UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Fondamenti di Informatica (117IN)

A.A. 2022 / 2023

Lezione 4 - I Dati

Sylvio Barbon Junior
sylvio.barbonjunior@units.it

Sommario:

1) Tipi
2) Definizioni di variabili
3) Espressioni
4) Operatore condizionale
5) Incremento e decremento
6) Altri operatori
7) Conversione di tipo
8) Classe Math

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

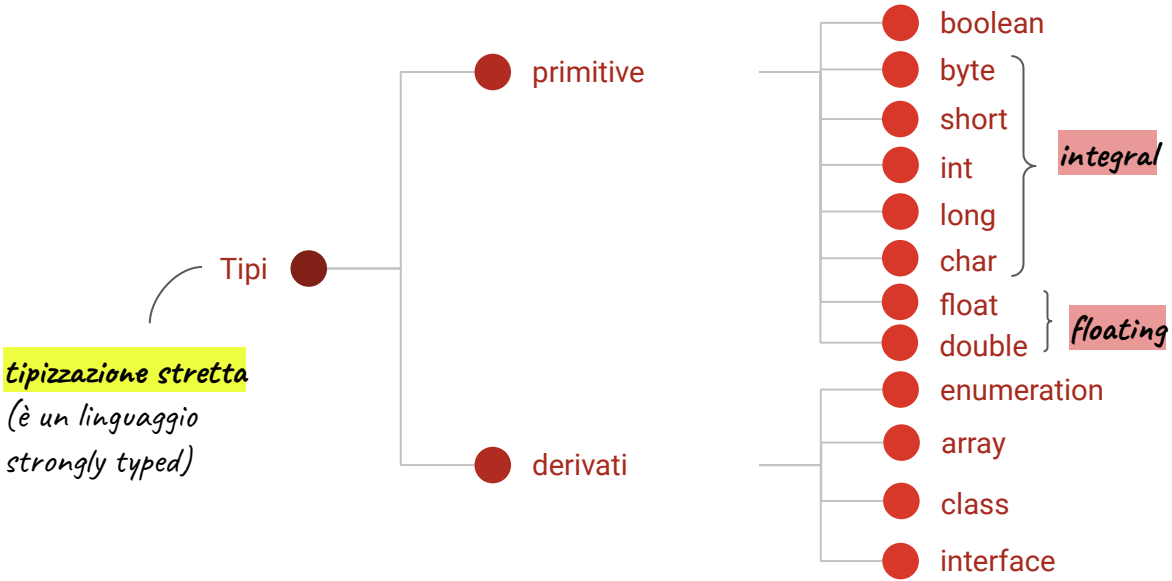
1) Tipi

- L'algoritmo non è soltanto un **insieme di istruzioni**;
- Una componente fondamentale dell'algoritmo è costituita dai **dati**;
- Una fase **essenziale** della stesura dell'algoritmo è la **scelta dei dati** da utilizzare;

La comunicazione tra l'utente e programma

1
dati di input (l'utente deve fornire al programma per l'esecuzione.)

2
dati di output, (il risultato dell'elaborazione, cioè i dati che il programma restituisce all'utente.)



Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Tipi

Tipo Primitivo	Descrizione
boolean	valori che possono essere true o false
char	caratteri sono di 16 bit (unicode)
byte	interi di 8 bit con segno
short	interi di 16 bit con segno
int	interi di 32 bit con segno
long	interi di 64 bit con segno
float	reali di 32 bit in virgola mobile
double	reali di 64 bit

*Una variabile di tipo primitivo può essere utilizzata **direttamente** (**senza new**)*

concatenazione String + int

Dati.java

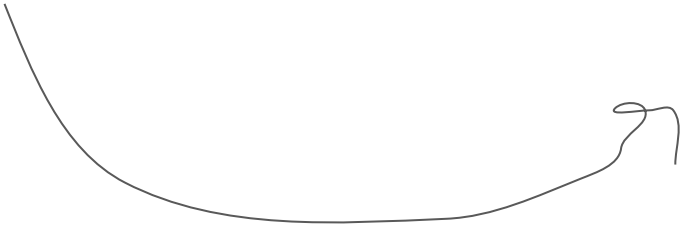
```
1  public class Dati{
2      public static void main(String args[]){
3          int x;
4          x = 5;
5          System.out.println("X = "+x);
6      }
7  }
```

```
sylviobarbon@ws-piccola:~/Scaricati/Fi/CodiciSorgenti$ java Dati
X = 5
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Tipi

- Il tipo boolean (**booleano**) ha come insieme di valori le due costanti simboliche false e true;
- Con elementi di questo tipo si possono fare operazioni logiche, che sono:
  a. | (barra verticale o “pipe”) OR logico (disgiunzione)
  b. ^ (esponente) OR-ESCLUSIVO logico o XOR
  c. & (E commerciale) AND logico (congiunzione)
  d. ! (Punto esclamativo) NOT logico (negazione)
  e. == uguale
  f. != diverso



```
sylviobarbon@ws-piccola:~/Scaricati/Fi/
CodiciSorgenti$ java EsempioBoolean
t = true
f = false
t|f = true
t^f = true
t&f = false
!f = true
!t = false
t==f = false
t==t = true
f==f = true
t!=f = true
t!=t = false
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Tipi

- Un tipo **intero** ha per valori tutti i numeri interi compresi fra -2^{N-1} e $+2^{N-1}-1$, dove N è il numero di bit usati per la sua rappresentazione.
- Con elementi di questo tipo si possono fare operazioni elementari, che sono:
  a. \+ somma
  b. \- sottrazione
  c. * moltiplicazione
  d. / quoziente della divisione
  e. % resto della divisione
  f. ++ incremento
  g. – decremento

che producono un risultato di tipo booleano

Operatori di confronto:

- **==** **uguale**
- **!=** **diverso**
- **>** **maggiore**
- **>=** **maggiore o uguale**
- **<** **minore**
- **<=** **minore o uguale**

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Tipi

- Con gli **interi** possono essere effettuate operazioni bit a bit, che richiedono peraltro delle conoscenze sulle rappresentazione dei numeri interi;

Operazioni bit a bit

~	complemento (unario) bit a bit
&	AND bit a bit
|	OR bit a bit
^	OR ESCLUSIVO bit a bit
<<	traslazione a sinistra della distanza specificata inserendo bit uguali a 0
>>	traslazione a destra della distanza specificata replicando il segno (bit più significativo)
>>>	traslazione a destra della distanza specificata inserendo bit uguali a 0.

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Tipi

BitBit.java

```java
public class BitBit{
    public static void main(String args[]){
        int i1 = 1;   //0001
        int i10 = 10; //1010
        System.out.println("i1 ("+Integer.toString(i1, 2)+") = "+i1);
        System.out.println("~i1 ("+Integer.toString(~i1, 2)+") = "+(~i1));
        System.out.println("i10 ("+Integer.toString(i10, 2)+") = "+i10);
        System.out.println("~i10 ("+Integer.toString(~i10, 2)+") = "+(~i10));
        System.out.println("i10<<i1 ("+Integer.toString(i10<<i1, 2)+")= "+(i10<<i1));
        System.out.println("i1<<i10 ("+Integer.toString(i1<<i10, 2)+")= "+(i1<<i10));
        System.out.println("i10>>i1 ("+Integer.toString(i10>>i1, 2)+")= "+(i10>>i1));
        System.out.println("i1>>>i1 ("+Integer.toString(i1>>>i1, 2)+")= "+(i1>>>i1));
        System.out.println("i10^i10 ("+Integer.toString(i10^i10, 2)+")= "+(i10^i10));
        System.out.println("i1|i10 ("+Integer.toString(i1|i10, 2)+")= "+(i1|i10));
    }
}
```

```
i1 (1) = 1
~i1 (-10) = -2
i10 (1010) = 10
~i10 (-1011) = -11
i10<<i1 (10100)= 20
i1<<i10 (10000000000)= 1024
i10>>i1 (101)= 5
i1>>>i1 (0)= 0
i10^i10 (0)= 0
i1|i10 (1011)= 11
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Tipi

- Con gli **“floating”** possono essere rappresentati i tipi reali:

1. +, - più unario e meno unario
2. ++, -- incremento e decremento
3. +, - somma e sottrazione
4. *, / moltiplicazione e divisione reale
5. % resto **r** della divisione reale fra n e d: **r** = n – (d*q)

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

2) Definizioni di variabili

Le variabili di un tipo primitivo si definiscono come i seguenti esempi:

```
int n;

double dd = 10.0;

boolean b, bb = true;

char cc = '\t';
```

I dati che descrivono un problema si possono suddividere in costanti e variabili.

I dati variabili possono cambiare valore.

*I dati **costanti** sono dati che hanno un valore predefinito, che non cambia.. Una costante è una variabile **final** con valore iniziale specificato*

```
final int n = 10;
final int m = n*3;
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

3) Espressioni

- Un’espressione è composta da operandi e operatori, e viene calcolata eseguendo in sequenza una serie di operazioni.
- Per ogni operatore, vengono prima valutati gli operandi e quindi applicato l’operatore stesso.
- Alcuni operatori richiedono esplicitamente che un operando sia una variabile.
- L’utilizzo dei delimitatori “parentesi tonde” individua sottoespressioni che vengono considerate operandi e che quindi vengono calcolate per prime.

Esempio 1 (x è una variabile reale)

```
x / 3 * 2
```

Gli operatori ‘/’ e ‘*’ hanno la stessa precedenza, per cui l'ordine di esecuzione viene determinato dall’associatività: gli operatori aritmetici binari sono associativi a sinistra, e l'espressione viene calcolata come se fosse scritta:

```
(x / 3) * 2
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

4) Operatore condizionale

L'operatore condizionale ha la forma:

`op1 ? op2 : op3`

Esso richiede che il primo operando op1 sia di tipo booleano: se questo vale **true** il risultato è il valore dell'operando op2, altrimenti il valore dell'operando op3.

OperCondizionale.java

```
1  public class OperCondizionale{
2      public static void main(String args[]){
3          int a = 1;
4          int b = 10;
5          String c = a>b ? "a maggiore di b" : "b maggiore di a";
6          System.out.println(c);
7      }
8  }
```

```
sylviobarbon@ws-piccola:
b maggiore di a
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

5) Incremento e decremento

- Gli operatori (unari) '++' e '--' richiedono come operando una variabile:
  - il valore aggiornato della variabile se prefissi, o
  - il valore originario della variabile se postfissi

IncrementoDecremento.java

```
1  public class IncrementoDecremento{
2      public static void main (String args[]){
3          int a = 5;
4          System.out.println("a = "+a);
5          System.out.println("--a = "+ (--a));
6          System.out.println("a = "+a);
7          System.out.println("a-- = "+ (a--));
8          System.out.println("a = "+a);
9          System.out.println("++a = "+ (++a));
10         System.out.println("a = "+a);
11         System.out.println("a++ = "+ (a++));
12         System.out.println("a = "+a);
13     }
14 }
```

```
a = 5
--a = 4
a = 4
a-- = 4
a = 3
++a = 4
a = 4
a++ = 4
a = 5
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

6) Altri operatori

Operatore	Descrizione
[ ]	Definisce un array, e.g., char[]
.	Riferisce a un membro di un oggetto.
(*tipo*)	converte (cast) op al tipo *tipo*.
new	crea un nuovo oggetto
instanceof	restituisce true se op1 è una istanza di op2, e.g., op1 *instanceof* op2.

Cast.java

```
1  public class Cast{
2      public static void main(String args[]){
3          char a = 66;
4          System.out.println(a);
5          System.out.println((int)a);
6          System.out.println((char)65);
7          System.out.println((float)65);
8      }
9  }
```

```
B
66
A
65.0
```

7) Conversione di tipo

Una conversione di tipo consente:

- nel caso di un operatore aritmetico binario, di trasformare il tipo di un operando nel tipo dell'altro operando;
- nel caso dell'operatore di assegnamento, di trasformare il tipo del risultato di una espressione nel tipo della variabile a cui il nuovo valore deve essere assegnato.
- nell'ambito dei tipi "integral" e "floating", rispettivamente, verso il tipo che usa un maggior numero di bit per la rappresentazione, con nessuna perdita di informazione;

Conversion.java

```
1  public class Conversion{
2      public static void main(String args[]){
3          byte a = 1;
4          int b = 2;
5          char c = 'C';
6          double d = 32.1;
7          double e = b;
8
9          System.out.println(a);
10         System.out.println(a+b);
11         System.out.println(c);
12         System.out.println(a+c);
13         System.out.println(b+d);
14         System.out.println(e);
15     }
16 }
```

```
1
3
C
68
34.1
2.0
```

8) Classe Math

La classe Math, che fa parte del package java.lang, contiene come membri statici alcune costanti e diverse funzioni matematiche di uso comune;

UsoMath.java

```
1  public class UsoMath{
2      public static void main(String args[]){
3          double pi = Math.PI;
4          double e =  Math.E;
5          int a = Math.abs(-32);
6          double b = Math.sqrt(81);
7          double d = Math.pow(3, 3);
8          double sin_1 = Math.sin(Math.PI/2);
9          double asin_2 = Math.asin(1);
10         double g = Math.exp(2);
11         double log = Math.log(1000);
12         double causale= Math.random();
13         double f = Math.round(32.1);
14         System.out.println(pi); //3.141592653589793
15         System.out.println(e);  //2.718281828459045
16         System.out.println(a);  //32
17         System.out.println(b);  //9.0
18         System.out.println(d);  //27.0
19         System.out.println(sin_1);  //1.0
20         System.out.println(asin_2); //1.5707963267948966
21         System.out.println(g);      //7.38905609893065
22         System.out.println(log);    //6.907755278982137
23         System.out.println(causale);//0.13963740950111236
24         System.out.println(f);      //32.0
25     }
26 }
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Grazie!

Sylvio Barbon Junior - sylvio.barbonjunior@units.it