Terzo Appello di Programmazione I

10 Luglio 2007
Prof. Roberto Sebastiani

Codice:

Nome	Cognome	Matricola

La directory 'esame' contiene 4 sotto-directory: 'uno', 'due', 'tre' e 'quattro'. Le soluzioni vanno scritte negli spazi e nei modi indicati esercizio per esercizio. NOTA: il codice dato non può essere modificato

Modalità di questo appello

Durante la prova gli studenti sono vincolati a seguire le regole seguenti:

- Non è consentito l'uso di alcun libro di testo o fotocopia. In caso lo studente necessitasse di carta (?), gli/le verranno forniti fogli di carta bianca su richiesta, che dovranno essere riconsegnati a fine prova. È consentito l'uso di una penna. Non è consentito l'uso di alcuno strumento calcolatore.
- È vietato lo scambio di qualsiasi informazione, orale o scritta. È vietato guardare nel terminale del vicino.
- È vietato l'uso di telefoni cellulari o di qualsiasi strumento elettronico.
- È vietato allontanarsi dall'aula durante la prova, anche se si ha già consegnato. (Ogni necessità fisiologica va espletata PRIMA dell'inizio della prova.)
- È vietato qualunque accesso, in lettura o scrittura, a file esterni alla directory di lavoro assegnata a ciascun studente. Le uniche operazioni consentite sono l'apertura, l'editing, la copia, la rimozione e la compilazione di file all'interno della propria directory di lavoro.
- Sono ovviamente vietati l'uso di email, ftp, ssh, telnet ed ogni strumento che consenta di accedere a file esterni alla directory di lavoro. Le operazioni di copia, rimozione e spostamento di file devono essere circoscritte alla directory di lavoro.
- Ogni altra attività non espressamente citata qui sopra o autorizzata dal docente è vietata.

Ogni violazione delle regole di cui sopra comporterà automaticamente l'annullamento della prova e il divieto di accesso ad un certo numero di appelli successivi, a seconda della gravità e della recidività della violazione.

NOTA IMPORTANTE: DURANTE LA PROVA PER OGNI STUDENTE VERRÀ ATTIVATO UN TRACCIATORE SOFTWARE CHE REGISTRERÀ TUTTE LE OPERAZIONI ESEGUITE (ANCHE ALL'INTERNO DELL'EDITOR!!). L'ANNULLAMENTO DELLA PROVA DI UNO STUDENTE POTRÀ AVVENIRE ANCHE IN UN SECONDO MOMENTO, SE L'ANALISI DELLE TRACCE SOFTWARE RIVELASSERO IRREGOLARITÀ.

1 Nel file `esercizio1.cc` sono presenti le funzioni `stampa_array` e `main` definite come segue:

- la funzione `stampa_array` prende in input un array di interi e ne stampa a video il contenuto;
- la funzione `main` prende da standard input una parola e la memorizza in un array di caratteri `parola`, invoca poi la funzione `calcola_frequenza` ed infine chiama la funzione `stampa_array`.

Per semplicità supponiamo che la parola data in input sia composta solamente da caratteri minuscoli.

Realizzare la funzione `calcola_frequenza` che, preso in input un array di caratteri `parola` contenente una parola scritta in caratteri minuscoli, restituisca un array di interi contenente la frequenza delle lettere dell'alfabeto presenti nella parola data. In particolare ogni cella dell'array di interi deve corrispondere ad una lettera minuscola dell'alfabeto e deve contere il numero di lettere di questo tipo presenti in `parola`: ad esempio il primo elemento dell'array corrisponderà alla lettera 'a' dell'alfabeto e conterrà il numero di 'a' contenute nella parola data. L'array di interi deve quindi avere 26 caselle.

Se ad esempio la parola memorizzata in `parola` è *zibibbo* la funzione `calcola_frequenza` deve restituire il seguente array:

0	3	0	0	0	0	0	0	2	0	0	0	0	0	1	0	...	0	1

che corrisponde a dire: "Nella parola data ci sono: 0 lettere 'a', 3 lettere 'b', 0 lettere 'c', 0 lettere 'd', ..., 2 lettere 'i', ... ed 1 lettera 'z'".

NOTA: all'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static`, ad eccezione di quelle già presenti.

VALUTAZIONE: questo esercizio vale 7 punti (al punteggio di tutti gli esercizi va poi sommato 10).

1 Nel file `esercizio1.cc` sono presenti le funzioni `stampa_frequenza` e `main` definite come segue:

- la funzione `stampa_frequenza` prende in input un array di interi e ne stampa a video il contenuto;
- la funzione `main` prende da standard input una parola e la memorizza in un array di caratteri chiamato `parola`, invoca poi la funzione `calcola` ed infine chiama la funzione `stampa_frequenza`.

Per semplicità supponiamo che la parola data in input sia composta solamente da caratteri maiuscoli.

Realizzare la funzione `calcola` che, preso in input un array di caratteri `parola` contenente una parola scritta in caratteri maiuscoli, restituisca un array di interi contenente la frequenza delle lettere dell'alfabeto presenti nella parola data. In particolare ogni cella dell'array di interi deve corrispondere ad una lettera maiuscola dell'alfabeto e deve contenere il numero di lettere di questo tipo presenti in `parola`: ad esempio il primo elemento dell'array corrisponderà alla lettera 'A' dell'alfabeto e conterrà il numero di 'A' contenute nella parola data. L'array di interi deve quindi avere 26 caselle.

Se ad esempio la parola memorizzata in `parola` è *ZIBIBBO* la funzione `calcola` deve restituire il seguente array:

0	3	0	0	0	0	0	0	2	0	0	0	0	0	1	0	...	0	1

che corrisponde a dire: "Nella parola data ci sono: 0 lettere 'A', 3 lettere 'B', 0 lettere 'C', 0 lettere 'D', ..., 2 lettere 'I', ... ed 1 lettera 'Z'".

NOTA: all'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static`, ad eccezione di quelle già presenti.

VALUTAZIONE: questo esercizio vale 7 punti (al punteggio di tutti gli esercizi va poi sommato 10).

1 **esercizio1.cc**

```
#include <iostream>
using namespace std;

const int dim = 100;

void stampa_array (int []);
int* calcola_frequenza (char []);

int main(){

  char parola [dim];
  int * frequenza;

  cout << "Inserisci la parola (lettere minuscole): ";
  cin >> parola;

  frequenza = calcola_frequenza(parola);

  cout << "Nella parola data ci sono:" << endl;
  stampa_array(frequenza);

  return(0);
}

void stampa_array (int vett[])
{
  for(int i=0; i<26; i++)
    cout << vett[i] << " lettere " << char(i+97) << endl;
  cout<<endl;
}

//Scrivete il vostro codice al di sotto di questo commento
int* calcola_frequenza (char parola[]){
   int *frequenza=new int[26];
   int i;

   for(i=0;i<26;i++)
     frequenza[i]=0;

   i=0;
   while (parola[i]!='\0' && i<dim) {
     frequenza[parola[i]-97]++;
     i++;
   }
   return frequenza;
}
```

2 Nel file `esercizio2.cc` scrivere un programma che, presi come argomento del `main` i nomi di due file, e utilizzando i dati anagrafici di una lista di persone contenuti nel primo file, calcoli e scriva nel secondo file la frequenza degli anni di nascita.

In particolare, su ogni riga del file contentente i dati di input, vengono date le informazioni relative ad una persona nel seguente formato:

```
cognome nome citta_nascita giorno_nascita mese_nascita anno_nascita
```

in cui ogni campo indicato (`nome`, `cognome`, `citta_nascita`, ...) puo' essere considerato come formato da una sola parola senza spazi bianchi o caratteri speciali. Il programma deve leggere le informazioni contenute nel primo file e, utilizzando opportune strutture dati, calcolare per gli anni compresi tra il 1900 ed il 1999 quante persone sono nate in ciascun anno. Infine le frequenze calcolate, **diverse da zero**, dovranno essere memorizzate nel secondo file nel seguente formato:

```
anno numero_nati
```

Dato ad esempio come file di input il file `anagrafica_studenti` contenente i seguenti dati:

```
AMADORI GIORGIO ROMA 15 07 1982
BERTI LUIGI MILANO 12 10 1988
CAPUTO SERGIO TRENTO 30 12 1986
CASIRAGHI UMBERTO PADOVA 02 02 1981
MODENA ALESSANDRO VENEZIA 05 11 1988
SEBASTIANI ROBERTO MESSINA 23 09 1985
TOMASI ALESSANDRO ANCONA 01 01 1985
ROSSI PAOLO GENOVA 28 02 1981
BIANCHI STEFANO TORINO 31 01 1981
```

se l'eseguibile è `a.out`, il comando

```
./a.out anagrafica_studenti statistica_anni
```

creerà il file `statistica_anni` con il seguente contenuto:

```
1981 3
1982 1
1985 2
1986 1
1988 2
```

VALUTAZIONE: questo esercizio vale 6 punti (al punteggio di tutti gli esercizi va poi sommato 10).

2 Nel file `esercizio2.cc` scrivere un programma che, presi come argomento del `main` i nomi di due file, e utilizzando i dati di una lista di persone contenuti nel primo file, calcoli e scriva nel secondo file la frequenza dei voti degli esami di maturitá.

In particolare, su ogni riga del file contentente i dati di input, vengono date le informazioni relative ad una persona nel seguente formato:

```
cognome nome citta_scuola_superiore anno_maturita voto_maturita
```

in cui ogni campo indicato (`nome`, `cognome`, `citta_scuola_superiore`, ...) puo' essere considerato come formato da una sola parola senza spazi bianchi o caratteri speciali. Il programma deve leggere le informazioni contenute nel primo file e, utilizzando opportune strutture dati, calcolare per i voti compresi tra 0 e 100 quante persone hanno ottenuto ciascun voto. Infine le frequenze calcolate, **diverse da zero**, dovranno essere memorizzate nel secondo file nel seguente formato:

```
voto_maturita numero_persone
```

Dato ad esempio come file di input il file `dati_maturita` contenente i seguenti dati:

```
AMADORI GIORGIO ROMA 1982 80
BERTI LUIGI MILANO 1988 75
CAPUTO SERGIO TRENTO 1986 100
CASIRAGHI UMBERTO PADOVA 1981 66
MODENA ALESSANDRO VENEZIA 1988 66
SEBASTIANI ROBERTO MESSINA 1985 100
TOMASI ALESSANDRO ANCONA 1985 75
ROSSI PAOLO GENOVA 1981 25
BIANCHI STEFANO TORINO 1981 75
ROVATI DANTE BOLZANO 1990 95
SAVONA EMANUELA RIMINI 2004 63
```

se l'eseguibile è `a.out`, il comando

```
./a.out dati_maturita statistica_voti
```

creerà il file `statistica_voti` con il seguente contenuto:

```
25 1
63 1
66 2
75 3
80 1
95 1
100 2
```

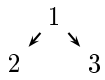
VALUTAZIONE: questo esercizio vale 6 punti (al punteggio di tutti gli esercizi va poi sommato 10).

2 **esercizio2.cc**

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <string>
using namespace std;

int main(int argc,char* argv[]){

  fstream my_in, my_out;
  int frequenze [100];
  char tmp[20];
  int i, anno, init=1900;

  if (argc != 3) {
    cout << "Usage: ./a.out <file_anagrafica> <file_frequenza>\n";
    exit(0);
  }

  for (i=0; i<100; i++) frequenze[i]=0;

  my_in.open(argv[1],ios::in);
  my_in >> tmp; //cognome
  while (!my_in.eof()) {
    my_in >> tmp; //nome
    my_in >> tmp; //luogo
    my_in >> tmp; //giorno
    my_in >> tmp; //mese
    my_in >> tmp; //anno;

    anno = atoi(tmp);

    i=0;
    do {
      if (anno==init+i)
        frequenze[i]++;
      i++;
    } while (i<100 && anno!=init+i-1);

    my_in >> tmp; //cognome
  }
  my_in.close();


  my_out.open(argv[2],ios::out);
  for (i=0; i<100; i++) {
    if (frequenze[i]!=0)
      my_out << init+i << " " << frequenze[i] << endl;
  }
  my_out.close();

  return(0);
}
```

3 Implementare un albero di ricerca binaria, i cui nodi contengano le informazioni *nome e cognome*, con le seguenti funzionalità:

- inizializzazione della struttura dati albero;
- controllo se un albero è vuoto;
- inserimento di un elemento con il seguente ordine alfabetico crescente per cognome: i cognomi più bassi ($a < b < c < \dots$) a sinistra;
- ricerca di un elemento per cognome;
- stampa dell'albero in ordine alfabetico. Esempio: l'albero seguente



  deve essere stampato come 213.

Scrivere nel file `albero.cc` la definizione delle funzioni dichiarate nello header file `albero.h` sviluppando un albero la cui struttura di ogni singolo nodo sia quella definita nello header file `albero.h`.

Nel file `albero_main.cc` è definita la funzione `main` che contiene un menu per gestire l'albero.

CONSIGLI UTILI: Per copiare e confrontare stringhe usare:

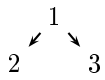
```
char *strcpy(char *dest, const char *src);
int strcmp(const char *s1, const char *s2);
int strlen(const char * string );
```

includendo la libreria `<string.h>`.

VALUTAZIONE: questo esercizio vale 7 punti (al punteggio di tutti gli esercizi va poi sommato 10).

3 Implementare un albero di ricerca binaria, i cui nodi contengano le informazioni *nome e cognome*, con le seguenti funzionalità:

- inizializzazione della struttura dati albero;
- controllo se un albero è vuoto;
- inserimento di un elemento con il seguente ordine alfabetico decrescente per cognome: i cognomi più bassi ($a < b < c < \dots$) a destra;
- ricerca di un elemento per cognome;
- stampa dell'albero in ordine alfabetico. Esempio: l'albero seguente



  deve essere stampato come 312.

Scrivere nel file `albero.cc` la definizione delle funzioni dichiarate nello header file `albero.h` sviluppando un albero la cui struttura di ogni singolo nodo sia quella definita nello header file `albero.h`.

Nel file `albero_main.cc` è definita la funzione `main` che contiene un menu per gestire l'albero.

CONSIGLI UTILI: Per copiare e confrontare stringhe usare:

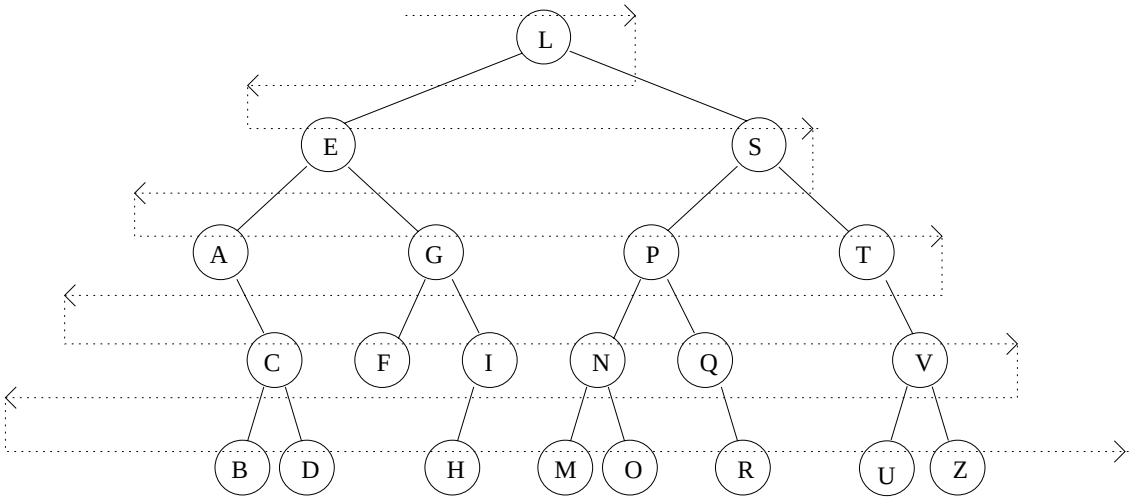
```
char *strcpy(char *dest, const char *src);
int strcmp(const char *s1, const char *s2);
int strlen(const char * string );
```

includendo la libreria `<string.h>`.

VALUTAZIONE: questo esercizio vale 7 punti (al punteggio di tutti gli esercizi va poi sommato 10).

3 **esercizio3.cc**

4 Per stampa breadth-first di un albero si intende la stampa degli elementi dell'albero partendo dalla radice e proseguendo per livelli di profondità crescenti, da sinistra a destra.



Ad esempio. la sequenza stampata in modalità breadth-first nella figura precedente è

L E S A G P T C F I N Q V B D H M O R U Z

Data la libreria `tree.h` per la gestione di alberi di ricerca binaria, si realizzi all'interno del file `esercizio4.cc` la funzione NON-RICORSIVA `stampa_bfs` che stampa un'albero in modo breadth-first.
NOTE:

- la funzione non deve essere ricorsiva;
- deve richiedere un numero di operazioni proporzionale alla dimensione dell'albero.

SUGGERIMENTO:
si usi la coda di alberi, disponibile nella libreria `queue.h`.

VALUTAZIONE:
questo esercizio permette di conseguire la lode se tutti gli esercizi precedenti sono corretti.

4 esercizio4.cc

```
using namespace std;
#include <iostream>
#include "tree.h"
#include "queue.h"

void print_bfs(const tree & t);

int main()
{
  char option,val;
  tree t, tmp;
  retval res;
  init(t);
  do {
    cout << "\nOperazioni possibili:\n"
         << "Inserimento (i)\n"
         << "Ricerca (r)\n"
         << "Stampa ordinata (s)\n"
         << "Stampa indentata (e)\n"
         << "Stampa depth-first-search (d)\n"
         << "Fine (f)\n";
    cin >> option;
    switch (option) {
    case 'i':
      cout << "Val? : ";
      cin >> val;
      res = insert(t,val);
      if (res == FAIL)
        cout << "spazio insufficiente!\n";
      break;
    case 'r':
      cout << "Val? : ";
      cin >> val;
      tmp=cerca(t,val);
      if (!nullp(tmp))
        cout << "Valore trovato!: " << val << endl;
      else
        cout << "Valore non trovato!\n";
      break;
    case 's':
      cout << "Stampa ordinata:\n";
      print_ordered(t);
      break;
    case 'e':
      cout << "Stampa indentata:\n";
      print_indented(t);
      break;
    case 'd':
      cout << "Stampa breadth-first-search:\n";
      print_bfs(t);
      break;
    case 'f':
```

```
      break;
    default:
      cout << "Optione errata\n";
    }
  } while (option != 'f');
}


void print_bfs(const tree & t) {
  if (!nullp(t)) {
    enqueue(t);
    while (!queue_emptyp()) {
      tree t1;
      dequeue(t1);
      cout << t1->item << endl;
      if (t1->left!=NULL)
enqueue(t1->left);
      if (t1->right!=NULL)
enqueue(t1->right);
    }
  }
}
```