`smat`: un programma interattivo per operazioni su matrici sparse

`http://didawiki.di.unipi.it/doku.php/fisica/informatica/start`

Progetto di recupero del corso di Informatica A/B (CDL Fisica) 2014/15

Contents

1 Introduzione

Il progetto consiste nello sviluppo di un programma C che realizza operazioni su matrici sparse.

1.1 Materiale in linea

Tutto il materiale si trova alla pagina web del corso, seguendo il link appropriato (Corso A o Corso B).

Eventuali chiarimenti possono essere richiesti consultando i docenti l'orario di ricevimento e per posta elettronica.

1.2 Struttura del progetto e consegna

La consegna del progetto avviene *esclusivamente* per posta elettronica, attraverso il target `consegna` del `Makefile` contenuto nel kit di sviluppo del progetto (scaricabile dalla pagina degli assegnamenti) e deve contenere: tutti i file sorgente (`sparse.c` e `smat.c`), la relazione in formato PDF e un file `gruppo.txt` che specifica numero di matricola, cognome, nome ed indirizzo di mail dell'autore, secondo il formato (notare la virgola come separatore)

```
Zini, Gino, 123456,  zini@madoc.it
```

secondo le indicazioni del README nel kit di progetto. Tutti i file del progetto si devono trovare nella directory `SPARSE/` e non sono ammesse sottodirectory.

I progetti che non rispettano il formato o non consegnati con il target `consegna` *non verranno accettati.*

1.3 Valutazione del progetto

Al progetto viene assegnato un punteggio da 0 a 30 in base ai seguenti criteri:

- motivazioni, originalità ed economicità delle scelte progettuali
- strutturazione del codice (suddivisione in moduli, uso di makefile e librerie etc.)
- efficienza e robustezza del software
- aderenza alle specifiche

- qualità del codice C e dei commenti
- chiarezza ed adeguatezza della relazione

La discussione del progetto in sede di orale tenderà a stabilire se lo studente è realmente l'autore e verterà su tutto il programma del corso. Il voto dello scritto + orale (ancora da 0 a 30) fa media con la valutazione del progetto per delineare il voto finale.

2 Il progetto

Lo scopo del progetto è lo sviluppo un programma C per il calcolo di operazioni su matrici sparse e della relativa interfaccia d'uso.

Il programma mette a disposizione un insieme di operazioni su matrici (somma, trasposta etc.) assieme ad alcune funzioni di utilità come il salvataggio su file delle matrici, la visualizzazione di una matrice, ecc). L'interfaccia fra l'utente ed il programma e' un semplice *interprete di comandi*: una volta attivato il programma interagisce con l'utente tramite una linea di comando digitata da tastiera. In particolare, si mette in attesa di comandi da parte dell'utente, esegue le richieste e visualizza i risultati.

Si assume inoltre che in ogni momento l'utente possa lavorare su un *insieme illimitato* di matrici. Ogni matrice è composta da un numero illimitato di righe e colonne e puó contenere elementi di tipo `double`. Inoltre ogni matrice è univocamente identificata da una stringa di lunghezza compresa fra 1 e 8 caratteri alfanumerici. Il primo carattere deve essere una lettera.

2.1 Il programma `smat`

Il programma `smat` viene attivato digitando

```
bash:~$ ./smat
```

Una volta attivato, visualizza un opportuno prompt

```
bash:~$ ./smat utente
Welcome in smat!
-?
```

ed entra in un ambiente interattivo che permette l'esecuzione di un insieme di comandi. Nella Sezione 2.2 per ogni comando accettato, è specificata la sintassi, il significato, e alcune delle condizioni in cui è richiesto di segnalare un errore. Ulteriori condizioni di errore devono essere fissate dallo studente durante la stesura del progetto e specificate nella documentazione.

2.2 Comandi di `smat`

L'interprete di `smat` accetta dall'utente l'insieme di comandi descritti nel seguito. I comandi vanno utilizzati uno alla volta e non sono annidabili.

2.2.1 Aggiunta matrice (`add`)

Comando `add`:
`add` *nomemat n m*

Descrizione:
Se la matrice *nomemat* non è ancora stata definita la aggiunge all'insieme di lavoro (con n righe ed m colonne).

Errori:
Se la matrice e' già presente o se i valori di n ed m non sono corretti (es. negativi).

Esempio:

```
-?  add Tmp 10 20
```

aggiunge la matrice `Tmp` con 10 righe e 20 colonne all'insieme di lavoro.

2.2.2 Cancellazione matrice (`del`)

Comando `del`:
`del` *nomemat*

Descrizione:
La matrice *nomemat* viene cancellata dall'insieme di lavoro.

Errori:
Se la matrice non è presente.

Esempio:

```
-?  del Tmp
-?  del Prodotto
Prodotto:  non presente
```

cancella la matrice `Tmp` e segnala un errore nella richiesta di cancellazione della matrice `Prodotto`.

2.2.3 Aggiunta/cancellazione elementi (=)

Comando =:

nomemat i j = *val*
nomemat i j =

Descrizione:

Se la matrice *nomemat* è presente nell'insieme ne modifica il contenuto. In particolare se *val* non è presente l'elemento corrispondente viene messo a 0.

Errori:

Devono essere segnalati errori nel caso di matrice non presente o valori di *i e* j non congrui (es. maggiori delle dimensioni)

Esempio:

```
-?  Tmp 1 3 = 1.2
-?  Prodotto 4 3 =
```

assegna 1.2 all'elemento (1,3) di `Tmp` e mette a 0 l'elemento (4,3) di `Prodotto`.

2.2.4 Stampa matrice (`print`)

Comando `print`:

`print` *nomemat*

Descrizione:

Visualizza il contenuto della matrice *nomemat* (utilizzando la funzione fornita dai docenti)

Esempio:

```
-?  print A
1:  <1,4.000000><4,8.000000>
2:  <1,4.000000>
3:  <4,4.000000>
5:  <1,3.000000>
```

2.2.5 Matrici correnti (`list`)

Comando `list`:

`list`

Descrizione:

Visualizza l'elenco dei nomi delle matrici presenti nell'insieme e la loro dimensione (Righe; Colonne), secondo il formato specificato nell'esempio:

Esempio:

```
-?  list
Tmp(3;4)
Prodotto(1;2)
```

2.2.6 Salva su file (`save`)

Comando `save`:

`save` *nomefile*

Descrizione:

Salva l'insieme delle matrici (nome, dimensioni, valori) sul file *nomefile*. Durante la fase di salvataggio è necessario verificare l'esistenza di *nomefile* e, se il file esiste, chiedere conferma per la riscrittura. Per la scrittura è richiesto di usare la funzione di stampa fornita dai docenti per la stampa degli elementi e di separare due matrici consecutive con un newline (`\n`).

Esempio:

```
-?  save miofile
miofile saved
```

2.2.7 Carica da file (`load`)

Comando `load`:

`load` *nomefile*

Descrizione:

Carica un insieme di matrici (nome, dimensioni, dati) dal file *nomefile* verificando se sono già presenti in memoria di lavoro. Nel caso in cui una matrice del file sia già presente in memoria, chiede all'utente conferma per la riscrittura.

Errori:

Deve essere segnalato un errore nel caso in cui non sia possibile aggiungere una matrice all'insieme di lavoro.

Esempio:

```
-?  load miofile
miofile loaded
```

2.2.8 Fine sessione (`exit`)

Comando `exit`:

`exit`

Descrizione:
Provoca la terminazione del programma. Se esistono matrici non precedentemente salvate, si deve chiedere conferma all'utente.

Esempio:
```
-?  exit
Ci sono matrici non salvate:  exit ?
```

2.2.9 Aiuto in linea (`help`)

Comando:
`help`

Descrizione:
Visualizza l'elenco dei comandi disponibili e la loro sintassi, descrivendo brevemente il loro significato.

2.2.10 Trasposta (`TRASP`)

Comando `TRASP`:
nomemat1 `=` `TRASP` *nomemat2*

Descrizione:
Esegue la trasposta della matrice *nomemat2*. La matrice risultato *nomemat1* viene aggiunta all'insieme delle matrici oppure, se già presente, vengono modificati i valori dei suoi elementi.

Errori:
Deve essere segnalato un errore nel caso in cui non sia possibile aggiungere la matrice all'insieme delle matrici.

Esempio:
`-?  TraspTmp = TRASP Tmp`
assegna la trasposta di `Tmp` a `TraspTmp`.

2.2.11 Operazioni matriciali (`+` `*`)

Comandi `+` `*`:
nomemat `=` *nomemat1 op nomemat2*
dove *op* può essere solo `+` oppure `*`.

Descrizione:
Esegue l'operazione indicata sulle matrici operandi *nomemat1* e *nomemat2*. La matrice risultato *nomemat* viene aggiunta all'insieme delle matrici oppure, se già presente, vengono modificati i valori dei suoi elementi. È possibile utilizzare una sola operazione alla volta.

Errori:
Devono essere segnalati gli errori numerici rilevati durante il calcolo e gli errori nel caso di inammissibilità dell'operazione a causa dell'incompatibilità degli operandi. Inoltre deve essere segnalato un errore nel caso in cui non sia possibile aggiungere la matrice all'insieme delle matrici.

Esempio:
```
-?  Prodotto = Tmp * TraspTmp
-?  Somma = matrice1 + matrice2
```

3 Codice e documentazione

In questa sezione, vengono riportati alcuni requisiti del codice sviluppato e della relativa documentazione.

3.1 Vincoli sul codice

Il codice consegnato deve rispondere ai seguenti vincoli:

- il codice deve utilizzare la struttura dati definita nel file `sparse.h` contenuto nel kit per rappresentare le matrici sparse;
- il codice delle funzioni definite in `sparse.h` deve superare tutti i test previsti nel Makefile contenuto nel kit;
- la compilazione del codice deve avvenire definendo delle regole appropriate nel Makefile contenuto nel kit;
- il codice deve compilare senza errori o warning gravi utilizzando le opzioni `-Wall -pedantic`
- DEVONO essere usati dei nomi significativi per le costanti nel codice (con opportune `#define` o `enum`)

3.2 Formato del codice

Il codice sorgente deve adottare una convenzione di indentazione e commenti chiara e coerente. In particolare deve contenere

- una intestazione per ogni file che contiene: il nome ed il cognome dell'autore, la matricola, il nome del programma; dichiarazione che il programma è, in ogni sua parte, opera originale dell'autore; firma dell'autore.
- un commento all'inizio di ogni funzione che specifichi l'uso della funzione (in modo sintetico), l'algoritmo utilizzato (se significativo), il significato delle variabili passate come parametri, eventuali variabili globali utilizzate, effetti collaterali sui parametri passati per puntatore etc.
- un breve commento che spieghi il significato delle strutture dati e delle variabili globali (se esistono);
- un breve commento per i punti critici o che potrebbero risultare poco chiari alla lettura
- un breve commento all'atto della dichiarazione delle variabili locali non di uso ovvio, che spieghi a cosa servono

3.3 Relazione

La documentazione del progetto consiste nei commenti al codice e in una breve relazione (massimo 10 pagine) il cui scopo è quello di descrivere la struttura complessiva del lavoro svolto. La relazione *NON deve ripetere le presenti specifiche ma deve rendere comprensibile il lavoro svolto per realizzarle ad un estraneo, senza bisogno di leggere il codice se non per chiarire dettagli.* In pratica la relazione deve contenere:

- le principali scelte di progetto (strutture dati principali, algoritmi fondamentali) e loro motivazioni
- la strutturazione del codice (logica della divisione su più file, librerie etc.)
- le difficoltà incontrate e le soluzioni adottate
- quanto altro si ritiene essenziale alla comprensione del lavoro svolto
- istruzioni per l'utente su come compilare/eseguire ed utilizzare il codice (in quale directory deve essere attivato, quali sono le assunzioni fatte etc) (da riportare anche nel README)

La relazione deve essere in formato ".pdf".