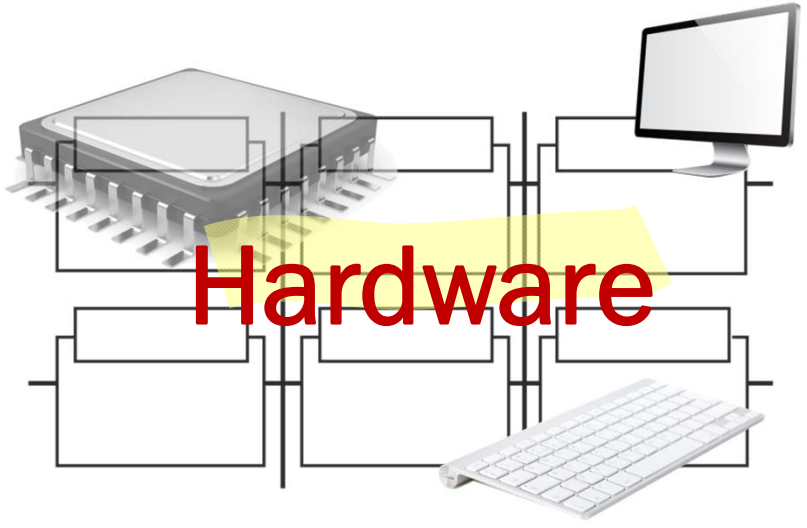
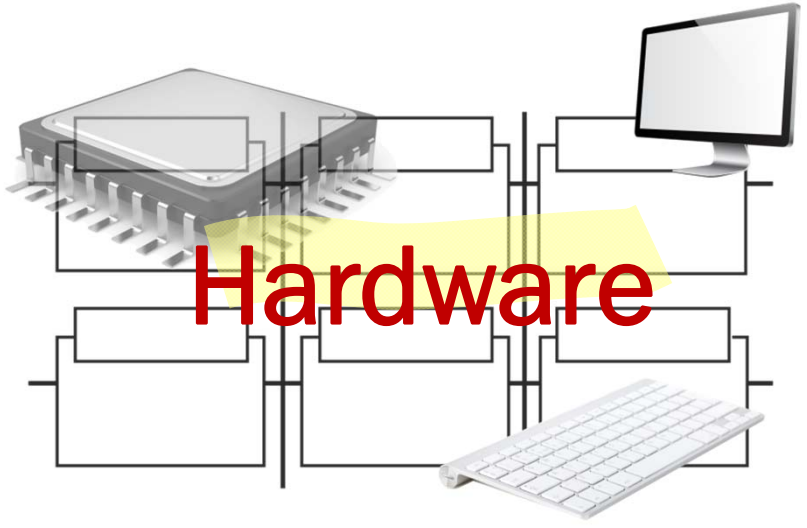
POLITECNICO DI MILANO

INFORMATICA

LA LEGGIBILITÀ DEI PROGRAMMI



elaboratore

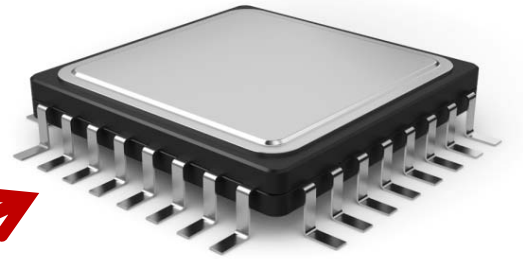


Hardware

elaboratore

Software

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

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

  cout << "Fornire valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
  { potParziale = potParziale * base;
    prodMancanti = prodMancanti - 1;
  }
  cout << "L'elevamento a potenza di " << base << " per "
       << esponente << " vale " << potenza;
}
```



Programma

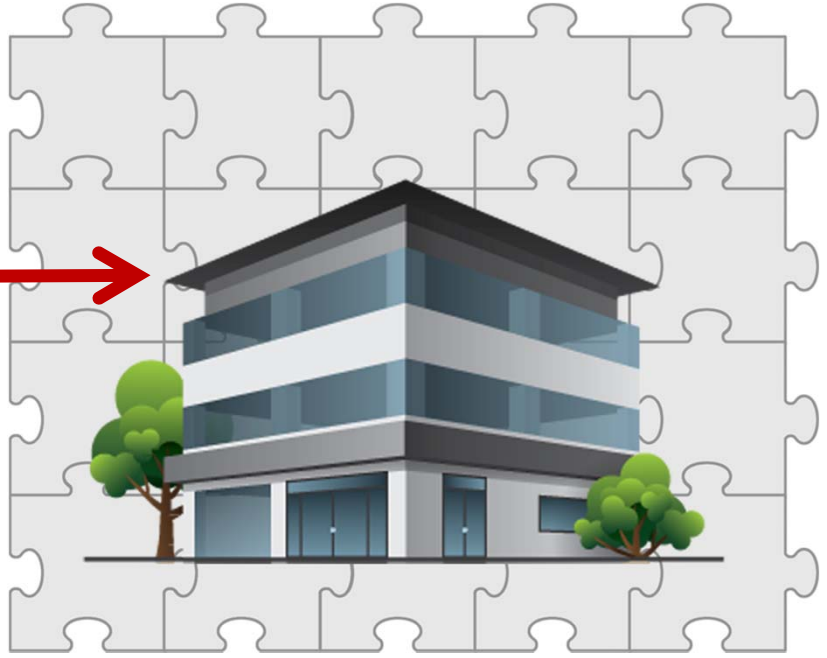
CORREGGERE GLI ERRORI

MODIFICARE IL PROGRAMMA

MODELLIZZAZIONE

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

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

  cout << "Fornire la coppia per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per "
       << esponente << " vale" << potenza;
}
```

Programma ⟷



SISTEMI INFORMATIVI

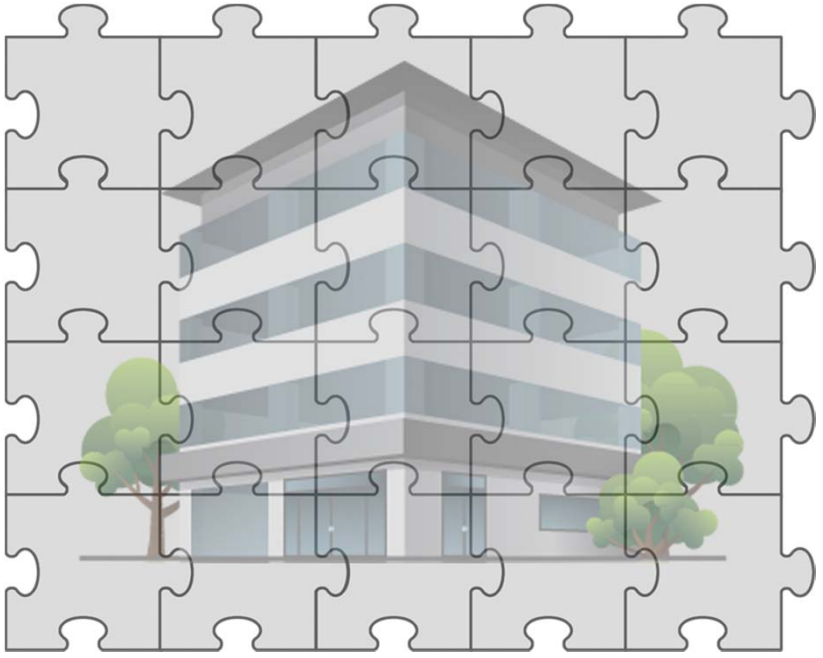
Programma

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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per "
       << esponente << " vale" << potenza;
}
```

```
insertNode (TreeNode* & ptr, int value)
{
  if (ptr == 0)
    ptr = new TreeNode (value);
  else if (value < (ptr)->data)
    insertNode (ptr ->leftPtr, value);
  else if (value > ptr->data)
    insertNode (ptr)->rightPtr, value);
  else
    cout << value << " duplicato" << endl;
}

// Questa funzione riceve un puntatore a un puntatore
// in tale modo il puntatore può essere modificato.
insertNode (TreeNode** ptr, int value)
{
  if (*ptr == 0)
    { *ptr = new insertNode (value);
      assert (*ptr != 0);
    }
  else if (value < (*ptr)->data)
    insertNode ( & ((*ptr)->leftPtr), value);
  else if (value > (*ptr)->data)
    insertNode ( & ((*ptr)->rightPtr), value);
  else
    cout << value << " duplicato" << endl;
}
preOrder (TreeNode*ptr) const
{
  if (ptr != 0)
    {  cout << ptr->data << ' ';
       preOrder (ptr->leftPtr);
       preOrder (ptr->rightPtr);
    }
}
postOrder (TreeNode*ptr) const
{
  if (ptr != 0)
    (postOrder (ptr->leftPtr);
```

Programma

SISTEMI INFORMATIVI



Programma

Creatore

Modificatore
Correttore

PARTICOLARE SEQUENZA DI DATI

Evento eccezionale non previsto

Programma

Correttore

LEGGIBILITÀ

Programma ← Modifica ←

LEGGIBILITÀ

Programma

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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per "
       << esponente << " vale" << potenza;
}
```

- Statico (codice)
- Dinamico (esecuzione)

LEGGIBILITÀ

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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per "
       << esponente << " vale" << potenza;
}
```

Dinamico

Scrivere programmi corretti

che parlano

- Scegliere opportuni nomi per le variabili

Scrivere programmi corretti

che parlano

- Scegliere opportuni nomi per le variabili
- Aggiungere delle frasi esplicative

COMMENTI

COMMENTI

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

1) Su più righe /* … */

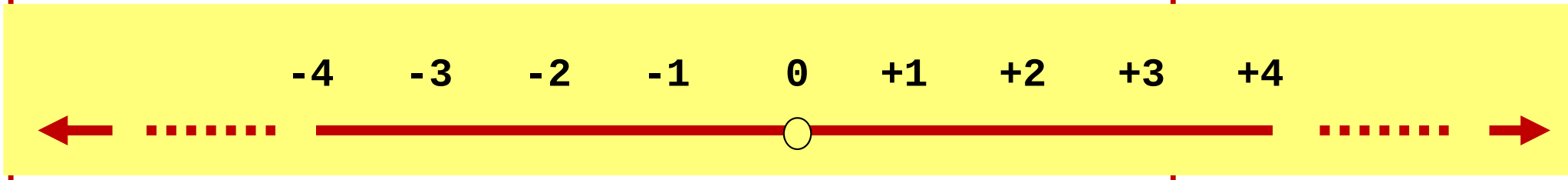
COMMENTI

```
/* 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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

2) Su una // riga

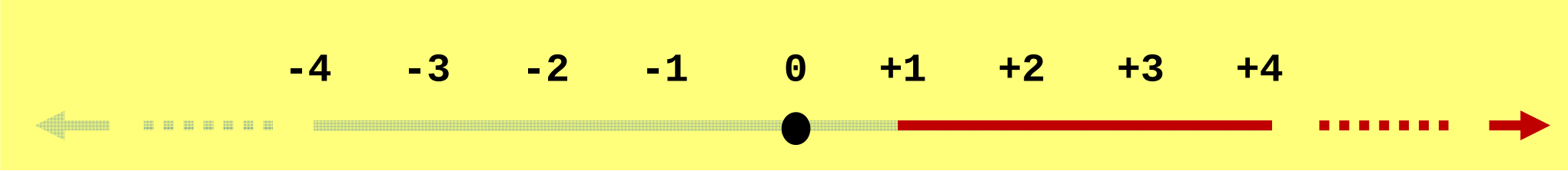
```
 /* Programma che calcola l'elevamento a potenza
  * di una base intera ad un esponente intero positivo
  */
#include <iostream.h>
void main()
{ int esponente;                    // positivo
  int base, potenza;
  int potParziale;
  int prodMancanti;                 // positivo o nullo

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
/* 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
```




```
  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
 /* 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
```




```
  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
 /* 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
```

Verificare che i dati siano in un dominio accettabile

```
  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

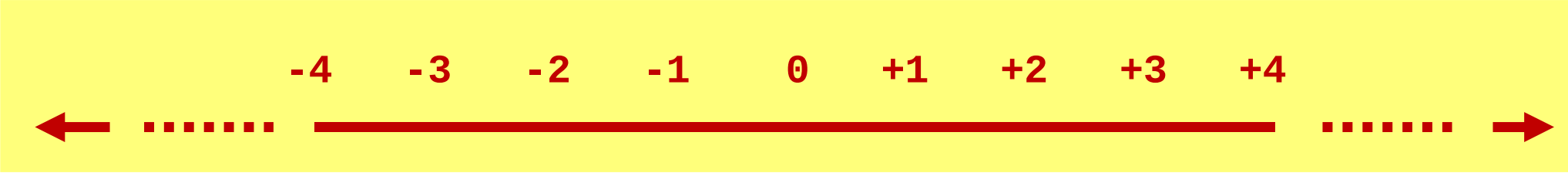
```
 /* Programma che calcola l'elevamento a potenza
  * di una base intera ad un esponente intero positivo
  */
#include <iostream.h>
void main()
{ int esponente;                    // positivo
  int base, potenza;
  int potParziale;
  int prodMancanti;                 // positivo o nullo

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
/* 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

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

  cout << "Fornire i valori per base ed esponente,"
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti - 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

Ogni informazione supplementare
è utile per capire
eventuali malfunzionamenti

```
 /* 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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
 /* 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
```




```
  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
/* 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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

```
 /* 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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

$-3^3 = -27$

Descrizione dei sottoproblemi

```
 /* 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

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

  cout << "Fornire i valori per base ed esponente, "
       << "separati da uno o più spazi: ";
  cin >> base >> esponente;
  potParziale = 1;
  prodMancanti = esponente;
  do
   { potParziale = potParziale * base;
     prodMancanti = prodMancanti – 1;
   } while (prodMancanti > 0);
  potenza = potParziale;
  cout << "L'elevamento a potenza di " << base << " per"
       << esponente << " vale" << potenza;
}
```

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

potParziale = potParziale * base

```
  prodMancanti = esponente;
  do
   {
     prodParziale = 0;
     somMancanti = base;
     do
       { prodParziale = prodParziale + potParziale;
         somMancanti = somMancanti -1;
       } while (somMancanti > 0);
     potParziale = prodParziale;

     prodMancanti = prodMancanti -1;
   } while (prodMancanti > 0);

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

```
cin >> base >> esponente;

  prodMancanti = esponente;
  do
   { // potParziale = potParziale * base
     prodParziale = 0;
     somMancanti = base;
     do
       { prodParziale = prodParziale + potParziale;
         somMancanti = somMancanti -1;
       } while (somMancanti > 0);
     potParziale = prodParziale;

     prodMancanti = prodMancanti -1;
   } while (prodMancanti > 0);

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

~~potParziale = potParziale * base~~

```
cin >> base >> esponente;

  prodMancanti = esponente;
  do
   { // potParziale = potParziale * base
      prodParziale = 0;
      somMancanti = base;
      do
        { prodParziale = prodParziale + potParziale;
          somMancanti = somMancanti -1;
        } while (somMancanti > 0);
      potParziale = prodParziale;

     prodMancanti = prodMancanti -1;
   } while (prodMancanti > 0);

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

```
cin >> base >> esponente;

  prodMancanti = esponente;
  do
   { // potParziale = potParziale * base
      prodParziale = 0;
      somMancanti = base;
      do
        { prodParziale = prodParziale + potParziale;
          somMancanti = somMancanti -1;
        } while (somMancanti > 0);
      potParziale = prodParziale;

    prodMancanti = prodMancanti -1;
   } while (prodMancanti > 0);

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

Livelli di astrazione

Duplice lettura del programma

Calcolatore

Uomo

Linguaggio di programmazione

Linguaggio naturale

INCOLONNAMENTI

```
/* 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;
}
```

INCOLONNAMENTI

```
 /* 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;
}
```

INCOLONNAMENTI

```
 /* 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;
}
```

Livelli più bassi

```
 /* 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; }
```