

 **Federica**
UNIVERSITÀ

 **Facoltà di Scienze
Matematiche
Fisiche Naturali**

Laboratorio di Algoritmi e Strutture Dati

Prof. Aniello Murano

Linguaggio C – Seconda Parte

Corso di Laurea Codice insegnamento Email docente Anno accademico	Informatica 13917 murano@na.infn.it 2007/2008	Lezione numero: 2 Parole chiave: Funzioni, Array, Puntatori, Preprocessore
--	---	--

next

 **e-Learning**
Università degli Studi di Napoli Federico II

 **Federica**
UNIVERSITÀ

27/09/2007

 **Facoltà di Scienze
Matematiche
Fisiche Naturali**


Indice della lezione

- **Funzioni**
- **Array**
- **Puntatori**
- **Preprocessore**


back

✕

next


Federica

27/09/2007


Facoltà di Scienze
Matematiche
Fisiche Naturali

Funzioni

Nel C è possibile scomporre problemi complessi in moduli più semplici sfruttabili singolarmente.


Le funzioni sono blocchi di programmi indipendenti da altri moduli, ciascuno destinato ad una precisa operazione.

Un programma nel C non è altro che una grossa funzione *main()* che ingloba nel suo interno altre funzioni.


La comunicazione tra i diversi moduli avviene mediante gli argomenti, i valori di ritorno e le variabili esterne.

L'uso delle funzioni consente di nascondere l'implementazione di una certa operazione e concentrarsi solo sul "*cosa fa*" e non sul "*come lo fa*".

back
✖
next


Federica

27/09/2007


Facoltà di Scienze
Matematiche
Fisiche Naturali

Funzioni (2)

Una funzione può:

- compiere un'azione: provoca il verificarsi di una certa azione
- effettuare un calcolo: il risultato del calcolo viene restituito dalla funzione stessa


Una funzione viene definita nel seguente modo:

- *tipo-ritornato nome_f (dichiarazione argomenti)*
- *{dichiarazioni ed istruzioni }*


Ogni funzione presenta un valore di ritorno:

- Può essere di qualsiasi tipo predefinito o definito dall'utente.
- Nel caso di funzione che compie un'azione, ovvero non deve ritornare nessun valore, si usa il tipo predefinito *void* come valore di ritorno.
- Una funzione può avere o meno una lista di argomenti

back
✖
next

 Federica
UNIVERSITÀ

27/09/2007

5  Facoltà di Scienze
Matematiche
Fisiche Naturali

Controllo dell'esecuzione


All'atto della chiamata di una funzione il controllo nell'esecuzione viene passato alla prima istruzione del corpo della funzione stessa.

Esistono due modi per restituire il controllo al programma chiamante:


- attraverso l'istruzione: `return espressione;`
- termine dell'esecuzione della funzione `}`

E' opportuno controllare sempre che la chiamata di una funzione ed il suo valore di ritorno siano consistenti.

back ✖ next

 Federica
UNIVERSITÀ

27/09/2007

6  Facoltà di Scienze
Matematiche
Fisiche Naturali

Lista degli argomenti

La lista argomenti è usata per passare dati ad una funzione chiamata. La lista argomenti può essere anche vuota.


Le variabili da passare devono essere specificate tra parentesi dopo il nome della funzione.

Nel corpo della funzione le variabili devono essere dichiarate con il loro tipo corrispondente.


Esempio:

```
int lower(int c)
{
    int k;
    k = (c >= 'A' && c <= 'Z') ? (c + 'a' - 'A') : (c + 'A' - 'a');
    return k;
}
```


back ✖ next

 Federica
UNIVERSITÀ


27/09/2007

7  Facoltà di Scienze
Matematiche
Fisiche Naturali


back ✖ next

 Federica
UNIVERSITÀ


27/09/2007

8  Facoltà di Scienze
Matematiche
Fisiche Naturali

back ✖ next

 **Federica**
UNIVERSITÀ

27/09/2007

 **Facoltà di Scienze
Matematiche
Fisiche Naturali**

9


Prototipi

Se i prototipi vengono inseriti prima della definizione di una funzione, il compilatore quando incontra la dichiarazione conosce il numero ed i tipi degli argomenti e può così controllarli.


La dichiarazione e la definizione di una funzione devono essere consistenti sia nel numero che nel tipo dei parametri.

Attenzione: se dichiarazione e definizione di funzione si trovano nello stesso file sorgente eventuali non corrispondenze nei tipi degli argomenti saranno rilevati dal compilatore, in caso contrario al più ci sarà un **warning**.

back ✖ next

 **Federica**
UNIVERSITÀ

27/09/2007

 **Facoltà di Scienze
Matematiche
Fisiche Naturali**

10

Array


Un array è una collezione di variabili dello stesso tipo che condividono un nome comune.

Un array viene dichiarato specificando il tipo, il nome dell'array e uno o più coppie di parentesi quadre contenenti le dimensioni.


Esempio:

- `int nomi[4];` /* 1 dimensione di 4 interi */
- `float valori[3][4]` /* 2 dimensioni di 12 float */
- `char caratteri[4][3][5][7]` /* 4 dimensioni 420 elementi */

back ✖ next


Federica

27/09/2007


Facoltà di Scienze
Matematiche
Fisiche Naturali

Rappresentazione di Array

Il C memorizza i valori degli elementi di un array in locazioni consecutive di memoria.


int stanze[6]
→

3	2	0	2	1	4
---	---	---	---	---	---


- *locazione base* + 0: stanze[0] "3"
- *locazione base* + 1: stanze[1] "2"
- *locazione base* + 2: stanze[2] "0"
- *locazione base* + 3: stanze[3] "2"
- *locazione base* + 4: stanze[4] "1"
- *locazione base* + 5: stanze[5] "4"

E' molto importante conoscere come sono stati memorizzati gli elementi poiché questo consente di capire come sia possibile puntare ad un preciso elemento.

back
✖
next


Federica

27/09/2007


Facoltà di Scienze
Matematiche
Fisiche Naturali

Dichiarazione


In C è obbligatorio specificare in modo esplicito la dimensione di un array in fase di dichiarazione.

In una definizione di funzione, come vedremo, non occorre specificare la dimensione del vettore passato come parametro; sarà il preprocessore a risolvere l'ambiguità all'atto della chiamata.

Esempio:

```
int somma(int numeri[],dimensioni)
.....
main()
int num[3];
{
.....
totale = somma(num,3);
}
```


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

13




**Facoltà di Scienze
Matematiche
Fisiche Naturali**

Array multidimensionali

- Gli array multidimensionali vengono dichiarati specificando il numero di elementi per ciascuna dimensione.
- Un array bidimensionale con 6 elementi per ciascuna dimensione viene dichiarato come: `int alfa[2][6];`
- Per referenziare un singolo elemento è necessario utilizzare due coppie di parentesi quadre: `alfa[1][2] = 1;`
- In pratica un array multidimensionale è una collezione di oggetti, ciascuno dei quali è un vettore.
- Esempio:


```
main()
{
    int tabelline[10][10]; int i, j;
    for (i = 0 ; i < 10 ; i++)
        for(j = 0 ; j < 10 ; j++)
            tabelline[i][j] = (i + 1) * (j + 1);
}
```


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

14



**Facoltà di Scienze
Matematiche
Fisiche Naturali**

Indirizzamento

Per accedere ad un array si usano gli indici.


E' compito del programmatore fare in modo di non andare oltre i limiti di dimensione dell'array in questione.

Gli array partono dall'indice 0 fino alla lunghezza dichiarata meno 1;

Referenziare un array al di fuori dei suoi limiti può portare a errori di indirizzamento (**memory fault**) oppure può "sporcare" altre variabili in memoria diventando così molto difficile da localizzare.

Per copiare elementi da un array all'altro bisogna copiare singolarmente ogni elemento.

back
✖
next



Federica

27/09/2007

15

Facoltà di Scienze
Matematiche
Fisiche Naturali

Array di stringhe

Una stringa è un array monodimensionale di caratteri ASCII terminati da un caratter *null* '\0'

Ad esempio "Questa è una stringa" è un array di 21 caratteri.

L'array è quindi il seguente:

elemento zero 'Q'

primo elemento 'u'

secondo elemento 'e'

.....

20^{mo} elemento 'a'

21^{mo} elemento '\0'

Esempio:

```
/* Stampa carat. e codifica ASCII */
char str[] = "Questa è una stringa";
main()
{
    int i = 0;
    while ( str[i] != '\0' )
    {
        printf("%c=%d\n", str[i], str[i] );
        ++i;
    }
}
```

back

✖

next



Federica

27/09/2007

16

Facoltà di Scienze
Matematiche
Fisiche Naturali

Array come argomenti di funzioni

Il metodo di default di passaggio delle variabili è per valore, quindi si potrà modificare solo una copia locale della variabile.

Eccezione a questa regola globale sono gli array.

Gli array, infatti, vengono sempre passati per **reference** ossia è il loro indirizzo che viene passato invece del loro contenuto.

Questo consente, solo nel caso dei vettori, di poter agire direttamente sulla variabile e non sulla copia. Ovvero le azioni che si effettuano sull'argomento della funzione avranno effetto anche al termine dell'esecuzione della funzione.

back

✖

next



Federica

UNIVERSITÀ

27/09/2007

17



Facoltà di Scienze
Matematiche
Fisiche Naturali

Puntatori

Un **puntatore** è una variabile che contiene l'indirizzo di un'altra variabile.


I puntatori sono "**type bound**" cioè ad ogni puntatore è associato il tipo a cui il puntatore si riferisce.

Nella dichiarazione di un puntatore bisogna specificare un asterisco (*) di fronte al nome della variabile pointer.

Esempio:

<code>int *pointer;</code>	puntatore a intero
<code>char *pun_car;</code>	puntatore a carattere
<code>float *flt_pnt;</code>	puntatore a float

back
✖
next




Federica

UNIVERSITÀ

27/09/2007

18



Facoltà di Scienze
Matematiche
Fisiche Naturali

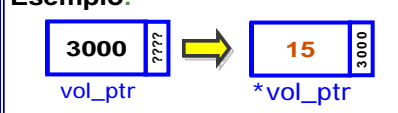
Inizializzazione di puntatori

Un pointer, prima del suo utilizzo, deve essere inizializzato, ovvero deve contenere l'indirizzo di un oggetto.

Per ottenere l'indirizzo di un oggetto si usa un operatore unario "&".

```
int volume, *vol_ptr;
vol_ptr = &volume;
```

Esempio:




Il puntatore `vol_ptr` contiene ora l'indirizzo della var. `volume`.

Per assegnare un valore all'oggetto puntato da `vol_ptr` occorre utilizzare l'operatore di indirizione " * ".

```
*vol_ptr= 15;
```


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

19



Facoltà di Scienze
Matematiche
 Fisiche Naturali


Esempi

```

/* Dichiarazioni */
int v1, v2, *v_ptr;
/* moltiplica v2 per il valore puntato da v_ptr */
v1 = v2* (*v_ptr);
/* somma v1,v2 e il contenuto di v_ptr */
v1 = v1+v2 +*v_ptr;
/* assegna a v2 il valore che si trova tre interi dopo v_ptr
*/
v2 = *(v_ptr + 3);
/* incrementa di uno l'oggetto puntato da v_ptr */
*v_ptr += 1;

```


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

20



Facoltà di Scienze
Matematiche
 Fisiche Naturali

Puntatori


Vantaggi nell'uso dei puntatori:

- Riduzione quantità di memoria statica usata dal sistema.
- Passaggio di parametri per "indirizzo" nelle chiamate a funzioni
 - In questo modo è possibile rendere globale ogni modifica sui parametri passati. Viceversa, se i parametri vengono passati per valore, ogni modifica della funzione sui parametri si perde all'uscita della funzione.

Aritmetica dei puntatori:

- Assegnamento tra puntatori dello stesso tipo,
 - **int *ptr1, *ptr2; *ptr1 = 1; ptr2 = ptr1;**
- Addizione e sottrazione tra puntatori ed interi,
 - **int *ptr, arr[10]; ptr = &arr[0];**
 - **ptr = ptr + 4 /* punta al quinto elemento dell'array arr[4] */**

back
✖
next



Federica

27/09/2007


21

Facoltà di Scienze
Matematiche
Fisiche Naturali

Puntatori e stringhe di caratteri

- Molto spesso vengono usati i puntatori a caratteri in luogo degli array di caratteri (stringhe), questo perché il C non fornisce il tipo predefinito stringa.
- Esiste una differenza sostanziale tra array di caratteri e puntatori a carattere. Ad esempio in:
 - `char *ptr_chr = "Salve mondo";`
 - il compilatore non crea una copia della stringa "Salve mondo"
 - Il compilatore crea un puntatore che punta ad una locazione di memoria in cui risiede il primo carattere della stringa costante.
 - Quindi e' possibile modificare il puntatore senza modificare il contenuto della stringa costante.
- Esempi di inizializzazione di array di caratteri:
 - `char caratteri[4] = { 'a' , 'A' , 'H' , 'k' };`
 - `char stringa_2 [] = "MMMM";`

back
✖
next



Federica

27/09/2007

22

Facoltà di Scienze
Matematiche
Fisiche Naturali

Puntatori come argomenti di funzioni

- Se l'argomento di una funzione è una variabile puntatore il passaggio della variabile avviene per **reference** (per indirizzo) ossia la funzione chiamata sarà in grado di modificare il valore globale della variabile che riceve.
- Passaggio argomenti per valore:
 - `int numero; square(numero);`
- Passaggio per indirizzo:
 - `square(&numero);`

```
void swap(int *x_ptr, int *y_ptr)
{
    int temp;
    temp = *x_ptr;
    *x_ptr = *y_ptr;
    *y_ptr = temp;
}
```

```
main()
{
    int a = 3;
    int b = 5;
    swap(&a,&b);
    printf("%d %d\n",a,b);
}
```

back
✖
next

Federica 27/09/2007 23 Facoltà di Scienze Matematiche Fisiche Naturali

Array di puntatori

- Un array di puntatori è un array i cui elementi sono dei puntatori a variabili: `int *arr_int[10]`
- `arr_int[0]` contiene l'indirizzo della locazione di memoria contenente un valore intero.
- Nel C i puntatori a caratteri vengono usati per rappresentare il tipo stringa che non risulta definito nel linguaggio, e gli array di puntatori per rappresentare stringhe di lunghezza variabile.
- Un insieme di stringhe potrebbe essere rappresentato come un array bidimensionale di caratteri, ma ciò comporta uno spreco di memoria.
- Ad esempio:
 - `char *term[100]` - Indica che gli elementi di `term` sono dei puntatori a carattere, cioè `term[i]` è l'indirizzo di un carattere.
 - `term[7] = "Ciao"` - Indica che il contenuto di `term[7]` è il puntatore alla stringa "Ciao";

back [X] next

Pochi char*
String

Federica 27/09/2007 24 Facoltà di Scienze Matematiche Fisiche Naturali


Esempio

```
#include <stdio.h>
main()
{
    char *giorni[7] = {"Lunedì", "Martedì", "Mercoledì", "Giovedì",
                      "Venerdì", "Sabato", "Domenica"};
    int i;
    for( i=0; i < 7; i++)
    {
        printf("\n %d ", *giorni[i]);
        printf("%s", giorni[i]);
    }
}
```

76 Lunedì || 77 Martedì ||

- **Domanda:** quale istruzione devo scrivere per stampare la "b" di Sabato?


back [X] next



Federica
UNIVERSITÀ

27/09/2007

25



**Facoltà di Scienze
Matematiche
Fisiche Naturali**

Preprocessore C


Il **preprocessore C** è una estensione al linguaggio che fornisce le seguenti possibilità:

- definizione delle costanti
- definizione di macro sostituzioni
- inclusione di file
- compilazione condizionale

I comandi del preprocessore iniziano con **#** nella prima colonna del file sorgente e non richiedono il ";" alla fine della linea.

Un compilatore C esegue la compilazione di un programma in due passi successivi. Nel primo passo ogni occorrenza testuale definita attraverso la direttiva **#** viene sostituita con il corrispondente testo da inserire (file, costanti, macro)


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

26



**Facoltà di Scienze
Matematiche
Fisiche Naturali**

Preprocessore C : Costanti

Attraverso la direttiva **#define** del preprocessore è possibile definire delle costanti:

Sintassi:


#define	nome	testo da sostituire :
#define	MAXLEN	100
#define	YES	1
#define	NO	0
#define	ERROR	"File non trovat\n"

E' uso comune usare lettere maiuscole per le costanti di **#define**

Perché usare queste costanti?

- favoriscono la leggibilità del programma
- consentono un facile riuso del codice


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

27



Facoltà di Scienze
Matematiche
 Fisiche Naturali

Preprocessore C : Macro


L'uso della direttiva **#define** consente anche di definire delle macro.

Una macro è una porzione di codice molto breve che è possibile rappresentare attraverso un nome; il preprocessore provvederà ad espandere il corrispondente codice in linea.

Una macro può accettare degli argomenti, nel senso che il testo da sostituire dipenderà dai parametri utilizzati all'atto della chiamata.

Il preprocessore espanderà il corrispondente codice in linea avendo cura di rimpiazzare ogni occorrenza del parametro formale con il corrispondente argomento reale.


back
✖
next



Federica
UNIVERSITÀ

27/09/2007

28



Facoltà di Scienze
Matematiche
 Fisiche Naturali

Macro : Esempi

```

#define square(x) ((x)*(x))
#define MIN(a,b) ( (a<b)? (a) : (b) )
#define ASSERT(expr) if(!(expr)) printf("error")

```

Nel file sorgente le linee :

```

square(2);
MIN(2,3);
ASSERT (a > b);

```

saranno sostituite in compilazione con

```

((2) * (2));
( (2 < 3) ? (2) : (3) );
if (!(a > b)) printf("error");

```

back
✖
next

Federica 27/09/2007 29 Facoltà di Scienze Matematiche Fisiche Naturali

Cosa non è una Macro

Anche se ciò può trarre in inganno, le macro NON sono funzioni.

Per esempio sugli argomenti delle macro non esiste controllo sui tipi.

Inoltre, una chiamata del tipo : MAX (i++, j++) verrà sostituita con:

((i++ > j++) ? (i++) : (j++));

E' importante stare attenti all'uso delle parentesi, ad esempio in:

#define square(x) x*x

Una chiamata del tipo: x = square(3+1); , genera:

x = 3 + 1 * 3 + 1; --> x = 7 ??????

back X next

Federica 27/09/2007 30 Facoltà di Scienze Matematiche Fisiche Naturali

Preprocessore C : Compilazione condizionale

Le direttive : #if #ifdef #ifndef #elif #else #endif consentono di associare la compilazione di alcune parti di codice alla valutazione di alcune costanti in fase di compilazione.

#if < espressione_costante>
< statement_1>
#else
<statement_2>
#endif

Se l'espressione costante specificata, valutata in compilazione ritorna TRUE allora verranno compilati gli statement_1 altrimenti verranno compilati gli statement_2

back X next

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.