Secondo Appello di Programmazione I

09 Febbraio 2010
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 Scrivere un programma "sniffer" per codici di carte di credito che, dato un file di testo contenente un numero indefinito di caratteri, ricerchi stringhe nel seguente formato:

```
NNNN NNNN NNNN NNNN
```

dove ogni carattere `N` corrisponde ad una cifra compresa tra 0 e 9.

Al termine dell'esecuzione di questo programma, dovrà essere stampato a video il numero delle stringhe compatibili con questo formato presenti nel file di testo specificato in input.

Dato, ad esempio, in input il file `testo` contenente i seguenti dati:

```
Spettabile ditta specilizzata in attrezzature informatiche, sono interessato
all'acquisto del vostro portatile in promozione del valore di 699,00 euro.
Per il pagamento potete usare il numero di questa mia carta di
credito: 6784 5608 9629 9271 intestata a Mario Rossi oppure il seguente numero
di carta di credito: 7896 4563 7294 1862 intesata a mia moglie Carla Bianchi.
```

se l'eseguibile è `a.out`, il comando

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

stamperà a video le seguenti informazioni:
`Il numero di stringhe trovate compatibili con il formato richiesto e' 2.`

NOTA: supporre, per semplicità, che questa sequenza di 4 gruppi da 4 cifre da individuare sia sempre precededuta e seguita da un separatore.

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

1 Scrivere un programma "sniffer" per indirizzi di posta elettronica che, dato un file di testo contenente un numero indefinito di caratteri, ricerca stringhe nel seguente formato:

```
<caratteri alfanumerici>@<caratteri alfanumerici con almeno un carattere '.'>
```

Al termine dell'esecuzione di questo programma, dovrà essere stampato a video il numero delle stringhe compatibili con questo formato presenti nel file di testo specificato in input.

Dato, ad esempio, in input il file `testo` contenente i seguenti dati:

```
Spettabile ditta specilizzata in attrezzature informatiche, sono interessato
all'acquisto del vostro portatile in promozione del valore di 699,00 euro.
Potete inviarmi via mail al seguente indirizzo mario.rossi@gmail.com
le specifiche di tale portatile? Qualora questo indirizzo mail fosse errato
le chiederei di usare il seguente indirizzo info@mariorossi.it
```

se l'eseguibile è `a.out`, il comando

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

stamperà a video le seguenti informazioni:

```
Il numero di stringhe trovate compatibili con il formato richiesto e' 2.
```

NOTA: supporre, per semplicità, che questa sequenza di caratteri da individuare sia sempre precededuta e seguita da un separatore.

NOTA 2: questa sequenza contiene solo carattteri alfanumerici o punti, non è quindi richiesto di gestire altri caratteri validi per gli indirizzi di posta elettronica come, ad esempio, '-' o '_'. In tutte queste sequenze deve esistere una chiocciola sola e la prima parte della stringa deve avere almeno un carattere alfanumerico.

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

1 **esercizio1.cc**

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <cstring>

using namespace std;

int main(int argc,char* argv[])
{
  fstream my_in;
  char tmp[100];
  int numero_cifre=0,numero_stringhe=0,numero_target=0;

  if (argc != 2) {
    cout << "Usage: ./a.out <testo>\n";
    exit(0);
  }

  my_in.open(argv[1],ios::in);
  my_in >> tmp; //lettura prima parola
  while (!my_in.eof()) {
    for (int i=0; i<4 ; i++)
      if ((tmp[i]>='0') && (tmp[i]<='9'))
numero_cifre++;
    if ((numero_cifre==4) && (strlen(tmp)==4))
      numero_stringhe++;
    else
      numero_stringhe=0;
    numero_cifre=0;

    if (numero_stringhe==4)
      {
numero_target++;
numero_stringhe=0;
      }
    my_in >> tmp; //lettura parola successiva
  }
  my_in.close();

  cout << "Il numero di stringhe trovate compatibili con il formato richiesto e' " << numero_ta

  return(0);
}
```

2 Nel file `esercizio2.cc` è definita una funzione iterativa `calcola_it` che prende come parametri formali due interi, esegue alcune operazioni e restituisce un intero come risultato.

Scrivere nel file `esercizio2.cc` la definizione della funzione `calcola_ric` in modo tale che traduca la funzione iterativa `calcola_it` utilizzando la **ricorsione**.

NOTA: la funzione `calcola_ric` non può essere iterativa: al suo interno, non ci possono quindi essere cicli o chiamate a funzioni contenenti cicli.

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

2 Nel file `esercizio2.cc` è definita una funzione iterativa `calcola_it` che prende come parametri formali due interi, esegue alcune operazioni e restituisce un intero come risultato.

Scrivere nel file `esercizio2.cc` la definizione della funzione `calcola_ric` in modo tale che traduca la funzione iterativa `calcola_it` utilizzando la **ricorsione**.

NOTA: la funzione `calcola_ric` non può essere iterativa: al suo interno, non ci possono quindi essere cicli o chiamate a funzioni contenenti cicli.

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

2 **esercizio2.cc**

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

int calcola_it(int x, int y);
int calcola_ric(int x, int y);

int main()
{
  int a, b, ris_ric, ris_it;

  cout << "Inserisci il primo numero: ";
  cin  >> a;

  cout << "Inserisci il secondo numero: ";
  cin  >> b;

  if (a <= 0 || b <= 0){
     cout << "Inserire interi positivi!" << endl;
     return 0;
  }

  ris_it = calcola_it(a, b);

  ris_ric = calcola_ric(a, b);

  cout << "Il risultato della funzione iterativa e': " << ris_it << endl
       << "Il risultato della funzione ricorsiva e': " << ris_ric << endl;

  return 0;
}

int calcola_it(int x, int y){
  int res = 0;
  while(x >= y){
    x = x-y;
    res++;
  }
  return res;
}

// Inserire qui la definizione della funzione ricorsiva "calcola_rec"

int calcola_ric(int x, int y){
  if (x < y){
return 0;
  } else {
return 1 + calcola_ric(x-y,y);
  }
}
```

3 Nel file `queue_main.cc` è definita la funzione `main` che contiene un menu per gestire una coda di interi `long`. 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;
- `deinit` liberi la memoria utilizzata dalla coda;
- `empty` restituisca `FALSE` se la coda contiene almeno un valore, e `TRUE` altrimenti;
- `enqueue` inserisca il valore passato come parametro nella coda;
- `dequeue` elimini il valore in testa alla coda, restituendo `TRUE` se l'operazione è andata a buon fine, e `FALSE` altrimenti;
- `first` legga il valore in testa alla coda e lo memorizzi nella variabile passata come parametro, restituendo `TRUE` se l'operazione è andata a buon fine, e `FALSE` altrimenti;
- `print` stampi a video il contenuto della coda (senza modificarla), nell'ordine in cui i valori contenuti ne verrebbero estratti.

La coda deve essere implementata internamente come una lista concatenata i cui elementi sono allocati dinamicamente.

NOTA: ad eccezione delle operazioni `print` e `deinit`, si ricorda che ogni altra operazione sulla coda deve avere costo costante, indipendentemente dal numero di elementi contenuti nella struttura dati.

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

3 Nel file `queue_main.cc` è definita la funzione `main` che contiene un menu per gestire una coda di caratteri `char`. 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;
- `deinit` liberi la memoria utilizzata dalla coda;
- `empty` restituisca `FALSE` se la coda contiene almeno un carattere, e `TRUE` altrimenti;
- `enqueue` inserisca il carattere passato come parametro nella coda;
- `dequeue` elimini il carattere in testa alla coda, restituendo `TRUE` se l'operazione è andata a buon fine, e `FALSE` altrimenti;
- `first` legga il carattere in testa alla coda e lo memorizzi nella variabile passata come parametro, restituendo `TRUE` se l'operazione è andata a buon fine, e `FALSE` altrimenti;
- `print` stampi a video il contenuto della coda (senza modificarla), nell'ordine in cui i valori contenuti ne verrebbero estratti.

La coda deve essere implementata internamente come una lista concatenata i cui elementi sono allocati dinamicamente.

NOTA: ad eccezione delle operazioni `print` e `deinit`, si ricorda che ogni altra operazione sulla coda deve avere costo costante, indipendentemente dal numero di elementi contenuti nella struttura dati.

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

3 esercizio3.cc

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

#include "queue.h"


void init (queue &q)
{
    q.head = q.tail = NULL;
}

void deinit (queue &q)
{
    node *n = q.head;
    while (n != NULL)
    {
      node *tmp = n;
      n = n->next;
      delete tmp;
    }
    q.head = q.tail = NULL;
}

retval empty (const queue &q)
{
    return (q.head == NULL ? TRUE : FALSE);
}

void enqueue (queue &q, long val)
{
    node *n = new node;

    n->val = val;
    n->next = NULL;
    if (empty(q))
      q.head = q.tail = n;
    else
    {
      q.tail->next = n;
      q.tail = n;
    }
}

retval dequeue (queue &q)
{
    node *first;
    if (empty(q))
      return FALSE;

    first = q.head;
    q.head = q.head->next;
```

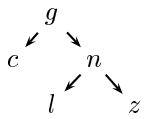
```
    delete first;

    if (empty(q))
      q.tail = NULL;

    return TRUE;
}

retval first (const queue &q, long &result)
{
    if (empty(q))
      return FALSE;

    result = q.head->val;
    return TRUE;
}


void print (const queue &q)
{
    node *n = q.head;
    while (n != NULL)
    {
      cout << n->val << " ";
      n = n->next;
    }
    cout << endl;
}
```

4 Dati i seguenti file:

- `tree.h` e `tree.o`, che implementano una libreria "albero di ricerca binaria di caratteri", a cui manca la funzione di stampa ordinata dell'albero;
- `stack.h` e `stack.o`, che implementano una libreria "stack di alberi";
- `esercizio4.cc` che contiene la funzione `main` con il menu che gestisce l'albero di ricerca binaria;

scrivere all'interno del file `esercizio4.cc` la definizione della funzione **iterativa** "`stampa`" che, preso in input un albero di ricerca binaria, ne stampi il contenuto in modo ordinato crescente.

Ad esempio, il seguente albero:



deve essere stampato nell'ordine 'c', 'g', 'l', 'n', 'z'.

NOTA: **La funzione non può essere ricorsiva: al suo interno, non ci possono quindi essere chiamate a sè stessa o ad altre funzioni ricorsive o mutualmente ricorsive.**

SUGGERIMENTO: utilizzare lo stack offerto nei file `stack.h` e `stack.o`.

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

4 esercizio4.cc

```
#include <iostream>
#include "tree.h"
#include "stack.h"

using namespace std;

int main()
{
  char res,val;
  tree t, tmp;
  init(t);
  do {
    cout << "\nOperazioni possibili:\n"
         << "Inserimento (i)\n"
         << "Ricerca (r)\n"
         << "Stampa DFS (s)\n"
         << "Fine (f)\n";
    cin >> res;
    switch (res) {
    case 'i':
      cout << "Val? : ";
      cin >> val;
      insert(t,val);
      if (t==NULL)
        cout << "memoria esaurita!\n";
      break;
    case 'r':
      cout << "Val? : ";
      cin >> val;
      tmp=cerca(t,val);
      if (tmp!=NULL)
        cout << "Valore trovato!: " << val << endl;
      else
        cout << "Valore non trovato!\n";
      break;
    case 's':
      cout << "Stampa:\n";
      stampa(t);
      break;
    case 'f':
      break;
    default:
      cout << "Optione errata\n";
    }
  } while (res != 'f');
}


retval stampa(tree t) {
  stack_init();
  tree current = t;
  while(!vuoto(current) || !stack_vuoto()) {
```

```
    if (!vuoto(current)) {  // percorso all'ingiu'
      stack_push(current);
      current= current->left;
    }
    else { // !stack_vuoto() // torno su'
      stack_top(current);
      stack_pop();
      tree_node_print(current);
      current=current->right;
    }
  }
}
```