IEIM 2019-2020

Esercitazione V
"Puntatori"

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;

}
```

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

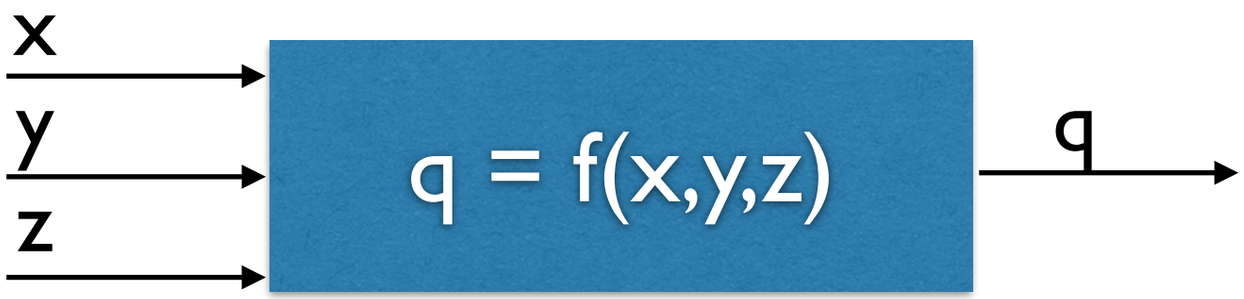
```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

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

}
```

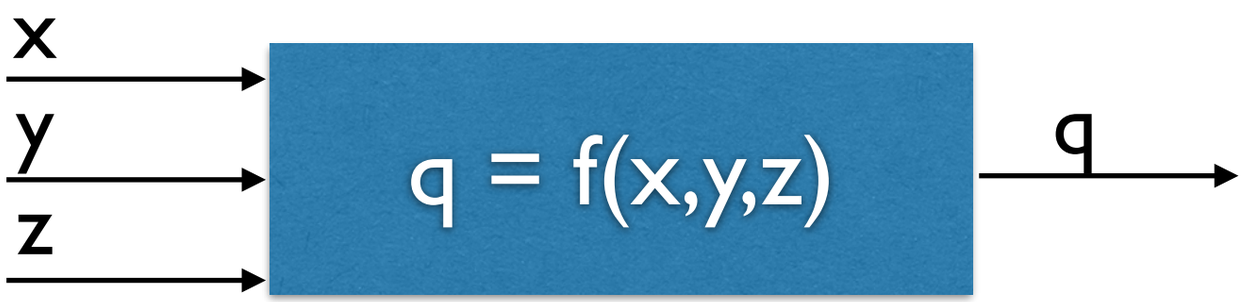
Le funzioni

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



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

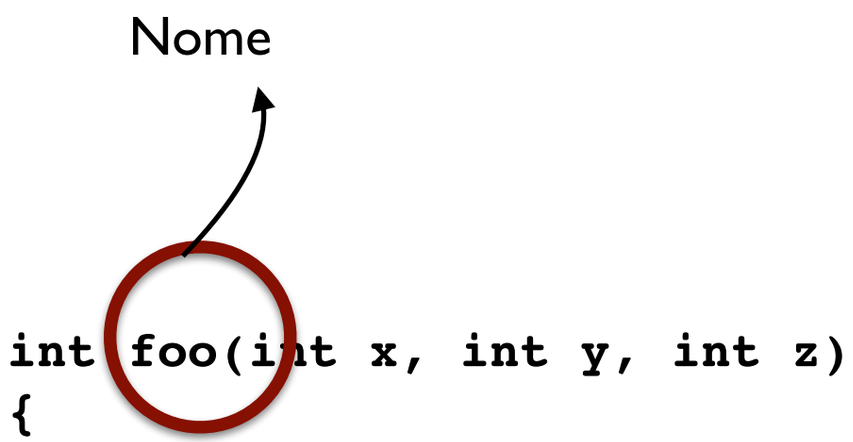
```
int foo(int x, int y, int z)
{
     int q;

     // codice vario
     // che effettua calcoli


     return q;

}
```

Le funzioni



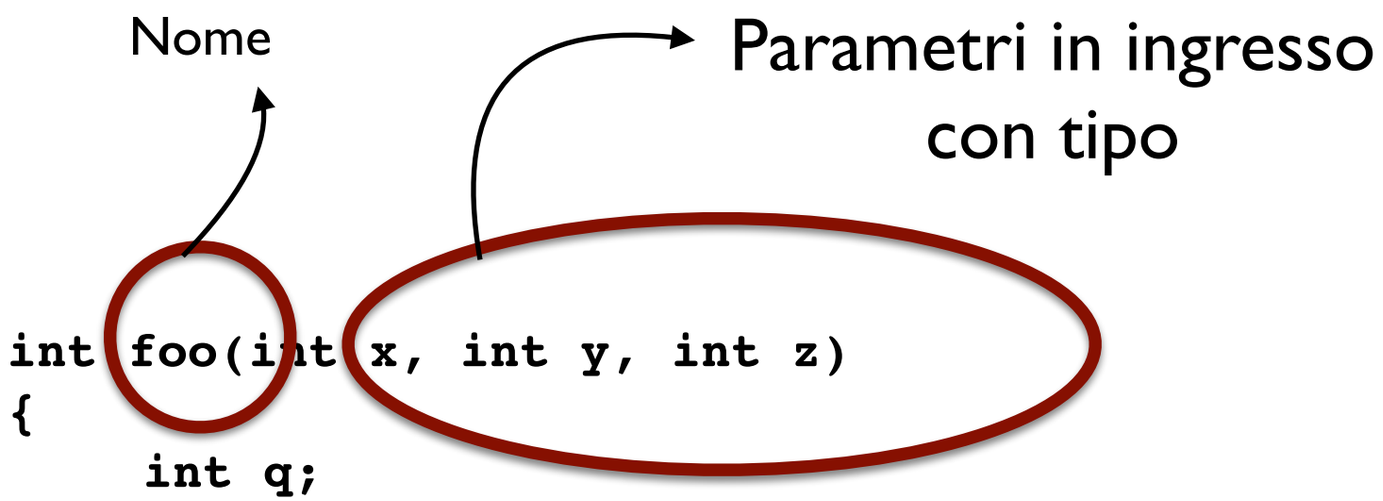

```
int foo(int x, int y, int z)
{
     int q;

     // codice vario
     // che effettua calcoli


     return q;

}
```

Le funzioni

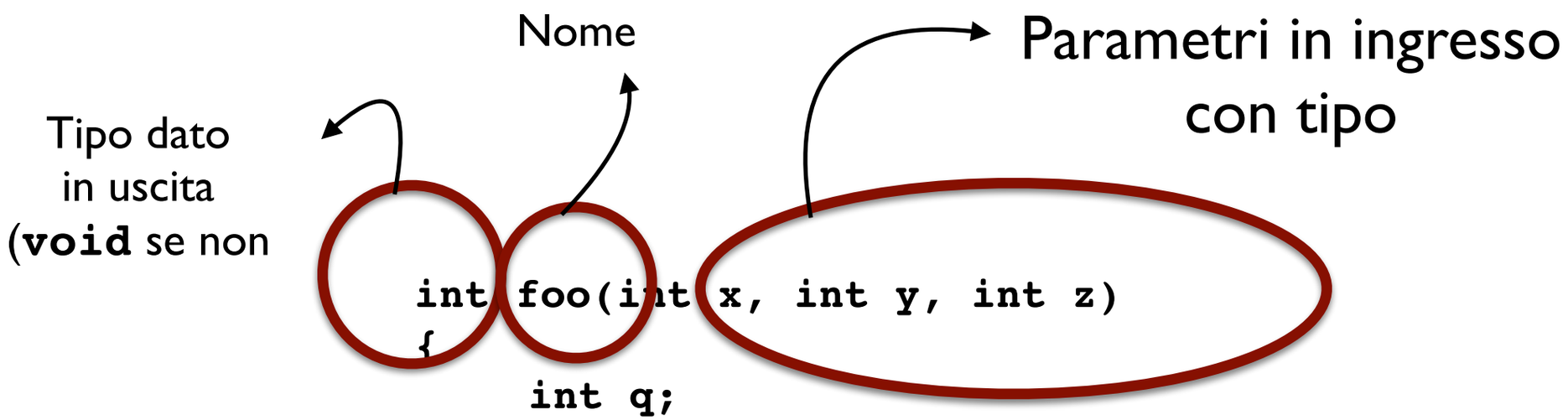



```
    // codice vario
    // che effettua calcoli


    return q;

}
```

Le funzioni




```
    // 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

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
}
```

Vediamo cosa succede in memoria

```
int somma_speciale(int d, int c)
{
    int q = 3;
    return d + c + q;
}

int main()
{
    int a;
    int b;
    int c;

    a = 2;
    b = 5;

    c = somma_speciale(a,b);
}
```

Address	Value	Notes
0	2	a
1	5	b
2	10	c
3	10	somma_speciale/return
4	2	somma_speciale/a
5	5	somma_speciale/b
6	3	somma_speciale/q

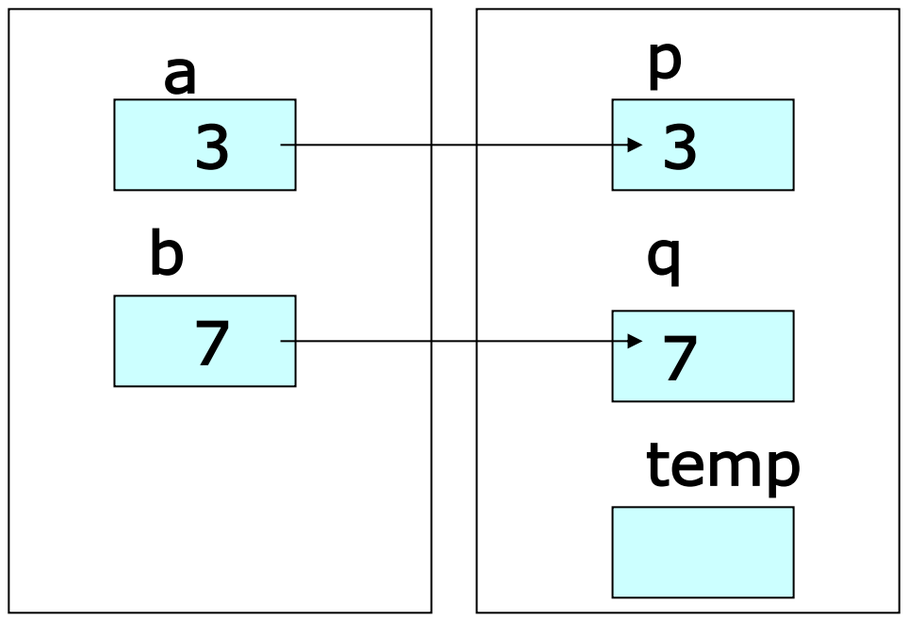
Esempio: scambio valori

- Si scriva una funzione in C che, *dati* due valori interi, gli *restituisce* scambiati

Un esempio: funzione swap

Esempio: scambio di 2 valori interi


```
void swap (int p, int q) {
        int temp;
        temp = p;
        p = q;
        q = temp;
}
```



- Nel main: swap(a,b)

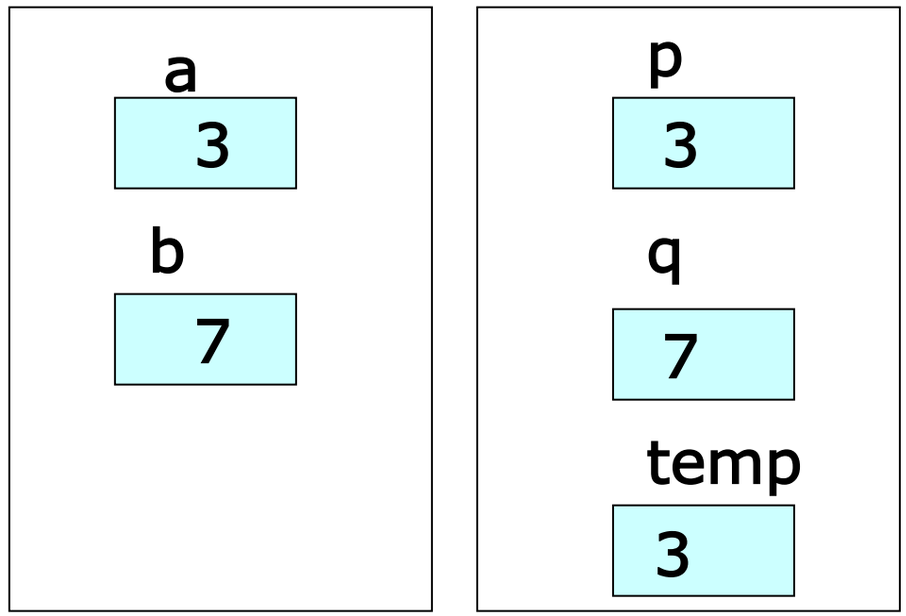
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {
        int temp;
        temp = p;
        p = q;
        q = temp;
}
```

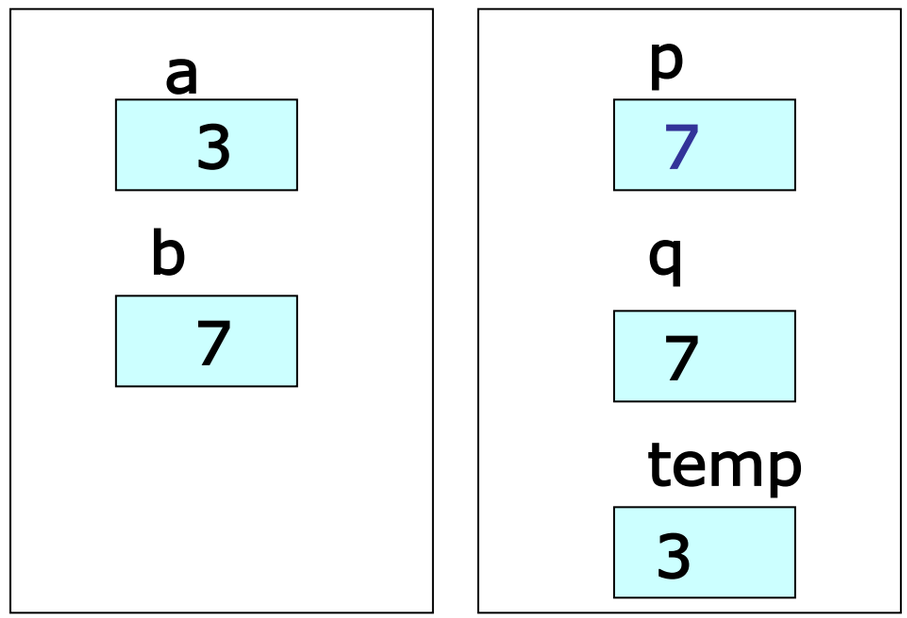


- Nel main: swap(a,b)

Un esempio: funzione swap

Esempio: scambio di 2 valori interi


```
void swap (int p, int q) {
        int temp;
        temp = p;
        p = q;
        q = temp;
}
```



- Nel main: swap(a,b)

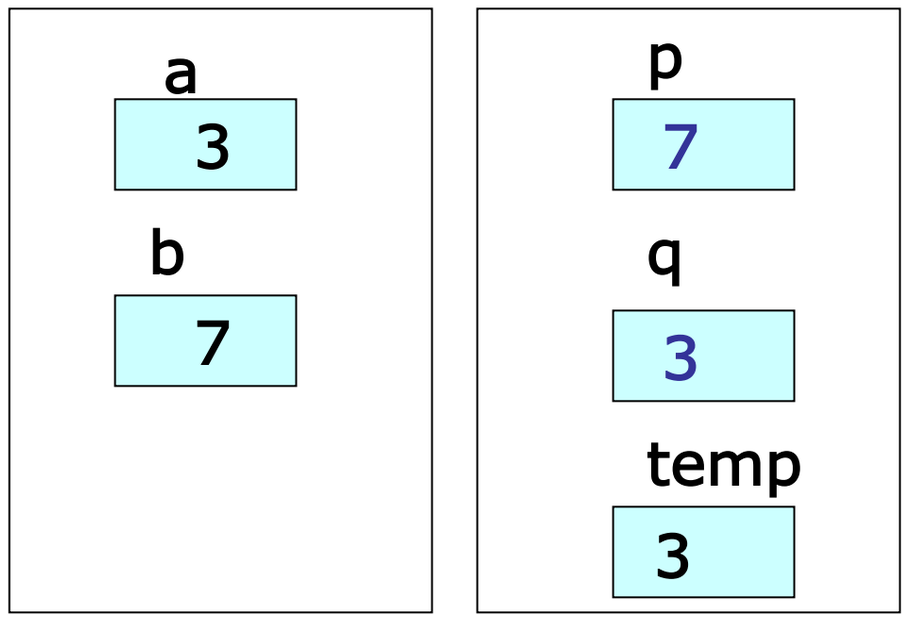
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {
        int temp;
        temp = p;
        p = q;
        q = temp;
}
```

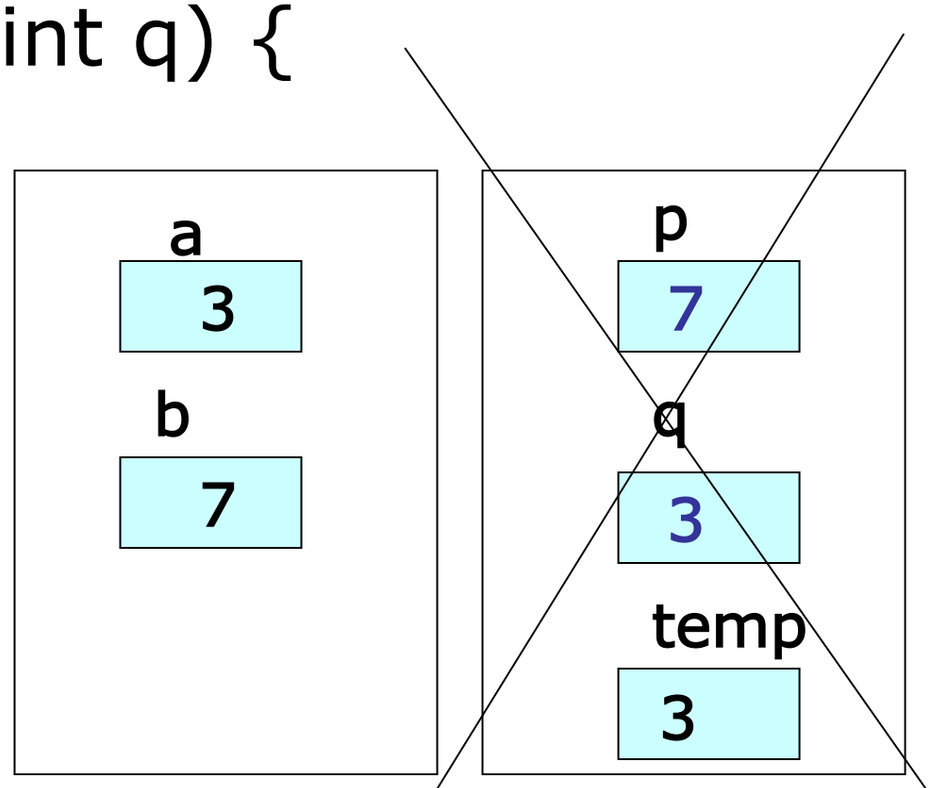


- Nel main: swap(a,b)

Un esempio: funzione swap

Esempio: scambio di 2 valori interi


```
void swap (int p, int q) {
        int temp;
        temp = p;
        p = q;
        q = temp;
}
```



Al termine dell'esecuzione di `swap` le variabili nel main restano inalterate!

ap(a,b)

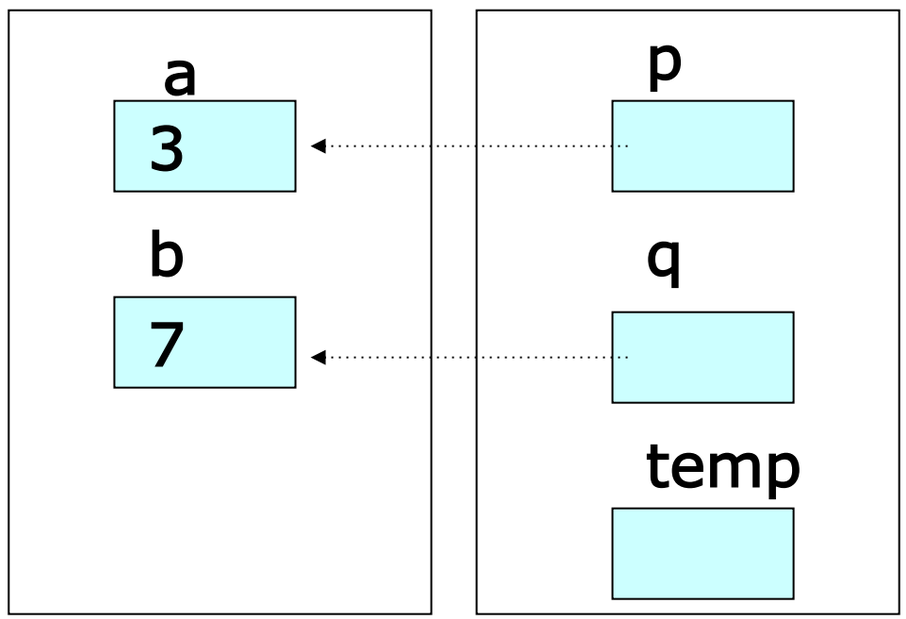
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}
```



- Nel main: swap(&a, &b)

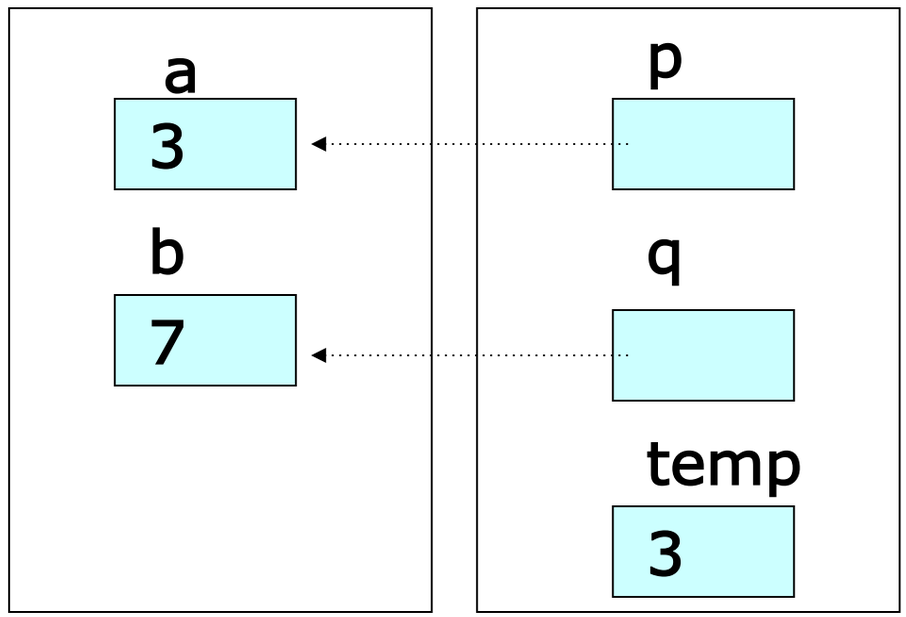
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}
```



- Nel main: swap(&a, &b)

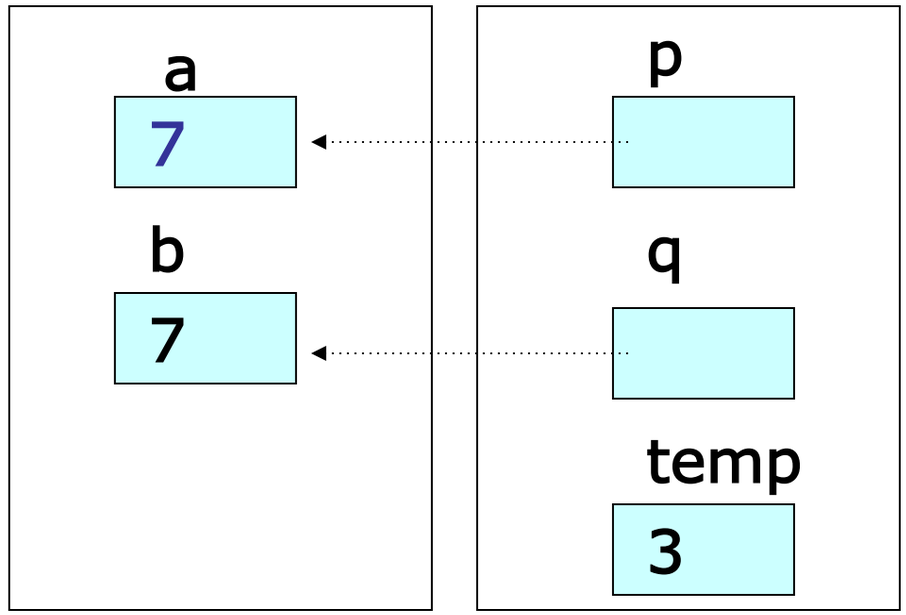
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}
```



- Nel main: swap(&a, &b)

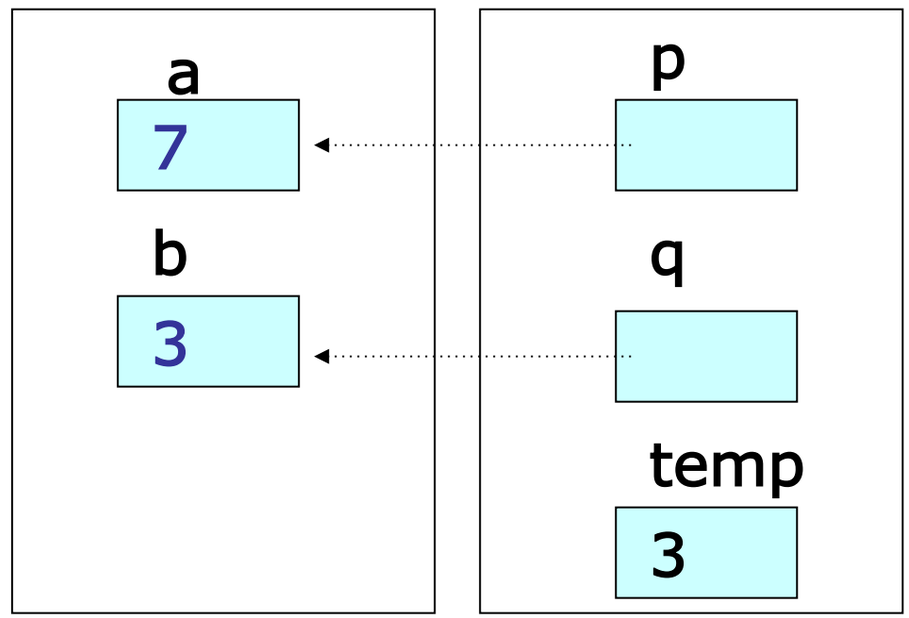
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}
```



- Nel main: swap(&a, &b)

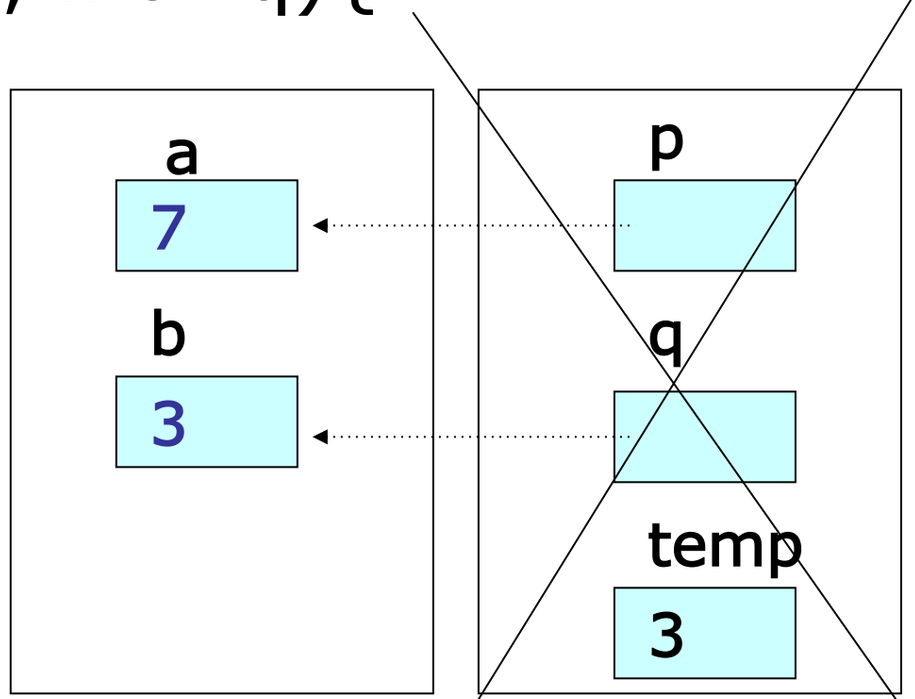
Un esempio: funzione swap

Esempio: scambio di 2 valori interi

POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){
        int temp;
        temp = *p;
        *p = *q;
        *q = temp;
}
```



Al termine dell'esecuzione di `swap` le variabili nel main vengono modificate

ap(&a, &b)

Fonti per lo studio + Credits

- Fonti per lo studio
  - Binky Pointer Fun Video: http://cslibrary.stanford.edu/104/

- Credits
  - Gianluca Palermo

Tutte il materiale sarà disponibile sul mio sito internet!

www.alessandronacci.it