Esercizi
Programmazione I e Informatica II

13 Novembre 2015

Esercizio 1 Tipo espressione condizionale

Valutare e dare il tipo delle seguenti espressioni condizionali.

```
1  char a= 'a', b= 'b'; // a ha valore decimale 97
2  int i= 1, j= 2;
3  double x= 7.07;
4
5  i == j ? a - 1 : b + 1;
6  j % 3 == 0 ? i + 4: x;
7  j % 3 ? i + 4: x;
8
```

Soluzione dell'esercizio 1

```
(i == j) ? (a - 1) : (b + 1)     Valore 99, Tipo int
((j % 3) == 0) ? (i + 4): x      Valore 7.07, Tipo double
(j % 3) ? (i + 4): x             Valore 5.0, Tipo double
```

Ricordarsi che il tipo di un espressione *expr1? expr2: expr3* condizionale è uno solo, e dipende dai tipi delle espressioni *expr2* e *expr3*: utilizzare le regole di conversione di tipo tra *expr2* e *expr3* per trovare il tipo di tutta l'espressione. Esempio, nel terzo caso sopra, $i + 4$ ha tipo *int*, ma viene ritornato un valore 5.0 di tipo *double*, dato che *expr3* ha tipo *double*. Il tipo di *expr2* viene quindi convertito a quello di *expr3*.

Esercizio 2 Operatore virgola

Studiare la seguente porzione di codice; rispondere alle domande nei commenti. Infine controllare la correttezza delle proprie risposte eseguendo tale codice.

```
1  int a= 0;
2  double b= 3.5;
3
4  // Dichiarare c con il tipo appropriato, in modo che non ci siano conversioni di tipo
5  c= (a = 5, b = b + 0.3);
6
7  // Cosa stampa?
8  printf("%d\n%f\n%f\n", a, b, c);
9
10 // Dichiarare d con il tipo appropriato, in modo che non ci siano conversioni di tipo
11 d= (a++, --a + 2);
12
13 // Cosa stampa?
14 printf("%d\n%d\n", a, d);
15
16 int i, j, k= 3;
17 double x= 3.3;
18
19 // Dichiarare f con il tipo appropriato, in modo che non ci siano conversioni di tipo
20 f= (i = 1, j = 2, ++k + 1);
21
22 // Dichiarare g con il tipo appropriato, in modo che non ci siano conversioni di tipo
23 g= (j= k != 1, ++x * 2.0 + 1);
```

```c

  // Cosa stampa?
  printf("%d\n%d\n%d\n%f\n%d\n%f\n", i, j, k, x, f, g);
```

Soluzione dell'esercizio 2

```c
  int a= 0;
  double b= 3.5;

  double c = 0.0;
  c= (a = 5, b = b + 0.3);

  printf("%d\n%f\n%f\n", a, b, c);

  int d= 0.0;
  d= (a++, --a + 2);

  printf("%d\n%d\n", a, d);

  int i, j, k= 3;
  double x= 3.3;

  int f= (i = 1, j = 2, ++k + 1);
  double g= (j= k != 1, ++x * 2.0 + 1);

  printf("%d\n%d\n%d\n%f\n%d\n%f\n\n", i, j, k, x, f, g);
```

Output del programma precedente

```
5
3.800000
3.800000
5
7
1
1
4
4.300000
5
9.600000
```

Esercizio 3 Precedenza operatori

Studiare la seguente porzione di codice e scrivere quello che si pensa stampi. Cosa succede se si toglie la prima o la seconda coppia di parentesi, o entrambe? Controllare solo alla fine la correttezza delle proprie risposte eseguendo il codice.

```c
  int x= 0, y= 0,z= 0;
  x = (y = 2) + (z = 3);
  printf("%d %d %d\n" , x, y, z);
```

Soluzione dell'esercizio 3

```
5 2 3 (senza togliere parentesi)
5 5 3 (togliendo la prima coppia di parentesi)
error: expression is not assignable (togliendo la seconda coppia)
error: expression is not assignable (togliendo entrambe le coppie)
```

Togliendo la prima coppia, viene valutata per prima l'espressione $(y = 2)$, a cui poi si somma z. $(y = 2) + z$ NON rappresenta però un *lvalue*, quindi non può stare a sinistra di un operatore di assegnamento. Togliendo la seconda coppia di parentesi invece, viene valutata per prima $2 + z$ (che non è un *lvalue*), a cui si cerca poi di assegnare 3.

Esercizio 4 Effetti collaterali (side effects)

Studiare la seguente porzione di codice e scrivere quello che si pensa stampi. Controllare solo alla fine con un programma.

```c
  int a, b =0, c =0;
  a =++b + ++c;
  printf("%d %d %d\n", a, b, c);
  a= b+++c++;
  printf("%d %d %d\n", a, b, c);
  a= ++b + c++;
  printf("%d %d %d\n" , a, b, c);
  a= b-- + --c;
  printf("%d %d %d\n" , a, b, c);
  int q = q;
  printf("%d\n", q);
  int r = 2+ q++;
  printf("%d %d\n", r, q);

```

Soluzione dell'esercizio 4

Sul mio ambiente viene stampato questo:

```
2 1 1
2 2 2
5 3 3
5 2 2
1499863904
1499863906 1499863905
```

Alcuni compilatori C possono inizializzare una variabile a 0, ma, in generale, una variabile (non dichiarata come *static* o *extern*) può essere assegnata a qualsiasi valore. INIZIALIZZARE SEMPRE SUBITO TUTTE LE VARIABILI. Questo per rendere il codice portabile su compilatori che non effettuano automaticamente l'inizializzazione a 0.

Esercizio 5 While ed espressioni di controllo

Rispondere alle domande nei commenti per ciascuna porzione di codice

```c
// 1) Che funzione implementa questo programma?
// 2) Cosa succede se da tastiera passate un numero minore di 0?
#include <stdio.h>

int main() {
  int n= 0, res= 1;
  scanf("%d", &n);

  while (n--)
    res*= (n + 1);

  printf("%d\n", res);

return 0;
}

```

Soluzione dell'esercizio 5

Il primo programma calcola il fattoriale di n (passato da tastiera). Se n è negativo, il ciclo non termina, dato che qualsiasi valore diverso da 0 in C equivale a *true*.

Esercizio 6 Loops

- Utilizzare un ciclo *while*, un *do...while*, e un *for* per stampare tutti i numeri tra 0 e n con il valore n letto da tastiera (*scanf()*). Per esempio, se $n = 10$, la stampa deve essere 12345678910.
- Calcolare il massimo comune divisore (MCD) tra due valori di tipo *int* presi in input da tastiera. Per farlo, scandire tutti i valori tra 1 ed il minimo dei due valori, controllando per ogni valore scandito se è divisore di entrambi. Il massimo di essi corrisponde al MCD.
- Leggere 10 valori di tipo *int* da tastiera assegnadoli a ciscun elemento di un array di 10 posizioni. In seguito, risolvere ciascuno dei seguenti punti con un ciclo *for*: trovare *i)* la media dei valori memorizzati nell'array, *ii)* il massimo ed il minimo elemento dell'array, *iii)* ordinare l'array in ordine crescente utilizzando due *for* annidati, *iv)* stampare tutti gl elementi dell'array. Inizializzare max a 0 e *min* a *INT_MAX* (importare *limits.h*).

Soluzione dell'esercizio 6

Di seguito, la soluzione solo per il secondo e terzo punto.

```c
#include <stdio.h>

int main(void)  {
  int i, num1, num2, min, gcd=1;

  printf("Ricevo i due numeri per i quali calcolare il GCD: ");
  scanf("%d %d", &num1, &num2);

  min = (num1 < num2) ? num1 : num2;

  for(i=1; i <= min; i++)  {

    // Se i e' fattore di entrambi i numeri
    if(num1 % i == 0 && num2 % i== 0)
      gcd = i;
  }

  printf("GCD di %d e %d = %d\n", num1, num2, gcd);
  return 0;
}

```

```c
#include <stdio.h>
#include <limits.h>
#define ARRAY_ELEM 10

int main(void)  {

  int arrayElem[ARRAY_ELEM];

  for (int i= 0; i < ARRAY_ELEM; i++) {
    scanf("%d", &arrayElem[i]);
  }

  int sum= 0;
  for (int i= 0; i < ARRAY_ELEM; i++) {
    sum+= arrayElem[i];
  }

  printf("%f\n", (double) sum / ARRAY_ELEM);

  int max= 0;
  int min= INT_MAX;
  for (int i= 0; i < ARRAY_ELEM; i++) {
    if (arrayElem[i] > max)
      max= arrayElem[i];
    if (arrayElem[i] < min)
      min= arrayElem[i];
  }

  printf("Max = %d\nMin = %d\n", max, min);

  for (int i= 0; i < ARRAY_ELEM; i++)
    for (int j= i+1; j < ARRAY_ELEM; j++)
      if (arrayElem[i] > arrayElem[j]) {
        int tmp= arrayElem[i];
        arrayElem[i] = arrayElem[j];
        arrayElem[j] = tmp;
      }

  for (int i= 0; i < ARRAY_ELEM; i++)
  printf("Elemento [%d]= %d\n", i+1, arrayElem[i]);

}

```

Esercizio 7 Triangoli

Si scriva un programma in linguaggio C che legga da tastiera i valori delle lunghezze dei tre lati di un triangolo (detti A, B e C), e determini:

- se il triangolo e equilatero,
- se il triangolo e isoscele,
- se il triangolo e scaleno,
- se il triangolo e rettangolo.

Prima controllare se le misure dei tre lati ricevuti in input (A, B, C) rappresentano correttamente un triangolo:

- tutti i lati devono essere positivi,
- ogni lato deve essere minore della somam degli altri due,
- ogni lato deve essere maggiore della differenza degli altri due.

Soluzione dell'esercizio 7

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void) {

  float a, b, c ; /* lati del triangolo */
  /* LEGGI A, B e C */
  printf("Immetti A:"); scanf("%f", &a);
  printf("Immetti B: ");
  scanf("%f", &b);
  printf("Immetti C: ");
  scanf("%f", &c);


  if( a<=0 || b<=0 || c<=0 )
    printf("Errore: i lati devono essere positivi\n") ;
  else
    if( a>=b+c || b>=a+c || c>=a+b )
      printf("Errore: ogni lato deve essere minore della somma degli altri due\n") ;
      else
      if((b>c && a<= b-c)||
      ( b<=c && a <= c-b ) ||
      (a>c && b<= a-c)||
      ( a<=c && b <= c-a ) ||
      (a>b && c<= b-a)||
      ( a<=b && c<=a-b ) )
      printf("Errore: ogni lato deve essere maggiore della differenza degli altri due\n");
      else {


      printf("Verifico le proprieta  del triangolo di lati: %f, %f, %f\n", a, b, c) ;

  /* VERIFICA SE E' EQUILATERO (3 LATI UGUALI)*/
  if( a==b && b==c )
    printf("Il triangolo e' equilatero\n");
  else
  /* VERIFICA SE E' ISOSCELE (2 LATI UGUALI)*/
  if( a==b || b==c || a==c )
    printf("Il triangolo e' isoscele\n");
  else
      printf("Il triangolo non e' isoscele\n");
  /* VERIFICA SE E' SCALENO (3 LATI DIVERSI)*/
  if( a!=b && b!=c && a!=c )
    printf("Il triangolo e' scaleno\n");
  else
      printf("Il triangolo non e' scaleno\n");

  /* VERIFICA SE E' RETTANGOLO (TEOREMA DI PITAGORA) */
```

```
50
51    /* VERIFICA SE IL LATO A E' L IPOTENUSA */
52    if( a*a == b*b + c*c )
53        printf("Il triangolo e' rettangolo (ipotenusa A)\n");
54    else
55      printf("Il triangolo non e' rettangolo (ipotenusa A)\n");
56
57    /* VERIFICA SE IL LATO B E' L IPOTENUSA */
58    if ( b*b == a*a + c*c )
59      printf("Il triangolo e' rettangolo (ipotenusa B)\n");
60    else
61      printf("Il triangolo non e' rettangolo (ipotenusa B)\n");
62
63    /* VERIFICA SE IL LATO C E' L'IPOTENUSA */
64    if( c*c == b*b + a*a )
65      printf("Il triangolo e' rettangolo (ipotenusa C)\n");
66    else
67        printf("Il triangolo non e' rettangolo (ipotenusa C)\n");
68
69    /* VERIFICA SE IL TRIANGOLO E' RETTANGOLO */
70    if ( ( a*a == b*b + c*c ) || ( b*b == a*a + c*c ) || ( c*c == b*b + a*a ) )
71        printf("Il triangolo e' rettangolo\n");
72    else
73        printf("Il triangolo non e' rettangolo\n");
74      }
75
76    exit(0);
77  }
78
```

La funzione *exit()* serve per provocare la terminazione regolare di un programma, proprio come quando da *main* si fa *return*. Sia il parametro di *exit* sia il valore (eventuale) di *return* vengono restituiti all'ambiente da cui il programma è stato chiamato. La differenza fra *return* ed *exit()*, oltre al fatto che *exit()* è una funzione e *return* non lo è, è che *exit()* provoca l'uscita dal programma in qualunque funzione si trovi. *Return* provoca l'uscita dal programma solo se si trova in *main()*; in caso contrario provoca il ritorno alla funzione che ha chiamato quella corrente.