CALCOLO NUMERICO con ELEMENTI DI PROGRAMMAZIONE (BATR) - (A.A. 2012-2013)

Prof. F. Pitolli

Vettori e Matrici, Input/Output, Utilità

Ing. Gabriele Colosimo, Ing. Andrea Nascetti

Area di Geodesia e Geomatica
Dipartimento di Ingegneria Civile Edile e Ambientale
Università di Roma "La Sapienza"

<gabriele.colosimo, andrea.nascetti>@uniroma1.it

Indice

Indice

I vettori

Un primo sguardo

- Insieme di variabili **dello stesso tipo** posizionate in celle di memoria contigue
- Per dichiarare e accedere i vettori si usano **[]**
- **int a[n]** dichiara *a* come un vettore di *n* numeri interi
- **a[0]** accede al primo elemento di *a*
- **a[n-1]** accede all'ultimo elemento di *a*

ATTENZIONE

- Gli elementi sono accessibili **singolarmente**
- Non esiste la possibilità di controllare gli estremi del vettore
- Evitare l'accesso agli elementi **a[i]**, i $\notin [0, n-1]$
  è compito del programmatore

Dichiarare i vettori e accedere i loro elementi

Dichiarazioni

- **int a[10];** dichiara un vettore di 10 interi
- **int a[10] = {3};** dichiara un vettore di 10 interi e inizializza il primo elemento a 3
  - a[0] = 3
  - a[1] = 0
  - ...
  - a[9] = 0
- **int a[] = {2, 3, 4, 50, 45};** dichiara un vettore di 5 interi inizializzati con i rispettivi valori
- **int a[5] = {2, 3, 4};** dichiara un vettore di 5 interi ed inizializza
  - a[0] = 2
  - a[1] = 3
  - a[2] = 4
  - a[3] = 0
  - a[4] = 0

Calcolo della media

```c
/*
 * Calcolo della media utilizzando i vettori
 * Dopo aver inserito 10 numeri da tastiera, il programma
 * ne calcola la media
 */

#include <stdio.h>

int main()
{
   int i=0, n = 10;
   double a[n], media = 0.0;

   /* Lettura dati di input */
   while(i < n)
   {
      printf("Inserire il numero (%2d): ", i);
      scanf("%lf", &a[i]);
      i++;
   } /* while */
```

Calcolo della media

```
21
22    /* Calcolo della media */
23    for(i=0; i < 10; i++)
24    {
25       printf("a[%d] = %10.4lf\n", i, a[i]);
26       media += a[i];
27    } /* for */
28
29    printf("La media dei %d elementi e': %9.5lg\n", n, media/n);
30
31    return 0;
32 } /* main */
```

Vettori e funzioni

"Passaggio" alla funzione

- Per passare un vettore come argomento si utilizza il nome e si omettono le **[]**
- Bisogna passare anche **le dimensioni** del vettore
- Tecnicamente, è un **passaggio per riferimento**
- La funzione lavora con **l'indirizzo** del vettore

Funzione per il calcolo della media I

```
1  /*
2   * Calcolo della media utilizzando i vettori
3   * Dopo aver inserito n numeri da tastiera, il programma
4   * ne calcola la media
5   */
6  #include <stdio.h>
7  double CALCOLO_MEDIA(int num, double a[num]); /* PROTOTIPO */
8
9  int main()
10 {
11    int i=0, n=0;
12
13    printf("Inserire la dimensione del vettore: ");
14    scanf("%d", &n);
15
16    double a[n];
17
18    /* Lettura dati di input */
19    while(i < n)
```

Funzione per il calcolo della media II

```
20    {
21       printf("Inserire il numero (%2d): ", i);
22       scanf("%lf", &a[i]);
23       i++;
24    } /* while */
25
26    /* Calcolo della media */
27    printf("La media dei %d elementi e': %9.5lg\n", n,
          CALCOLO_MEDIA(n, a));
28
29    return 0;
30 } /* main */
31
32 /* DICHIARAZIONE DELLA FUNZIONE */
33 double CALCOLO_MEDIA(int num, double a[num])
34 {
35    int i;
36    double media=0.0;
37
38    for(i=0; i < num; i++)
```

Funzione per il calcolo della media III

```
39        media += a[i];
40
41    return media/num;
42 }
```

Indice

Le Matrici

In realtà, vettori di vettori

- Il C non prevede direttamente il concetto di matrice
- Però, i vettori **possono essere multidimensionali**
- **int a[n][m];** dichiara un vettore di interi a due dimensioni (matrice) con *n* **righe** e *m* **colonne**
- **int a[4][4];** dichiara un vettore di 4 interi, **ognuno vettore di 4 interi**

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]
a[3][0]	a[3][1]	a[3][2]	a[3][3]

Inizializzazione

- **int a[2][2] = {{1, 2}, {3, 4}};**

Funzione per la stampa di una matrice I

```
1  /*
2   * Funzione per la stampa di una matrice
3   */
4
5  #include <stdio.h>
6
7  void STAMPA_MATRICE(int righe, int colonne, double A[righe][
       colonne]);
8  double MATR_TRACCIA(int righe, double A[righe][righe]);
9
10 int main()
11 {
12   int i, j, righe, colonne;
13
14   printf("Inserisci righe: ");
15   scanf("%d", &righe);
16   printf("Inserisci colonne: ");
17   scanf("%d", &colonne);
18
```

Funzione per la stampa di una matrice II

```
19    double A[righe][colonne]; /* allocazione matrice */
20
21    /* Inizializzazione della matrice */
22    for(i = 0; i < righe; i++)
23    {
24      for(j = 0; j < colonne; j++)
25      {
26        A[i][j] = 1. + i + j; /* Valore iniziale */
27      }
28    }
29
30    /* STAMPA MATRICE */
31    STAMPA_MATRICE(righe, colonne, A);
32
33    /* CALCOLO DELLA TRACCIA DELLA MATRICE */
34    double traccia = 0.;
35    traccia = MATR_TRACCIA(righe, A);
36    printf("La traccia della matrice A e' %8.3lf\n", traccia);
37
38    return 0;
```

Funzione per la stampa di una matrice III

```
39 } /* main */
40
41 /* DICHIARAZIONE DELLA FUNZIONE */
42
43 void STAMPA_MATRICE(int righe, int colonne, double A[righe][
       colonne])
44 {
45   int i, j;
46
47   for(i=0; i < righe; i++)
48   {
49     for(j = 0; j < colonne; j++)
50     {
51       printf("% 8.3lf ", A[i][j]);
52     }
53     printf("\n");
54   }
55 }
56
57 double MATR_TRACCIA(int righe, double A[righe][righe])
```

Funzione per la stampa di una matrice IV

```
58 {
59   int i;
60   double traccia = 0.;
61
62   for(i = 0; i < righe; i++)
63     traccia += A[i][i];
64
65   return traccia;
66 }
```

Indice

Come immagazzinare dati e presentare i risultati

I flussi (stream)

- **Input** e **Output** (I/O) sono flussi di comunicazione (in **ingresso** e in **uscita**) del programma
- Tutto l'I/O è eseguito attraverso i **flussi** (**stream**): sequenze di caratteri organizzate in righe
- Ogni riga consiste di zero o più caratteri e **termina con un *newline*** ($\backslash n$)
- Ogni programma, quando avviato, è automaticamente connesso a 3 flussi:
  1. standard input (**stdin**), connesso alla tastiera
  2. standard error (**stderr**), connesso allo schermo
  3. standard output (**stdout**), connesso allo schermo
- Il programmatore può **ridirigere** questi flussi su altri dispositivi (stampa su file, . . .)

Formattare l'output

I flussi (stream)

- **printf(stringa_controllo_formato, . . .)** (print format) dirige la stringa di controllo verso lo **stdout**
- La stringa di controllo del formato è composta di
  1. indicatori di conversione (**d, f, ld, lf, s, u, . . .**)
  2. flag (**0, #, +, -**)
  3. dimensioni di campo
  4. precisioni
  5. caratteri letterali
- Uniti al segno di percentuale (%) questi formano la **specifica di conversione**

Visualizzare i numeri con printf()

Indicatore di conversione	Descrizione
	Numeri interi
d	Visualizza un intero decimale con segno (**int**)
u	Visualizza un intero decimale senza segno (**unsigned int**)
h o *l*	Posto davanti *d* o *u* **indica il tipo short o long**
	Numeri a virgola mobile
e o *E*	Visualizza un valore in virgola mobile **nella notazione esponenziale**
f	Visualizza (**valore in virgola mobile**)
g o *G*	Sceglie una **rappresentazione** tra *f* e *e*
L	Posto davanti a *e* o *f* indica il tipo **long double**

Dimensioni di campo

printf()

- La **dimensione di campo** definisce la misura del campo di visualizzazione dei dati
- L'**intero** che stabilisce la dimensione del campo è inserito tra % e l'indicatore di conversione (es. **printf("%4d\n", 123);**)

```
1 #include <stdio.h>
2
3 int main()
4 {
5 printf("%d\n", 123);
6 printf("%6d\n", 123);
7 printf("%6d\n", 1234569);
```

```
123
   123
1234569
```

Precisioni

printf()

- E' possibile anche specificare la **precisione** con cui il dato deve essere visualizzato
  - per gli interi, indica il **numero minimo** di cifre da visualizzare
  - per i valori in virgola mobile, indica il numero di cifre da visualizzare **dopo la virgola**

```
1 printf("%6.6d\n", 123);
2 printf("%6.6f\n", 2.3);
3 printf("%6.6e\n", 2.3);
```

```
000123
2.300000
2.300000e+00
```

Formattare l'input I

Leggere i dati del problema

- Con **scanf()** è possibile formattare precisamente l'input
- **scanf(stringa_controllo, altri_argomenti)**
- La stringa di controllo **descrive il formato dei dati da prendere in input**
- Gli altri argomenti associano i valori letti alle variabili
- Il carattere speciale **&** effettua **l'associazione tra la variabile e il valore letto**

Regole di formattazione

- Valgono le regole per indicatori di conversione, precisioni, e dimensioni di campo **già definite per printf()**

Formattare l'input II

```c
#include <stdio.h>

int main()
{
   int a = 0, b = 0;

   printf("Inserire due numeri interi: ");
   scanf("%d%d", &a, &b);
   printf("L'utente ha inserito i numeri %d e %d\n", a, b);

   return 0;
}
```

I files I

FILE *: un tipo speciale di dato (struttura)

- **FILE *stream_name;** dichiara una struttura di tipo **FILE** e la chiama *stream_name*
- Tutte le funzioni che operano sui file sono definite in **stdio.h**
- Per aprire i file si usa la funzione **FILE * = fopen(**<*nome_file*>**, "modo_apertura");**
  - "r" per aprire un file in modalità lettura
  - "w" per creare un **nuovo file** in modalità scrittura
  - "a" per creare un nuovo file o per **appendere il testo in coda** ad un file esistente
- I file si chiudono con la funzione **fclose(FILE *stream_name)**

I files II

Scrivere su un file

- **fprintf(FILE *stream_name, formato, ...);**
- Funzione che permette di **ridirigere** il testo di output su uno stream
- Esempio: **fprintf(OUT, "Hello World\n");** scrive il testo *Hello World* sullo stream OUT

```
1  #include <stdio.h>
2  int main()
3  {
4     FILE *OUT;
5
6     OUT = fopen("Il_mio_file", "w");
7     fprintf(OUT, "Hello World\n");
8     fclose(OUT);
9
10    return 0;
11 } /* main */
```

I files III

```
user@CALCOLO_NUMERICO: ls
file_1.c
user@CALCOLO_NUMERICO: gcc -o file_1.out file_1.c
user@CALCOLO_NUMERICO: ls
file_1.c  file_1.out
user@CALCOLO_NUMERICO: ./file_1.out
user@CALCOLO_NUMERICO: ls
file_1.c  file_1.out  ll_mio_file
user@CALCOLO_NUMERICO: cat ll_mio_file
Hello World
user@CALCOLO_NUMERICO:
```

I files IV

Leggere da un file

- **fscanf(FILE *stream, formato, ...);**
- Funzione che permette di leggere l'input da uno stream secondo il formato specificato e associarlo alle eventuali variabili
- Esempio: **fscanf(IN, "%d%d", &a, &b);** legge due interi dallo stream *IN* e li associa alle variabili *a* e *b*

```
1  #include <stdio.h>
2
3  int main()
4  {
5     int a = 0, b = 0;
6     FILE *IN;
7
8     IN = fopen("input.dat", "r");
9
10    fscanf(IN, "%d%d", &a, &b);
```

I files V

```
11    printf("IL file \"input.dat\" contiene i numeri %d e %d\n", a,
          b);
12
13    fclose(IN);
14
15    return 0;
16 }
```

```
user@CALCOLO_NUMERICO: cat input.dat
54 87
user@CALCOLO_NUMERICO: ./fscanf_1
IL file "input.dat" contiene i numeri 54 e 87
user@CALCOLO_NUMERICO:
```

Indice

Principali opzioni di compilazione I

Assegnare il nome al file binario compilato

- L'opzione **-o <nome_eseguibile>** permette di dare il nome **voluto dall'utente** al file binario che stiamo compilando

```
gcc -o hello.out hello.c
```

- Se omessa, il file binario si chiamerà di default **a.out**

Collegare librerie esterne

- L'opzione **-l<nome_libreria>** permette di collegare librerie al programma che si sta compilando
- La libreria matematica si include nell'header con il comando *#include<math.h>* e si collega con l'opzione **-lm**

```
gcc -o file_matematico.out file_matematico.c -lm
```

Principali opzioni di compilazione II

Seguire i suggerimenti del compilatore

- Il compilatore ci aiuta controllando il codice sorgente e **segnalando errori e avvisi (warning)**
- L'opzione **-Wall** attiva tutti gli avvisi di warning

```
1 #include <stdio.h>
2 int main()
3 {
4    int a = 4;
5    printf("%f\n", a);
6    return 0;
7 }
```

```
user@CALCOLO_NUMERICO: gcc -Wall -o gcc_1.out gcc_1.c
gcc_1.c: In function main :
gcc_1.c:5:2: warning: format % f expects argument of type
        double , but argument 2 has type int [-Wformat]
user@CALCOLO_NUMERICO:
```

Principali opzioni di compilazione III

Attivare il debug

- L'opzione **-g** inserisce nel file eseguibile informazioni utili per eseguire il **debug** con **GDB** (GNU Project Debugger)

```
user@CALCOLO_NUMERICO: gcc -g -o hello.out hello.c
user@CALCOLO_NUMERICO:
```

Indice

Riferimenti Bibliografici

Daniele Giacomini, *Appunti di informatica libera*
Introduzione al linguaggio C

Deitel & Deitel,
Corso completo di programmazione, Apogeo