Programming, control structures

Loriano Storchi

loriano@storchi.org

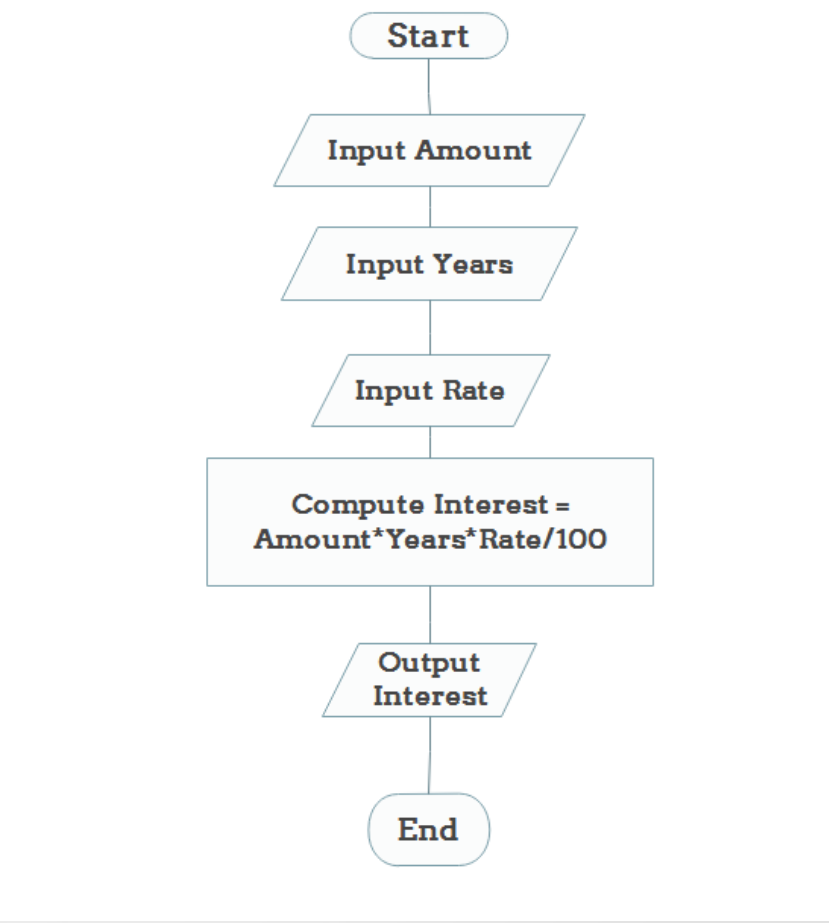
http::://www.storchi.org/

FLOWCHART AND PSEUDOCODE

FLOWCHART AND PSEUDOCODE

- Esamineremo solo alcuni esempi di base: calcolare l'interesse di un deposito bancario

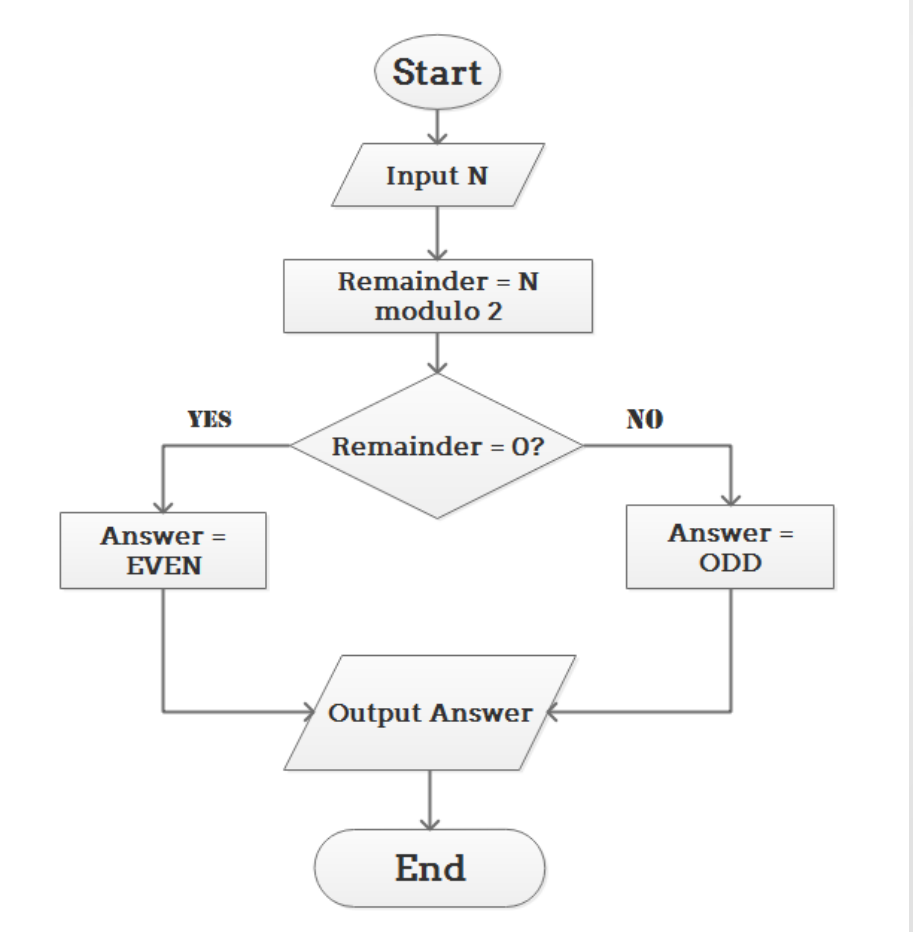
Step 1: Read amount,
Step 2: Read years,
Step 3: Read rate,
Step 4: Calculate the interest with formula
"Interest=Amount*Years*Rate/100
Step 5: Print interest,



FLOWCHART AND PSEUDOCODE

- Determina e mostra se il numero N è pari o dispari

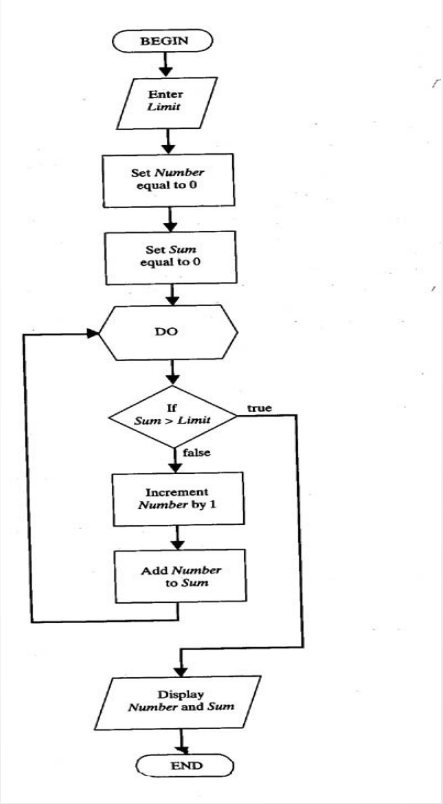
Step 1: Read number N,
Step 2: Set remainder as N modulo 2,
Step 3: If remainder is equal to 0 then number N is even, else number N is odd,
Step 4: Print output.



FLOWCHART AND PSEUDOCODE

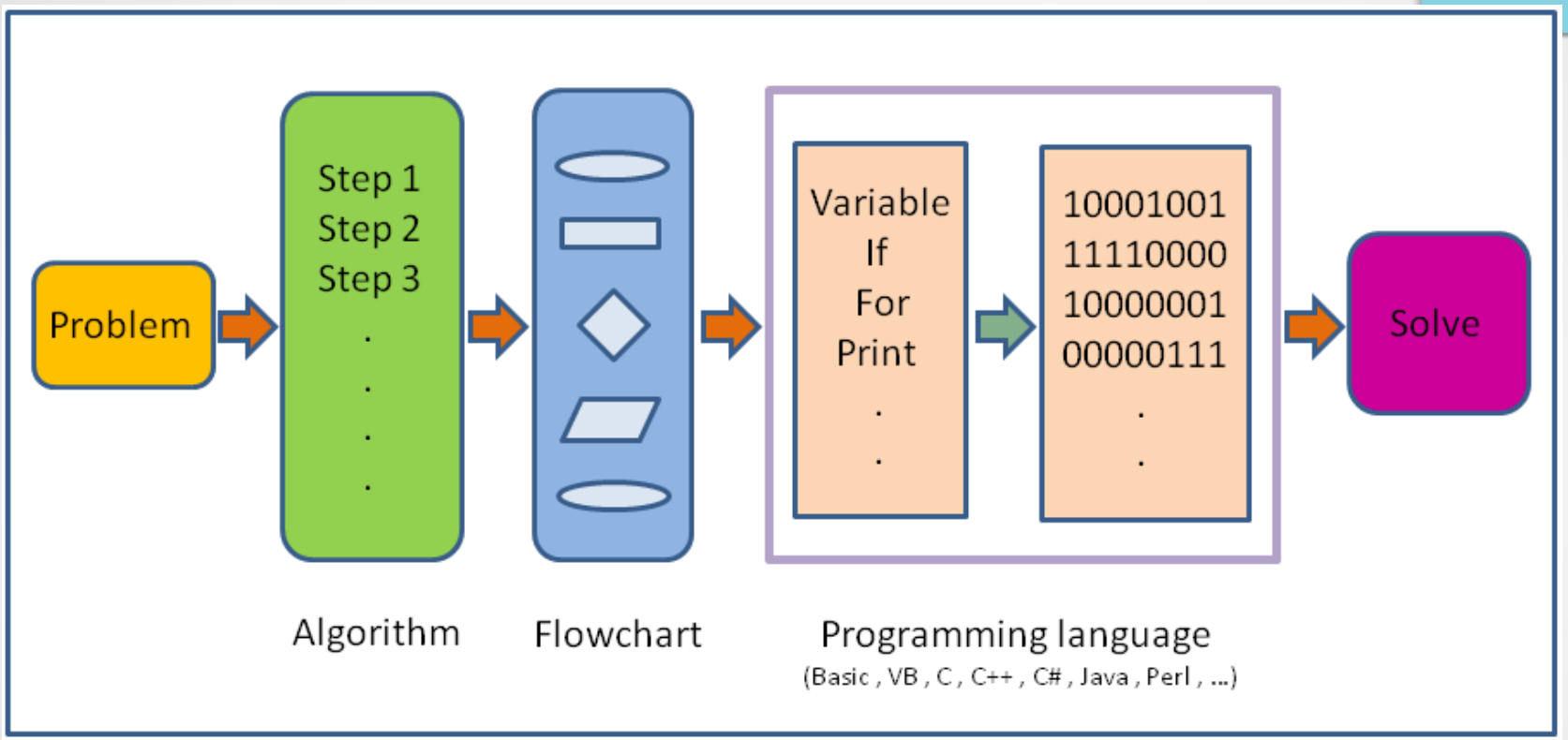
- Per un dato valore, **Limite**, qual è il numero intero positivo più piccolo per cui la somma **Somma = 1 + 2 + ... + Numero** è maggiore di Limite. Qual è il valore di questa somma?

1. Enter Limit
2. Set Number = 0.
3. Set Sum = 0.
4. Repeat the following:
   a. If Sum > Limit, terminate the repitition, otherwise.
   b. Increment Number by one.
   c. Add Number to Sum and set equal to Sum.
5. Print Number and Sum.



STRUTTURE DI CONTROLLO

CONTROL STRUCTURES



Sono tre le strutture fondamentali che vengono utilizzate per la risoluzione algoritmica dei problemi: selezione, iterazioni e sequenza (sequenza di istruzioni) (il GOTO presente nei linguaggi macchina dagli anni '70 è stato progressivamente scoraggiato / eliminato)

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

SEQUENZE

Sequences

- **La struttura di controllo della sequenza** si riferisce all'esecuzione riga per riga mediante la quale le istruzioni vengono eseguite in sequenza, nello stesso ordine in cui appaiono nel programma. Potrebbero, ad esempio, eseguire una serie di operazioni di lettura o scrittura, operazioni aritmetiche o assegnazioni a variabili.

Step 1: Read amount,
Step 2: Read years,
Step 3: Read rate,
Step 4: Calculate the interest with formula
"Interest=Amount*Years*Rate/100
Step 5: Print interest,

SCELTE

Selections

- La struttura generale di un'istruzione di **selezione** è la seguente:

if (condition 1)

statements 1

else if (condition 2)

statements 2

...

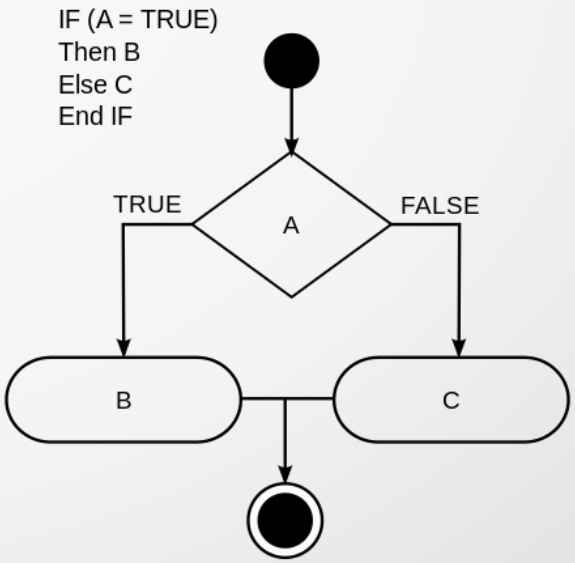
else

statements N

endif

Ci POSSONO ESSERE MOLTI else if

Non devono necessariamente mostrare tutti e tre gli elementi, IF, ELSE IF e ELSE, posso anche avere un solo IF



Selections

- **Annidamento**,posso ovviamente avere if..then.else annidati:

```
if (condition 1)
  statement 1
  if (condituion 2)
    statements 2
  end if
else
  statements 3
end if
```

Operators

- In tutti i linguaggi di programmazione, posso utilizzare operatori di relazione per confrontare numeri e variabili, ad esempio:

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;
}
```

Logical Operators

- Dò per scontati gli 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

Esempio la seguente disuguaglianza

5 < a < 7

Nei linguaggi di programmazione è suddiviso in due espressioni elementari collegate dall'operatore AND:

(a > 5) AND (a < 7)

Bitwise operations

- Non confondere il precedente con le operazioni bit per bit
- Queste sono le operazioni che servono per manipolare dati bit per bit, da non confondere con quanto visto nella diapositiva precedente
  - & (AND)
  - | (OR)
  - ^ (XOR I.e. 1 XOR 1 is zero)
  - ~ (ones' complement i.e. 0 to 1 and 1 to 0)
  - >> (right shift 11100101 >> 1 is 01110010)
  - << (left shift)

Bitwise operations

- Un semplice esempio:

```
└─ $ python3
Python 3.6.9 (default, A
[GCC 8.4.0] on linux
Type "help", "copyright"
>>> a = 60
>>> b = 13
>>> print(a&b)
12
>>> exit()
```

```
a = 60
b = 13

print(a&b)
```

```
12
```

```
a = 0011 1100

b = 0000 1101

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

a&b = 0000 1100
```

Case structure

- Fondamentalmente una serie di if-then-else con qualche vincolo. In pratica, la scelta tra i blocchi di istruzione è guidata dal valore di una certa variabile:

```
switch variable:
   case val1:
      statements 1
   case val2:
      statements 2
   …
   default:
     default statements
```

```
If (variable == val1)
{
    Statements 1
}
else if (variables == val2)
{
   Statements 2
}
…
else
{
  Default statements
}
```

Example

```
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;
}
```

LOOPS

Loops

- Ci sono tre diversi tipi: while..do, do.. while e for
- Non tutti I linguaggi hanno necessariamente tutte e tre queste strutture
- Queste strutture consentono di ripetere un blocco di istruzioni finché non si verifica una condizione
- Anche in questo caso è possibile annidare più loop uno dentro l'altro

While..do e Do..while

- Il blocco di istruzioni inoltre non può mai essere eseguito, poiché la condizione è verificata all'inizio, finché la condizione è vera, il blocco di istruzioni viene eseguito

  WHILE (condition)

  statements

  END DO

- do..while esegue invece il blocco di istruzioni finché la condizione non è falsa

  DO

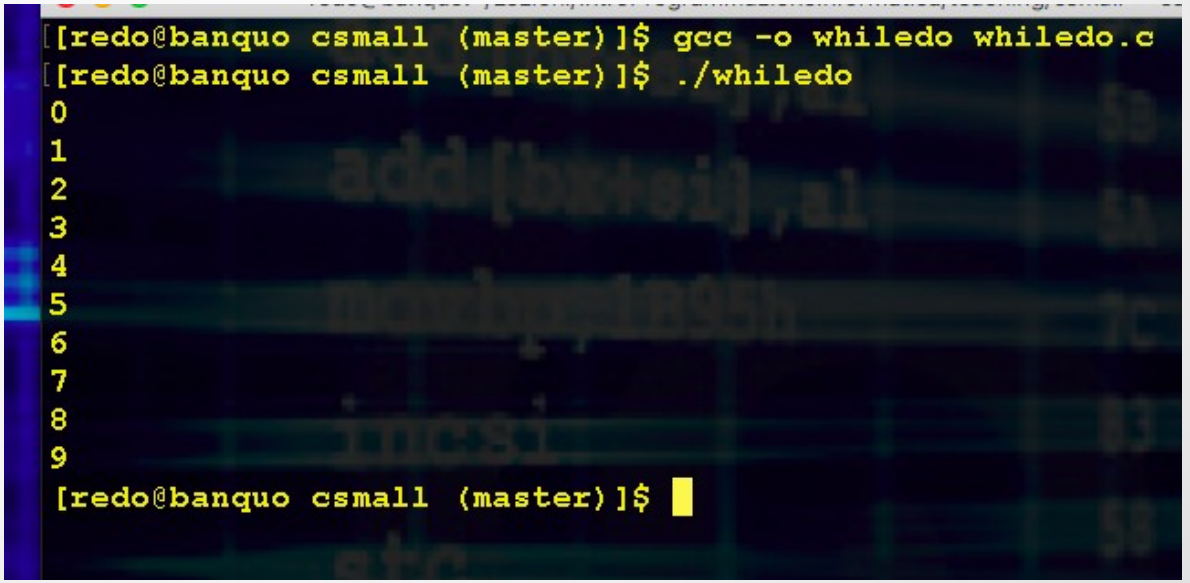
  statements

  WHILE (condition)

Example

- A simple C example:

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



Example

```
Type "help", "copyright"
>>> while True:
...     print("here")
...
```

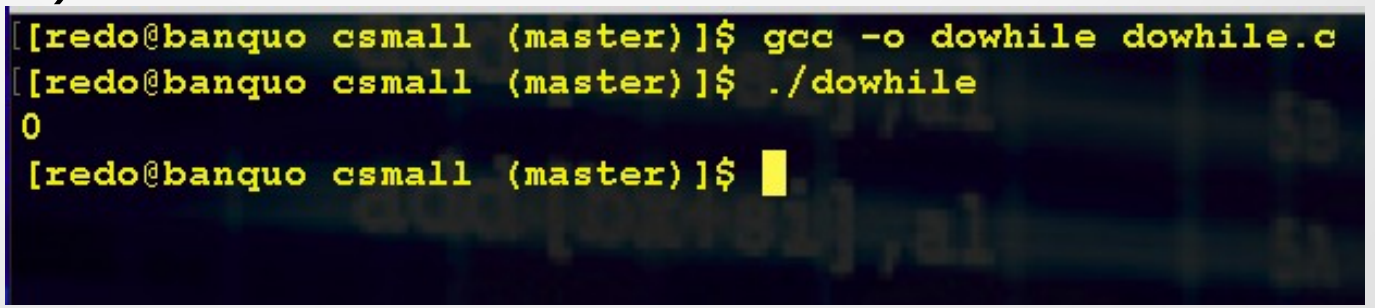
```
while True:
    print("here")
```

```
here
here
here
here
here
here
here
here
here
here
here
```

Example

```
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 numero di volte noto dall'inizio. Molti linguaggi di programmazione costringono il programmatore a utilizzare un contatore.
- Generalmente esiste un contatore che è una variabile intera il cui valore viene "modificato" passo dopo passo
- La condizione finale viene generalmente determinata confrontando la variabile contatore con un valore

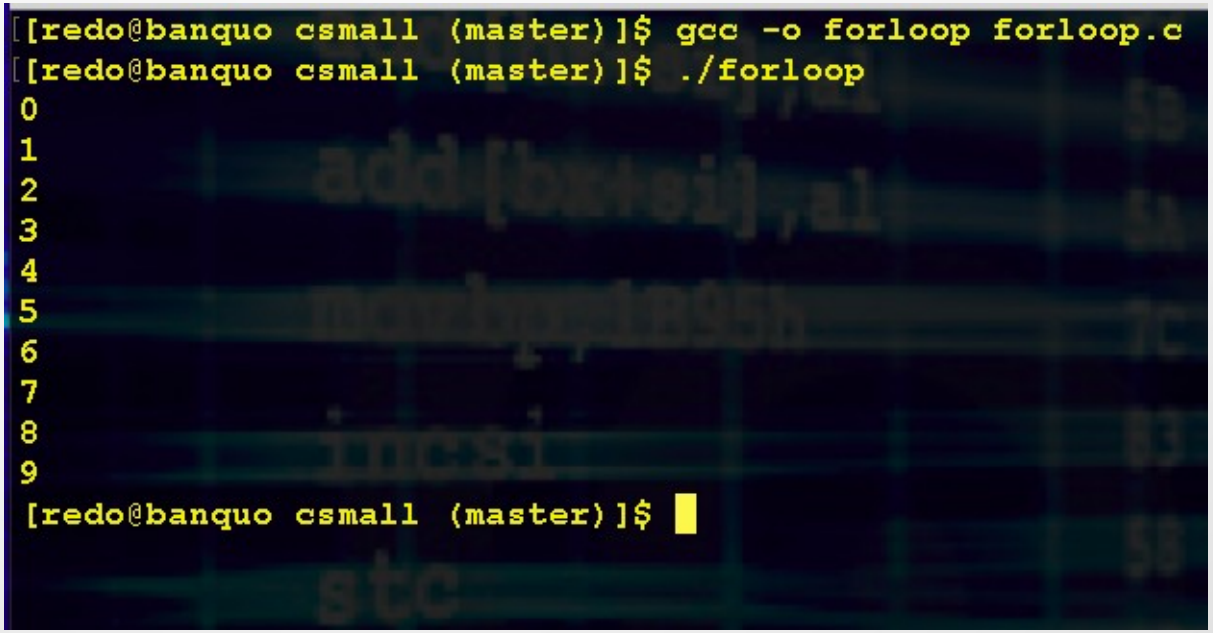
for (counter = startingvalue A endvalue STEP = stepvalue)

statements

end for

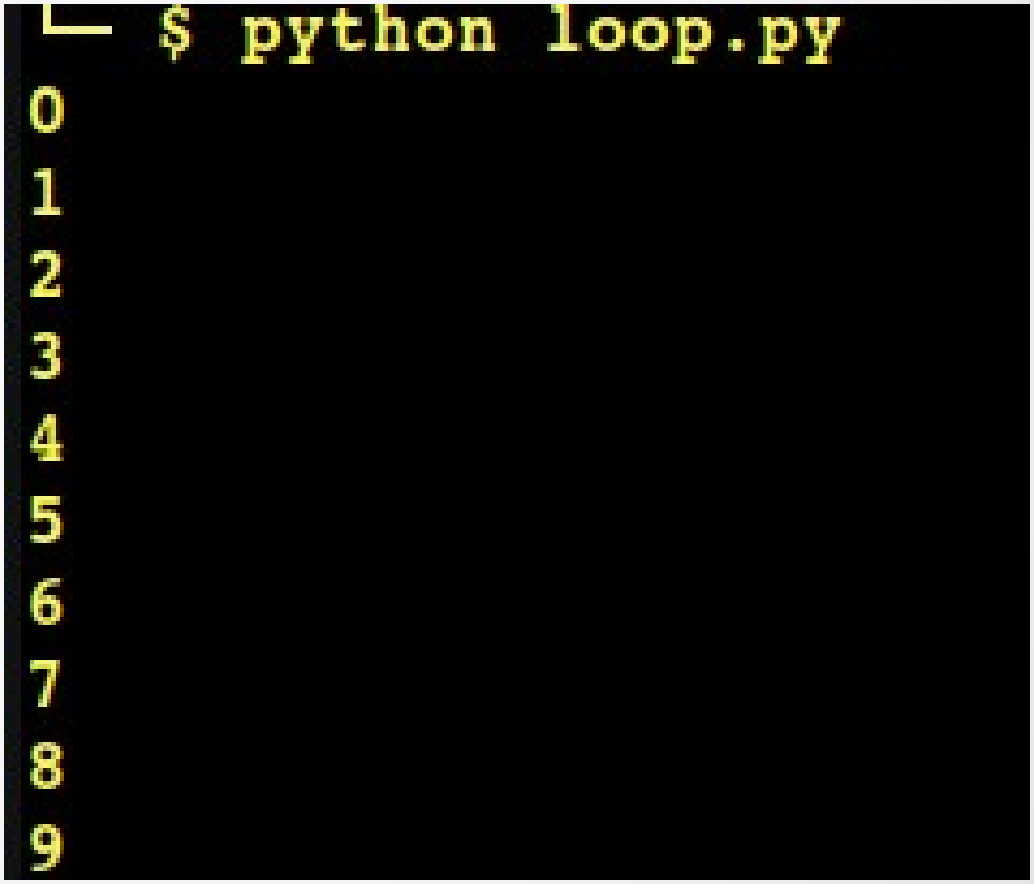
Example

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



Example

```
for i in range(10):
    print(i)
```

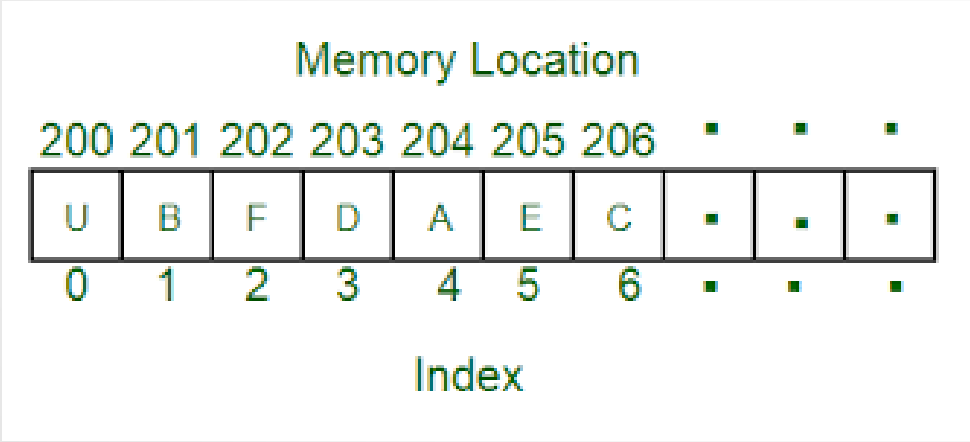


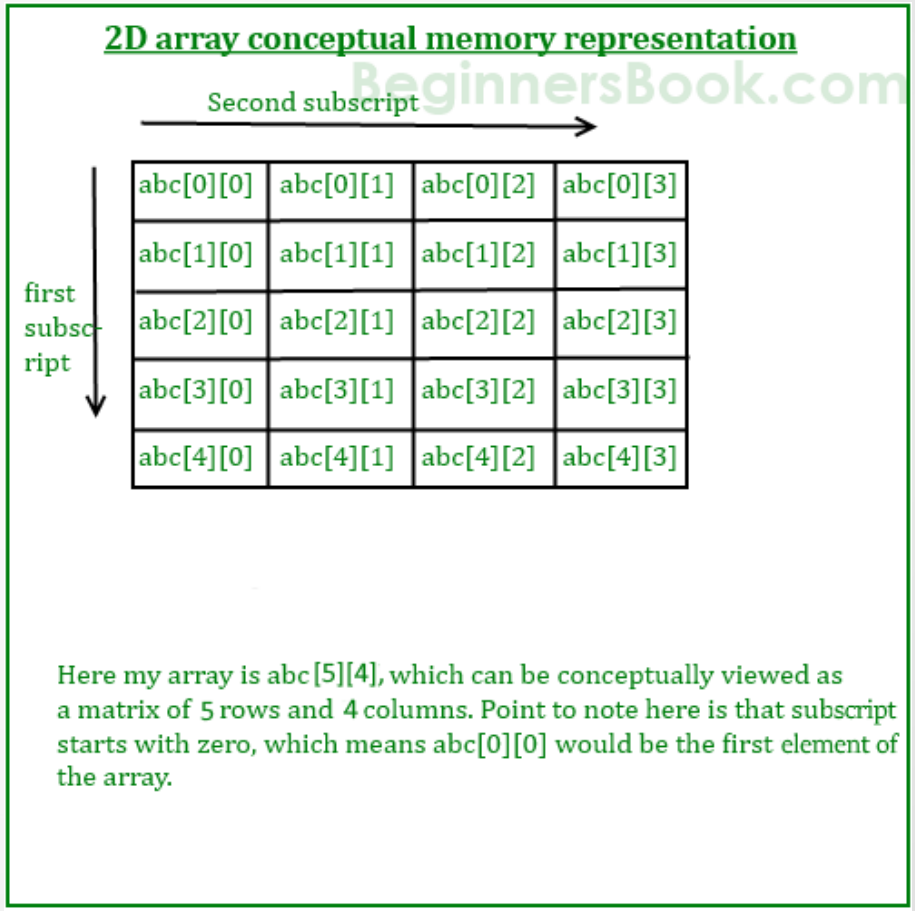
ARRAY

Array

- Come posso gestire facilmente le informazioni strutturate per natura? Ad esempio un numero complesso, o un vettore o una matrice?
- Le variabili strutturate tipiche sono gli array
- Ad esempio, un vettore di numeri in virgola mobile in C può essere dichiarato come: float v [10];
- Possiamo quindi accedere all'elemento i-esimo del vettore: v [i-1] = 3.5;
- Allo stesso modo posso definire una matrice (array bidimensionale) come: float m [10] [10];

Array

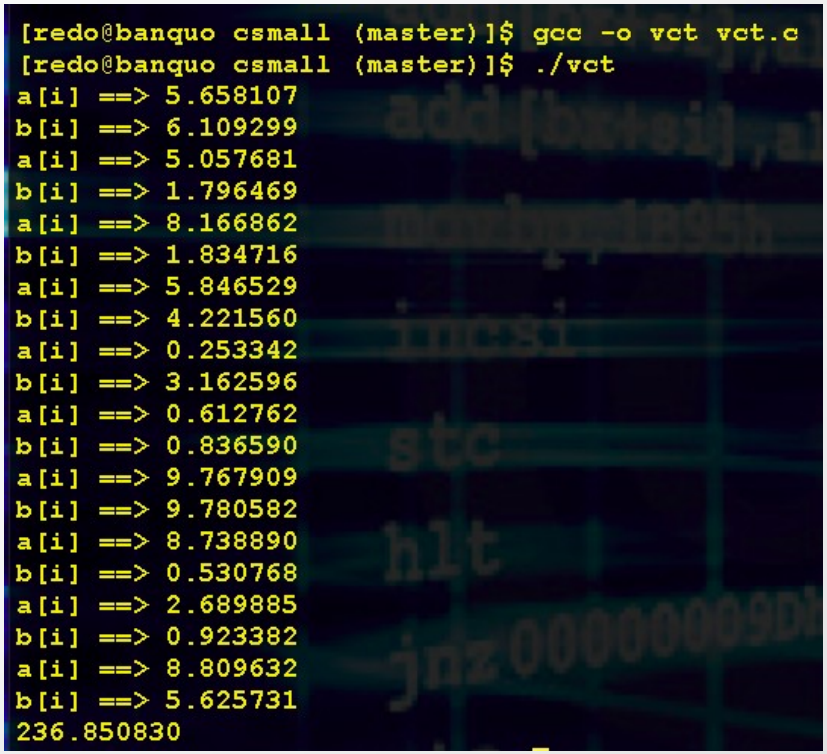

Memory Location
200 201 202 203 204 205 206
U B F D A E C
0 1 2 3 4 5 6
Index


2D array conceptual memory representation
BeginnersBook.com
Second subscript
first subscript
abc[0][0] abc[0][1] abc[0][2] abc[0][3]
abc[1][0] abc[1][1] abc[1][2] abc[1][3]
abc[2][0] abc[2][1] abc[2][2] abc[2][3]
abc[3][0] abc[3][1] abc[3][2] abc[3][3]
abc[4][0] abc[4][1] abc[4][2] abc[4][3]
Here my array is abc[5][4], which can be conceptually viewed as a matrix of 5 rows and 4 columns. Point to note here is that subscript starts with zero, which means abc[0][0] would be the first element of the array.

Example

- Loop di esempio per eseguire una moltiplicazione scalare tra i 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];
}
```



Example

```
import random

N = 20

a = []
b = []

for i in range(N):
    a.append(random.randrange(1, 30))
    b.append(random.randrange(1, 30))

for i in range(N):
    print("%3d %3d"%(a[i], b[i]))

s = 0.0
for i in range(10):
    s = s + float(a[i]*b[i])

print("")
print("S = %.5e"%s)
```