

Laboratorio di Informatica

Flusso del controllo

Teorema di Bohm e Jacopini

Tutti i programmi possono essere scritti in termini di tre strutture di controllo:

- Sequenza: permette di eseguire le istruzioni secondo l'ordine in cui sono scritte
- Selezione: permette di scegliere il blocco di istruzioni da eseguire in base al valore di una condizione
- Iterazione: permette di eseguire ripetutamente un blocco di istruzioni

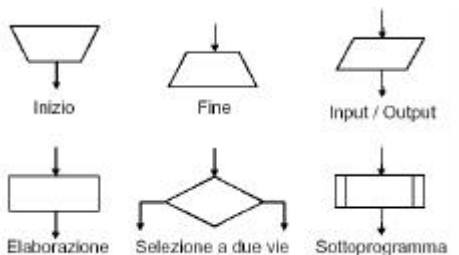
Flusso del controllo

- Le istruzioni di un programma finora sono state eseguite in sequenza
- Spesso è però necessario alterare l'ordine sequenziale per:
 - Scegliere tra azioni alternative (selezione)
 - `if`, `if - else`, `switch`
 - Iterare un blocco di istruzioni (iterazione)
 - `while`, `do ... while`, `for`

Rappresentazione degli algoritmi

- Finora abbiamo affrontato problemi molto semplici, ma per problemi più complessi è opportuno pensare prima all'algoritmo di risoluzione e scriverlo in modo indipendente dal linguaggio di programmazione.
- Questo ci permette di concentrarci sul metodo di risoluzione senza preoccuparci della sintassi del linguaggio e di ottenere un algoritmo indipendente dall'implementazione
- Gli algoritmi vengono solitamente rappresentati con due metodologie:
 - Diagramma di flusso (grafico)
 - Pseudo-codice (simile alla scrittura del programma ma privo di una sintassi rigorosa)

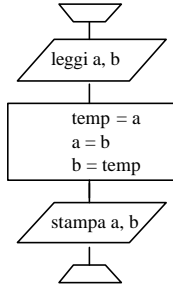
Diagramma di flusso: elementi grafici



Diagrammi di flusso

- Rappresentazione grafica di un algoritmo
 - Si ottiene collegando diversi elementi grafici con frecce di entrata e di uscita
 - Nella programmazione strutturata ogni blocco deve avere una e una sola freccia di entrata e una e una sola freccia di uscita

Flow-chart e pseudo-codice: scambio di variabili



Procedura scambio

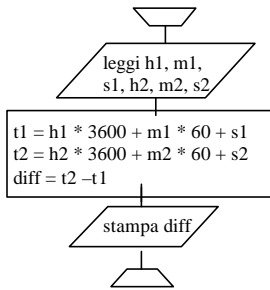
```

leggi a, b
temp = a
a = b
b = temp
stampa a, b
stop
  
```

Esercizio

- Scrivere il diagramma di flusso e lo pseudo-codice del problema: “dati due tempi espressi in ore, minuti e secondi, ne calcoli la differenza, esprimendola in secondi”

Soluzione

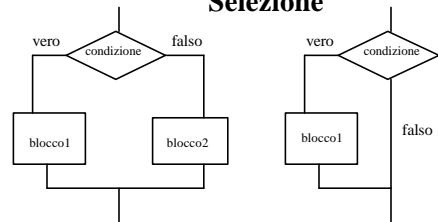


procedura tempi

```

leggi h1, m1, s1, h2, m2, s2
t1 = h1*3600 + m1*60 + s1
t2 = h2*3600 + m2*60 + s2
diff = t2 -t1
stampa diff
stop
  
```

Selezione



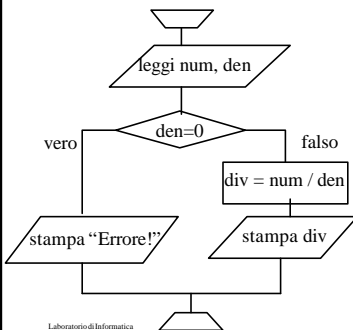
```

if (condizione) then
    blocco 1
else
    blocco2
endif
  
```

```

if (condizione) then
    blocco 1
endif
  
```

Esempio: divisione di due numeri



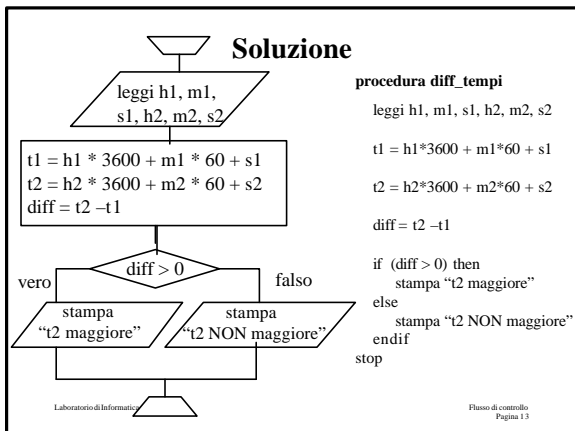
procedura div

```

leggi num, den
if (den =0) then
    scrivi "Errore"
else
    div = num / den
    stampa div
endif
stop
  
```

Esercizio

- Scrivere un programma che, dati due tempi espressi in ore, minuti e secondi, stabilisca qual è il più piccolo



if annidati

- In una selezione ogni blocco può essere a sua volta una selezione, creando una serie di if annidati

Laboratorio di Informatica Flusso di controllo Pagina 14

Esercizio: radici equazione

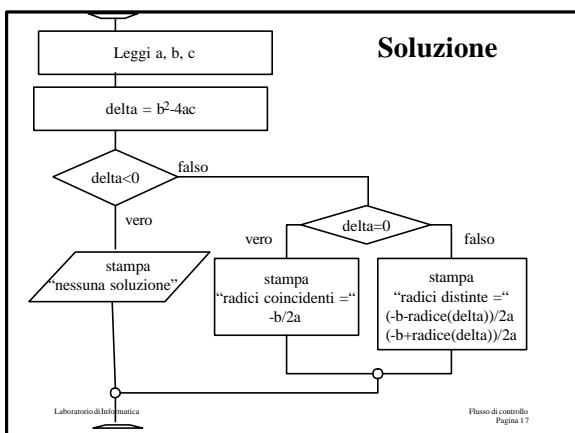
- **Argomenti o ingressi:**
 - **a, b, c** : numeri reali, coefficienti dell'equazione da elaborare
- **Risultati o uscite:**
 - "nessuna radice"
 - " $x_1 = x_2 = r$ " se l'equazione $ax^2 + bx + c$ ha radici coincidenti = **r**
 - " $x_1 = r_1, x_2 = r_2$ " se l'equazione $ax^2 + bx + c$ ha radici distinte = **r1, r2**

Laboratorio di Informatica Flusso di controllo Pagina 15

L'idea dell' algoritmo di soluzione di RADICI

- Risolvo il problema calcolando il discriminante delta dell'equazione, usando le operazioni di prodotto e differenza disponibili nel tipo float;
- analizzo poi i vari casi di delta:
 - < 0
 - $= 0$
 - > 0
- e caso per caso costruisco il messaggio da inviare in uscita.
- Il dettaglio si ottiene con un diagramma di flusso:

Laboratorio di Informatica Flusso di controllo Pagina 16



Pseudo-codice

```

procedura radici
leggi a, b, c
delta = b2 - 4ac
if (delta < 0) then
  stampa "Nessuna radice reale"
else
  if (delta = 0) then
    r = -b/2a
    stampa "Radici coincidenti: " r
  else
    r1 = (-b - sqrt(delta))/2a
    r2 = (-b + sqrt(delta))/2a
    stampa r1, r2
  endif
endif
stop

```

Laboratorio di Informatica Flusso di controllo Pagina 18

Espressioni logiche o booleane

- Andiamo ora ad approfondire il concetto intuitivo della “condizione”.
- Significa valutare un’*espressione logica*, cioè un’espressione che può assumere solo due valori genericamente chiamati:
 - True (vero o 1)
 - False (falso o 0)
- Esempio: (a < b)

C: Operatori relazionali e d’uguaglianza

- Minore di <
- Maggiore di >
- Minore di o uguale a <=
- Maggiore di o uguale a >=
- Uguale a ==
- Diverso da !=

Operatori logici

- And &&
- Or ||
- Not !

```
int a = 7, b = 5;
(a < b) && (b != 0)    → false
(a < b) || (b != 0)    → true
!(a < b)               → true
```

Istruzioni composte e blocchi

- Un’istruzione composta è una sequenza di dichiarazioni e istruzioni racchiusa tra parentesi graffe
- In C ovunque sia possibile scrivere un’istruzione è possibile scrivere anche un’istruzione composta
- Un blocco è un’istruzione composta con almeno una dichiarazione, ma, per abuso di linguaggio, noi chiameremo “blocchi” anche istruzioni composte

C: if

- La forma generale è:

```
if (<espressione>
    <istruzione>
```
- Se espressione è vera allora viene eseguita “istruzione” altrimenti il controllo passa all’istruzione successiva
- “istruzione” può anche essere composta:

```
if (<espressione>
{
    <istruzioni>
}
```

C: if-else

- La forma generale è:

```
if (<espressione>
    <istruzione1>
else
    <istruzione2>
```
- Se “espressione” è vera allora viene eseguita istruzione1, se “espressione” è falsa viene eseguita istruzione2
- “istruzione” può anche essere composta

C: if-else divisione di due numeri

```
leggi num, den
if (den == 0)
    printf ("errore");
else
{
    div = num / den;
    printf ("la divisione è %f", div);
}
```

C: if-else esercizio del minimo

Leggi due numeri a, b e calcolane il minimo

C: if-else esercizio del minimo

Leggi due numeri a, b e calcolane il minimo

```
Leggi a, b
if (a <= b)
    min = a;
else
    min = b;
```

C: else if

Riprendiamo l'esercizio delle radici e scriviamo in C la parte riguardante l'if:

procedura radici	procedura radici
leggi a, b, c delta = b ² - 4ac if (delta < 0) stampa "Nessuna radice reale" else if (delta = 0) r = -b/2a stampa "Radici coinc.: " r else r1 = (-b - sqrt(delta))/2a r2 = (-b + sqrt(delta))/2a stampa r1, r2 stop	leggi a, b, c delta = b ² - 4ac if (delta < 0) stampa "Nessuna radice reale" else if (delta = 0) r = -b/2a stampa "Radici coinc.: " r else r1 = (-b - sqrt(delta))/2a r2 = (-b + sqrt(delta))/2a stampa r1, r2 stop

C: switch

Implementa la scelta tra diverse alternative quando la condizione discriminante è la coincidenza del valore di un'espressione di tipo intero o carattere con un valore dello stesso tipo scelto da un insieme finito

C: switch

Nelle radici abbiamo il problema di non avere costanti ma relazioni.

procedura radici	procedura radici
leggi a, b, c delta = b ² - 4ac if (delta < 0) stampa "Nessuna radice reale" else if (delta = 0) r = -b/2a stampa "Radici coinc.: " r else r1 = (-b - sqrt(delta))/2a r2 = (-b + sqrt(delta))/2a stampa r1, r2 stop	if (delta != 0) absdelta=delta/abs(delta) else absdelta=delta switch (absdelta) { case -1: printf ("Nessuna rad. reale"); break; case 0: r = -b/2a break; case +1: r1 = (-b - sqrt(delta))/2a r2 = (-b + sqrt(delta))/2a break; } Stop

Elementi di C

#define

- E' una direttiva al preprocessore
- Esempio:
#define PIGRECO 3.14159
- Il preprocessore sostituisce tutte le occorrenze dell'identificatore PIGRECO con 3.14159 tranne quelle che compaiono tra apici e nei commenti
- PIGRECO viene chiamata *costante simbolica*
- #define può apparire in qualunque punto del programma e ha effetto solamente sulle righe successive
- **Buone Abitudini:**
 - Le costanti simboliche si nominano usualmente con lettere maiuscole)
 - I define vengono posti usualmente a inizio programma

#define

```
/* area del cerchio */
#include <stdio.h>
#define PIGRECO 3.14159
int main(void)
{
    double raggio = 5.;
    double area = 0.;
    area = PIGRECO * raggio^2;
    printf("\nArea = %lf",area);
    getchar();
    return 0;
}
```

scanf

- Funzione usata per ricevere input da tastiera
- E' analoga a printf
- **scanf (<Stringa di controllo> , <parametri>)**
 - dove <Stringa di controllo> precisa il formato delle variabili specificate in <parametri>
 - NOTA: la scanf richiede che i nomi delle variabili in <parametri> siano preceduti dall'operatore &

scanf

- Es: **scanf ("%d%f", &a, &b)**
Legge un intero e un reale
- Es: **scanf ("%c%c", &a, &b)**
Legge due caratteri
- Es: **scanf ("%c%f", &a, &b)**
Legge un carattere e un reale
- Es: **scanf ("%d%c", &a, &b)**
Legge un intero e un carattere
- NOTA: nella lettura di numeri lo spazio è l'elemento delimitatore.
- NOTA: nella lettura di caratteri lo spazio è un carattere come qualunque altro
- NOTA: gli interi sono identificati dal carattere di conversione "d"

Esercizio "Anno Bisestile"

- Scrivere un programma che richieda in input un numero intero, stabilisca e stampi a video se rappresenta un anno bisestile o meno
- NOTE:
 - bisestili sono tutti gli anni non terminanti con due zeri e divisibili per 4, e quegli anni terminanti con due zeri ma divisibili per 400
 - Scrivere questa condizione in un'unica espressione logica usando gli operatori logici (&&, ||)

Esercizio “Differenza Tempi”

- Scrivere un programma che:
 1. richiedi in input due tempi espressi come 3 interi separati da uno spazio
 2. Controlli che i minuti e i secondi non superino 60 (usare un’espressione logica unica)
 3. Calcoli la differenza di due tempi esprimendola in ore, minuti e secondi
 - Suggerimento: occorre calcolare il resto di divisioni intere.
 - In C l’operatore che determina il resto è “%”
 - Es: (129 % 60) → 9

Esercizio “Temperature-IF”

- Scrivere un programma che visualizzi il seguente messaggio:

```
C:\Materiale Didattica\Chimica\AA2005-2006\Esempi
Scegli che conversione vuoi tra
OK da Celsius a Kelvin
KC da Kelvin a Celsius
CF da Celsius a Fahrenheit
FC da Fahrenheit a Celsius
"
```

- Catturi la scelta dell’utente
- Richiedi la temperatura
- Effettui la conversione richiesta e stampi il risultato
- NOTE:
 - Usare il define per le costanti di conversione
 - Per gestire la scelta di conversione, ricordarsi che:
 - Ogni variabile di tipo char cattura un solo carattere
 - I caratteri vanno indicati tra apici singoli ES: ‘a’

Esercizio “Temperature-SWITCH”

- Riscrivere il programma “Temperature-IF” utilizzando il costrutto “switch”
- NOTA: l’espressione di controllo deve essere di tipo intero