UNIVERSITY OF TRENTO - Italy

Laboratorio 10

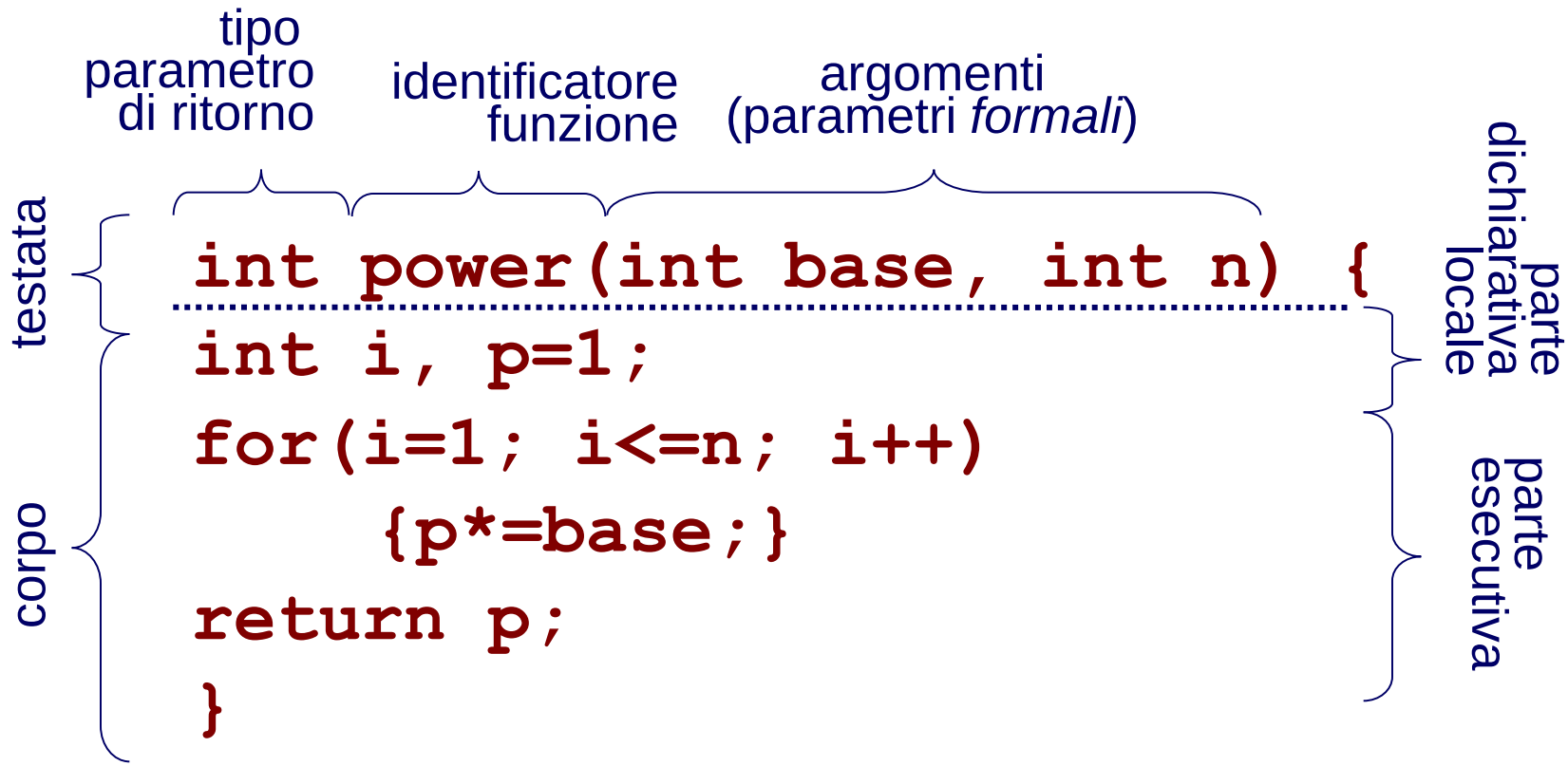
Pierluigi Roberti
Carmelo Ferrante

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

Definizione delle funzioni: sintassi

- La definizione di una funzione è composta da:
  - Una ***testata*** (o *header*), che contiene informazioni rilevanti ai fini di un uso corretto della funzione e cioè:
    - Tipo del risultato (il codominio della funzione)
    - Identificatore del sottoprogramma
    - Lista di dichiarazioni degli argomenti della funzione (il dominio della funzione)
    - **PRIMA DEL MAIN**
  - Un blocco, detto ***corpo*** (o *body*) della funzione
    - **DOPO DAL MAIN**
- A sua volta, il corpo della funzione è costituito da:
  - Una parte dichiarativa, detta ***parte dichiarativa locale***, che contiene le variabili necessarie all'esecuzione
  - Una ***parte esecutiva***, che contiene le istruzioni che costituiscono il corpo vero e proprio

Esempio

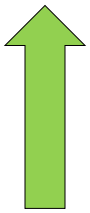

tipo parametro di ritorno
identificatore funzione
argomenti (parametri *formali*)
testata
corpo
parte dichiarativa locale
parte esecutiva
int power(int base, int n) {
int i, p=1;
for(i=1; i<=n; i++)
{p*=base;}
return p;
}

Nota

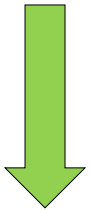
Quantità: 0 oppure N parametri

Ogni parametro: tipo_parametro nome_parametro

tipo_parametro ➔ int, char, float, TipoStruct, ...



```
Tipo_dato nome( [parametri] );
```



Quantità:
0 (`void`) oppure 1 parametro (`int, float, char, o tipo definito utente`)

Tipo funzione ➔ tipo valore di ritorno

Nessun valore di ritorno ➔ tipo **void**
Un valore di ritorno ➔ tipo int, char, float, TipoStruct, ...

Nota

```
void f1();
```

```
void f2(int a);
void f3(int a, char b);
void f4(int a, char b, TipoStruct c);
```

```
int f5();
int f6(float q);
int f7(float q, char w);
int f8(float q, char w, TipoStruct e);
```

```
TipoStruct f9();
TipoStruct fa(int x);
TipoStruct fb(int x, char y, TipoStruct z);
```

Funzioni nel linguaggio C/C++

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

// Dichiarazioni/Prototipi delle funzioni
// NOTA: non è rilevante l'ordine con cui
vengono scritti

int main(void){
  return 0;
}

// Implementazione delle funzioni
// NOTA: non è rilevante l'ordine con cui
vengono scritti
```

Funzioni nel linguaggio C/C++

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
//bool numero_primo(int numero);

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 *\
}
```

```
Ordine rilevante
FALSE = 0
TRUE  = 1
Nelle condizioni i numeri
hanno la seguente
interpretazione
0 => falso
1 (diverso da 0) => vero
```

dichiarazione (prototipo) della funzione

invocazione della funzione

definizione della funzione

Funzioni: passaggio variabili

```
#include <stdio.h>
#include <stdlib.h>
int random (int min, int max);

int main (int argc, const char * argv[]){
 int valint;
 srand(time(0));
 valint = random(-3,+3);
 printf("%d \n",valint);
  //...
 system("PAUSE");
 return 0;
}

int random (int min, int max){
 return rand()%(max-min+1)+min;
}
```

dichiarazione (prototipo) della funzione

invocazione della funzione

definizione della funzione

Funzioni void – esempi

```
void stampaacapo();

int main(){
   // …
   stampaacapo();
   // …
}

void stampaacapo(){
   printf("\n");
}
```

Funzioni void – esempi

```
void stampaInt(int n);

int main(){
   int k;
   // …
   stampaInt(2);
   stampaInt(k);
   // …
}

void stampaInt(int n){
   printf("%d\n", n);
}
```

Funzioni void – esempi

```
void stampaF(float n);

int main(){
   float k;
   // …
   stampaF(3.8);
   stampaF(k);
   // …
}

void stampaF(float n){
   printf("%f\n", n);
}
```

Funzioni void – esempi

```
void stampa(int n1, float n2);

int main(){
  float k=1.5; int a=3;
  // …
  stampa(4, 3.8);
  stampa(a, a*k);
  // …
}

void stampa(int n1, float n2){
  printf("%d %f\n", n1, n2);
}
```

Esercizio 1

- Con gli algoritmi visti a lezione, scrivere:
  - una funzione che trova il minimo tra 2 numeri;
  - una funzione che dati due numeri genera un numero casuale compreso tra questi numeri
  - una funzione che verifica se tra due numeri il primo divide il secondo;
  - una funzione che verifica se un numero è primo;
- Queste sono le dichiarazioni delle funzioni:
  - `int minimo(int primo, int secondo);`
  - `int random(int min, int max);`
  - `Boolean checkdivisore(int divisore, int dividendo);`
  - `Boolean numero_primo(int numero);`
- Queste sono le dichiarazioni dei tipi di dati

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

Suggerimenti

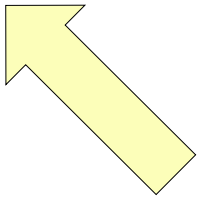
- Scrivere le funzioni e provarle, chiamandole all'interno del **`main()`**
- Ricordarsi di inserire la dichiarazione delle funzioni prima del **`main()`** mentre la definizione delle funzioni va dopo il **`main()`**
- Per la funzione che calcola il numero primo, si può riusare qualcuna delle funzioni?
- Attenzione a quanti argomenti servono in ogni funzione ed al loro tipo
- Attenzione ai valori di ritorno delle funzioni (**`return`**)

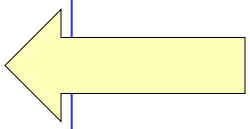
int minimo(int primo, int secondo);

```
int minimo (int primo, int secondo){
  if (primo < secondo) {
    return primo;
  } else {
    return secondo;
  }
}
```

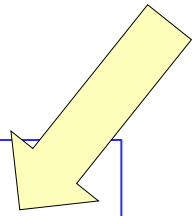
```
int minimo (int primo, int secondo){
  if (primo < secondo) {
    return primo;
  }
  return secondo;
}
```

```
int minimo (int primo, int secondo){
  return primo < secondo ? primo : secondo;
}
```



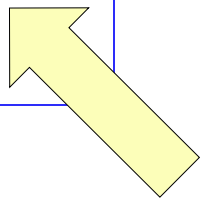


ALTERNATIVE



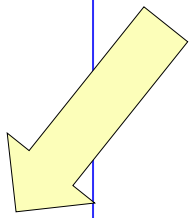
int random(int min, int max);

```
int random(int min, int max) {
  return rand()%(max-min+1)+min;
}
```



ALTERNATIVE

```
int random(int min, int max) {
  int n;
  n = rand()%(max-min+1)+min;
  return n;
}
```



int checkdivisore(int divisore, int dividendo);

```
int checkdivisore(int divisore, int dividendo) {
  //if( dividendo % divisore ) { /* != 0 */
  if( dividendo % divisore == 0 ) {
    return 1;
  } else {
    return  0;
  }
}
```

ALTERNATIVE

```
int checkdivisore(int divisore, int dividendo){
   return ( dividendo % divisore ) ? 0 : 1;
}
```

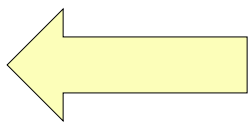
```
int checkdivisore(int divisore, int dividendo){
   return !( dividendo % divisore );
}
```

Boolean checkdivisore(int divisore, int dividendo);

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

Boolean checkdivisore(int divisore, int
dividendo){
  if(dividendo%divisore==0) { return TRUE; }
  return FALSE;
}

void main() {
  Boolean checkprimo;
  checkprimo = numero_primo(7);
  //if(checkprimo) { … }
  if(checkprimo == TRUE) {
    printf ("e' un numero primo\n");
  }else{
    printf ("non e' un numero primo\n");
  }
```



In C non esiste il tipo bool ALTERNATIVE

typedef enum {FALSE, TRUE} Boolean;
FALSE corrisponde a 0
TRUE corrisponde a 1
ORDINE VOLUTO!!

Per poter scrivere nel main solo
if (checkprimo){
//…
}

int numero_primo(int numero);

```
int numero_primo(int numero){
   int n=2;
   while(n<=numero/2){
    if( numero%n==0 ){ return 0; } /*if(!(numero%n))*/
    n++;
   }
   return 1;
}
```

```
Boolean numero_primo(int numero){
   int n=2;
   while(n<=numero/2){
     if( checkdivisore(n, numero) !=0 ){
       return FALSE;
     }
     n++;
   }
   return TRUE;
}
```

Invocazioni - esempi

```
int main(){
  int a, b, c, d, ris;
  // … inizializzazione variabili …
  ris = min (a, b);
  ris = random (a, c);
  ris = checkdivisore (d, c);
  ris = numero_primo (d);
  printf(" minimo: %d", min(a, c));
  printf(" %d %s" , c,
         numero_primo(c) ? "primo" : "non primo");

}
```

Esercizio 2

- Con gli algoritmi visti a lezione, scrivere:
  - una funzione che restituisca un valore reale y secondo un data relazione y = F(x)
  - una funzione che dati due numeri genera un numero casuale compreso tra questi numeri
    (usarla per generare un valore casuale float)
  - una funzione che stampa 2 numeri float nella forma [x,y]
  - una funzione che verifica se un numero è nell'intorno dello 0;
- Queste sono le dichiarazioni delle funzioni:

```
float funz (float x);
int random (int min, int max);
void stampa (float x, float y);
Boolean zero (float numero);
```

Verifica su un intorno dello zero: usare EPS

- Queste sono le dichiarazioni dei tipi di dati

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

- Queste sono le dichiarazioni delle costanti

```
#define N 20
#define EPS 0.00001
```

Esercizio 2

- Includere la libreria matematica

```
#include <math.h>
```

- relazioni y = F(x) da provare:
  - $x * (x - 1)^2$ /*3.05 $^{1.98}$ => pow(3.05, 1.98) */
  - $2 * x^2 + x - 3$
  - 2 * cos (x)
  - sin(x) * cos (x)
  - sqrt (x + 6) /*radice quadrata*/

Esercizio 2

Nel main

- Prima parte

  Generare **N** volte

  - un numero casuale (funzione random) reale compreso tra -5 e +5 (con 2 cifre decimali) (variabile x)
  - Richiamare la funzione funz passando il valore casuale così generato e salvare il risultato nella variabile y
  - Stampare x e y invocando la funzione stampa
  - Verificare se il valore di y è pari a zero in tal caso stampare («intercetta asse x»), usare la funzione zero
- Seconda parte
  - Generare i numeri reali compresi tra -5 e +5 passo 0.2 (variabile x), usando un ciclo for
  - Richiamare la funzione funz passando il valore così generato e salvare il risultato nella variabile y
  - Verificare se il valore di y è pari a zero in tal caso stampare («intercetta asse x»), usare la funzione zero
  - Stampare x e y invocando la funzione stampa

Esercizio 3

- Definire una struct Tpunto

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

- Nel main Definire variabile p1 e p2 di tipo **Tpunto**
- Inizializzare p1 e p2
  - Funzione `init_punto`
  - Campi x e y inizializzati con valori compresi tra -2.00 e +2.00
- Stampare il contenuto di p1 e p2
- Calcolare la distanza tra 2 punti passando p1 e p2

Esercizio 3

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

/* ritorna un punto inizializzato */
Tpunto init_punto();

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

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

/* calcola distanza tra punto p1 e p2 */
float distanza(Tpunto p1, Tpunto p1);
```

Esercizio 4

- Calcolo del coefficiente binomiale

$$\binom{n}{k} = C(n;k) = \frac{n!}{k! \cdot (n-k)!}, \qquad n, k \in \mathbb{N},\ 0 \leq k \leq n,$$

```
int fattoriale(int n) {
    if (n <= 1) {
      return 1;// Caso Base
    }  else {
      return n * fattoriale(n - 1);  // Chiamata Ricorsiva
    }
}

int binomio(int n, int k)  {
// C(n,k)= n! / (k! * (n-k)!)
  int res;
  res = fattoriale(n)/( fattoriale(k) * fattoriale(n-k) );
  return res;
}
```

Esercizio 4

```
int main(int argc, char *argv[]){
  int n,k;
  printf("n="); scanf("%d",&n);
  printf("k="); scanf("%d",&k);

  //manca il controllo sul valore delle variabili
  printf("c(n,k)=%d \n", binomio(n,k));

  system("PAUSE");  return 0;
}
```

$$n, k \in \mathbb{N},\ 0 \leq k \leq n,$$

$$\binom{5}{3} = \frac{5!}{3!(5-3)!} = \frac{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{3 \cdot 2 \cdot 1 \cdot (2 \cdot 1)} = \frac{120}{12} = 10$$