Secondo Appello di Programmazione I

13 Giugno 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 esame

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 Implementare la funzione `elimina_ripetizioni` in modo tale che, preso in input una array di caratteri `caratteri` e la sua dimensione `numero`, restituisca un array contenente i medesimi caratteri dell'array preso in input MA SENZA CARATTERI RIPETUTI CONSECUTIVI, e la nuova dimensione di questo array.

Ad esempio se `caratteri` è il seguente array:
`a b b f g g g h t z z`
il nuovo array generato dalla funzione `elimina_ripetizioni` sarà:
`a b f g h t z`

NOTA:
all'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static`.
Per semplicità supporre che l'array `caratteri`:

- non sia mai vuoto,
- abbia tutti i caratteri già ordinati alfabeticamente,
- abbia sempre un numero di caratteri inferiore a 100.

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

1 Implementare la funzione `elimina_ripetizioni` in modo tale che, preso in input una array di interi `numeri` e la sua dimensione `dimensione`, restituisca un array contenente i medesimi numeri interi dell'array preso in input MA SENZA NUMERI RIPETUTI CONSECUTIVI, e la nuova dimensione di questo array.

Ad esempio se `numeri` è il seguente array:
`34 56 56 123 145 145 145 678`
il nuovo array generato dalla funzione `elimina_ripetizioni` sarà:
`34 56 123 145 678`

NOTA:
all'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static`.
Per semplicità supporre che l'array `numeri`:

- non sia mai vuoto,
- abbia tutti i numeri interi ordinati in modo cresecente,
- abbia sempre una quantità di numeri interi inferiore a 100.

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

1 **esercizio1.cc**

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

void leggi_caratteri (char caratteri [100], int numero);
void stampa_caratteri (char caratteri [100], int numero);
char* elimina_ripetizioni (char caratteri [100], int numero, int& numero_diversi);

int main(){

  int numero=0;
  int numero_diversi=0;
  char caratteri [100];
  char *caratteri_diversi;

  cout << "Inserisci il numero di caratteri: ";
  cin >> numero;

  leggi_caratteri(caratteri, numero);
  caratteri_diversi = elimina_ripetizioni(caratteri,numero,numero_diversi);

  cout << "Sono stati trovati "<< numero_diversi << " caratteri diversi."<<endl;
  cout << "Ecco la lista:" << endl;
  stampa_caratteri(caratteri_diversi, numero_diversi);
  return(0);
}

void leggi_caratteri (char caratteri[100], int numero)
{
  int i;

  for (i=0;i<numero;i++)
    {
      cout << "Inserire carattere " << i+1 <<" : ";
      cin >> caratteri[i];
    }
}

void stampa_caratteri (char caratteri[100], int numero)
{
  int i;

  for(i=0; i<numero; i++)
    {
      cout << caratteri[i] << " ";
    }
  cout<<endl;
}

//Scrivete il vostro codice al di sotto di questo commento

char* elimina_ripetizioni (char caratteri [100], int numero, int& numero_diversi)
{
```

```
  int i,k=0;
  char *caratteri_diversi;

  for(i=0; i<numero-1; i++)
  {
    if (caratteri[i] != caratteri[i+1])
      numero_diversi++;
  }
  numero_diversi++;

  caratteri_diversi=new char[numero_diversi];

  for(i=0;i<numero-1;i++){
    if (caratteri[i] != caratteri[i+1])
      {
caratteri_diversi[k]=caratteri[i];
k++;
      }
  }
  caratteri_diversi[k]=caratteri[numero-1];
  return caratteri_diversi;
}
```

2 Scrivere un programma che, presi come argomenti del `main` i nomi di due file, copi il testo contenuto nel primo file nel secondo, "evidenziando" le preposizioni proprie presenti nel testo. Le preposizioni proprie da ricercare nel testo sono le seguenti: **di**, **a**, **da**, **in**, **con**, **su**, **per**, **tra**, **fra**. Sono da considerare anche le preposizioni proprie che iniziano con lettera maiuscola.

Se ad esempio l'eseguibile è `a.out`, il comando

```
./a.out testo testo_analizzato
```

creerà un nuovo file di nome `testo_analizzato`, vi copierà il contenuto di `testo` aggiungendo, dopo ogni preposizione propria individuata, la stringa "`(PP)`".

Per semplicità si assuma che il testo contenuto del primo file sia su un'unica riga e che ogni occorrenza delle parole da cercare sia separata dalle altre da spazi bianchi.

CONSIGLI UTILI: Per copiare e confrontare stringhe usare:

```
char *strcat(char *dest, const char *src);
int strcmp(const char *s1, const char *s2);
```

includendo la libreria `<string>`.

Supponendo di avere nel file di input il seguente testo:

```
Un mio amico di nome Sam ricevette un' automobile come regalo di Natale
da suo fratello. La vigilia di Natale, quando Sam usci' dall' ufficio,
un monello di strada stava girando attorno all' auto nuova luccicante,
ammirandola. " E' sua questa macchina, signore? " domando'. Sam
annui'. " Me l' ha regalata mio fratello per Natale."
```

il programma dovrà generare un altro file contenente questo testo:

```
Un mio amico di (PP) nome Sam ricevette un' automobile come regalo di (PP) Natale
da (PP) suo fratello. La vigilia di (PP) Natale, quando Sam usci' dall' ufficio,
un monello di (PP) strada stava girando attorno all' auto nuova luccicante,
ammirandola. " E' sua questa macchina, signore? " domando'. Sam
annui'. " Me l' ha regalata mio fratello per (PP) Natale."
```

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

2 Scrivere un programma che, presi come argomenti del `main` i nomi di due file, copi il testo contenuto nel primo file nel secondo, "evidenziando" gli articoli determinativi presenti nel testo. Gli articoli determinativi da ricercare nel testo sono i seguenti: **il**, **lo**, **la**, **i**, **gli**, **le**, **l'**. Sono da considerare anche gli articolo determinativi che iniziano con lettera maiuscola.

Se ad esempio l'eseguibile è `a.out`, il comando

```
./a.out testo testo_analizzato
```

creerà un nuovo file di nome `testo_analizzato`, vi copierà il contenuto di `testo` aggiungendo, dopo ogni articolo determinativo individuato, la stringa "`(AD)`".

Per semplicità si assuma che il testo contenuto del primo file sia su un'unica riga e che ogni occorrenza delle parole da cercare sia separata dalle altre da spazi bianchi.

CONSIGLI UTILI: Per copiare e confrontare stringhe usare:

```
char *strcat(char *dest, const char *src);
int strcmp(const char *s1, const char *s2);
```

includendo la libreria `<string>`.

Supponendo di avere nel file di input il seguente testo:

```
Un mio amico di nome Sam ricevette un' automobile come regalo di Natale
da suo fratello. La vigilia di Natale, quando Sam usci' dall' ufficio,
un monello di strada stava girando attorno all' auto nuova luccicante,
ammirandola. " E' sua questa macchina, signore? " domando'. Sam
annui'. " Me l' ha regalata mio fratello per Natale."
```

il programma dovrà generare un altro file contenente questo testo:

```
Un mio amico di nome Sam ricevette un' automobile come regalo di Natale
da suo fratello. La (AD) vigilia di Natale, quando Sam usci' dall' ufficio,
un monello di strada stava girando attorno all' auto nuova luccicante,
ammirandola. " E' sua questa macchina, signore? " domando'. Sam
annui'. " Me l' (AD) ha regalata mio fratello per Natale."
```

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

2 **esercizio2.cc**

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

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

  fstream my_in, my_out;
  char * articoli [18] = {"di", "a", "da", "in", "con", "su", "per", "tra", "fra",
                          "Di", "A", "Da", "In", "Con", "Su", "Per", "Tra", "Fra"};
  char tmp[20];

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

  my_in.open(argv[1],ios::in);
  my_out.open(argv[2],ios::out);

  my_in >> tmp;
  while (!my_in.eof()) {
    int i=0;
    int trovato = 0;
    do {
      if (!strcmp(tmp,articoli[i])) {
        strcat(tmp," (PP)");
        trovato = 1;
      }
      i++;
    } while (i<18 && trovato!=1);
    my_out << tmp << " ";
    my_in >> tmp;
  }

  my_in.close();
  my_out.close();
  return(0);
}
```

3 Nel file `queue_main.cc` è definita la funzione `main` che contiene un menu per gestire una coda di puntatori a elementi di tipo struct `persona`. La struct `persona` e le funzioni per gestirla sono dichiarate nel file `persona.h` e definite nel file binario `persona.o`. Scrivere, in un nuovo file `queue.cc`, le definizioni delle funzioni dichiarate nello header file `queue.h` in modo tale che:

- `init` inizializzi la coda;
- `enqueue` inserisca l'elemento passato come parametro nella coda;
- `dequeue` tolga l'elemento in testa alla coda e lo memorizzi nella variabile passata come parametro;
- `stampa` stampi a video il contenuto della coda.

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

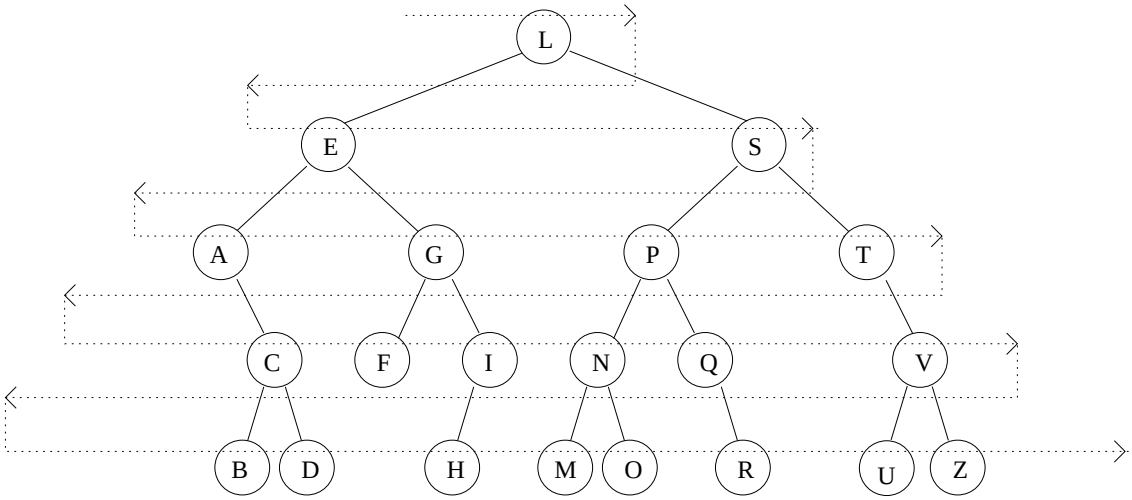
3 Nel file `stack_main.cc` è definita la funzione `main` che contiene un menu per gestire una pila di puntatori a elementi di tipo struct `persona`. La struct `persona` e le funzioni per gestirla sono dichiarate nel file `persona.h` e definite nel file binario `persona.o`. Scrivere, in un nuovo file `stack.cc`, le definizioni delle funzioni dichiarate nello header file `stack.h` in modo tale che:

- `init` inizializzi la pila;
- `push` inserisca l'elemento passato come parametro nella pila;
- `pop` tolga l'elemento in testa alla pila e lo memorizzi nella variabile passata come parametro;
- `top` legga l'elemento in testa alla pila e lo memorizzi nella variabile passata come parametro;
- `stampa` stampi a video il contenuto della pila.

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);
    }
  }
}
```