11 - ESERCIZI VARI

Può darsi che in certi casi tu trovi opportuno tracciare
il diagramma di flusso dell'algoritmo prima di metterti a scrivere il programma;
in tal caso, ricorda la forma che convenzionalmente si assegna ai vari blocchi:
OVALE per INIZIO/FINE; PARALLELOGRAMMO per INPUT/OUTPUT;
RETTANGOLO per ELABORAZIONE; ROMBO per SELEZIONE.

Iniziare o meno col diagramma di flusso
dipende dalla tua volontà, oltre che dal tipo di esercizio.

Esercizio 22)

Si simula ripetutamente il lancio di un dado, mediante l'istruzione x:=random(6)+1,
e ci si arresta non appena esce il numero 6. Quanti "colpi" sono stati effettuati in totale?

Esercizio 23)

PREMESSA
Un intero si dice "perfetto" se è uguale alla somma dei suoi divisori,
inclusa l'unità ma escluso il numero stesso:
ad esempio 28 è un numero perfetto,
perché ha come divisori (a parte sé stesso) 1, 2, 4, 7, 14, e $1+2+4+7+14=28$.

Letto in input un intero n, stabilire se n è un numero "perfetto".

Esercizio 24)

Elenco **dei numeri perfetti $\leq$ m**, con m letto in ingresso.

Esercizio 25)

Elenco degli interi da a fino a b, con a, b letti in input,
coi numeri PRIMI scritti in rosso, i PERFETTI in verde (textcolor).
Utilizza una opportuna "ampiezza" (vedi paragrafo 3) per collocare i numeri ordinatamente sul monitor:
ad es., write(n:10) fa scrivere l'intero n allineato a destra, in un campo la cui *ampiezza* è di 10 caratteri.

Esercizio 26)

Il computer "inventa" un intero pseudocasuale non superiore a 100
e invita l'utente a indovinarlo, dandogli 7 tentativi al massimo.
Ogni volta che l'utente "tenta", il computer gli dice
"più alto…" oppure "più basso…" o "giusto!!!" a seconda dei casi.
Il gioco si deve arrestare quando l'utente indovina,
oppure quando sono stati effettuati i 7 tentativi concessi.

Esercizio 27)

Si lancia un dado finché escano due risultati consecutivi uguali,
e si conta il numero di colpi effettuato.

Esercizio 28)

Programma "**fuochi_di_artificio**". Output desiderato:

……..BOOM!
…BOOM!
……BOOM!
…

con 10 "esplosioni"
e un numero pseudocasuale di puntini prima di ogni esplosione.
Ci deve essere un ritardo opportuno fra la comparsa di ciascun puntino e il successivo,
nonché fra una riga e la successiva.
Vogliamo far comparire la scritta BOOM! in colore, anch'esso (pseudo)casuale.
Ricordiamo che a tale scopo si ricorre all'istruzione textcolor (n) dove n è il codice del colore desiderato.

Esercizio 29)

Scrivi un programma che, letti in ingresso due interi positivi y, k, calcoli la **somma algebrica**

$$\mathbf{1 - y + y^2 - y^3 + \ldots \pm y^k}$$

Ad esempio, se y=2 e k=5, l'output dovrà essere il numero –21.

Esercizio 30)

PREMESSA: le equazioni di 2° grado

$x_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ è la FORMULA RISOLUTIVA DELL'EQUAZIONE $ax^2 + bx + c = 0$

Esempio di applicazione della formula:

$$3x^2 + x - 2 = 0 \qquad (a = 3,\ \ b = 1,\ \ c = -2)$$

$$x_{1,2} = \frac{-1 \pm \sqrt{1^2 - 4 \cdot 3 \cdot (-2)}}{2 \cdot 3} = \frac{-1 \pm \sqrt{1 + 24}}{6} = \frac{-1 \pm \sqrt{25}}{6} = \frac{-1 \pm 5}{6} = \begin{cases} \dfrac{-1-5}{6} = -\dfrac{6}{6} = -1 \\ \dfrac{-1+5}{6} = \dfrac{4}{6} = \dfrac{2}{3} \end{cases}$$

La quantità $\mathbf{b^2 - 4ac}$ che sta sotto radice quadrata nella formula risolutiva, è chiamata "delta" o "discriminante" e indicata col simbolo Δ (la lettera greca "*delta*" maiuscola).

Sono possibili tre casi:

- se $\Delta > 0$, l'equazione ha due soluzioni distinte
- se $\Delta = 0$, l'equazione ha una sola soluzione (si può anche dire che ha "due soluzioni coincidenti")
- se $\Delta < 0$, l'equazione non ha nessuna soluzione (= è impossibile)

Scrivi un programma che,
letti in input i 3 coefficienti a, b, c di un'equazione di 2° grado, ne fornisca le soluzioni.

Converrà far calcolare innanzitutto il delta; poi:

```
if delta > 0 then … ;
if delta = 0 then … ;
if delta < 0 then …
```

Le variabili a, b, c, delta dovranno essere di tipo *real* e si vuole l'output in notazione non esponenziale.

Esercizio 31)

Scrivi un programma per la risoluzione di un sistema di 1° grado **a*x+b*y=c, a1*x+b1*y=c1** **col metodo di Cramer**.

L'utente verrà invitato a inserire semplicemente i 6 numeri a, b, c, a1, b1, c1.

Esercizio 32)

Si chiama "**successione di Fibonacci**" la sequenza 0 1 1 2 3 5 8 13 21 …
costruita nel modo seguente:

- i primi due numeri della sequenza sono 0 e 1 rispettivamente;
- a partire dal terzo, ciascun termine della sequenza è ottenuto sommando i due termini che lo precedono.

Scrivi **un programma che, letto in input n, scriva i primi n termini della successione di Fibonacci.**

Esercizio 33)

Un metodo per calcolare la radice cubica di un numero, approssimata per difetto a meno di 0.1 (cioè, con la prima cifra decimale esatta), è il seguente:

i) dato il radicando x, si parte da un valore intero y che sia una approssimazione per difetto di $\sqrt[3]{x}$;
ii) poi si passa a considerare, via via, i numeri y; y+0.1; y+0.2; y+0.3; …
ciascuno di questi numeri viene elevato al cubo, finché il risultato superi x.

Allora il valore cercato sarà il *penultimo* fra i numeri considerati.

Ad esempio, data l'operazione $\sqrt[3]{70}$, si partirà da $y = 4$ e si farà:

$4^3 = 64 < 70$; poi $4.1^3 = 68.921 < 70$; poi $4.2^3 = 74.088 > 70$; quindi il valore cercato è 4.1.

Scrivi un programma che, letti in input il radicando x (la variabile x dovrà essere di tipo *real*)
e un numero intero, fornito dall'utente, che sia approssimazione per difetto di $\sqrt[3]{x}$,
determini e mandi in output il valore $y = \sqrt[3]{x}$ approssimato per difetto a meno di 0.1.

AVVERTENZA (INTERESSANTE):
può darsi che a volte il valore in output non sia quello corretto.
Ad esempio, con x = 125, l'output anziché essere 5.0 potrà essere inaspettatamente 5.1.
Per comprendere il motivo di questo strano fenomeno, vai a rileggere il paragrafo 7b a pag. 11.

Esercizio 34)

Un intero n letto in input è sottoposto alla seguente procedura:

- lo si fa diventare $3n+1$ se è dispari;
- lo si fa diventare la metà, ossia $n \text{ div } 2$, se è pari.

Il procedimento viene poi iterato sul nuovo numero ottenuto,
fino ad arrestarsi quando si ottenga il valore 1.
Ad esempio, se $n = 7$, in questo modo si genera la sequenza:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

I numeri con cui si è provato finora ad innescare questa procedura iterativa si sono tutti "abbattuti a 1"
dopo un numero più o meno alto di passi;
la comunità matematica sta tentando di provare che questo "abbattersi a 1"
deve necessariamente aver luogo per *ciascun* numero intero,
ma fino ad oggi questa congettura non è ancora stata dimostrata.

**Scrivi un programma PASCAL che, letto in input un intero n fornito dall'utente,
mandi in output la successione dei numeri che si ottengono sottoponendo n alla procedura descritta.
L'algoritmo si deve arrestare quando n diventa uguale a 1,
e deve contare il numero di passi che sono stati necessari.**

Esercizio 35)

Se tre interi a, b, c sono tali che $a^2 + b^2 = c^2$,
allora si dice che a, b, c costituiscono una "terna pitagorica".

**Utilizzando delle strutture FOR … DO … "annidate",
dovresti riuscire a scrivere un programma che fornisca in output un elenco di terne pitagoriche;
ad es., tutte le terne pitagoriche nelle quali c sia non superiore ad un numero N scelto dall'utente.**

Esercizio 36)

Una terna pitagorica si dice "fondamentale"
se i suoi 3 elementi sono numeri primi fra loro (= privi di divisori comuni).
Ad esempio, la terna (5, 12, 13) è fondamentale, mentre la (6, 8, 10) non lo è.

**Modifica il programma precedente in modo che le terne fondamentali
compaiano, nell'elenco, scritte in rosso: textcolor (12).**

Tieni conto del fatto che, in una terna (a, b, c) tale che $a^2 + b^2 = c^2$,
i tre numeri a, b, c sono primi fra loro se e solo se sono primi fra loro due qualsiasi di essi
(sapresti giustificare questa affermazione?)

Esercizio 37)

Si lancia n volte una moneta, con n letto in ingresso.
Si vuole stabilire quale è stato il **numero massimo di risultati consecutivi uguali**.
Ad esempio, se la successione dei lanci è stata

1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 1,

la risposta è "4".

Esercizio 38)

Per noi esseri umani è facile, **letto un numero intero a, calcolare quanto vale la somma delle sue cifre**.
Ad esempio, se $a = 30227$, la somma delle cifre di a è 14.
Non è invece altrettanto immediato far sì che *il computer*, dopo aver letto da tastiera un intero a
per effetto di un'istruzione read (a) contenuta in un programma PASCAL,
calcoli il valore della somma delle cifre di a.
Voglio dire:
se si comunicano al computer le cifre del numero UNA PER UNA, allora il problema è banale;
ma se invece si digita alla tastiera IL NUMERO a,
e il computer, eseguendo una read (a), lo acquisisce in memoria "tutto in una volta",
allora far calcolare dal computer la somma delle cifre di a è un po' più complicato.
Ci vuoi provare?

Esercizio 38')

**Esistono dei numeri naturali, non superiori a 10000, che siano uguali al
CUBO DELLA SOMMA DELLE PROPRIE CIFRE**?
In caso affermativo, quali e quanti sono?
Scrivi un programma PASCAL che possa fornire la risposta.