A.A. 08/09

Fondamenti di Programmazione

(canale E-O)

Docente: Prof.ssa Tiziana Calamoneri
calamo@di.uniroma1.it

Esercitatore: Dott. Roberto Petroccia
petroccia@di.uniroma1.it

Pagina del corso:
http://twiki.di.uniroma1.it/twiki/view/Programmazione1/EO/WebHome

---

Esercitazione del 05/11/08

Indice

1. Richiami sulla call by value e call by reference
2. Array come parametri di funzioni
3. Esercizi su call by reference ed array

05/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Call by value

```
void    raddoppia    (int x) {
        x *= 2;
}
int main() {
    int y = 3;
    raddoppia(y);
    printf("La y vale %d\n", y);
}
```

Call by value

```
void    raddoppia    (int x) {
        x *= 2;
}
int main() {
    int y = 3;
    raddoppia(y);
    printf("La y vale %d\n", y);
}
```

Call by value

```
void    raddoppia    (int x) {
        x *= 2;
}
int main() {
    int y = 3;
    raddoppia(y);
    printf("La y vale %d\n", y);
}
```

Alla chiamata della funzione viene creata nuova variabile x che contiene lo stesso valore di y

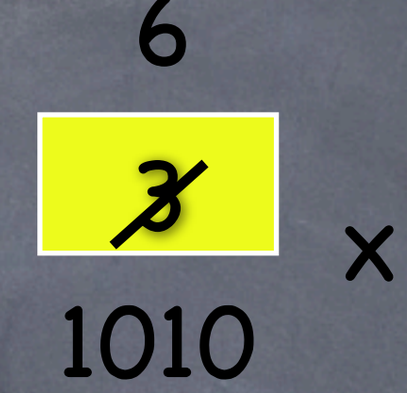
Call by value

```
void    raddoppia    (int x) {
        x *= 2;
}
int main() {
    int y = 3;
    raddoppia(y);
    printf("La y vale %d\n", y);
}
```

Alla chiamata della funzione viene creata nuova variabile x che contiene lo stesso valore di y

Call by value

```
void    raddoppia    (int x) {
        x *= 2;
}
```

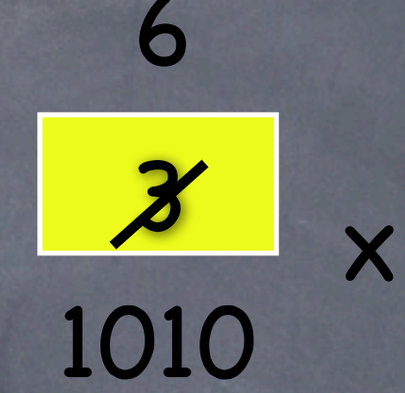


x viene modificata ma non y

```
int main() {
    int y = 3;
    raddoppia(y);
    printf("La y vale %d\n", y);
}
```

Call by value

```
void    raddoppia    (int x) {
        x *= 2;
}
```



x viene modificata ma non y

```
int main() {
    int y = 3;
    raddoppia(y);
    printf("La y vale %d\n", y);
}
```

y vale sempre 3

Call by reference

```
void    raddoppia    (int * x) {
        *x *= 2;
}
int main() {
    int y = 3;
    raddoppia(&y);
    printf("La y vale %d\n", y);
}
```

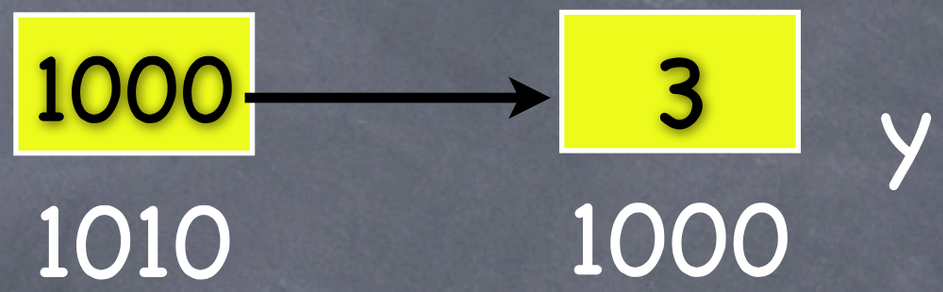
Call by reference

```
void    raddoppia    (int * x) {
        *x *= 2;
}
int main() {
    int y = 3;
    raddoppia(&y);
    printf("La y vale %d\n", y);
}
```

modificata il valore puntato da x, quindi y

Call by reference

```
void    raddoppia    (int * x) {
        *x *= 2;
}
```

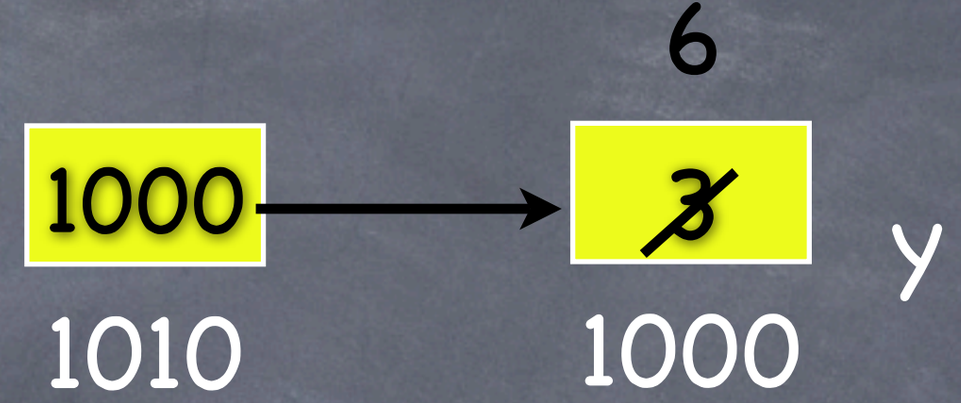


modificata il valore puntato da x, quindi y

```
int main() {
    int y = 3;
    raddoppia(&y);
    printf("La y vale %d\n", y);
}
```

Call by reference

```
void    raddoppia    (int * x) {
        *x *= 2;
}
```

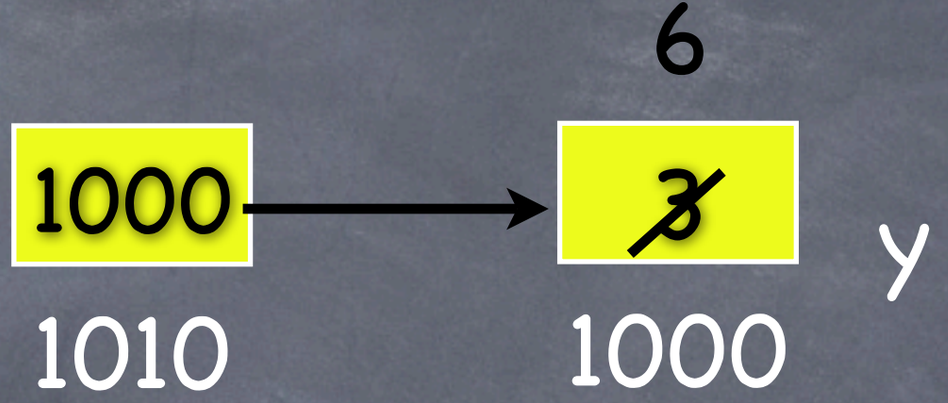


modificata il valore puntato da x, quindi y

```
int main() {
    int y = 3;
    raddoppia(&y);
    printf("La y vale %d\n", y);
}
```

Call by reference

```
void    raddoppia    (int * x) {
        *x *= 2;
}
```



modificata il valore puntato da x, quindi y

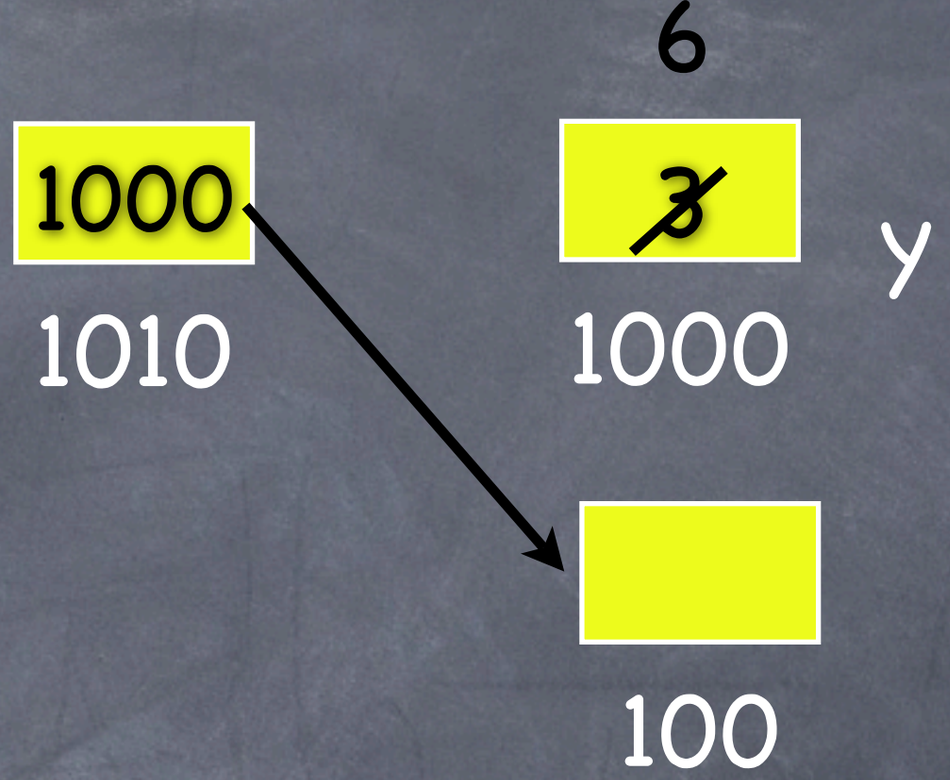
```
int main() {
    int y = 3;
    raddoppia(&y);
    printf("La y vale %d\n", y);
}
```

Viene creata una copia che pero' contiene lo stesso indirizzo di memoria, cambiando il valore in memoria cambiamo il valore della variabile puntata

Call by reference

```
void    raddoppia    (int * x) {
        *x *= 2;
        x = 100;
        *x += 1;
}

int main() {
    int y = 3;
    raddoppia(&y);
    printf("La y vale %d\n", y);
}
```

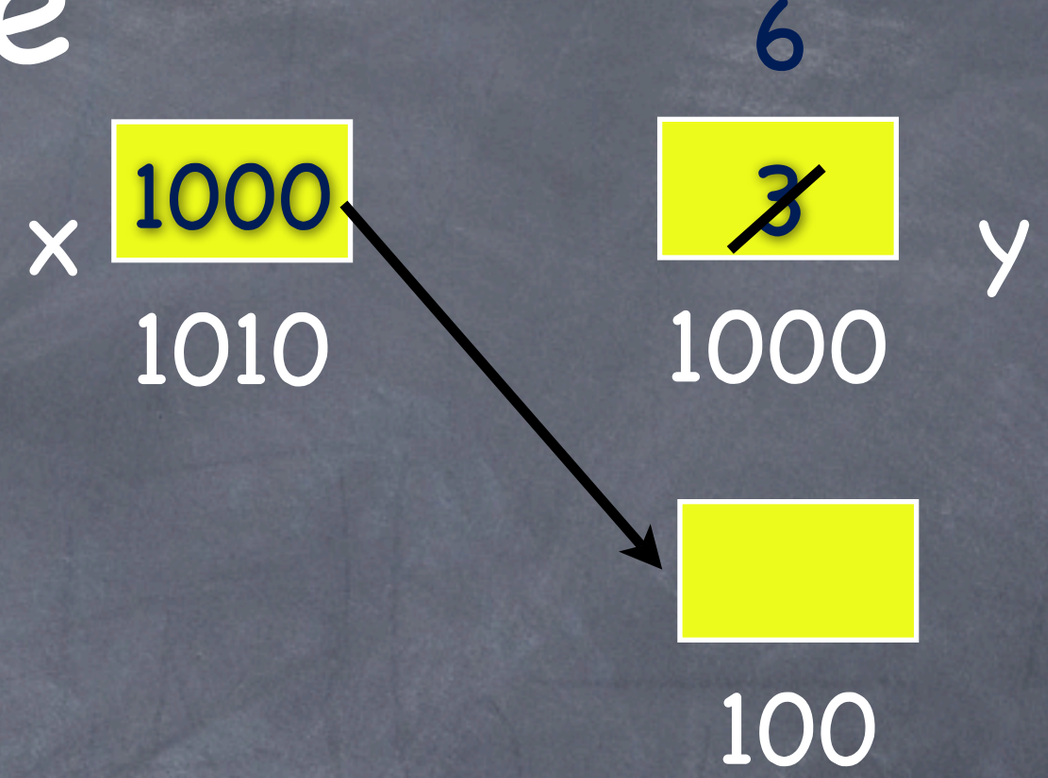


Viene creata una copia che pero' contiene lo stesso indirizzo di memoria, cambiando il valore in memoria cambiamo il valore della variabile puntata

Call by reference

```
void    raddoppia    (int * x) {
          *x *= 2;
          x = 100;
          *x += 1;
}

int main() {
        int y = 3;
        raddoppia(&y);
        printf("La y vale %d\n", y);
}
```

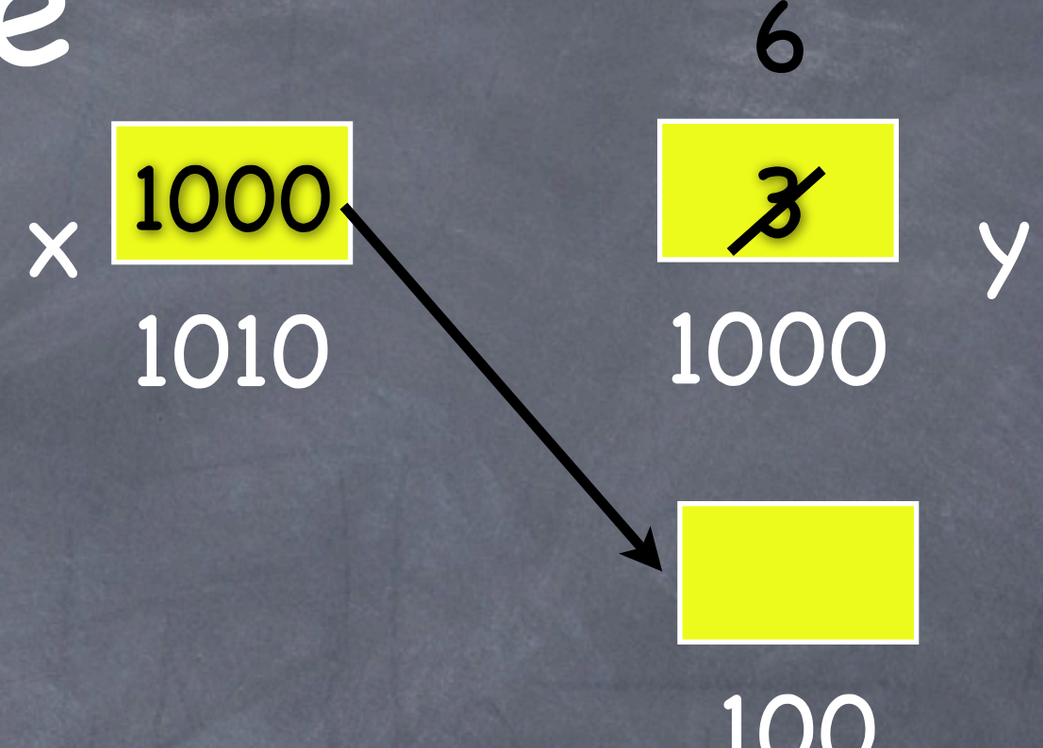


potrebbe essere memoria sporca
o protetta

Viene creata una copia che pero' contiene lo stesso indirizzo di memoria, cambiando il valore in memoria cambiamo il valore della variabile puntata

Call by reference

```
void    raddoppia    (int * x) {
          *x *= 2;
          x = 100;
          *x += 1;
}
int main() {
        int y = 3;
        raddoppia(&y);
        printf("La y vale %d\n", y);
}
```



Non cambia piu' il valore di y, anzi potrebbe mandare il programma in errore

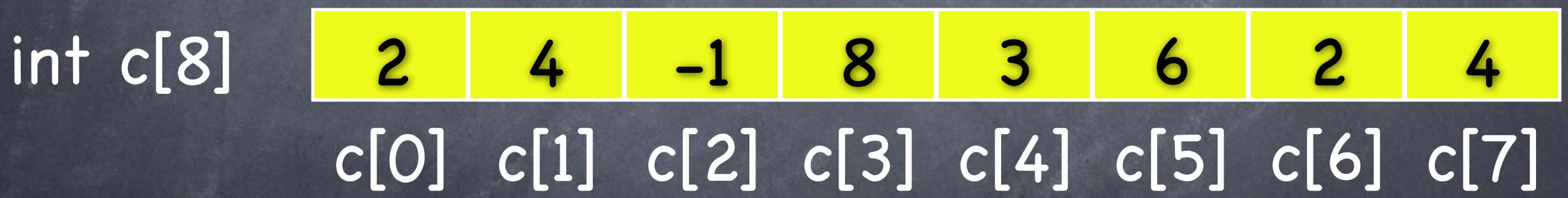
Viene creata una copia che pero' contiene lo stesso indirizzo di memoria, cambiando il valore in memoria cambiamo il valore della variabile puntata

Array

- L'array è un insieme di locazioni di memoria sequenziali

Tipo Nome Dimensione

```
int arrayInteri[100]
```



Array

- L'array è un insieme di locazioni di memoria sequenziali

Tipo Nome Dimensione

```
int arrayInteri[100]
```

int c[8]

2	4	-1	8	3	6	2	4
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]

Tutte stesso nome

Array

- L'array è un insieme di locazioni di memoria sequenziali

Tipo Nome Dimensione

```
int arrayInteri[100]
```

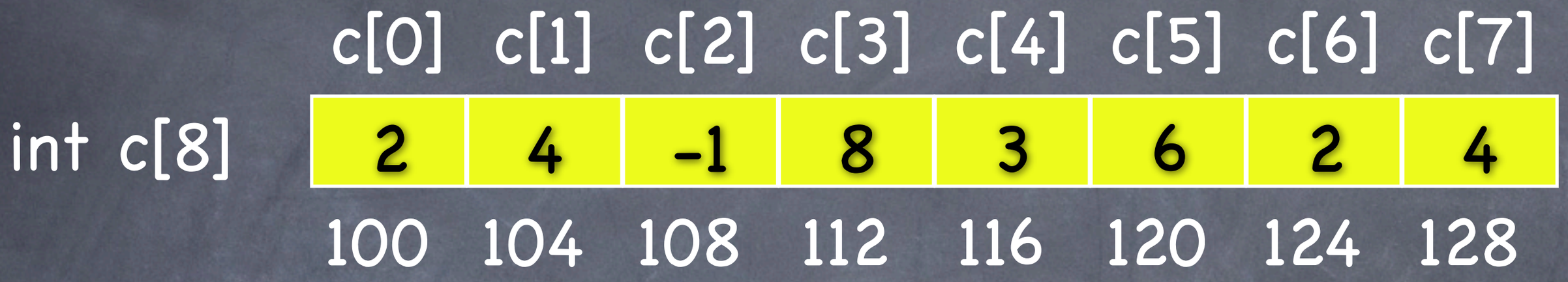
int c[8]

2	4	-1	8	3	6	2	4
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]

Tutte stesso nome

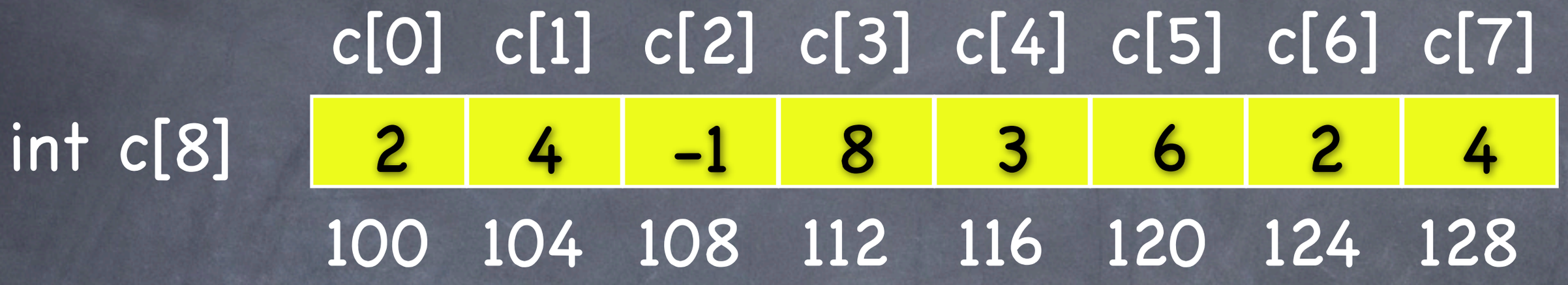
c[3] = 8

Array



05/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Array



La variabile c contiene l'indirizzo della prima locazione di memoria

Array

	c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]
int c[8]	2	4	-1	8	3	6	2	4
	100	104	108	112	116	120	124	128

La variabile c contiene l'indirizzo della prima locazione di memoria

c ha lo stesso valore di &c[0], in questo caso 100

Funzioni con array

```
void  stampaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            printf("%d ", A[i]);
      printf("\n");
   }
```

Funzioni con array

```
void  stampaArray (int  A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

```
void raddoppiaArray (int  A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        A[i] *= 2;
}
```

Funzioni con array

```
void  stampaArray (int  A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

```
void raddoppiaArray (int  A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        A[i] *= 2;
}
```

```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

Funzioni con array

```
void  stampaArray (int  A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

```
void raddoppiaArray (int  A[], int n) {
        int i;
        for (i = 0; i < n; i++)
            A[i] *= 2;
    }
```

Cosa succede ai valori del vettore vengono raddoppiati oppure no?

Funzioni con array

```
void  stampaArray (int  A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

```
void raddoppiaArray (int  A[], int n) {
        int i;
        for (i = 0; i < n; i++)
            A[i] *= 2;
    }
```

Cosa succede ai valori del vettore vengono raddoppiati oppure no?

SI VENGONO RADDOPPIATI

Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```

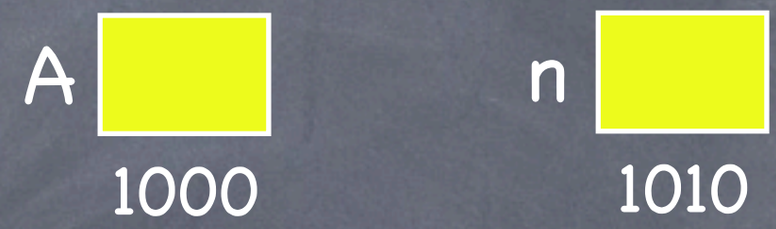
```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

y

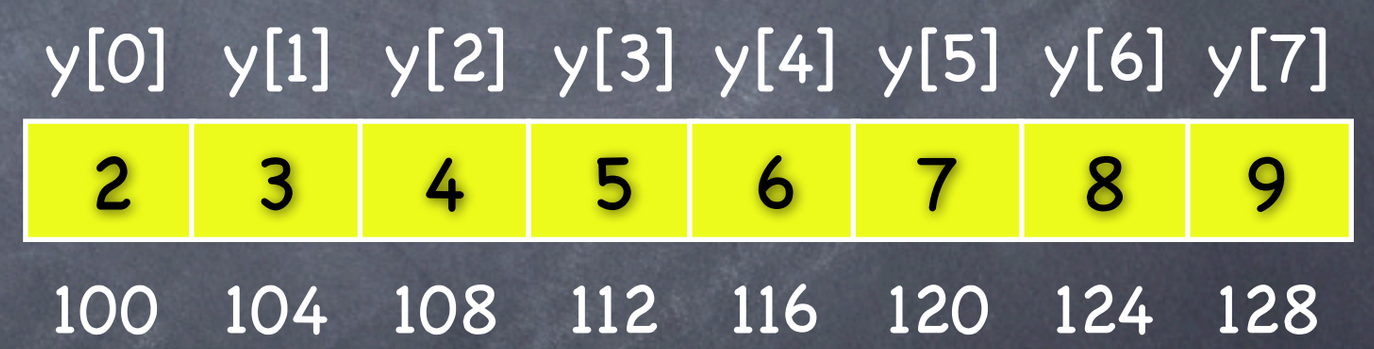
y[0]	y[1]	y[2]	y[3]	y[4]	y[5]	y[6]	y[7]
2	3	4	5	6	7	8	9
100	104	108	112	116	120	124	128

Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```

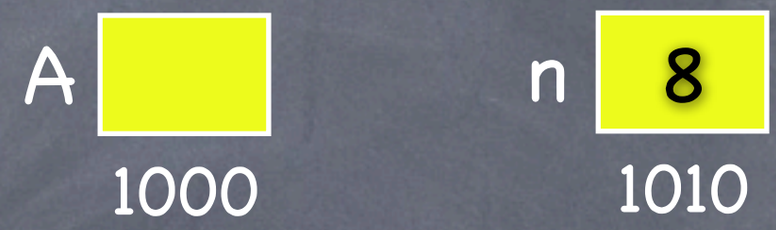



```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

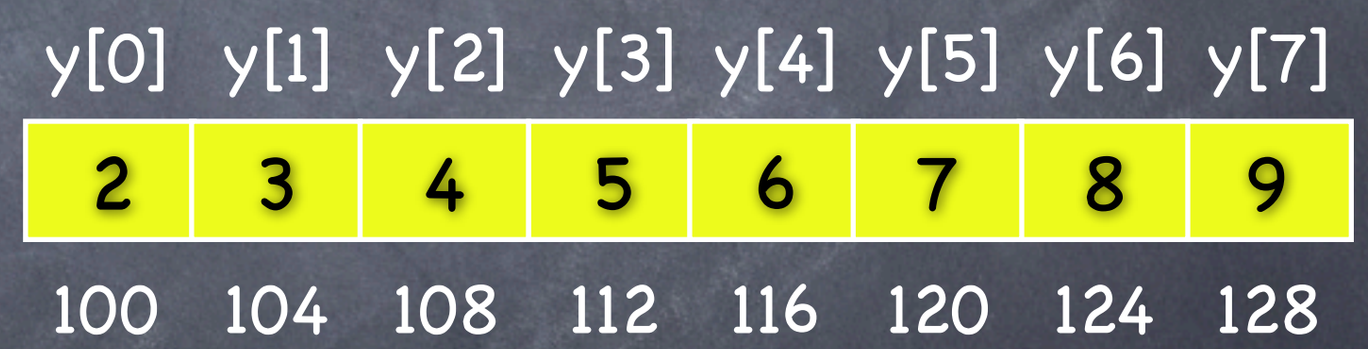


Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```

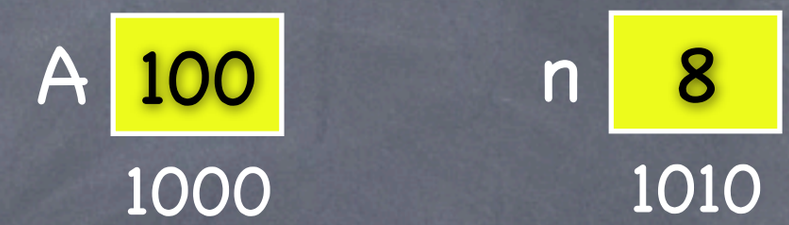



```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

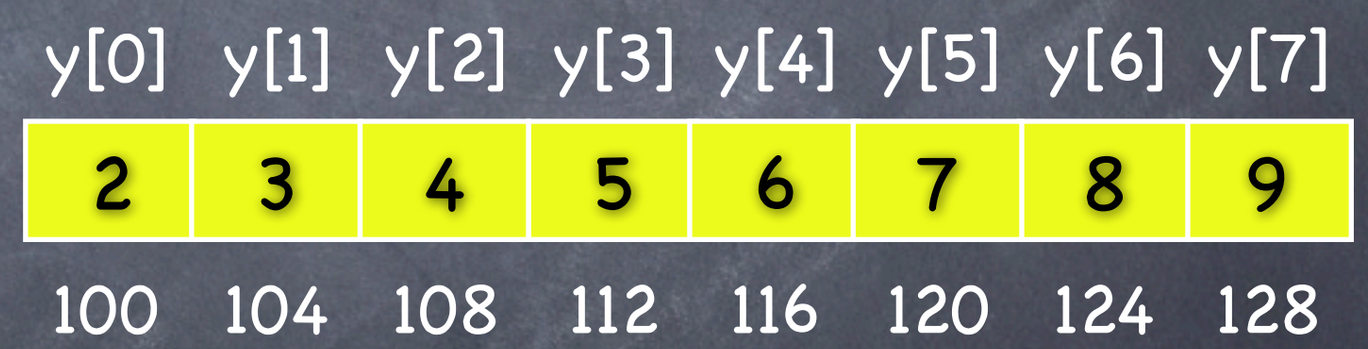


Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```




```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```



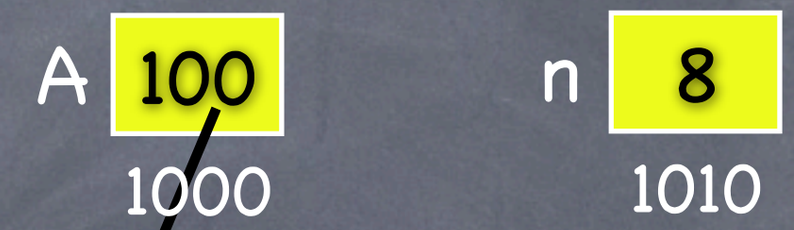
Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```


```
int main() {
      int i, n = 0;
      scanf(“%d”, &n);
      int y[n];
      for (i = 0; i < n; i++)
            scanf(“%d”, &y[i]);
      raddoppiaArray(y, n);
      stampaArray(y, n);
}
```

Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```

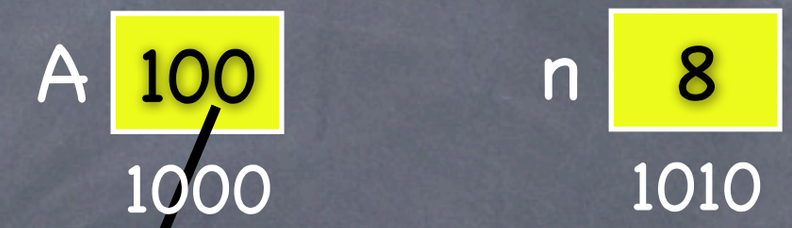


Scrivere A[i] vuol dire spostati di "i" elementi a partire dall'indirizzo di A = &A[0]

```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```

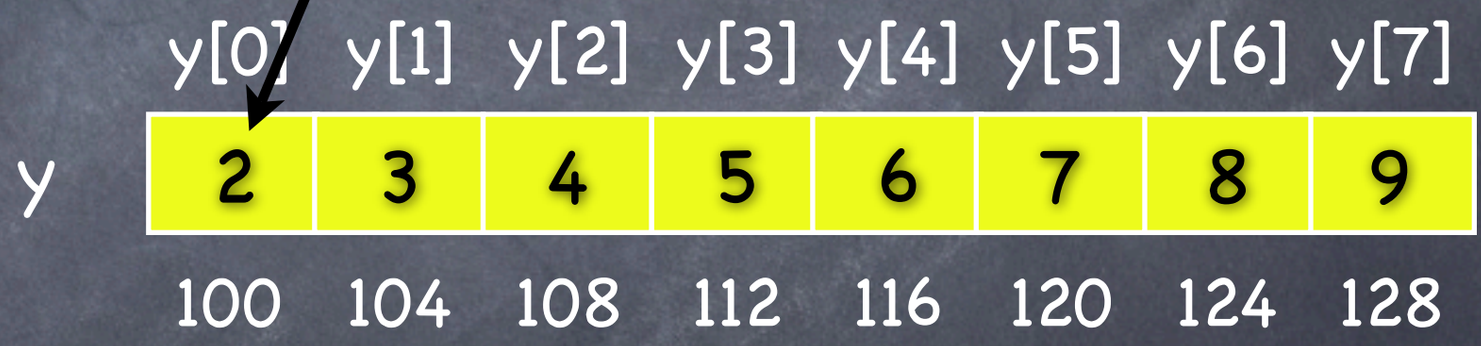
Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```



Scrivere A[i] vuol dire spostati di "i" elementi a partire dall'indirizzo di A = &A[0]

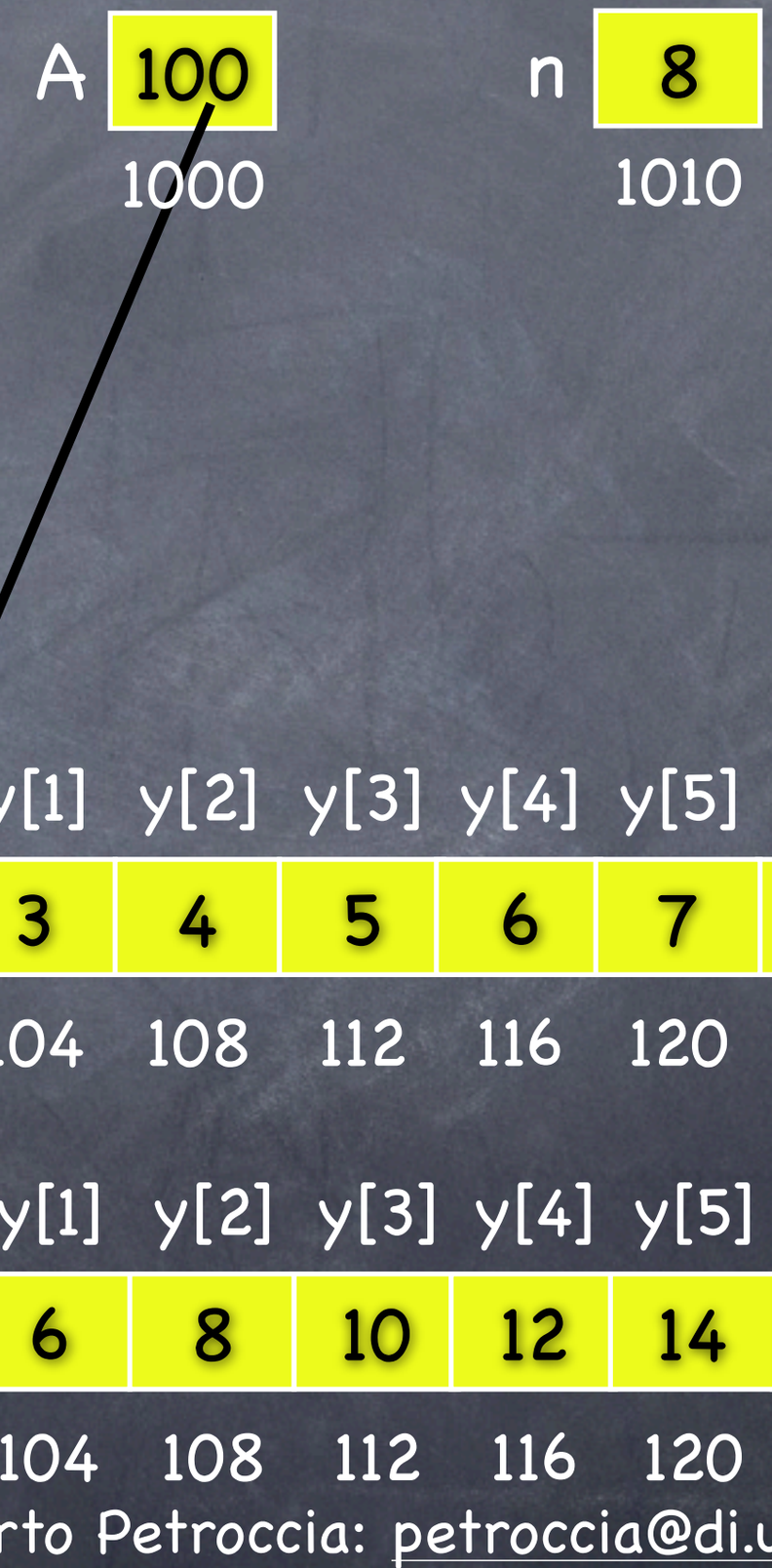
```
int main() {
    int i, n = 0;
    scanf("%d", &n);
    int y[n];
    for (i = 0; i < n; i++)
        scanf("%d", &y[i]);
    raddoppiaArray(y, n);
    stampaArray(y, n);
}
```



Cambiando A[i] cambiamo il contenuto della locazione di memoria a cui riferisce che è la stessa di y, quindi cambiamo anche y

Funzioni con array

```
void raddoppiaArray (int  A[], int n) {
      int i;
      for (i = 0; i < n; i++)
            A[i] *= 2;
}
```




```
int main() {
      int i, n = 0;
      scanf(“%d”, &n);
      int y[n];
      for (i = 0; i < n; i++)
            scanf(“%d”, &y[i]);
      raddoppiaArray(y, n);
      stampaArray(y, n);
}
```

DOMANDE ???

Suggerimento input

Se dovete dare in input ad un programma una sequenza di valori (ad esempio interi se il vostro programma esegue delle scanf("%d", &v)), invece di digitare ogni volta tali interi potete fare nel seguente modo:

1. Create un file che contiene gli interi da passare in input

2. Date il file in input al vostro programma (se l'eseguibile si chiama a.out ed il file con i dati di input dati.txt):

    ./a.out < dati.txt

    "<" questo simbolo reindirizza il contenuto del file al programma

Se i dati sono su una cartella diversa dovete scrivere il percorso per arrivare ai dati. Così facendo potete cambiare il file se vi serve invece di riscrivere ogni volta i valori di input

05/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Esercizi

1. Scrivere una funzione “swap” che presi in input due interi scambia i loro valori. Il cambiamento deve essere mantenuto all’uscita dalla funzione.

2. Scrivere una funzione che stampa gli elementi di un vettore se il loro indice è pari ed un’altra che stampa solo quelli con indice dispari. Assumiamo zero come elemento pari.

3. Scrivere una funzione che stampa gli elementi di un vettore se il loro valore è pari ed un’altra che stampa solo quelli con valore dispari. Assumiamo zero come elemento pari.

Soluzione Ex. 1

```
#include <stdio.h>
void swap (int* x, int* y) {
      int aux = *x;
      *x = *y;
      *y = aux;
   }
```

```
int main() {
        int a, b;
        scanf (“%d”, &a);
        scanf (“%d”, &b);
        printf(“a = %d - b = %d\n”, a, b);
        swap(&a, &b);
        printf(“a = %d - b = %d\n”, a, b);
        return 0;
 }
```

Soluzione Ex. 2

```
#include <stdio.h>
void stampaIndicePari (int A[], int n) {
      int i = 0;
      for (i = 0; i < n; i += 2) {
         printf(“%d ”, A[i]);
      }
      printf(“\n”);
}
```

Soluzione Ex. 2

```
#include <stdio.h>
void stampaIndicePari (int A[], int n) {
      int i = 0;
      for (i = 0; i < n; i += 2) {
         printf(“%d ”, A[i]);
      }
      printf(“\n”);
   }

void stampaIndiceDispari (int A[], int n) {
      int i = 0;
      for (i = 1; i < n; i += 2) {
         printf(“%d ”, A[i]);
      }
      printf(“\n”);
   }
```

Soluzione Ex. 3

```
#include <stdio.h>
void stampaValorePari (int A[], int n) {
      int i = 0;
      for (i = 0; i < n; i++) {
         if (A[i] % 2 == 0)
            printf("%d ", A[i]);
      }
      printf("\n");
}
```

Soluzione Ex. 3

```
#include <stdio.h>
void stampaValorePari (int A[], int n) {
      int i = 0;
      for (i = 0; i < n; i++) {
         if (A[i] % 2 == 0)
            printf(“%d ”, A[i]);
      }
      printf(“\n”);
 }

void stampaValoreDisapri (int A[], int n) {
   int i = 0;
   for (i = 0; i < n; i++) {
      if (A[i] % 2 != 0)
         printf(“%d ”, A[i]);
   }
   printf(“\n”);
}
```