ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Ingegneria
Industriale, Elettronica e Meccanica

Collegio Didattico di Ingegneria Elettronica

Elementi di Informatica

Errori

Giordano Da Lozzo e Giuseppe Sansonetti

Errori

Frasi che non rispettano la sintassi o la semantica di C.

Sono classificabili in:

- Errori riconosciuti dal compilatore
- Errori non riconosciuti dal compilatore

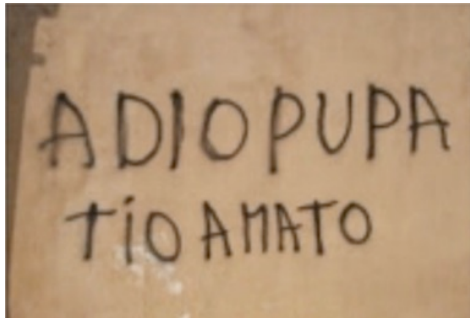
I primi sono facili da individuare e correggere: il compilatore li segnala e non genera il codice eseguibile per l'applicazione.

Errori riconosciuti dal compilatore

Sono errori che impediscono la compilazione: come effetto della presenza di tali errori, non viene generato alcun codice eseguibile.

Prima di poter eseguire il programma, il programmatore deve individuare e correggere gli errori.

- Sono errori di sintassi, ovvero errori nell'utilizzo della grammatica (sintassi) di C. Portano alla formazione di frasi mal formate.



Errori riconosciuti dal compilatore

```
int x;
x=5
printf("Valore %d", x);
```

⇒**Utilizzo sbagliato di punteggiatura**
error: expected ';' before 'printf'

```
printf(Voglio stampare questa frase);
```

⇒**Utilizzo sbagliato di punteggiatura**
error: 'Voglio' undeclared (first use in this function)
Il messaggio di errore non è sempre indicativo dell'errore stesso

```
dabol x;
x=3.14;
printf("Valore %f", x);
```

⇒**Scrittura errata di parola chiave**
error: unknown type name 'dabol'

```
float x;
x=sqrt(9,18);
printf("Valore %f", x);
```

⇒**Invocazione di funzione con numero di argomenti errato**
error: too many arguments to function 'sqrt'

Errori riconosciuti dal compilatore

```
int main() {
   x=5;
   printf("Valore %d", x);
}
```

⇒Utilizzo di variabile non dichiarata
error: ‘x’ undeclared (first use in this function)

```
prIntf("Mi pare giusto!\n");
```

⇒Invocazione di funzione che non esiste
error: undefined reference to ‘prIntf’

```
for(int i=0;i<10) {
   printf("Mi pare giusto!\n");
   i++;
}
```

⇒Istruzione di controllo che non rispetta la sintassi
error: expected ‘;’ before ‘)’ token

```
/*programma che stampa 10 volte una frase*/
int main() {
   for(int i=0;i<10;i++)
      printf("Ciao belli!\n");
   }
   printf("Fatto!\n");
}
```

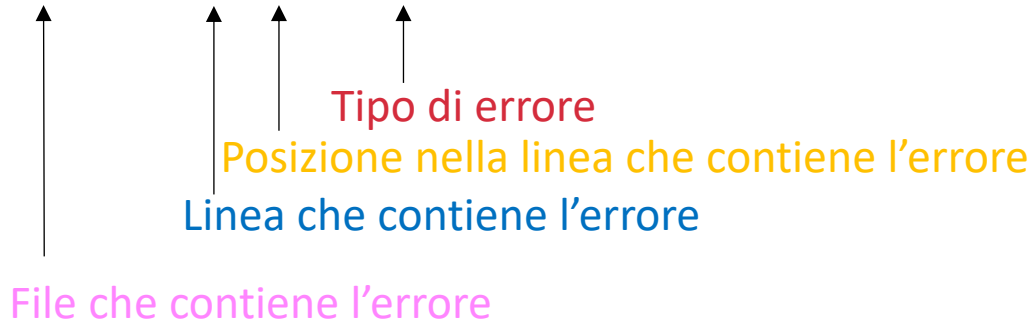
⇒Utilizzo sbagliato di punteggiatura
error: expected declaration specifiers or ‘...’ before string constant

$\Rightarrow$ ***sono fuori da main***

Errori riconosciuti dal compilatore: cosa fare?

Leggi i messaggi che ti sono stati inviati dal compilatore.

- Molto spesso ti conducono all'istruzione che contiene l'errore
- Esempio:
  main.c:6:18: error: expected ';' before ')' token



➔ Se i messaggi del compilatore non ti hanno permesso di individuare l'errore, controlla il codice dall'alto verso il basso:

- un errore all'inizio del programma può generare una sequenza di messaggi d'errore per le successive linee di codice del programma.

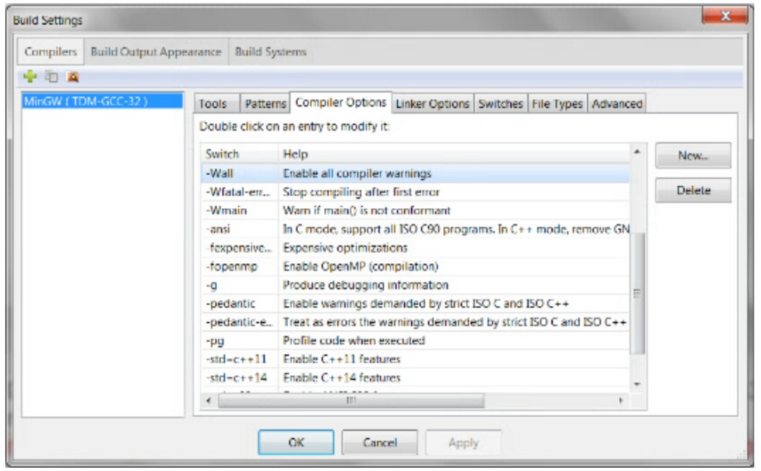
Warning

Sono segnalazioni di possibili errori: istruzioni che hanno un senso ma che il compilatore ha motivo di credere che genereranno comportamenti non voluti.

E' possibile eseguire un programma la cui compilazione ha generato dei warning: il compilatore produce un eseguibile!

E' sconsigliato ignorare i warning: probabilmente porteranno a dei risultati non voluti in fase di esecuzione.

Non esiste una classificazione precisa di quali istruzioni generano errori, quali generano warning e quali non generano messaggi una volta che vengono compilate:
infatti questo dipende dallo specifico compilatore e spesso può essere modificato a mano.



Alcuni warning

```
int x;
printf("%d", x);
```

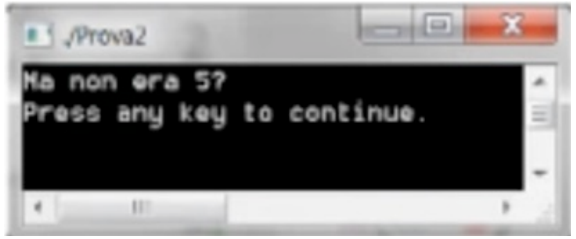
⇒Accesso a variabile non inizializzata
warning: ‘x’ is used uninitialized in this function [-Wuninitialized]

```
int x=3.1344;
printf("%d", x);
```

⇒Conversioni da tipo "più capiente" a tipo "meno capiente"
warning: conversion to ‘int’ alters ‘double’ constant value [-Wfloat-conversion]

```
int x=5;
if (x=8)
   printf("Ma non era 5?");
```

⇒Utilizzo di = invece di ==
warning: suggest parentheses around assignment used as truth value [-Wparentheses]



Per visualizzare alcuni di questi warning bisogna impostare le giuste opzioni di compilazione (-Wall per abilitare la visualizzazione di un insieme piuttosto vasto di warning, -Wextra per ulteriori warning).

Errori non riconosciuti dal compilatore

Il fatto che un programma possa essere compilato non garantisce la sua correttezza! Infatti, potrebbe succedere che i malfunzionamenti legati alla presenza di errori in un programma si manifestino solo durante la sua esecuzione.

Errori di questo tipo sono in genere chiamati bugs e l'operazione di individuazione e correzione di tali errori prende il nome di debugging.

I bugs si dividono in due tipi di errori:

- runtime errors
- errori logici

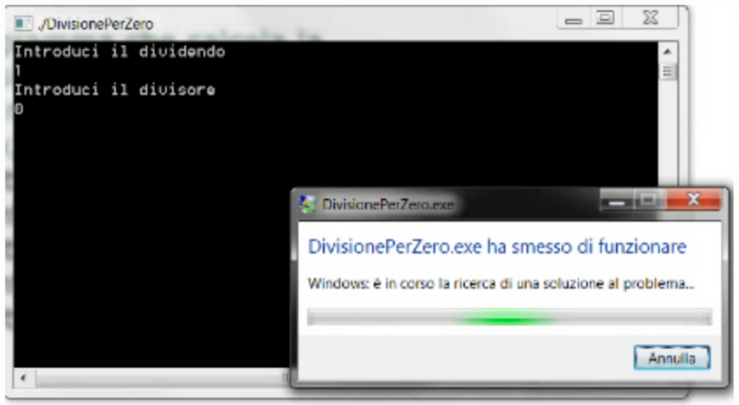
Runtime errors

Errori non riconosciuti dal compilatore che si manifestano a tempo di esecuzione, ovvero quando si esegue un programma.

Le istruzioni che generano questi errori potrebbero non generare un errore ad ogni esecuzione, ma solo qualche volta

L'effetto dell'insorgere di tali errori è (in genere) la terminazione dell'esecuzione del programma.

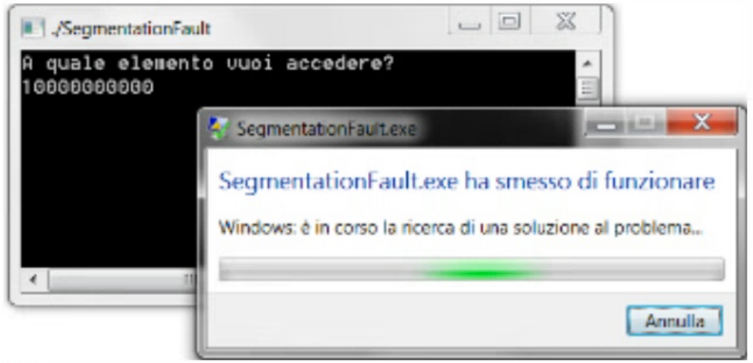
Runtime Errors



1. Divide by zero

```
/*programma che calcola la
* divisione intera fra due numeri */
int main() {
   int x,y;
   printf("Introduci il dividendo\n");
   scanf("%d", &x);
   printf("Introduci il divisore\n");
   scanf("%d", &y);
   printf("La divisione ha quoziente %d\n", x/y);
}
```

2. Segmentation fault: errore che ha luogo quando un programma tenta di accedere ad una posizione di memoria alla quale non è permesso accedere, oppure tenta di accedere in scrittura ad un'area di sola lettura.




```
int main(int argc, char **argv) {
   int sequenza[10];
   for(int i=0; i<10;i++)
      sequenza[i]=i*2;
   int indice;
   printf("A quale elemento vuoi accedere?\n");
   scanf("%d", &indice);
   printf("L'elemento ha valore %d\n"), sequenza[indice]);
}
```

3. Stack overflow: errore conseguente ad una eccessiva richiesta di memoria nello stack

```
void f() {
   f();
}

int main() {
   f();
}
```

Errori logici

Il tuo programma può essere compilato, può essere eseguito, ma non fa quello per cui è stato pensato. Ad esempio, produce risultati non corretti, oppure la sua esecuzione non termina mai.


```
/*programma che legge un numero e ne stampa il doppio*/
int main() {
   int numero;
   int 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);
}
```

Q: Dov'è l'errore logico?

```
/*programma che chiede all'utente di inserire un numero e ne calcola e stampa il fattoriale*/
int main() {
   int numero;          // il numero da leggere
   int fattoriale;      // il valore da calcolare
   int i;               // variabile contatore

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

   /*CALCOLA IL FATTORIALE. Algoritmo: moltiplica la variabile fattoriale per ogni numero
compreso tra 1 ed n */
   fattoriale = 0;          // valore iniziale
   for (i=1;i<=numero;i++)
      fattoriale *= i;

   /*OUTPUT*/
   printf("Il fattoriale di %d %c %d\n", numero, 138, fattoriale);
}
```

Q: Dov'è l'errore logico?

```
/*programma che chiede all'utente di inserire un numero e ne calcola e stampa il fattoriale*/
int main() {
   int numero;          // il numero da leggere
   int fattoriale;      // il valore da calcolare
   int i;               // variabile contatore

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

   /*CALCOLA IL FATTORIALE. Algoritmo: moltiplica la variabile fattoriale per ogni numero
compreso tra 1 ed n */
   fattoriale = 0;          // valore iniziale
   for (i=1;i<=numero;i++)
      fattoriale *= i;

   /*OUTPUT*/
   printf("Il fattoriale di %d %c %d\n", numero, 138, fattoriale);
}
```

Inizializzazione sbagliata:
Fattoriale rimane 0

Q: Dov'è l'errore logico?

```
/*programma che chiede all'utente di inserire un numero e ne calcola e stampa il fattoriale*/
int main() {
   int numero;          // il numero da leggere
   int fattoriale;      // il valore da calcolare
   int i;               // variabile contatore

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

   /*CALCOLA IL FATTORIALE. Algoritmo: moltiplica la variabile fattoriale per ogni numero
compreso tra 1 ed n */
   fattoriale = 1;          // valore iniziale
   while (i<=numero)
      fattoriale *= i;

   /*OUTPUT*/
   printf("Il fattoriale di %d %c %d\n", numero, 138, fattoriale);
}
```

Q: Dov'è l'errore logico?

Manca inizializzazione *variabile i*.

Manca incremento della *variabile i*
L'istruzione ripetitiva non termina mai.

```
/*programma che chiede all'utente di inserire un numero e ne calcola e stampa il fattoriale*/
int main() {
   int numero;          // il numero da leggere
   int fattoriale;      // il valore da calcolare
   int i;               // variabile contatore

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

   /*CALCOLA IL FATTORIALE. Algoritmo: moltiplica la variabile fattoriale per ogni numero
compreso tra 1 ed n */
   fattoriale = 1;          // valore iniziale
   while (i<=numero)
      fattoriale *= i;

   /*OUTPUT*/
   printf("Il fattoriale di %d %c %d\n", numero, 138, fattoriale);
}
```

Altre risorse

- Bellini, Guidi: Linguaggio C — 7.3 (cenni)