Laboratorio di Programmazione

08:45 – 12:30 lunedì – aula Omega e Sigma

08:45 – 10:30 martedì – aula Omega e Delta

Lezione 3 – Buone norme

Marco Anisetti (teoria)
Dipartimento di Informatica
marco.anisetti@unimi.it

Matteo Luperto (lab. turno A)
Dipartimento di Informatica
matteo.luperto@unimi.it

Nicola Bena (lab. turno B)
Dipartimento di Informatica
nicola.bena@unimi.it

Costrutto `switch`

```
int a = …;
switch(a){
      case 0:
            /* operazione */
            break;
      case 1:
            /* operazione */
            break;
      …
      default:
            /* operazione */
}
```

Costrutto `switch`

```
int a = …;
switch(a){
      case 0:
            /* operazione */
            break;
      case 1:
            /* operazione */
            break;
      …
      default:
            /* operazione */
}
```

Uguaglianza tra la variabile `a` oggetto dello `switch` e l'espressione che segue il `case`

(Insieme di) istruzione eseguita quando l'uguaglianza è verificata

Interrompe la valutazione! Altrimenti esegue tutte le istruzioni successive fino al primo `break`

Ricordarsi di usare i `break` !

Costrutto `switch`

```
int a = …;
switch(a){
     case 0:
     case 1:
          /* operazione */
          break;
     …
     default:
          /* operazione */
}
```

Raggruppamento di diverse condizioni

Usate lo `switch` con attenzione

Costrutto `while`

```
int i = 0;
while ( i < n ) {
     /* calcoli */
     i++;
}
```

- Struttura tipica
  - Variabile *indice* `i` inizializzata a un valore iniziale
  - Espressione nel `while` usa la variabile indice (es confronto con soglia)
  - Valore della variabile indice modificata in modo che l'espressione del `while` diventi false

Costrutto `while`

```
int i = 0;
while ( i < n ) {
    /* calcoli */
    i++;
}
```

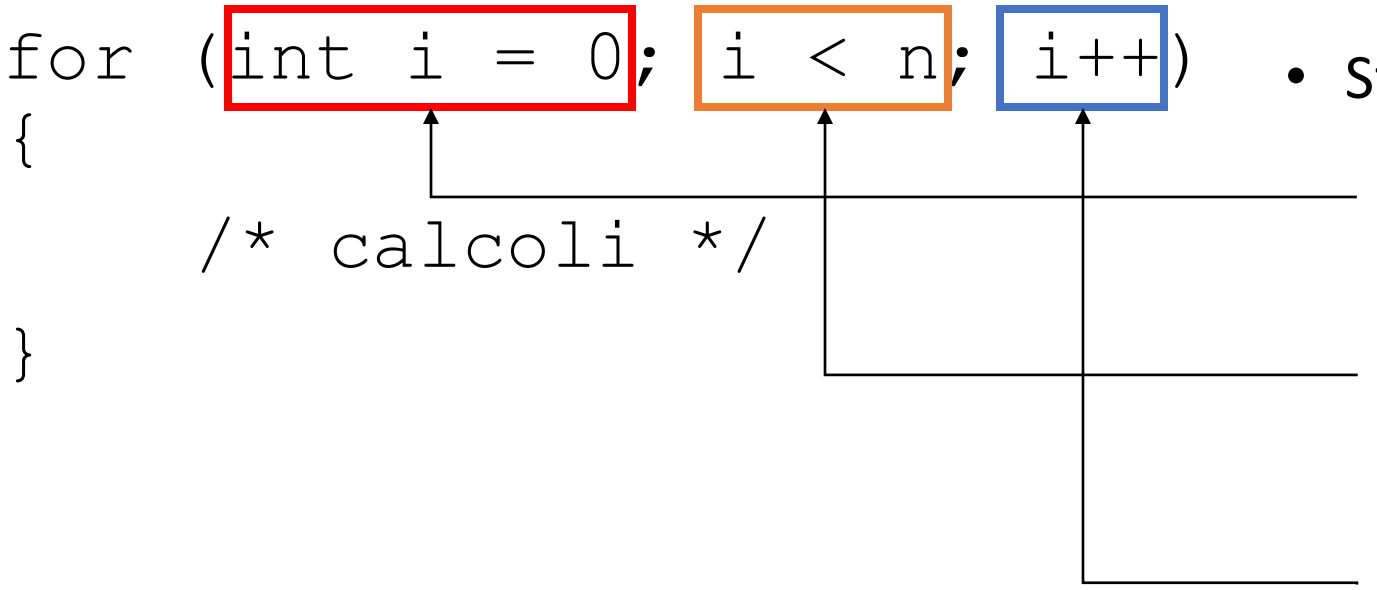
- Struttura tipica
  - Variabile *indice* `i` inizializzata a un valore iniziale
  - Espressione nel `while` usa la variabile indice (es confronto con soglia)
  - Valore della variabile indice modificata in modo che l'espressione del `while` diventi falsa (`0`)

Costrutto `for`

```
for (int i = 0; i < n; i++)
{
    /* calcoli */
}
```

- Struttura tipica
  - Variabile *indice* `i` inizializzata a un valore iniziale
  - Espressione che identifica la condizione per cui si rimane nel ciclo
  - Modifica variabile indice → eseguita per ultima alla fine del ciclo

Costrutto `for`



- Struttura tipica
  - Variabile *indice* `i` inizializzata a un valore iniziale
  - Espressione che identifica la condizione per cui si rimane nel ciclo
  - Modifica variabile indice → eseguita per ultima alla fine del ciclo

Le variabili sentinella o *flag*

- Variabili che identificano se un dato evento è successo o no
- Si possono usare variabili `boolean` o `int` (1/0)

Uso:

- Se un risultato è stato ottenuto
- Se input è valido (o meno)
- Se una condizione è verificata (o meno)

Note:

- Inizializzarle (e reinizializzarle all'interno dei cicli, se serve)
- Dare nomi significativi

Le variabili sentinella o *flag*

Esempio:

Scrivere un programma che chiede all'utente, in un numero prefissato di tentativi, di indovinare un valore intero che viene deciso a compile time.

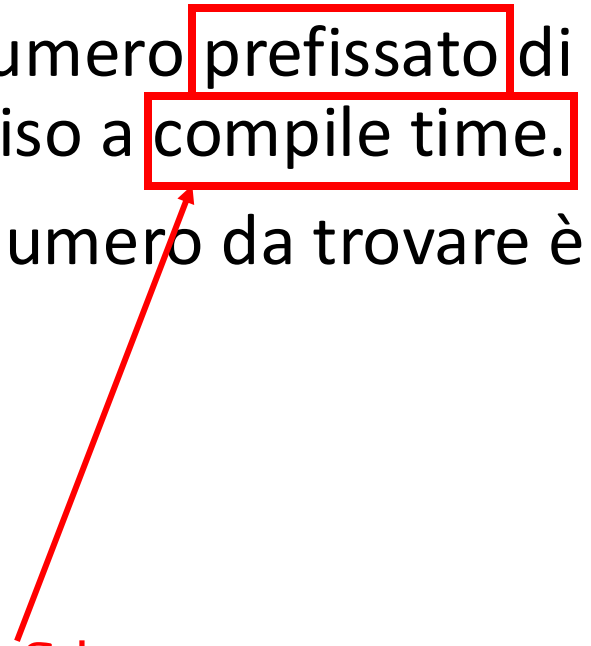
Per ogni tentativo, il programma avvisa l'utente se il numero da trovare è maggiore o minore del tentativo.

Le variabili sentinella o *flag*

Esempio:

Scrivere un programma che chiede all'utente, in un numero prefissato di tentativi, di indovinare un valore intero che viene deciso a compile time.

Per ogni tentativo, il programma avvisa l'utente se il numero da trovare è maggiore o minore del tentativo.



Parole chiave: usare una `define`

Le variabili sentinella o *flag*

```
int trovato = 0;

do {
        counter++;
        printf("Tentativo numero %d\nInserisci valore\n",counter);
        scanf("%d",&guess);
        if(guess > N)
                printf("valore troppo alto!\n\n");
        if(guess < N)
                printf("valore troppo basso!\n\n");
        if(guess == N)
                trovato = 1;
} while (counter < TENTATIVI && !trovato);

if (trovato)
        printf("Hai trovato il valore %d in %d tentativi\n",N,counter);
```

`trovato` è la flag

Somma dei primi *n* numeri naturali

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Parte 2. Esercizi facili con la programmazione strutturata

Tre sommatorie facili

Sia dato un intero $n \geqslant 1$. Valgono le tre identità:

(1) $$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

(2) $$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

(3) $$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

Somma dei primi *n* numeri naturali

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Sono sicuro di quello che fa il programma?

Ho il controllo di quello che fa ogni in suo passaggio?

Se la risposta è **NO**, devo collaudarlo.

Verifica e debug del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

- Collaudo i singoli pezzi del programma in maniera indipendente assicurandomi che siano corretti (più avanti: dividendolo in *funzioni - sottoprogrammi*)
- Verifico output del (sotto)programma in alcuni casi limite (es: `input=0`) e in casi semplici in cui so la soluzione
- Se possibile, simulo l'esecuzione del programma per verificarne il flusso

Divide et impera per debug del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Collaudo i singoli pezzi del programma in maniera indipendente assicurandomi che siano corretti (più avanti: dividendolo in *funzioni - sottoprogrammi*)

In questo caso è semplice: posso identificare due blocchi, dove l'utente inserisce i dati e dove vengono svolti i calcoli.

Posso verificare la loro correttezza in sequenza.

Divide et impera per debug del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    printf("Il numero letto è %d\n",n);

    return 1;
}
```

Creo una versione «semplificata» del programma dove verifico i singoli blocchi in maniera indipendente. Quando è corretto, unisco il tutto.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Divide et impera per debug del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    n = 5; // 1+2+3+4+5 = 15

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Verifico la parte di calcolo, quando possibile, con casi di cui so la risposta e con i casi limite.

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

memoria

Variabile	Valore
i	
somma	
n	
check	

Istruzione Corrente:
Valore:

stato corrente del programma

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	
somma	
n	
check	

Istruzione Corrente:
Valore:

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	
somma	
n	3
check	

Istruzione Corrente: scanf e while
Valore: utente inserisce 3

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	
somma	0
n	3
check	

Istruzione Corrente: inizializzo somma
Valore: somma = 0

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	1
somma	0
n	3
check	

Istruzione Corrente: prima iterazione del for
Valore: i=1; controllo valore i<=n

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	1
somma	**1**
n	3
check	

Istruzione Corrente: somma = somma + i
Valore: 0 + 1 = 1 → somma diventa 1

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	**2**
somma	1
n	3
check	

Istruzione Corrente: termino ciclo, incremento i; check i<=n
Valore: i++ → i=2

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	2
somma	**3**
n	3
check	

Istruzione Corrente: somma = somma + i
Valore: 1 + 2 = 3 → somma diventa 3

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	**3**
somma	3
n	3
check	

Istruzione Corrente: termino ciclo, incremento i; check i<=n
Valore: i++ → i=3

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	3
somma	**6**
n	3
check	

Istruzione Corrente: somma = somma + i
Valore: 3 + 3 = 6 → somma diventa 6

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	**4**
somma	6
n	3
check	

Istruzione Corrente: termino ciclo, incremento i; check i<=n
Valore: i++ → i=4; i<= n non verificata, esco dal ciclo.

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	4
somma	6
n	3
check	**6**

Istruzione Corrente: calcolo check
Valore: 3*(3+1)/2=3*4/2=3*2=6

Simulo esecuzione del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Variabile	Valore
i	4
somma	6
n	3
check	6

Istruzione Corrente: Stampo somma = 6

Verifica e debug del programma

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    do {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }
    while(n<=0);

    somma = 0;
    for(i=1;i<=n;i++){
        printf("Iterazione %d:\t%d+%d=\t%d\n",i,somma,i,somma+i);
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

Posso (*devo*) ispezionare quello che fa il programma; per ora, con delle `printf`

```
Inserisci la cifra:
4
Iterazione 1:   0+1=    1
Iterazione 2:   1+2=    3
Iterazione 3:   3+3=    6
Iterazione 4:   6+4=    10
La sommatoria calcolata nei due modi vale 10 10
```

Evitate la duplicazione del codice

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int i;
    int somma;
    int n;
    int check;

    printf("Inserisci la cifra:\n");
    scanf("%d",&n);

    while(n<=0) {
        printf("Inserisci la cifra:\n");
        scanf("%d",&n);
    }


    somma = 0;
    for(i=1;i<=n;i++){
        somma = somma+i;
    }

    check = n*(n+1)/2;

    printf("La sommatoria calcolata nei due modi vale %d %d\n",somma,check);

    return 1;
}
```

In questo caso il do-while è il costrutto corretto, perché altrimenti dovreste duplicare lo stesso codice due volte…

Copincollare il codice crea una propagazione di eventuali errori.
Se il codice che copiancollate ha un errore, lo duplicherete in tanti punti del codice.
Sarà poi difficile trovarli tutti dopo!
Soluzione: fare una funzione

`while` vs `do-while`

```
/* costrutto inappropriato */
scanf("%d",&n);
while (n <=0)
      scanf("%d",&n);
```

```
/* costrutto appropriato */
do
      scanf("%d",&n);
while (n <=0);
```

In questo caso è corretto il `do-while` perché evita la duplicazione del codice contenuto nel ciclo, che deve essere eseguito almeno una volta

`while` vs `do-while`

In questo caso, meglio il `while`, perché evita un doppio controllo sulla stessa condizione;
se avete dei controlli duplicati sulla stessa condizione, spesso potete evitare un controllo riorganizzando meglio il codice

```
/* costrutto inappropriato */
do {
      scanf("%d",&n);
      if (n != STOP)
            /*calcoli*/
while ( n != STOP );
```

```
/* costrutto appropriato */
scanf("%d",&n);
while ( n != STOP ) {
      /* calcoli */
      scanf("%d",&n);
}
```

while vs do-while

```
/* costrutto inappropriato */
scanf("%d",&n);
while (n <=0)
     scanf("%d",&n);
```

```
/* costrutto appropriato */
do
     scanf("%d",&n);
while (n <=0);
```

```
/* costrutto inappropriato */
do {
     scanf("%d",&n);
     if (n != STOP)
          /*calcoli*/
while ( n != STOP );
```

```
/* costrutto appropriato */
scanf("%d",&n);
while ( n != STOP ) {
     /* calcoli */
     scanf("%d",&n);
}
```

while vs for

```
/* costrutto inappropriato */
int i = 0;
while ( i < n ) {
      /* calcoli */
      i++;
}
```

```
/* costrutto appropriato */
for (int i = 0; i < n; i++)
{
      /* calcoli */
}
```

if vs switch

```
/* costrutto inappropriato */
if (a == 0)
     /* caso 1*/
if (a ==1 )
     /* caso 2*/
else if (a==3)
     /*caso 3*/
     else
          /* caso default*/
```

Avere tanti «if» non mutualmente esclusivi è un modo molto semplice per commettere degli errori a *runtime,* che sarà difficile trovare dopo.

```
/* costrutto appropriato */
switch (a)
case 0:
     /*caso 1*/
     break;
case 1:
     /*caso 2*/
     break;
case 2:
     /*caso 3*/
     break;
default :
     /* caso default */
```

Indentazione e spazi

```
/* codice poco leggibile */
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if (i>j) counter1++;
else counter2++;
```

```
/* codice leggibile */
for( i=0; i<n; i++)
     for(j=0; j<n; j++)
          if ( i>j )
               counter1++;
          else
               counter2++;
```

Trovare un errore in un programma poco leggibile è estremamente più difficile.

- Utilizzate la spaziatura e l'indentazione.
- Spezzate su più righe le righe di codice troppo lunghe.
- Usate righe vuote e commenti per spezzare il programma
- Andate a capo dopo ogni { o }
- Usate le parentesi ( ) e { } anche quando possono essere omesse, se non siete sicuri…

Nomi delle variabili e convenzioni

```
/* codice poco leggibile */
for(i=0; i<n; i++){
scanf("%d",&mario);
pippo=pippo+mario;}
eeeee=(float) pippo/n;
printf("Media: %d",eeeee);
```

```
/* codice leggibile */
somma = 0;
for(i=0; i<NVAL; i++){
      scanf("%d",&var);
      somma = somma + var;
}
media = (float) somma/NVAL;
printf("Media: %d\n",media);
```

- Nomi delle variabili in lettera minuscola in `camelCase` (es: `numMax`)
- Date nomi significativi alle variabili, che rispondano alla domanda *«cosa è?»*
- Inizializzate le variabili *quando serve*
- Usate le costanti, indicandole con `LETTERE MAIUSCOLE`
- Mettete spazi bianchi tra gli operatori:
  - `a = i*j+k*var/2;`
  - `a  = (i * j) + (k * var / 2);`

Scrivere codice bene o male

Scrivere codice «male» non è molto più veloce di scrivere codice «bene».

Ma è
- estremamente più difficile da collaudare,
- è più probabile che vi siano degli errori,
- e sarà più difficile trovarli.

In particolare, sarà difficile capire cosa fa il programma a *runtime*, specie se si comporterà in maniera anomala rispetto a quanto previsto.

I vostri programmi verranno usati e letti da altri. Facilitare la leggibilità del codice aiuterà loro (e aiuterà voi quando dovrete leggere il codice altrui).

Spesso è più semplice riscrivere *bene* un (sotto)programma da 0 piuttosto che debuggarne uno scritto *male*.

Trova l'errore:

```c
#include <stdio.h>

int main() {
int i,j;
int n;
int flag = 0;
do {
printf("Inserisci una cifra massima per calcolare i numeri primi: ");
scanf("%d", &n);
} while (n<=0);

printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2); {
for (j = 2; j * j <= i; j++)
if (n % j == 0)
flag = 0; // Il numero non è primo se è divisibile per i

if (flag) {
printf("%d ", i);
}
}

printf("\n");

return 0;
}
```

Questo programma deve stampare a schermo i numeri primi fino ad `n`, con `n` inserito dall'utente. Purtroppo però ci sono 3 errori. Riuscite a debuggarlo?