



**POLITECNICO
DI MILANO**

INFORMATICA

Espressioni e
strutture di
controllo

Ulteriori critiche al programma

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // o nullo
  // ...
  cout << "Calcolo dell'elevamento a potenza di " << base << " a un esponente " << esponente << endl;
  // ...
  cout << "inserisci la base e l'esponente separati da uno spazio: ";
  cin >> base >> esponente;
  potenza = 1;              // = base elevata all'esponente 0
  prodMancanti = esponente;
  do
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  } while (prodMancanti > 0); // volte mancano per finire il ciclo
  // stampa il risultato
  cout << "L'elevamento a potenza di " << base << " per "
    << esponente << " vale" << potenza;
}
```

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;                                // positivo
  int base, potenza;
  int potParziale;
  int prodMancanti;                             // positivo o nullo
                                              // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << a un esponente intero positivo." << endl;
                                              // legge base ed esponente

  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;                             // = base elevata all'esponente 0
  prodMancanti = esponente;
  do
  { potParziale = potParziale * base;           // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1;             // moltiplicazione per base e conta quante
  } while (prodMancanti > 0);                   // volte mancano per finire il ciclo
  potenza = potParziale;

                                              // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

POTENZA

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;                // positivo
  int base, potenza;
  int prodMancanti;             // positivo o nullo
                                // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                                // legge base ed esponente

  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;
  potenza = 1;                  // = base elevata all'esponente 0
  prodMancanti = esponente;
  do
  { potenza = potenza * base;    // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  } while (prodMancanti > 0);     // volte mancano per finire il ciclo
                                // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

**E' pericoloso usare una
stessa variabile per due
funzionalità differenti**

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente; // positivo
  int base, potenza;
  int prodMancanti; // positivo o nullo
  // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;
  // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;
  potenza = 1; // = base elevato all'esponente 0
  prodMancanti = esponente;
  do
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  } while (prodMancanti > 0); // volte mancano per finire il ciclo
  // stampa il risultato
  cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```

COMMENTO

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente

  cin >> base >> esponente;

  potenza = 1;             // = base elevata all'esponente 0
  prodMancanti = esponente;

  do
  { potenza = potenza * base;           // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1;     // moltiplicazione per base e conta quante
  } while (prodMancanti > 0);           // volte mancano per finire il ciclo
                                       // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

ALTRE OSSERVAZIONI

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cin >> base >> esponente;
  potenza = 1;
  prodMancanti = esponente;
  do
  { potenza = potenza * base;
    prodMancanti = prodMancanti -
  } while (prodMancanti > 0);

  cout << "L'elevamento a potenza
        << esponente << " vale" <<
}
```

**Il programma non
funziona con
esponente ≤ 0**


```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                          // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                          // legge base ed esponente

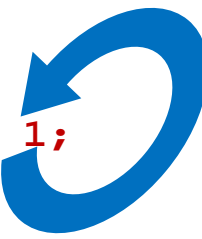
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  potenza = 1;             // = base elevata all'esponente 0
  prodMancanti = esponente;

  do
  { potenza = potenza * base;
    prodMancanti = prodMancanti - 1;
  } while (prodMancanti > 0);

                          // il corpo del ciclo esegue una
                          // moltiplicazione per base e conta quante
                          // volte mancano per finire il ciclo
                          // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```



Almeno un'esecuzione


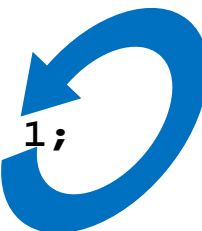

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                          // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                          // legge base ed esponente

  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;
  potenza = 1;             // = base elevata all'esponente 0
  prodMancanti = esponente;

  ???
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  } // volte mancano per finire il ciclo
    // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```



zero o più volte

C++

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                          // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                          // legge base ed esponente

  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  potenza = 1;             // = base elevata all'esponente 0
  prodMancanti = esponente;

  do
  { potenza = potenza * base;           // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1;    // moltiplicazione per base e conta quante
  } while (prodMancanti > 0);           // volte mancano per finire il ciclo
                                      // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

zero o più volte

C++

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente

  cin >> base >> esponente;

  potenza = 1;             // = base elevata all'esponente 0
  prodMancanti = esponente;

  ???
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  } // volte mancano per finire il ciclo
    // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

WHILE (CONDIZIONE) { ... }

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente

  cin >> base >> esponente;

  potenza = 1;              // = base elevata all'esponente 0
  prodMancanti = esponente;

  while (prodMancanti > 0)
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  }                                // volte mancano per finire il ciclo
                                   // stampa il risultato

  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cin >> base >> esponente;
  potenza = 1;              // = base elevata all'esponente 0
  prodMancanti = esponente;
  while (prodMancanti > 0)
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  }                                // volte mancano per finire il ciclo
                                   // stampa il risultato
  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

0^0

valore di potenza
indeterminato

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                             // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                             // legge base ed esponente
  cin >> base >> esponente;
  potenza = 1;              // = base elevata all'esponente 0
  prodMancanti = esponente;
  while (prodMancanti > 0)
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  }                                 // volte mancano per finire il ciclo
                                   // stampa il risultato
  cout << "L'elevamento a potenza di " << base << " per"
        << esponente << " vale" << potenza;
}
```

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori pari a 0
  int prodMancanti;        // positivo o nullo

  // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera
    << "a un esponente intero positivo." << endl;

  // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;
  potenza = 1;              // = base elevata all'esponente 0
  prodMancanti = esponente;

  while (prodMancanti > 0)
  { potenza = potenza * base;
    prodMancanti = prodMancanti - 1;
  }

  // il corpo del ciclo esegue una
  // moltiplicazione per base e conta quante
  // volte mancano per finire il ciclo
  // stampa il risultato
  cout << "L'elevamento a potenza di " << base << " per"
    << esponente << "vale" << potenza;
} }
```

0⁰

valore di potenza indeterminato



ALTERNATIVA

```
/* Elevamento a potenza
 * Esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "base e esponente interi positivi." << endl;
  // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;
  potenza = 1;              // = base elevata all'esponente 0
  prodMancanti = esponente;
  while (prodMancanti > 0)
  { potenza = potenza * base; // il corpo del ciclo esegue una
    prodMancanti = prodMancanti - 1; // moltiplicazione per base e conta quante
  } // volte mancano per finire il ciclo
  // stampa il risultato
  cout << "base " << base << " elevata all'esponente " << esponente << " vale " << potenza;
}
```

**se base e potenza
entrambe nulle**

altrimenti

messaggio

effettua il calcolo

C++

STRUTTURA DI CONTROLLO

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else                                // esponente OK, calcola
  { potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)          // il corpo del ciclo esegue una
    { potenza = potenza * base;       // moltiplicazione per base e conta quante
      prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    }                                // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
          << esponente << " vale" << potenza;
  }
}
```

STRUTTURA DI CONTROLLO

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else                                // esponente OK, calcola
  {
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)          // il corpo del ciclo esegue una
    {
      potenza = potenza * base;      // moltiplicazione per base e conta quante
      prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    }
                                     // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
          << esponente << " vale" << potenza;
  }
}
```

STRUTTURA DI CONTROLLO

```
/* Programma che calcola l'elevamento a potenza
 * di una base intera ad un esponente intero positivo
 */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else                                // esponente OK, calcola
  {  potenza = 1;
     prodMancanti = esponente;
     while (prodMancanti > 0)          // il corpo del ciclo esegue una
     {  potenza = potenza * base;      // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
     }                                // stampa il risultato

     cout << "L'elevamento a potenza di " << base << " per"
           << esponente << " vale" << potenza;
  }
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * data una base e un esponente intero positivo
 */Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else                                // esponente OK, calcola
    { potenza = 1;
      prodMancanti = esponente;
      while (prodMancanti > 0)        // il corpo del ciclo esegue una
      { potenza = potenza * base;    // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
      }                               // stampa il risultato

      cout << "L'elevamento a potenza di " << base << " per"
            << esponente << " vale" << potenza;
    }
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{
    int esponente;           // positivo o nullo
    int base, potenza;       // potenza rappresenta anche i valori parziali
    int prodMancanti;        // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;
                                // legge base ed esponente
    cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
    cin >> base >> esponente;

    if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
        cout << "La potenza non è definita";
    else                                // esponente OK, calcola
    {
        potenza = 1;
        prodMancanti = esponente;
        while (prodMancanti > 0)        // il corpo del ciclo esegue una
        {
            potenza = potenza * base;   // moltiplicazione per base e conta quante
            prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
        }                               // stampa il risultato

        cout << "L'elevamento a potenza di " << base << " per"
             << esponente << " vale" << potenza;
    }
}
```

CONDIZIONE COMPOSTA

(base vale zero ¹
e
esponente vale zero) ²


```
cin >> base >> esponente;
```

```
if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    { potenza = potenza * base; // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```


CONFRONTO ==

(base == 0
e
esponente == 0)



```
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    { potenza = potenza * base; // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
}
```

```
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;

  // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else // esponente OK, calcola
  {
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
      potenza = potenza * base; // moltiplicazione per base e conta quante
      prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    }
    // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
          << esponente << " vale" << potenza;
  }
}
```

(base == 0
e
esponente == 0)

= assegnamento

== confronto

C + +

efficienza

= assegnamento

== confronto

PASCAL

didattica

: = assegnamento

= confronto

C + +

efficienza

= assegnamento

== confronto

!= diverso

PASCAL

didattica

: = assegnamento

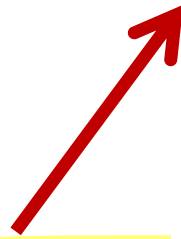
= confronto

OPERATORE LOGICO AND

(base == 0

e

esponente == 0)

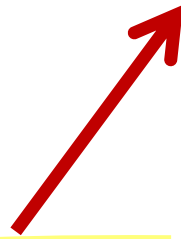


```
if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
        potenza = potenza * base; // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```

OPERATORE LOGICO AND

(base == 0 V }
e V } V
esponente == 0) V }



```
if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
        potenza = potenza * base; // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

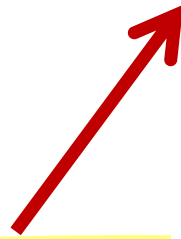
    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```

OPERATORE LOGICO AND

(base == 0

&&

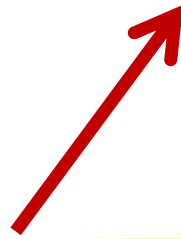
esponente == 0)



```
if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {                       // moltiplicazione per base e conta quante
        potenza = potenza * base; // volte mancano per finire il ciclo
        prodMancanti = prodMancanti - 1;
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```

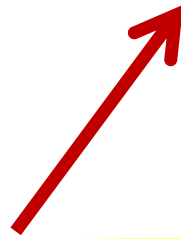

`((base == 0) && (esponente == 0))` } 0
1



```
if ((base == 0) && (esponente == 0)) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
        potenza = potenza * base; // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```

`((base == 0) && (esponente == 0))`



```
if ((base == 0) && (esponente == 0)) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
        potenza = potenza * base; // moltiplicazione per base e conta quante
        prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
}
```

STESSA SEMANTICA

((base == 0)**&&****(esponente == 0))**

```

#include <iostream>
using namespace std;

void main()
{ int esponente;
  int base, potenza;
  int prodMancanti;

  // positivo o nullo
  // potenza sempre maggiore o uguale a 1
  // positivo o nullo
  // presenta le funzionalità del programma

  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;

  // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if ((base == 0) && (esponente == 0)) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else
  { // esponente OK, calcola
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;
      prodMancanti = prodMancanti - 1;
    }

    cout << "L'elevamento a potenza di " << base << " per"
          << esponente << " vale" << potenza;
  }
}

```

== ha maggiore precedenza di &&**(base == 0****&&****esponente == 0)**

$$a + b + c * d$$

// positivo o nullo

// potenza rappresenta anche i valori parziali

// positivo o nullo

// presenta le funzionalità del programma

cout << "Calcolo dell'elevamento a potenza di una base intera "

<< "a un esponente intero positivo." << endl;

// legge base ed esponente

cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";

cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio

cout << "La potenza non è definita";

else // esponente OK, calcola

{ potenza = 1;

prodMancanti = esponente;

while (prodMancanti > 0) // il corpo del ciclo esegue una

{ potenza = potenza * base; // moltiplicazione per base e conta quante

prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo

} // stampa il risultato

cout << "L'elevamento a potenza di " << base << " per"

<< esponente << " vale" << potenza;

}

```
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresentabile su tipi interi
  int prodMancanti;        // positivo o nullo
                             // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;

                             // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else
  {
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
      potenza = potenza * base; // moltiplicazione per base e conta quante
      prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    }

    cout << "L'elevamento a potenza di " << base << " per "
          << esponente << " vale" << potenza;
  }
}
```

robusto

risponde ad un uso scorretto in modo adeguato
(input con dati fuori dominio)

a + b + c * d

```
#include <iostream.h>
void main()
{ int esponente;
  int base, potenza;
  int prodMancanti;

  // positivo o nullo
  // potenza rappresenta anche i valori parziali
  // positivo o nullo
  // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;

  // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi:";
  cin >> base >> esponente;

  if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
  else // esponente OK, calcola
  {
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {
      potenza = potenza * base; // moltiplicazione per base e conta quante
      prodMancanti = prodMancanti - 1; // volte mancano per finire il ciclo
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
          << esponente << " vale" << potenza;
  }
}
```

*** ha maggiore precedenza di +**

DEBOLEZZE

Il programma non è
robusto

un oggetto che resiste
a sollecitazioni per cui
non è stato progettato





esponente < 0

```
// legge base ed esponente
cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0) // il corpo del ciclo esegue una
    {                       // moltiplicazione per base e conta quante
        potenza = potenza * base; // volte mancano per finire il ciclo
        prodMancanti = prodMancanti - 1;
    } // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
```



esponente < 0

```
// legge base ed esponente
cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
cin >> base >> esponente;

if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else                                // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)        // il corpo del ciclo esegue una
    {                               // moltiplicazione per base e conta quante
        potenza = potenza * base;  // volte mancano per finire il ciclo
        prodMancanti = prodMancanti - 1;
    }                               // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
}
```

INTERVENTO MINIMO

istruzioni che verificano
la correttezza dei dati

e in caso di dati scorretti
lo segnalano all'utente

```
                                // legge base ed esponente
cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
cin >> base >> esponente;

if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else                                // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)          // il corpo del ciclo esegue una
    {                                // moltiplicazione per base e conta quante
        potenza = potenza * base;    // volte mancano per finire il ciclo
        prodMancanti = prodMancanti - 1;
    }                                // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
}
```

istruzioni che verificano
la correttezza dei dati
e in caso di dati scorretti
lo segnalano all'utente

```
                                // legge base ed esponente
cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
cin >> base >> esponente;

if (base == 0 && esponente == 0)    // potenza non definita? Segnala con messaggio
    cout << "La potenza non è definita";
else                                // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)          // il corpo del ciclo esegue una
    {                                // moltiplicazione per base e conta quante
        potenza = potenza * base;    // volte mancano per finire il ciclo
        prodMancanti = prodMancanti - 1;
    }                                // stampa il risultato

    cout << "L'elevamento a potenza di " << base << " per"
    << esponente << " vale" << potenza;
}
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)       // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                     // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;    // prodotto per base e conta quante
    }                                     // volte mancano per finire il ciclo
                                          // stampa il risultato
  }
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)       // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                     // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;    // prodotto per base e conta quante
    }                                     // volte mancano per finire il ciclo
                                          // stampa il risultato
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)       // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                     // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;    // prodotto per base e conta quante
    }                                     // volte mancano per finire il ciclo
                                          // stampa il risultato
  }
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)        // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                      // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;    // prodotto per base e conta quante
    }                                     // volte mancano per finire il ciclo
                                          // stampa il risultato
  }
```



```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)       // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                     // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;    // prodotto per base e conta quante
    }                                     // volte mancano per finire il ciclo
                                          // stampa il risultato
  }
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)       // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                     // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;    // prodotto per base e conta quante
    }                                     // volte mancano per finire il ciclo
                                          // stampa il risultato
  }
```

ROBUSTEZZA MINIMA

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0. */
#include <iostream.h>
void main()
{ int esponente;           // positivo o nullo
  int base, potenza;       // potenza rappresenta anche i valori parziali
  int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
  cout << "Calcolo dell'elevamento a potenza di una base intera "
        << "a un esponente intero positivo." << endl;
                           // legge base ed esponente
  cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
  cin >> base >> esponente;

  if (esponente < 0)       // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

  else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

  else                    // esponente OK, calcola
  { prodMancanti = esponente;
    while (prodMancanti > 0)
    { potenza = potenza * base;           // il corpo del ciclo esegue il
      prodMancanti = prodMancanti - 1;     // prodotto per base e conta quante
                                           // volte mancano per finire il ciclo
    }
                                           // stampa il risultato
  }
```

*in caso di esponente < 0 ,
permettiamo un nuovo
inserimento*

```
                                // legge base ed esponente
cout << "inserisci la base e l'esponente, separati da uno o più spazi: ";
cin >> base >> esponente;

if (esponente < 0)                // esponente negativo? Segnala con messaggio
    cout << "L'esponente introdotto è negativo" << endl;

else if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else                                // esponente OK, calcola
{
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza = potenza * base;           // il corpo del ciclo esegue il
        prodMancanti = prodMancanti - 1;     // prodotto per base e conta quante
                                            // volte mancano per finire il ciclo
    }
                                            // stampa il risultato
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0.
 * Se l'utente introduce un esponente negativo, il programma segnala che
 * il valore della potenza non viene calcolato e richiede un valore accettabile*/
#include <iostream.h>

void main()
{
    int esponente;           // positivo o nullo
    int base, potenza;       // potenza rappresenta anche i valori parziali
    int prodMancanti;        // positivo o nullo
                                // presenta le funzionalità del programma

    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

do
{
    // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;
    if (esponente < 0)        // esponente negativo? Errore, riprova
        cout << "L'esponente introdotto è negativo: riprova" << endl;
} while (esponente < 0 );

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else                            // esponente OK, calcola
{
    prodMancanti = esponente;
    while (prodMancanti > 0)
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero positivo o nullo.
 * Viene considerato anche il caso di 0 elevato 0.
 * Se l'utente introduce un esponente negativo, il programma segnala che
 * il valore della potenza non viene calcolato e richiede un valore accettabile*/
#include <iostream.h>

void main()
{
    int esponente;           // positivo o nullo
    int base, potenza;       // potenza rappresenta anche i valori parziali
    int prodMancanti;        // positivo o nullo
                                // presenta le funzionalità del programma

    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

do
{
    // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;
    if (esponente < 0)        // esponente negativo? Errore, riprova
        cout << "L'esponente introdotto è negativo: riprova" << endl;
} while (esponente < 0 );

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else                                // esponente OK, calcola
{
    prodMancanti = esponente;
    while (prodMancanti > 0)
```

2 CICLI

a condizione
finale

```
do  
{  
    [ciclo]  
} while [condiz.]
```

Il corpo del ciclo
deve essere eseguito
almeno *una* volta

**a condizione
finale**


```
do
{
    [ciclo]
} while [condiz.]
```


SEMANTICA PIÙ RISTRETTA

**a condizione
finale**

```
do
{
    [ciclo]
} while [condiz.]
```

Il corpo del ciclo
può essere eseguito
0 o più volte



a condizione
iniziale

```
while [condiz.]  
{  
    [ciclo]  
}
```

a condizione
finale

```
do  
{  
    [ciclo]  
} while [condiz.]
```

ha una semantica
più ricca

a condizione
iniziale

```
while [condiz.]  
  {  
    [ciclo]  
  }
```

a condizione
finale

```
do  
  {  
    [ciclo]  
  } while [condiz.]
```

FORME ABBREVIATE DI =

`[variabile] +-*/`

```
int prodMancanti;           // positivo o nullo
                           // presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;
```

```
do
{
    // legge base ed esponente
```

```
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;
    if (esponente < 0)           // esponente negativo? Errore, riprova
        cout << "L'esponente introdotto è negativo: riprova" << endl;
    } while (esponente < 0 );
```

```
if (base == 0 && esponente == 0) // potenza non definita, segnala con
    cout << "La potenza non è definita"; // messaggio
```

```
else
{
    // esponente OK, calcola
    potenza = 1;
```

```
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza = potenza * base;
        prodMancanti = prodMancanti -1;
    }
```

```
    cout << "L'elevamento a potenza di" << base << " per"
    << esponente << " vale " << potenza;
```

```
}
```

```
int prodMancanti;           // positivo o nullo
                           // presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
     << "a un esponente intero positivo." << endl;
```

```
do
```

```
{                               // legge base ed esponente
```

```
    cout << "Inserisci la base e l'esponente";
```

```
    cin >> base >> esponente;
```

```
    if (esponente < 0)           // esponente negativo? Errore, riprova
```

```
    {
        potenza = potenza * base
    } while (esponente < 0 );
```

stessa

semantica

```
if (base == 0)                 // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio
```

```
else                           // esponente OK, calcola
```

```
{    potenza = 1;
```

```
    prodMancanti = esponente;
```

```
    while (prodMancanti > 0)
```

```
    {    potenza = potenza * base;
```

```
        prodMancanti = prodMancanti -1;
```

```
    }
```

```
    cout << "L'elevamento a potenza di" << base << " per"
```

```
    << esponente << " vale " << potenza;
```

```
}
```



```
int prodMancanti; // positivo o nullo
// le funzionalità del programma
// di una base intera "
// a un esponente intero positivo." << endl;
```

```
do
{
    // legge base ed esponente
```

```
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;
    if (esponente < 0) // esponente negativo? Errore, riprova
    {
        prodMancanti = prodMancanti - 1;
        cout << "Errore: esponente negativo. Riprova." << endl;
    } while (esponente < 0);
```

```
if (base < 0) // potenza non definita? Segnala con
{
    cout << "La potenza non è definita"; // messaggio
```

```
else // esponente OK, calcola
{
    potenza = 1;
```

```
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza = potenza * base;
        prodMancanti = prodMancanti - 1;
    }
```

```
    cout << "L'elevamento a potenza di" << base << " per"
    << esponente << " vale " << potenza;
```

```
}
```

```
int prodMancanti; // positivo o nullo
// le funzionalità del programma
// di una base intera "
// a un esponente intero positivo." << endl;
```

```
do
{
    // legge base ed esponente
```

```
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;
    if (esponente < 0) // esponente negativo? Errore, riprova
    {
        prodMancanti = prodMancanti + 1;
        cout << "Errore: esponente negativo. Riprova." << endl;
    } while (esponente < 0);
```

```
if (base < 0) // potenza non definita? Segnala con
{
    cout << "La potenza non è definita"; // messaggio
```

```
else // esponente OK, calcola
{
    potenza = 1;
```

```
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza = potenza * base;
        prodMancanti = prodMancanti - 1;
    }
```

```
    cout << "L'elevamento a potenza di" << base << " per"
    << esponente << " vale " << potenza;
```

```
}
```



```
int prodMancanti;           // positivo o nullo
                           // presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

do
{
    // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;
    if (esponente < 0)       // esponente negativo? Errore, riprova
        cout << "L'esponente introdotto è negativo: riprova" << endl;
} while (esponente < 0 );

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else                          // esponente OK, calcola
{
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base;           // il corpo del ciclo esegue
        prodMancanti --;         // prodotto per base e conta quante
    }                               // volte mancano per finire il ciclo
    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero, anche negativo.
 * Viene considerato anche il caso di 0 elevato 0.
 */
#include <iostream.h>
void main()
{
    int esponente;                // positivo o nullo
    int base, potenza;           // potenza rappresenta anche i valori parziali
    int prodMancanti;            // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

    do
    {
        // legge base ed esponente
        cout << "Inserisci la base e l'esponente";
        cin >> base >> esponente;
        if (esponente < 0)        // esponente negativo? Errore, riprova
        {
            cout << "L'esponente introdotto è negativo: riprova" << endl;
        } while (esponente < 0 );

        if (base == 0 && esponente == 0) // potenza non definita? Segnala con
            cout << "La potenza non è definita";// messaggio

        else                    // esponente OK, calcola
        {
            potenza = 1;
            prodMancanti = esponente;
            while (prodMancanti > 0)
            {
                potenza *= base;    // il corpo del ciclo esegue
                prodMancanti --;    // prodotto per base e conta quante
            }                      // volte mancano per finire il ciclo
        }
    }
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero, anche negativo.
 * Viene considerato anche il caso di 0 elevato 0.
 */
#include <iostream.h>
void main()
{
    int esponente;           // positivo o nullo
    int base, potenza;        // potenza rappresenta anche i valori parziali
    int prodMancanti;         // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

    do
    {
        // legge base ed esponente
        cout << "Inserisci la base e l'esponente";
        cin >> base >> esponente;
        if (esponente < 0)      // esponente negativo? Errore, riprova
        {
            cout << "L'esponente introdotto è negativo: riprova" << endl;
        } while (esponente < 0 );

        if (base == 0 && esponente == 0) // potenza non definita? Segnala con
            cout << "La potenza non è definita";// messaggio

        else                    // esponente OK, calcola
        {
            potenza = 1;
            prodMancanti = esponente;
            while (prodMancanti > 0)
            {
                potenza *= base;    // il corpo del ciclo esegue
                prodMancanti --;    // prodotto per base e conta quante
            }                      // volte mancano per finire il ciclo
        }
    }
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero, anche negativo.
 * Viene considerato anche il caso di 0 elevato 0.
 */
#include <iostream.h>
void main()
{
    int esponente;           // positivo, negativo o nullo
    int base, potenza;       // potenza rappresenta anche i valori parziali
    int prodMancanti;        // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

    do
    {
        // legge base ed esponente
        cout << "Inserisci la base e l'esponente";
        cin >> base >> esponente;
        if (esponente < 0) // esponente negativo? Errore, riprova
        {
            cout << "L'esponente introdotto è negativo: riprova" << endl;
        } while (esponente < 0 );

        if (base == 0 && esponente == 0) // potenza non definita? Segnala con
            cout << "La potenza non è definita";// messaggio

        else // esponente OK, calcola
        {
            potenza = 1;
            prodMancanti = esponente;
            while (prodMancanti > 0)
            {
                potenza *= base; // il corpo del ciclo esegue
                prodMancanti--; // prodotto per base e conta quante
                                // volte mancano per finire il ciclo
            }
        }
    }
}
```

```
/* Programma che calcola l'elevamento a potenza di una base
 * intera ad un esponente intero, anche negativo.
 * Viene considerato anche il caso di 0 elevato 0.
 */
#include <iostream.h>
void main()
{
    int esponente;           // positivo o nullo
    int base, potenza;       // potenza rappresenta anche i valori parziali
    int prodMancanti;        // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di base intera "
         << "a un esponente positivo";

    do
    {
        // legge base e esponente
        cout << "Inserisci la base e l'esponente";
        cin >> base >> esponente;
        if (esponente < 0) // esponente negativo? Errore, riprova
        {
            cout << "L'esponente introdotto è negativo: riprova" << endl;
        } while (esponente < 0 );

        if (base == 0 && esponente == 0) // potenza non definita? Segnala con
            cout << "La potenza non è definita";// messaggio

        else // esponente OK, calcola
        {
            potenza = 1;
            prodMancanti = esponente;
            while (prodMancanti > 0)
            {
                potenza *= base; // il corpo del ciclo esegue
                prodMancanti--; // prodotto per base e conta quante
                                // volte mancano per finire il ciclo
            }
        }
    } while (esponente < 0 );
}
```

$$a^{-b} = 1/a^{+b}$$

```
int esponente;           // positivo, negativo o nullo
int base, potenza;       // potenza rappresenta anche i valori parziali
int prodMancanti;        // positivo o nullo
                           // presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
      << "a un esponente intero positivo." << endl;

                           // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else
{
    if (esponente < 0) // potenza definita, calcola
                       // esponente negativo?
    {
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base; // il corpo del ciclo esegue
        prodMancanti --; // prodotto per base e conta quante
    }                   // volte mancano per finire il ciclo
                       // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
          << esponente << " vale " << potenza;
```

```
int esponente;           // positivo, negativo o nullo
int base, potenza;       // potenza rappresenta anche i valori parziali
int prodMancanti;        // positivo o nullo
                          // presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
     << "a un esponente intero positivo." << endl;

                          // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else
{
    if (esponente < 0)
    {
        esponente = - esponente
    }
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base;           // il corpo del ciclo esegue
        prodMancanti --;           // prodotto per base e conta quante
    }                               // volte mancano per finire il ciclo
    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
         << esponente << " vale " << potenza;
```

~~potenza~~

```
int esponente;           // positivo, negativo o nullo
int base, potenza;       // potenza rappresenta anche i valori parziali
int prodMancanti;        // positivo o nullo
                          // presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
      << "a un esponente intero positivo." << endl;

                          // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else
{
    if (esponente < 0)
    {
        esponente = - esponente
    }
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base;           // il corpo del ciclo esegue
        prodMancanti --;           // prodotto per base e conta quante
    }                               // volte mancano per finire il ciclo
    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
          << esponente << " vale " << potenza;
}
```

} 1/potenza


```
#include <iostream.h>
void main()
{
    int esponente;
    int base, potenza;
    int segno;
    int prodMancanti;
    // potenza rappresenta anche i valori parziali
    // tiene conto del segno dell'esponente
    // positivo o nullo
    // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

    // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;

    if (base == 0 && esponente == 0) // potenza non definita? Segnala con
        cout << "La potenza non è definita"; // messaggio

    else
        // potenza definita, calcola
        // esponente negativo?
        {
            if (esponente < 0)
            {
                esponente = - esponente // rendi positivo l'esponente
            }
            potenza = 1;
            prodMancanti = esponente;
            while (prodMancanti > 0)
            {
                potenza *= base; // il corpo del ciclo esegue
                prodMancanti --; // prodotto per base e conta quante
            } // volte mancano per finire il ciclo
            // stampa il risultato
            cout << "L'elevamento a potenza di" << base << " per"
```

```
#include <iostream.h>
void main()
{
    int esponente;
    int base, potenza;           // potenza rappresenta anche i valori parziali
    int segno = 1;               // tiene conto del segno dell'esponente
    int prodMancanti;            // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

                                // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;

    if (base == 0 && esponente == 0) // potenza non definita? Segnala con
        cout << "La potenza non è definita"; // messaggio

    else                          // potenza definita, calcola
    {                             // esponente negativo?
        if (esponente < 0)
        {
            esponente = - esponente // rendi positivo l'esponente
        }
        potenza = 1;
        prodMancanti = esponente;
        while (prodMancanti > 0)
        {
            potenza *= base;        // il corpo del ciclo esegue
            prodMancanti --;        // prodotto per base e conta quante
        }                          // volte mancano per finire il ciclo
                                    // stampa il risultato
        cout << "L'elevamento a potenza di" << base << " per"
```

```
#include <iostream.h>
void main()
{
    int esponente;
    int base, potenza;           // potenza rappresenta anche i valori parziali
    int segno;                   // tiene conto del segno dell'esponente
    int prodMancanti;            // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

                                // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;

    if (base == 0 && esponente == 0) // potenza non definita? Segnala con
        cout << "La potenza non è definita"; // messaggio

    else                          // potenza definita, calcola
    {                             // esponente negativo?
        if (esponente < 0)
        {
            esponente = - esponente // rendi positivo l'esponente
        }
        potenza = 1;
        prodMancanti = esponente;
        while (prodMancanti > 0)
        {
            potenza *= base;           // il corpo del ciclo esegue
            prodMancanti --;           // prodotto per base e conta quante
        }                             // volte mancano per finire il ciclo
                                // stampa il risultato
        cout << "L'elevamento a potenza di" << base << " per"
```

CICLO FOR

```
#include <iostream>
using namespace std;

int main()
{
    int esponente;
    int base, potenza;
    int segno = 1;
    int prodMancanti;

    // potenza rappresenta anche i valori parziali
    // tiene conto del segno dell'esponente
    // positivo o nullo
    // presenta le funzionalità del programma

    cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "esponente: ";

    do {
        // legge base ed esponente
        // legge base ed esponente
        cout << "Inserisci la base e l'esponente: ";
        cin >> base >> esponente;
        if (esponente < 0) // esponente negativo? Errore, riprova
            cout << "L'esponente introdotto è negativo: riprova" << endl;
    } while (esponente < 0)

    if (base == 0 && esponente == 0) // potenza non definita? Segnala con
        cout << "La potenza non è definita"; // messaggio

    else
    {
        if (esponente < 0) // a condizione
        { // iniziale
            segno = -1;
            esponente = -esponente;
        }
        potenza = 1;
        prodMancanti = esponente;
        while (prodMancanti > 0)
        {
            potenza *= base;
            prodMancanti--;
        }

        // volte mancano per finire il ciclo
    }
}
```

```
#include <iostream.h>
void main()
{
    int esponente;
    int base, potenza;           // potenza rappresenta anche i valori parziali
    int segno;                   // tiene conto del segno dell'esponente
    int prodMancanti;            // positivo o nullo
                                // presenta le funzionalità del programma
    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

                                // legge base ed esponente
    cout << "Inserisci la base e l'esponente";
    cin >> base >> esponente;

    if (base == 0 && esponente == 0) // potenza non definita? Segnala con
        cout << "La potenza non è definita"; // messaggio

    else                          // potenza definita, calcola
    {
        if (esponente < 0)        // esponente negativo?
        {
            segno = -1;           // cambia il segno
            esponente = - esponente // rendi positivo l'esponente
        }
        potenza = 1;
        prodMancanti = esponente;
        while (prodMancanti > 0)
        {
            potenza *= base;      // il corpo del ciclo esegue
            prodMancanti --;       // prodotto per base e conta quante
        }                        // volte mancano per finire il ciclo
                                // stampa il risultato
        cout << "L'elevamento a potenza di" << base << " per"
```

C++**CICLO FOR**

```
#include <iostream.h>
void main()
{
    int esponente;

    // potenza rappresenta anche i valori parziali
    // tiene conto del segno dell'esponente
    // positivo o nullo
    // presenta le funzionalità del programma

    cout << "Calcolo dell'elevamento a potenza di una base intera "
         << "a un esponente intero positivo." << endl;

    do
    {
        // legge base ed esponente
        // legge base ed esponente
        cout << "Inserisci la base e l'esponente";
        cin >> base >> esponente;
        if (esponente < 0) // esponente negativo? Errore, riprova
            cout << "L'esponente introdotto è negativo: riprova" << endl;
    } while (esponente < 0)

    if (base == 0 && esponente == 0) // potenza non definita? Segnala con
        cout << "La potenza non è definita"; // messaggio

    else
    {
        if (esponente < 0)
        {
            // potenza definita, calcola
            // esponente negativo? cambia il segno
            segno = -1; // cambia il segno
            esponente = -esponente // rendi positivo l'esponente
        }

        potenza = 1;
        prodMancanti = esponente;
        while (prodMancanti > 0)
        {
            potenza *= base; // il corpo del ciclo
            prodMancanti--; // prodotto per base e conta quante
        } // volte mancano per finire il ciclo
    }
}
```

**a condizione
iniziale****while [condiz.]**
{
 [ciclo]
}**a condizione
finale****do**
{
 [ciclo]
} **while [condiz.]**

CICLO FOR

```
for ( prodMancanti = esponente;  
    prodMancanti > 0;  
    prodMancanti --)
```

```
#include <iostream>
using namespace std;

int esponente;
int base, potenza;           // potenza rappresenta anche i valori parziali
int segno = 1;               // tiene conto del segno dell'esponente
int prodMancanti;            // positivo o nullo
// presenta le funzionalità del programma

cout << "Calcolo dell'elevamento a potenza di una base intera "
<< "esponente ";

do {
    // legge base ed esponente
    // legge base ed esponente
    cout << "Inserisci la base e l'esponente ";
    cin >> base >> esponente;
    if (esponente < 0)          // esponente negativo? Errore, riprova
        cout << "L'esponente introdotto è negativo: riprova" << endl;
} while (esponente < 0)

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                          // potenza definita, calcola
{
    if (esponente < 0)        // esponente negativo?
    {
        segno = -1;          // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base;      // il corpo del ciclo esegue
        prodMancanti--;        // prodotto per base e conta quante
    }                          // volte mancano per finire il ciclo
}
```


CICLO FOR

```
for ( prodMancanti = esponente;  
    prodMancanti > 0;  
    prodMancanti --)
```

```
#include <iostream>
using namespace std;

int esponente;
int base, potenza;           // potenza rappresenta anche i valori parziali
int segno = 1;               // tiene conto del segno dell'esponente
int prodMancanti;             // positivo o nullo
// presenta le funzionalità del programma

cout << "Calcolo dell'elevamento a potenza di una base intera "
<< "esponente ";

do {
    // legge base ed esponente
    // legge base ed esponente
    cout << "Inserisci la base e l'esponente ";
    cin >> base >> esponente;
    if (esponente < 0)          // esponente negativo? Errore, riprova
        cout << "L'esponente introdotto è negativo: riprova" << endl;
} while (esponente < 0)

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                           // potenza definita, calcola
{
    if (esponente < 0)          // esponente negativo?
    {
        segno = -1;           // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base;       // il corpo del ciclo esegue
        prodMancanti--;         // prodotto per base e conta quante
    }                           // volte mancano per finire il ciclo
}
```



```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base; // il corpo del ciclo esegue il
        prodMancanti --; // prodotto per base e conta quante
    } // volte mancano per finire il ciclo
    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    prodMancanti = esponente;
    while (prodMancanti > 0)
    {
        potenza *= base; // il corpo del ciclo esegue il
        prodMancanti --; // prodotto per base e conta quante
    } // volte mancano per finire il ciclo
    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

{ Inizializzazione

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

{ Condizione di permanenza
nel ciclo


```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

Aggiornamento del contatore
(al termine del corpo)

```
int segno; // tiene conto del segno dell'esponente
int prodMancanti; // positivo o nullo
// presenta le funzionalità del programma
cout << "Calcolo dell'elevamento a potenza di una base intera "
    << "a un esponente intero positivo." << endl;

// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita";// messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;

    for (prodMancanti = esponente;
        prodMancanti > 0;
        prodMancanti --)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
        << esponente << " vale " << potenza;
}
}
```

```
                                // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                                // potenza definita, calcola
{
    if (esponente < 0)                // esponente negativo?
    {
        segno = -1;                // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente;    // ciclo a conteggio il cui corpo
        prodMancanti > 0;            // eseguito esponente volte,
        prodMancanti--;)             //aggiorna potenza
        potenza *= base;             //moltiplicandola per base

                                // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1)                // esponente positivo?
        cout << potenza;           // stampa il valore di potenza
    else                            // esponente negativo?
        cout << 1 / potenza;       // stampa il reciproco di potenza
    cout << endl;
}
```

```
                                // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                                // potenza definita, calcola
{
    if (esponente < 0)                // esponente negativo?
    {
        segno = -1;                // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente;    // ciclo a conteggio il cui corpo
        prodMancanti > 0;            // eseguito esponente volte,
        prodMancanti--;)             //aggiorna potenza
        potenza *= base;             //moltiplicandola per base

                                // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1)                // esponente positivo?
        cout << potenza;          // stampa il valore di potenza
    else                            // esponente negativo?
        cout << 1 / potenza;      // stampa il reciproco di potenza
    cout << endl;
}
```

```
                                // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                                // potenza definita, calcola
{
    if (esponente < 0)              // esponente negativo?
    {
        segno = -1;                // cambia il segno
        esponente = - esponente    // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente;  // ciclo a conteggio il cui corpo
        prodMancanti > 0;           //eseguito esponente volte,
        prodMancanti--;)            //aggiorna potenza
        potenza *= base;            //moltiplicandola per base
    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1)                 // esponente positivo?
        cout << potenza;           // stampa il valore di potenza
    else                            // esponente negativo?
        cout << 1 / potenza;       // stampa il reciproco di potenza
    cout << endl;
}
```

PARTE SIGNIFICATIVA

```

// legge base ed esponente
e l'esponente";

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; // calcolo a conteggio all'indietro corpo
        prodMancanti > 0; //eseguito esponente volte,
        prodMancanti--) //aggiorna potenza
        potenza *= base; //moltiplicandola per base

    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}

```

1 / potenza

int /

int

= int

0.7645

```
                                // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                                // potenza definita, calcola
{
    if (esponente < 0)                // esponente negativo?
    {
        segno = -1;                // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente;      // ciclo a conteggio il cui corpo
        prodMancanti > 0;                //eseguito esponente volte,
        prodMancanti--;)                //aggiorna potenza
        potenza *= base;                //moltiplicandola per base

                                // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";

    if (segno == 1)                // esponente positivo?
        cout << potenza;            // stampa il valore di potenza
    else                            // esponente negativo?
        cout << 1 / potenza;        // stampa il reciproco di potenza
    cout << endl;
}
```

```
// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente; // ciclo a conteggio il cui corpo
        prodMancanti > 0; //eseguito esponente volte,
        prodMancanti--); //aggiorna potenza
        potenza *= base; //moltiplicandola per base

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1)
        cout << potenza;
    else
        cout << 1 / potenza;
    cout << endl;
}
```

→ **potenza = 0**


```
// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; // calcolo a conteggio all'indietro corpo
        prodMancanti > 0; // eseguito esponente volte,
        prodMancanti--) // aggiorna potenza
        potenza *= base; //moltiplicandola per base
    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}
```

1 / potenza

int / int = int

float

```
// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base;

    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}
```

1 / potenza

int / float

```
// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio
```

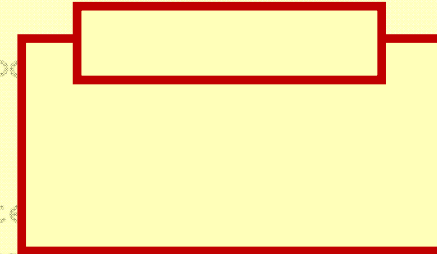
```
else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = -esponente; // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base; // ciclo a conteggio // eseguito esponente volte, //aggiorna potenza //moltiplicandola per base

    // stampa il risultato
```

potenza

27

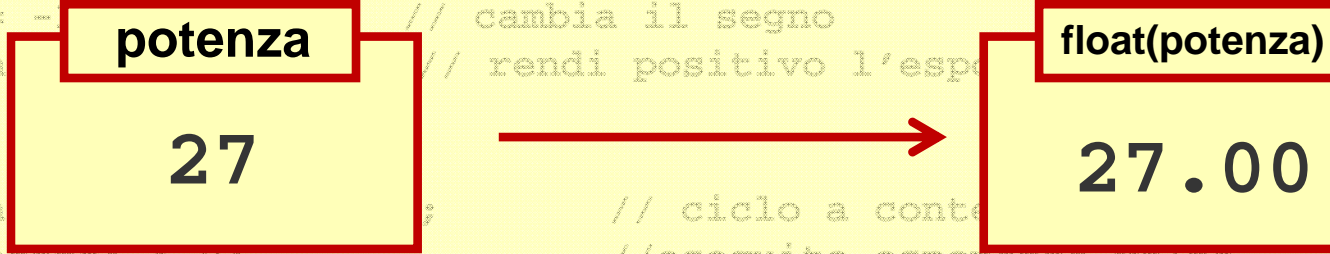


```
cout << "L'elevamento a potenza di" << base << " per"
<< segno * esponente << " vale ";
if (segno == 1) // esponente positivo?
    cout << potenza; // stampa il valore di potenza
else // esponente negativo?
    cout << 1 / potenza; // stampa il reciproco di potenza
cout << endl;
}
```

```
        // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio
```

```
else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = -esponente; // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base; // ciclo a conteggio decrescente // eseguito esponente volte, //aggiorna potenza //moltiplicandola per base
    // stampa il risultato
}
```



The diagram illustrates the conversion of an integer to a floating-point number. On the left, a box labeled "potenza" contains the integer value "27". A red arrow points from this box to another box on the right labeled "float(potenza)", which contains the floating-point value "27.00".

```
cout << "L'elevamento a potenza di" << base << " per"
<< segno * esponente << " vale ";
if (segno == 1) // esponente positivo?
    cout << potenza; // stampa il valore di potenza
else // esponente negativo?
    cout << 1 / potenza; // stampa il reciproco di potenza
cout << endl;
}
```



```
// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1;
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base;

    // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}
```

1 / float(potenza)

int / float

un operatore binario deve operare su valori dello stesso tipo

PROMOTION

```

// legge base ed esponente
base e l'esponente";
te;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1;
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base;
    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}

```

1 / float(potenza)

float / float

int

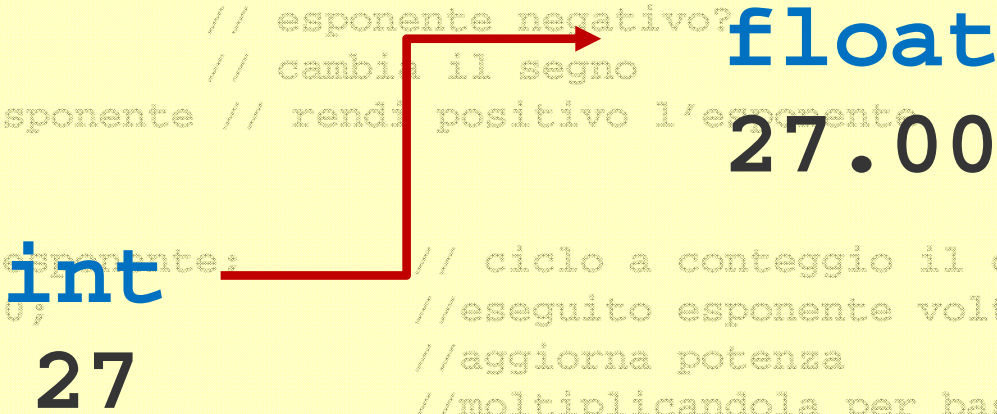
PROMOTION

```
// legge base ed esponente
base e l'esponente";
te;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente; // ciclo a conteggio il cui corpo
        prodMancanti > 0; //eseguito esponente volte,
        prodMancanti--) //aggiorna potenza
        potenza *= base; //moltiplicandola per base
    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}
```



```
// legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1; // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base; // ciclo a conteggio il cui corpo
                          //eseguito esponente volte,
                          //aggiorna potenza
                          //moltiplicandola per base
    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}
```

int
27

float
27.43


```
        // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else // potenza definita, calcola
{
    if (esponente < 0) // esponente negativo?
    {
        segno = -1;
        esponente = - esponente // rendi positivo l'esponente
    }

    potenza = 1;
    for (prodMancanti = esponente; prodMancanti > 0; prodMancanti--)
        potenza *= base;

    // stampa il risultato

    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1) // esponente positivo?
        cout << potenza; // stampa il valore di potenza
    else // esponente negativo?
        cout << 1 / potenza; // stampa il reciproco di potenza
    cout << endl;
}
```

1 / float(potenza)

1.00

/ float = float

```
                                // legge base ed esponente
cout << "Inserisci la base e l'esponente";
cin >> base >> esponente;

if (base == 0 && esponente == 0) // potenza non definita? Segnala con
    cout << "La potenza non è definita"; // messaggio

else                                // potenza definita, calcola
{
    if (esponente < 0)                // esponente negativo?
    {
        segno = -1;                // cambia il segno
        esponente = - esponente // rendi positivo l'esponente
    }
    potenza = 1;
    for (prodMancanti = esponente;    // ciclo a conteggio il cui corpo
        prodMancanti > 0;            // eseguito esponente volte,
        prodMancanti--);             //aggiorna potenza
        potenza *= base;              //moltiplicandola per base
        // stampa il risultato
    cout << "L'elevamento a potenza di" << base << " per"
    << segno * esponente << " vale ";
    if (segno == 1)                    // esponente positivo?
        cout << potenza;              // stampa il valore di potenza
    else                                // esponente negativo?
        cout << 1 / (float)potenza;    // stampa il reciproco di potenza
    cout << endl;
}
```