Programmazione strutture di controllo

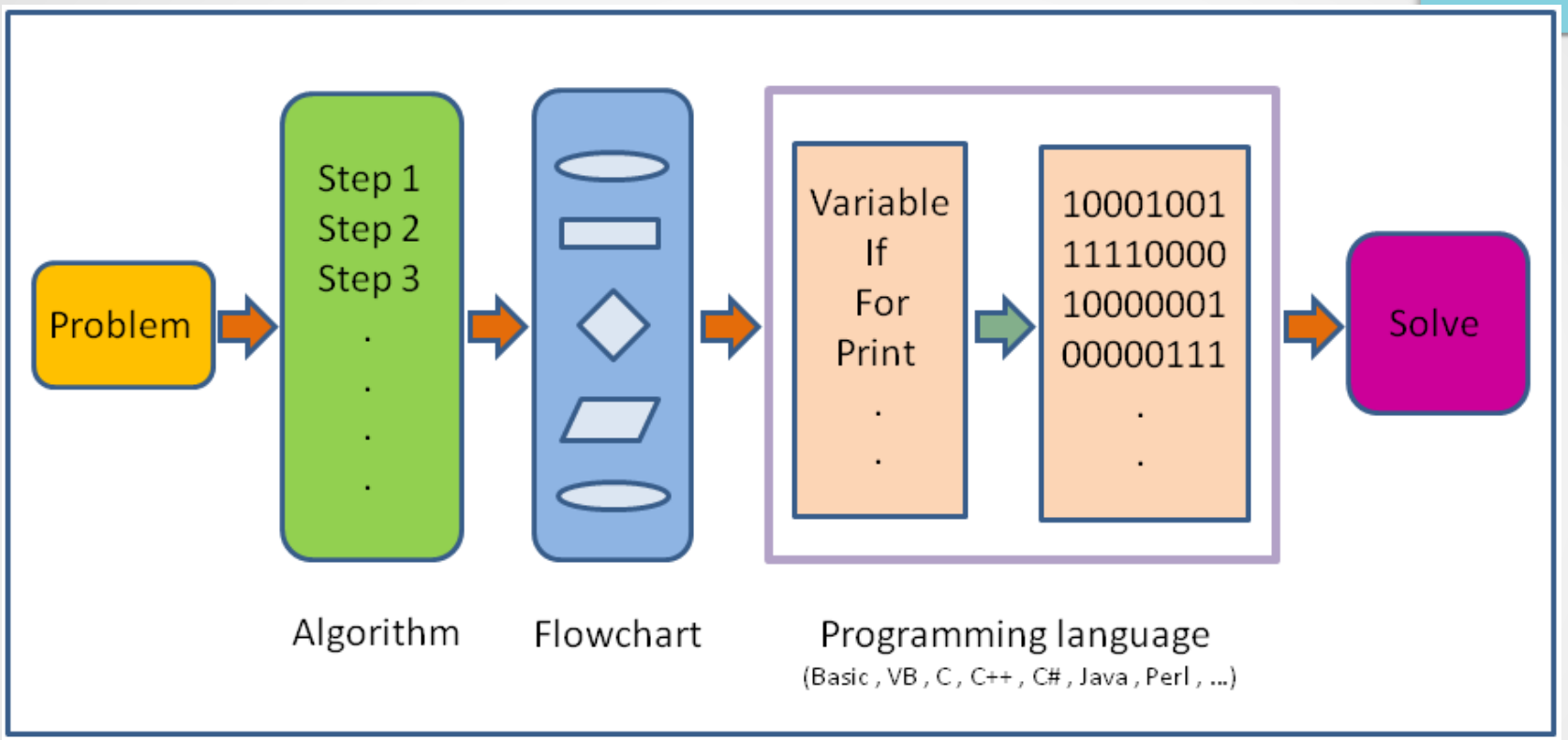
Loriano Storchi

loriano@storchi.org

http::/www.storchi.org/

STRUTTURE DI CONTROLLO

I fondamenti



Ci sono tre strutture fondamentali che sono usate per l risoluzione algoritmica dei problemi : selezione, iterazioni e sequenza (successione delle istruzioni) (il GOTO presente nei linguaggi macchina dagli anni 70 e' stata progressivamente sconsigliata / eliminata)

Esempi: https://bitbucket.org/lstorchi/teaching
https://github.com/lstorchi/teaching

Istruzione di selezione

- La struttura generale di una istruzione di selezione e' la seguente:

if (considzione1)

blocco istruzioni 1

else if (condizione2)

blocco istruzioni 2

...

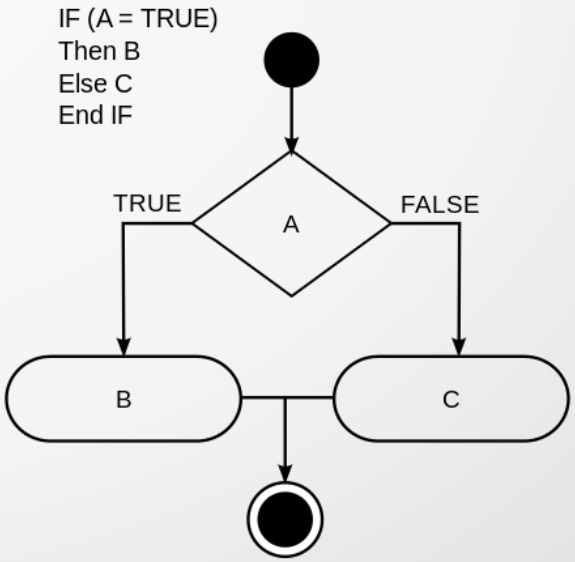
else

blocco istruzioni else

endif

Ci possono essere tanti else if

Non devono necessariamente comparire tutti e tre gli elementi , if, else if ed else , posso anche avere solo un singolo if …



Istruzione di selezione

- La nidificazione, posso nidificare le istruzioni di selezione ovviamente:

```
if (condizione1)
  blocco istruzioni 2
  if (condizione2)
    blocco istruzioni 2
  end if
else
  blocco istruzioni else
end if
```

Operatori di relazione

- In tutti i linguaggi di programmazione posso usare operator di relazione per ad esempio confrontare numeri e variabili:

Description	Java, C, C++	Fortran
Greater than	>	.GT.
Greater than or equal	>=	.GE.
Less than	<	.LT.
Less than or equal	<=	.LE.
Equal	==	.EQ.
Not Equal	!=	.NE.

```
Int a;

a = 4 // operatori di  assegnazione

If  (a == 5)
{
   cout << “Hello” << std::endl;
}
else if (a > 5)
{
    cout <<  a << “ > 5 “ << std::endl;
}
```

Operatori logici

- Diamo per scontate le propriteà degli operatori logici AND, OR e NOT

Logical Operators		
Operator	Description	Example
&&	AND	x=6 y=3 x<10 && y>1 Return True
||	OR	x=6 y=3 x==5 || y==5 Return False
!	NOT	x=6 y=3 !(x==y) Return True

Ad esempio l'espressione

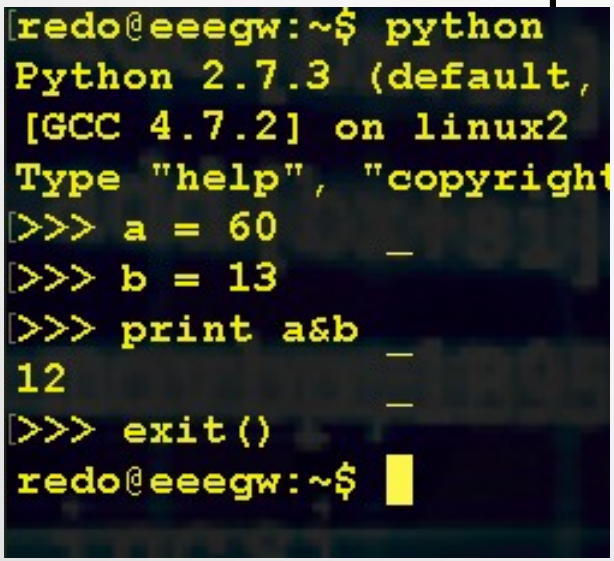
5 < a < 7

Nei linguaggi di programmazione e' spezzata in due espressioni elementari legate dall'operatore AND:

(a > 5) AND (a < 7)

Operazioni bit a bit (bitewise operations)

- Sono le operazioni che vengono usare per manipolare bit a bit i dati, da non confondersi quindi con qunto visto nella slide precedente
- & (AND) | (OR) ^ (XOR) ~ (Complemento ad 1) << (shift a sinistra) >> (shift a destra)
- Facciamo un semplice esempio solo per chiarire:




```
a = 0011 1100

b = 0000 1101

---------------

a&b = 0000 1100
```

Alternativa case

- In buona sostanza una serie di if-then-else con qualche vincolo. In pratica la scelta fra i blocchi di istruzione e' guidata dal valore di una determinata variabile:

```
switch variabile:
   case val1:
      istruzioni1
   case val2:
      istruzioni2
   …
   default:
     istruzionidefault
```

Esempio

```
int i;
i = 4;
switch (i)
{
  case 1:
    printf("vale uno \n");
    break;
  case 2:
    printf("vale due \n");
    break;
  default:
    printf("non uno non due\n");
    break;
}
```

Iterazione

- Si chiama anche ciclo o molto comunemente si usa il termine inglese loop.
- Le tre forme di iterazione piu’ conosciute sono tre: **while..do** , **do..while** e **for**
- Non tutti i linguaggio hanno necessariamente tutti e tre questi strutture
- Questa/e struttura/e permettono di ripetere un blocco di istruzioni fino al verificarsi di una condizione
- Anche in questo caso e’ possibile nidificare piu’ loop uno dentro l’altro

While..do e Do..while

- Il blocco di istruzioni puo' anche non essere mai eseguito, visto che la condizione viene controllata all'inizio, fincheè la condizione è vera viene eseguito il blocco di istruzioni

  WHILE (condizione)

  blocco istruzioni

  END DO

- Il loop do..while invece esegue il blocco di istruzioni finche' la condizione non e' false

  DO

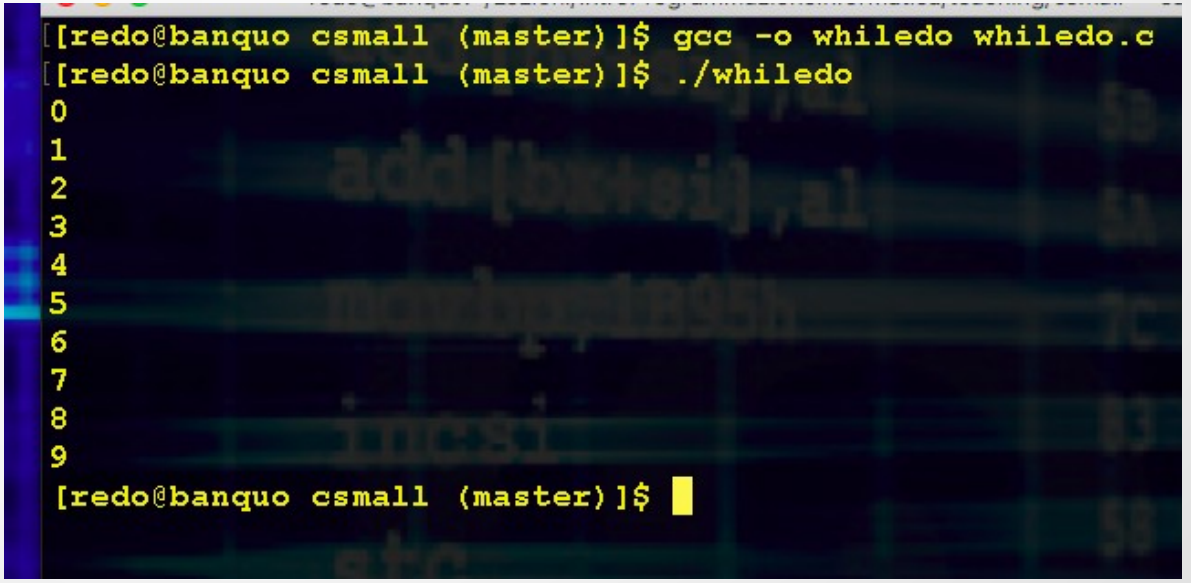
  blocco istruzioni

  WHILE (condizione)

Esempio

- Vediamo qualche esempio in C

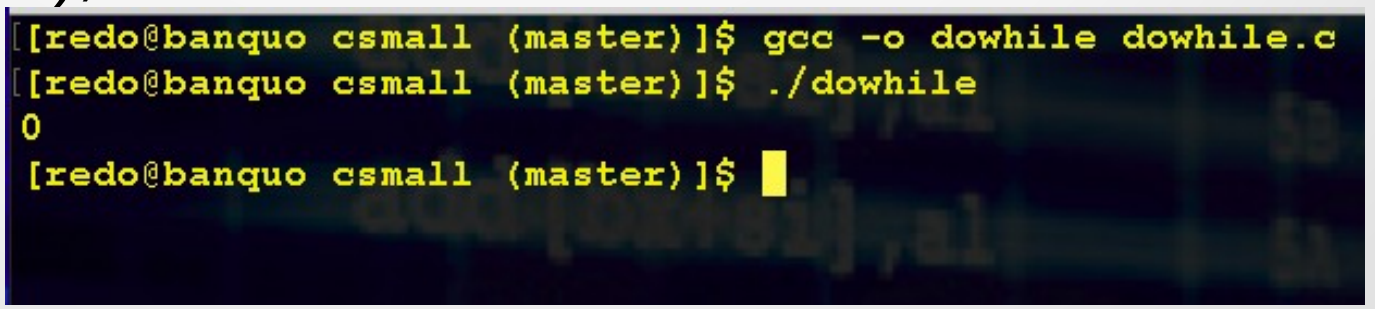
```
int i = 0, N = 10;
while (i != N)
{
   printf("%d n", i);
   i++;
}
```



Esempio

```
Int i = 0, N = 10;


do
{
    printf (“%d \n”, i);
    i++; // i = i + 1
} while (i >= N);
```



For loop

- Esegue un blocco di istruzioni un certo numero di volte noto si dall'inizio. Molti linguaggi di programmazione costringono il programmatore ad usare un contatore.
- Generalmente c'c' dunque un contatore che e' una variabile intera il cui valore viene "variato" passo dopo passo
- La condizione di fine e' generalmente determinata dal confronto della variabile contatore con un valore

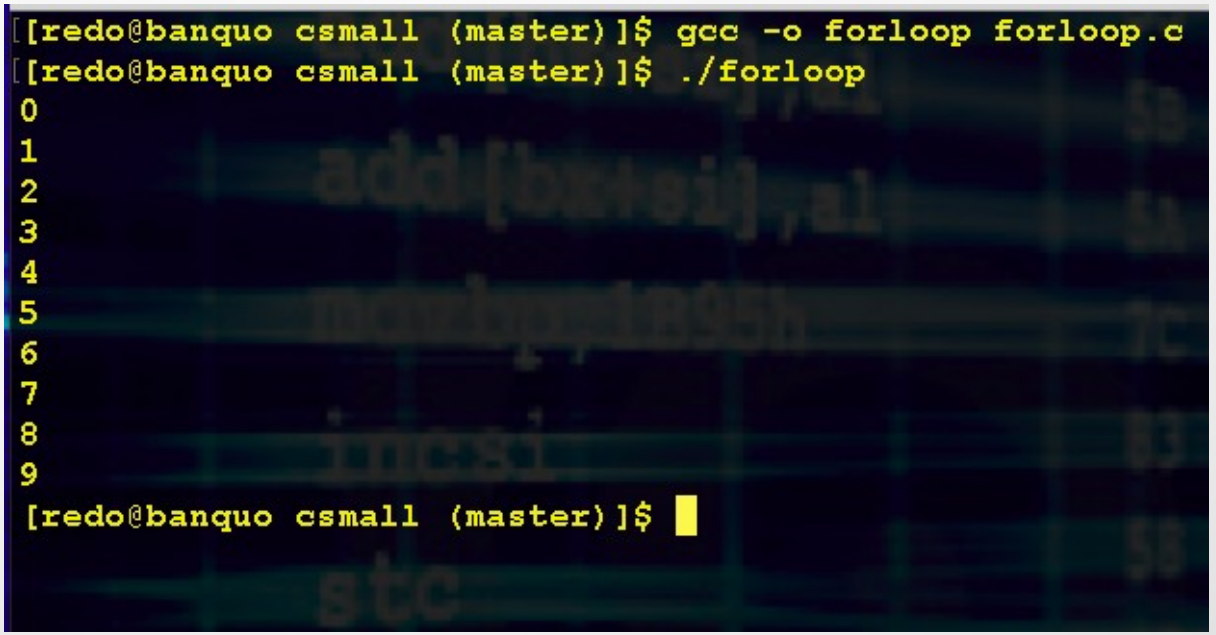
for (contatore = valoreinizio A valorefine PASSO = passo)

blocco_istruzioni

end for

Esempio

```
int i;
for (i=0; i<10; ++i)
{
   printf (“%d \n”, i);
}
```



Array

- Come posso facilmente maneggiare informazioni per loro natura strutturate ? Ad esempio un numero complesso, oppure un vettore od una matrice ?
- Tipiche variabili strutturate sono gli array
  - Ad esempio un vettore di numeri in virgola mobile in C lo possiamo dichiarare come: float v[10];
  - Possiamo poi accedere all'elemento i-esimo del vettore : v[i-1] = 3.5;
- Similmente posso definire una matrice (array bidimensionale) come : float m[10][10];

Esempio

- Esempio i loop per eseguire una moltiplicazione scalare fra vettori

```
float a[N], b[N];
for (i=0; i<N; ++i)
{
  a[i] = (float)rand()/(float)(RAND_MAX/N);
  b[i] = (float)rand()/(float)(RAND_MAX/N);
}
s = 0.0;
for (i=0; i<N; ++i)
{
    s = s + a[i]*b[i];
}
```