Passaggio di parametri per valore e per riferimento

Liceo G.B. Brocchi – Bassano del Grappa (VI)
Liceo Scientifico – opzione scienze applicate
Giovanni Mazzocchin

Passaggio per valore

```
void f (int x_arg) {
        x_arg++;
        cout << "value of actual parameter is: " << x_arg << endl;
}

int main (int argc, char* argv[]) {
        int x_main = 6;
        f(x_main);
        cout << "value of main's variable is: " << x_main << endl;

        return 0;
}
```

```
value of actual parameter is: 7
value of main's variable is: 6
```

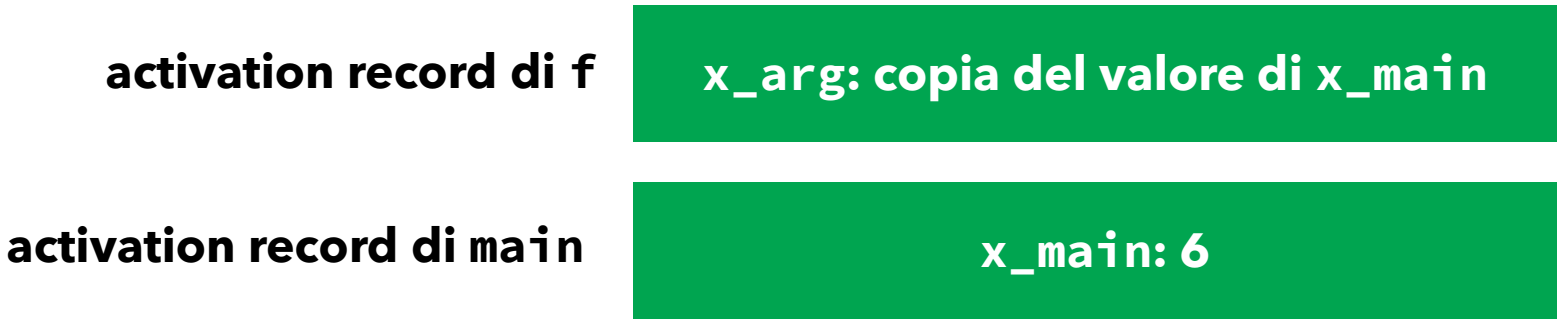
perché il valore di `x_main` non è cambiato?

Passaggio per valore

```
void f(int x_arg) {
        x_arg++;
        cout << "value of actual parameter is: " << x_arg << endl;
}

int main(int argc, char* argv[]) {
        int x_main = 6;
        f(x_main);
        cout << "value of main's variable is: " << x_main << endl;

        return 0;
}
```



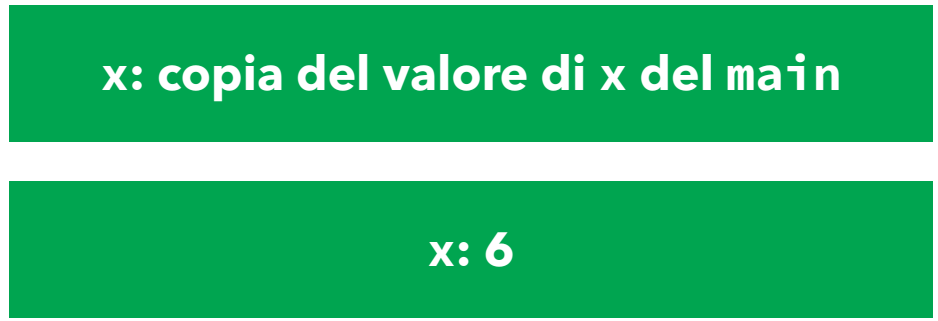
Passaggio per valore

```
void f(int x) {
        x++;
        cout << "value of actual parameter is: " << x << endl;
}

int main(int argc, char* argv[]) {
        int x = 6;
        f(x);
        cout << "value of main's variable is: " << x << endl;

        return 0;
}
```

x di `f` e x di `main` sono 2 variabili diverse. x di `f` prende solo il valore di x di `main`, quindi non stiamo passando una variabile, ma soltanto un valore



Passaggio per valore

```
void f(int x) {
        x++;
        cout << "address of f's x is: " << &x << endl;
        cout << "value of actual parameter is: " << x << endl;
}

int main(int argc, char* argv[]) {
        int x = 6;
        cout << "address of main's x is: " << &x << endl;
        f(x);
        cout << "value of x is: " << x << endl;
}
```

```
address of main's x is: 0093FF24
address of f's x is:    0093FF20
```

indirizzi diversi, posizioni di memoria diverse, quindi variabili diverse

Passaggio per valore (indirizzo)

```
void f(int* xptr) {
        *xptr = *xptr + 1;
}

int main(int argc, char* argv[]) {
        int x = 9;
        f(&x);
        cout << "value of x is: " << x << endl;
        return 0;
}
```

Passaggio per valore (indirizzo)

```
void f(int* xptr) {
        *xptr = *xptr + 1;
        cout << "value of xptr parameter in f function is: " << xptr << endl;
}

int main(int argc, char* argv[]) {
        int x = 9;
        cout << "initial value of x in the main function is: " << x << endl;
        cout << "address of x in the main function is: " << &x << endl;
        f(&x);
        cout << "value of x is: " << x << endl;

        return 0;
}
```

```
initial value of x in the main function is: 9
address of x in the main function is:     008FFE20
value of xptr parameter in f function is: 008FFE20
value of x is: 10
```

viene stampata la stessa cosa, perché?

Passaggio per valore (indirizzo)

```
void f(int* xptr, int inc) {
        *xptr = *xptr + inc;
}

int main(int argc, char* argv[]) {
        //call f in the right way
}
```

Passaggio per riferimento

- Finora abbiamo utilizzato i **puntatori** per fare **side-effect** sui parametri di una funzione, ossia per modificarli e rendere la modifica visibile al chiamante
- Sappiamo però che i puntatori sono scomodi
- Abbiamo visto i riferimenti. Usiamoli per fare side-effect sui parametri!

Passaggio per riferimento

```
void multiplicator(int& n, int m) {
  n = n * m;
}

int main() {
  int start = 1;

  for (int i = 0; i < 20; i++) {
    cout << start << endl;
    multiplicator(start, 2);
  }
}
```

```
address of start in main function is: 0133F968
address of n in multiplicator function is: 0133F968
```