Array, stringhe e puntatori

Laboratorio di Calcolo

Corso di Laurea in Fisica

Università degli Studi di Roma “La Sapienza”

Array

- Array = variabile multidimensionale
- Elementi dell'array accessibili attraverso l'uso di indici interi $\leftrightarrow$ vettori, matrici
- Array monodimensionali (vettore):

```
int a[3];       /* definisce un array
                   di interi con 3
                   componenti */
float x[25];    /* array di float con
                   25 componenti */
```

Convenzioni

- L’indice degli array in C inizia da 0

- `int a[3];`
  - definisce un array di 3 componenti individuate dagli indici 0, 1 e 2.
  - le componenti sono: a[0], a[1] e a[2]

Convenzioni

- L’indice degli array in C inizia da 0

- `int a[3];`
  - definisce un array di 3 componenti individuate dagli indici 0, 1 e 2.
  - le componenti sono: a[0], a[1] e a[2]

Array monodimensionali

```
int a[3];
```

a[0] a[1] a[2]

Array monodimensionali

```
a[0]=3;
```

Array monodimensionali

```
a[1]=0;
```

Array monodimensionali

```
a[2]=10;
```

Array: esempio

```
int main() {
   float misure[100], somma;
   int n, i;
   printf(“Inserisci numero misure:”);
   scanf(“%f”, &n);
   for (i=0; i<n; i++) {
      printf(“Misura n. %i:”, i+1);
      scanf(“%f”, &misura[i]);
   }
   somma = 0.;
   for (i=0; i<n; i++) {
      somma += misura[i];
   }
   printf(“La media vale: %f”, somma/n);
   return 0;
}
```

Array: esempio

```
int main() {
    float misure[100], somma;
    int n, i;
    printf("Inserisc
    scanf("%f", &n);
    for (i=0; i<n; i
        printf("Misur
        scanf("%f", &
    }
    somma = 0.;
    for (i=0; i<n; i
        somma += misu
    }
    printf("La media vale: %f", somma/n);
    return 0;
}
```

Esercizio:

1) fai in modo che l'utente non possa inserire più di 100 misure;
2) calcola e mostra, oltre alla media, la deviazione standard delle misure;

Stringhe

- Una stringa è un array di caratteri

- **`char string[80];`**
  - definisce una stringa di lunghezza massima di 79 caratteri
  - la stringa termina all'elemento precedente a quello che vale 0 (null terminating character)
  - **ATTENZIONE**: Il null terminating character NON è il carattere '0', ma il carattere che ha codice ASCII 0!!!

Esempio

```
char test[7] = “ciao”;

test[0] =  99 (c)
test[1] = 110 (i)
test[2] =  97 (a)
test[3] = 111 (o)
test[4] =   0 (nul)
test[5] =   x (?)
test[6] =   x (?)
```

Assegnazione delle stringhe

- ```
  char str[80] = “valore iniziale”;
  ```

- ```
  char str[80];
  scanf(“%s”, str);
  ```

- ```
  char str[80];
  scanf(“%[a-zA-Z0-9 ]”, str);
  ```

- ```
  char str[80];
  for (i=0; (c=getchar()) != ‘\n’; ++i) {
     str[i] = c;
  }
  ```

Funzioni utili

- **`stdio.h`**
  - **`c = getchar();`**

    restituisce un carattere letto da tastiera. Equivale a:

    `scanf("%c", &c);`

  - **`putchar(c);`**

    scrive un carattere sullo schermo. Equivale a:

    `printf("%c", c);`
- **`ctype.h`**
  - contiene funzioni utili per la manipolazione dei caratteri

Funzioni utili

- Supponiamo che **`c`** sia una variabile di tipo **`char`**:

- **`isalpha(c)`** restituisce TRUE se c è un carattere alfabetico
- **`isdigit(c)`** restituisce TRUE se c è un carattere numerico
- **`islower(c)`** restituisce TRUE se c è un carattere alfabetico minuscolo
- **`isupper(c)`** restituisce TRUE se c è un carattere alfabetico maiuscolo
- **`tolower(c)`** restituisce il corrispondente minuscolo di c
- **`toupper(c)`** restituisce il corrispondente maiuscolo di c

Puntatori

- Puntatore = indirizzo di una variabile in memoria
- Un variabile di tipo puntatore contiene l'indirizzo di memoria presso il quale possono essere memorizzati dei dati del tipo specificato

- Utile per lavorare con strutture complesse (array)

Variabili e Memoria

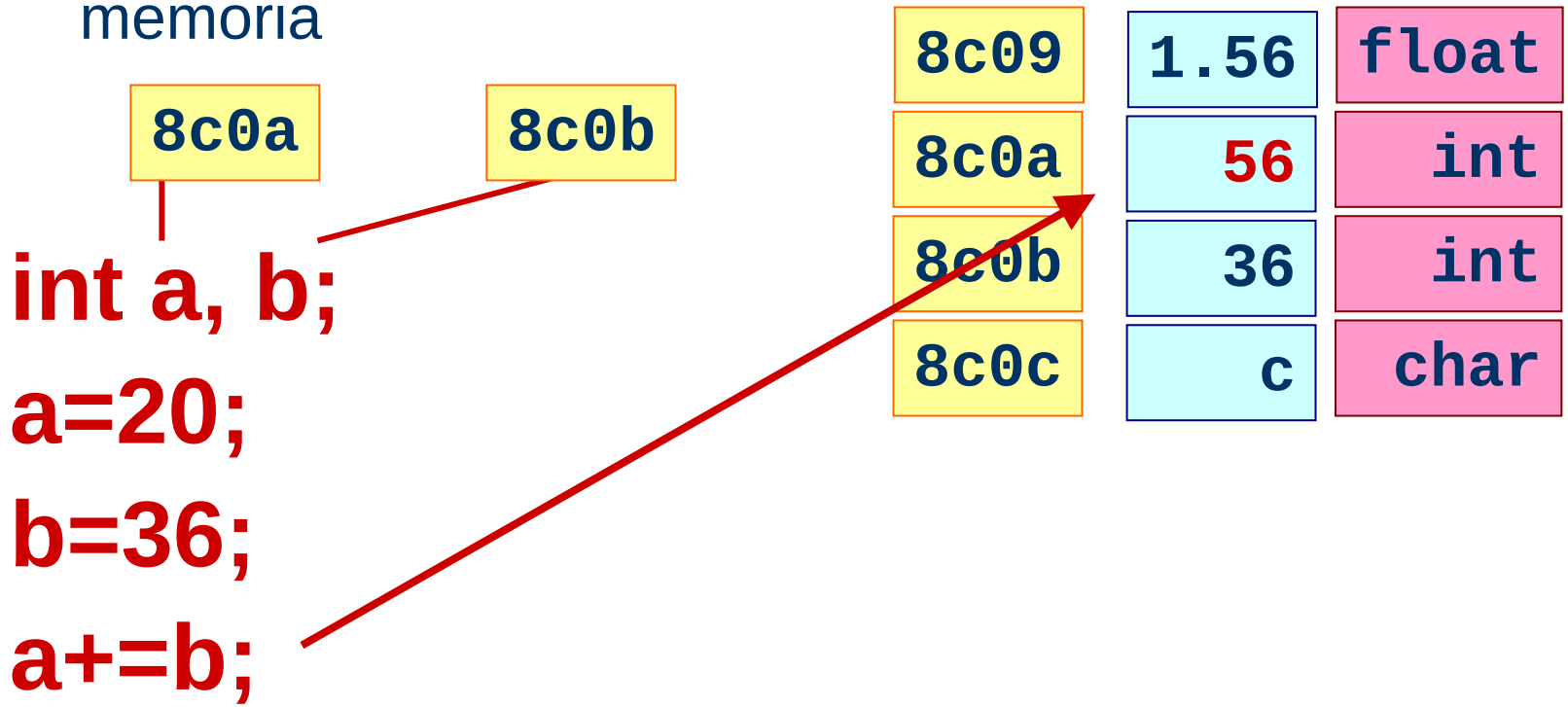
- In un programma una variabile è un simbolo per indicare un indirizzo di memoria


```
int a, b;
a=20;
b=36;
a+=b;
```

8c09	1.56	float
8c0a	20	int
8c0b	36	int
8c0c	c	char

Variabili e Memoria

- In un programma una variabile è un simbolo per indicare un indirizzo di memoria



Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (**`*`**)

- **`int *n;`**
- **`float *x;`**
- **`char *s;`**

- La dichiarazione si legge *a rovescio*

Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (**`*`**)

- `int *n;` `n`
- `float *x;`
- `char *s;`

- La dichiarazione si legge *a rovescio*

Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (*****)

- `int *n;` `n è un puntatore`
- `float *x;`
- `char *s;`

- La dichiarazione si legge *a rovescio*

Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (*****)

- `int *n;` **n è un puntatore ad int**
- `float *x;`
- `char *s;`

- La dichiarazione si legge *a rovescio*

Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (**`*`**)

- `int *n;` **`n è un puntatore ad int`**
- `float *x;`
- `char *s;`

- La dichiarazione si legge *raggruppando i termini*

Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (**`*`**)

- `int *n;` **`n è un puntatore ad int`**
- `float *x;` **`*x`**
- `char *s;`

- La dichiarazione si legge *raggruppando i termini*

Dichiarazione dei puntatori

- I puntatori si dichiarano facendo precedere il nome della variabile da un asterisco (**`*`**)

- **`int *n;`** **`n è un puntatore ad int`**
- **`float *x;`** **`*x è un float`**
- **`char *s;`**

- La dichiarazione si legge *raggruppando i termini*

Uso dei Puntatori

- Un puntatore è una variabile che contiene un indirizzo di memoria (un nuovo tipo…) a cui può essere assegnato un valore (che rappresenta un indirizzo!)

int *p = &a;

& è un operatore che restituisce l'indirizzo dell'operando

scanf("%f", &x);

L'istruzione scanf modifica il contenuto della locazione di memoria puntata da &x

Assegnazione e stampa

```
int *p;
p = &a;  /* p contiene l'indirizzo
            di a */
*p = 30; /* la variabile puntata da p
            vale 30 (a=30) */
printf("%x\n", p); /* stampa l'indirizzo
                      di a (contenuto di
                      p) in esadecimale
                   */
printf("%d\n", *p); /* stampa il valore
                       di a (puntato da
                       p) */
```

Perché i puntatori?

- L’uso dei puntatori è importante nel passaggio dei parametri delle funzioni
- In particolare i puntatori vengono utilizzati per manipolare stringhe (array di caratteri)
- Esistono anche altri usi dei puntatori che esulano dagli scopi di questo corso