
POLITECNICO
DI MILANO
INFORMATICA
L’uso delle librerie per un programma completo in C++

DICHIARAZIONE

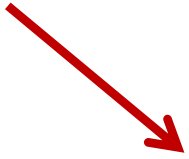
1. Allocare la memoria necessaria
2. Controllare eventuali incongruenze

fra le proprietà delle variabili e il loro uso

DICHIARAZIONE

1 Allocare la memoria necessaria

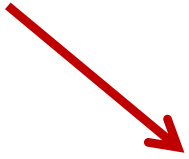
2 Controllare eventuali incongruenze



Strategie di correzione

DICHIARAZIONE

1. Allocare la memoria necessaria
2. Controllare eventuali incongruenze



Strategie di correzione

Pascal, Fortran, C, C++

FATTORIZZAZIONE

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  leggi base;
  leggi esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

VIRGOLA ,

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  leggi base;
  leggi esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

Un linguaggio è tanto meglio definito quanto più costante è la sua sintassi


```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  leggi base;
  leggi esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

LEGGERE UN VALORE

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  leggi base;
  leggi esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

C++

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  leggi base;
  leggi esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

Variabile che può contenere un numero illimitato di caratteri

OGGETTO

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```


```
int potParziale, p
leggi base;
leggi esponente;
potParziale = 1;
```

"abcd0123"

01010010


```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

25^3

2

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

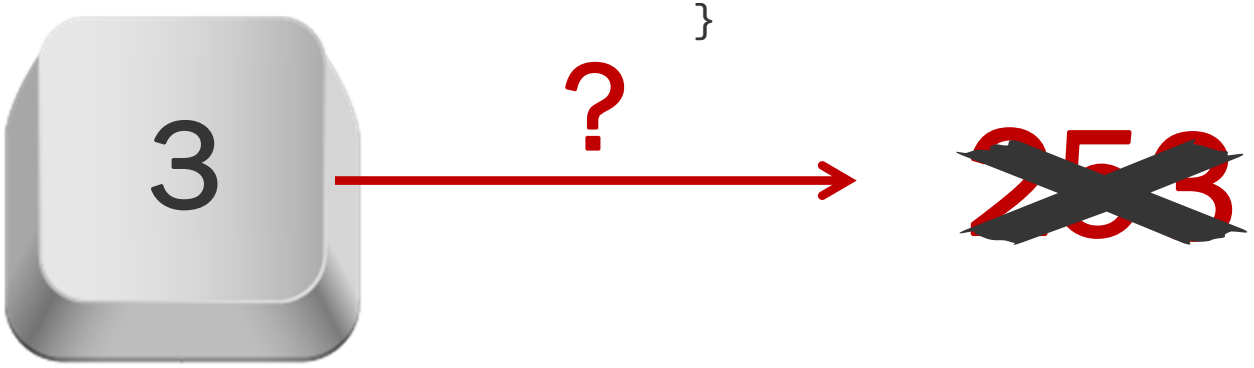
25^3

25

SEPARATORE

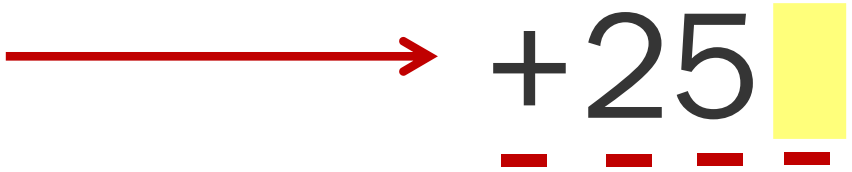
```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

25^3




```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

25^3



```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

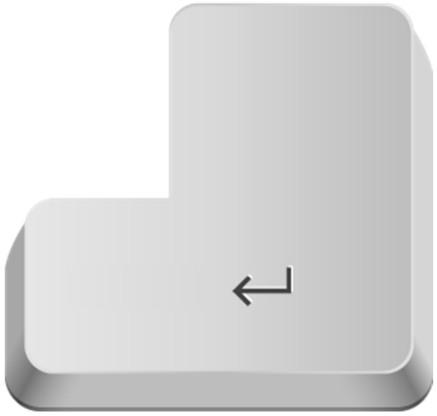
25^3

25 3

INVIO - RETURN

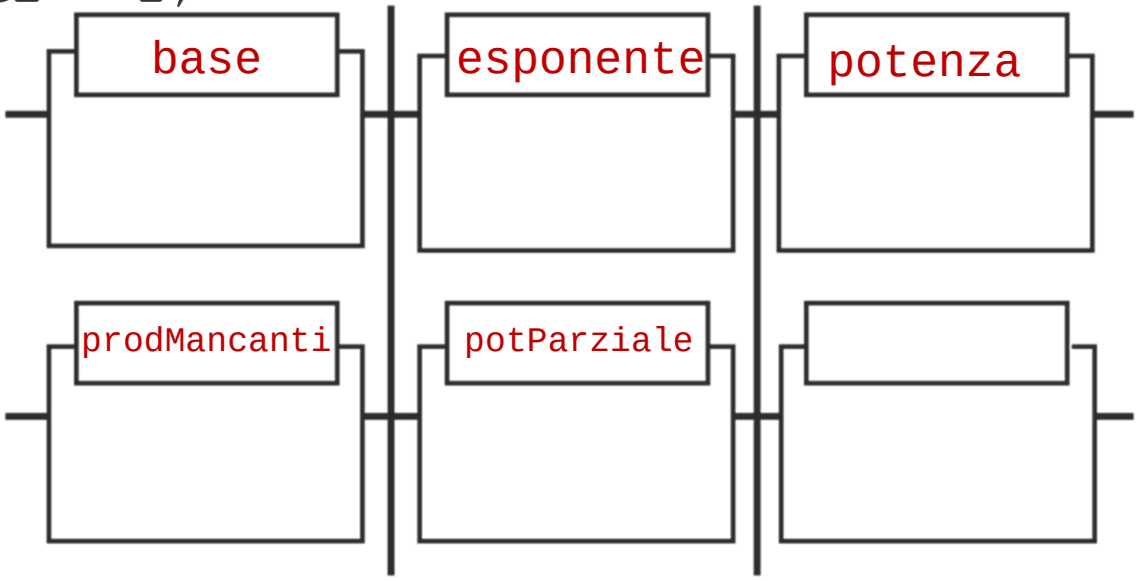
```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

25^3



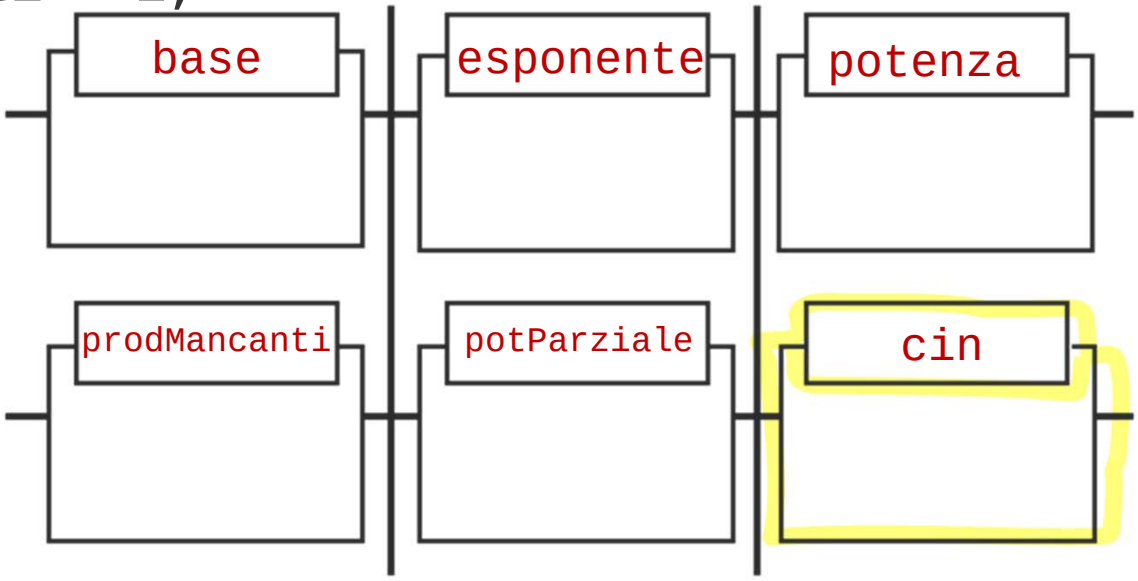
25 3 ↵

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

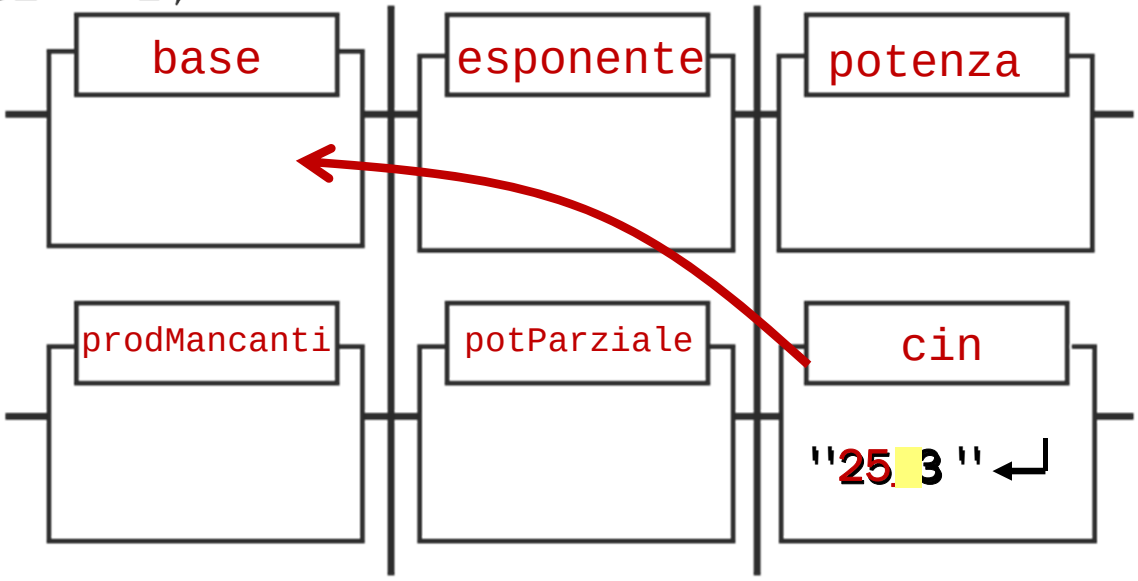


```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

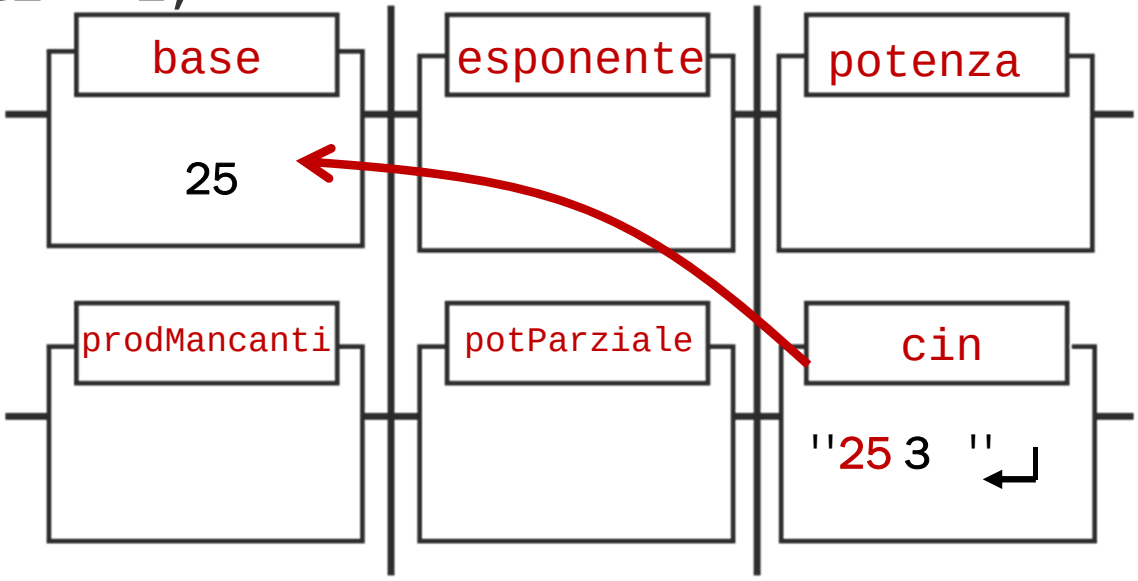
>> ESTRAZIONE



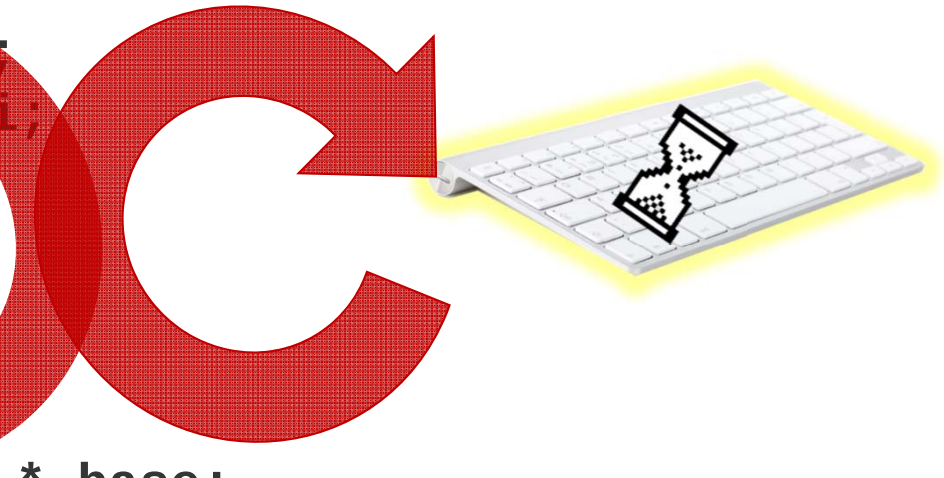
```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

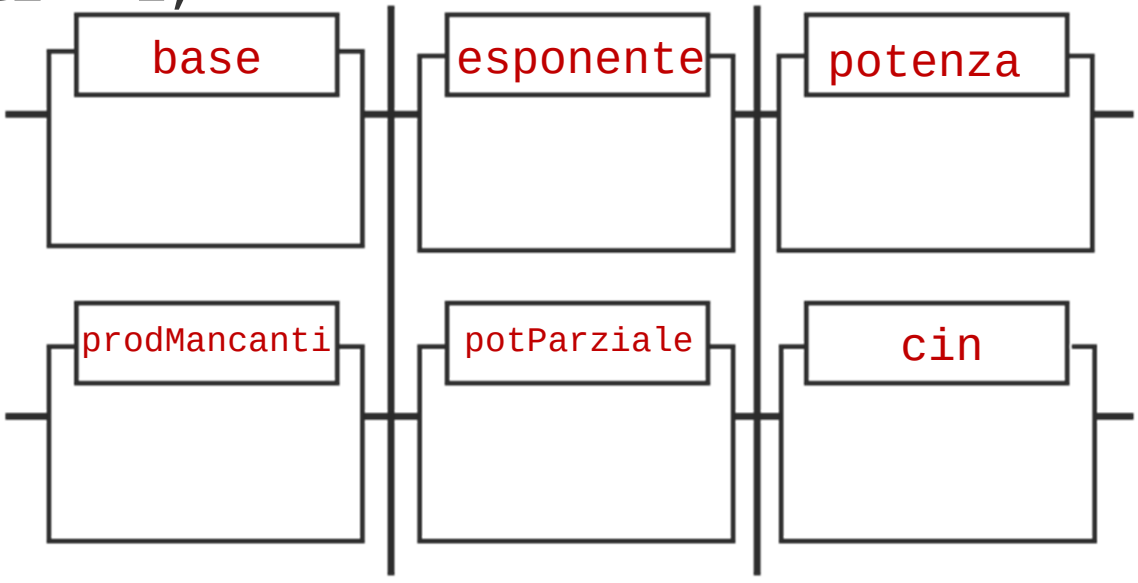


```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```



```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

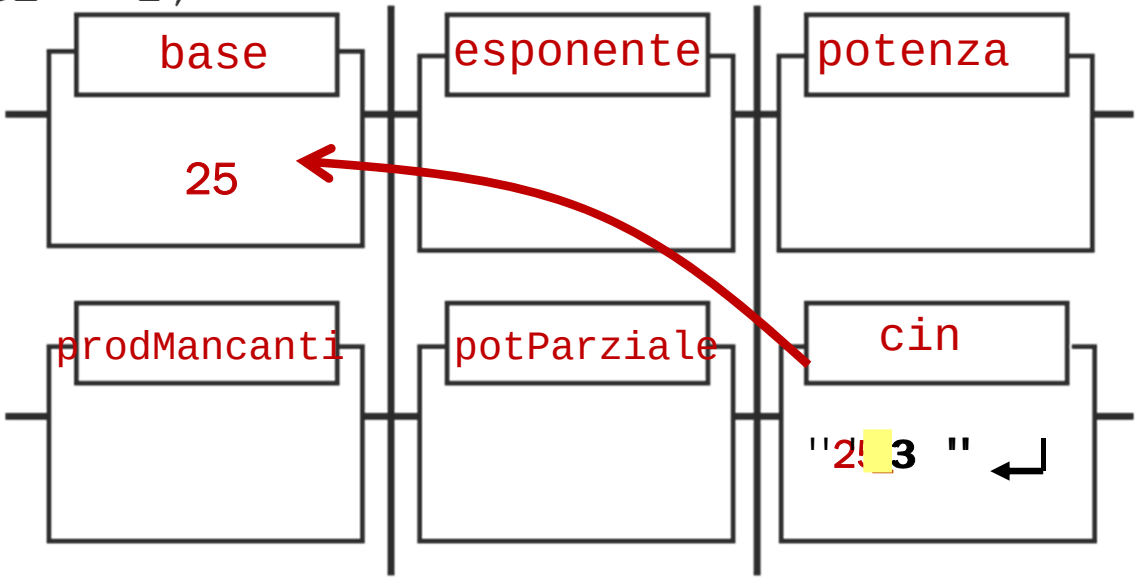




```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

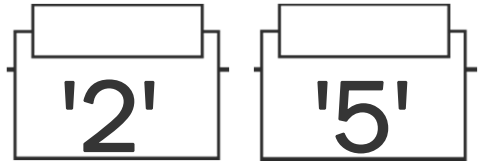
GUIDARE L'UTENTE

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```



TRASFORMAZIONE

```
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
```

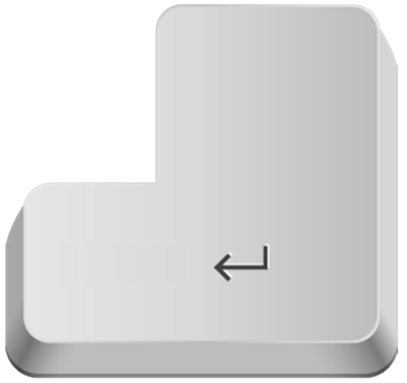


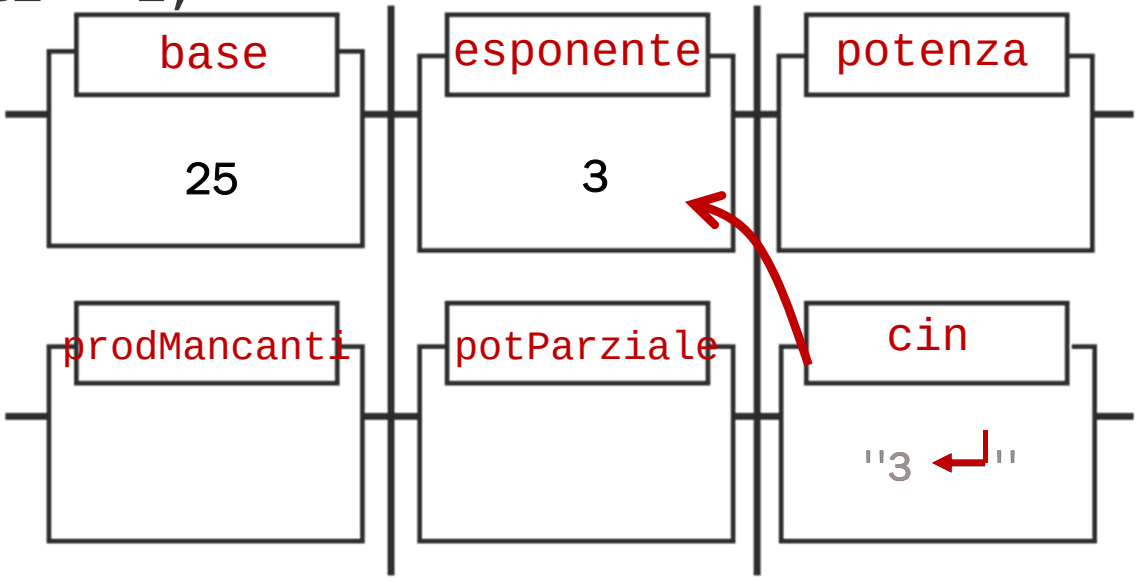
25

Stringa di caratteri

Numero intero

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```



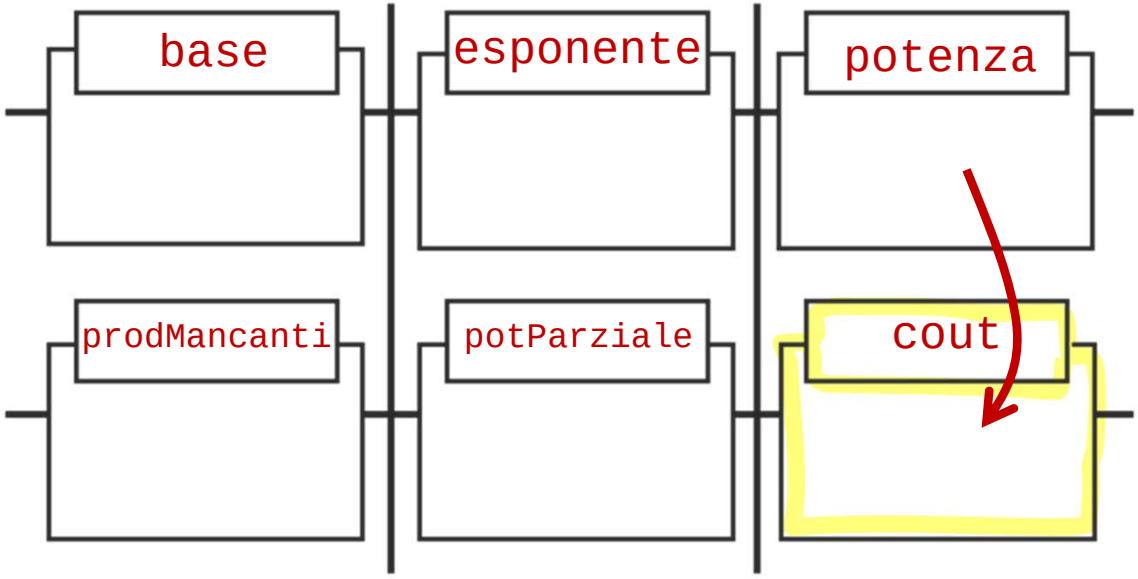


```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  scrivi potenza;
}
```

SCRITTURA

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  count << potenza;
}
```

COUNT



OGGETTO

```
    potenza = potParziale;
    cout << potenza;
}
```

potenza

15625

Numero intero

"15625"

Stringa di caratteri

```
    } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

potenza

15625

Numero intero

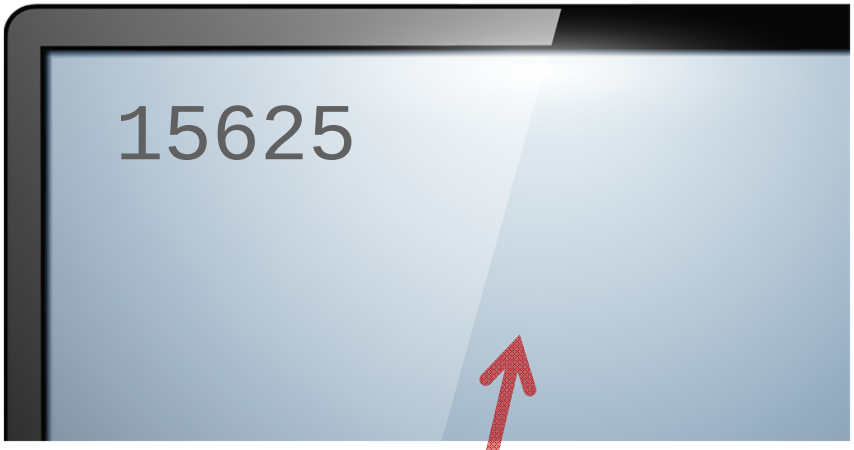
11110100001001

"15625"

Stringa di caratteri

'1' '5' '6' '2' '5'

```
    } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```



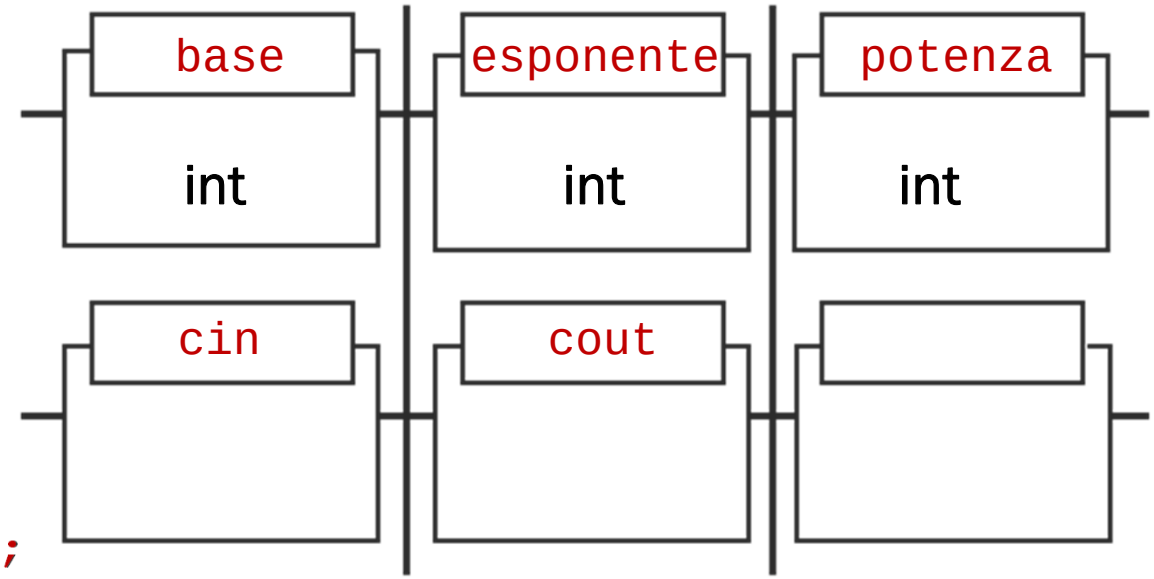
potenza

15625

11110100001001

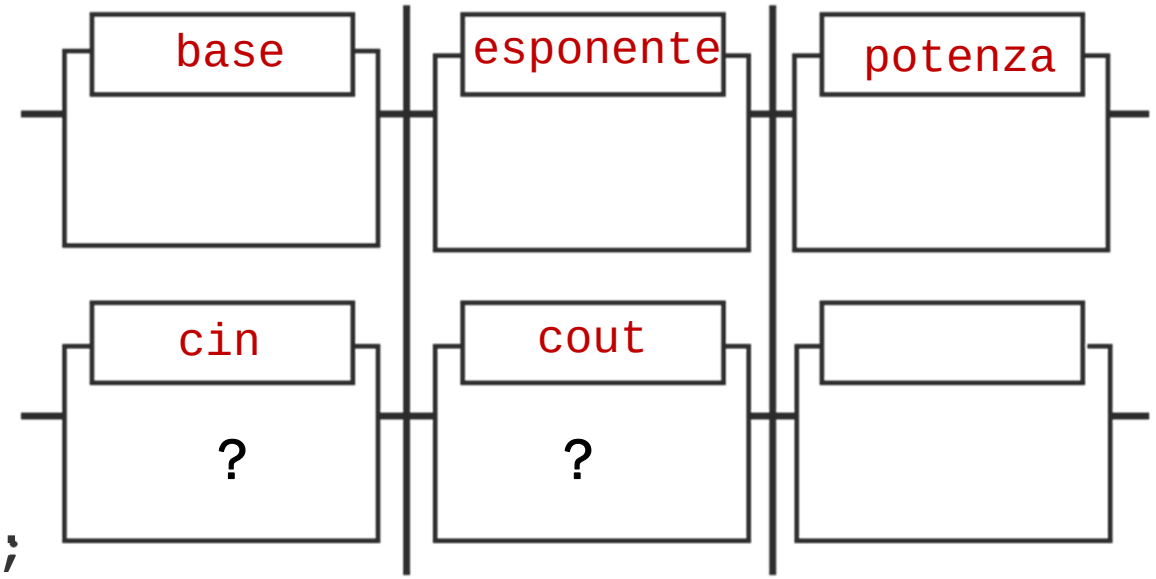
"15625"

'1' '5' '6' '2' '5'




```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

ESTRAZIONE >>
INSERIMENTO <<



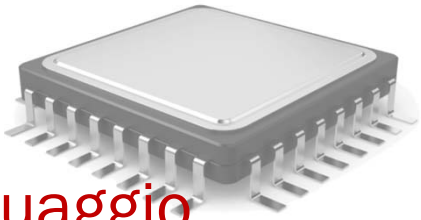

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

ESTRAZIONE

caratteri → valore
int, float, ...

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

SEMANTICA FLESSIBILE



Linguaggio oggetto

Dichiarazione

C++

```
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

LIBRERIE

Input – Output (IO)

<iostream.h>

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

Input – Output (IO)

<iostream.h>

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

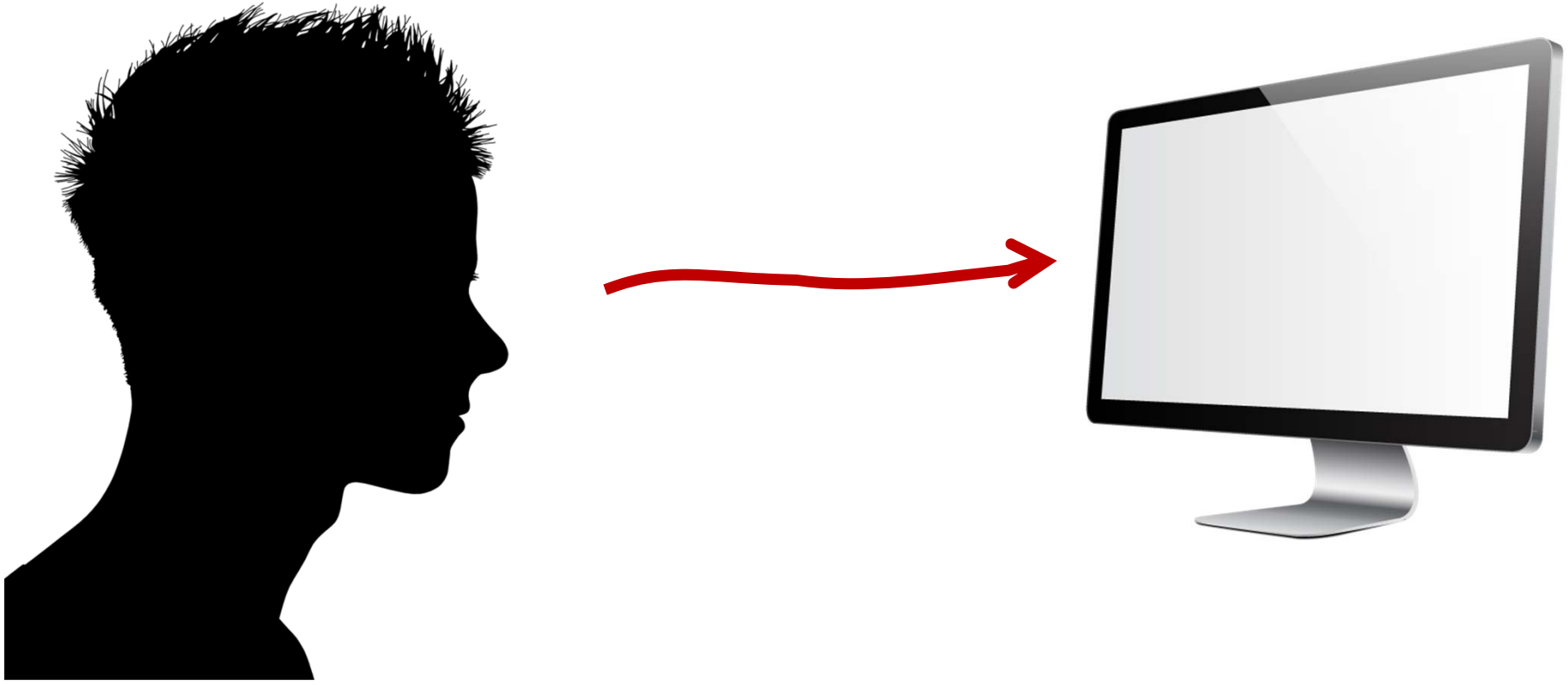
Input – Output (IO)

<iostream.h>

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base;
  cin >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

GUIDARE L'UTENTE



SCHERMO

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi:";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi:
```

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi:
```

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi:
```

FORMATTAZIONE

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi:
```

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati "
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi:
```

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi: 25 3
15625
```

MESSAGGI E VARIABILI

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di" << base << " per"
       << esponente << " vale" << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi: 25 3
L'elevamento a potenza di       25 per       3 vale   15625
```

MESSAGGI E VARIABILI

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di" << base << " per"
       << esponente << " vale" << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi: 25 3
L'elevamento a potenza di       25
                        XXXXXX
```

GIUSTIFICAZIONE A DESTRA

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di" << base << " per"
       << esponente << " vale" << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi: 25 3
L'elevamento a potenza di       25
```

XXXXXXX

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di" << base << " per"
       << esponente << " vale" << potenza;
}
```

XXXX

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi: 25 3
L'elevamento a potenza di       25 per
```

XXXX

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di" << base << " per"
       << esponente << " vale" << potenza;
}
```

```
Calcolo dell'elevamento a potenza di una base intera positiva a un esponente intero positivo.
Fornire i valori per base ed esponente, separati da uno o più spazi: 25 3
L'elevamento a potenza di       25 per       3 vale   15625
```

IL PROGRAMMA IN C++

```
#include <iostream.h>
void main()
{ int base, esponente, potenza;
  int potParziale, prodMancanti;
  cout << "Calcolo dell'elevamento a potenza di una base
intera"
       << " positiva a un esponente intero positivo." << endl;
  cout << "Fornire i valori per base ed esponente, separati"
       << " da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di" << base << " per"
       << esponente << " vale" << potenza;
}
```