LABORATORIO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO
2021–2022

Indice

Ultima revisione: 29 marzo 2022.

Parte 1. Esercizi facili con la programmazione strutturata

Tre sommatorie facili

Sia dato un intero $n \geqslant 1$. Valgono le tre identità:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{1}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{2}$$

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4} \tag{3}$$

Esercizio 1

Somma dei primi n numeri naturali.

Tempo: 20 min.

Scrivere un programma che chieda in ingresso all'utente un intero $n \geqslant 1$. Se l'intero inserito non soddisfa questa disuguaglianza, il programma chiede il reinserimento fino a che la condizione non sia verificata. Il programma visualizza poi la somma dei primi n numeri naturali calcolata con un ciclo `for`, assieme alla stessa somma calcolata tramite la formula al membro destro di (1).

Esercizio 2

Somma dei primi n quadrati.

Tempo: 5 min.

Scrivere un programma che chieda in ingresso all'utente un intero $n \geqslant 1$. Se l'intero inserito non soddisfa questa disuguaglianza, il programma chiede il reinserimento fino a che la condizione non sia verificata. Il programma visualizza poi la somma dei primi n quadrati calcolata con un ciclo `for`, assieme alla stessa somma calcolata tramite la formula al membro destro di (2).

Esercizio 3

Somma dei primi n cubi.

Tempo: 5 min.

Scrivere un programma che chieda in ingresso all'utente un intero $n \geqslant 1$. Se l'intero inserito non soddisfa questa disuguaglianza, il programma chiede il reinserimento fino a che la condizione non sia verificata. Il programma visualizza poi la somma dei primi n cubi calcolata con un ciclo `for`, assieme alla stessa somma calcolata tramite la formula al membro destro di (3).

radq.c

```c
#include <stdio.h>
#include <math.h> //Intestazione necessaria per double sqrt(double)

int main(void)
{
        double d;
        do
        {
           printf("Inserisci un numero reale non negativo: ");
           scanf("%lf",&d);
        } while (d<0);

        printf("La radice quadrata di %g e' %g.\n",d,sqrt(d));
        return 0;
}
```

FIGURA 1. Esempio d'uso della funzione `sqrt` della libreria standard.

ESERCIZIO 4

Equazioni di secondo grado.

Tempo: 25 min.

Scrivere un programma che permetta di calcolare le soluzioni reali delle equazioni di secondo grado a coefficienti reali. Considereremo quindi equazioni della forma

$$ax^2 + b^2 + c = 0 \tag{4}$$

con $a, b, c \in \mathbb{R}$, $a \neq 0$, e soluzioni in $\mathbb{R}$.

All'avvio, il programma chiede all'utente di inserire i coefficienti `a`, `b` e `c`, memorizzandoli in variabili di tipo `double`. Se il valore di `a` è zero il programma chiede all'utente di reinserire `a` fino a che `a` $\neq 0$. Il programma calcola quindi il discriminante

$$\Delta := b^2 - 4ac$$

di (4), e ne visualizza il valore. Basandosi sull'analisi del segno di Δ il programma comunica all'utente il numero delle soluzioni reali di (4). Nel caso vi siano soluzioni reali, il programma le calcola e le visualizza prima di terminare.

Per calcolare le soluzioni di (4) dovrete estrarre una radice quadrata. A tal fine potrete usare la funzione `sqrt()` (contrazione di *square root*) del file di intestazione `math.h`. La Figura 1 riporta il codice sorgente di un programma che estrae la radice quadrata di un numero reale non negativo.

Richiesta di *linking* delle librerie matematiche

Alcune implementazioni del compilatore C prevedono che si richieda esplicitamente l'uso delle librerie matematiche con l'opzione `-lm`, che sta per *link mathematical library*. Per compilare il codice in Figura 1 dovrete quindi invocare il compilatore in questo modo:

```
gcc -lm radq.c
```

FIGURA 2. Una statua di Leonardo Pisano Fibonacci.

Parte 2. Il m.c.d. con le divisioni: Euclide, Fibonacci e Lamé

ESERCIZIO 5

L'algoritmo euclideo con le divisioni.

Tempo: 20 min.

La Figura 3 mostra lo pseudocodice dell'algoritmo euclideo basato sulle divisioni successive. Scrivete il codice C corrispondente, assumendo che `a` e `b` siano variabili intere il cui valore è impostato all'inizio del programma tramite istruzioni di assegnamento; per esempio, `int a=100;` e `int b=46;` .

Input : $\mathtt{a}, \mathtt{b} > 0$
Output : $\mathtt{m.c.d.}(\mathtt{a}, \mathtt{b})$
while $(\mathtt{b} \neq 0)$
    $\mathtt{aux} := \mathtt{b}$
    $\mathtt{b} := \mathtt{a}$ **mod** $\mathtt{b}$
    $\mathtt{a} := \mathtt{aux}$
return a

FIGURA 3. L'algoritmo euclideo delle divisioni successive in pseudocodice.

ESERCIZIO 6

L'algoritmo euclideo: input dell'utente.

Tempo: 10 min.

Modificate la vostra soluzione all'Esercizio 5 in modo che sia l'utente a inserire da terminale i valori delle variabili `a` e `b`, all'inizio dell'esecuzione.

ESERCIZIO 7

L'algoritmo euclideo: validazione dell'input.

Tempo: 10 min.

Modificate la vostra soluzione all'Esercizio 6 in modo che il programma si assicuri che i valori di `a` e `b` inseriti dall'utente soddisfino le precondizioni: $\texttt{a}, \texttt{b} > 0$. Nel caso ciò non avvenga, visualizzate un appropriato messaggio d'errore per l'utente e terminate immediatamente l'esecuzione con l'istruzione `return -1`.

Come abbiamo già visto, restituire il valore -1 segnala convenzionalmente una condizione eccezionale di terminazione.

Esercizio 8

L'algoritmo euclideo: conteggio delle divisioni.

Tempo: 10 min.

Modificate la vostra soluzione all'Esercizio 7 in modo che, al termine del calcolo del m.c.d., il programma visualizzi il numero di operazioni di divisione che ha eseguito. Usate una variabile intera per tenere traccia del numero di divisioni. Incrementate di una unità il valore della variabile ad ogni successiva divisione.

Esercizio 9

La successione di Fibonacci (*versione iterativa*).

Tempo: 15 min.

La Figura 4 mostra il sorgente parziale di un programma in C che calcola l'n-esimo termine della successione di Fibonacci. Manca una sola riga di codice, come indicato in figura. Aggiungetela, compilate ed eseguite il programma, ed assicuratevi con qualche esperimento del suo corretto funzionamento.

La successione di Fibonacci

Si consideri la successione di numeri naturali induttivamente definita da:

$$\begin{aligned} F_1 &:= 1, \\ F_2 &:= 1, \\ F_n &:= F_{n-1} + F_{n-2} \text{ per } n \geqslant 3. \end{aligned} \qquad (\star)$$

La successione $(\star)$ è nota come *successione di Fibonacci*.[a] In Tabella 1 sono tabulati i primi termini della successione di Fibonacci.

[a] Dal nome di Leonardo Pisano Fibonacci (ca. 1170–ca. 1250), così detto perché figlio del mercante pisano Guglielmo dei Bonacci. Leonardo fu un importante matematico medievale. Nel 1202 pubblicò il *Liber Abaci*, che svolse un ruolo fondamentale nella diffusione in Occidente del sistema di numerazione indiano basato sulle cifre 0–9, comunemente dette "arabe". La successione di Fibonacci compare in un esempio del *Liber Abaci*.

n	1	2	3	4	5	6	7	8	9	10	11	12	13
F_n	1	1	2	3	5	8	13	21	34	55	89	144	233

Tabella 1. I primi termini della successione di Fibonacci

$a \setminus b$	1	2	3	4	5	6	7	8
1	1							
2	1	1						
3	1	2	1					
4	1	1	2	1				
5	1	2	3	2	1			
6	1	1	1	2	2	1		
7	1	2	2	3	3	2	1	
8	1	1	3	1	4	2	2	1

TABELLA 2. Il numero di divisioni eseguito dall'algoritmo euclideo della Figura 3 per $a \geqslant b > 0$.

Fibonacci.c

```c
#include <stdio.h>

int main(void)
{
        int n;          //il programma calcola F(n)

        do
        {
                printf()"Inserisci un intero positivo: ");
                scanf("%d",&n);
        } while ( (n<=0) ); //sono ammessi solo interi positivi

        printf("F(%d)=",n);

        int i;              //contatore del ciclo for
        int f_i=1,f_prec=1; //valori iniziali F(2)=F(1)=1

        for (i=3;i<=n;i++)  //il ciclo comincia da i=3
        {
                int aux=f_i;    //variabile ausiliaria aux
                ...;            //somma ad f_i il valore di f_prec
                f_prec=aux;     //f_prec = f_i prima dell'iterazione
        }
        printf("%d\n",f_i);
}
```

FIGURA 4. Una implementazione iterativa in C della successione di Fibonacci. L'implementazione è completa, eccezion fatta per la riga 21.

ESERCIZIO 10

Verifica sperimentale della proposizione di Lamé.

Tempo: 15 min.

Il matematico francese Gabriel Lamé (1795–1870) dimostrò il fatto seguente.

Proposizione

Sia $n > 0$ un intero, e siano $a > b > 0$ interi tali che l'algoritmo euclideo della Figura 3 applicato ad a e b esegua esattamente n divisioni. Se a è il minimo intero che soddisfa tali condizioni, si ha

$$a = F_{n+2}$$
$$b = F_{n+1},$$

dove F_n denota la successione di Fibonacci.

Usando i programmi che avete sviluppato negli Esercizi 9 e 8, verificate la proposizione di Lamé in qualche caso concreto. Per esempio, la Tabella 2 riporta il numero di divisioni eseguito dall'algoritmo euclideo per $8 \geqslant a \geqslant b > 0$. La vostra soluzione dell'Esercizio 8 dovrebbe produrre valori identici. Ora notate che la prima coppia (a, b) che richieda 4 divisioni è $(8, 5)$ — e infatti, dalla Tabella 1 si vede che $F_6 = 8$ ed $F_5 = 5$, in accordo con la proposizione di Lamé. Provate con almeno altri due valori di n, uno più piccolo e uno più grande di 4.

Parte 3. Selezioni multiple annidate

Esercizio 11

Il signor Pignolino.

Tempo: 35 min.

Il signor Pignolino è pensionato. Egli esce di casa ogni mattina, ma solo a condizione che non piova e che le previsioni metereologiche per la giornata siano buone. Nel caso in cui decida di uscire, se il giorno del mese è pari (ad esempio, è il 20 di febbraio, o di marzo, etc.), si reca ai giardini pubblici, portando con sé un libro; altrimenti, si reca al Caffé, portandosi dietro il suo diario personale invece di un libro. Quanto detto vale a meno che non sia domenica, giorno in cui il signor Pignolino va invariabilmente a far visita alla sua cara amica signora Precisina, avendo l'accortezza di prendere un ombrello nel caso in cui piova o le condizioni metereologiche non siano buone, o un parasole altrimenti.

Si scriva un programma che rivolga al signor Pignolino un certo numero di domande, tenga traccia delle sue risposte per mezzo di variabili intere, e produca in uscita una descrizione accurata del comportamento del signor Pignolino per la giornata. Il signor Pignolino è impersonato dall'utente. Si assuma che le risposte del signor Pignolino siano solo del tipo Sì/No, codificate dai caratteri `S` e `N`, rispettivamente. Se la risposta a una qualsiasi delle domande non è `S` o `N`, il programma termina con un messaggio d'errore. Se invece tutte le risposte del signor Pignolino sono state acquisite correttamente, il programma deve produrre in uscita, prima di terminare, una e una sola fra le frasi seguenti, quella che riflette esattamente le risposte fornite da Pignolino:

- `Il signor Pignolino oggi non e' uscito.`

- `Il signor Pignolino oggi e' uscito per recarsi ai giardini pubblici. Ha portato con se' un libro.`
- `Il signor Pignolino oggi e' uscito per recarsi al Caffe'. Ha portato con se' il suo diario.`
- `Il signor Pignolino oggi e' uscito per recarsi dalla signora Precisina. Ha portato con se' un ombrello.`
- `Il signor Pignolino oggi e' uscito per recarsi dalla signora Precisina. Ha portato con se' un parasole.`

Esercizio 12

Il signor Pignolino interrogato con insistenza.

Tempo: 15 min.

Modificate il programma che avete scritto per risolvere l'Esercizio 11 di modo che, nel caso in cui Pignolino risponda a un domanda con un carattere diverso da `S` e `N`, il programma, invece di terminare con errore, ponga nuovamente la stessa domanda fino ad ottenere una risposta nel formato corretto. Per il resto il comportamento del programma è invariato.

Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano