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

2018–2019

20.III.2019

VINCENZO MARRA

INDICE

Ultima revisione: 20 marzo 2019.

Parte 1. L'algoritmo euclideo

Esercizio 1

L'algoritmo euclideo.

Tempo: 15 min.

La Figura 1 mostra un'implementazione parziale dell'algoritmo euclideo delle sottrazioni successive. Completatela.

E poi **sperimentate** con i possibili valori delle variabili: cosa succede se non si rispetta la specifica $a \geqslant b > 0$ per i valori in ingresso? Vi sono casi in cui il programma *non termina*?

Euclide.c

```c
#include <stdio.h>

/*******************************************************
 * Calcola il m.c.d di due interi a >= b > 0 applicando *
 * l'algoritmo delle sottrazioni successive di Euclide. *
 *******************************************************/

int main(void)
{
        int a=214, b=128;        //Deve essere a>=b>0

        printf("Il m.c.d. di %d e %d e': ",a,b);

        while (a != b)               //Fino a quando a e b sono diversi...
                if (a > b)        //Se a>b,
                        ....        //sostituisci a con a-b.
                else                //Altrimenti,
                        ....        //sostituisci b con b-a.

        printf("%d\n",a);
        return 0;
}


```

Figura 1. L'algoritmo euclideo delle sottrazioni successive in C.

"M'è andato in loop il programma..." — `ctrl+C`

Così si dice a volte in gergo per indicare la circostanza che, a fronte di certi valori in ingresso, il programma in questione non termini. La locuzione è dovuta al fatto che spesso (ma non sempre, naturalmente) la mancata terminazione del programma è dovuta a un ciclo — in inglese, *loop* — che continua indefinitamente a eseguire iterazioni. La sequenza `ctrl+C` invia un *segnale di terminazione* al programma in esecuzione, costringendolo[a] a interrompere l'esecuzione.

[a] È possibile che un programma si rifiuti di terminare nonostante glielo si intimi con `ctrl+C`. Senza entrare nei dettagli, basterà dire che il codice scritto da voi in questo corso risponderà sempre a `ctrl+C` con la terminazione.

Esercizio 2

L'algoritmo euclideo: input dell'utente.

Tempo: 15 min.

Modificate la vostra soluzione all'Esercizio 1 in modo che sia l'utente a inserire da terminale i valori delle variabili `a` e `b`, all'inizio dell'esecuzione.[1]

Esercizio 3

L'algoritmo euclideo: validazione dell'input.

Tempo: 15 min.

Quale costrutto della programmazione strutturata è necessario usare qui? Sequenza, iterazione o selezione?

Modificate la vostra soluzione all'Esercizio 2 in modo che il programma si assicuri che i valori di `a` e `b` inseriti dall'utente soddisfino le precondizioni: $a \geqslant b > 0$.

Nel caso ciò non avvenga, visualizzate un appropriato messaggio d'errore per l'utente e terminate immediatamente l'esecuzione con l'istruzione `return -1`.

Restituire il valore -1 indica una condizione eccezionale di terminazione della funzione `main`. Ne parleremo meglio più avanti nel corso.

Esercizio 4

L'algoritmo euclideo: conteggio delle iterazioni.

Tempo: 10 min.

Modificate la vostra soluzione all'Esercizio 3 in modo che, al termine del calcolo del m.c.d., il programma visualizzi il numero di iterazioni `while` che ha eseguito. Usate una variabile intera per tenere traccia del numero di iterazioni. Incrementate di una unità il valore della variabile ad ogni successiva iterazione.

Parte 2. Una rudimentale calcolatrice

Esercizio 5

Calcolatrice usa e getta.

[1] In tutta questa lezione, usate le funzioni della libreria didattica `Prog1` per acquisire i dati inseriti dall'utente.

Tempo: 20 min.

Scrivete un programma secondo le specifiche seguenti.

(1) All'avvio, il programma chiede all'utente di inserire due valori reali (tipo `double`).
(2) Acquisiti i due operandi, il programma chiede all'utente se voglia eseguire una somma, una sottrazione, una moltiplicazione, o una divisione.
(3) L'utente inserisce uno fra gli interi `1`, `2`, `3`, e `4` per indicare la sua scelta.
(4) Acquisita la scelta dell'utente, il programma esegue l'operazione richiesta, visualizza il risultato, e termina.

Osservate: l'analisi dei dati inseriti dall'utente si basa sul costrutto selezione. È necessario usare il costrutto iterazione in questo programma?

Esercizio 6

Calcolatrice usa e getta, gestione degli errori.

Tempo: 10 min.

Modificate la calcolatrice scritta per l'Esercizio 5 di modo che:

(1) Se la scelta dell'utente non è uno degli interi `1`, `2`, `3`, e `4`, il programma termina visualizzando un appropriato messaggio d'errore.
(2) Se l'utente sceglie di eseguire una divisione, ma inserisce valore zero per il divisore, il programma termina visualizzando un appropriato messaggio d'errore.

Esercizio 7

Calcolatrice riutilizzabile.

Tempo: 15 min.

Modificate la calcolatrice con gestione degli errori scritta per l'Esercizio 6 di modo che, in caso di visualizzazione di un messaggio d'errore, il programma riprenda l'esecuzione dall'inizio, proponendo all'utente la scelta dell'operazione da compiere. Analogamente, fate sì che quando una operazione è stata eseguita con successo, il programma ritorni all'inizio.
(*Suggerimento.* Usate un'istruzione `while` con condizione sempre verificata per realizzare un ciclo infinito.)

Osservate: "riutilizzare la clacolatrice" vorrà quindi dire "ripetere (l'esecuzione di un brano di codice) tramite iterazione"

La versione più semplice di una tale iterazione è:

```
while (1)
    istruzione
```

Esercizio 8

Calcolatrice riutilizzabile con pulsante `Off`.

Tempo: 10 min.

Modificate la calcolatrice riutilizzabile scritta per l'Esercizio 7 di modo che l'utente abbia anche la scelta (oltre a `1`–`4`) di uscire dal programma. (*Suggerimento.* Modificate la condizione del ciclo `while`.)

Parte 3. Analisi dei caratteri con l'intestazione `ctype.h`

Classi di caratteri

Il file di intestazione `ctype.h` della libreria standard contiene funzioni di utilità per l'analisi dei caratteri. Ad esempio, la funzione `int isalpha(char c)` restituisce un intero non nullo se `c` è un carattere alfabetico, e zero altrimenti. La funzione `int isdigit(char c)` restituisce un intero non nullo se `c` è una cifra (cioè un carattere nell'insieme {`'0'`, `'1'`, ..., `'9'`}) e zero altrimenti. La funzione `int isupper(char c)` restituisce un intero non nullo se `c` è una lettera maiuscola, e zero altrimenti; la funzione `int islower(char c)` è analoga per le lettere minuscole. La funzione `int toupper(char c)` restituisce (come `int`) il carattere `c` convertito in maiuscola, se esso è un carattere alfabetico; altrimenti, poiché non ha senso convertire in maiuscola un carattere non alfabetico, la funzione restituisce il carattere `c` privo di modifiche. La funzione `int tolower(char c)` è analoga per le lettere minuscole.

Esercizio 9

Lettere maiuscole e minuscole.

Tempo: 15 min.

Scrivete un programma che chieda all'utente di inserire un carattere, e visualizzi poi le informazioni seguenti. Se il carattere non è alfabetico, il programma scrive in uscita `Il carattere c non è alfabetico`, dove `c` sta per il carattere digitato dall'utente. Se il carattere è alfabetico, il programma scrive in uscita `Il carattere c è una lettera maiuscola`, oppure `Il carattere c è una lettera minuscola`, a seconda del caso che si verifica. Una volta visualizzate le informazioni appropriate, il programma termina. (*Suggerimenti.* Occorre includere `ctype.h` con la direttiva `#include <ctype.h>`. Come esempio d'uso delle funzioni definite in questo file di intestazione, si consideri la chiamata

```
ris=isupper('A');
```

dove `ris` è una variabile di tipo `int`. Dopo la chiamata `ris` contiene un valore non nullo, perché `'A'` è un carattere maiuscolo. Si consideri d'altronde la chiamata

```
c=tolower('A');
```

dove `c` è una variabile di tipo `char`. Dopo la chiamata `c` contiene il carattere `'a'`.)

Esercizio 10

Lettere maiuscole e minuscole iterate.

Tempo: 15 min.

Modificate la soluzione dell'Esercizio 9 di modo che, analizzato il primo carattere inserito dall'utente e visualizzate le informazioni relative, il programma torni a chiedere all'utente l'inserimento di un nuovo carattere da analizzare. (Ciò vale anche nel caso in cui il carattere inserito non sia alfabetico.) L'iterazione continua fino a che l'utente non preme direttamente il tasto INVIO alla richiesta di inserimento del carattere. A quel punto, il programma termina.

Esercizio 11

*Conversione in maiuscola***.**

Tempo: 10 min.

Scrivete un programma che chieda all'utente di inserire un carattere, e lo visualizzi convertito in maiuscola. Se il carattere inserito non è alfabetico il programma lo visualizza invariato.

Esercizio 12

*Conversione in maiuscola o minuscola***.**

Tempo: 15 min.

Modificate la soluzione dell'Esercizio 11 di modo che il programma, dopo aver acquisito il carattere dall'utente, chieda all'utente se intende eseguire una conversione in maiuscola o in minuscola, e proceda poi di conseguenza. (*Suggerimento.* Visualizzate la frase `Conversione in maiuscola o in minuscola? (1=maiuscola, altro numero=minuscola)` e leggete un intero. Confrontatelo con `1` usando l'operatore `==` nella condizione di una selezione `if-else`.)

Chiedetevi: cosa fa il mio programma se l'utente inserisce un dato non atteso, come per esempio, in questo caso, il valore `2`? Un buon programma tiene conto anche dell'eventualità che l'utente si comporti in modo inatteso — esso è, come si suol dire, "a prova di sciocco" (*foolproof*, in inglese).

Esercizio 13

*Conversione in maiuscola o minuscola con controllo d'errore***.**

Tempo: 5 min.

Modificate la soluzione dell'Esercizio 12 di modo che il programma visualizzi un messaggio d'errore appropriato nel caso in cui il carattere inserito dall'utente non sia alfabetico, e termini.

Domanda

Qual è il *primo* punto del codice nella soluzione dell'Esercizio 13 in cui è sensato inserire il controllo sulla natura alfabetica del carattere inserito dall'utente? Poiché il testo dell'esercizio non specifica altrimenti, è opportuno assicurarsi di aver inserito il controllo in questo punto.

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
E-mail address: vincenzo.marra@unimi.it