UNIVERSITY OF TRENTO - Italy

Laboratorio 11

Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2024-2025
Università degli Studi di Trento
pierluigi.roberti@unitn.it

Funzioni

Input
Parametri: 0-N

Output
Parametri: 0-1

Se 0:
output di tipo void

Funzioni nel linguaggio C

```
#include <stdio.h>
#include <stdlib.h>

typedef enum {FALSE, TRUE} Boolean;

Boolean numero_primo(int numero);

int main(void)
{
  int i;
  ...
  if(numero_primo(i)==TRUE){
    printf("%d è primo!\n",i);}
  else{
    printf("%d non è primo!\n" ,i);}
  ...

  return 0;
}

Boolean numero_primo(int numero){
  \* Se argomento è primo return TRUE *\
  \* Se argomento non è primo return FALSE *\
}
```

dichiarazione (prototipo) della funzione

invocazione della funzione

definizione della funzione

Funzioni: passaggio array (vettori)

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
void stampa(const int vet[], int dim);

int main(void)
{
  int elenco[N];
   ...
  stampa(elenco,N);  //stampo contenuto array

  return 0;
}

void stampa(const int vet[], int dim)
  int i;
  for(i=0; i<dim; i++){
    printf("%d",vet[i]);
  }
}
```

dichiarazione (prototipo) della funzione

invocazione della funzione

definizione della funzione

Funzioni: passaggio array (matrici)

```
#include <stdio.h>
#include <stdlib.h>
#define NR 10
#define NC 5
void stampa(int mat[NR][NC], int dimR, int dimC);

int main(void)
{
  int elenco[NR][NC];
   ...
  stampa(elenco,NR,NC);  //stampo contenuto array multidimensionale
  return 0;
}
void stampa(int mat[NR][NC], int dimR, int dimC)
    int i,j;
    for(i=0; i<dimR; i++){
        for(j=0; j<dimC; j++){
            printf("%d",mat[i][j]);
        }
        printf("\n");
    }
}
```

dichiarazione (prototipo) della funzione

invocazione della funzione

definizione della funzione

Note sintassi

Modalità passaggio array monodimensionali

È sempre per riferimento, cioè la modifica nella funzione comporta la modifica della variabile passata come parametro attuale!

OK `void f1(int v[], int dim);`

OK `void f1(int v[MAX], int dim);`

Va passata SEMPRE la dimensione dell'array

Modalità passaggio array bidimensionali

È sempre per riferimento, cioè la modifica nella funzione comporta la modifica della variabile passata come parametro attuale!

OK `void f2(int v[][MAXC], int nr, int nc);`

OK `void f2(int v[MAXR][MAXC], int nr, int nc);`

NO `void f2(int v[][], int nr, int nc);`

Va passata SEMPRE le 2 dimensioni dell'array

Note sintassi

Modalità passaggio array monodimensionali
in sola lettura, cioè non è possibile modificare il contenuto dell'array nella funzione

OK `void f1(const int v[], int dim);`

OK `void f1(const int v[MAX], int dim);`

Putroppo non è possibile «passare» ad una funzione un array bidimensionali in sola lettura!!!
Va prestata massima attenzione a non effettuare modifiche all'array nella funzione!

NO `void f2(const int v[MAXR][MAXC], int nr, int nc);`

Esercizio 0

- Definire una variabile vet array di N (100) interi
- Chiedere all'utente quanti elementi sono da considerare (dim)
- Inizializzare i primi dim elementi della variabile vet usando con valori casuali tra 1 e 10
- Stampare la variabile vet
  (solo i primi dim elementi: quelli inizializzati)

Soluzione senza funzioni

```
#define N 100
#define VAL_MIN 1
#define VAL_MAX 10
int main(){
  int vet [N];
  int i, dim;
  printf("dim:"); scanf("%d", &dim);
  // inizializza array
  for(i=0; i<dim; i++){
      vet[i] = rand()%(VAL_MAX - VAL_MIN + 1) + VAL_MIN;
  }
  // stampa array
  for(i=0; i<dim; i++){
      printf("%d ", vet[i]);
  }
  printf("\n");
  return 0;
}
```

Esercizio 1

- Definire una variabile vet array di N (100) interi
- Chiedere all'utente quanti elementi sono da considerare (dim)
- Inizializzare i primi dim elementi della variabile vet usando con valori casuali tra 1 e 10
  - Definire ed usare funzione ***init***
  - Definire ed usare funzione ***casuale***
- Stampare la variabile vet
  (solo i primi dim elementi: quelli inizializzati)
  - Definire ed usare funzione ***stampa***

Esercizio 1

- Funzioni

```
/* Genera numeri casuali tra i valori min e max */
int casuale(int min, int max);
/* Inizializza dim elementi di vett */
void init(int v[N], int dim);
/* Stampa array (passato in sola lettura) */
void stampa(const int v[N], int dim);
```

const

Parametro array non sarà modificato dalla funzione

NOTA

Quando si passa come parametro ad una funzione un array sono necessarie 2 informazioni:

- Dove inizia l'array in memoria
  (int vet[N] o int vet[] o int* vet)
- Di quanti elementi è composto l'array
  (int dim)

Esercizio 1

- Main

```
int vet[N];
int dim;
//manca il controllo sul valore di dim >= 2
printf("dim="); scanf("%d", &dim);
init(vet, dim);
stampa(vet, dim);
```

NOTA

Quando si passa come parametro ad una funzione un array sono necessarie 2 informazioni:

- Dove inizia l'array in memoria (int vet[N] o int vet[]
- Di quanti elementi è composto l'array (int dim)

Esercizio 2

Estendere esercizio 1

Stampare tutti i numeri primi presenti in vet

Definire ed usare funzione ***numero_primo***

```
/* Verifica se l'argomento numero e' primo */
Boolean numero_primo(int numero);
```

Boolean
Tipo definito come elenco enumerativo

Definire ed usare funzione ***stampa_primi***

```
/* Stampa i numeri primi presenti nel vettore */
void stampa_primi(const int v[N], int dim);
```

const
Parametro array non sarà modificato dalla funzione

Esercizio 2

- Main

```
int vet[N];
int dim;
printf("quanti? "); scanf("%d", & dim);
init(vet, dim);
stampa(vet, dim);
stampa_primi(vet, dim);
```

- Typedef

```
typedef enum {FALSE, TRUE} Boolean;
```

FALSE vale 0 => considerato **false** in una condizione

TRUE vale 1 => considerato **true** in una condizione

Esercizio 2 – ulteriori funzioni

- Estendere esercizio 1

- Cercare l'indice del massimo valore in vet
  - Definire ed usare funzione ***cerca_imax***
- Stampare i numeri primi in vet che sono divisori di altri elementi nell'array
  - Definire ed usare funzione ***stampa_divisori***
  - Definire ed usare funzione ***divisore***
- Cercare il minimo valore in vet
  - Definire ed usare funzione ***cerca_minimo***
  - Definire ed usare funzione ***minimo***

Esercizio 2 – ulteriori funzioni

- Funzioni

```
/* Cerca valore massimo – restituisce indice array */
int cerca_imax(const int v[N], int dim);

/* Verifica se d divide n  → n/d non ha resto */
Boolean divisore(int d, int n);

/* Stampa i numeri che sono divisori */
void stampa_divisori(const int v[N], int dim);

/* Restituisce il minimo tra i due argomenti */
int minimo(int primo, int secondo);

/* Cerca il minimo tra gli elementi dell'array */
int cerca_minimo(const int v[N], int dim);
```

const: il parametro non sarà modificato dalla funzione

Esercizio 2 – ulteriori funzioni

- Main

```
int vet[N];
int dim;
printf("quanti? "); scanf("%d", & dim);
init(vet, dim);
stampa(vet, dim);
stampa_primi(vet, dim);
printf("massimo; %d", vet[ cerca_imax(vet, dim) ]);
stampa_divisori(vet, dim);
printf("minimo; %d", cerca_minimo(vet, dim));
```

- Typedef

```
typedef enum{FALSE, TRUE} Boolean;
```

Esercizio 3/1

Esercizio visto in Lab10

- Definire una struct Tpunto

```
typedef struct Tpunto{ float x, y; } Tpunto;
```

- Definire variabile p di tipo Tpunto
- Inizializzare p
  - Funzione ***init_punto***
  - Valori x e y passati come parametri con valori casuali
    - x compreso tra -5.00 e 5.00 (estremi inclusi)
    - y compreso tra -5.00 e 5.00 (estremi inclusi)
- Stampare il contenuto di p
  - Funzione ***stampa_punto***

```
Tpunto init_punto(float x, float y);
void stampa_punto(Tpunto p);
int casuale(int min, int max);
```

Esercizio 3/1

```
typedef struct Tpunto{float x, y;} Tpunto;

/* Inizializza singolo punto */
Tpunto init_punto(float x, float y);

/* Stampa singolo punto */
void stampa_punto(Tpunto p);

/* Genera valori casuali tra i valori min e max */
int casuale(int min, int max);
```

Esercizio 3/1

```
/* Inizializza singolo punto */
Tpunto init_punto(float x, float y){
  Tpunto daRit;
  daRit.x = x;
  daRit.y = y;
  return daRit;
}
/* Stampa singolo punto */
void stampa_punto(Tpunto p){
  printf("[%.3f %.3f]", p.x, p.y);
}
/* Genera valori casuali tra i valori min e max */
int casuale(int min, int max){
  return rand()%(max-min+1)+min;
}
```

Esercizio 3/1

```
int main(){
  Tpunto p;

  p=init_punto( casuale(-500, 500)/100.0,
                casuale(-500, 500)/100,0);
  stampa_punto(p);

  return 0;
}
```

Esercizio 3 /2

- Definire un array punti di MAX=100 elementi di tipo Tpunto
- Chiedere all'utente il numero di punti da considerare
- Popolare l'array con valori casuali di x e y compresi tra -5.00 e 5.00
  - Funzione ***init***
- Stampare l'array
  - Funzione ***stampa***
- Cercare il punto più vicino e più lontano dall'origine e stamparne a video le coordinate
  - Funzione ***minimo*** e ***massimo***

Usare ANCHE le funzioni definite nell'esercizio precedente

Esercizio 3 /2

```
typedef struct Tpunto{float x, y;} Tpunto;

/* Inizializza singolo punto */
Tpunto init_punto(float x, float y);
/* Stampa singolo punto */
void stampa_punto(Tpunto p);
/* Genera numeri casuali tra i valori min e max */
int casuale(int min, int max);

/* Inizializzazione array */
void init(Tpunto vett[], int dim);
/* Stampa array */
void stampa(const Tpunto vett[], int dim);
/* Restituisce indice di elemento minimo */
int minimo(const Tpunto vett[], int dim);
/* Restituisce indice di elemento massimo */
int massimo(const Tpunto vett[], int dim);
```

Esercizio 3 /2

```
int main(){
  Tpunto punti[MAX];
  int imin, imax, dim;
  printf("dimensione: "); scanf("%d", &dim);

  init(punti, dim);
  stampa(punti, dim);

  imin = minimo(punti, dim);
  imax = massimo(punti, dim);

  printf("\nminimo: ");
  stampa_punto(punti[imin]);

  printf("\nmassimo: ");
  stampa_punto(punti[imax]);

  return 0;
}
```

Esercizio 4

- Scrivere un programma in grado di eseguire operazioni di somma e moltiplicazione tra matrici di interi. Date 2 matrici quadrate A e B (n x n) il programma deve stampare il risultato di A+B e A*B.
  - Le matrici A e B hanno dimensioni massime MAX x MAX con valore MAX definito tramite direttiva define.
  - La dimensione effettiva delle matrici da utilizzare (n) e' inserita dall'utente
  - Le matrici A e B sono inizializzate con numeri casuali.

Esercizio 4 – struttura main

- Struttura della funzione main():
  - dichiarazione variabili e matrici
  - richiesta all'utente della dimensione delle matrici (n<MAX)
  - (**) inizializzazione matrice A
  - (**) inizializzazione matrice B
  - (**) stampa matrice A
  - (**) stampa matrice B
  - (**) calcola e stampa somma: A + B
  - (**) calcola e stampa prodotto: A * B

NB: (**) operazioni da eseguire tramite l'utilizzo di una funzione

Esercizio 4 – funzioni

```
void init(int m[MAX][MAX], int nr, int nc);

void stampa(int m[MAX][MAX], int nr, int nc);

void stampa_somma(int m1[MAX][MAX],
                  int m2[MAX][MAX], int nr, int nc);

void stampa_prod(int m1[MAX][MAX],
                 int m2[MAX][MAX], int nr, int nc);

int casuale(int min, int max);
```

Esercizio 4 – main

```
int main(){
  int ma[MAX][MAX];
  int mb[MAX][MAX];
  int nr; int nc;
  nr=nc=4;
  init(ma, nr, nc);
  init(mb, nr, nc);
  stampa(ma, nr, nc);
  stampa(mb, nr, nc);
  stampa_somma(ma, mb, nr, nc);
  stampa_prodotto(ma, mb, nr, nc);

  return 0;
}
```

Esercizio 5

- Definire un programma che
  - crea un array di MASSIMO 200 elementi (**char**)
  - dimensione effettiva chiesta all'utente
  - popola automaticamente l'array con valori compresi fra 'a' e 'z' in modo casuale
  - stampa a video
    - l'array
    - l'elemento di valore minimo nell'array
    - gli elementi che sono vocali
    - l'elemento più ricorrente
- Utilizzare le funzioni

Esercizio 5

```
typedef enum{FALSE, TRUE} Boolean;

/* Genera valori casuali tra i valori min e max */
int casuale(int min, int max);
/* Inizializzazione array */
void init(char vett[], int dim);
/* Stampa array */
void stampa(const char vett[], int dim);
/* Verifica se l'argomento numero e' una vocale. */
Boolean vocale(char c);
/* Stampa le vocali presenti in array */
void stampa_vocali(const char vett[], int dim);
/* Restituisce il minimo tra i due argomenti */
int minimo(int primo, int secondo);
/* Restituisce il minimo carattere */
char cerca_minimo(const char vett[], int dim);
/* Restituisce il carattere con massima frequenza */
char max_occorrenza(const char vett[], int dim, int vmin, int
vmax);
```

Esercizio 5

```
int main(){
  char array[LUNGHEZZA_ARRAY]; int dim;
  printf("dimensione: "); scanf("%d", &dim);

  init(array, dim);
  stampa(array, dim);

  printf("\nminimo: %c\n", cerca_minimo(array, dim));

  stampa_vocali(array, dim);
  printf("Carattere piu' frequente: %c",
                    max_occorrenza(array, dim, VAL_MIN, VAL_MAX));

  return 0;
}
```

Esercizio 6

- Estensione esercizio 3
- Definire una struct Tpunto

```
typedef struct Tpunto{
    float x, y; char nome[MAX];
}Tpunto;
```

- Definire variabile p di tipo Tpunto
- Inizializzare la variabile p
  - Creare una funzione init_punto
  - Valori x e y passati come parametri (-5.0 – +5.0)
  - Nome inizializzato in modo casuale
- Stampare il contenuto di p

```
Tpunto init_punto(float x, float y);
void stampa_punto(Tpunto p);
void str_casuale(char s[MAX], int len);
int casuale(int min, int max);
```

Esercizio 6

```
typedef struct Tpunto{
  float x, y;
  char nome[MAX];
} Tpunto;

/* Inizializza singolo punto */
Tpunto init_punto(float x, float y);
/* Stampa singolo punto */
void stampa_punto(Tpunto p);
/* Genera valori casuali tra i valori min e max */
int casuale(int min, int max);
/* Popola str in modo casuale con len caratteri */
void str_casuale(char s[MAX], int len);
```

Esercizio 7

- Definire con typedef la struttura dati FANTACALCIO: una lista di squadre con i relativi allenatori.
- L'**allenatore** è una persona caratterizzata da
  - nome (stringa)
  - cognome (stringa)
  - numero di coppe vinte (intero >=0)
- La singola **squadra** è identificata dal
  - nome squadra (stringa)
  - colore della casacca (stringa)
  - punteggio corrente (intero >=0)
  - l'allenatore.
- Calcolare (**con un ciclo**) e visualizzare il solo cognome degli allenatori di squadre che hanno più di 30 punti in classifica, nonché il numero totale di allenatori che hanno vinto almeno una coppa.

Soluzione – struct

```
#define NUM_SQUADRE 30
#define MAX_L 20
#define LEN 5

typedef char Stringa[MAX_L];

typedef struct Tallenatore {
        Stringa nome;
        Stringa cognome;
        int NCoppe;
} Tallenatore;

typedef struct Tsquadra{
        Stringa nome;
        Stringa colore;
        int punteggio;
        Tallenatore allenatore;
}Tsquadra;

typedef Tsquadra Tfantacalcio[NUM_SQUADRE];
```

Soluzione – senza funzioni

```
int main (int argc, const char * argv[]) {

  Tfantacalcio fantacalcio;
  int NAllConCoppa, i, n_squadre;

  printf("Quante squadre partecipano? (< %d)", NUM_SQUADRE);
  scanf("%d", &n_squadre);
  // controllo input
  while (n_squadre<0 || n_squadre>30) {
    printf("Numero errato. Num squadre? (< %d)", NUM_SQUADRE);
    scanf("%d", &n_squadre);
  }
  printf("\n\t\t**********\n\t\t* FANTACALCIO *\n\t\t**********");

  /* INSERIMENTO DATI */
  for (i=0; i<n_squadre; i++){
    scanf("%s", fantacalcio[i].nome);
    scanf("%s", fantacalcio[i].colore);
    scanf("%d", &fantacalcio[i].punteggio);
    scanf("%s", fantacalcio[i].allenatore.nome);
    scanf("%s", fantacalcio[i].allenatore.cognome);
    scanf("%d", &fantacalcio[i].allenatore.NCoppe);
  }
```

Soluzione – senza funzioni

```
  /* VISUALIZZAZIONE DATI */
  for (i=0; i<n_squadre ; i++){
    printf("\n\t\t******\n\t\t* FANTACALCIO *\n\t\t******");
    printf("\n\n\t\t Squadra nø%d\n",i+1);
    printf("%s %s %d\n", fantacalcio[i].nome,
        fantacalcio[i].colore,
        fantacalcio[i].punteggio);
    printf("\t%s %s %d\n", fantacalcio[i].allenatore.nome,
        fantacalcio[i].allenatore.cognome,
        fantacalcio[i].allenatore.NCoppe);
  }
  NAllConCoppa=0;

  printf("\n\t\t******\n\t\t* FANTACALCIO *\n\t\t******");
  printf("\n\n Gli allenatori con punteggio > di 30 punti: \n");
  for (i=0; i<n_squadre; i++){
    if (fantacalcio[i].punteggio > 30)
      printf("\n\t\t%s",fantacalcio[i].allenatore.cognome);
    if (fantacalcio[i].allenatore.NCoppe > 0)
      NAllConCoppa++;
  }
  printf("\n\n Il numero di allenatori che hanno vinto almeno una coppa sono
: %d",NAllConCoppa);

  return 0;
}
```

Esercizio 7 – modifiche

- Rifare l'esercizio precedente ristrutturando il codice usando le funzioni:
  - popola(…)
    - popola i campi delle strutture
  - stampa(…)
    - stampa i campi delle strutture
  - elabora(…)
    - elabora l'output - stampa
  - casuale (…)
    - generazione casual

- Tutti i campi sono generati con valori casuali
  - nome, cognome, nome squadra, colore della casacca (stringa di 3 caratteri minuscoli)
  - numero di coppe vinte (intero 0-15 estremi inclusi)
  - punteggio corrente (intero 0-70 estremi inclusi)

Esercizio 7

```
/* Inizializzazione struttura */
void popola(Tfantacalcio f, int dim);

/* Elabora i dati della struttura */
void elabora(Tfantacalcio f, int dim);

/* Generazione numeri casuali */
int casuale(int min, int max);

/* Stampa */
void stampa(const Tfantacalcio f, int dim);
```

Esercizio 7 - ulteriori funzioni

```
/* Popola singolo alleantore */
Tallenatore popola_allenatore(Stringa nome, Stringa
cognome, int coppe);

/* Popola singola squadra */
Tsquadra popola_squadra(Stringa nome, Stringa colore, int
punti, Tallenatore all);

/* Stampa alleantore */
void stampa_allenatore(Tallenatore a);

/* Stampa squadra */
void stampa_squadra(Tsquadra s);

/* Generazione stringhe casuali - stringa in str */
void str_casuale(char str[], int len);
```

Esercizio 8

- Modifica Esercizio 4

- Implementare le seguenti funzioni

```
/* Restituisce a+b in ris */
void somma(int a[][MAX], int b[][MAX], int ris[][MAX],
int nr, int nc);

/* Restituisce a*b in ris */
void prodotto(int a[][MAX], int b[][MAX], int
ris[][MAX], int nr, int nc);
```

Esercizio 8 – main

```
int main(){
  int ma[MAX][MAX];
  int mb[MAX][MAX];
  int o[MAX][MAX];
  int nr; int nc
  nr=nc=4;
  init(ma, nr, nc);
  init(mb, nr, nc);
  stampa(ma, nr, nc);
  stampa(mb, nr, nc);
  somma(ma, mb, o, nr, nc);
  stampa(o, nr, nc);
  prodotto(ma, mb, o, nr, nc);
  stampa(o, nr, nc);
  return 0;
}
```

Esercizio 9

- Estensione Esercizio 8

- Considerare una struttura struct Tmatrice con i seguenti campi
  - matrice mat di dimensioni massima MAX x MAX
  - nr, nc: dimensioni effettive della matrice
- Implementare le funzioni

```
/* Stampa matrice */
void stampa_v2(Tmatrice m);
```

Esercizio 9 – main

```
int main(){
  Tmatrice ma, mb, o;
  ma.nr=ma.nc=mb.nr=ma.nc=o.nr=o.nc=4
  init(ma.mat, ma.nr, ma.nc);
  init(mb.mat, ma.nr, ma.nc);
  //stampa(ma.mat, ma.nr, ma.nc);
  stampa_v2(ma);
  stampa_v2(mb);
  somma(ma.mat, mb.mat, o.mat, ma.nr, ma.nc);
  stampa_v2(o);
  prodotto(ma.mat, mb.mat, o.mat, ma.nr, ma.nc);
  stampa_v2(o);
  return 0;
}
```

Esercizio 10

- Estensione Esercizio 9
- Sfruttare la struct Tmatrice
- Implementare le funzioni

```
/* Restituisce a+b */
Tmatrice somma(Tmatrice a, Tmatrice b);

/* Restituisce a*b */
Tmatrice prodotto(Tmatrice a, Tmatrice b);
```

Esercizio 11

- Si leggano due stringhe da tastiera e si verifichi se le due stringhe sono uguali o diverse
- Se le due stringhe sono diverse stamparle in ordine alfabetico
  - Per la soluzione dell'esercizio si richiede di definire una funzione **`confronta_stringhe`** che restituisca i valori:
    - 0 se le stringhe sono uguali
    - 1 se la prima stringa precede in ordine alfabetico la seconda
    - 2 se la seconda precede la prima in ordine alfabetico

Esercizio 11

- Confronto tra stringhe

```
int confronta_stringhe(char stringa1[],
char stringa2[]);
```

  - La funzione confronta_stringhe, deve utilizzare al suo interno la funzione lunghezza_stringa (dichiarata sotto) , che calcola la lunghezza di una stringa. Questa deve essere utilizzata per calcolare le lunghezze delle stringhe passate alla funzione confronta_stringhe()

```
int lunghezza_stringa(char stringa[]);
```

Esercizio 11

- Suggerimenti
  - Controllare le stringhe carattere per carattere
  - I caratteri si possono confrontare (<,>,==,!=)
  - Il carattere terminatore '\0' è minore di qualunque lettera
  - Usare la funzione lunghezza_stringa per sapere quanti caratteri delle due stringhe bisogna controllare

Note – stringhe

```
#include <string.h>
```

```
int strcpy(char dst[], char src[]);
```

- dst: stringa di destinazione
- src: stringa sorgente
- Copia src in dst

```
int strlen(char s[]);
```

- Calcola e restituisce lunghezza stringa

Note – stringhe

- Generazione casuale di stringa

```
void str_casuale(char str[], int len){
  int i;
  for(i=0 ; i<len ; i++){
    str[i] = casuale('a', 'z');
  }
  // terminatore stringa
  str[len]='\0';
  // prima lettera maiuscola
  str[0]=str[0]-'a'+'A';
}
```