IEIM 2018-2019

Esercitazione V
"Puntatori, Matrici, Funzioni"

Alessandro A. Nacci
alessandro.nacci@polimi.it - www.alessandronacci.it

Puntatori e memoria

Esercizio 2

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO
9	

ATTENZIONE!

Il comportamento della memoria mostrato in questo esercizio non è del tutto coerente con quanto avviene su un reale calcolatore. L'esempio mostrato è però funzionale alla spiegazione del comportamento dei puntatori in C.

```
}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	~	b
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	~	mat
5	~	mat
6	~	mat
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	~	mat
5	~	mat
6	~	mat
7	~	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

11

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	~	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8	3	d
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8	3	d
9	~	e

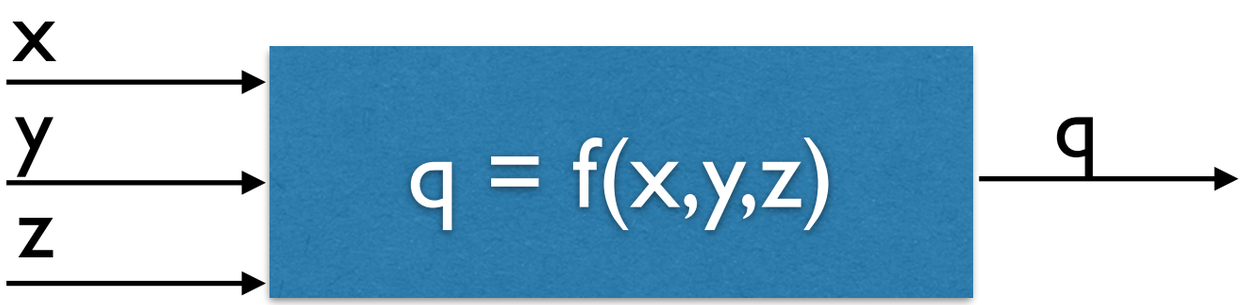
```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Le funzioni

- Le funzioni sono "blocchi" di codice che prendono in ingresso dei valori tramite i *parametri* e restituiscono, dopo della computazione, un risultato.




```
int foo(int x, int y, int z)
{
     int q;


     // codice vario
     // che effettua calcoli


     return q;

}
```

Le funzioni

Tipo dato in uscita (**void** se non

Nome

Parametri in ingresso con tipo

```
int foo(int x, int y, int z)
{
    int q;
    // codice vario
    // che effettua calcoli

    return q;
}
```

Singolo valore in uscita
(in C si può restituire un solo valore alla volta)

Le funzioni con gli array

- Per alcuni motivazioni tecniche del C, con gli array ci sono alcuni "problemini"
- Semplificando, In C non è possibile fare **`return`** di un array
- **In C, quando si passa in ingresso un array come parametro, le modifiche fatte su quel parametro agiscono direttamente sull'array originale passato dal chiamante.**

Le funzioni con gli array

```
void foo(int x_arr[])
{
    x_arr[3] = 13;
    return;
}


int main()
{
  int x_arr[10];
  x_arr[3]= 2;

  printf(“%d”,x_arr[3]); // Stampa 2

  foo(x_arr);

  printf(“%d”,x_arr[3]); // Stampa 13
}
```


WARNING
MATRICI E FUNZIONI
Il passaggio di una matrice ad una funzione
e' sempre per indirizzo e mai per copia.

Esercizio 1: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo1(int *mat)
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  *(mat+i*C+j) = (i+j) * 2;
}
```

Puntatore ad intero!

Calcolo manuale dello spiazzamento

Valore contenuto all'indirizzo tra parentesi

```c
*(mat+i*C+j)
```

INDIRIZZO

Esercizio 1: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo2(int mat[R][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 3;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice RxC

Accesso comodo alla matrice!

Esercizio 1: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

Il valore di mat cambia

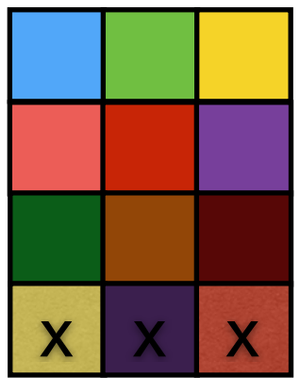
Il valore di mat cambia

Il valore di mat cambia

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

Tutte il materiale sarà disponibile sul mio sito internet!

www.alessandronacci.it