

ESERCIZI DI PROGRAMMAZIONE DA TEMI D'ESAME

- condizionali e cicli -

AVVISI (e risposte anticipate alle domande che mi fareste)

- Vedremo una serie di esercizi tratti dai temi d'esame:
in un singolo esercizio, i diversi “consigli” possono avere un *ruolo più o meno accentuato* [per alcuni è più complesso il processo di raffinamento successivo, per altri si arriva direttamente al codice ma la chiave è dare un significato preciso alle variabili, ecc. ecc.]
- In aula, vedremo *pochi esercizi significativi*: tanti esercizi sarebbero controproducenti, perché quello che è importante è
 - *esaminare approfonditamente* pochi esercizi significativi, per *imparare “in teoria” la tecnica*
 - *provare a rifarli* da soli (NB: non esiste una sola soluzione!) per *imparare “in pratica” la tecnica*
 - *provare poi da soli* (con l'aiuto del compilatore) a farne altri, per *far diventare la tecnica “spontanea”*

QUINDI:

- Faremo in aula esercizi tratti dai temi d'esame e verranno poi proposti **esercizi d'esame individuali senza soluzione** perché:
 - è possibile risolverli studiando bene gli esempi già visti:
se non avete idea di come risolverli, significa che non avete capito bene la tecnica “in teoria” e non bisogna guardare la soluzione, ma **riesaminare gli esempi fatti in aula e rifletterci su**
 - per imparare la tecnica “in pratica” dovete **arrivare in fondo da soli**, senza guardare la soluzione (altrimenti non imparate)
 - è assolutamente indispensabile commettere errori ma abituarsi a scoprirli – e correggerli – da soli (usando il compilatore!) per poi non commetterli più: guardando la soluzione uno non sbaglia e **se non si sbaglia non si impara**

INOLTRE...

SARETE INGEGNERI

Esercizio 1

Scrivere un programma che riceva in ingresso una sequenza arbitraria di interi terminata da uno zero e produca come risultato la media intera dei valori di ingresso (escludendo lo zero).

Esercizio 1

Scrivere un programma che riceva in ingresso una sequenza arbitraria di interi terminata da uno zero e produca come risultato la media intera dei valori di ingresso (escludendo lo zero).

Primo passo: capire il problema

Se ho acquisito $\text{num}_1, \text{num}_2, \dots, \text{num}_n$:

$$\text{devo calcolare} \quad \text{media} = \frac{\text{num}_1 + \text{num}_2 + \dots + \text{num}_n}{n}$$

Esercizio 1

Scrivere un programma che riceva in ingresso una sequenza arbitraria di interi terminata da uno zero e produca come risultato la media intera dei valori di ingresso (escludendo lo zero).

Primo passo: capire il problema

Se ho acquisito $\text{num}_1, \text{num}_2, \dots, \text{num}_n$:

$$\text{devo calcolare} \quad \text{media} = \frac{\text{num}_1 + \text{num}_2 + \dots + \text{num}_n}{n}$$

Secondo passo: l'idea

Ciclo per acquisire i numeri (esce quando acquisisce 0):

- durante il ciclo, tengo conto di:

somma + quanti numeri inseriti

Terzo passo: sviluppare l'algoritmo, eventualmente per “raffinamenti successivi” (si può partire da “pseudocodice”)

Come sviluppo il ciclo? do-while o while?

- con do-while (una acquisizione si fa in ogni caso)

```
do{  
    scanf(“%d”,&num);  
    if(num>0)  
        aggiorna somma e la quantità di numeri inseriti;  
while(num>0);  
  
calcola e stampa la media
```

Notare che `if(num>0)` è fondamentale, ma non serve se uso `while...`

- con while:

```
scanf("%d",&num);
```

```
while(num>0)
```

```
    aggiorna somma e la quantità di numeri inseriti;
```

```
    scanf("%d",&num);
```

```
calcola e stampa la media
```

NB: va bene qualsiasi algoritmo.

Vediamo il codice risultante usando il ciclo while

```
int somma, i, num, media;

printf("Inserire i numeri positivi, 0 per terminare\n");

somma=0;                // somma parziale
i=0;                    // quanti numeri già inseriti e sommati
scanf("%d",&num);
while(num>0){            // num acquisito ma non ancora sommato
    somma=somma+num;
    i++;
    scanf("%d",&num);
}

if(i!=0){
    media=somma/i;
    printf("Media dei numeri inseriti=%d\n", media);
}
else printf("Devi inserire almeno un numero\n");
```

Esercizio 1.2

Scrivere un programma che riceva in ingresso una sequenza arbitraria di interi positivi terminata da uno zero e produca come risultato la media intera dei valori di ingresso (escludendo lo zero).

Ogni volta che viene inserito un numero, il programma deve controllare che sia positivo e, nel caso non lo sia, avvisare l'utente dell'errore e riacquisire un nuovo numero.

Esercizio 1.2

Scrivere un programma che riceva in ingresso una sequenza arbitraria di interi positivi terminata da uno zero e produca come risultato la media intera dei valori di ingresso (escludendo lo zero).

Ogni volta che viene inserito un numero, il programma deve controllare che sia positivo e, nel caso non lo sia, avvisare l'utente dell'errore e riacquisire un nuovo numero.

Sviluppare l'algoritmo

dato che l'acquisizione ora è più articolata, uso il do-while...

```
do{  
    acquisisci num eventualmente ripetendo acquisizione;  
    if(num>0)  
        aggiorna somma e la quantità di numeri inseriti;}  
while(num>0);  
  
calcola e stampa la media
```

do{

acquisisci num eventualmente ripetendo acquisizione;

if(num>0)

aggiorna somma e la quantità di numeri inseriti;}

while(num>0);

calcola e stampa la media

→ scanf("%d", &num);

while(num<0){

printf("Devi inserire un numero non negativo!\n");

scanf("%d", &num);

}

```
int somma, i, num, media;

printf("Inserire i numeri positivi, 0 per terminare\n");

somma=0; // somma parziale
i=0; // quanti numeri già inseriti e sommati
do{
    scanf("%d",&num); // acquisisci num (con ripetizioni)
    while(num<0){
        printf("Devi inserire un numero non negativo!\n");
        scanf("%d",&num);
    }
    if(num>0){ // aggiorna somma e i
        somma=somma+num;
        i++; }
while(num>0);

if(i!=0){
    media=somma/i;
    printf("Media dei numeri inseriti=%d\n", media);
}
else printf("Devi inserire almeno un numero\n");
```

Esercizio 2

Scrivere un programma che calcoli il fattoriale di un numero intero fornito dall'utente.

Esercizio 2

Scrivere un programma che calcoli il fattoriale di un numero intero fornito dall'utente.

Primo passo: capire il testo e soprattutto il problema (se “non si sa da che parte prendere”, provare a risolvere qualche istanza a mano)

$$5! = 5*4*3*2$$

$$3! = 3*2$$

$$1! = 1$$

$$0! = 1$$



In generale, $N!$ è pari a:

1

se $N = 0$ o $N = 1$

$2*...*N$

se $N > 1$

Esercizio 2

Scrivere un programma che calcoli il fattoriale di un numero intero fornito dall'utente.

Secondo passo: individuare un metodo risolutivo

Usare una variabile *fatt*, posta inizialmente a 1.
Successivamente moltiplicarla per 2, ..., N
per ottenere il fattoriale:

`fatt=1`

`fatt=fatt*2`

`fatt=fatt*3`

`...`

`fatt=fatt*N`



Si userà un ciclo per ripetere la moltiplicazione

Esercizio 2

Scrivere un programma che calcoli il fattoriale di un numero intero fornito dall'utente.

Terzo passo: sviluppare l'algoritmo

In questo caso, probabilmente uno arriva quasi al codice...

Come visto, abbiamo dei casi particolari ($n=0$ e $n=1$)
e il caso generale

CONSIGLIO: pensare prima al caso generale,
poi considerare i casi particolari!

Esercizio 2

Scrivere un programma che calcoli il fattoriale di un numero intero fornito dall'utente.

```
printf("Inserire il numero N:\n");  
scanf("%d",&n);  
  
fatt=1;  
  
for(i=2; i<=n; i++)          // i: numero per cui devo moltiplicare  
    fatt=fatt*i;  
  
printf("Fattoriale di %d = %d\n",n,fatt);
```

NB: ci si accorge che i casi particolari sono già risolti!!!
per $n = 0$ o 1 , *fatt* risulta correttamente 1
(il ciclo *for* non viene mai eseguito)

Ovviamente, il programma completo sarà il seguente

```
#include<stdio.h>
#include<stdlib.h>

main(){
int n, fatt, i;

printf("Inserire il numero N:\n");
scanf("%d",&n);

fatt=1;
for(i=2; i<=n; i++)
    fatt=fatt*i;

printf("Fattoriale di %d = %d\n",n,fatt);

system("pause");
}
```

Esercizio 3

Scrivere un programma che riceva in ingresso un numero positivo N e determini il massimo intero K tale che la somma dei primi K interi sia minore o uguale a N.

Ad esempio, se $N=20$ allora K risulta 5, infatti

$$1 + 2 + 3 + 4 + 5 = 15 \quad \text{mentre}$$

$$1 + 2 + 3 + 4 + 5 + 6 = 21$$

Esercizio 3

Scrivere un programma che riceva in ingresso un numero positivo N e determini il massimo intero K tale che la somma dei primi K interi sia minore o uguale a N .

Ad esempio, se $N=20$ allora K risulta 5, infatti

$$1 + 2 + 3 + 4 + 5 = 15 \quad \text{mentre}$$

$$1 + 2 + 3 + 4 + 5 + 6 = 21$$

Primo passo (se si è in difficoltà): provare a risolvere a mano qualche istanza del problema

P. es. $N=6$

$$1 + 2 + \textcircled{3} = 6 \leq 6$$

$$1 + 2 + 3 + \textcircled{4} = 10 > 6$$

\Rightarrow Risultato: 3

Secondo passo: individuare un metodo risolutivo

Dato N

- una variabile i scandisce i numeri 1, 2, 3, ... (userò un ciclo)
- una variabile $somma$ tiene conto della somma parziale

$SOMMA = SOMMA + i$

$$1 + 2 + 3 + 4 + 5 + \dots + (i - 1) + i > N$$

**FERMATI
QUANDO**

RISULTATO

Terzo passo: sviluppare l'algoritmo:

è importante dare un significato preciso alle variabili!

```
int i, somma, N, K;
```

```
printf("Inserire il numero positivo N\n");
```

```
scanf("%d", &N);
```

```
i=1; // ultimo indice già sommato
```

```
somma=1; // somma corrente (avendo già sommato i)
```

```
while(somma<=N){ // esci quando somma supera strettamente N
```

```
    i++;
```

```
    somma=somma+i;
```

```
}
```

```
K = i-1;
```

Esercizio 4

Scrivere un programma che, ricevuto in ingresso un intero $N \geq 0$, calcoli l' N -simo elemento della sequenza F dei numeri di Fibonacci, definita così:

$$F(0) = 0$$

$$F(1) = 1$$

$$F(K) = F(K-1) + F(K-2) \quad \text{per } K \geq 2$$

In altre parole, la sequenza dei numeri di Fibonacci è la seguente:

0, 1, 1, 2, 3, 5, 8, 13, ...

in cui ciascun numero, dal terzo in poi, è la somma dei due che lo precedono.

Primo passo: provare a risolvere a mano qualche istanza del problema

N=2: 0, 1, 1 il programma restituisce 1

N=3: 0, 1, 1, 2 il programma restituisce 2

Primo passo: provare a risolvere a mano qualche istanza del problema

N=2: 0, 1, 1 il programma restituisce 1

N=3: 0, 1, 1, 2 il programma restituisce 2

Secondo passo: individuare un metodo risolutivo

N=0, N=1: casi base (il programma deve restituire 0 o 1)

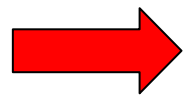
N \geq 2: uso un contatore I che arriva fino a N:

- ad ogni passo devo calcolare il nuovo numero di Fibonacci F(I):
devo sommare gli ultimi due numeri di Fibonacci ottenuti
- quindi, memorizzo in due variabili FIBP e FIBU gli ultimi due numeri di Fibonacci ottenuti e, ad ogni passo:
 - FIBP deve diventare FIBU
 - FIBU deve diventare FIBP+FIBU

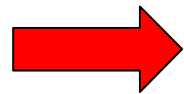
Terzo passo: sviluppare l'algoritmo

NB: per cominciare trascuriamo i casi base (quelli sono semplici e ci pensiamo dopo): risolviamo il “cuore” del problema

Errore comune: cominciare a scrivere il codice senza avere in testa l'algoritmo o, cosa ancora peggiore, un'idea del metodo risolutivo



TIPICAMENTE, QUESTO PORTA A SCRIVERE UN PO' DI IF PER GESTIRE I PRIMI CASI SPECIFICI (se $N==0$, se $N==1$, se $N==2$, ...)



TIPICAMENTE, QUESTO INDUCE A RITENERE CHE SI E' COMINCIATO A SCRIVERE IL CODICE SENZA PRIMA PENSARE ALL'ALGORITMO



SEMPRE, IL DOCENTE RITIENE CHE CIO' SIA VERO

Terzo passo: sviluppare l'algoritmo

NB: per cominciare trascuriamo i casi base (quelli sono semplici e ci pensiamo dopo): risolviamo il “cuore” del problema

```
FIBP = 0;
```

```
FIBU = 1;    // comincio con i primi due numeri della serie
```

```
i = 1;      // i riferito all'ultimo numero di Fibonacci trovato FIBU
```

```
while(i < N){    // esco quando i=N, cioè quando FIBU contiene F(i)=F(N)
```

Terzo passo: sviluppare l'algoritmo

NB: per cominciare trascuriamo i casi base (quelli sono semplici e ci pensiamo dopo): risolviamo il “cuore” del problema

```
FIBP = 0;
```

```
FIBU = 1;    // comincio con i primi due numeri della serie
```

```
i = 1;      // i riferito all'ultimo numero di Fibonacci trovato FIBU
```

```
while(i < N){    // esco quando i=N, cioè quando FIBU contiene F(i)=F(N)
```

Aggiorna FIBU

Aggiorna FIBP

```
i++;
```

Terzo passo: sviluppare l'algoritmo

NB: per cominciare trascuriamo i casi base (quelli sono semplici e ci pensiamo dopo): risolviamo il “cuore” del problema

```
FIBP = 0;
```

```
FIBU = 1;    // comincio con i primi due numeri della serie
```

```
i = 1;      // i riferito all'ultimo numero di Fibonacci trovato FIBU
```

```
while(i < N){    // esco quando i=N, cioè quando FIBU contiene F(i)=F(N)
```

<i>Aggiorna FIBU</i>	}	FIBU = FIBP + FIBU; FIBP = ? FIBU ?
<i>Aggiorna FIBP</i>		

```
    i++;
```


Terzo passo: sviluppare l'algoritmo

NB: per cominciare trascuriamo i casi base (quelli sono semplici e ci pensiamo dopo): risolviamo il “cuore” del problema

```
FIBP = 0;
```

```
FIBU = 1;    // comincio con i primi due numeri della serie
```

```
i = 1;      // i riferito all'ultimo numero di Fibonacci trovato FIBU
```

```
while(i<N){    // esco quando i=N, cioè quando FIBU contiene F(i)=F(N)
    NEWFIB= FIBP+FIBU;    // il nuovo numero di Fibonacci FIBU
    FIBP=FIBU;            // aggiorno FIBP
    FIBU=NEWFIB;          // aggiorno FIBU
    i++;                  // aggiorno i, che si riferisce di nuovo a FIBU
}
```

Terzo passo: sviluppare l'algoritmo

NB: per cominciare trascuriamo i casi base (quelli sono semplici e ci pensiamo dopo): risolviamo il “cuore” del problema

```
FIBP = 0;
FIBU = 1;    // comincio con i primi due numeri della serie
i = 1;      // i riferito all'ultimo numero di Fibonacci trovato FIBU

while(i<N){   // esco quando i=N, cioè quando FIBU contiene F(i)=F(N)
    NEWFIB= FIBP+FIBU; // il nuovo numero di Fibonacci FIBU
    FIBP=FIBU;         // aggiorno FIBP
    FIBU=NEWFIB;       // aggiorno FIBU
    i++;              // aggiorno i, che si riferisce di nuovo a FIBU
}
```

Ora posso considerare i casi base:

- il caso $N=0$ non è gestito (il programma porta FIBU a 1)
- il caso $N=1$ è gestito (il ciclo while non viene eseguito!)

```

#include <stdio.h>
#include <stdlib.h>
main(){
    int n, i, fibu, fibp, newfib;
    printf("Inserire il numero N:\n");
    do
        scanf("%d",&n);
    while(n<0);

    fibp=0;
    fibu=1;
    i=1;

    while(i<n){
        newfib=fibp+fibu;
        fibp=fibu;
        fibu=newfib;
        i++;
    }

    if(n==0)        //gestione del caso base
        fibu=0;    //si poteva gestire all'inizio con return 0

    printf("Numero di Fibonacci %d = %d\n",n,fibu);
    system("PAUSE");
}

```

NB: alternativa senza l'uso della variabile temporanea *newfib*

```
while(i<n){  
    fibu=fibp+fibu;  
    fibp=fibu-fibp;  
    i++;  
}
```

Esercizio 5

Scrivere un programma che, ricevuto in ingresso un intero strettamente maggiore di 0, determini se tale numero è primo.

Esercizio 5

Scrivere un programma che, ricevuto in ingresso un intero strettamente maggiore di 0, determini se tale numero è primo.

Primo passo: provare a risolvere a mano qualche istanza del problema

Es. 8 non è primo (è divisibile per 2 e per 4)

7 è primo (è divisibile solo per 1 e per 7, non per 2, 3, 4, ...6)

Esercizio 5

Scrivere un programma che, ricevuto in ingresso un intero strettamente maggiore di 0, determini se tale numero è primo.

Primo passo: provare a risolvere a mano qualche istanza del problema

Es. 8 non è primo (è divisibile per 2 e per 4)

7 è primo (è divisibile solo per 1 e per 7, non per 2, 3, 4, ...6)

Secondo passo: individuare un metodo risolutivo

Dato N , verifico se è divisibile per 2, 3, ... $n-1$

- se non è divisibile per nessuno: il numero è primo
- se esiste un divisore: il numero non è primo

E' facile rendersi conto che basta un ciclo con un indice i che va da 2 a $n/2$ (divisione intera)

Terzo passo: sviluppare un algoritmo

```
#include <stdio.h>
#include <stdlib.h>

main(){
    int n, i, trovato;

    printf("Inserisci un numero positivo\n");
    scanf("%d",&n);

    trovato=0;                                //verifica se si è trovato un divisore (prima del ciclo: no!)
    for(i=2; i<=n/2; i++)                      //prova con tutti i numeri da 2 a n/2
        if(n%i == 0)
            trovato=1;

    if(trovato)
        printf("Il numero non è primo\n");
    else printf("Il numero è primo\n");

    system("pause");
}
```


Esercizio 6

Scrivere un programma che acquisisca da tastiera un numero intero assicurandosi che sia positivo e, successivamente, stampi a video i 5 anni bisestili strettamente superiori al numero acquisito.

Esercizio 6

Scrivere un programma che acquisisca da tastiera un numero intero assicurandosi che sia positivo e, successivamente, stampi a video i 5 anni bisestili strettamente superiori al numero acquisito.

Sul problema c'è poco da capire: la difficoltà sta nell'algoritmo...

```
acquisisci N;  
numbisestili=0;           // numero di anni trovati (e stampati)  
ultbisestile=N;           // ultimo anno bisestile trovato (quasi)  
while(numbisestili<5){    // esci quando sono già stati trovati 5 anni  
    calcola e stampa il primo anno bisestile superiore a ultbisestile;  
    aggiorna ultbisestile con l'anno trovato;  
    numbisestili++;  
}
```

VEDIAMO LA PARTE CENTRALE DEL CODICE...

```
numbisestili=0;           // numero di anni trovati e stampati
ultbisestile=n;           // (serve trovare il successivo bisestile)
```

```
while(numbisestili<5){
```

```
    Trova  
    bisestile  
    successivo {  
        ultbisestile++;  
        while( !( ((ultbisestile % 100 != 0) && (ultbisestile % 4 ==0))  
                || (ultbisestile % 400 ==0)) )  
            ultbisestile++;  
        numbisestili++;  
        printf("Anno bisestile successivo numero %d = %d\n",  
                numbisestili, ultbisestile);  
    }
```

Esercizi proposti per lavoro individuale

- 1) Acquisire da tastiera un numero intero n (ripetendo l'acquisizione in caso di numero negativo) e un numero reale x .
Calcolare la sommatoria $\sum_{i=0}^n x^i$
- 2) Acquisire un numero positivo N e calcolarne la radice quadrata intera (ovvero il massimo intero x tale che $x^2 \leq N$).
- 3) Acquisire numeri da tastiera finché viene inserito lo zero e stampare la somma degli ultimi tre numeri inseriti (0 escluso).
- 4) Acquisire una sequenza di numeri interi e calcolare la somma di quelli positivi. Il programma deve terminare non appena l'utente inserisce per due volte consecutive un valore negativo.
- 5) Calcolare il massimo comun divisore e il minimo comune multiplo di due numeri interi forniti dall'utente.

Esercizi proposti per lavoro individuale

- 6) Acquisire dall'utente la lunghezza dei tre lati di un triangolo fintantoché queste lunghezze non sono positive e non soddisfano la disuguaglianza triangolare (il lato maggiore è inferiore alla somma degli altri due) e calcolare il perimetro del triangolo.
- 7) Acquisire dall'utente un intero A , assicurarsi che sia maggiore o uguale a 1950 e stampare gli anni in cui si disputano i primi 4 mondiali di calcio successivamente ad A .
- 8) Tema d'esame di ingegneria dell'informazione del 13 gen 2009 (vedi lucido seguente)
- 9) Tema d'esame di ingegneria dell'informazione del 28 gen 2009 (nel lucido successivo)

Dal tema d'esame del 13 gennaio 2009 [Ing. dell'Informazione]

Si sviluppi un programma in linguaggio C che, ricevendo in ingresso una sequenza di lunghezza arbitraria di almeno due numeri interi diversi da zero, terminata da uno zero, produca in uscita i due valori minimi letti in ingresso (escluso l'ultimo zero).

Ad esempio, ricevendo in ingresso la sequenza

7 2 19 4 45 3 7 9 3 0

produce in uscita 2 3

Altro esempio: ricevendo in ingresso la sequenza

7 2 19 4 2 3 7 9 3 0

produce in uscita 2 2

[10]

Dal tema d'esame del 13 gennaio 2009 [Ing. dell'Informazione]

Si sviluppi un programma in linguaggio C che, come nel caso di una macchina distributrice di caffè, riceve in ingresso un numero intero positivo N (corrispondente ad un importo da pagare in centesimi) e, successivamente, una sequenza di numeri interi corrispondenti alle monete inserite, che possono essere da 1, 5, 10, 20 e 50 centesimi. E' richiesto che il programma ripeta l'acquisizione di ciascun numero se non corrisponde ad una moneta tra quelle indicate. Appena l'importo richiesto N viene raggiunto o superato, il programma interrompe l'acquisizione della sequenza e restituisce una serie di numeri interi corrispondenti al resto in monete da 1 e 5 centesimi.

Ad esempio, se il programma riceve $N=101$ e la sequenza

50, 20, 20, 20, produce in uscita 5, 1, 1, 1, 1