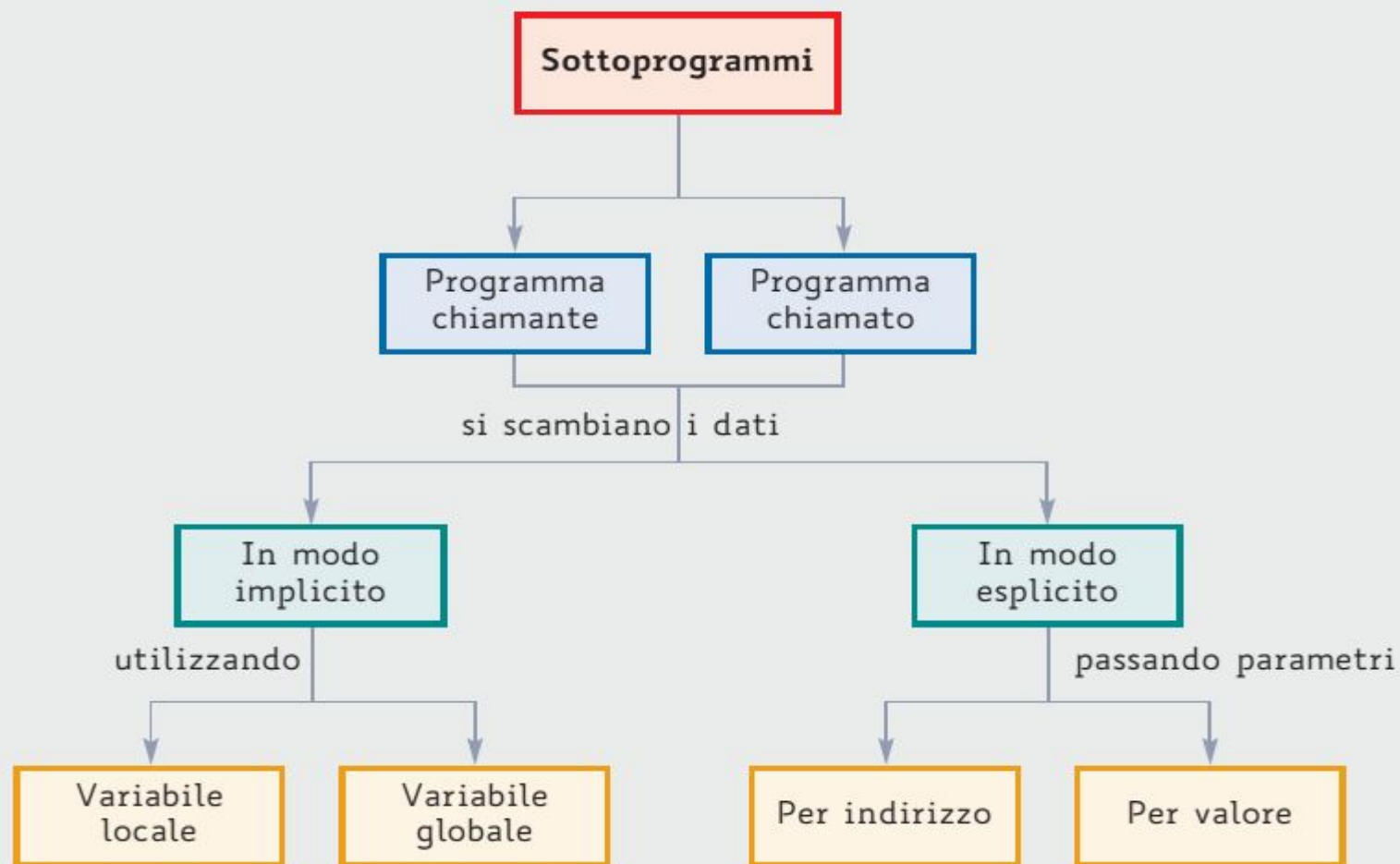


WEB DEVELOPER
Fondamenti di Programmazione
Massimo PAPA

Le funzioni e le procedure

In questa lezione impareremo...

- a definire una funzione
- a definire la modalità del passaggio dei parametri
- a utilizzare funzioni predefinite nei programmi
- a utilizzare funzioni personali



Sottoprogramma e funzioni

- La programmazione moderna **suddivide** il programma in più parti
- Sono **segmenti di codice** che realizzano operazioni semplici o complesse
- Possono essere visti come programmi “autonomi”, di servizio, da utilizzare all’interno di altri programmi.
- Sono chiamati **sottoprogrammi**

Sottoprogramma e funzioni

- I **sottoprogrammi** sono componenti o **atomi software**, da utilizzare come “mattoni” per la costruzione di programmi più complessi.
- Vantaggi

- ▮ è inutile scrivere due volte lo stesso codice;
- ▮ è inutile scrivere un codice che qualcun altro ha già scritto;
- ▮ è inutile scrivere un codice che qualcun altro ha già scritto e che funziona.

Sottoprogramma e funzioni

- L'utilizzo delle funzioni deriva quindi dalle tre motivazioni seguenti:
 - **riusabilità** (cioè utilizzare lo stesso codice come “mattoncino” per la soluzione di problemi diversi);
 - **astrazione** (esprimere operazioni complesse in modo sintetico);
 - **risparmio** (scrivere una sola volta codice usato più volte).

Funzioni in linguaggio C++: definizione

- la funzione necessita di una definizione formale con la seguente struttura:
 - testata (**header**);
 - istruzioni, racchiuse tra parentesi graffe e costituite da due parti:
 - la **parte dichiarativa** (detta parte dichiarativa locale);
 - la **parte esecutiva** (detta corpo della funzione).

Funzioni in linguaggio C++

- La testata è la parte più delicata, e presenta la struttura seguente:

tipo_restituito **nome_funzione** (*<lista_parametri_formali>*)

tipo_restituito o tipo del risultato

nome_funzione o identificatore di funzione

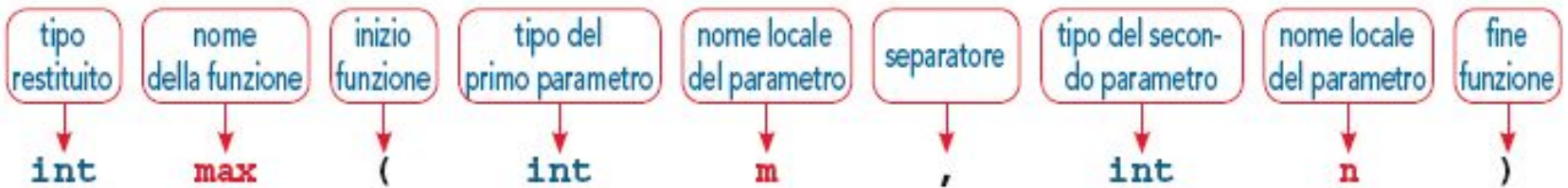
<lista_parametri_formali>: è un elenco di coppie costituite da **tipo_variabile** **identificatore_variabile**

Funzioni in linguaggio C++

- **Esempio:** Maggiore tra due numeri
- Realizziamo una funzione che determina e restituisce il valore maggiore tra due numeri.
- La funzione avrà:
 - in ingresso due numeri interi;
 - in uscita un valore intero;
 - un nome significativo che ne permette l'utilizzo (per esempio, maggiore).

Maggiore tra due numeri

```
int maggiore(int n1, int n2) // header: parametri formali
{
    if (n1 >= n2)
        return n1;           // valore di ritorno
    else
        return n2;           // valore di ritorno
}
```



Maggiore tra due numeri

```
int maggiore(int n1, int n2) // header: parametri formali
{
    if (n1 >= n2)
        return n1;           // valore di ritorno
    else
        return n2;           // valore di ritorno
}



int main(void)
{
    int max, num1, num2;
    cout << "\nInserisci due numeri interi :\n";
    cin >> num1;
    cin >> num2;
    max = maggiore(num1, num2); // chiamata: parametri attuali
    cout << "Il maggiore tra i due numeri e' " << max << endl;

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

- parametri formali
- parametri attuali

Funzioni: chiamata e parametri attuali

prototipo	<code>int max(int m, int n)</code>	// formale
chiamata	<code>massimo=max(numero1, numero2);</code>	// attuale

elenco parametri formali	(<code>int m</code> , <code>int n</code>)
	
elenco parametri attuali	(<code>numero1</code> , <code>numero2</code>);
elenco parametri formali	(<code>int m</code> , <code>int n</code>)
	
elenco parametri attuali	(<code>base</code> , <code>altezza</code>);

ESEMPIO: **Maggiore tra tre numeri**

- Leggiamo ora tre numeri e individuiamo il maggiore effettuando una doppia chiamata alla funzione

```
int main(void)
{
    int max, num1, num2, num3;
    cout << "\nInserisci tre numeri interi :\n";
    cin >> num1;
    cin >> num2;
    cin >> num3;
    max = maggiore(num1, num2); // I chiamata: parametri attuali
    max = maggiore(max, num3);  // II chiamata: parametri attuali
    cout << "Il maggiore tra i tre numeri e' " << max << endl;
    return 0;
}
```

all'atto di questa chiamata della
funzione si effettua un legame tra

n1 e n2
↓ ↓
num1 e num2

all'atto di questa chiamata della
funzione si effettua un legame tra

n1 e n2
↓ ↓
max e num3

Un esempio completo

✓ Il problema

Leggi il raggio di una circonferenza e calcola il perimetro e l'area del cerchio che essa delimita.

✓ L'analisi e la strategia risolutiva

Dalla matematica, conosciamo le formule che ci permettono di calcolare l'area e il perimetro del cerchio:

- $\text{perimetro} = 2 \times \text{raggio} \times \pi$
- $\text{area} = \text{raggio} \times \text{raggio} \times \pi$

Un esempio completo

- Durante la fase di input verifichiamo che l'utente inserisca un numero positivo.

I affinamento

- leggi un numero
- calcola perimetro
- calcola l'area
- visualizza i risultati

II affinamento

- chiama leggi Dati ()
- chiama calcola Perimetro (raggio)
- chiama calcola Area (raggio)
- chiama stampa Risultati (perimetro, area)

III affinamento – pseudocodifica

```
int leggiDati()  
  mentre (raggio <=0) fai  
    leggi(raggio)  
  fineMentre  
ritorna raggio
```

```
int calcolaPerimetro(int raggio)  
  perimetro <- 2*raggio*pigreco  
ritorna perimetro
```

```
int calcolaArea(int raggio)  
  area <- raggio *raggio*pigreco  
ritorna area
```

```
void stampaRisultati(int perimetro, int area)  
  scrivi("la misura del perimetro e'", perimetro)  
  scrivi("la misura dell'area e'", area)  
ritorna void
```

```
inizio //programma principale  
  raggio <- leggiDati()  
  perimetro <- calcolaPerimetro(raggio)  
  area <- calcolaArea(raggio)  
  stampaRisultati(perimetro, area)  
fine
```

Parametri formali

Parametri attuali

✓ Codifica in C++ ed esecuzione del programma

```
#include <iostream>
using namespace std;
const double PIGRECO = 3.1415;           // costante comune - globale
int leggiDati(){                          // senza parametri di ingresso
    int dato = 0;                        // variabile locale
    while (dato <= 0){
        cout << "\nInserisci il raggio : ";
        cin >> dato;
    }
    return dato;                         // parametro di ritorno
}
double calcolaPerimetro(int raggio){
    double numero1;                     // variabile locale
    numero1 = 2 * raggio*PIGRECO;        // usa dati globali
    return numero1;                     // parametro di ritorno
}
double calcolaArea(int raggio){
    double numero1;                     // variabile locale
    numero1 = raggio * raggio * PIGRECO; // usa dati globali
    return numero1;                     // parametro di ritorno
}
void stampaRisultati(double dato1, double dato2){
    cout << "la misura del perimetro e': " << dato1 << endl;
    cout << "la misura dell'area e'      : " << dato2 << endl;
}
int main(void){
    int raggio;
    double perimetro, area;
    raggio = leggiDati();
    perimetro = calcolaPerimetro(raggio); // parametro attuale
    area = calcolaArea(raggio);           // parametro attuale
    stampaRisultati(perimetro, area);      // parametro attuale
    system("PAUSE");
    return 0;
}
```

```
Inserisci il raggio : 20
la misura del perimetro e': 125.66
la misura dell'area e'      : 1256.6
Prendere un tasto per continuare . . . _
```



METTITI ALLA PROVA



• Scomposizione • Chiamata di sottoprogrammi

Scrivi un programma che, tramite una funzione con il seguente prototipo:

```
bool paridispari(int numero); // prototipo C++
```

riceve un numero inserito dall'utente e determina se tale numero è pari oppure dispari.