

Introduzione alla programmazione in C

Stefano Cherubin*

22/10/2015

[Informatica A] Esercitazione #3

corso per Ing. Gestionale a.a. 2015/16

*<nome.cognome>@polimi.it

Indice

1	Numeri non decrescenti	3
1.1	Approccio al problema	3
1.1.1	Analisi del testo dell'esercizio	3
1.2	Iniziare la stesura	4
1.2.1	Scrivere un programma in linguaggio C	4
1.2.2	tre numeri interi	4
1.2.3	Letti tre numeri interi dallo standard input	4
1.2.4	Stampare a terminale la sequenza dei numeri	5
1.3	Soluzione 1 - analisi per casi	5
1.4	Soluzione 2 - ordinamento dell'input	6
2	Implementare il quadrato tramite somme successive	8
2.1	Approccio alla soluzione	8
2.1.1	Premessa	8
2.1.2	Pseudocodice (stesura informale dell'algoritmo)	8
2.1.3	Blindare l'input	9
2.2	Soluzione C	10
3	Numeri triangolari	11
3.1	Approccio alla soluzione	11
3.1.1	Pseudocodice (stesura informale dell'algoritmo)	11
3.1.2	Distruggere l'input	11
3.2	Soluzione C	12
4	Conversione in binario	12
4.1	Approccio alla soluzione	13
4.1.1	LSB e MSB	13
4.1.2	Indicazioni per gestire un ciclo	13
4.2	Soluzione C	14

1 Numeri non decrescenti

Buongiorno ragazzi, benvenuti all'esame di Informatica A. Scrivete un programma in linguaggio C che risolva il seguente problema. Letti tre numeri interi a , b , c dallo standard input, stampare a terminale la sequenza dei tre numeri in ordine non decrescente.

Esempio: $a = 10$, $b = 7$, $c = 9$ deve dare in uscita 7 9 10.

1.1 Approccio al problema

Quando si parla di programmazione, la formulazione di un problema contiene quasi sempre

- vincoli
- descrizione della logica
- consigli su come trovare più facilmente una soluzione al problema
- elementi inutili o fuorvianti¹
- casi di test o esempi di output atteso

Sta al programmatore riconoscere questi elementi e gestirli al meglio delle sue possibilità.

1.1.1 Analisi del testo dell'esercizio

Buongiorno ragazzi elemento totalmente fuorviante. Oggi dovrete concentrarvi e produrre cose sensate, non ci sarà spazio per il divertimento.

benvenuti all'esame elemento inutile. Si tratta solo di convenevoli.

di informatica A consiglio. Se una volta ricevuto in mano il foglio dell'esame vi siete scordati di cosa si sta parlando, probabilmente dovrete ricordarvi delle lezioni di Informatica A.

Scrivete un programma vincolo. L'esercizio è un test di produzione e verrete valutati sulla capacità di scrivere un programma.

in linguaggio C vincolo. In questo esercizio dovrete rispettare lo standard previsto dal linguaggio C. La capacità di attenersi a questo standard sarà oggetto di valutazione.

che risolva il seguente problema elemento inutile. Dovrebbe essere sottinteso.

Letti tre numeri descrizione della logica. Dovete fare questo.

¹in sede di esame questi elementi tenderanno ad essere minimizzati. Al di fuori, costituiscono una parte decisamente non trascurabile.

interi vincolo. Dovrete lavorare con numeri interi.

a, b, c consiglio. Nello specifico caso, questo non è un brutto modo di chiamare i dati su cui lavorate.

dallo standard input vincolo. I dati dovranno essere acquisiti dallo standard input.

stampare [...] la sequenza dei tre numeri descrizione della logica.

a terminale vincolo. L'output deve essere emesso sul terminale, quindi standard output.

in ordine non decrescente vincolo. La parte di logica che non è descritta nel testo dovrà essere studiata per soddisfare questo vincolo. L'impostazione di tale logica è lasciata alla libera interpretazione del programmatore.

Esempio [...] Si tratta di un esempio. Al termine della produzione del programma, si consiglia di usare questi dati per eseguire una simulazione di esecuzione del programma e verificare che l'output fornito in simulazione coincida con l'output atteso.

1.2 Iniziare la stesura

Partendo dai vincoli e dalla logica, si possono identificare alcuni elementi ricorrenti che possono essere tradotti immediatamente in codice.

1.2.1 Scrivere un programma in linguaggio C

```
int main() {  
    return 0;  
}
```

1.2.2 tre numeri interi

```
int main() {  
    int a, b, c;  
    return 0;  
}
```

1.2.3 Letti tre numeri interi dallo standard input

```
#include <stdio.h> /* necessaria se voglio utilizzare  
                  standard input o standard output */  
int main() {
```

```

    int a, b, c;
    printf("\n Inserisci il numero a: ");
    scanf("%d",&a);
    printf("\n Inserisci il numero b: ");
    scanf("%d",&b);
    printf("\n Inserisci il numero c: ");
    scanf("%d",&c);

    return 0;
}

```

1.2.4 Stampare a terminale la sequenza dei numeri

```

#include <stdio.h>
int main() {
    int a, b, c;
    printf("\n Inserisci il numero a: ");
    scanf("%d",&a);
    printf("\n Inserisci il numero b: ");
    scanf("%d",&b);
    printf("\n Inserisci il numero c: ");
    scanf("%d",&c);

    printf("\n L'ordine voluto e': %d, %d, %d",a,b,c);
    return 0;
}

```

1.3 Soluzione 1 - analisi per casi

Un possibile approccio al problema è quello di valutare tutti i possibili ordinamenti dei numeri interi in input e, in base ad opportuni controlli, eseguire l'unica tra le istruzioni di output che utilizza l'ordinamento corretto.

```

#include <stdio.h> /* inclusione della libreria standard I/O */
int main( ) {

    int a,b,c;          /* dichiarazione delle variabili      */

    /* legge tre interi a,b,c dallo standard input            */
    printf("\n Inserisci il numero a: ");
    scanf("%d",&a);
    printf("\n Inserisci il numero b: ");
    scanf("%d",&b);
    printf("\n Inserisci il numero c: ");

```

```

scanf("%d",&c);

if (a < b) {
    if (b < c) {
        printf("\n L'ordine voluto e': %d, %d, %d",a,b,c);
    }
    else {
        if (a < c) {
            printf("\n L'ordine voluto e': %d, %d, %d",a,c,b);
        }
        else {
            printf("\n L'ordine voluto e': %d, %d, %d",c,a,b);
        }
    }
}
else {
    if (c < b) {
        printf("\n L'ordine voluto e': %d, %d, %d",c,b,a);
    }
    else {
        if (a < c) {
            printf("\n L'ordine voluto e': %d, %d, %d",b,a,c);
        }
        else {
            printf("\n L'ordine voluto e': %d, %d, %d",b,c,a);
        }
    }
}
return 0;
}

```

1.4 Soluzione 2 - ordinamento dell'input

Un secondo approccio alla risoluzione del problema prevede di fissare un solo output ed eseguire delle istruzioni utili a rendere i dati acquisiti coerenti nel loro ordinamento con quanto richiesto in output.

```

#include <stdio.h> /* inclusione della libreria standard I/O */
int main( ) {

    int a,b,c,t; /*    dichiarazione delle variabili        */

    /* legge tre interi a,b,c dallo standard input          */
    printf("\nInserisci il numero a: ");
    scanf("%d",&a);

```

```

printf("\nInserisci il numero b: ");
scanf("%d",&b);
printf("\nInserisci il numero c: ");
scanf("%d",&c);

/* ordinamento dei valori delle variabili a,b */
if (a > b) {

    /* Scambio dei valori delle due variabili a,b */
    t = a;
    a = b;
    b = t;
}

/* ordinamento dei valori delle variabili a,c */
if (a > c) {

    /* Scambio dei valori delle due variabili a,c */
    t = a;
    a = c;
    c = t;
}
/* la variabile a contiene ora sicuramente il */
/* valore più piccolo tra quelli inseriti. */

/* ordinamento dei valori delle variabili b,c */
if (b > c) {

    /* Scambio dei valori delle due variabili b,c */
    t = b;
    b = c;
    c = t;
}
printf("\nL'ordine voluto e': %d, %d, %d",a,b,c);
return 0;
}

```

2 Implementare il quadrato tramite somme successive

Si scriva un programma in linguaggio C che letto un numero intero positivo dallo standard input, visualizzi a terminale il quadrato del numero stesso facendo uso soltanto di operazioni di somma.

Si osservi che il quadrato di ogni numero intero positivo N può essere costruito sommando tra loro i primi N numeri dispari.

Esempio:

$$\begin{aligned} N &= 5 \\ N^2 &= 1 + 3 + 5 + 7 + 9 = 25 \end{aligned}$$

2.1 Approccio alla soluzione

2.1.1 Premessa

L'idea di soluzione è quella di scandire i primi N numeri dispari esprimendoli nella forma $(i + i + 1)$ al variare dell'intero $i \in [0; N - 1]$ e accumulare la loro somma man mano che si procede nella loro scansione in un'altra variabile.

$$\begin{aligned} N^2 &= \sum_{i=0}^{N-1} (2 \cdot i + 1) \\ &= \sum_{i=0}^{N-1} (i + i + 1) \\ &= (2 \cdot 0 + 1) + \dots + (2 \cdot i + 1) + \dots + (2 \cdot (N - 1) + 1) \end{aligned}$$

quindi si utilizzeranno almeno 3 variabili:

N numero in input

i contatore di iterazioni

S accumulatore

2.1.2 Pseudocodice (stesura informale dell'algoritmo)

1. Inizio dell'algoritmo
2. Leggi un numero intero positivo dallo standard input.
3. Inizializza un contatore i a 0
4. Inizializza un accumulatore S a 0
5. Finché il valore del contatore è minore del numero letto

- (a) Somma all'accumulatore il doppio del valore del contatore incrementato di 1
 - (b) Incrementa il contatore
 - (c) Torna al punto 5.
6. Stampa a terminale il valore dell' accumulatore
 7. Fine dell' algoritmo

2.1.3 Blindare l'input

Quando viene richiesto che un numero in input rispetti determinate caratteristiche (in questo caso che sia positivo) è possibile scrivere il programma in modo che non accetti dati non conformi alle caratteristiche richieste e continui a chiedere all'utente di fornire dei dati fino a quando l'utente non inserisce dati corretti.

Questo è possibile farlo grazie a un ciclo do-while con controllo sul dato immesso.

```
do {
    printf("\nInserisci un numero positivo N: ");
    scanf("%d",&N);
} while (N <=0);
```

Variante con messaggio di errore A volte può capitare che debbano essere inseriti più dati dello stesso tipo e il messaggio che invita l'utente a inserire i dati può non variare. In queste situazioni può essere utile segnalare quando si verifica un errore e non è stato accettato il dato.

```
do {
    printf("\nInserisci un numero positivo N: ");
    scanf("%d",&N);
    if (N <=0) {

        printf ("\nErrore: il numero inserito non e' positivo.");
    }
} while (N <=0);
```

Variante con ciclo while

```
printf("\nInserisci un numero positivo N: ");
scanf("%d",&N);
while (N <= 0) {
```

```

printf ("\nErrore: il numero inserito non e' positivo.");
printf("\nInserisci un numero positivo N: ");
scanf("%d",&N);
}

```

2.2 Soluzione C

```

#include <stdio.h> /* inclusione della libreria standard I/O */
int main( ) {

    /*      dichiarazione delle variabili      */
    int i; /* contatore                        */
    int N; /* variabile di cui si vuole calcolare il quadrato */
    int S; /* accumulatore per il risultato del calcolo      */

    /* ciclo di controllo che garantisce N > 0 */
    do {

        printf("\nInserisci un numero positivo N: ");
        scanf("%d",&N); /* legge N dallo standard input */
        /* Finché il numero inserito non è positivo ripetere
           l'immissione dati. */
    } while (N <= 0 );
    /* Il numero inserito è positivo */

    S = 0; /* inizializzazione della variabile di accumulo */
    /* ciclo di scansione dei primi N numeri dispari */
    i = 0;
    while(i < N) {

        /* Finché il contatore è minore del numero letto
           aggiorna il contenuto della variabile accumulatore */
        S = S + (i+i+1);
        i = i + 1; /* incrementa il contatore */
    }
    printf("\n Il quadrato del numero inserito e': %d \n",S);
    return 0;
}

```

3 Numeri triangolari

Si definisce Triangolare un numero costituito dalla somma dei primi N numeri interi positivi per un certo N .

Esempio: per $Q = 10$ si ha $Q = 1 + 2 + 3 + 4$, da cui $N = 4$.

Scrivere un programma C che stabilisca se un numero intero positivo Q , letto dallo standard input, è un numero triangolare o meno, utilizzando soltanto operazioni tra numeri interi. In caso affermativo stampare a video il numero inserito e il massimo tra gli addendi che lo compongono.

3.1 Approccio alla soluzione

3.1.1 Pseudocodice (stesura informale dell'algoritmo)

Idea di soluzione: se $Q - 1 - 2 - 3 - \dots - i - \dots - N = 0$ per un certo N allora Q è Triangolare.

1. Leggi il numero positivo Q dallo standard input
2. Inizializza un contatore i a zero;
3. Memorizza in una variabile S il valore della variabile in ingresso.
4. Finché il numero S è maggiore di zero
 - (a) Incrementa di 1 il valore del contatore
 - (b) Sottrai a S il valore del contatore i
 - (c) Torna a 4.
5. Se il valore residuo di S è zero allora
 - (a) Il numero è triangolare
 - (b) Il valore del massimo degli addendi è uguale al contatore i
 - (c) La variabile Q contiene il valore della variabile in ingresso
6. Altrimenti il numero NON è triangolare.

3.1.2 Distruggere l'input

È buona norma, in generale, non modificare le variabili contenenti i dati in ingresso perché può accadere che sia necessario accedere a tali valori in diversi punti del programma.

Nel caso appena mostrato senza l'ausilio di un'altra variabile (S) non sarebbe possibile, al termine della computazione, stampare a video il valore del numero inserito, così come richiesto dalla traccia del problema.

3.2 Soluzione C

```
#include <stdio.h> /* inclusione della libreria standard input */
int main( )
{
    /* dichiarazione delle variabili */

    int i;          /* variabile contatore */
    int Q;          /* variabile per il numero letto da tastiera */
    int S;          /* variabile accumulatore */
    /* ciclo di controllo per l'immissione dati che garantisce Q > 0 */
    do {

        printf("\n Inserisci un numero positivo Q: ");
        scanf("%d",&Q);
    } while (Q <= 0);
    S = Q;          /* copia del valore del dato in ingresso */
    i = 0;          /* inizializzazione del contatore */
    while (S > 0) {
        i = i + 1;
        S = S - i;
    }
    /* all'uscita dal ciclo il valore assunto dalla variabile S
       permette di procedere in base a due alternative */
    if (S == 0) {

        /* il valore del contatore i contiene qui il valore del massimo
           degli addendi che compongono il numero triangolare inserito*/
        printf("\n %d = alla somma dei primi %d numeri positivi!", Q, i);
    } else {

        printf("\n Il numero %d non e' un numero triangolare! \n", Q);
    }
    return 0;
}
```

4 Conversione in binario

Dato un numero intero positivo Q ,

- Scrivere la sua rappresentazione in binario naturale, applicando il tradizionale algoritmo per divisioni successive

(per convenzione, in questo esercizio l'output si intende corretto se letto da destra a sinistra);

- Indicare anche il minimo numero di bit utilizzato.

Esempio:

Input 19_{10}

Output $\underline{11001}_2$

4.1 Approccio alla soluzione

4.1.1 LSB e MSB

LSB **Less Significant Bit**. Nelle rappresentazioni binarie è il bit con il peso minore. Nel sistema di numerazione posizionali è solitamente la cifra più a destra di tutte (perché solitamente i numeri vengono letti da sinistra a destra). In questo esercizio ci viene chiesto di considerare corretto il risultato se letto da destra a sinistra. Il **LSB** sarà quindi il bit più a sinistra.

MSB **Most Significant Bit**. Cifra di valore maggiore nella rappresentazione di un numero binario. In questo esercizio sarà la cifra più a destra.

4.1.2 Indicazioni per gestire un ciclo

Per non sbagliare a impostare un ciclo, ecco alcune domande a cui è necessario dare risposta prima di scrivere codice

1. Da quali valori si deve partire?
2. Qual è la logica di ogni passo?
3. Come si aggiornano le variabili?
4. Quando si deve uscire dal ciclo?

Nello specifico caso di questo esercizio, ecco le risposte

1. Inizializzazioni:
 - (a) contatore = 0
 - (b) variabile da dividere = numero in input
2. Logica: divido il numero per la base (2) e scrivo il resto
3. aggiornamento variabili:
 - (a) il contatore aumenta di 1
 - (b) il numero viene diviso per due (approssimato all'intero inferiore)
4. Uscita dal ciclo: quando il numero raggiunge il valore 0

4.2 Soluzione C

```
#include <stdio.h> /* inclusione della libreria standard I/O */
int main( ) {

    int n; /* contatore */
    int Q; /* variabile per il numero letto da tastiera */
    /*ciclo di controllo per l'immissione dati che garantisce Q >= 0 */
    do {

        printf("\nInserisci un numero positivo o nullo Q: ");
        scanf("%d", &Q);
    } while (Q < 0);
    printf("\nIn binario: ");
    n = 0;
    do {
        printf("%d", (Q % 2));
        Q = Q / 2;
        n = n + 1;
    } while (Q != 0);
    printf("\nNumero di bit n = %d", n);
    return 0;
}
```

Licenza e crediti

Crediti

Quest'opera contiene elementi tratti da materiale di Gerardo Pelosi redatto per il corso di Fondamenti di Informatica per Ingegneria dell'Automazione a.a. 2013/14.

Licenza beerware²

Quest'opera è stata redatta da Stefano Cherubin. Mantenendo questa nota, puoi fare quello che vuoi con quest'opera. Se ci dovessimo incontrare e tu ritenessi che quest'opera lo valga, in cambio puoi offrirmi una birra.

²<http://people.freebsd.org/~phk/>