Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 26 febbraio 2007

COGNOME e NOME	
Matricola	

	a	b	c	d	------	*PUNTI*
1		x			2	
2		x			2	
3		x			2	
4	X				2	
5		y			1	
6			y	y	2	
7		y		y	2	
8	y				1	
9		y		y	2	
10		y	y		2	
11		y		y	2	
12	y	y	y		3	
13	y				1	
14		y	y		2	
TOT					***26***	

PARTE 1 – RISPOSTA SINGOLA - Ogni domanda ha una sola risposta VERA.
- **Una risposta esatta fa acquisire il punteggio riportato a fianco della domanda**
- **Una risposta errata viene calcolata: -1**
- **Una risposta lasciata in bianco viene calcolata: 0**

1. **(2 pt.)** Le istruzioni:

```
int a=3; int *P1 = &a ; int *P2 = &a ;
*P2 += 3;
printf("%d",*P2);
```

a) stampano il valore 3;
b) stampano il valore 6;
c) contengono un errore di gestione della memoria;
d) stampano l'indirizzo della variabile **a**.
.

2. (**2 pt**) Indicare quale delle seguenti affermazioni riguardanti le funzioni nel linguaggio C è corretta:

a) una funzione ha sempre almeno una variabile dichiarata al suo interno;
b) una funzione può non avere nessun valore di ritorno;
c) una funzione ha sempre almeno un parametro;
d) una volta invocata, è sempre garantito che una funzione termini in un tempo finito.

3. **(2 pt.)** Dato un file sorgente C contenente la seguente direttiva

#define COSTANTE1 15

a) tale direttiva è perfettamente equivalente all'istruzione
`const int COSTANTE1 = 15 ;`
b) il preprocessore sostituisce testualmente ogni occorrenza della parola COSTANTE1 con la sequenza di due caratteri '1' e '5';
c) la parola COSTANTE1 è ancora presente nel file prodotto dal preprocessore e preso in ingresso dal compilatore;
d) la presenza di tale direttiva e della seguente istruzione

```
int COSTANTE1 ;
```

all'interno di una qualche funzione nello stesso file sorgente non genera errori di compilazione.

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 26 febbraio 2007

COGNOME e NOME	
Matricola	

4. **(2 pt)** Data la **struct elem** { int inf; struct elem *next } , la seguente funzione:

```
void fun(struct elem **testa)
{
struct elem *p ;
int i ;
p = (struct elem *) malloc(sizeof(struct elem)) ;
p->inf = 1 ;
*testa = p ;
for (i = 2 ; i <= 10 ; i++) {
p = p->next ;
p = (struct elem *) malloc(sizeof(struct elem)) ;
p->inf = i ;
}
}
```

a) alloca 10 elementi, contenenti i valori 1, 2, ..., 10;
b) alloca 10 elementi, contenenti i valori 1, 2, ..., 10, e li concatena l'uno all'altro;
c) causa errori durante la sua esecuzione;
d) alloca un elemento di troppo.

PARTE 2 – (POSSIBILI) RISPOSTE MULTIPLE - Ogni domanda può avere una o più risposte CORRETTE.
- **Ogni risposta esatta viene calcolata: +1**
- **Ogni risposta errata viene calcolata: -0.5**
- **Una risposta lasciata in bianco viene calcolata: 0**

5 Il seguente frammento di codice

```
float b;
b = 5/2 ;
```

a) causa un errore a tempo di compilazione;
b) assegna il valore 2 alla variabile b;
c) assegna il valore 2.5 alla variabile b;
d) causa un errore a tempo di esecuzione.

6. L'istruzione **FILE *f = `fopen("DATI", "w");`**

a) apre in scrittura un file di nome DATI, necessariamente non di tipo testo;
b) fallisce se il file di nome DATI esiste già;
c) non scrive nulla all'interno del file dopo averlo aperto;
d) se il file DATI non esiste, lo crea.

7. La memoria occupata da un oggetto allocato dinamicamente all'interno di una funzione

a) è pre-allocata all'inizio dell'esecuzione del programma;
b) è allocata quando l'istruzione di allocazione viene eseguita;
c) è liberata quando la funzione termina;
d) se non esplicitamente deallocata, è liberata solo al termine del programma.

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 26 febbraio 2007

COGNOME e NOME	
Matricola	

8. In C, una stringa:

a) ha una lunghezza pari al numero di caratteri che precedono il terminatore;
b) se il terminatore non è esplicitamente inserito, è assunta come implicitamente terminata in modo tale da avere un numero di caratteri uguale alla dimensione dell'array meno 1;
c) può contenere un numero arbitrariamente grande di caratteri, anche maggiore delle dimensioni dell'array utilizzato per rappresentarla, purché siano opportunamente terminati dal carattere di fine stringa;
d) contiene elementi di tipo **char *.**

9. Dato un *array* così dichiarato

```
char a[5] ;
```

all'interno di una funzione,

a) l'*array* continua ad occupare spazio in memoria finché non si dealloca esplicitamente la memoria da esso occupata;
b) l'espressione **a** fornisce l'indirizzo dell'*array*;
c) l'istruzione ***(++a) = 'c';** è corretta ed assegna il valore **'c'** al secondo elemento dell'*array;*
d) l'istruzione **char c = *(a+3)**; è corretta ed assegna alla variabile **c** il contenuto del quarto elemento dell'*array*.

10. Dato il seguente programma (ogni statement è numerato in sequenza)

```
1:  float a = 3.1 ;
2:
3:  void fun(int a, int b)
4:  {
5:   a += b ;
6:  }
7:  main()
8:  {
9:   fun(a, 3) ;
10:   printf("%g\n", a) ;
11:  }
```

a) la variabile **a** definita alla riga 1 ha scope relativo a tutto il programma;
b) la variabile **a** definita alla riga 1 ha tempo di vita relativo **a** tutto il programma;
c) prima dell'assegnamento, la variabile **a** riferita alla riga 5 ha un valore diverso da 3.1;
d) l'assegnamento alla riga 5 fa aumentare il valore della variabile **a** definita alla riga 1.

11. Dato un ciclo **for** o **while**,

a) mediante l'istruzione **break** si esce dal ciclo più interno contenente l'istruzione e da altri eventuali cicli esterni;
b) mediante l'istruzione **break** si esce solo dal ciclo più interno contenente l'istruzione;
c) mediante l'istruzione **continue** si termina l'esecuzione del ciclo più interno contenente l'istruzione e si continua l'esecuzione degli eventuali cicli più esterni;
d) mediante l'istruzione **continue** non si termina l'esecuzione del ciclo che contiene l'istruzione.

12. Il prototipo di una funzione:

a) specifica il tipo dei parametri di ingresso ed uscita di una funzione;
b) è indipendente dal codice presente nel corpo della funzione;
c) è utilizzato dal compilatore per verificare, tra l'altro, la correttezza del tipo dei parametri attuali utilizzati nelle chiamate alla funzione, e per effettuare eventuali conversioni implicite;
d) deve specificare anche il nome dei parametri formali.

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 26 febbraio 2007

COGNOME e NOME	
Matricola	

13.Il ciclo seguente

```
int i=4;
for( ;i>0; ) {
if (i==1)
break;
printf("%d", i);
i--;
}
```

a) stampa gli interi tra 4 e 2;
b) stampa gli interi tra 4 e 0 tranne l'1;
c) stampa gli interi tra 4 e 7;
d) non stampa alcun valore poiché la printf (“%d”, i); interna al ciclo non viene mai eseguita.

14. Date la stringa char str[]="Ciao" e la funzione:

```
int myfunction(char *p)
{int i=0;
while(*p)
{p=p+1;
i++;}
return i;}
```

quale/i delle seguenti affermazioni è/sono corrette:

```
a) myfunction(str) causa un errore a tempo di compilazione;
b) myfunction(str) da come valore di ritorno il valore 4;
c) è equivalente a
int myfunction(char *p)
{ int i=0;
while(*p++)
i++;
return i;
}
d) myfunction(str) da come valore di ritorno l'indirizzo della cella di
memoria dell'ultimo carattere della stringa.
```