

# Fondamenti di Programmazione (A)

## I5 - Funzioni

Vincenzo Arceri - Università degli Studi di Parma - [vincenzo.arceri@unipr.it](mailto:vincenzo.arceri@unipr.it)

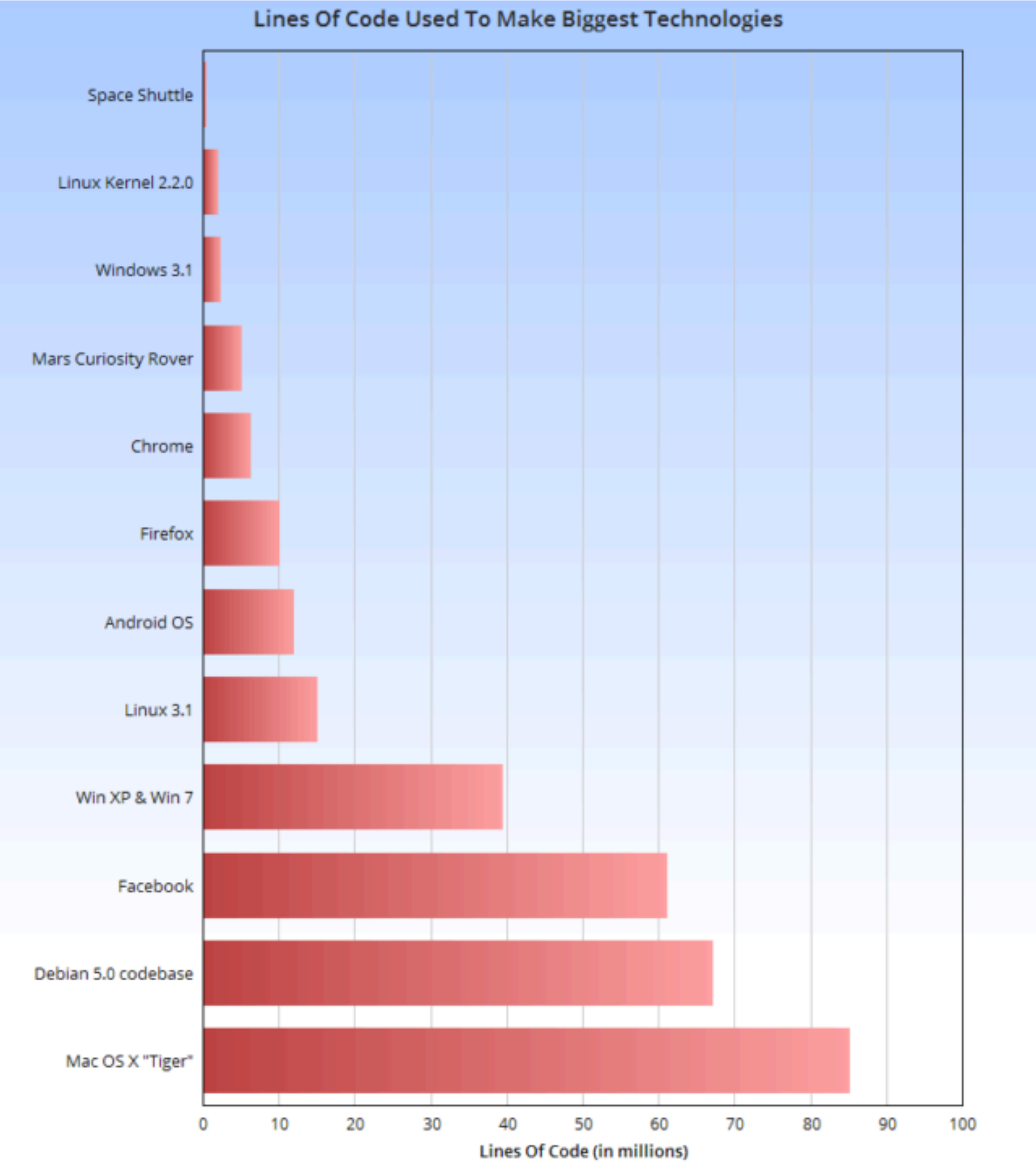
# Programmazione modulare

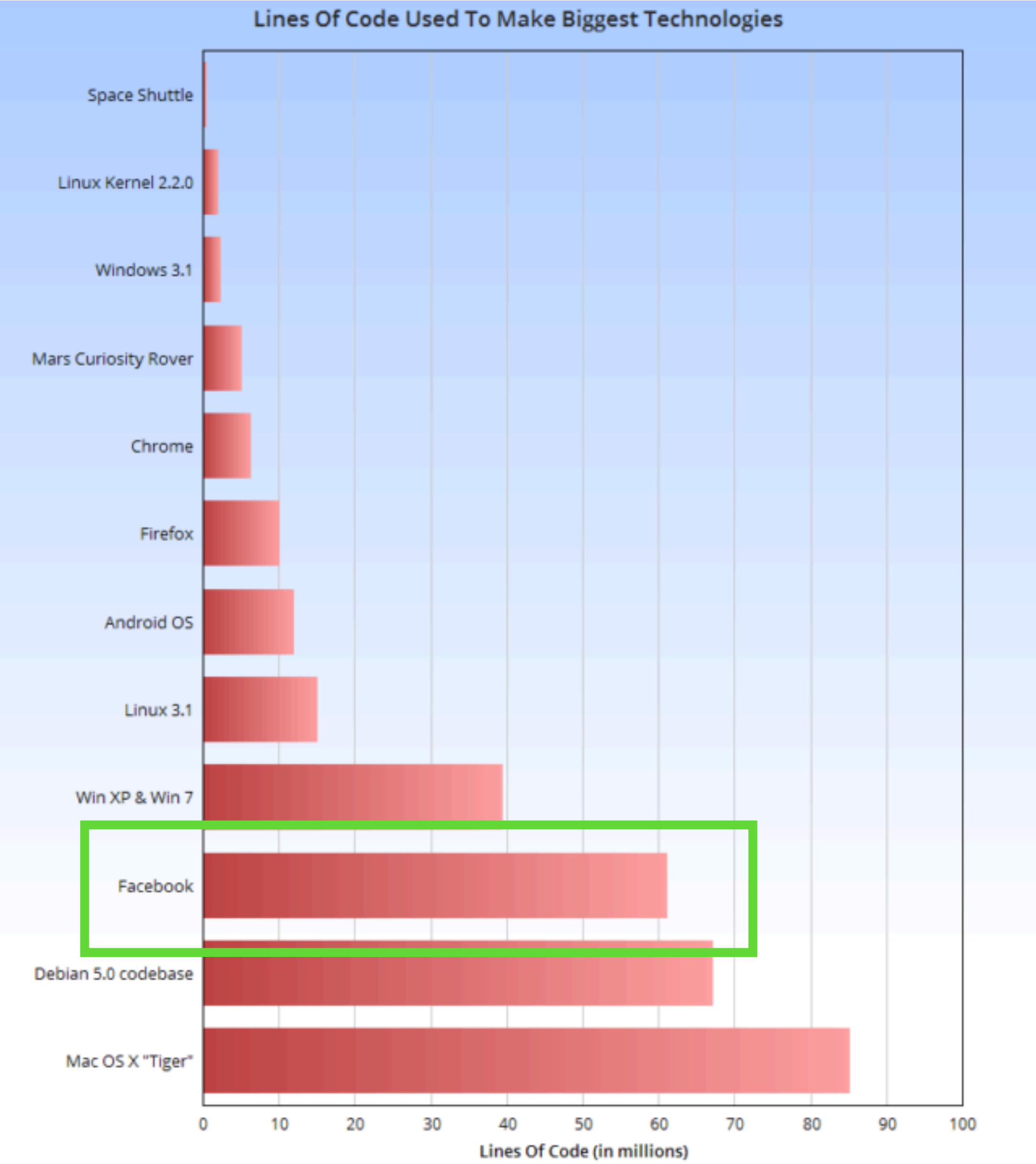
- Finora abbiamo scritto tutti i nostri programmi in un'unica funzione `main`

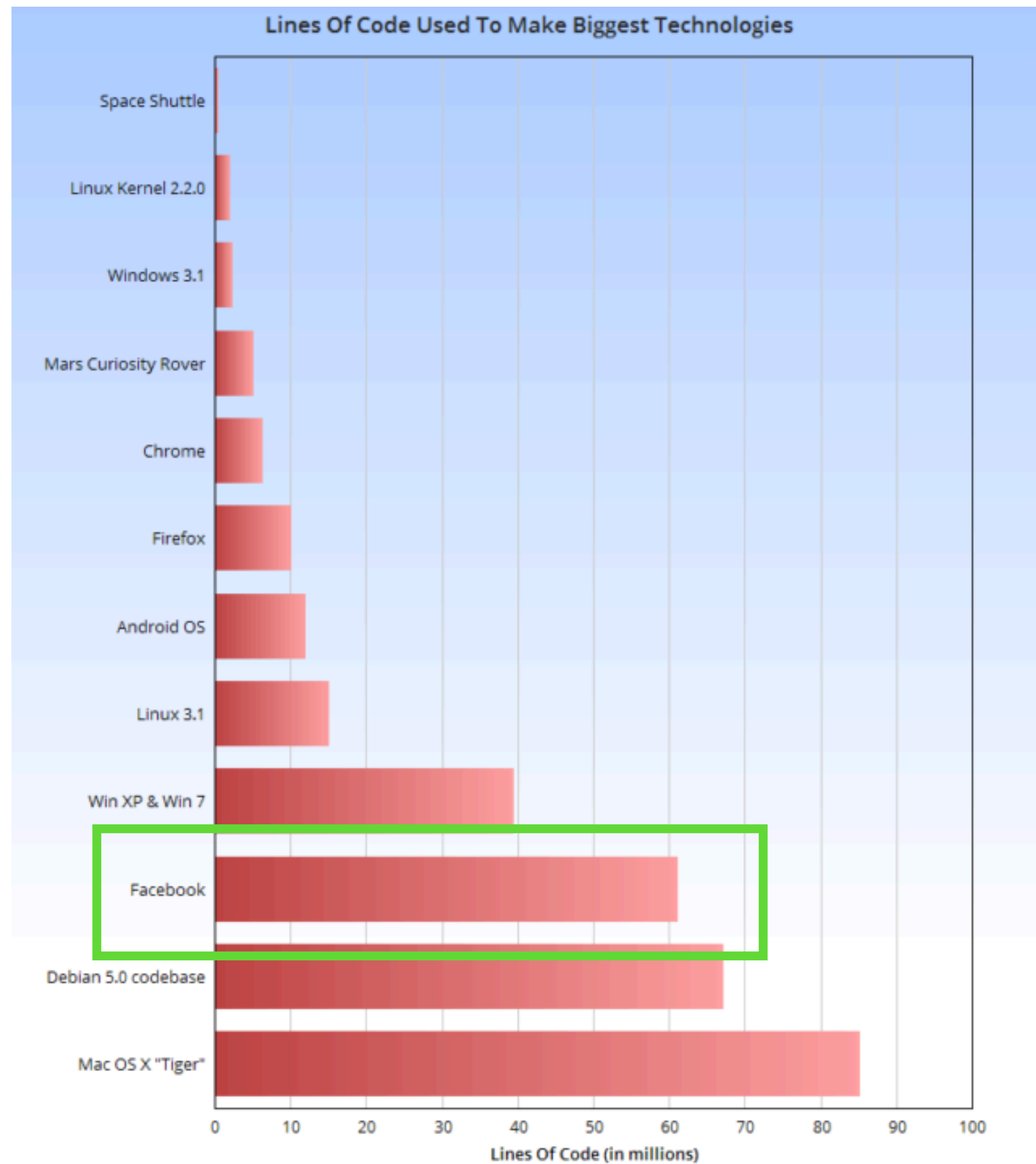
`main`



```
// code
```



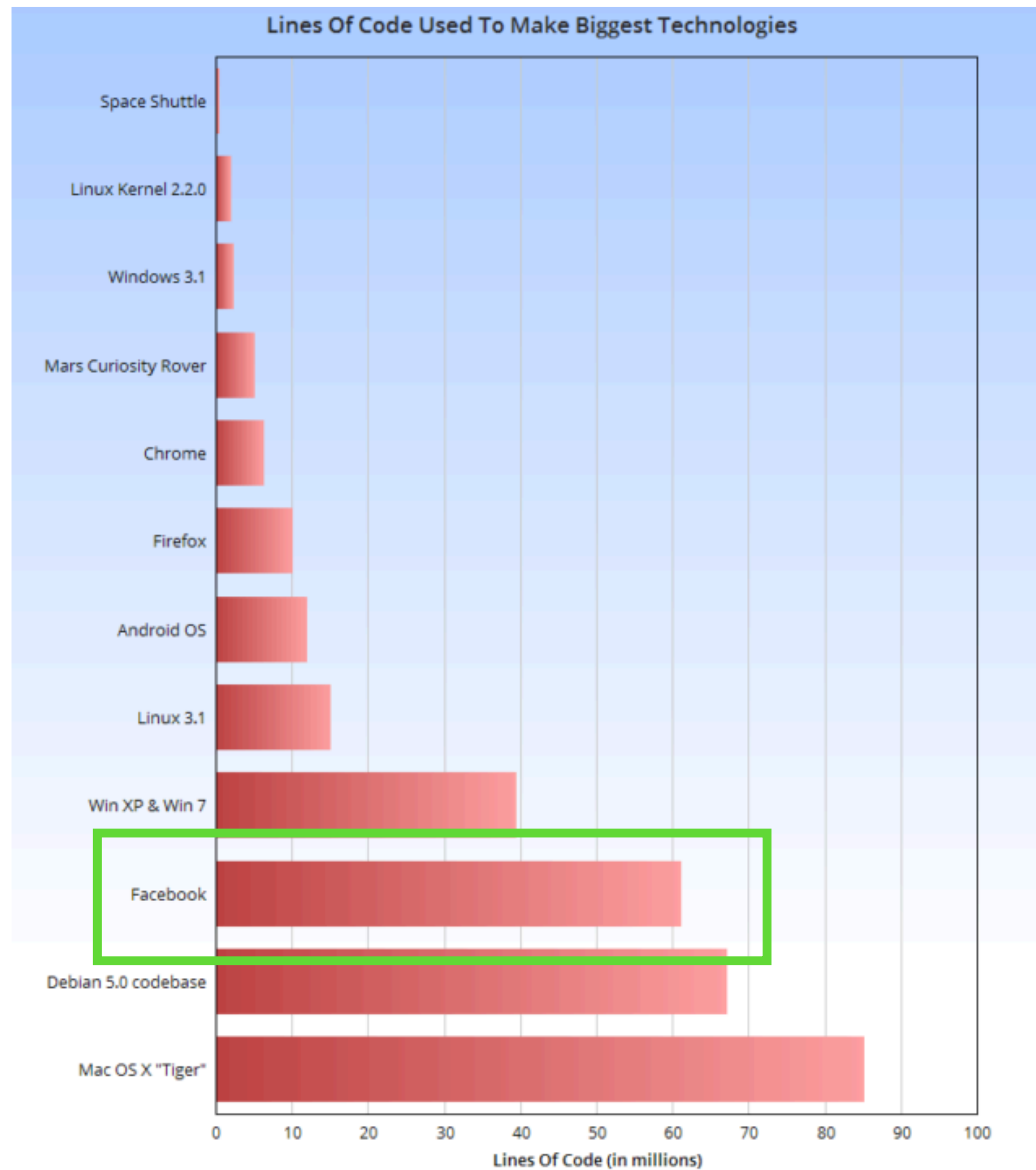




SOURCE: NASA, QUORA, WIKIPEDIA  
INFORMATION IS BEAUTIFUL



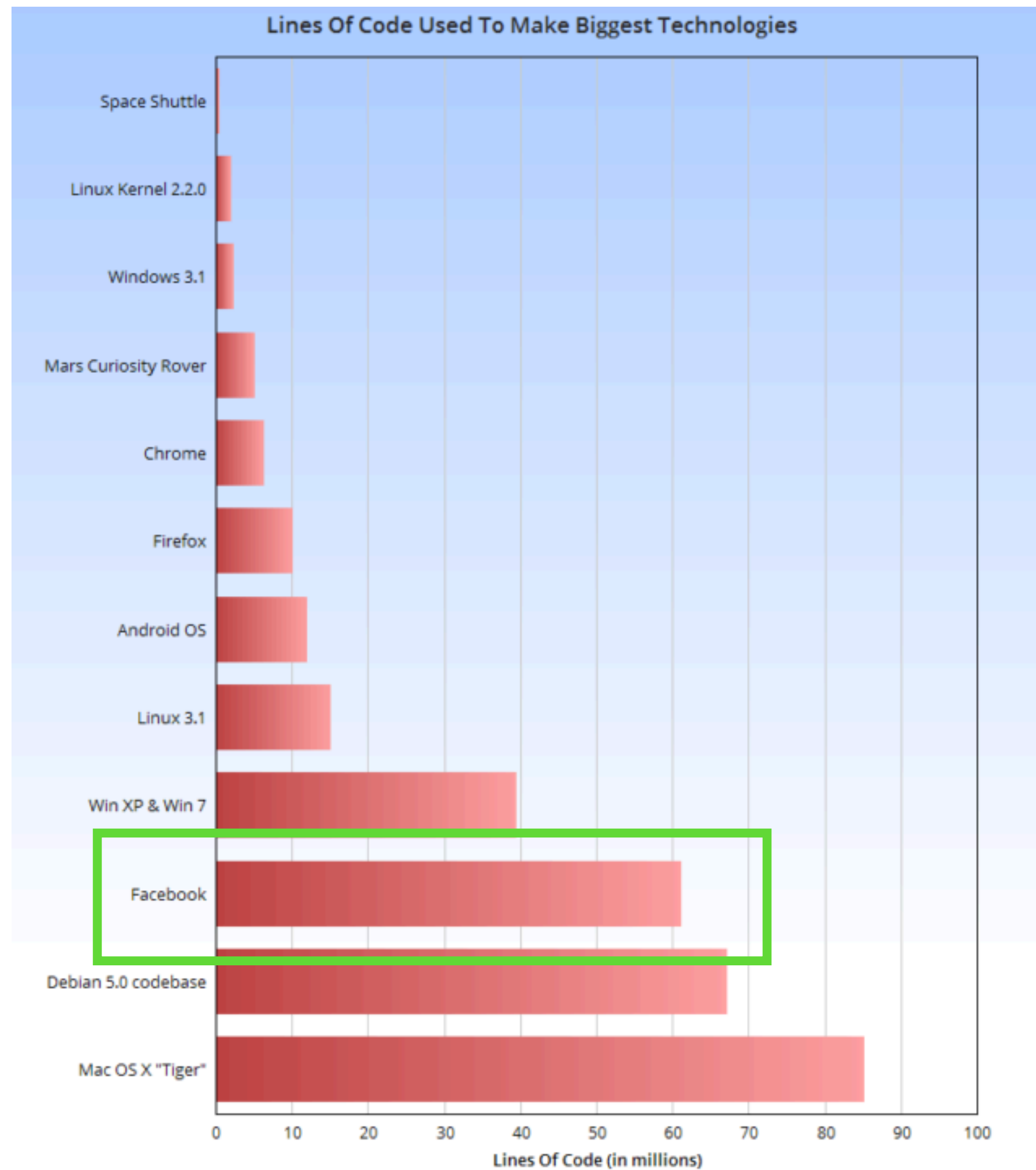
- Esempio: Facebook



SOURCE: NASA, QUORA, WIKIPEDIA  
INFORMATION IS BEAUTIFUL



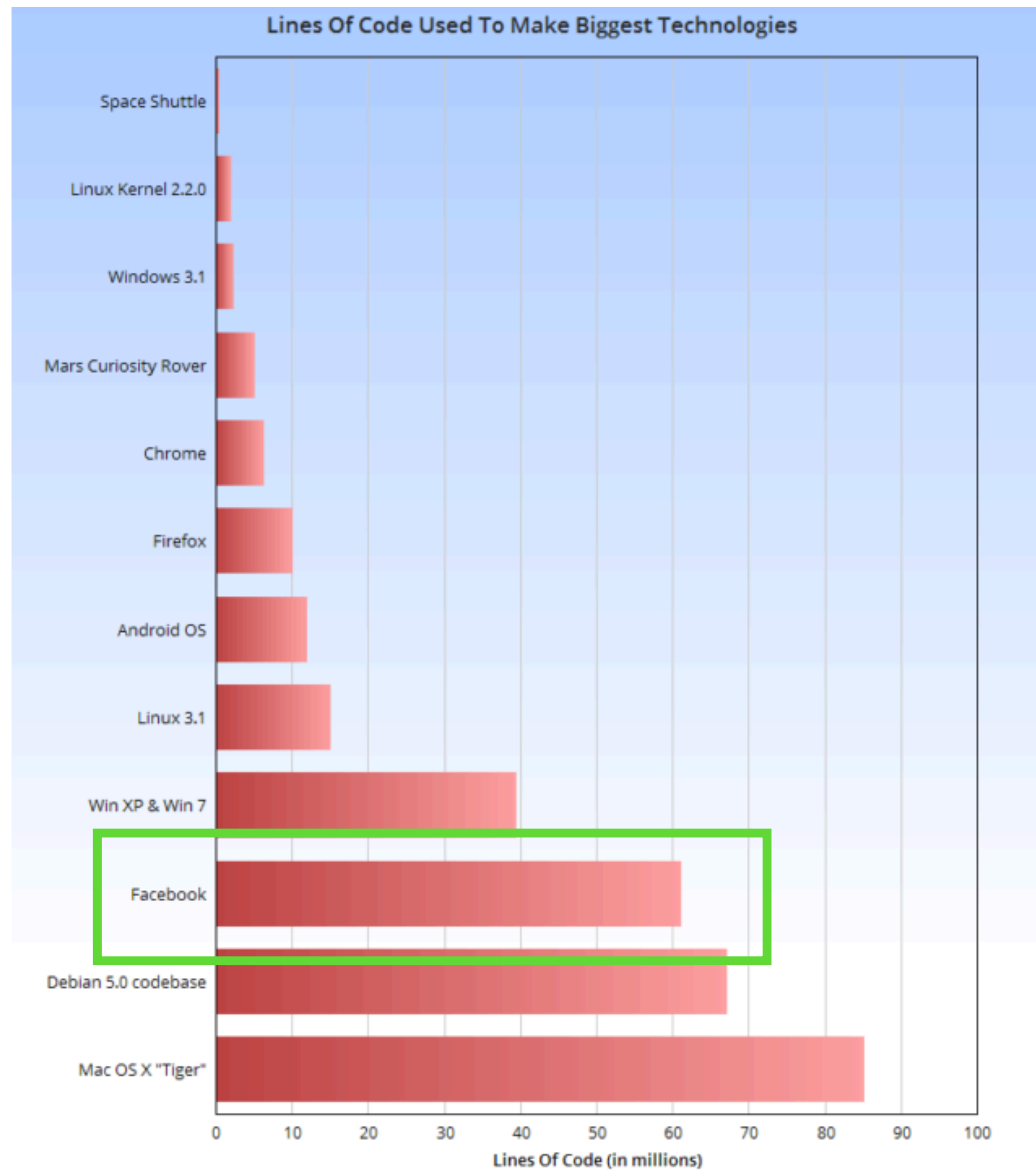
- Esempio: Facebook
- 60M LoCs in un'unica funzione main...



SOURCE: NASA, QUORA, WIKIPEDIA  
INFORMATION IS BEAUTIFUL



- Esempio: Facebook
- 60M LoCs in un'unica funzione `main...`
- Difficile da debuggare

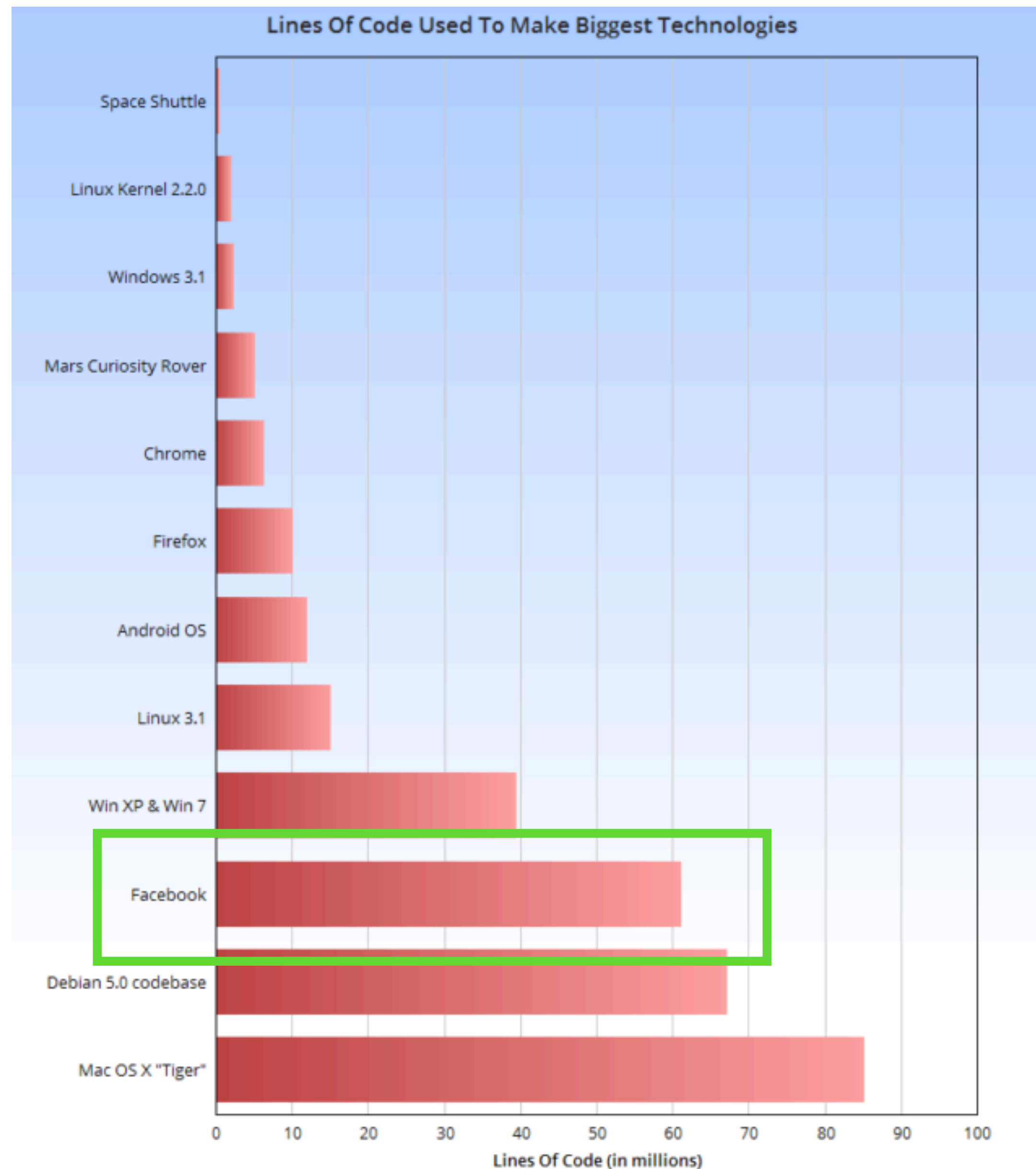


SOURCE: NASA, QUORA, WIKIPEDIA  
INFORMATION IS BEAUTIFUL



- Esempio: Facebook
- 60M LoCs in un'unica funzione `main...`
  - Difficile da debuggare
  - Non mantenibile

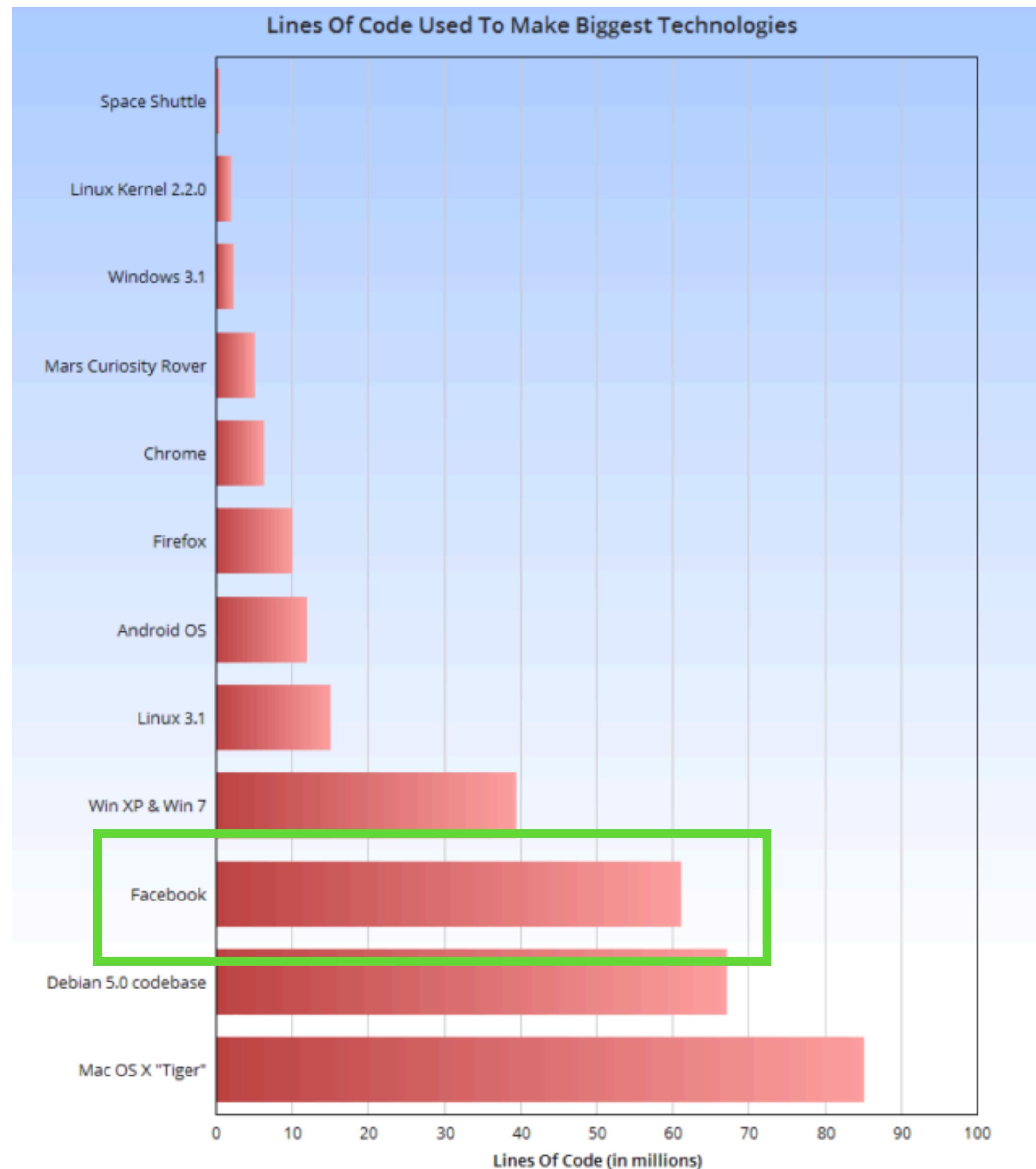




SOURCE: NASA, QUORA, WIKIPEDIA  
INFORMATION IS BEAUTIFUL



- Esempio: Facebook
- 60M LoCs in un'unica funzione `main...`
  - Difficile da debuggare
  - Non mantenibile
  - Non riutilizzabile



SOURCE: NASA, QUORA, WIKIPEDIA  
INFORMATION IS BEAUTIFUL



- Esempio: Facebook
- 60M LoCs in un'unica funzione `main...`
  - Difficile da debuggare
  - Non mantenibile
  - Non riutilizzabile
  - Non modularizzabile

# Programmazione modulare

- Dividere il programma in *sottomoduli*, **funzioni**, che risolvono sotto problemi specifici

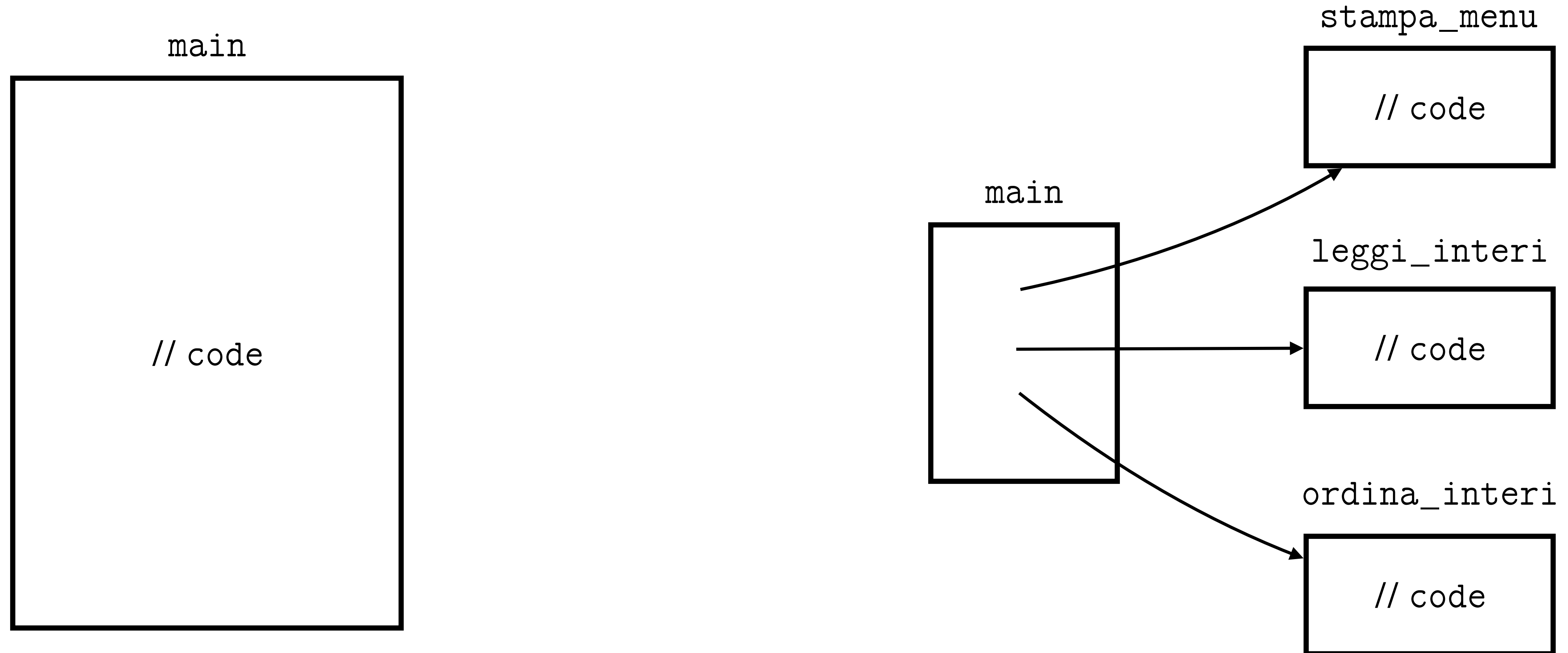
main



// code

# Programmazione modulare

- Dividere il programma in *sottomoduli*, **funzioni**, che risolvono sotto problemi specifici



# Funzioni

## Esempio

- Problema: leggere una lista di interi e stampare a video la lista di interi letti ordinata in ordine crescente

main

Codice per  
ottenere la lista  
di interi

...

Codice per  
ordinare la lista

...

Codice per  
visualizzare la  
lista

# Funzioni

## Esempio

- Problema: leggere una lista di interi e stampare a video la lista di interi letti ordinata in ordine crescente

main

Codice per  
ottenere la lista  
di interi

...

Codice per  
ordinare la lista

...

Codice per  
visualizzare la  
lista

leggi\_interi

Codice per  
ottenere la lista  
di interi

ordina\_interi

Codice per  
ordinare la lista

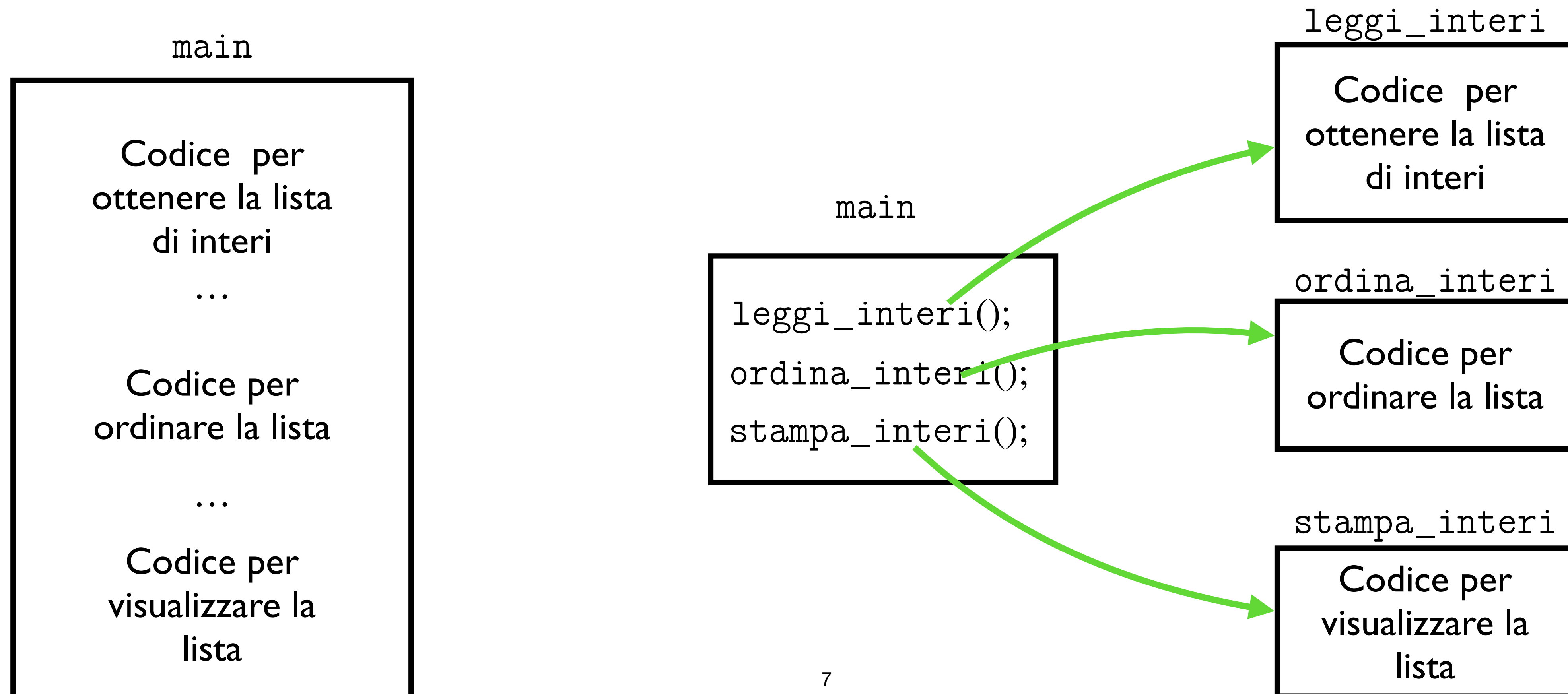
stampa\_interi

Codice per  
visualizzare la  
lista

# Funzioni

## Esempio

- Problema: leggere una lista di interi e stampare a video la lista di interi letti ordinata in ordine crescente



# Funzioni

## Esempio

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

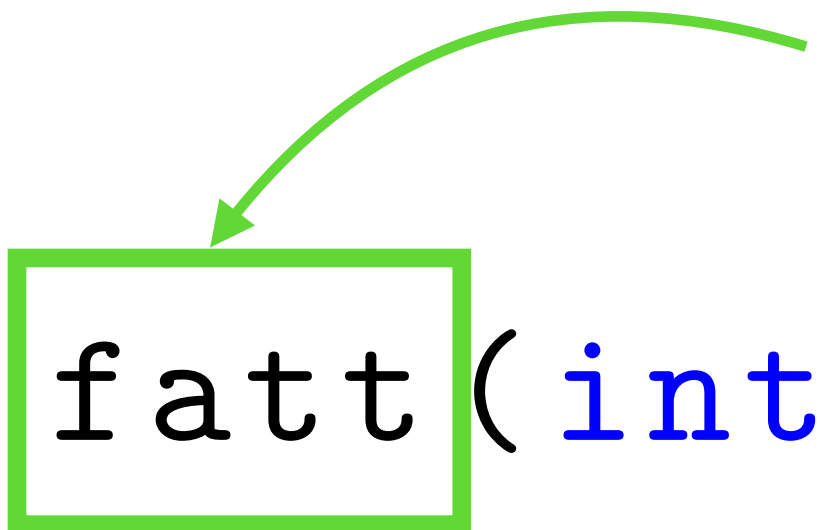


# Funzioni

## Esempio

Nome della funzione (è un identificatore)

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```



# Funzioni

## Esempio

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

Corpo della funzione



# Funzioni

## Esempio

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

Tipo di ritorno della funzione

# Funzioni

## Esempio

Argomenti (**parametri formali**) della funzione  
separati da virgola

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

# Funzioni

## Esempio

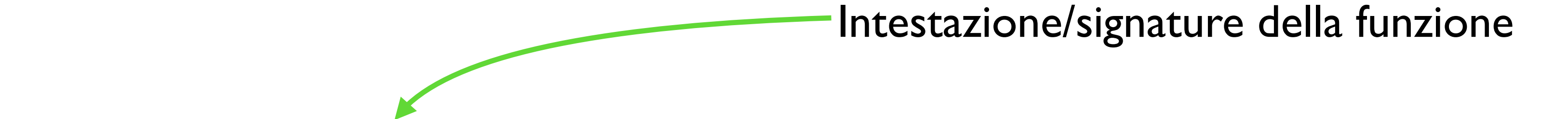
```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

Valore di ritorno della funzione (compatibile col tipo di ritorno)

# Funzioni

## Esempio

Intestazione/signature della funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

# Funzioni

## Chiamata a funzione

- Per mandare in esecuzione una funzione è necessario chiamarla (nella funzione `main` o un'altra funzione)

# Funzioni

## Chiamata a funzione

- Per mandare in esecuzione una funzione è necessario chiamarla (nella funzione `main` o un'altra funzione)

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

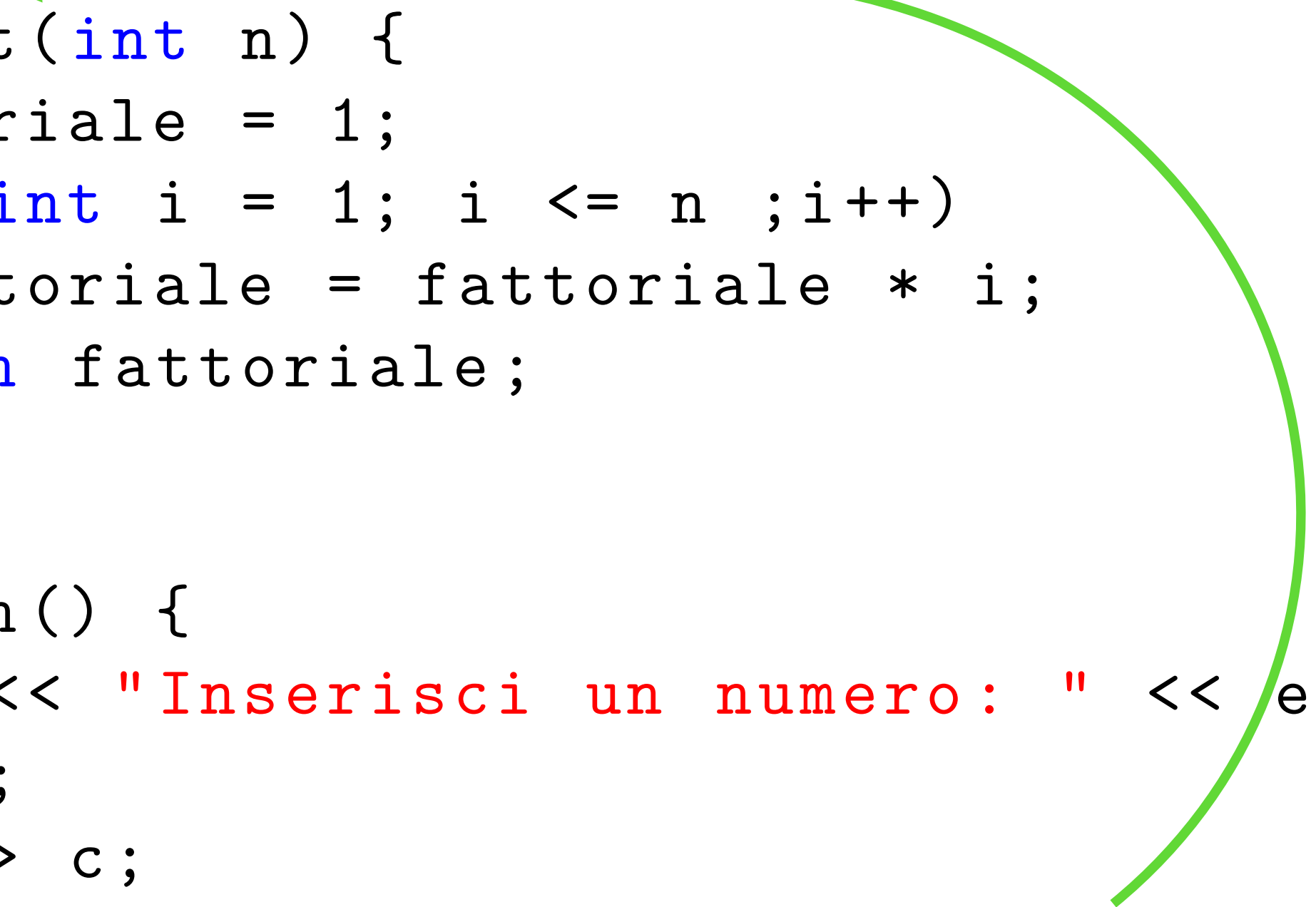
```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```



# Funzioni

## Chiamata a funzione

- Per mandare in esecuzione una funzione è necessario chiamarla (nella funzione `main` o un'altra funzione)

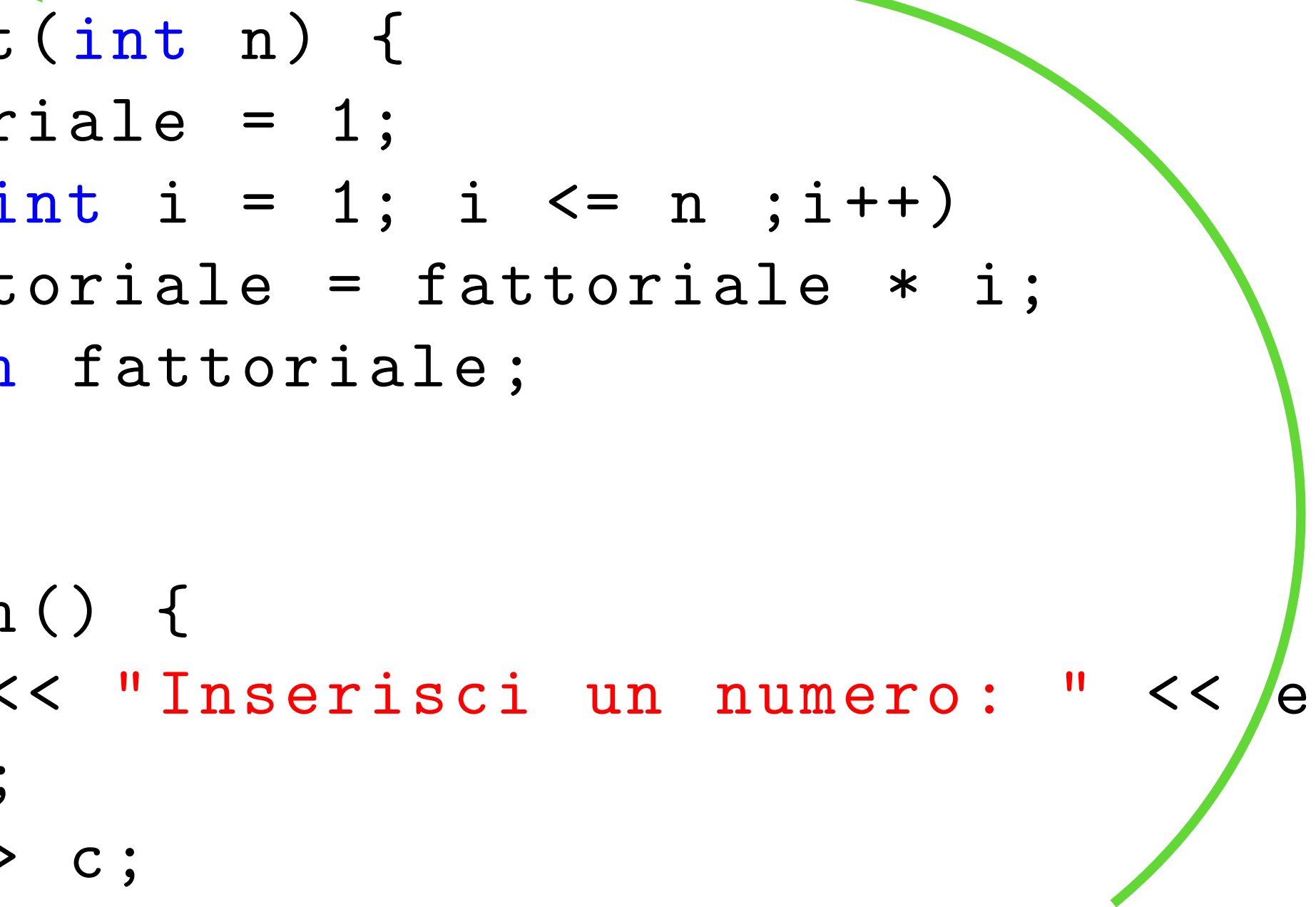


```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione

- Per mandare in esecuzione una funzione è necessario chiamarla (nella funzione `main` o un'altra funzione)



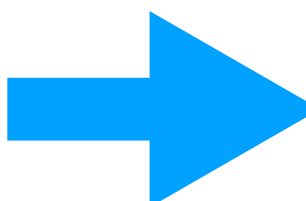
```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

La funzione che chiama un'altra funzione è detta **funzione chiamante**, la funzione controllata viene detta **funzione chiamata**

# Funzioni

## Chiamata a funzione

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

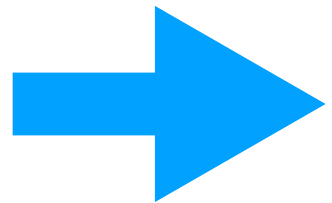


```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

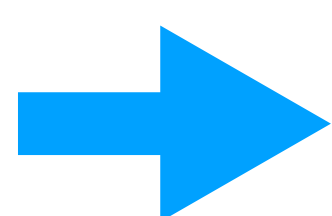


```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

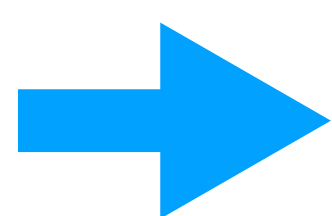


```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

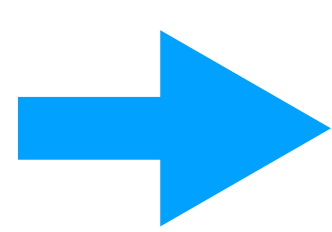


```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione

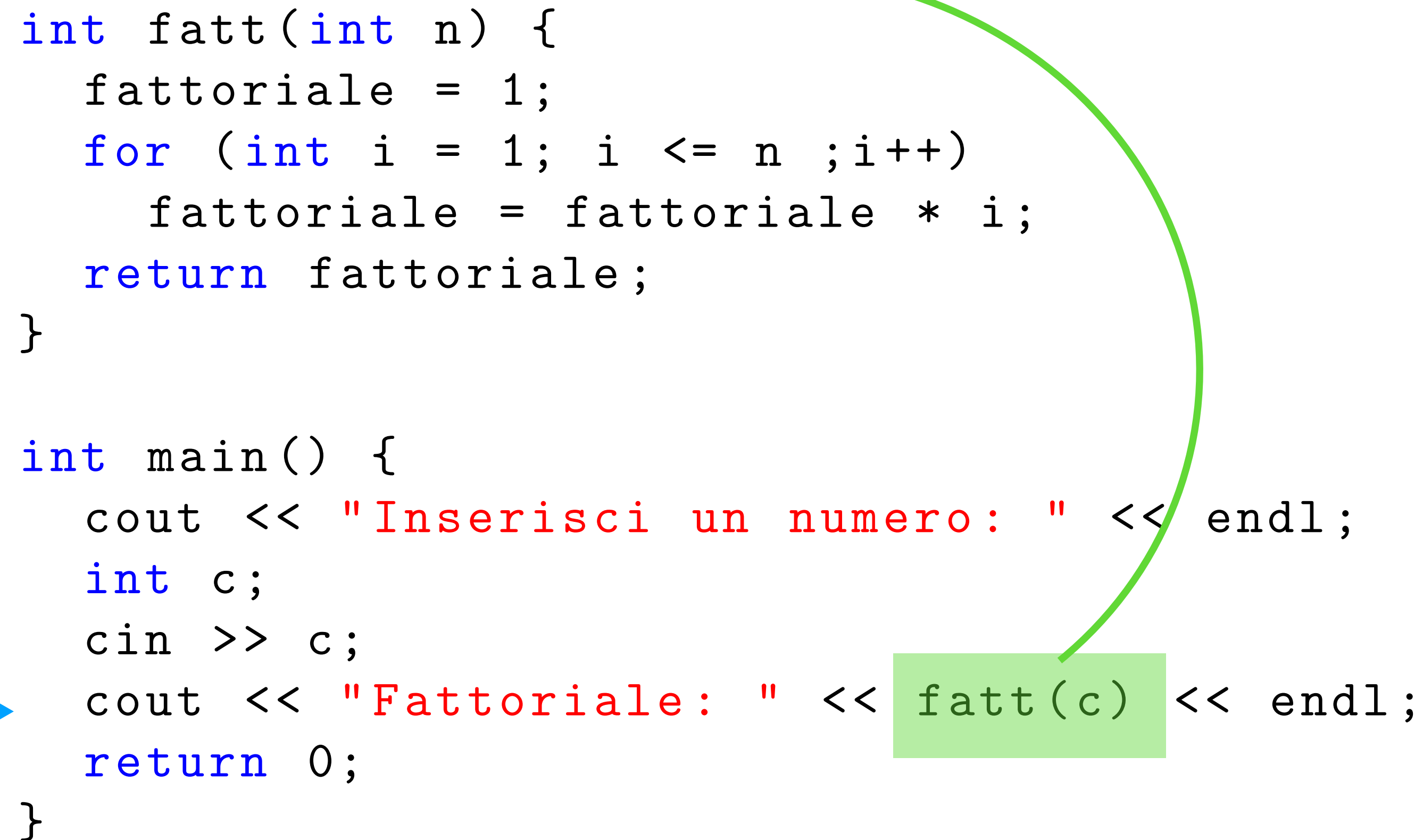
```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```



```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione



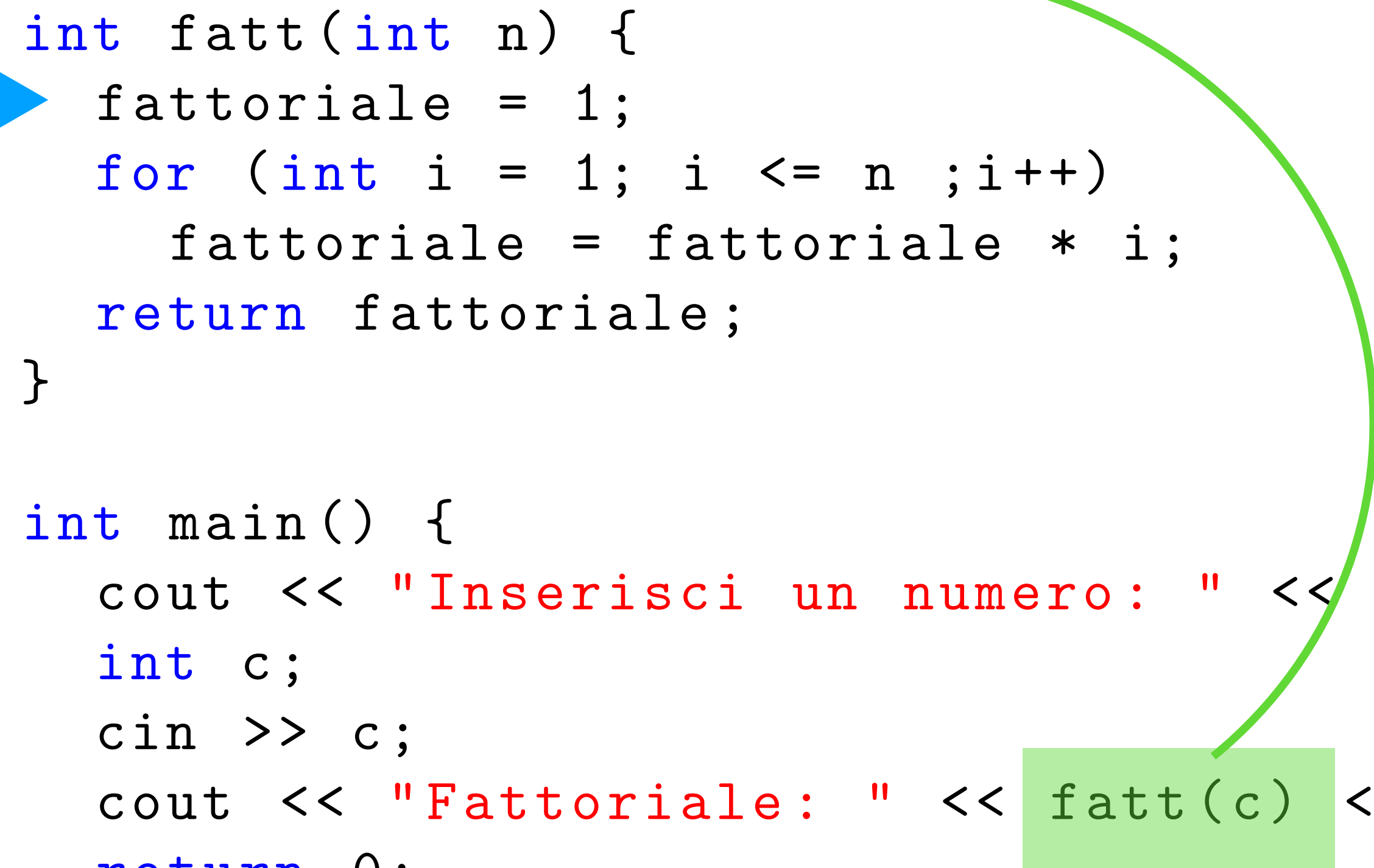
```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

Chiamata a funzione: viene copiato il valore di c (**parametro attuale**) in n (**parametro formale**) e il controllo passa alla funzione fatt



# Funzioni

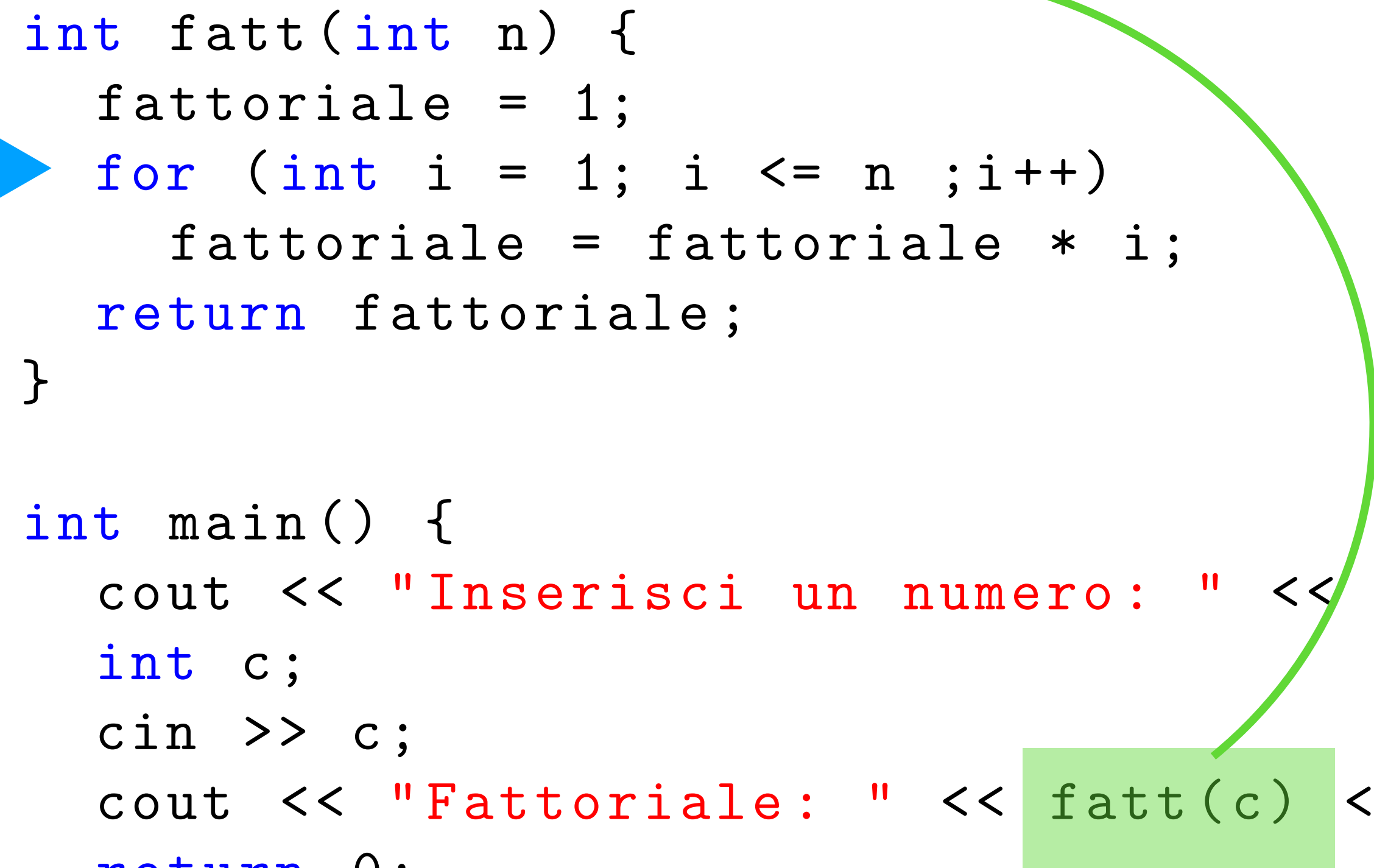
## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

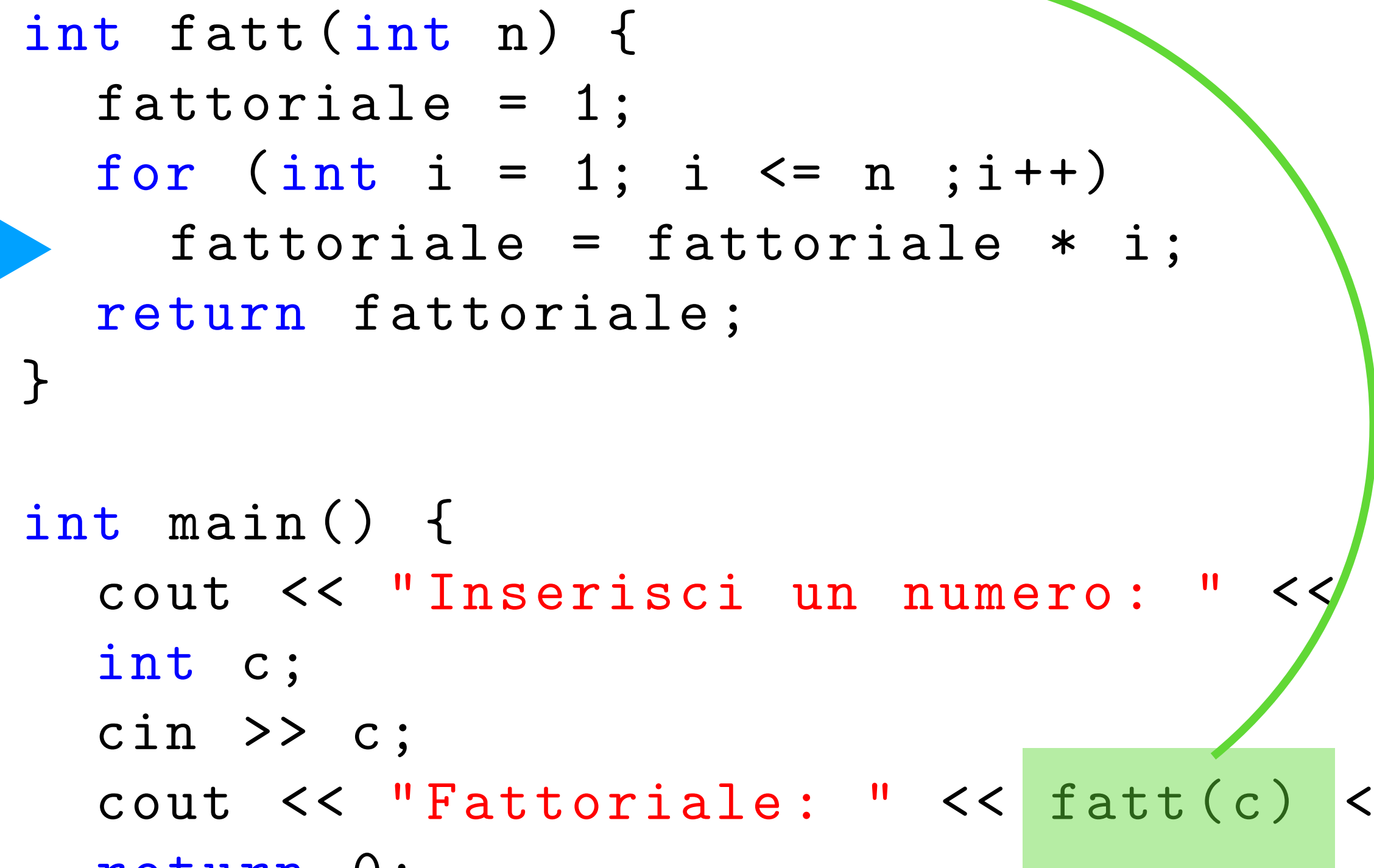
## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

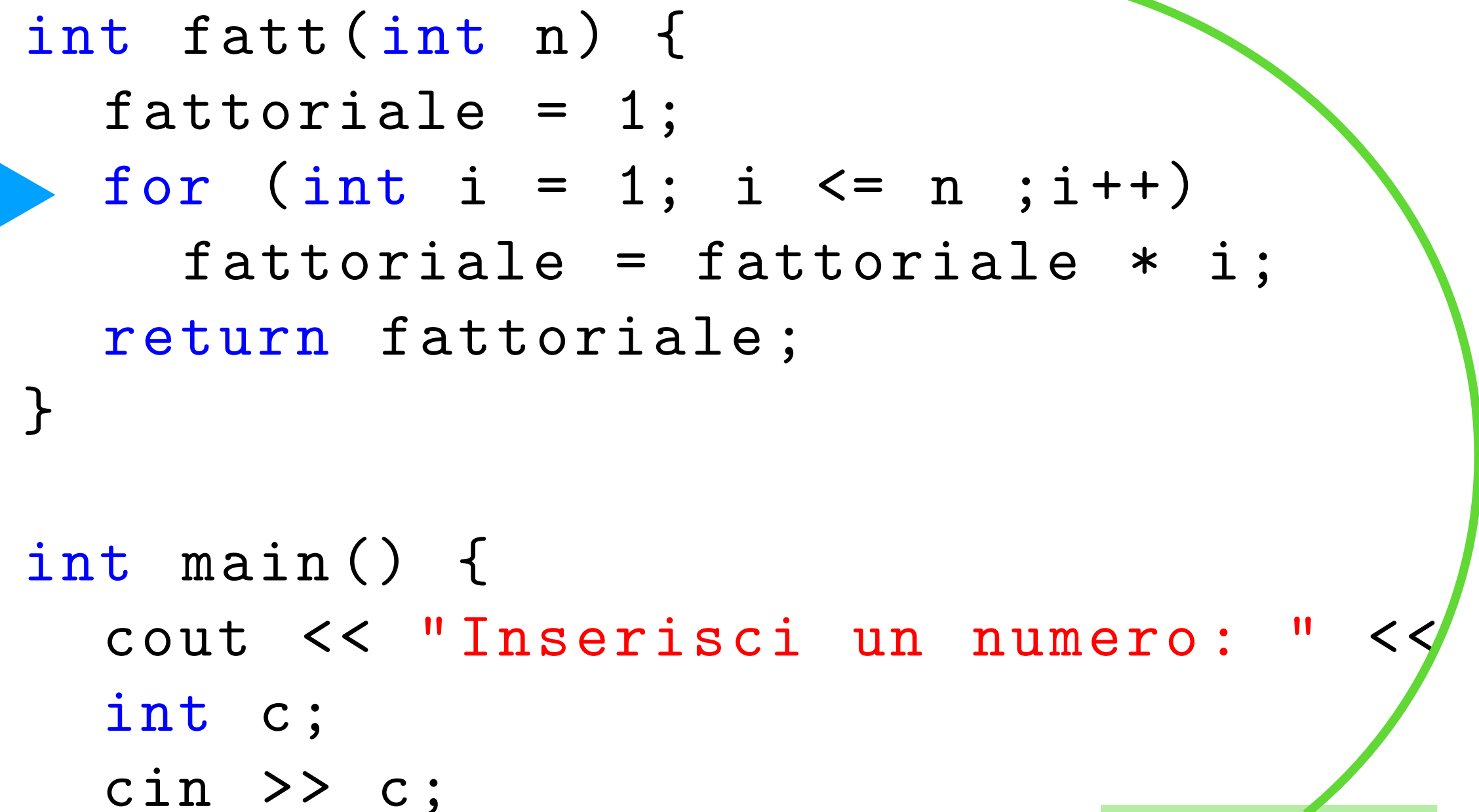
## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

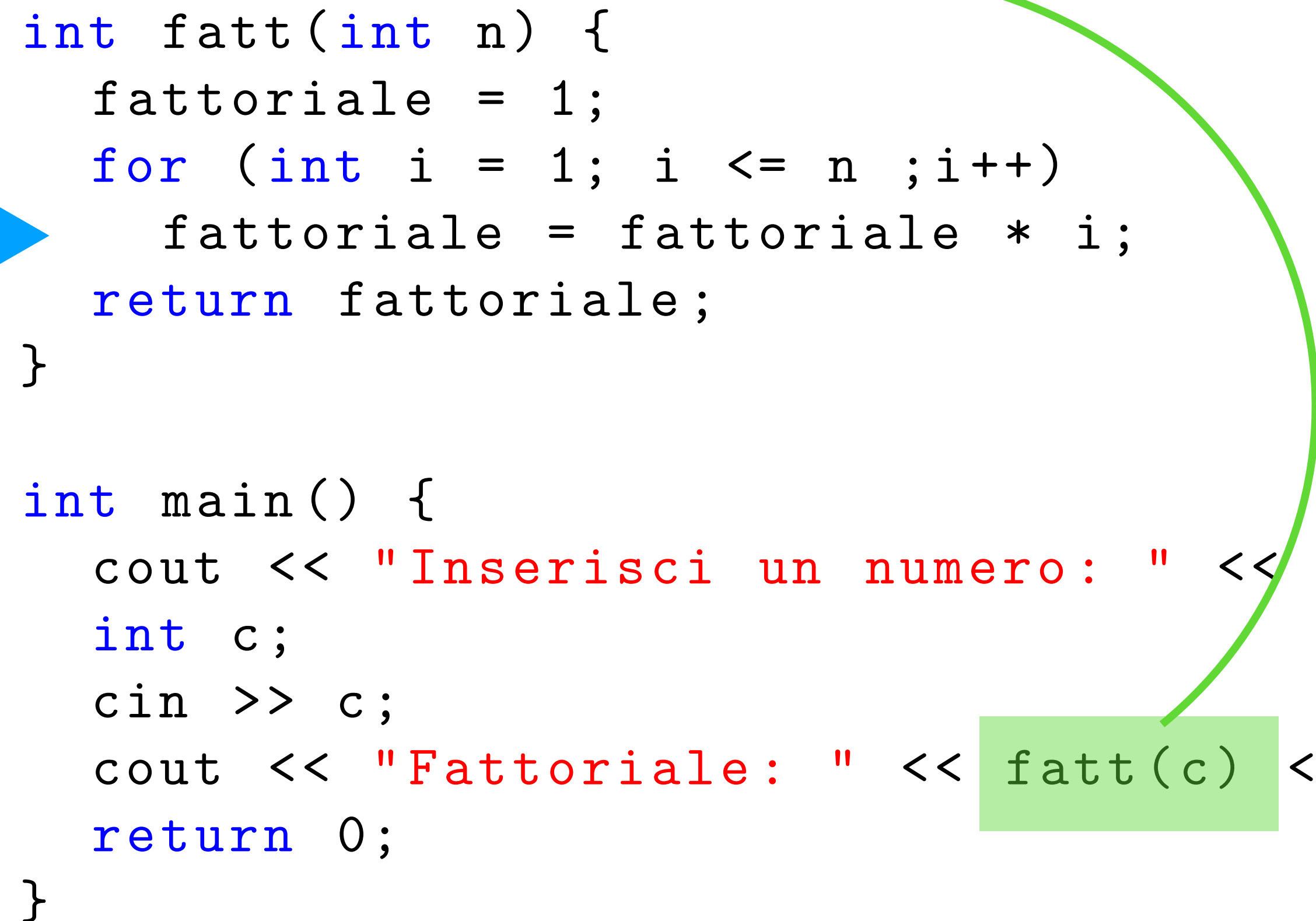
## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

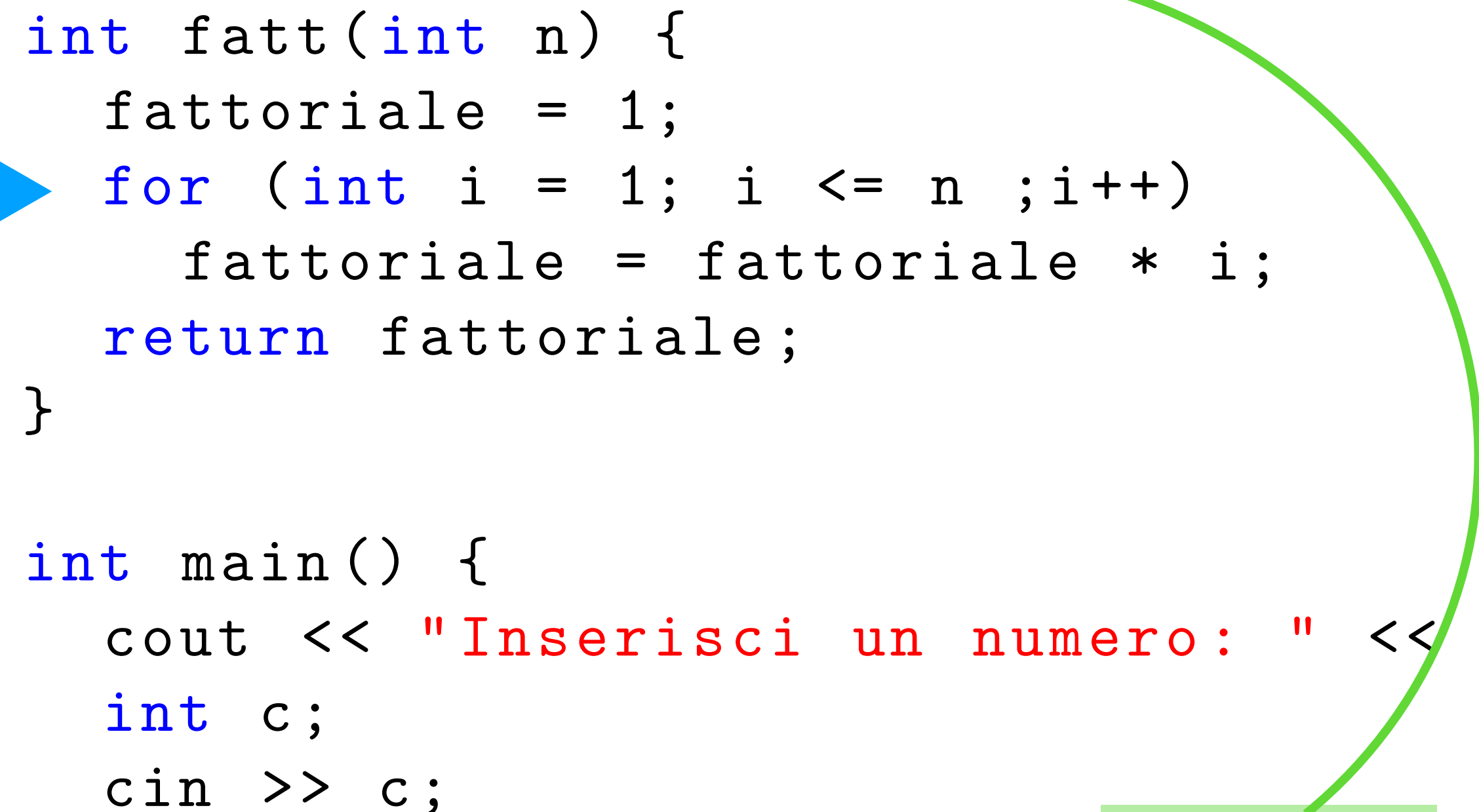
## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

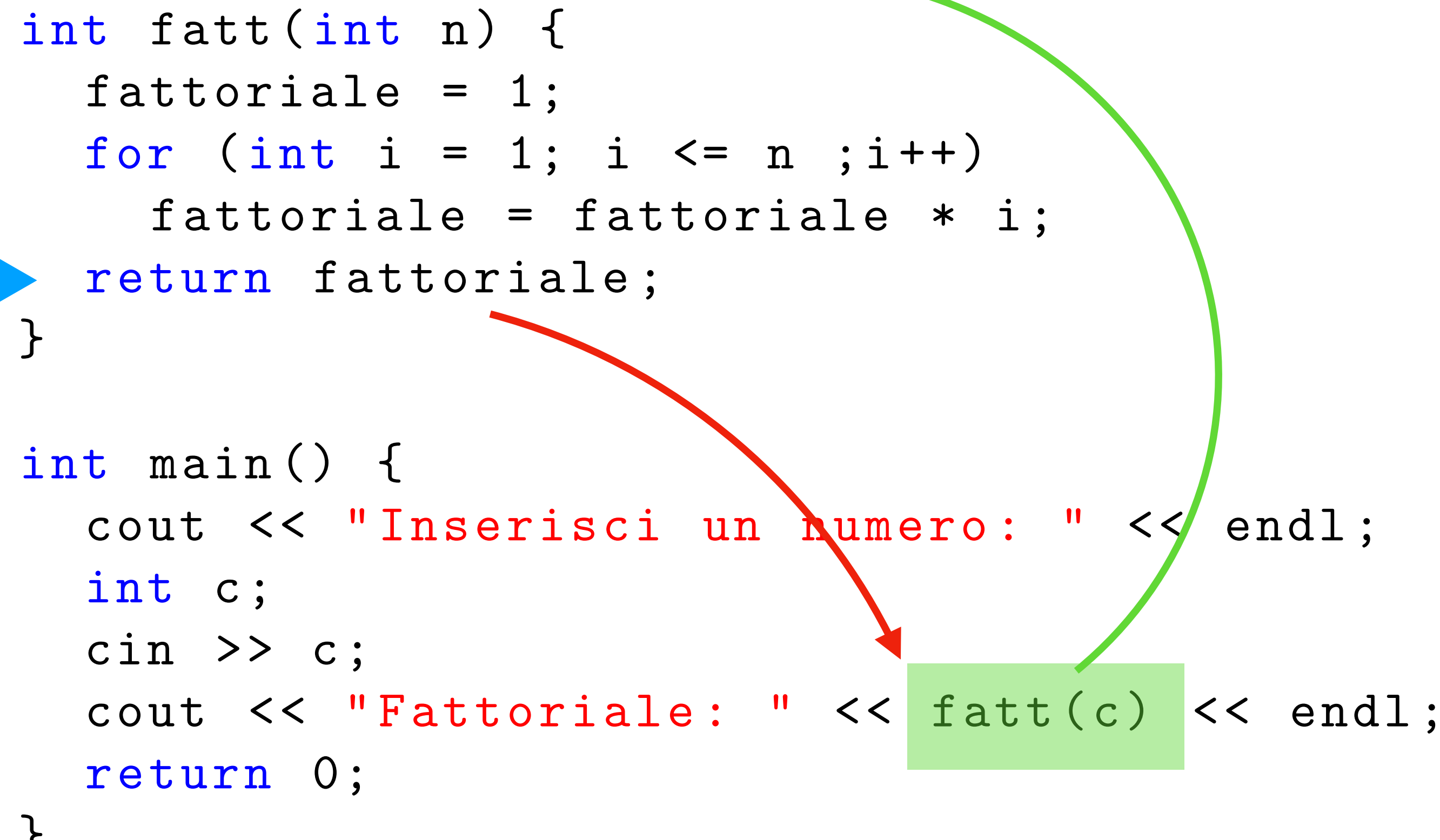
## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

# Funzioni

## Chiamata a funzione



```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}  
  
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```

The diagram illustrates the execution flow between two functions. A blue arrow points to the `return fattoriale;` statement in the `fatt` function. A red arrow originates from the `fatt(c)` call within the `main` function and points to the `return` statement in `fatt`. A green arrow originates from the `fatt(c)` call and points back to the `main` function, indicating the return of control and the value of the factorial.

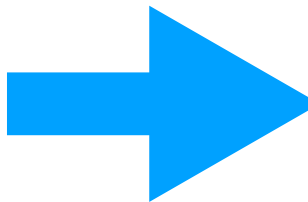
return: il controllo torna alla  
funzione chiamante (main) e gli  
viene ritornato il valore di  
fattoriale

# Funzioni

## Chiamata a funzione

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```



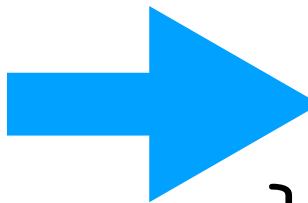


# Funzioni

## Chiamata a funzione

```
int fatt(int n) {  
    fattoriale = 1;  
    for (int i = 1; i <= n ;i++)  
        fattoriale = fattoriale * i;  
    return fattoriale;  
}
```

```
int main() {  
    cout << "Inserisci un numero: " << endl;  
    int c;  
    cin >> c;  
    cout << "Fattoriale: " << fatt(c) << endl;  
    return 0;  
}
```





# Funzioni

## Dichiarazione di funzione

$$\begin{array}{c} t \text{ id}(t_1 \text{ id}_1, t_2 \text{ id}_2, \dots, t_n \text{ id}_n) \{ \\ \quad \textit{stmts} \\ \} \end{array}$$

- $t$ : tipo della funzione, cioè il tipo di risultato ritornato dalla funzione









# Funzioni

## Chiamata di funzione

$$\text{id}(e_1, e_2, \dots, e_n)$$

La chiamata a funzione è un'espressione, dove:

- $\text{id}$ : nome di una funzione
- $e_1, e_2, \dots, e_n$ : espressioni dette **parametri attuali**

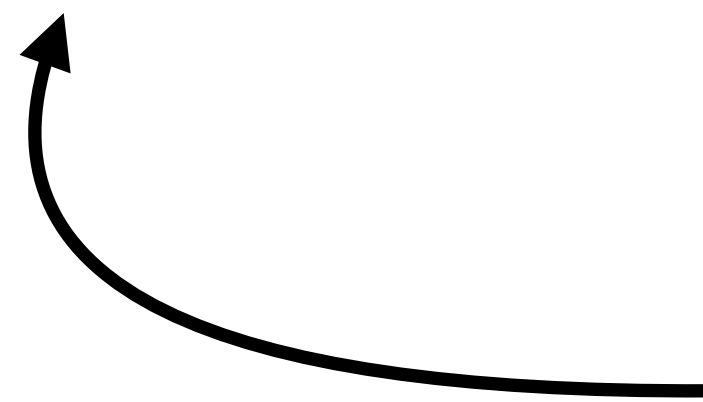


# Funzioni

## Chiamata a funzione

$t \quad id(t_1 \ id_1, \ t_2 \ id_2, \ \dots, \ t_n \ id_n) \ \{$   
 $\quad \quad \quad \textit{stmts}$   
 $\}$

$id(e_1, e_2, \dots, e_n)$



Valutazione dei parametri attuali

$e_1 \rightsquigarrow v_1 \quad e_2 \rightsquigarrow v_2 \quad \dots \quad e_n \rightsquigarrow v_n$

# Funzioni

## Chiamata a funzione

$t \quad id(t_1 \quad id_1, \quad t_2 \quad id_2, \quad \dots, \quad t_n \quad id_n) \quad \{$   
*stmts*  
 $\}$

$id(e_1, e_2, \dots, e_n)$

$e_1 \rightsquigarrow v_1$

$e_2 \rightsquigarrow v_2$

...

$e_n \rightsquigarrow v_n$

Passaggio dei parametri:  
associazione dei valori  
 $v_1, \dots, v_n$  ai parametri  
formali  $id_1, \dots, id_n$

# Funzioni

## Chiamata a funzione

$t \quad id(t_1 \quad id_1, \quad t_2 \quad id_2, \quad \dots, \quad t_n \quad id_n) \quad \{$   
*stmts*  
 $\}$

$id(e_1, e_2, \dots, e_n)$

$e_1 \rightsquigarrow v_1$

$e_2 \rightsquigarrow v_2$

...

$e_n \rightsquigarrow v_n$

Passaggio dei parametri:  
associazione dei valori  
 $v_1, \dots, v_n$  ai parametri  
formali  $id_1, \dots, id_n$

Come vengono passati i  
parametri?

# Funzioni

## Chiamata a funzione

$$t \quad id(t_1 \quad id_1, \quad t_2 \quad id_2, \quad \dots, \quad t_n \quad id_n) \quad \{$$

*stmts*

$$\}$$

Esecuzione del corpo della  
funzione *id*

$$id(e_1, e_2, \dots, e_n)$$
$$e_1 \rightsquigarrow v_1 \quad e_2 \rightsquigarrow v_2 \quad \dots \quad e_n \rightsquigarrow v_n$$

# Funzioni

## Chiamata a funzione

```
t  id(t1 id1, t2 id2, ..., tn idn) {  
    stmts  
}
```

Valore della chiamata a funzione: valore  
dell'espressione ritornata tramite lo  
statement return eseguito

$\text{id}(e_1, e_2, \dots, e_n)$

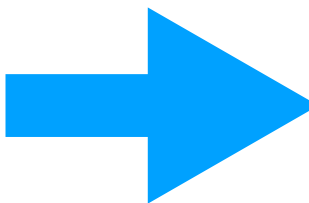
$e_1 \rightsquigarrow v_1 \quad e_2 \rightsquigarrow v_2 \quad \dots \quad e_n \rightsquigarrow v_n$

# Funzioni

## Chiamata a funzione

$$\begin{array}{l} t \quad id(t_1 \ id_1, \ t_2 \ id_2, \ \dots, \ t_n \ id_n) \ \{ \\ \quad \quad \quad \textit{stmts} \\ \} \end{array}$$

$id(e_1, e_2, \dots, e_n)$



Terminata la valutazione della chiamata di  
funzione, il controllo ritorna al programma  
chiamante

# Funzioni

Esercizio - Calcolatrice (quadrato, sommatoria, fattoriale)

# Funzioni

## Funzioni senza risultato esplicito

```
struct persona {  
    char nome[32];  
    char cognome[32];  
    int eta;  
}
```

- Problema: scrivere una funzione che presa in input una persona stampi i suoi dati con un opportuno formato di stampa