ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Ingegneria
Industriale, Elettronica e Meccanica

Collegio Didattico di Ingegneria Elettronica

Elementi di Informatica

Stile di Programmazione

Giordano Da Lozzo e Giuseppe Sansonetti

Leggibilità

Qualità del software che indica quanto facilmente il codice sorgente può essere compreso leggendolo.

Importante per la modificabilità del codice sorgente. Quasi sempre il codice che realizza un'applicazione deve essere modificato più volte dopo essere stato scritto.

Il codice sorgente viene spesso letto da persone diverse da coloro che lo hanno scritto.

Convenzioni di codifica

La leggibilità viene ottenuta seguendo delle indicazioni di carattere stilistico su come scrivere codice, dette convenzioni di codifica.

Queste indicazioni non influenzano la sintassi o la semantica del programma, ma ne favoriscono la leggibilità.

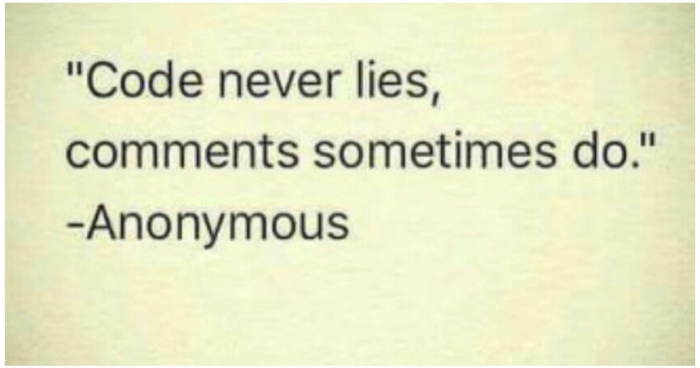
- Commenti
- Scelta dei nomi
- Indentazione
- Ordine Istruzioni

Commenti

Un commento è una frase che illustra lo scopo o il significato di una porzione di codice.

Non influenzano il significato di un programma – il compilatore li ignora (se sono scritti in maniera sintatticamente corretta).

Forniscono informazioni per la comprensione del codice, descrivendolo in maniera non dettagliata. Permettono di ricordare o comprendere il significato di una porzione di codice.



Commenti Documentazione

Descrivono lo scopo di una applicazione o di una sua funzione.

Si scrivono prima della funzione che documentano (per documentare un'applicazione il commento si scrive prima della funzione *main*)

Iniziano con il delimitatore /* e terminano con il delimitatore */

```
/* programma che legge un intero e ne stampa il doppio */
int main() {
    int intero, doppio;          // intero da leggere e suo doppio
    printf("Caro utente, introduci un intero.\n");
    scanf("%d", &intero);
    doppio = raddoppia(intero);
    printf("Il doppio del numero da te inserito %c %d.\n", 138, doppio);
}

/* funzione che riceve un intero e ne restituisce il doppio */
int raddoppia(int numero) {
    return numero * 2;
}
```

Commenti Implementazione

Descrivono delle scelte realizzative: a cosa serve questa variabile? A cosa serve questa istruzione? Quale algoritmo è realizzato da questa porzione di codice?

Si scrivono: 1) fra /* e */, prima della corrispondente porzione di codice, come i commenti documentazione; oppure 2) dopo // collocati sulla stessa linea e dopo la corrispondente porzione di codice

```
/* inizializza risultato al minimo fra x ed y */
if(x<y)
    risultato = x;
else
    risultato = y;


int numero;          // il numero da leggere
int fattoriale;      // il valore da calcolare
int i;               // variabile contatore
```

Commenti Asserzione

Descrivono proprietà che si verificano sempre in fase di esecuzione.

Si scrivono con la stessa modalità con cui si scrivono i commenti implementazione.

```
fattoriale = 1;
for(i=1; i<=numero; i++)
    fattoriale *= i;

/* adesso la variabile fattoriale memorizza il fattoriale di numero */
printf("Il fattoriale di %d %c %d\n", numero, 138, fattoriale);


massimo = x;

if(y>massimo)
    massimo = y;   // adesso massimo è il massimo fra x ed y
```

Quando scrivere commenti

documentazione

- Prima della definizione di una funzione per illustrarne lo scopo (/* … */).

implementazione

- Dopo ciascuna dichiarazione di variabile locale ad una funzione (//…).
- Prima di ciascun passo significativo di un algoritmo (/* … */) e dopo ciascuna singola istruzione che si voglia commentare (//…).

asserzione

- Quando si vuole fare un'asserzione dopo che risulta vera

Melius abundare quam deficere!

Scelta dei nomi

Variabili: sostantivi, anche composti con aggettivi. La prima parola tutta minuscola, ciascun'altra tutta minuscola a parte la prima lettera maiuscola (camel case), oppure ciascuna parola tutta minuscola separata da *under score* _ (snake case)

radiceReale	radice_reale
somma3Numeri	somma_3_numeri
cognomeNome	cognome_nome

Funzioni: verbi o «frasi verbali». Stesse regole delle variabili.

Programmi: sostantivi, anche composti con aggettivi. Ciascuna parola è tutta minuscola, tranne la prima lettera che è maiuscola.

Indentazione

Indentare vuol dire evidenziare la relazione di contenimento fra diverse porzioni di codice mediante incolonnamento.

Se una porzione di codice è contenuta in un'altra, allora è indentata, ovvero è una unità di spaziatura più a destra della porzione contenitore.

Sublime indenta automaticamente!

Elementi indentati

Dichiarazioni ed istruzioni rispetto alla funzione che li contiene

```
int main() {
→int x, y;        // gli interi da leggere
→int risultato;   // il MCD fra i due interi
```

Il corpo di una istruzione ripetitiva rispetto all'istruzione ripetitiva stessa.

```
/* fintanto che risultato non divide entrambi, decrementalo */
while(x%risultato!=0 || y%risultato!=0)
→risultato--;
```

La parte if e la parte else di una istruzione di condizionale rispetto all'istruzione condizionale stessa.

```
/* inizializza risultato al minimo fra x ed y */
if(x<y)
→risultato = x;
else
→risultato = y;
```

Elementi non indentati

Diverse funzioni nello stesso programma, diverse istruzioni nello stesso blocco e i commenti /* … */ rispetto alla porzione di codice che commentano.

```
/* programma che legge un intero e ne stampa il doppio */
int main() {
    int intero, doppio;          // intero da leggere e suo doppio
    printf("Caro utente, introduci un intero.\n");
    scanf("%d", &intero);
    doppio = raddoppia(intero);
    printf("Il doppio del numero da te inserito %c %d.\n", 138, doppio);
}

/* funzione che riceve un intero e ne restituisce il doppio */
int raddoppia(int numero) {
    return numero * 2;
}
```

Graffe dei blocchi

Si aprono come ultimo carattere della linea che precede il blocco.

Si chiudono come unico carattere della linea successiva al blocco.

```
do{
    printf("Utente, inserisci un bell'intero!\n");
    scanf("%d", &numero);
    if(numero>0)
        printf("POSITIVO!\n\n");
    if(numero<0)
        printf("NEGATIVO!\n\n");
}
```

Ordine dichiarazione ed istruzioni

Prima le dichiarazioni – poi le istruzioni.

Una sola dichiarazione o istruzione per riga.

Lo stile è importante!

Programma scritto senza stile

```
int main(
) { int numero,doppio

                    ;printf
(
                                "Caro utente inserisci un intero\n");scanf(
"%d"
                                ,
&
numero);doppio=numero                                   *
                2;printf("Il doppio del numero da te introdotto %c %d\n", 138, doppio)
;}
```

Programma scritto con stile

```
/* programma che legge un intero e ne stampa il doppio */
int main() {
    int numero;         // intero da leggere
    int doppio;         // il suo doppio

    /* INPUT */
    printf("Caro utente inserisci un intero\n");
    scanf("%d", &numero);

    /* CALCOLA IL DOPPIO */
    doppio = numero * 2;

    /* OUTPUT */
    printf("Il doppio del numero da te introdotto %c %d\n", 138, doppio);
}
```

Altre risorse

- Bellini, Guidi: Linguaggio C — 5.2 (cenni), 9.9 (cenni)