for

```
for ( i=0; i<n; i=i+1 )
    corpo del ciclo;
```

è equivalente a

```
i=0;
while ( i<n ) {
    corpo del ciclo;
    i=i + 1;
}
```

Operatori di incremento e decremento

- L’operatore unario `++` incrementa il suo operando
- La frase `i = i + 1` è quindi equivalente all’espressione `++i`

```
for ( i=0; i<n; i++ )
```

- L’operatore unario `--` ha significato analogo

Operatori di incremento e decremento

- Differenza tra l'espressione `++i` e l'espressione `i++`

```
i = 0;
cout << i;
cout << ++i;
cout << i++;
cout i;
```

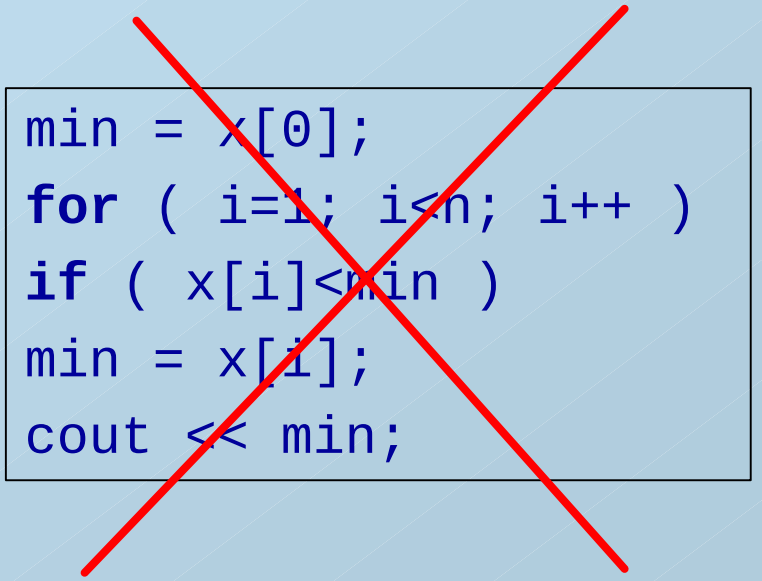
- Output:

```
0 1 1 2
```

Indentazione

- Disporre il testo del programma in modo da evidenziare la struttura delle istruzioni composte

```
min = x[0];
for ( i=1; i<n; i++ )
   if ( x[i]<min )
      min = x[i];
cout << min;
```



Indentazione

```
min = x[0];
max = x[0];
sum = x[0];
for ( i=1; i<n; i++ ) {
   sum = sum + x[i];
   if ( x[i]<min )
      min = x[i];
   else
      if (x[i]>max )
         max = x[i];
}
cout << min; cout << max;
cout << sum/n;
```

indentazione alternativa

```
min = x[0];
max = x[0];
sum = x[0];
for ( i=1; i<n; i++ ) {
   sum = sum + x[i];
   if ( x[i]<min )
      min = x[i];
   else if (x[i]>max )
      max = x[i];
}
cout << min; cout << max;
cout << sum/n;
```

Commenti

- Fare uso (moderato) di commenti
  - testo racchiuso tra i simboli:

/*	inizio commento
*/	fine commento

  - /* in un commento può comparire qualunque cosa [^* :!?] bla bla bla */

Dichiarazione di costanti simboliche

- La dichiarazione:

  `const` *tipo nome = espressione costante*`;`

  associa al *nome* il valore dell'*espressione costante*

- Esempi:

```
const double PIGRECO = 3.1415;
const int INFINITO = 1000000000;
const char BLANK = ' ';
```

Vantaggi di usare le costanti simboliche

- Le costanti compaiono sotto forma di un nome simbolico
- Il valore delle costanti può essere cambiato modificando solo la dichiarazione corrispondente
- Il programma risulta *parametrico* rispetto alle costanti che usa

Esempi

```
const double  PIGRECO = 3.1415;
      . . .
circonferenza = 2*PIGRECO*raggio
area_cerchio = PIGRECO*raggio*raggio;
```

è la stessa cosa che scrivere:

```
      . . .
circonferenza = 2*3.1415*raggio
area_cerchio = 3.1415*raggio*raggio;
```

Esempi

Definizione delle dimensioni degli array:

```
const int MAX_LEN = 1000;
    . . .
int a[MAX_LEN], b[2*MAXLEN];
```

è la stessa cosa che scrivere:

```
    . . .
int a[1000], b[2*1000];
```

Definizione di nuovi tipi

- La dichiarazione:

  `typedef` *tipo nome*`;`

  permette di usare il *nome* indicare il *tipo*

- Esempi:

```
typedef float reale;
typedef double reale_preciso;
typedef unsigned naturale;
```

Esempi

Per dicharare variabili:

```
typedef unsigned naturale;
naturale N, M;
```

Per dare un nome a un “tipo array”:

```
const int STR_LEN = 20;
typedef int stringa[20];
stringa nome1, nome2;
```