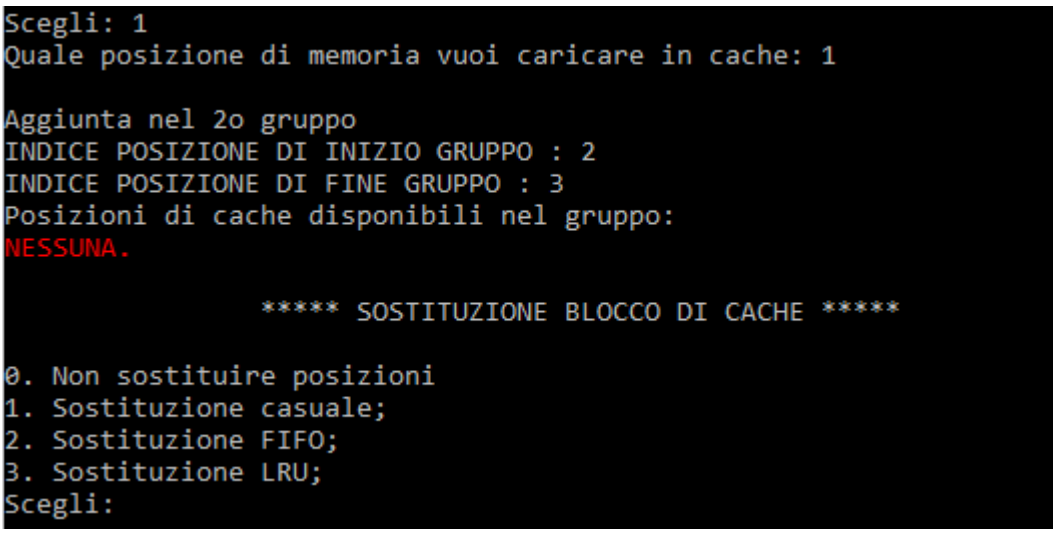
Associativo a gruppi

A scopo dimostrativo, creiamo memoria centrale e cache formate rispettivamente da 8 e 4 blocchi, in cui ogni blocco contiene 2 parole di memoria. Come tipo di indirizzamento per la cache, scegliamo l'indirizzamento associativo a gruppi.

1) ***INSERIMENTO DI UN NUOVO VALORE IN CACHE***: permette di inserire un blocco di memoria nella corretta posizione di cache. Nel caso si usi l'indirizzamento associativo / associativo a gruppi, e le posizioni di cache in cui caricare il blocco siano tutte occupate, è possibile usare uno dei tre algoritmi di sostituzione elencati precedentemente.

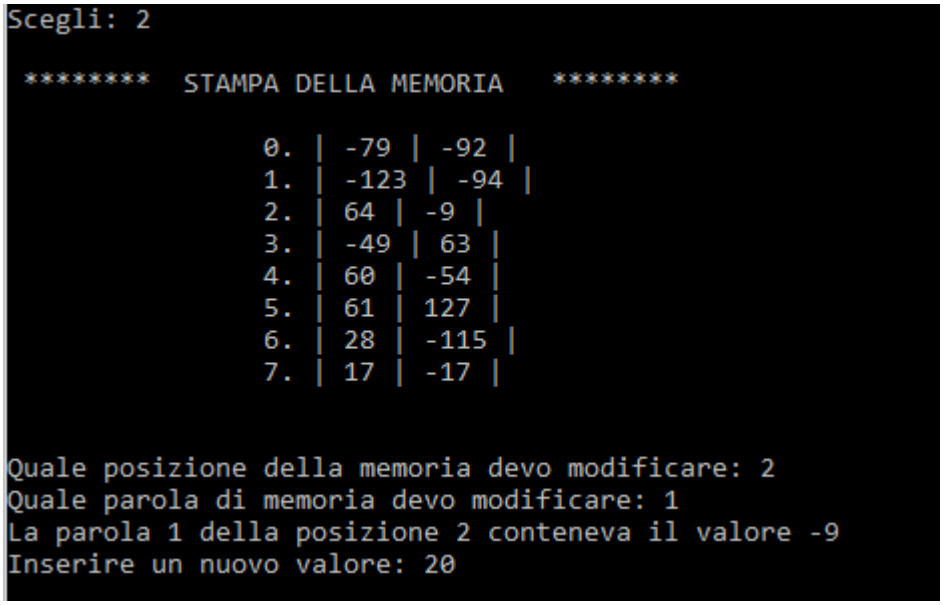


Caso in cui ci sono posizioni di cache libere: il blocco di memoria viene caricato nella prima posizione libera

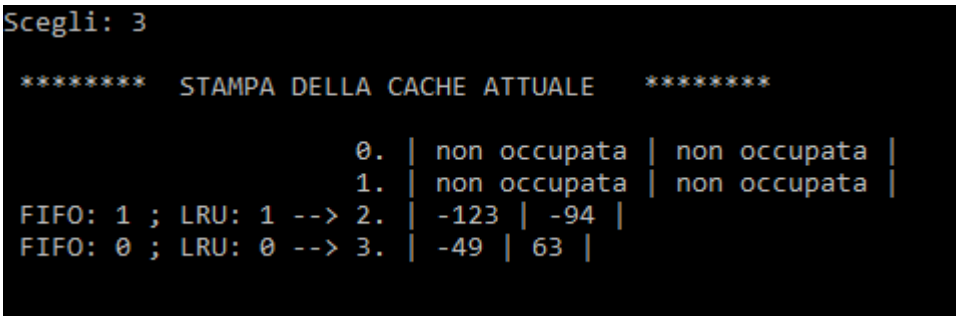


Caso in cui NON ci sono posizioni di cache libere: viene data la possibilità di scegliere quale algoritmo di sostituzione utilizzare, oltre alla possibilità di lasciare invariate le posizioni di cache (opzione "0")

2) ***MODIFICA VALORI IN MEMORIA***: permette di modificare i valori interi contenuti in singole parole di memoria.
   Dopo aver stampato i valori presenti attualmente in memoria, viene richiesto l'inserimento del blocco e della parola di memoria da modificare, oltre ovviamente al nuovo dato da inserire

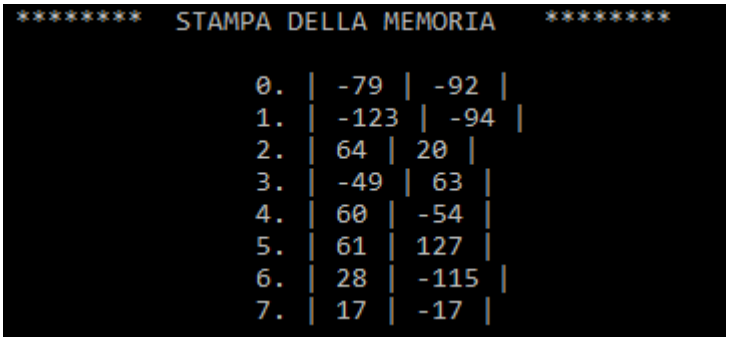


3) ***STAMPA DELLA CACHE***: stampa lo stato corrente della cache. Se una posizione di cache è occupata, vengono anche stampati i contatori FIFO e LRU relativi ad essa.

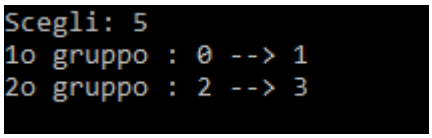


Esempio di stampa della cache dopo aver caricato in cache i blocchi di memoria 1 e 3

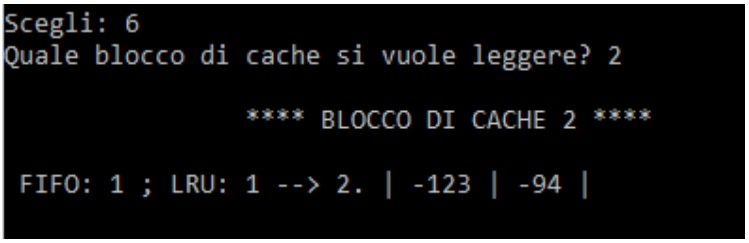
4) ***STAMPA DEL CONTENUTO DELLA MEMORIA***: stampa lo stato corrente della memoria centrale, inclusi i valori contenuti in ogni parola



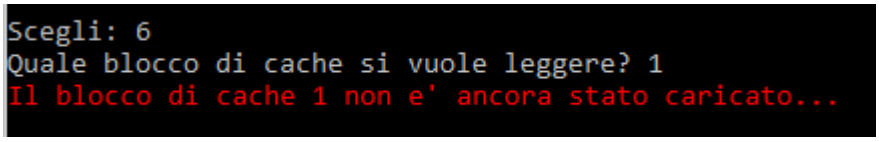
5) ***STAMPA INTERVALLI GRUPPI***: se si è scelto di utilizzare l'indirizzamento associativo a gruppi, è possibile visualizzare gli indici di inizio e fine dei gruppi formati



6) ***STAMPA BLOCCO DI CACHE***: funzionamento uguale alla funzione del punto 3), ma funziona su singole posizioni di cache



Esempio di stampa nel caso in cui la posizione che si vuole stampare è già caricata



Esempio di stampa nel caso in cui la posizione che si vuole stampare NON È ANCORA stata caricata

7) ESCI DAL PROGRAMMA: l'esecuzione del programma termina…