Introduzione al C

Lez. 3
Passaggio dei parametri

Esercizio 1

```c
/*
Scrivere una funzione ricorsiva f che, dato un intero N, restituisca
la somma dei primi N interi dispari. Scrivere un semplice programma
per testare la funzione che prenda in input un intero x e stampi il
valore di f(x)
*/

#include <stdio.h>

int f(int n)
{
	if (n == 0) return 0;
	else return (n*2)-1 + ricorsiva(n-1);
}

int main()
{
	int x, somma;
	printf("Inserisci un numero intero:\n");
	scanf("%d", &x);
	somma = f(x);
	printf("La somma dei primi %d numeri dispari è %d\n", x, somma);
	return 0;
}
```

Esercizio 2

```c
/*
All'interno del main dichiarare due array a e b di 10 elementi
ciascuno. Stampare a video “contigui” (risp. “non contigui”) se
le celle di memoria di a e b sono contigue in memoria. Stampare
inoltre il nome dell'array che tra i due ha l'indirizzo più piccolo.
*/

#include <stdio.h>
#define N 10
int main()
{
	int a[N];
	int b[N];

	int contigui = 0;
	if (a+N == b || b+N == a) contigui = 1;

	if (contigui) printf("Gli array sono contigui\n");
	else printf("Gli array NON sono contigui\n");

	if (a<b) printf("L'array con l'indirizzo più piccolo è A\n");
	else printf("L'array con l'indirizzo più piccolo è B\n");

	return 0;
}
```

Esercizio 3

```
/*
Scrivere un programma che dichiara un array di 10 interi e
stampa gli indirizzi di ogni cella dell'array. Provare ad eseguirlo
due volte e notare che gli indirizzi cambiano.
*/

#include <stdio.h>
#define N 10

int main()
{
	int a[N];
	int i;
	for(i=0; i<N; i++)
	  printf("Indirizzo di A[%d] = %p\n", i, &a[i]);

	return 0;
}
```

Funzioni e passaggio di parametri

Tutti i parametri delle funzioni C sono sempre passati per valore.

- il loro valore viene copiato al momento del passaggio

- eventuali modifiche nel corpo della funzione non si ripercuotono sull'originale!

Funzioni e passaggio di parametri

Tutti i parametri delle funzioni C sono sempre passati per valore.

- il loro valore viene copiato al momento del passaggio

- eventuali modifiche nel corpo della funzione non si ripercuotono sull'originale!

Esempio

```
void foo(int a) {
   a++;
}

void main() {
   int x = 10;
   foo(x);
   printf(“%d”, x); // 10 o 11?
}
```

Funzioni e passaggio di parametri

Tutti i parametri delle funzioni C sono sempre passati per valore.

- il loro valore viene copiato al momento del passaggio

- eventuali modifiche nel corpo della funzione non si ripercuotono sull'originale!

Esempio

```
void foo(int a) {
   a++;
}

void main() {
   int x = 10;
   foo(x);
   printf("%d", x); // 10 o 11?
}
```

10: foo modifica una copia di x

Funzioni e passaggio di parametri

Come posso fare in modo che le modifiche siano visibili fuori dal corpo della funzione chiamata?

Si simula il passaggio per riferimento con i puntatori!

- Si passa un puntatore anziché il suo valore
- questo viene copiato nel parametro formale
- si può modificare il valore nella cella puntata

Esempio

```
void foo(int *a) {
   (*a)++;
}

void main() {
   int x = 10;
   foo(&x);
   printf(“%d”, x); // 10 o 11?
}
```

Funzioni e passaggio di parametri

Come posso fare in modo che le modifiche siano visibili fuori dal corpo della funzione chiamata?

Si simula il passaggio per riferimento con i puntatori!

- Si passa un puntatore anziché il suo valore
- questo viene copiato nel parametro formale
- si può modificare il valore nella cella puntata

Funzioni e passaggio di parametri

Come posso fare in modo che le modifiche siano visibili fuori dal corpo della funzione chiamata?

Si simula il passaggio per riferimento con i puntatori!

- Si passa un puntatore anziché il suo valore
- questo viene copiato nel parametro formale
- si può modificare il valore nella cella puntata

Esempio

```
void foo(int *a) {
   (*a)++;
}

void main() {
   int x = 10;
   foo(&x);
   printf("%d", x); // 10 o 11?
}
```

11: foo modifica il valore contenuto in x attraverso una copia di &x

Funzioni e passaggio di parametri

Funzione che scambia il contenuto di due variabili

```
void scambio(int a, int b) {
   int tmp;
   tmp = a;
   a = b;
   b = tmp;
}

void main() {
   int m, n;
   m = 1;
   n = 2;
   scambio(m, n);
   printf("m = %d n = %d\n");
}
```

Output?

1) m = 2 n = 1

2) m = 1 n = 2

Funzioni e passaggio di parametri

Funzione che scambia il contenuto di due variabili

```
void scambio(int a, int b) {
   int tmp;
   tmp = a;
   a = b;
   b = tmp;
}

void main() {
   int m, n;
   m = 1;
   n = 2;
   scambio(m, n);
   printf("m = %d n = %d\n");
}
```

Output?

1) ~~m = 2 n = 1~~

2) m = 1 n = 2

Funzioni e passaggio di parametri

Funzione che scambia il contenuto di due variabili

```
void scambio(int a, int b) {
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
}

void main() {
    int m, n;
    m = 1;
    n = 2;
    scambio(m, n);
    printf("m = %d n = %d\n");
}
```

Output?

1) ~~m = 2 n = 1~~

2) m = 1 n = 2

NON FUNZIONA! Lo scambio viene fatto sulle copie!

Funzioni e passaggio di parametri

Funzione che scambia il contenuto di due variabili

```
void scambio(int *a, int *b) {
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}

void main() {
    int m, n;
    m = 1;
    n = 2;
    scambio(&m, &n);
    printf("m = %d n = %d\n");
}
```

Output?

1) m = 2 n = 1

2) m = 1 n = 2

Funzioni e passaggio di parametri

Funzione che scambia il contenuto di due variabili

```
void scambio(int *a, int *b) {
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}

void main() {
    int m, n;
    m = 1;
    n = 2;
    scambio(&m, &n);
    printf("m = %d n = %d\n");
}
```

Output?

1) m = 2 n = 1

2) ~~m = 1 n = 2~~

Funzioni e passaggio di parametri

Funzione che scambia il contenuto di due variabili

```
void scambio(int *a, int *b) {
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}

void main() {
    int m, n;
    m = 1;
    n = 2;
    scambio(&m, &n);
    printf("m = %d n = %d\n");
}
```

Output?

1) m = 2 n = 1

2) ~~m = 1 n = 2~~

Perché ora è corretta?

Funzioni e passaggio di parametri: Array

Gli array sono sempre passati per riferimento!
Ciò che viene passato (e copiato) è il puntatore al primo elemento dell'array.

Funzioni e passaggio di parametri: Array

Gli array sono sempre passati per riferimento!
Ciò che viene passato (e copiato) è il puntatore al primo elemento dell'array.

```
void init(int a[], int len) { // passare lunghezza!

   int i;
   for(i=0; i<len; i++;)
      a[i] = 0;              // == *(a+i) = 0;

}

void main() {
   int a[10];
   init(a, 10);
   // qui a[i] = 0 per ogni i
}
```

Funzioni e passaggio di parametri: Array

Le due scritture

```
void init(int a[], int len) {
   int i;
   for(i=0; i<len; i++;)
      a[i] = 0;
}
```

e

```
void init(int *a, int len) {
   int i;
   for(i=0; i<len; i++;)
      a[i] = 0;
}
```

sono equivalenti. Si preferisce la prima per leggibilità.

Esercizi

1) Scrivere una funzione

```
void MinMax(int a[], int len, int *min, int *max)
```

che, dato un array `a` e la sua lunghezza `len`, scrive il valore del massimo e del minimo elemento di `a` nelle celle puntate da `min` e `max`.

Scrivere un programma che utilizzi `rand()` per la generazione di un array di 10 elementi interi casuali in `[0,100]` e provare le funzione stampando il valore del minimo e il suo indirizzo.

2) Scrivere una funzione

```
int* FindVal(int a[], int len, int val)
```

che, dato un array `a` e la sua lunghezza `len`, cerca il valore `val` all'interno di `a` e restituisce un puntatore ad un elemento di `a` che lo contiene, o la costante predefinita `NULL` se `val` non è contenuto in `a.`

Eseguire dei test con un array di elementi casuali, come nell'esercizio 1)