ESERCIZI DI PROGRAMMAZIONE

- condizionali e cicli -

Esercizio 1: dal tema d'esame ING-INF del 28 gennaio 2009

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. Il programma deve ripetere 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

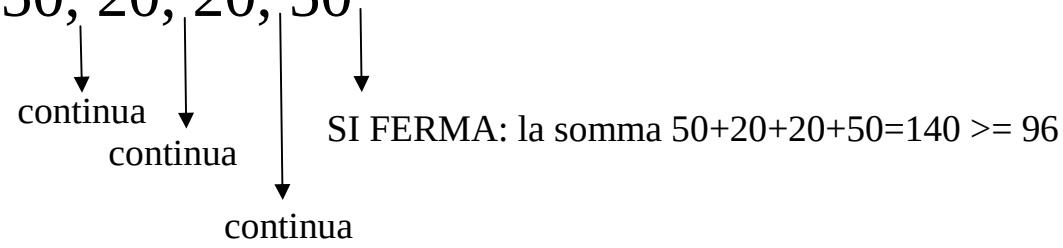
50, 20, 20, 20, produce in uscita 5, 1, 1, 1, 1

NOTA: a scopo didattico si risolverà il problema senza utilizzare le operazioni di divisione intera e di resto, che semplificano il problema
(gli studenti invece erano autorizzati ad usarle)

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

ES: N=96 e inserisco

50, 20, 20, 50



Poi devo restituire un resto di 140-96 = 44 in monete da 5 e 1

"conto" 5-5-5-5… fino ad arrivare a 40
poi "conto" 1-1-1-1 fino ad arrivare a 44

⇒ Si è già individuato un metodo risolutivo!

Secondo passo: una prima scomposizione (tipicamente a mente)

Acquisisci *N* e una successione di monete dall'utente, calcolando l'importo *pagato*

A questo punto *resto*= *pagato*-N

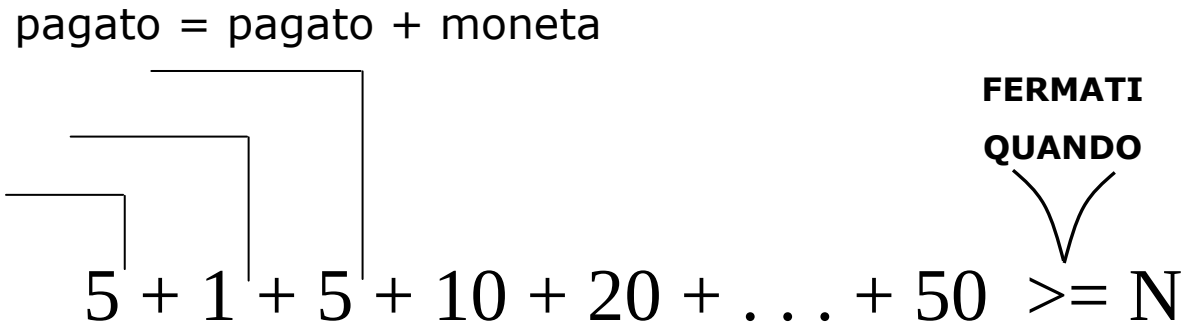
Ciclo per calcolare le monete da 5 (sommo 5 fino a "resto + o –")

Ciclo per calcolare le monete da 1 (sul resto mancante)

Terzo passo: individuare un metodo risolutivo per il primo punto

Dato N importo da pagare:

- devo fare un ciclo che continua ad acquisire una moneta
- scopo del ciclo: calcolare la somma pagata, immagazzinata in una variabile che chiamo *pagato*
- la variabile *pagato* tiene conto della somma parziale



Ultimo passo: sviluppare l'algoritmo:

è importante dare un significato preciso alle variabili!

```
int N, pagato, moneta, resto;

printf("Inserisci l'importo\n");
scanf("%d", &N);

printf("Inserisci sequenza monete\n");

pagato=0;              // somma corrente (già sommata ultima moneta)
while(pagato<N) {      // esci quando somma è maggiore o uguale a N
    <Acquisisci moneta controllandola>;
     pagato=pagato+moneta;
    }

resto=pagato-N;        //somma da restituire!

while(resto>=5){       //esci quando somma da restituire <5
    printf("5 ");
    resto=resto-5;
    }
```

```
//resto contiene ancora la somma da restituire

while(resto>0){    //continua se c'è ancora qualcosa da restituire
    printf("1 ");
    resto=resto-1;
    }
```

A questo punto bisogna solo esplodere la pseudoistruzione

```
    <Acquisisci moneta controllandola>;
```

Ecco come:

```
    do
        scanf("%d", &moneta);
        while(moneta!=1 && moneta!=5 && moneta!=10 && moneta!=20
                && moneta!=50);
```

Programma completo

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

int main()
{
 int N, pagato, moneta, resto;

 printf("Inserisci l'importo\n");
 scanf("%d", &N);

 printf("Inserisci sequenza monete\n");
 pagato=0;              // somma corrente (già sommata ultima moneta)
 while(pagato<N){    // esci quando somma è maggiore o uguale a N
    do
      scanf("%d", &moneta);
    while(moneta!=1 && moneta!=5 && moneta!=10 && moneta!=20 && moneta!=50);
    pagato=pagato+moneta;
 }

 resto=pagato-N;      //somma da restituire!

 while(resto>=5){     //esci quando somma da restituire <5
      printf("5 ");
      resto=resto-5;
  }

  while(resto>0){     //continua se c'è ancora qualcosa da restituire
      printf("1 ");
      resto=resto-1;
  }

  printf("\n");

  system("PAUSE");
  return 0;
}
```

Esercizio 2: dal tema d'esame ING-INF del 13 gennaio 2009

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]

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

7 2 19 4 45 3 7 9 3 0

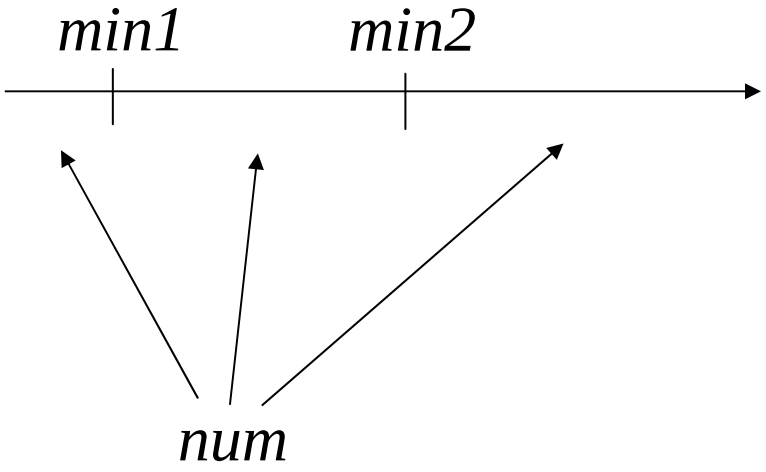
- All'inizio (7, 2) sono i valori minimi
- Considero 19: i minimi rimangono (7, 2)
- Considero 4: i minimi diventano (4, 2)
- Considero 45: i minimi rimangono (4, 2)
- Considero 3: i minimi diventano (3, 2)

…

⇒ Già si intravede un metodo risolutivo (passo2):

- usare un ciclo per acquisire i numeri (termina con 0)
- mantenere due variabili *min1* e *min2* con i minimi
- ad ogni iterazione, acquisire un nuovo numero e aggiornare di conseguenza *min1* e *min2*

Nota: come faccio ad aggiornare *min1* e *min2*?



E’ più facile se mantengo *min1* e *min2* ordinati (*min1<=min2*)

Terzo passo: sviluppare l'algoritmo

Acquisisci i primi due numeri e inizializza min1, min2
con i due numeri stessi in modo che valga min1<=min2

```
scanf("%d", &n);
while(n!=0){                    //min1 e min2: minimi correnti ordinati
    aggiorna min1 e min2 tenendo conto di n;
    scanf("%d%, &n);
    }
```

Terzo passo: sviluppare l'algoritmo

```
int num, min1, min2;

printf("Inserisci la sequenza di numeri\n");

scanf("%d", &num);
min1=num;
scanf("%d", &num);
if(min1<=num)
  min2=num;
  else{
     min2=min1;
     min1=num;
     }

…
```

```
…
scanf(“%d”, &num);
while(num!=0){
    if(num<min1){
       min2=min1;
       min1=num;
       }
    else if(num<min2)
            min2=num;
    scanf(“%d”, &num);
}
printf(“I numeri minimi sono %d e %d\n”, min1, min2);
```

Esercizio 3

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:

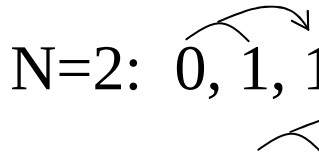
$F(0) = 0$
$F(1) = 1$
$F(K) = F(K-1)+F(K-2)$ per $K \geq 2$

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)

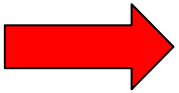
N≥2: uso un contatore I che arriva fino a N:

- 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

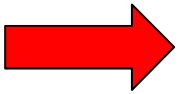
Terzo passo: sviluppare l'algoritmo

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

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)
```

Aggiorna FIBU

Aggiorna FIBP

FIBU = FIBP + FIBU;
FIBP = ? FIBU ?

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 4

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

Esercizio 4

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 4

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");
}
```