UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione
DISI - Trento

Programmazione 1

09 - Esercitazione

Andrea Mazzullo
andrea.mazzullo@unitn.it

Anno Accademico 2023/2024

Nelle puntate precedenti

```
1. #include <iostream>
2. using namespace std;
3.
4. int calcolaSomma(int addendo1, int addendo2);
5.
6. int main() {
7.   int a = 1, b = 2;
8.   cout << calcolaSomma(a, b);
9. }
10.
11.int calcolaSomma(int addendo1, int addendo2) {
12.  int somma;
13.  somma = addendo1 + addendo2;
14.  return somma;
15.}
```

Nelle puntate precedenti

```
1. #include <iostream>
2. using namespace std;
3.
4. int calcolaSomma(int addendo1, int addendo2);
5.
6. int main() {
7.    int a = 1, b = 2;
8.    cout << calcolaSomma(a, b);
9. }
10.
11.int calcolaSomma(int addendo1, int addendo2) {
12.   int somma;
13.   somma = addendo1 + addendo2;
14.   return somma;
15.}
```

Dichiarazione

Nelle puntate precedenti

```
1. #include <iostream>
2. using namespace std;
3.
4. int calcolaSomma(int addendo1, int addendo2);
5.
6. int main() {
7.   int a = 1, b = 2;
8.   cout << calcolaSomma(a, b);
9. }
10.
11.int calcolaSomma(int addendo1, int addendo2) {
12.  int somma;
13.  somma = addendo1 + addendo2;
14.  return somma;
15.}
```

Chiamata

Nelle puntate precedenti

```
1. #include <iostream>
2. using namespace std;
3.
4. int calcolaSomma(int addendo1, int addendo2);
5.
6. int main() {
7.   int a = 1, b = 2;
8.   cout << calcolaSomma(a, b);
9. }
10.
11 int calcolaSomma(int addendo1, int addendo2) {
12.   int somma;
13.   somma = addendo1 + addendo2;
14.   return somma;
15.}
```

Definizione

00 - Let’s begin!

Overloading è dare lo stesso nome a funzioni con diverso numero, ordine **o** tipo di parametri formali

```
int max(int numero1, int numero2);
int max(int numero1, int numero2, int numero3);
int max(char carattere1, char carattere2);
...
```

00 - Firma

La “firma” di una funzione si compone (i) del nome della funzione **e** (ii) del numero, ordine e tipo dei parametri formali

```
int calcolaSomma(int addendo1, int addendo2)
```

firma: `calcolaSomma(int, int)`

Il valore di ritorno della funzione **non** fa parte della firma

00 - Visibilità di una Variabile

“Each name that appears in a C++ program is only valid in some [...] portion of the source code called its **scope**.”

“Ogni identificatore che appare in un programma C++ è valido soltanto in una [...] porzione del codice sorgente chiamata **scope**.”

https://en.cppreference.com/w/cpp/language/scope

00 - Visibilità di una Variabile

```
1. char lettera = 'g';

2. void f() {
3.     char lettera = 'f';
4.     cout << lettera;
5. }

6. int main() {
7.     f();
8.     cout << lettera;
9.     char lettera = 'm';
10.    cout << lettera;
11.}
```

variabile globale

variabile locale

variabile locale

00 - Visibilità di una Variabile

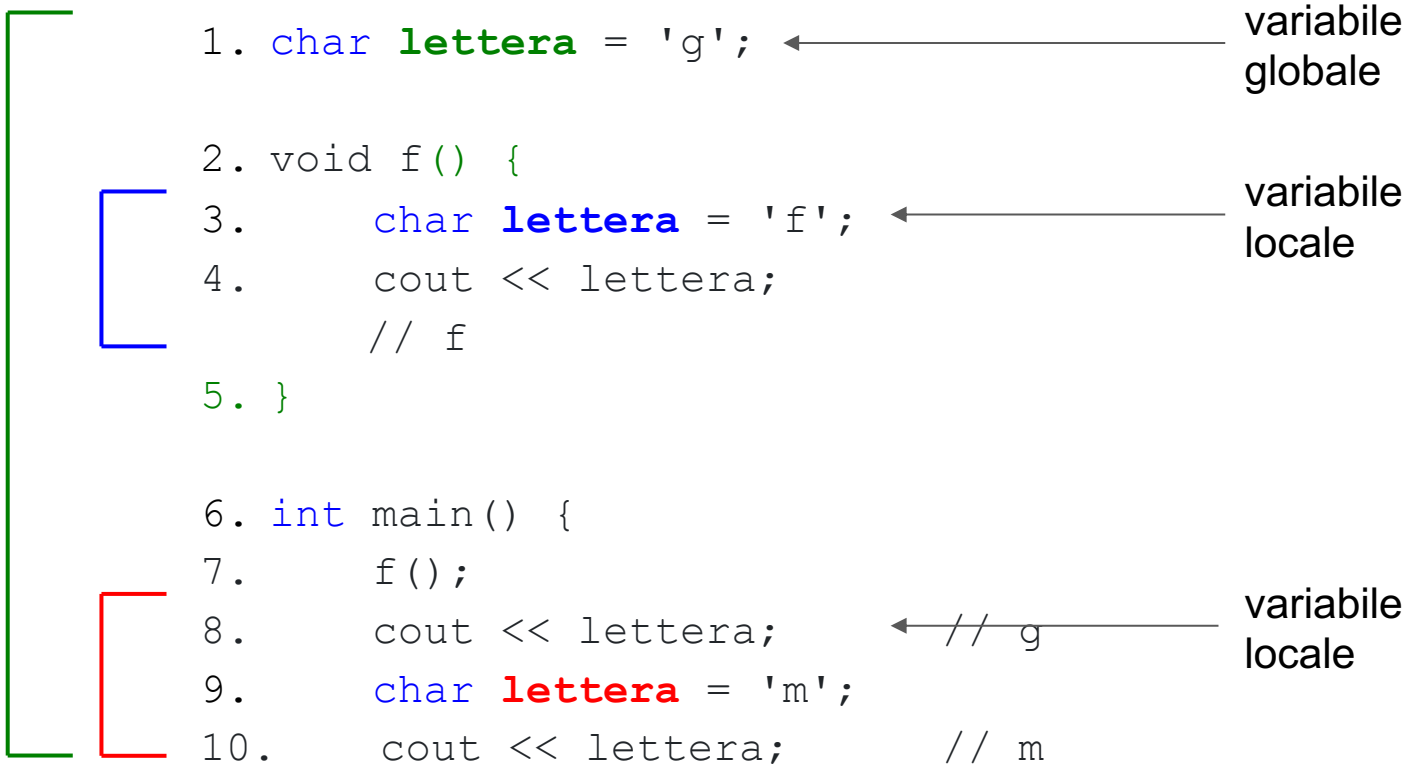
```
1. char lettera = 'g';

2. void f() {
3.     char lettera = 'f';
4.     cout << lettera;
       // f
5. }

6. int main() {
7.     f();
8.     cout << lettera;      // g
9.     char lettera = 'm';
10.    cout << lettera;      // m
11.}
```

variabile globale

variabile locale

variabile locale

00 - Visibilità di una Variabile



00 - Passaggio di Parametri

- **Per valore**
  copia il valore del parametro attuale ← eventuali modifiche non si riflettono sul parametro attuale
- **Per riferimento**
  il parametro è un riferimento (`&`) al parametro attuale
- **Per puntatore**
  il parametro è l'indirizzo del parametro attuale ← passaggio per valore del puntatore, ma ovviamente si può modificare la variabile puntata

00 - Passaggio di Parametri - Valore

```
void f(int numero) {
    numero = 2;
    cout << numero;    // 2
}

int main() {
        int numero = 1;
    f(numero);
    cout << numero;    // 1
}
```

00 - Passaggio di Parametri - Riferimento

```
void f(int & numero) {
    numero = 2;
    cout << numero;    // 2
}

int main() {
        int numero = 1;
    f(numero);
    cout << numero;    // 2
}
```

00 - Passaggio di Parametri - Puntatore

```
void f(int * pNumero) {
    *pNumero = 2;
    cout << *pNumero;  // 2
}

int main() {
        int numero = 1;
    f(&numero);
    cout << numero;    // 2
}
```

00 - Argomenti di Default

Usati per fornire parametri opzionali con valori di default

```
int max(int n1, int n2, int n3=0, int n4=0, int n5=0);

int main() {
    cout << max(1,2,3);              // 3
}

int max(int n1, int n2, int n3, int n4, int n5) {
        ...
}
```

01 - Swap

Dati in input due reali (`double`) in due variabili, scrivere un programma che usi una procedura per scambiare i valori delle due variabili usando il passaggio di parametri per puntatore.

02 - /

Dati in input due interi, scrivere un programma che faccia la divisione tramite una funzione; la funzione restituisce tramite valore di ritorno il quoziente e tramite parametro per riferimento il resto.
In questo programma non si può usare il simbolo '/' per la divisione

03 - secondi/minuti/ore

Dati in input tre interi in tre variabili (secondi, minuti, ore), scrivere un programma che con una procedura converta un eventuale eccesso di secondi in minuti e un eventuale eccesso di minuti in ore. Usare a scelta passaggio di parametri per riferimento o puntatore.

```
sec = 121;
min = 59;
hour = 2;
```

=>

```
sec = 1;
min = 1;
hour = 3;
```

04 - Max

Dati in input da 1 a 5 interi, scrivere una funzione che ritorni il valore massimo. Usare argomenti di default per il secondo, terzo, quarto e quinto input.

```
#include <climits>
INT_MIN // valore più piccolo per int
```

Per maggiori informazioni:
http://www.cplusplus.com/reference/climits/

05 - Sort

Dati in input tre interi positivi in tre variabili (n1, n2, n3), scrivere un programma che con una procedura “ri-ordini” i numeri in ordine crescente usando il passaggio di parametri per riferimento.

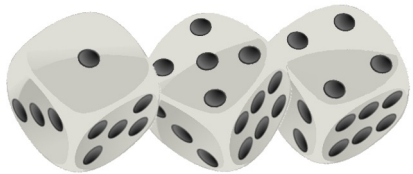
```
n1 = 3;          n1 = 2;
n2 = 7;    =>    n2 = 3;
n3 = 2;          n3 = 7;
```

Esercizi Aggiuntivi

Aggiuntivo - RisiKo! 3vs3

Scrivere un programma che simuli un attacco 3 contro 3 a RisiKo!. Tirare 3 dadi a 6 facce per l'attaccante e 3 per il difensore. Confrontare il dado più alto dell'attaccante contro il più alto del difensore, il medio dell'attaccante contro il medio del difensore e il più basso dell'attaccante contro il più basso del difensore. Dichiarare infine gli scontri vinti dall'attaccante e dal difensore.

Difensore



Attaccante

5 < **6** => Attaccante

4 = **4** => Difensore

1 < **2** => Attaccante

Aggiuntivo - Swap v2 - Difficile

Dati in input due interi (`short`), scrivere un programma che con una procedura scambi gli 8 bit meno significativi dei due numeri.

```
1855(0000011100111111)        1816(0000011100011000)
1048(0000010000011000)   =>   1087(0000010000111111)
```