5 - VARIE

a)  ERRORI FREQUENTI E LORO CORREZIONE

OCCHIO! Fra i più frequenti, "tipici" errori formali nella redazione di un programma citiamo i seguenti:

- **dimenticare il ";" alla fine di una istruzione**.
  OGNI ISTRUZIONE O DICHIARAZIONE (NOTA)
  DEVE OBBLIGATORIAMENTE TERMINARE COL ";" !!!
  Uniche eccezioni l'*end* finale (seguito da un ".") e, volendo, i vari *begin* (";" facoltativo)

  Il **messaggio di errore** che compare è in questo caso:
  ";" expected ossia **ci si aspettava un ";"**
  Nel messaggio troviamo pure l'**indicazione di "cosa è stato invece trovato (found)".**

  NOTA: le "dichiarazioni" sono, ad esempio:
  il "program …", la "uses crt", la specificazione del *tipo* per ogni variabile.

  QUANDO IL COMPILATORE SEGNALA UN ERRORE,
  simultaneamente indica pure
  LA RIGA E LA COLONNA IN CUI E' STATO RISCONTRATO L'ERRORE !!!
  … E QUESTO TI AIUTA TANTISSIMO, PER LA CORREZIONE !!!

- **mettere il ";" prima dell'ELSE** di una *if … then … else …* (in questo caso, non ci vuole!)
  Qui il **messaggio di errore** è:
  ";" expected" but "ELSE" found

- **dimenticare il "." dopo l' "end" conclusivo**
  Messaggio di errore:
  "." expected but "end of file" found

b) OPERAZIONI E SIMBOLI MATEMATICI

somma, sottrazione	**+, –**		
moltiplicazione	*		
divisione	/	OCCHIO! Il risultato di una "/" va sempre incasellato in una variabile di tipo *reale*! … Altrimenti la compilazione si arresterebbe, con un messaggio di errore.	
divisione intera	**div**	Può operare solo con variabili di tipo *intero*: *integer* o *longint*.	
resto della divisione intera	**mod**	(può operare solo con variabili di tipo *intero*!)	
radice quadrata	**sqrt**	**Esempio: y:=sqrt(x)**	
Altre operazioni (es. potenza …)	Volutamente non ne parliamo a questo livello. Richiedono una conoscenza più avanzata di Free Pascal. A questo proposito, i vari "HELP" si possono scaricare da www.freepascal.org/down/docs/docs.html Fra l'altro, una potenza ad esponente intero si può ottenere servendosi della moltiplicazione; se l'esponente è una *variabile* intera possiamo ottenere lo scopo con un programmino, che sarà richiesto a suo tempo come esercizio.		
diverso da	**<>**		
minore o uguale, maggiore o ug.	**<=, >=**		
parte intera di	**int**	**Esempi: int(4.72) dà come risultato 4; int(–3.8) dà –3**	

Esercizio 2) Realizza un programma PASCAL che, letto in ingresso un numero x, fornisca in output:

- ♪ la radice quadrata di x, in notazione NON ESPONENZIALE, se x>=0;
- ♫ la scritta "Mi spiace, ma non esiste la radice quadrata di un numero negativo" se x<0.

c) L'OVERFLOW (= traboccamento)

Si chiama così la **fuoriuscita di una variabile dal suo "range" (=campo di variabilità)**.
Ad esempio, se abbiamo dichiarato una variabile di tipo *integer* (*range* da –32768 a +32767)
e a questa variabile viene attribuito, o direttamente da una istruzione di assegnazione,
o in input, o per l'effetto di un calcolo, un valore fuori dal *range* (poniamo: il valore 50000),
allora in fase di compilazione o di esecuzione verrà segnalato un errore: il programma non funzionerà.

d) COME SALTARE UNA RIGA SUL MONITOR, IN FASE DI ESECUZIONE

L'istruzione
writeln
provoca l'effetto di **spostare il cursore all'inizio della riga successiva**;
se il cursore SI TROVA GIA' all'inizio di una riga vuota,
l'effetto sarà di passare all'inizio della riga successiva a questa, e quindi di **saltare una riga**.

e) APOSTROFO NELLE STRINGHE: CHE GUAIO! IL COMPUTER LO CONFONDEREBBE CON IL SIMBOLO DI "FINE STRINGA"!

Come fare, allora? I creatori del PASCAL hanno previsto che si operi come nell'esempio seguente:

write ('L''amicizia è una cosa stupenda')

Per realizzare l'**apostrofo all'interno di una stringa** basterà dunque digitare il **doppio apice**.

f) TESTO COLORATO IN OUTPUT

L'istruzione per ottenere in output un testo colorato è
textcolor (n),
dove **n è il codice del colore desiderato**
(**0 ≤ n ≤ 255, ma oltre il 15 si ripete la sequenza di colori precedente**).

Ad esempio:
9 = blu, 10 = verde, 11 = azzurro,
12 = rosso, 14 = giallo, 15 = bianco.

Per conoscere le altre corrispondenze codice-colore, possiamo scrivere un programmino apposito, ad es. un programma che, letto in input un intero, lo restituisca in output, colorato del colore che gli corrisponde →

```
program codici_dei_vari_colori;   uses crt;
var n: integer;
begin
     clrscr;
     writeln ('Scrivi un intero fra 0 e 15.');
     write ('n = ');  readln (n);
     writeln ('Colore corrispondente:');
     textcolor (n);  write (n);  readln;
end.
```

g) L' EFFETTO RITARDO

L'istruzione **delay (n)**, dove n è un numero di tipo intero,
ordina al computer di attendere n "unità di tempo" prima di eseguire l'istruzione successiva.
L' "unità di tempo" dovrebbe essere in teoria di 1 millesimo di secondo, ma in realtà differisce un po' da un computer all'altro. Si faranno delle prove, poi si sceglierà n a seconda dell'effetto che si vuol realizzare.

h) I "COMMENTI"

Nel "listato" (=sequenza delle istruzioni) di un programma si possono scrivere dei "**commenti**", **che verranno ignorati nella fase di "compilazione" (=trascrizione in linguaggio macchina).**

I commenti devono essere inseriti

- **fra "parentesi asteriscate": (* questo è un commento *)**
- **oppure fra parentesi graffe: { questo è un commento }**

OSSERVAZIONE - Nel linguaggio C le graffe hanno invece un ruolo completamente diverso: stanno a indicare inizio blocco e fine blocco (hanno cioè la funzione che in PASCAL compete a BEGIN e END)

i) LA SELEZIONE MULTIPLA: CASE … OF …

Quando le alternative sono più di due, potrà essere utile la CASE … OF … ,
una struttura di **SELEZIONE MULTIPLA** illustrata dal seguente esempio:

```
CASE punteggio OF
        0, 1, 2: writeln ('Vai a zappare');
        3, 4: writeln ('Gravemente insufficiente');
        5: writeln ('Insufficiente');
        6, 7, 8, 9, 10: writeln ('Esame superato');
END;
```