Secondo Appello di Programmazione I

10 Febbraio 2016
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 Scrivere un programma, nel file `esercizio1.cc`, che, presi come argomenti del `main` i nomi di due file, copi il primo file nel secondo correggendone la sintassi e generando in tal modo un testo "corretto" secondo le seguenti regole:

(a) la prima parola del testo deve iniziare con una lettera maiuscola;

(b) tutte le parole che seguono i seguenti caratteri: ".", "?" e "!", devono iniziare con una lettera maiuscola.

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

```
./a.out testo testocorretto
```

creerà un nuovo file di nome `testocorretto` e vi copierà il contenuto del file dato `testo`, modificando le parole quando queste non verificano le regole descritte sopra. Nelle figure 1 e 2 un esempio di file `testo` e `testocorretto`.

filastrocca delle parole:
Fatevi avanti! chi ne vuole?
di parole ho la testa piena,
con dentro la "luna" e la "balena".
ci sono parole per gli amici:
Buon giorno, Buon anno, Siate felici!
parole belle e parole buone;
parole per ogni sorta di persone.
di G. Rodari.

Figura 1: `testo`

Filastrocca delle parole:
Fatevi avanti! Chi ne vuole?
Di parole ho la testa piena,
con dentro la "luna" e la "balena".
Ci sono parole per gli amici:
Buon giorno, Buon anno, Siate felici!
Parole belle e parole buone;
parole per ogni sorta di persone.
Di G. Rodari.

Figura 2: `testocorretto`

NOTA 1: Per semplicità si assuma che il testo contenuto nel primo file inizi con un carattere alfabetico, non contenga "..." e che ".", "?" e "!" siano sempre preceduti da una parola e seguiti da uno spazio.

NOTA 2: Per semplicità si assuma che ogni parola contenuta nel testo del primo file abbia al massimo lunghezza 30 caratteri.

NOTA 3: È ammesso l'uso della funzione `strlen` della libreria `<cstring>`, non e' ammesso l'uso di altre funzioni di libreria, in particolare della funzione `toupper`.

NOTA 4: il programma deve potenzialmente funzionare con ogni possibile codifica dei caratteri secondo le regole di tali codifiche viste a lezione (quindi non solo ASCII). Per realizzare la conversione da caratteri minuscoli in maiuscoli, è vietato l'uso di tabelle o di 26 `if` o `switch-case`, uno per ogni carattere.

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

1 **soluzione_A11.cc**

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

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

  fstream my_in, my_out;
  char tmp[31];

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

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

  my_in >> tmp;
  if(!(tmp[0] >= 'A' && tmp[0] <= 'Z'))
    tmp[0] = tmp[0] + ('A'-'a');

  while(!my_in.eof()){
    my_out << tmp << " ";
    if(tmp[strlen(tmp)-1] == '.' || tmp[strlen(tmp)-1] == '?' || tmp[strlen(tmp)-1] == '!') {
      my_in >> tmp;
      if(!(tmp[0] >= 'A' && tmp[0] <= 'Z'))
        tmp[0] = tmp[0] + ('A'-'a');
    } else my_in >> tmp;
  }

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

2 Completare il programma definito nel file `esercizio2.cc` con la dichiarazione e la definizione della funzione **ricorsiva** `reverse_array` che, preso in input un array di caratteri ed un intero che rappresenti la sua lunghezza, inverta l'array di caratteri passato come parametro; inoltre, il codice deve sostituire ogni **vocale minuscola** ('a', 'e', 'i', 'o', 'u') con un punto esclamativo ('!').

L'array in ingresso corrisponde ai caratteri della stringa passata come primo argomento da riga di comando all'atto dell'invocazione del programma. Per esempio, se l'eseguibile è **a.out**, dato il comando:

```
./a.out aIUolA
```

l'output del programma dovrà essere:

```
Array invertito: A l ! U I !
```

NOTA 1: **La funzione** `reverse_array` **deve essere** `ricorsiva` **ed al suo interno non ci possono essere cicli o chiamate a funzioni contenenti cicli. Si può fare uso di eventuali funzioni ricorsive ausiliarie.**

NOTA 2: è **vietato** modificare in alcun modo il codice della funzione main.

NOTA 3: si assuma che la dimensione massima di una parola sia `DIMMAX` caratteri.

NOTA 4: all'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria, comprese quelle della libreria `cstring` o `string.h`.

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

2 **codice_A21.cc**

```
#include <iostream>
#include <string>

using namespace std;

const char SEGNAPOSTO = '!';
const int DIMMAX = 40;

// Dichiarare qui sotto la funzione reverse_array

int main (int argc, char* argv[]) {
  char input[DIMMAX];
  cout << "Introdurre stringa da invertire: ";
  cin >> input;

  // Assumiamo che sia sempre strlen(input) < DIMMAX
  reverse_array(input, strlen(input));
  cout << "Array invertito: " << input << endl;

  return 0;
}

// Definire qui sotto la funzione reverse_array
```

2 **soluzione_A21.cc**

```
#include <iostream>
#include <string>

using namespace std;

const char SEGNAPOSTO = '!';
const int DIMMAX = 40;

// Dichiarare qui sotto la funzione reverse_array
void reverse_array(char* in, int dim);
bool vocale_minuscola(char c);

int main (int argc, char* argv[]) {
  char input[DIMMAX];
  cout << "Introdurre stringa da invertire: ";
  cin >> input;

  // Assumiamo che sia sempre strlen(input) < DIMMAX
  reverse_array(input, strlen(input));
  cout << "Array invertito: " << input << endl;

  return 0;
}

// Definire qui sotto la funzione reverse_array
```

```
void reverse_array(char* in, int dim) {
  // Finisce quando non ha piu' caratteri da copiare
  if (dim > 0) {
    if(vocale_minuscola(in[dim - 1]) && vocale_minuscola(in[0])) {
      // Caso semplice: i due estremi vanno sovrascritti
      in[dim - 1] = in[0] = SEGNAPOSTO;
    } else if(vocale_minuscola(in[dim - 1])) {
      // Basta salvare il valore del primo elemento,
      // tanto l'ultimo elemento andrebbe comunque sovrascritto
      in[dim - 1] = in[0];
      in[0] = SEGNAPOSTO;
    } else if(vocale_minuscola(in[0])) {
      // Basta salvare il valore dell'ultimo elemento,
      // tanto il primo elemento andrebbe comunque sovrascritto
      in[0] = in[dim - 1];
      in[dim - 1] = SEGNAPOSTO;
    } else {
      // Scambio "classico"
      char tmp = in[dim - 1];
      in[dim - 1] = in[0];
      in[0] = tmp;
    }
    // Chiamata ricorsiva
    reverse_array(in + 1, dim - 2);
  }
}

bool vocale_minuscola(char c) {
  return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
}
```

3 Nel file `albero_main.cc` è definita la funzione `main` che contiene un menù per gestire un albero binario di ricerca di `int`. Scrivere nel file `albero.cc` le definizioni delle funzioni dichiarate nel file di header `albero.h` in modo tale che:

- `inizializza` inizializzi l'albero;
- `vuoto` controlli se l'albero è vuoto, restituendo `VERO` in caso affermativo e `FALSO` in caso contrario;
- `inserisci` inserisca l'elemento passato come parametro nell'albero, restituendo `VERO` se l'operazione è andata a buon fine, e `FALSO` altrimenti. L'albero deve essere ordinato in maniera **crescente** e se l'elemento è già presente deve essere inserito a **sinistra**. Esempio: l'inserimento dei seguenti valori:

  ```
  10  5  12  5
  ```

  deve produrre il seguente albero:

  ```
          10
      5       12
  5
  ```

- `cerca` cerchi nell'albero l'elemento passato in input, resituendo `VERO` se l'elemento è presente, e `FALSO` altrimenti;
- `stampa` stampi a video il contenuto dell'albero, in ordine **crescente**. Esempio: l'albero qui sopra deve essere stampato come:

  ```
  5  5  10  12
  ```

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

3 albero_main.cc

```
#include <iostream>
#include "albero.h"

using namespace std;

int main() {
  char res;
  Albero albero;
  int val;
  inizializza(albero);
  do {
    cout << "\nOperazioni possibili:\n"
         << " Inserimento (i)\n"
         << " Ricerca (r)\n"
         << " Stampa ordinata (s)\n"
         << " Fine (f)\n";
    cout << "Operazione selezionata: ";
    cin >> res;
    switch (res) {
    case 'i':
      cout << "Valore : ";
      cin >> val;
      if (inserisci(albero, val) == FALSO) {
        cout << "Valore gia' presente!" << endl;
      }
      break;
    case 'r':
      cout << "Valore: ";
      cin >> val;
      if (cerca(albero, val) == VERO) {
        cout << "Valore presente: " << val << endl;
      } else {
        cout << "Valore non presente" << endl;
      }
      break;
    case 's':
      if (vuoto(albero) == VERO) {
        cout << "Albero vuoto!" << endl;
      } else {
        stampa(albero);
      }
      break;
    case 'f':
      break;
    default:
      cout << "Opzione errata\n";
    }
  } while (res != 'f');

  return 0;
}
```

3 **albero.h**

```
// -*- C++ -*-
#ifndef ALBERO_H
#define ALBERO_H

struct Nodo {
    int val;
    Nodo *sx;
    Nodo *dx;
};

typedef Nodo * Albero;

enum boolean { VERO, FALSO };

void inizializza(Albero &t);
boolean vuoto(const Albero &t);
boolean inserisci(Albero &t, int val);
boolean cerca(const Albero &t, int val);
void stampa(const Albero &t);

#endif
```

3 **albero_.cc**

```
#include <iostream>
#include "albero.h"

using namespace std;

void inizializza(Albero &a) {
  a = NULL;
}

boolean vuoto(const Albero &a) {
  return (a == NULL) ? VERO : FALSO;
}

boolean inserisci(Albero &a, int val) {
  // Caso base
  if (vuoto(a) == VERO) {
    a = new (nothrow) Nodo;
    if (a == NULL) {
       return FALSO;
    }
    a->val = val;
    a->sx = a->dx = NULL;
    return VERO;
  }
  // Caso ricorsivo. Controllo se scendere a sinistra o a destra
  if (val <= a->val) {
    // Scendo a sinistra
    return inserisci(a->sx, val);
```

```
  } else if (val > a->val) {
    // Scendo a destra
    return inserisci(a->dx, val);
  }
}

boolean cerca(const Albero &a, int val) {
  if (vuoto(a) == VERO) {
    return FALSO;
  } else if (val == a->val) {
    // Trovato
    return VERO;
  } else if (val < a->val) {
    // Scendo a sinistra
    return cerca(a->sx, val);
  } else {
    // Scendo a destra
    return cerca(a->dx, val);
  }
}

void stampa(const Albero &a) {
  if (vuoto(a) == FALSO) {
    // Prima stampo gli elementi MINORI di a->val (cioe' quelli a sx)
    stampa(a->sx);
    // Poi stampo a->val
    cout << a-> val << ' ';
    // Poi stampo gli elementi MAGGIORI (cioe' quelli a dx)
    stampa(a->dx);
  }
}
```

4 Si definiscano le funzioni `sum`, `prod` e `power` dichiarate nel file `esercizio4.cc` le quali calcolino rispettivamente la somma, il prodotto e la potenza tra due interi n, m maggiori o uguali a zero, utilizzando esclusivamente:

(a) la costante `Zero` che rappresenta il valore 0 (non e' consentito usare altre costanti di alcun tipo);

(b) le funzioni `IsZero`, `succ`, `pred` definite nel file `auxiliary.cc` e `auxiliary.h`;

(c) i costrutti if-then-else e return;

(d) la RICORSIONE;

In particolare non è consentito usare:

(a) iterazioni;

(b) variabili globali o static;

(c) operatori di confronto (`==`, `!=`, `<`, `>`, `<=`, `>=`);

(d) operatori di assegnazione (`=`, `+=`, `*=`, ...);

(e) operatori aritmetici;

(f) altri operatori o funzioni di alcun tipo che non siano quelli sopra elencati.

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

4 **codice_A41.cc**

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

int sum(int n,int m);
int prod (int n,int m);
int power (int n,int m);


int main () {
  int n,m;
  cout << "n,m? ";
  cin >> n >> m;
  cout << "La somma tra n e m e' : " << sum(n,m) << endl;
  cout << "Il prodotto tra n e m e' : " << prod(n,m) << endl;
  cout << "La potenza di n alla m e' : " << power(n,m) << endl;
}

// --- definire le funzioni sum, prod e power qui sotto ---
```

4 **soluzione_A41.cc**

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

int sum(int n,int m);
int prod (int n,int m);
int power (int n,int m);


int main () {
  int n,m;
  cout << "n,m? ";
  cin >> n >> m;
  cout << "La somma tra n e m e' : " << sum(n,m) << endl;
  cout << "Il prodotto tra n e m e' : " << prod(n,m) << endl;
  cout << "La potenza di n alla m e' : " << power(n,m) << endl;
}

// --- definire le funzioni sum, prod e power qui sotto ---

int sum(int n,int m) {
  if (IsZero(m))
    return n;
  else
    return sum(succ(n),pred(m));
}

int prod (int n,int m) {
  if (IsZero(m))
```

```
    return Zero;
  else
    return sum(prod(n,pred(m)),n);
}

int power (int n,int m) {
  if (IsZero(m))
    return succ(Zero);
  else
    return prod(power(n,pred(m)),n);
}
```