PROGETTO L.A.S.D. Gestione Rotatoria.

Umberto Marotta N86/1035 e Francesco Lentisco N86/1092

April 8, 2014

Contents

1 Funzionalita' del Software

Al momento del lancio, il software presenta un semplice menù che propone all'utente un set completo di funzionalità per la gestione e la simulazione di una Rotatoria. Tra le varie voci vi sono inoltre funzionalità aggiuntive quali ad esempio la randomizzazione e automatizzazione di alcune operazioni che altrimenti andrebbero eseguite manualmente.

Le prime due voci riguardano l'entità Rotatoria, in particolare la prima voce si occupa unicamente di allocare l'oggetto in questione, mentre la seconda di generare una Rotatoria completamente casuale già pronta all'uso, dando all'utente la possibilità di scegliere di quante auto e uscite sarà provvista.

Le voci 3, 4 e 5 riguardano invece la stampa e la simulazione. Per quanto riguarda quest'ultima, si è convenuto di dare la possibilità all'utente stesso di scegliere il numero di Step da visualizzare, così come ogni quanto stampare a video una istantanea (in termini di secondi). Per comodità è stata implementata una simulazione temporizzata dalla pressione del tasto 'Invio'. Per accedervi basterà selezionare '0' alla voce 'Secondi per lo step'.

Al momento della stampa, così come ad ogni istante della simulazione, il programma mostrerà visivamente, servendosi di semplici accorgimenti grafici, la situazione corrente della Rotatoria.

Una singola Automobile è identificata da un numero univoco tra parentesi quadre e da un intero positivo che indica la sua posizione relativa rispetto la propria destinazione, ad esempio: '[17]2' è un'Automobile con ID pari a 17 che dovrà uscire tra due incroci.

Quando un'Automobile si trova in corrispondenza di una uscita che non è la sua ultima destinazione, tale indice verrà opportunamente decrementato. Una singola Uscita è identificata da una lettera maiuscola, e verrà stampata come una lista di Automobili in corrispondenza di una certa casella della Rotatoria. Le voci 6 e 7 riguardano l'acquisizione e il salvataggio di una Rotatoria da/su un file.

È possibile salvare su file ogni istantanea mostrata dal programma. I file altro non sono che una codifica di tutta l'informazione utile per ricostruire la situazione di una rotatoria al momento del salvataggio. Si è optato per un formato minimale, al fine di rendere il caricamento da file quanto più semplice e robusto possibile.

I file riconosciuti correttamente sono formati da interi positivi (fatta eccezione per -1, per indicare una casella della rotatoria vuota), lettere maiuscole dalla A alla Z per indicare, se c'è, il nome di una uscita adiacente, il carattere '|', per indicare l'assenza di una uscita, oppure, se si è già in acquisizione di una coda, per indicarne la terminazione, e infine il carattere '-', per indicare che l'acquisizione di una coda non è ancora terminata. Il formato è dunque formato da triple di due interi e un carattere, opportunamente separati da spazi. Nel file non ci saranno caratteri di newline ed è codificato interamente su una sola riga.

Le voci 8, 9 e 10 comprendono le operazioni possibili sulle Strade, ossia l'inserimento di un certo numero di strade (ed il suo corrispettivo randomizzato) e la cancellazione di una strada.

Le voci 11, 12, 13 e 14 sono le operazioni possibili sulle Automobili. Come per le strade, l'inserimento può essere randomizzato. Dopo il numero di Automobili da inserire, è richiesta la destinazione di ognuna di esse ed infine la strada di partenza. Oltre la cancellazione, è stata implementata una funzionalità di modifica di una Automobile.

2 Implementazione Oggetti

Le funzionalita' del software sono implementate attraverso l'uso di 4 tipi di oggetti con i loro rispettivi metodi:

2.1 Automobile

Come da traccia e' composta da un intero che la indentifica e un intero che indica la i-esima uscita, relativa però a dove l'auto entra, come risulta più naturale anche nel mondo reale.

2.2 Coda

Non è null altro che una coda puntata FIFO di Automobili.

2.3 Strada

E' la strada tramite la quale un Automobile può entrare e uscire dalla rotatoria, in essa è presente la coda a cui ci si può riferire tramite il suo inizio o la fine, è presente inoltre un "nome" per la strada, per identificarle e per permettere alle automobili di uscire in base al nome della strada qualora si voglia farlo.

2.4 Rotonda

Con l'oggetto Rotonda identifichiamo non solo la Rotatoria ma tutto il sistema (con Automobili, Code e Strade), come da traccia la sua dimensione è costante, è infatti realizzata tramite l'uso di due Array (posizione e incrocio) di dimensione 15 che contengono rispettivamente tutto ciò che si muove nella rotatoria (le automobili) e tutto ciò che sta fermo (le strade), l'uso di un doppio array consente di far muovere le automobili in tempo costante (una sola istruzione), infatti in rotonda sono presenti anche altri campi, che sono un intero per il numero di strade incidenti, un intero ed un carattere usati come sequenza e soprattutto due interi che sono l'inizio della rotonda e il numero di automobli, è infatti grazie al primo che il movimento delle auto è costante, il suo incremento fa "girare" il primo array come una roulette rispetto al secondo. Dal momento che anche il numero di auto presenti nella rotonda è preservato, anche la conta (e quindi la verifica che la rotonda sia vuota) diventa costante.

3 Implementazione metodi e complessità asintotica:

3.1 Automobile

3.1.1 automobile nuova_auto (int dest, int id)

(Costante) Crea ed inizializza una nuova automobile con dest (i-esima uscita che deve prendere) ed ID (il suo numero identificativo)

3.1.2 int set_id_auto (automobile car, int id)

(Costante) Metodo standard che permette ad altri oggetti di modificare l'ID dell'Automobile

3.1.3 int set_dest_auto (automobile car, int dest)

(Costante) Metodo standard che permette ad altri oggetti di modificare la destinazione dell'Automobile

3.1.4 int get_id_auto (automobile car)

(Costante) Metodo standard che permette ad altri oggetti di conoscre l'ID dell'Automobile

3.1.5 int get_dest_auto (automobile car)

(Costante) Metodo standard che permette ad altri oggetti di conoscre la destinazione dell'Automobile

3.1.6 int cancella_auto (automobile car)

(Costante) Si occupa di cancellare correttamente un Automobile

3.1.7 void stampa_auto (automobile brum)

(Costante) Stampa in Standard Output un Auto mobile nella forma "[ID]dest" dove ID è il numero in modulo 100 che indentifica l'auto e dest il numero in modulo 10 della sua destinazione

3.2 Coda

3.2.1 coda nuova_coda (automobile car)

(Costante) Crea ed inizializza un nuovo nodo della coda con l'automobile car

3.2.2 automobile get_auto_coda (coda brum)

(Costante) Metodo standard che permette ad altri oggetti di conoscre l'Automobile persente nel nodo "brum" della coda

3.2.3 automobile cerca_auto_coda (coda brum, int ID)

(Lineare sulla lunghezza della coda) Cerca e restituisce se la trova un Automobile identificata da ID nella coda che inizia con il nodo "brum"

3.2.4 coda get_next_coda (coda brum)

(Costante) Metodo standard che permette ad altri oggetti di conoscre il nodo successivo (se presente) a "brum"

3.2.5 coda get_prev_coda (coda inizio, coda fine)

(Lineare sulla lunghezza della coda)[1] Metodo che permette ad altri oggetti di conoscre il nodo precedente (se presente) a "fine" nella coda che inizia con il nodo "inizio"

3.2.6 coda accoda_auto_coda (coda brum, automobile car)

(Lineare sulla lunghezza della coda)[2] Metodo standard che permette di accodare un automobile ad una coda che inizia con il nodo "brum"

3.2.7 coda fine_coda (coda que)

(Lineare sulla lunghezza della coda)[3] Equivalente a get_prev_coda(coda, NULL), restituisce l'ultimo nodo della coda;

3.2.8 int cancella_coda (coda que)

(Lineare sulla lunghezza della coda) Si occupa di cancellare correttamente l'intera coda che inizia con il nodo "que" e restituisce il numero di Automobili cancellate

[1] N.B. Nel programma questo metodo viene chiamato solo quando viene cancellato l'ultimo nodo di una coda che non coincide con il primo

[2] N.B. Nel programma questo metodo viene chiamato sempre e solo sull'ultimo nodo della coda, rendendo di fatto l'operazione Costante

[3] N.B. Nel programma questo metodo viene chiamato sempre e solo sull'ultimo nodo della coda, rendendo di fatto l'operazione Costante

3.2.9 int cancella_auto_coda (coda parent, coda curr, int ID)

(Lineare sulla lunghezza della coda)[4] Cerca e cancella se lo trova il nodo contenente l'Automobile identificata da ID, nella sotto-coda che inizia con il nodo "curr", di cui il nodo "parent" è quello precedente (cancellazione con padre) restituendo 0 se il nodo non è presente nella coda, altrimenti 2 se il nodo cancellato era l'ultimo in coda e 1 in tutti gli altri casi.

3.2.10 void stampa_coda (coda brum)

(Lineare sulla lunghezza della coda) Stampa in Standard Output nella forma "< N < N < N" dove gli 'N' sono il contenuto dei nodi della coda che inizia con il nodo "brum"

3.2.11 void fout_coda (FILE *fp, coda brum)

(Lineare sulla lunghezza della coda) Scrive in un file, nella forma di 'int int char '. rispettivamente l'ID e la destinazione di una automobile presenti in una coda e un carattere che può essere '-' o '|' per indicare rispettivamente che esiste un altro elemento nella coda oppure no (terminazione della coda).

[4] N.B. Per cancellare la testa si dovrebbe usare un metodo interfaccia che alla fine corrisponde al metodo "decoda_auto_strada(strada brum)" di strada

3.3 Strada

3.3.1 strada nuova_strada (char nom, coda que)

(Lineare sulla lunghezza della coda que) [5] Crea ed inizializza una nuova strada con nome "nom", in cui è presente la coda che inizia con il nodo "que".

3.3.2 automobile decoda_auto_strada (strada brum)

(Costante) Decoda un nodo dalla sua coda e restituisce l'automobile presente in esso.

3.3.3 int cancella_strada (strada brum)

(Lineare sulla lunghezza della sua coda) Si occupa di cancellare correttamente la strada resituendo il numero di automobili cancellate

3.3.4 int cancella_auto_strada (strada road, int ID)

(Lineare sulla lunghezza della sua coda) Cerca e cancella se lo trova il nodo contenente l'Automobile identificata da ID, nella sua coda, restituendo 1 se lo trova, 0 altrimenti.

3.3.5 int accoda_auto_strada (strada brum, automobile car)

(Costante) Accoda un automobile nella sua coda restituendo 1 in caso di successo.

3.3.6 char get_nome_strada (strada brum)

(Costante) Metodo standard che permette ad altri oggetti di conoscere il nome della strada

3.3.7 coda get_inizio_strada (strada brum)

(Costante) Metodo standard che permette ad altri oggetti di conoscre il primo nodo della sua coda

3.3.8 int set_nome_strada (strada brum, char nom)

(Costante) Metodo standard che permette ad altri oggetti di modificare il nome della strada

[5] N.B. Nel programma questo metodo viene chiamato sempre e solo con una nuova coda, rendendo di fatto l'operazione Costante

3.4 Rotonda:

3.4.1 rotonda nuova_rotonda ()

(Lineare sulla dimensione della rotatoria, ovvero l'array) Crea ed inizializza una nuova rotonda

3.4.2 rotonda cancella_rotonda (rotonda rot)

(Lineare sulla dimensione dalla rotonda, ovvero il sistema) Si occupa di cancellare correttamente la rotonda

3.4.3 rotonda random_rotonda (int numauto, int numuscite)

(Randomica) Crea ed inizializza una nuova rotonda il cui numero di auto e strade è casuale

3.4.4 int get_numero_strade (rotonda rot)

(Costante) Metodo standard che permette ad altri oggetti di conoscre il numero delle strade che entrano nella rotonda

3.4.5 int rotonda_nextval (rotonda rot):

(Costante) Restituisce il prossimo numero della sequenza incrementandola.[6]

3.4.6 char rotonda_nextchar (rotonda rot)

(Costante) Restituisce il prossimo carattere della sequenza incrementandola.[7]

3.4.7 int rotonda_vuota (rotonda rot)

(Costante) Restituisce 1 se nella rotonda e nelle strade che incrociano non sono presenti auto, 0 altrimenti

3.4.8 strada cerca_strada (rotonda rot, char nome)

(Lineare sulla dimensione della rotatoria, ovvero l'array) Cerca e restituisce se la trova la strada con nome "nome" che incrocia la rotonda

3.4.9 int inserisci_strada (rotonda rot, int i)

(Costante) Inserisce restituendo 1 in caso di successo una nuova strada nell' i-esima posizione della rotatoria assegnandogli automaticamente un nome

3.4.10 int cancella_strada_rotonda (rotonda rot, char nome)

(Lineare dimensione della rotatoria, ovvero l'array, e sulla dimensione della strada di nome 'nome') Cancella l'intera strada identificata da 'nome'. Restituisce 1 in caso di cancellazione avvenuta con successo, altrimenti 0.

[6] N.B. Questo metodo è usato per ottenere facilmente un ID unico per le automobili
[7] N.B. Questo metodo è usato per ottenere facilmente un nome unico per le strade

3.4.11 int cancella_auto_rotonda (rotonda rot, int ID)

(Lineare sulla dimensione della rotonda, ovvero del sistema) Cerca una cerca Automobile identificata dall'ID nell'intero sistema cancellandola. Restituisce un intero: 1 se la cancellazione è avvenuta con successo, altrimenti 0.

3.4.12 automobile cerca_auto (rotonda rot, int ID)

(Lineare sulla dimensione della rotonda, ovvero del sistema) Cerca nell'intera Rotonda l'Automobile identificata da ID, restituendola in caso di successo. In caso di insuccesso viene restituito NULL.

3.4.13 int inserisci_auto_strada (rotonda rot, automobile car, char nome)

(Lineare sulla dimensione della rotatoria, ovvero l'array) Inserisce restituendo 1 in caso di successo un automobile nella strada di nome "nome"

3.4.14 int accoda_auto_rotonda_random (rotonda rot, int num)

(Randomica) Accoda restituendo 1 in caso di successo "num" auto alle strade presenti nella rotonda scelte casualmente.

3.4.15 void step_rotonda (rotonda rot, double secondi)

(Lineare sulla dimensione della rotatoria, ovvero l'array) Simula uno spostamento delle auto nella rotonda prendendosi il numero di secondi specificato. Benchè il metodo sia nel complesso lineare per via dei controlli, lo spostamento delle auto già presenti nella rotatoria è costante, rendendo di fatto il metodo il doppio più efficiente

3.4.16 void stampa_rotonda (rotonda rot)

(Lineare sulla dimensione della rotonda, ovvero del sistema) Stampa a video una Istantanea della situazione corrente della rotatoria.

3.4.17 rotonda fleggi_rotatoria (char *nomeF)

(Lineare sulla dimensione del file) Ricostruisce una situazione di una rotatoria codificata da file, leggendo volta per volta triple nella forma di 'int int char ' (ID, destinazione, e carattere di terminazione o continuazione). Inoltre setta i diversi attributi dell'oggetto Rotonda, calcolandoli on the fly.

3.4.18 void fout_rotatoria (rotonda rot, char* nomeF)

(Lineare sulla dimensione della rotonda, ovvero del sistema) Scrive su file una codifica della situazione corrente di una rotatoria nella forma di 'int int char '. rispettivamente l'ID e la destinazione di una automobile presenti in una coda e un carattere che può essere '-' o '|' per indicare rispettivamente che esiste un altro elemento nella coda oppure no (terminazione della coda).

4 Testing

lo stesso codice è stato compilato e testato su Windows XP x86, Windows 7 x64, Windows 8.1 x64 e Windows 8.1 x86 (usando MinGW), Ubuntu Linux x64 e Arch Linux x86 (usando GCC), e MAC OSX x64.
E' stato scritto utilizzando l'IDE Code::Blocks (http://it.wikipedia.org/wiki/Code::Blocks).
Il binario è stato infine testato con il Valgrind (http://it.wikipedia.org/wiki/Valgrind), un tool open source per trovare problemi di gestione della memoria in file eseguibili sotto architettura Linux x86. Esso individua principalmente problemi di memory leaks e memory corruption. Output di un caso d'uso del file eseguibile con Valgrind

```
==20221==
==20221== HEAP SUMMARY:
==20221==     in use at exit: 0 bytes in 0 blocks
==20221==   total heap usage: 92 allocs, 92 frees, 892 bytes
    allocated
==20221==
==20221== All heap blocks were freed -- no leaks are possible
==20221==
==20221== For counts of detected and suppressed errors, rerun with: -
    v
==20221== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
    0)
```