## Scheda di lavoro N. 4

Questa è la *scheda degli omogeneizzati*. Si tratta di scrivere alcuni programmini elementari che utilizzino le istruzioni `if`, `for` e `if ... else`, oltre a qualche vettore. Almeno nei primi casi ho cercato di descrivere in modo dettagliato l'algoritmo, in modo da facilitarti il compito e mostrarti quale sia il lavoro che devi fare sistematicamente tutte le volte che ti trovi a scrivere un programma. Per alcuni programmi troverai anche la soluzione dell'esercizio tra i files del dischetto di laboratorio. Ma non te la metto a disposizione semplicemente perché tu la esegua e ti illuda di aver capito come è fatto il programma: l'esercizio consiste nello *scriverlo*, il programma.

Avvertenze importanti:

- Prima di metterti a picchiare sui tasti rifletti e metti a punto un algoritmo. Questa è la fase più impegnativa: meglio farlo stando a dovuta distanza da video, tastiera e mouse.
- Per le prime volte ti sarà anche utile scrivere il programma su carta prima di batterlo in un file. Ti renderai conto che scrivere un programma non è come scrivere una lettera: è invece un continuo spostarsi su e giù per aggiungere, togliere, correggere. Meglio munirsi di matita e gomma: le cancellazioni con la penna lasciano troppe tracce che confondono il cammino.
- Quando batti il programma fa ben attenzione alla sintassi: il compilatore è dotato di scarsa fantasia, e basta dimenticare un punto e virgola o una parentesi per mandarlo in confusione.
- Nello scrivere il programma impara a curare anche l'estetica: utilizzare con coerenza l'allineamento delle istruzioni facilita notevolmente la lettura del programma e la ricerca di errori.
- Infine, non meravigliarti se scoprirai quanto sia facile commettere errori nella programmazione: i bravi programmatori non sono quelli che sanno buttar giù delle istruzioni più o meno sensate; sono quelli che sanno trovare gli errori!

**1.** Scrivi un programma che:

- legge un numero non negativo;
- ne calcola il fattoriale;
- scrive il risultato sul terminale.

Ricorda che $0! = 1$, quindi devi prevedere anche il valore zero. Dovrai scrivere due versioni distinte: una che usa un ciclo controllato con `if ... goto`, la seconda che utilizza un ciclo `for`. Troverai tutte le indicazioni necessarie nel paragrafo 3.1 delle *Note su linguaggio C e dintorni*.

Eccoti lo schema del programma:

(i) Invia un prompt sul terminale e leggi un intero `k`.
(ii) Se `k` è negativo segnala errore e termina il programma, altrimenti prosegui.
(iii) Poni il risultato `n=1`;
(iv) Se `k > 1` esegui il ciclo di calcolo:
  (iv.a) poni un contatore `j=2`;
  (iv.b) moltiplica `n` per `j`;

(iv.c) incrementa `j`; se `j` non ha superato `k` torna al passo (iv.b), altrimenti

(v) il risultato è in `n`. Stampa e termina il programma.

Non ti resta che tradurre questo algoritmo in istruzioni, usando un'etichetta e le istruzioni `if ... goto`. Trovi un esempio nel file `prog_fatt_if.c`

Per scrivere il programma con un ciclo `for` puoi usare lo stesso algoritmo. Oppure può essere conveniente eseguire il calcolo a rovescio: poni inizialmente `n=k` e lo moltiplichi per `k-1`, `k-2`,... `1`. A te la scelta, e l'onere di scrivere l'algoritmo. Trovi un esempio nel file `prog_fatt_for.c`

Una volta che il programma funziona prova a vedere cosa succede quando gli chiedi di calcolare il fattoriale di un numero superiore a 13 (supponendo che tu stia lavorando con dati `int`).

**2.** Scrivi un programma che calcoli il coefficiente binomiale $\binom{n}{k}$. Ti ricordo la definizione:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \;. \tag{1}$$

Dovrai:

- ∘ leggere due interi non negativi $n, k$, e verificare che sia $0 \le k \le n$;
- ∘ scrivere un ciclo che calcoli il coefficiente binomiale;
- ∘ scrivere il risultato.

Il programma dovrà terminare con una segnalazione di errore se le condizioni su $n, k$ non sono soddisfatte.

Veniamo all'algoritmo. Probabilmente la prima idea che ti verrà in mente sarà quella di sfruttare il calcolo del fattoriale che hai appena programmato. Ma c'è una forte controindicazione: il calcolo del fattoriale provoca facilmente un overflow (come dovresti aver visto, se hai seguito le indicazioni del primo esercizio). Meglio usare la formula

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1} \;.$$

Inoltre, la formula (1) è simmetrica rispetto a $k$ e $n-k$; quindi conviene sostituire comunque $k$ con il minimo tra $k$ e $n-k$. Infine, ti ricordo che se uno dei due numeri $k$ (minimizzato) e $n$ è nullo il coefficiente binomiale vale 1.

Per la lettura dei dati e la scrittura dei risultati ormai dovresti sapere come fare. Quindi ti illustro solo l'algoritmo per il calcolo; chiamerò b il risultato.

(i) Se `k > n-k` sostituisci `k` con `n-k`;

(ii) inizializza il risultato con `b=1`;

(iii) se `k > 0` esegui il ciclo:

(iii.a) poni il contatore `j=1`;

(iii.b) se `j` non ha superato `k` moltiplica il risultato `b` per (n-k+j) e dividilo per `j`;

(iii.c) incrementa `j` e torna al passo (iii.b);

(iv) il risultato è in `b`.

**Attenzione:** capita frequentemente che i principianti scrivano un test del tipo
`if(0 <= k <= n) ...` (tutto va bene)

Nulla di più sbagliato, anche se il compilatore non segnala errore. Il problema è che l'espressione è sintatticamente corretta, ma il compilatore la intende a modo suo. Trattandosi di due operazioni logiche, le esegue in fila: prima valuta l'espressione `0 <= k` e trova 0 se è falso e 1 se è vero; poi confronta il risultato con `n`. Ad esempio, se `k=3` e `n=2` prima valuta `0 <= 3`, che dà come risultato 1 (vero), poi valuta `1 <= 2`, che è vero, e conclude che l'espressione ha il valore vero. Non è quello che ti aspettavi. Se non ci credi, prova a leggere ed eseguire il programma `errore_if.c` che trovi sul dischetto.

Il test corretto si scrive:

`if(0<=k && k<=n) ...` (tutto va bene)

Detto questo, dovresti avere informazioni sufficienti per cimentarti nella scrittura del programma. Se ti trovi in difficoltà, prova a guardare il file `prog_binom.c` nel directory `dischetto`.

**3.** *Il crivello di Eratostene.* Si tratta di un metodo, ideato appunto da Eratostene, per costruire la tabella dei numeri primi. Si prepara una tabella che contiene tutti i numeri fino ad un $N$ fissato (ad esempio, $N = 100$). Poi si cancellano i multipli di 2, i multipli di 3 &c, fino ai multipli di $N/2$. È inutile proseguire oltre $N/2$, ovviamente. Quelli che restano sono i numeri primi cercati.

Problema: scrivi un programma che costruisca la tabella e stampi i numeri primi da 1 a 100.

Analizziamo il programma in dettaglio. Iniziamo con l'identifiare tre operazioni da eseguire sequenzialmente:

(1) l'allocazione ed il riempimento della tabella;
(2) l'eliminazione di tutti i multipli di $2, 3, \ldots, n/2$;
(3) la stampa di quello che è rimasto.

Ed ora, esaminiamo in maggior dettaglio ciascuna operazione. Chiamerò `primi` il vettore che contiene la tabella.

1. *Allocazione e riempimento.* La tabella può essere contenuta in un vettore di `int`. Se voglio $n$ numeri sarà conveniente allocare un vettore di $n+1$ elementi, tenuto conto che in C gli indici partono da 0. Poi riempirò il vettore assegnando all'elemento $j$–esimo il valore $j$. Per questo sarà utile un ciclo for del tipo

```
for(j=0; j<n; j++) primi[j]=j;
```

2. *Eliminazione dei multipli.* Qui occorre realizzare:
   (i) un ciclo for con un indice $j$ che va da 2 a $n/2$;
   (ii) un secondo ciclo, interno al primo, con un indice $k$ che parte da $2j$ e arriva al massimo ad $n$, incrementandosi ogni volta di $j$. Ad ogni passo azzera l'elemento $k$–esimo del vettore.

Le istruzioni saranno del tipo

```
for(j=2;j<n/2;j++) {
  for(k=j+j; k<n; k=k+j) primi[k]=0; }
```

3. *Stampa.* Qui basta un ciclo for che fa scorrere ciascun elemento della tabella; se non è zero lo stampa, altrimenti lo ignora.

A te ora il compito di scrivere il programma. Se vuoi un appoggio morale, guarda il programma `eratostene.c` sul dischetto di laboratorio.

**4.** Il vettore che hai generato col crivello di Eratostene conterrà molti elementi nulli, e come tali inutili. Sapresti compattarli eliminando tutti gli zeri e dicendo alla fine quanti ne sono rimasti? Eccoti qualche suggerimento.

Il metodo più banale consiste nell'allocare un secondo vettore e trasportarvi tutti gli elementi non nulli. Occorre un ciclo con due indici che chiamerò `j` e `k`. Chiamerò `primi1` il vettore destinatario della tabella compatta. Ecco lo schema:

(i) inizializzazione del ciclo: poni `j=k=0`;
(ii) il ciclo termina quando `j` ha raggiunto `n`, il numero di elementi del vettore `primi`.
(iii) Istruzioni del ciclo: se l'elemento `primi[j]` è non nullo memorizzalo in `primi1[k]` ed incrementa `k`; altrimenti non se ne fa nulla;
(iv) Istruzione di fine ciclo: incrementa `j`.

Alla fine del ciclo `k` contiene il numero di elementi utili del vettore `primi1`. Il resto del vettore si può considerate inutilizzato. Per completezza, eccoti le istruzioni:

```
for(j=k=0; j<n; j++) {
  if(primi[j]!=0){
    primi1[k]=primi[j]; k++; }}
```

Un'ultima osservazione: se riguardi bene l'algoritmo e le istruzioni qui sopra ti renderai conto che non è affatto necessario usare un secondo vettore: tutto funziona anche se compatti il vettore `primi` sul posto. In altre parole, se nelle istruzioni qui sopra sostituissi `primi1` con `primi` tutto funzionerebbe ancora. Un utile risparmio di memoria.

**5.** Ora che hai un vettore compattato di numeri utili, sapresti organizzare la stampa in modo che i numeri vengano allineati in 10 colonne ben ordinate? Qui devi saper giocare con l'istruzione `printf`. Ti potrà essere utile sapere (o ricordare) che il resto della divisione di un intero `k` per 10 si calcola con l'operazione `k%10`; se il risultato è zero significa che `k` è multiplo di 10.

Una volta sistemata per bene la stampa, perché non provare a calcolare e stampare i numeri primi fino a 1000?