Quinto Appello di Programmazione I

13 Settembre 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 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 nel file `esercizio1.cc` la dichiarazione e la definizione della procedura **ricorsiva** `inverti_array` che effettua in modo **ricorsivo** l'inversione di una sequenza di **numeri** contenuta in un array di al più 100 elementi di tipo `long`. In pratica la procedura deve scambiare il primo elemento della sequenza con l'ultimo, il secondo elemento con il penultimo, e così via... per tutti gli elementi della sequenza. Nel caso di sequenza di dimensione dispari l'elemento centrale resta quindi invariato. **La procedura deve iniziare dagli elementi più esterni ed interrompersi qualora gli elementi da invertire siano uguali tra loro**.

Ad esempio se la sequenza in input è:

```
9 21 5 6 2 19
```

dopo la chiamata a `inverti_array` il programma stamperà:

```
19 2 6 5 21 9
```

mentre nel caso:

```
1000000 8 33 9876543 10
```

verrà stampata la sequenza invertita:

```
10 9876543 8 33 1000000
```

Se invece la sequenza in input è:

```
10 2 7 999 0 44 18 7 8181 3
```

dopo la chiamata a `inverti_array` il programma stamperà:

```
3 8181 7 999 0 44 18 7 2 10
```

infatti la procedura viene interrotta dopo aver invertito i primi ed ultimi due elementi, in quanto il terzo numero a partire dalla testa dell'array (7) è uguale al terzo numero a partire dal fondo.

NOTA: **La funzione deve essere ricorsiva ed al suo interno non ci possono essere cicli o chiamate a funzioni contenenti cicli; può invece fare uso di eventuali funzioni ausiliarie purchè a loro volta ricorsive. Non è altresì ammesso l'uso di variabili globali o di tipo static.**

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

1 **esercizio1.cc**

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

// Inserire qui le dichiarazioni

void inverti_array (long*, int);

int main()
{
  int dim;
  long array[100];

  do
  {
    cout << "Inserisci il numero (0 < ... < 100) di elementi da memorizzare nell'array: ";
    cin >> dim;
  } while (dim <= 0);

  if (dim > 100)
  {
    cout << "Puoi inserire solo 100 elementi!" << endl;
    dim = 100;
  }
  for (int i = 0; i < dim; i++)
  {
    cout << "Inserire valore " << i+1 << ": ";
    cin  >> array[i];
  }

  inverti_array(array, dim);

  cout << "Array invertito:";
  for (int i = 0; i < dim; i++)
    cout << " " << array[i];
  cout << endl;

  return 0;
}

// Inserire qui le definizioni

void inverti (long &n, long &m)
{
  long tmp = n;
  n = m;
  m = tmp;
}

void inverti_array (long *a, int dim)
{
  if ((dim <= 1) || (a[0] == a[dim-1]))
    return;
```

```
  else
  {
    inverti(a[0], a[dim-1]);
    inverti_array(a+1, dim-2);
  }
}

// SOLUZIONE ALTERNATIVA
/*
void inverti_array (long *a, int primo, int ultimo)
{
  if ((primo >= ultimo) || (a[primo] == a[ultimo]))
    return;
  else
  {
    inverti(a[primo], a[ultimo]);
    inverti_array(a, primo+1, ultimo-1);
  }
}

void inverti_array (long *a, int dim)
{
  inverti_array(a, 0, dim-1);
}
*/
```

1 Scrivere nel file `esercizio1.cc` la dichiarazione e la definizione della procedura **ricorsiva** `scambia_array` che effettua in modo **ricorsivo** lo scambio di una sequenza di **caratteri** contenuta in un array di al più 100 elementi di tipo `char`. In pratica la procedura deve scambiare il primo elemento della sequenza con l'ultimo, il secondo elemento con il penultimo, e così via... per tutti gli elementi della sequenza. Nel caso di sequenza di dimensione dispari l'elemento centrale resta quindi invariato. **La procedura deve iniziare dagli elementi più esterni ed interrompersi qualora gli elementi da scambiare siano uguali tra loro**.

Ad esempio se la sequenza in input è:

```
Q W E R T Y
```

dopo la chiamata a `scambia_array` il programma stamperà:

```
Y T R E W Q
```

mentre nel caso:

```
A z # K 9
```

verrà stampata la sequenza invertita:

```
9 K # z A
```

Se invece la sequenza in input è:

```
M @ 2 ; * % 2 X 3
```

dopo la chiamata a `scambia_array` il programma stamperà:

```
3 X 2 ; * % 2 @ M
```

infatti la procedura viene interrotta dopo aver scambiato i primi ed ultimi due elementi, in quanto il terzo carattere a partire dalla testa dell'array (2) è uguale al terzo carattere a partire dal fondo.

NOTA: **La funzione deve essere ricorsiva ed al suo interno non ci possono essere cicli o chiamate a funzioni contenenti cicli; può invece fare uso di eventuali funzioni ausiliarie purchè a loro volta ricorsive. Non è altresì ammesso l'uso di variabili globali o di tipo static.**

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

1 **esercizio1.cc**

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

// Inserire qui le dichiarazioni

void scambia_array (char*, int);

int main()
{
  int dim;
  char array[100];

  do
  {
    cout << "Inserisci il numero (0 < ... < 100) di elementi da memorizzare nell'array: ";
    cin >> dim;
  } while (dim <= 0);

  if (dim > 100)
  {
    cout << "Puoi inserire solo 100 elementi!" << endl;
    dim = 100;
  }
  for (int i = 0; i < dim; i++)
  {
    cout << "Inserire carattere " << i+1 << ": ";
    cin  >> array[i];
  }

  scambia_array(array, dim);

  cout << "Array scambiato:";
  for (int i = 0; i < dim; i++)
    cout << " " << array[i];
  cout << endl;

  return 0;
}

// Inserire qui le definizioni

void scambia (char &n, char &m)
{
  char tmp = n;
  n = m;
  m = tmp;
}

void scambia_array (char *a, int dim)
{
  if ((dim <= 1) || (a[0] == a[dim-1]))
    return;
```

```
  else
  {
    scambia(a[0], a[dim-1]);
    scambia_array(a+1, dim-2);
  }
}

// SOLUZIONE ALTERNATIVA
/*
void scambia_array (char *a, int primo, int ultimo)
{
  if ((primo >= ultimo) || (a[primo] == a[ultimo]))
    return;
  else
  {
    scambia(a[primo], a[ultimo]);
    scambia_array(a, primo+1, ultimo-1);
  }
}

void scambia_array (char *a, int dim)
{
  scambia_array(a, 0, dim-1);
}
*/
```

2 Scrivere un programma che, dato un carattere ed un file di testo contenente un numero indefinito di caratteri, calcoli il numero di parole che contengono il carattere specificato in input consecutivamente ripetuto.

Al termine dell'esecuzione di questo programma, dovrà essere stampato a video il numero di parole calcolato e, nel caso in cui questo numero sia maggiore di zero, dovrà essere aggiunto alla fine del file di input la seguente stringa: "Presenti parole con carattere consecutivamente ripetuto".

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

```
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.
```

se l'eseguibile è `a.out`, il comando

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

stamperà a video le seguenti informazioni:
`Il numero di parole con il carattere 'l' consecutivamente ripetuto e':  2`
ed il file `testo` avrà il seguente contenuto:

```
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.
Presenti parole con carattere consecutivamente ripetuto
```

NOTA: è ammesso l'utilizzo della funzione di libreria `strlen`.

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

2 **esercizio2.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[100], target;
  int numero_parole=0, trovato, i;

  if (argc != 3) {
    cout << "Usage: ./a.out <carattere> <testo>\n";
    exit(0);
  }
  target=argv[1][0];
  my_in.open(argv[2],ios::in);

  my_in >> tmp; //lettura prima parola
  while (!my_in.eof()) {
    i=0;
    trovato=0;
    while ((i<strlen(tmp))&&(trovato==0)) {
      if ((tmp[i]==target) && (tmp[i+1]==target)) {
numero_parole++;
trovato=1;
      }
      i++;
    }
    my_in >> tmp; //lettura parola successiva
  }
  my_in.close();

  cout << "Il numero di parole con il carattere '" << target << "' consecutivamente ripetuto e'
  my_out.open(argv[2],ios::out|ios::app); //apertura file in append

  if (numero_parole!=0)
    my_out << "Presenti parole con carattere consecutivamente ripetuto" <<endl;;

  my_out.close();

  return(0);
}
```

2 Scrivere un programma che, dato un carattere ed un file di testo contenente un numero indefinito di caratteri, calcoli il numero di parole che contengono il carattere specificato in input consecutivamente ripetuto.

Al termine dell'esecuzione di questo programma, dovrà essere stampato a video il numero di parole calcolato e, nel caso in cui questo numero sia zero, dovrà essere aggiunto alla fine del file di input la seguente stringa: "Nessuna parola presente con carattere consecutivamente ripetuto".

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

```
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.
```

se l'eseguibile è `a.out`, il comando

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

stamperà a video le seguenti informazioni:
`Il numero di parole con il carattere 'l' consecutivamente ripetuto e':  0`
ed il file `testo` avrà il seguente contenuto:

```
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.
Nessuna parola presente con carattere consecutivamente ripetuto
```

NOTA: è ammesso l'utilizzo della funzione di libreria `strlen`.

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

2 **esercizio2.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[100], target;
  int numero_parole=0, trovato, i;

  if (argc != 3) {
    cout << "Usage: ./a.out <carattere> <testo>\n";
    exit(0);
  }
  target=argv[1][0];
  my_in.open(argv[2],ios::in);

  my_in >> tmp; //lettura prima parola
  while (!my_in.eof()) {
    i=0;
    trovato=0;
    while ((i<strlen(tmp))&&(trovato==0)) {
      if ((tmp[i]==target) && (tmp[i+1]==target)) {
numero_parole++;
trovato=1;
      }
      i++;
    }
    my_in >> tmp; //lettura parola successiva
  }
  my_in.close();

  cout << "Il numero di parole con il carattere '" << target << "' consecutivamente ripetuto e'
  my_out.open(argv[2],ios::out|ios::app); //apertura file in append

  if (numero_parole==0)
    my_out << "Nessuna parola presente con carattere consecutivamente ripetuto" <<endl;;

  my_out.close();

  return(0);
}
```

3 Nel file `stack_main.cc` è definita la funzione `main` che contiene un menu per gestire una **pila** di (al più cento) caratteri **char** con array. 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;
- `emptyp` restituisca `TRUE` se la pila non contiene alcun carattere e `FALSE` altrimenti;
- `fullp` restituisca `TRUE` se la coda e' piena e `FALSE` altrimenti;
- `push` inserisca il carattere passato come parametro nella pila, restituendo `TRUE` se l'operazione è andata a buon fine, e `FALSE` altrimenti;
- `pop` elimini il carattere in testa alla pila, restituendo `TRUE` se l'operazione è andata a buon fine, e `FALSE` altrimenti;
- `top` legga il carattere in testa alla pila 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 pila.

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

3 stack_main_A31.cc

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

int main()
{
    char res;
    char num;
stack S;

init(S);

    do {
        cout << "\nOperazioni possibili:\n"
             << "Push (u)\n"
             << "Pop (p)\n"
             << "Top (t)\n"
 << "Stampa (s)\n"
             << "Fine (f)\n";
        cin >> res;
        switch (res) {
        case 'u':
            cout << "Valore: ";
            cin >> num;
            if (push(S, num)==FALSE)
                cout << "Memoria esaurita!\n";
            break;
        case 'p':
            if (pop(S) == FALSE) {
                cout << "Stack vuoto!\n";
            }
break;
case 't':
            if (top(S,num)==FALSE) {
                cout << "Stack vuoto!\n";
} else {
    cout << "Valore: " << num << endl;
}
break;
        case 's':
            print(S);
            break;
        case 'f':
            break;
        default:
            cout << "Operazione non disponibile!\n";
        }
    } while (res != 'f');

    return 0;
}
```

3 stack_A31.h

```
#ifndef STACK_H
#define STACK_H

using namespace std;
#include <iostream>

// dichiarazioni per la gestine della pila di caratteri

enum retval {FALSE, TRUE};

static const int dim = 100;

struct stack {
    int indice;
    char elem[dim];
};

void init(stack &);
retval emptyp (stack);
retval fullp (stack);
retval push (stack &, char);
retval pop (stack &);
retval top (stack, char &);
void print(stack);


#endif
```

3 stack_A31.cc

```
#include "stack.h"

void init (stack &S) {
  S.indice = 0;
}

retval emptyp (stack S) {
retval res = FALSE;
    if (S.indice==0) {
        res = TRUE;
}
return res;
}

retval fullp (stack S) {
retval res = FALSE;
if (S.indice==dim) {
    res = TRUE;
}
return res;
}
```

```
retval top (stack S, char &n) {
    retval res;
    if (emptyp(S)) {
        res = FALSE;
    } else {
        n = S.elem[S.indice-1];
        res = TRUE;
    }
    return res;
}

retval push (stack &S, char n) {
    retval res;
    if (fullp(S)) {
        res = FALSE;
    } else {
        S.elem[S.indice++] = n;
        res = TRUE;
}
    return res;
}

retval pop (stack &S) {
    retval res;
    if (emptyp(S)) {
        res = FALSE;
    } else {
        S.indice--;
res = TRUE;
    }
    return res;
}

void print(stack S) {
    int i;
    for (i=0; i<S.indice; i++) {
        cout << S.elem[i] << " ";
    }
cout << endl;
}
```

4 Si definisca una funzione **ricorsiva** "`prodotto`" che prenda in ingresso due numeri interi positivi n e m e restituisca un valore intero che rappresenti il prodotto $n \cdot m$, utilizzando esclusivamente gli operatori di confronto `==,!=,>=,>,<=,<` e le funzioni predefinite `succ` e `pred`, che calcolano il successore e il predecessore di un numero intero.

Alcune note:

(a) non è consentito utilizzare gli operatori aritmetici `+,-,*, /, %`,

(b) non è consentito utilizzare alcuna funzione di libreria;

(c) non è consentito utilizzare alcuna forma di ciclo;

(d) è consentito definire e utilizzare eventuali funzioni ausiliarie, purché a loro volta rispettino le condizioni (a), (b) e (c).

Sono dati il seguente file header `predsucc.h`:

```
#ifdef  PREDSUCC_H
#define PREDSUCC_H
// se n>=0 restituisce n+1, altrimenti abortisce il programma
int succ(int n);
// se n>=1 restituisce n-1, altrimenti abortisce il programma
int pred(int n);
#endif
```

e il corrispondente file `predsucc.o` dove sono definite `succ` e `pred`, e il seguente file principale:

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

// introdurre qui la definizione della funzione prodotto

int main()
{
  int n,m;

  do {
    cout << "dammi due numeri interi positivi: ";
    cin >> n >> m;
    if (n < 0 || m < 0)
      cout << "Valori errati, programma terminato." << endl;
    else
      cout << n << "*" << m << " = " << prodotto(n,m) << endl;
  }
  while (n >= 0 && m >= 0);
}
```

dove deve essere definita la funzione `prodotto`.

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

4 esercizio4.cc

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

// introdurre qui la definizione della funzione prodotto
int somma (int n, int m) {
  int res;
  if (m==0)
    res = n;
  else
    res = succ(somma(n,pred(m)));
// Soluzione alternativa:
//  res = somma(succ(n),pred(m));
}

int prodotto (int n, int m) {
  int res;
  if (m==0)
    res = 0;
  else
    res = somma(n,prodotto(n,pred(m)));
  return res;
}

int main()
{
  int n,m;

  do {
    cout << "dammi due numeri interi positivi: ";
    cin >> n >> m;
    if (n < 0 || m < 0)
      cout << "Valori errati, programma terminato." << endl;
    else
      cout << n << "*" << m << " = " << prodotto(n,m) << endl;
  }
  while (n >= 0 && m >= 0);
}
```