Laboratorio di Metodi Informazionali

Laurea in Bioinformatica

Docente: *Carlo Drioli*

Web: www.scienze.univr.it/fol/main?ent=oi&id=39988

Lezione 4bis

Programmazione:

Dati

Strutture di controllo

Algoritmi

Materiale tratto dai lucidi ufficiali a corredo del testo:

D. Sciuto, G. Buonanno e L. Mari

"Introduzione ai sistemi informatici"

2005 - McGrawHill

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Problemi e algoritmi

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Definire il problema

- Eliminare le ambiguità nella formulazione del problema
- Individuare il risultato che si vuole ottenere, gli obiettivi da raggiungere
- Evidenziare
  - le regole da rispettare
  - i vincoli interni ed esterni
  - i dati espliciti ed impliciti
- Eliminare i dettagli inutili ed ambigui

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Soluzioni ed esecutori

- **La descrizione della soluzione di un problema dipende dall'esecutore**
  - esecutore con un livello medio di scolarità ⇒
    "**determina la superficie *s* di un cerchio di cui è noto il raggio *r***":
  - esecutore che non conosce come calcolare l'area del cerchio ⇒
    "**la superficie di un cerchio è $s = \pi r^2$**":
  - esecutore che non conosce π π ⇒
    "**la superficie di un cerchio è $s = \pi r^2$ e π indica *pi greco* ed è 3.1415**":
  - … ⇒ "**eleva al quadrato il raggio e quindi moltiplica il risultato per pi greco**":
  - … ⇒ "**moltiplica il raggio per se stesso e poi il risultato per 3.14**":
  - …

- **Descrizione della soluzione di un problema sia accettabile per un esecutore**
  - si scompone il problema originario in sottoproblemi;
  - si scompongono i sottoproblemi in sotto-sottoproblemi;
  - si prosegue nella scomposizione fino a giungere a ***problemi elementari*** (o ***primitivi***), cioè problemi che possono essere risolti direttamente dall'esecutore

Procedura effettiva

- **Si dice *procedura effettiva per un esecutore* una successione di azioni tale che:**
  - tutte le azioni della successione sono *elementari* per l'esecutore, che è in grado di eseguire ciascuna di esse in un tempo finito e in modo deterministico, cioè ottenendo sempre gli stessi risultati a parità di input;
  - è fissato l'ordine di esecuzione delle azioni;
  - è esplicitamente specificato il modo in cui un'azione utilizza i risultati delle azioni che la precedono.

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Soluzione effettiva

- **Dati un problema *P* e un esecutore *E*, si definisce *soluzione effettiva del problema P per l'esecutore E* una successione di istruzioni elementari tale che:**
  - l'esecutore è in grado di interpretare le istruzioni nella successione e quindi di associare a ciascuna di esse l'azione (o la successione di azioni) che deve compiere per eseguirla;
  - la successione di azioni risultante dall'interpretazione delle istruzioni costituisca una procedura effettiva per l'esecutore stesso.

- **In generale, possono esistere diverse soluzioni effettive dello stesso problema per lo stesso esecutore**

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Algoritmo (definizione informale)

- **Sequenza finita di istruzioni,**
- **comprensibili da un esecutore (si può trattare di uno strumento automatico),**
- **che descrive come realizzare un compito (come risolvere un "problema").**

- **Alcuni esempi**
  - Istruzioni di montaggio di un elettrodomestico
  - Uso di un terminale Bancomat
  - Calcolo del massimo comune divisore di numeri naturali

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Esecutori e linguaggi

- **Un esecutore è definito in base a tre elementi:**
  - l'insieme delle **operazioni** che è capace di compiere;
  - l'insieme delle **istruzioni** che capisce (**sintassi**);
  - quali **operazioni** associa ad ogni **istruzione** che riconosce (**semantica**).
- **Il calcolatore "capisce" le istruzioni che fanno parte del linguaggio macchina**
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore
- **La soluzione si dice *effettiva* se l'esecutore è in grado di:**
  - interpretarla
  - compiere le azioni (in un tempo finito!)

Dal problema alla soluzione automatica

- **Specifiche dei requisiti:**
  descrizione precisa e corretta dei requisiti (verificabilità) ---> cosa?
- **Progetto:**
  procedimento con cui si individua la soluzione ---> come?
- **Soluzione: algoritmo**

Proprietà degli algoritmi

- **Correttezza**
  - L'algoritmo perviene alla soluzione del compito cui è preposto, senza difettare di alcun passo fondamentale
- **Efficienza**
  - L'algoritmo perviene alla soluzione del problema usando la minima quantità di risorse fisiche
    - tempo di esecuzione, memoria, …

Alcuni concetti

- ***Algoritmo*** **= descrizione di come si risolve un problema**
- ***Programma*** **= algoritmo scritto in modo che possa essere eseguito da un calcolatore (linguaggio di programmazione)**
- ***Linguaggio macchina*** **= linguaggio effettivamente "compreso" da un calcolatore, caratterizzato da**
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore
- **Due aspetti rilevanti:**
  - **produrre algoritmi** (cioè capire la sequenza di passi che portano alla soluzione di un problema)
  - **codificarli in programmi** (cioè renderli comprensibili al calcolatore)

Algoritmi

Formalizzazione

Codifica degli algoritmi

- **Algoritmo formulato per essere comunicato tra esseri umani**
  - sintetico e intuitivo
  - codificato in linguaggi informali o semi-formali (linguaggio naturale, diagrammi di flusso, …)
- **Algoritmo formulato per essere eseguito automaticamente**
  - preciso ed eseguibile
  - codificato in linguaggi comprensibili dagli esecutori automatici (linguaggio macchina o linguaggio di programmazione di alto livello)

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Algoritmi e variabili

- **Gli algoritmi sono parametrici:**
  - producono un risultato che dipende da un insieme di dati di partenza;
  - descrivono la soluzione non di un singolo problema, ma di una intera **classe di problemi** strutturalmente equivalenti.
  - Esempi:
    - l'algoritmo per la moltiplicazione di due numeri specifica come effettuare il prodotto di *tutte* le possibili coppie di numeri;
    - l'algoritmo per la ricerca di un libro nello schedario della biblioteca vale per tutti i possibili libri;
    - …
- **Le istruzioni dell'algoritmo fanno riferimento a *variabili*, il cui valore non è fissato a priori ma cambia a seconda della situazione elaborativa in cui l'esecutore si trova.**

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

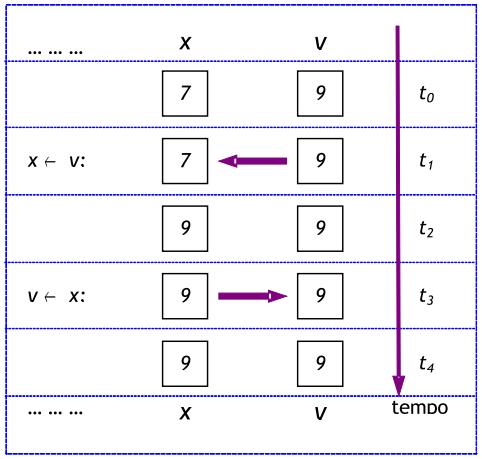
Uso delle variabili

- **All'interno di espressioni.**
  - l'esecutore usa il valore contenuto nelle variabili per calcolare il risultato dell'espressione.
  - per esempio `op1 + op2 × op3` oppure `op1 / op2 - op3`, …
- **in istruzioni di assegnamento**
  - introdurre nel contenitore identificato dal nome della variabile il valore specificato a destra dell'assegnamento;
  - per esempio `r ← 35` (assegna 35 alla variabile il cui nome è `r`), `pi ← 3.14`, …
- **in istruzioni di assegnamento combinate con espressioni**
  - assegna a una variabile il risultato ottenuto dalla valutazione di un'espressione, per esempio in "`circ ← 2 × r × pi`" il risultato dell'espressione `2 × r × pi` viene calcolato utilizzando i valori contenuti nelle variabili `r` e `pi` e il risultato viene poi assegnato alla variabile `circ`;
  - la stessa variabile può comparire in entrambi i lati dell'istruzione di assegnamento, per esempio in "`k ← k + 1`" il valore contenuto in `k` viene utilizzato per trovare il valore dell'espressione `k + 1` che viene memorizzato come nuovo valore di `k`.

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Assegnamento di valori a variabili

- Il valore assegnato a una variabile si **sostituisce** a quello che era presente in precedenza: il vecchio valore non potrà più essere recuperato.
- Esempio: si ipotizzi di voler **scambiare** i valori contenuti in due variabili ***x*** e ***v***.
- Soluzione proposta: doppio assegnamento del tipo
  x* ← *v
  v* ← *x
  per indicare che il valore di ***v*** deve essere copiato in ***x*** e che, nello stesso tempo, il valore di ***x*** sia trasferito in ***v***.
- Le istruzioni però vengono eseguite in **sequenza**! Quindi l'assegnamento ***x* ← *v*** viene completato prima di iniziare ***v* ← *x***.

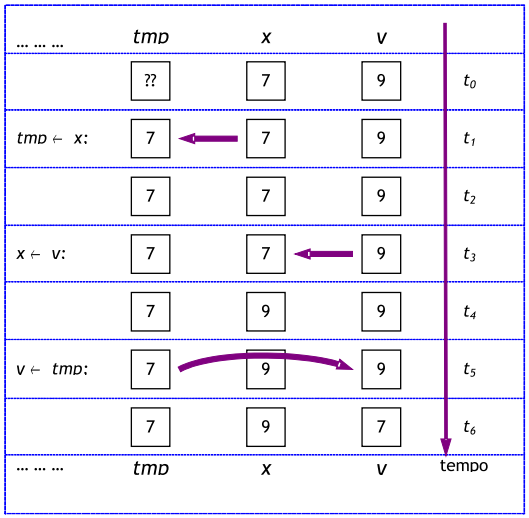


Assegnamento di valori a variabili

Soluzione corretta: uso di una variabile aggiuntiva (***tmp***), come strumento di memorizzazione temporanea ("**buffer**") del valore originariamente contenuto in ***x***

tmp* ← *x
x* ← *v
v* ← *tmp

In questo modo lo scambio avviene senza perdere i valori originari



Dati e istruzioni

- Tipi di dati
  - Numeri naturali o interi o reali (1, -2, 0.34)
  - Caratteri alfanumerici (A, B, ..)
  - Dati logici o booleani (Vero, Falso)
  - Array o vettore di n elementi ({1,2,3})
- Istruzioni
  - Operazioni di Input/Output (es. *leggi*, *scrivi*)
  - Operazioni Aritmetico-logiche (es. *max* = *A* + *B*)
  - Strutture di controllo (es. *SE*, *RIPETI*)

Operazioni elementari

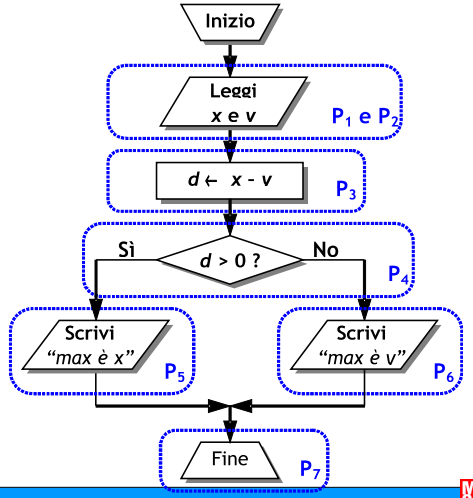
- **Operazioni aritmetiche e assegnamenti di valori a singole variabili**
  - Es. C ← (A + B)
- **Condizioni sul valore di singole variabili**
  - se (A > B) allora … altrimenti …
- **Lettura e scrittura di variabili**
  - "Leggi A" oppure "Stampa B"

Rappresentazione degli algoritmi

- Linguaggio naturale
- Diagramma a blocchi
- Pseudo codice
- Linguaggio di programmazione

Rappresentazione degli algoritmi

- Linguaggio Naturale
  - Leggi due numeri
  - Confrontali
  - Stampa un messaggio che dica quale dei due numeri e' il massimo
  - ...
- Diagramma di flusso



Rappresentazione degli algoritmi

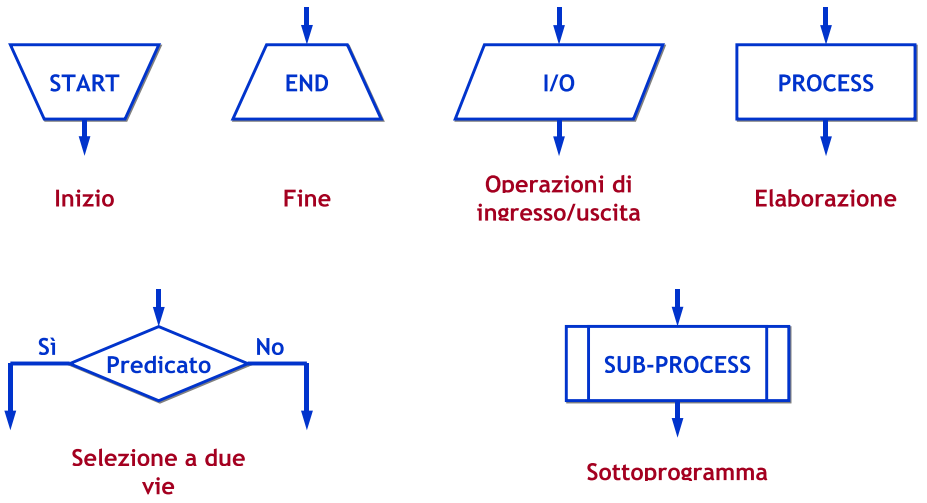
- Pseudo Codice

```
leggi alfa, beta
prod ⇦ 0
finché alfa ≠ 0 ripeti
    prod ⇦ prod + beta;
    alfa ⇦ alfa - 1;
stampa prod;
```

- Linguaggio di programmazione

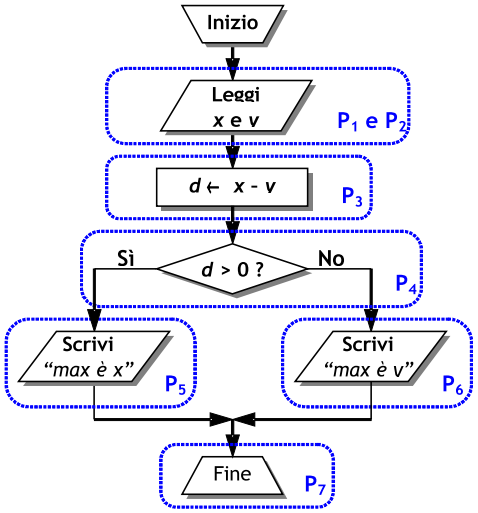
```
#include <stdio.h>

int main (void) {
  puts ("ciao mondo!");
  return Exit  success;
}
```


Diagrammi di flusso



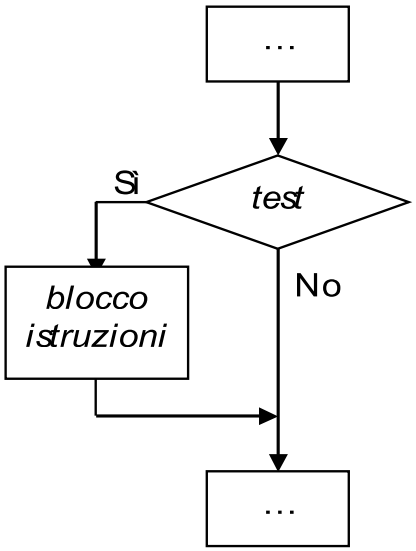
Esempio

Esempio:
dati in ingresso due numeri X e Y, si calcoli e stampi il maggiore.

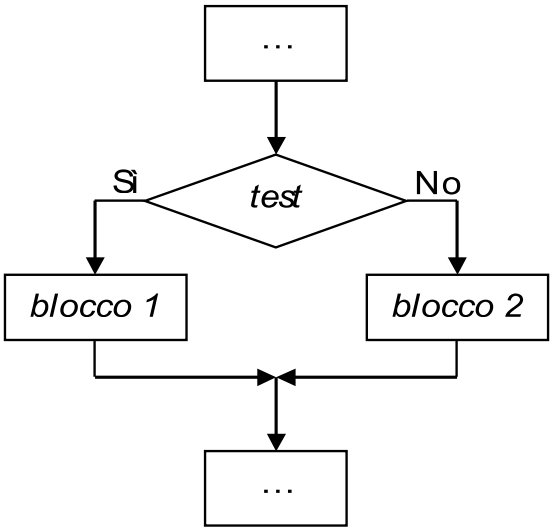


Le strutture di controllo

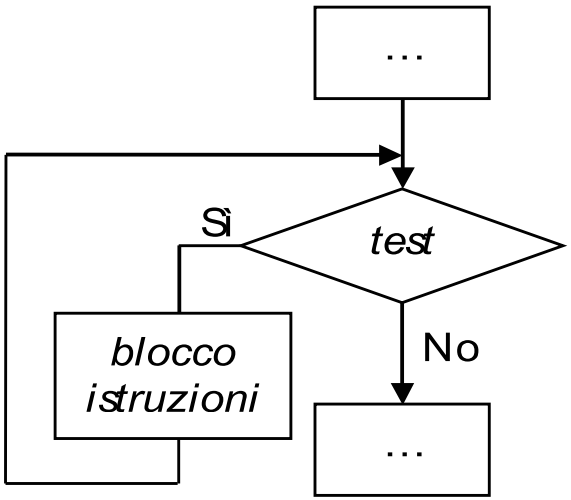
Selezione semplice



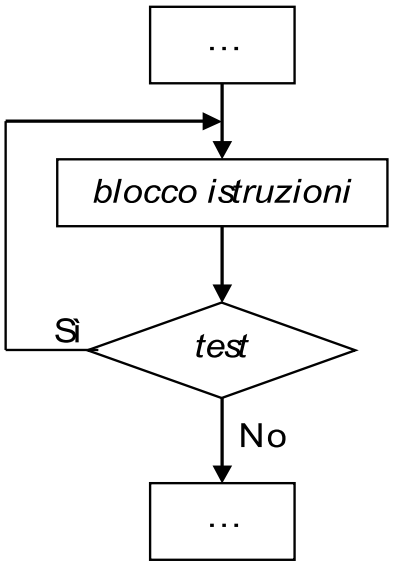
Selezione a due vie



Ciclo a condizione iniziale



Ciclo a condizione finale



Operazioni Logiche (Algebra Booleana)

Algebra di Boole

- L'algebra di Boole (dal suo inventore G. Boole) serve a descrivere le operazioni logiche.
- Componenti dell'algebra di Boole:
  - Operatori booleani
  - Regole di trasformazione ed equivalenza tra operatori booleani
- Gli operandi booleani assumono solo due valori:

  Vero/Falso True/False 1/0 Sì/No ...

Operatori e tavole di verità

A	not A
0	1
1	0

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A ≡ B
0	0	1
0	1	0
1	0	0
1	1	1

A	B	A nand B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0

Notazione

- Esistono convenzioni diverse:
  - **Negazione** not A ¬ A A ! - A
  - **Congiunzione** A and B A ∧ B A & B A × B
  - **Disgiunzione** A or B A ∨ B A | B A + B
  - **Disgiunzione esclusiva** A xor B A ^ B A ⊕ B
    [equivale a (A and (not B)) or ((not A) and B)]
  - **Implicazione** (se … allora) A → B A ⊃ B A ⇒ B
  - **Doppia implicazione** (se e solo se) A ↔ B A ≡ B A ⇔ B

Tabella della verità

a	b	c	a × b	-a × c	(a × b) + (-a × c)
F	F	F	F	F	F
F	F	T	F	T	T
F	T	F	F	F	F
F	T	T	F	T	T
T	F	F	F	F	F
T	F	T	F	F	F
T	T	F	T	F	T
T	T	T	T	F	T

I programmi

Alcuni concetti

- ***Algoritmo*** = descrizione di come si risolve un problema
- ***Programma*** = algoritmo scritto in modo che possa essere eseguito da un calcolatore (linguaggio di programmazione)
- ***Linguaggio macchina*** = linguaggio effettivamente "compreso" da un calcolatore, caratterizzato da
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

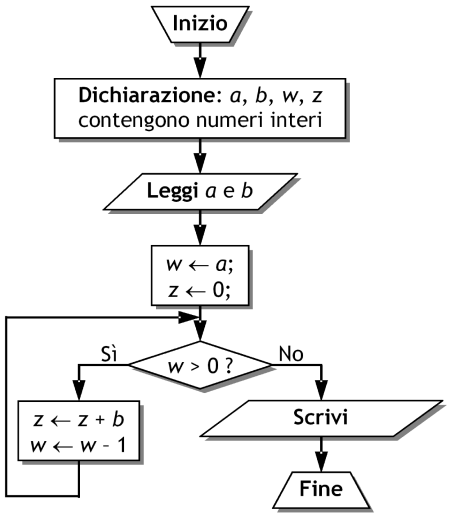
- Due aspetti rilevanti:
  - **produrre algoritmi** (cioè capire la sequenza di passi che portano alla soluzione di un problema)
  - **codificarli in programmi** (cioè renderli comprensibili al calcolatore)

I linguaggi di programmazione

- **Ogni linguaggio di programmazione è caratterizzato da due componenti, complementari l'una con l'altra:**
  - la sua ***sintassi***, cioè l'insieme delle regole che specificano come comporre istruzioni ben formate;
  - la sua ***semantica***, che di ogni istruzione ben formata specifica il significato, e quindi la successione delle operazioni che vengono compiute allorché l'istruzione viene eseguita.

Prodotto di due numeri naturali

```
Dati
   a, b interi positivi
   w, z interi
Risoluzione
   leggi a e b
   z ← 0
   w ← a
   finché w > 0 ripeti
      z ← z + b
      w ← w - 1
   fine ciclo
   scrivi z
   fine
```



Prodotto di due numeri naturali

Algoritmo

```
Dati
 a,b interi positivi
 w,z interi
Risoluzione
 leggi a e b
 z ← 0
 w ← a
 finché w > 0 ripeti
  z ← z + b
  w ← w − 1
 fine ciclo
 scrivi z
 fine
```

Programma in C

```
main() {/* prodotto C */
 unsigned int a,b;
 int w,z;

 scanf("%d %d",&a,&b);
 z = 0;
 w = a;
 while (w > 0) {
  z = z + b;
  w = w − 1;
 }
 printf("%d",z);
}
```

Parti fondamentali di un programma

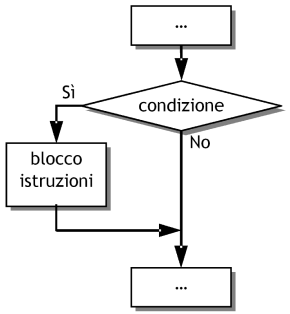
- *Identificazione* del programma
- *Dichiarazione* delle variabili utilizzate, di cui sono indicati tipo e nome
- Specificazione della parte *esecutiva* del programma, detta anche *corpo del programma*

Le istruzioni

- Istruzioni di **ingresso/uscita**
- Istruzioni **aritmetico-logiche**
- Istruzioni **di controllo**

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Selezione semplice

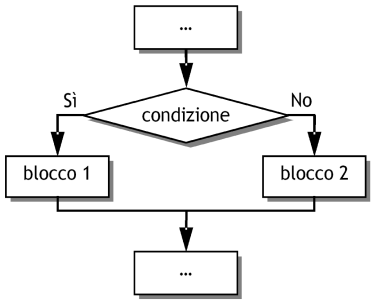


Frammento C

```
main() /*C */
{...
 /*selezione semplice */
 if(condizione){
  /*blocco istruzioni
    eseguito solo se
    condizione è true */
  ...
 }
 ...
}
```

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

Selezione a due vie

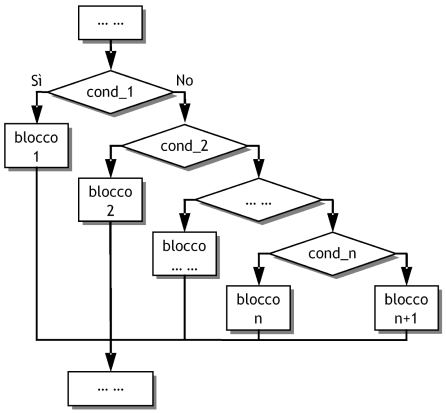


Frammento C

```
main() /*C */
{ ...
  /*selezione a due vie */
  if(condizione){
   ... ... /*blocco 1 */
  }else{
   ... ... /*blocco 2 */
  }
  ...
}
```

Introduzione ai sistemi informatici 3/ed
Donatella Sciuto, Giacomo Buonanno, Luca Mari
Copyright © 2005 – The McGraw-Hill Companies srl

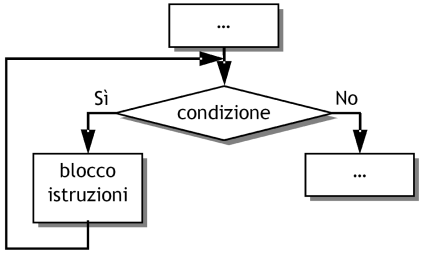
Selezione a più vie



Frammento C

```
main() /*C */
{ ... /*sela più vie */
 if(cond 1){
  ... /*blocco 1 */
 }else if(cond 2){
  ... /*blocco 2 */
 }else if(... ...){
  ... /*blocco ...*/
 }else if(cond n){
  ... /*blocco n */
 }else {
  ... /*blocco n+1*/
 }
 ...
}
```

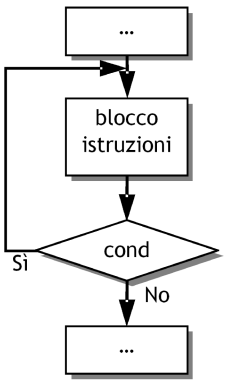

Ciclo a condizione iniziale



Frammento C

```
main() /*C */
{... ... /*ciclo a condizione iniziale */
 while (condizione){
  ... ... /*blocco istruzioni*/
  /*ripetuto finché condizione è true */
 }
 ... /*eseguito quando condizione è false */
}
```


Ciclo a condizione finale



Frammento C

```
main() /*C */
{... /*ciclo a condizione finale */
 do {
  ... ... /*blocco istruzioni*/
  /*eseguito una volta e
    ripetuto se cond è true */
 }while (cond)
 ... ... /*eseguito se cond è false */
}
```


I dati

- **Ogni variabile è caratterizzata dal suo *tipo*.**
  - Tipi predefiniti: numeri, caratteri, booleani, ...
  - Altri tipi: stringhe, date, ...
- **Variabili strutturate:**
  - Vettori (o array)
  - Record

Variabili strutturate

- **Consentono di riferirsi a più valori reciprocamente correlati come se si trattasse di un'unica variabile aggregata.**
  - *vettori* (*array*)
  - *matrici* (o *array multidimensionali*).
  - *record*

Uso degli array

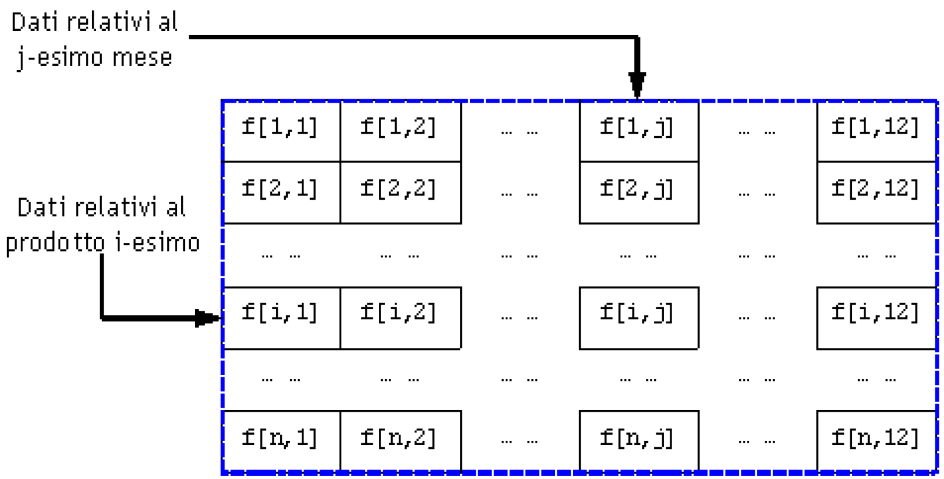
Algoritmo

```
Dati
 n = 100 intero
 f[] vettore di interi
 w, z interi positivi
Risoluzione
 ...
 w ← 1
 z ← 0
 finché (w ≤ n) ripeti
  z ← z + f[w]
  w ← w + 1
 fine ciclo
 scrivi z
 ...
```

Programma in C

```
main() {/* prodotto C */
 int f[100];
 int w, z;
 ...
 w = 0;
 z = 0;
 while (w ≤ 99) {
  z = z + f[w];
  w = w + 1;
 }
 printf("%d", z);
 ...
}
```


Un esempio di matrice



Esempi di algoritmo

Prodotto di due interi positivi

- Leggi *a*
- Leggi *b*
- Somma *b* a se stesso *a* volte
- Scrivi il risultato

Prodotto di due interi positivi

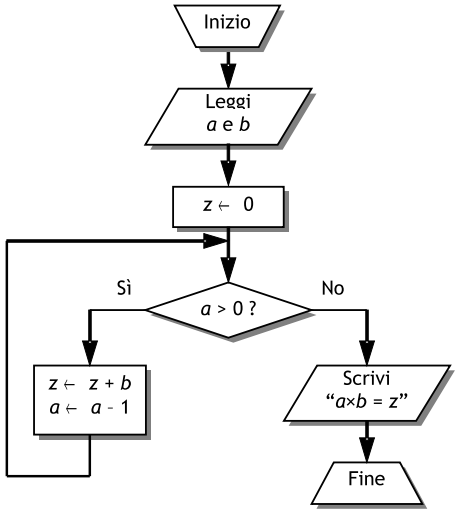
- Leggi *a*
- Leggi *b*
- *z* ← 0
- Se (a ≤ 0) salta a 8
- *z* ← *z* + *b*
- *a* ← *a* - 1
- Torna a 4
- Scrivi "*a* × *b* = *z*"

- Procedimento sequenziale
- Non ambiguo
- Formulazione generale

- Prevede tutti i casi ?? (che succede se *a*<0?)

Prodotto di due interi positivi (2)

- Leggi *a*
- Leggi *b*
- *z* ← 0
- Se (a ≤ 0) salta a 8
- *z* ← *z* + *b*
- *a* ← *a* - 1
- Torna a 4
- Scrivi "*a* × *b* = *z*"



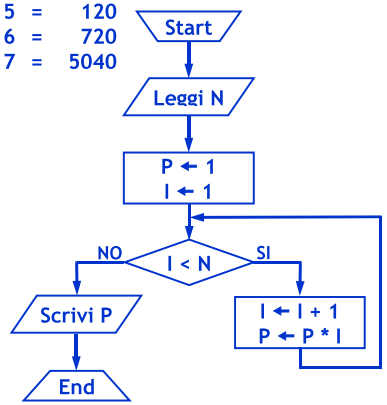
Esercizio

- L'esecutore deve leggere un intero N e restituire N! (fattoriale) dove
  N! = N × (N−1)! = N × (N−1) × (N−2) × … × 1

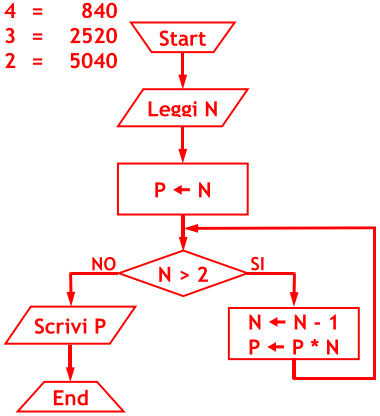
- Attenzione
  - scrivere una prima versione dell'algoritmo immaginando che i dati di ingresso siano sempre corretti (cioè sempre maggiori di zero);
  - scrivere una seconda versione dell'algoritmo in cui sia considerata anche la possibilità che siano inseriti valori inferiori a 1.

Diverse alternative (e.g. 7!)

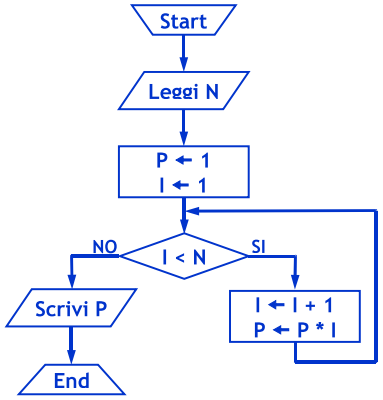
```
1   *  2  =     2
2   *  3  =     6
6   *  4  =    24
24  *  5  =   120
120 *  6  =   720
720 *  7  =  5040
```

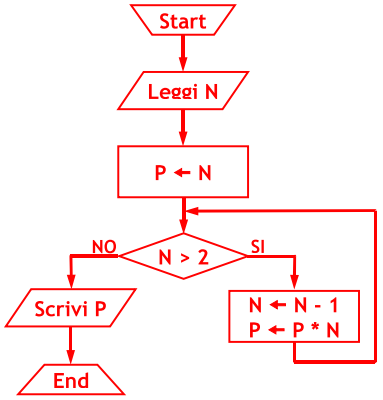



```
7    *  6  =    42
42   *  5  =   210
210  *  4  =   840
840  *  3  =  2520
2520 *  2  =  5040
```



Le alternative sono "diverse"?



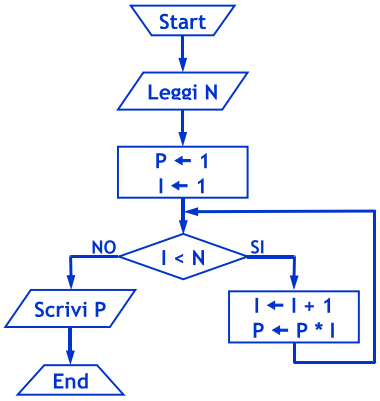


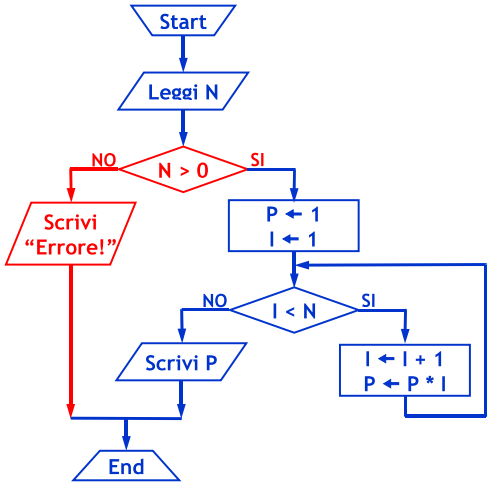
Cosa succede se il dato in ingresso non rispetta le specifiche (N > 0)?
Per esempio, che risultato restituisce l'esecutore per N = 0 ? e per N = -4 ?

N = 0 ⇨ P = 1
N = -4 ⇨ P = 1

N = 0 ⇨ P = 0
N = -4 ⇨ P = -4

Come gestire le "eccezioni"





Algoritmo per il caso "normale".
Come lo modifico per gestire anche i casi che non erano stati previsti?

Somma dei primi N numeri naturali