Classi numeriche e operatori

Classi numeriche

Esempi di classi numeriche:

- int, numeri interi che occupano una parola in memoria
- long, numeri interi che usano due parole (interi lunghi)
- float, numeri reali che occupano una parola (numeri a virgola mobile)
- double, numeri reali che usano due parole (doppia precisione)

Gli oggetti numerici sono gli oggetti appartenenti a classi numeriche

Espressioni aritmetiche

- Le espressioni aritmetiche sono sequenze di nomi di oggetti numerici, operatori e parentesi (tonde)

- Esempi:
  - x+y
  - -x
  - x%2 // % è l'operatore modulo (resto // della divisione tra interi)
  - x*(y-z)/(x+y)

Precedenza degli operatori aritmetici

- Gli operatori *, / e % hanno precedenza su + e –
- Le espressioni vengono valutare da sinistra a destra
- Esempi:
  - 5-7*2+1 equivale a 5-(7*2)+1 e vale –8
  - 4.1/2-2 equivale a (4.1/2)-2 e vale 0.05
  - 9%5+1 equivale a (9%5)+1 e vale 5

Assegnazione

- Un'assegnazione altera il valore (stato) di un oggetto:

  identificatore_oggetto = espressione;

  Il valore dell'espressione (*rValue: right value*) viene assegnato all'oggetto (*lValue: left value*)

  lab1 = 20.0 ;

- Le assegnazioni posso essere anche concatenate

  lab1 = lab2 = lab3 = 20.0;

  (sconsigliato! Il programma risulta meno leggibile)

Esempi

- Incremento del valore di un oggetto. Le seguenti espressioni forniscono tutte lo stesso nuovo valore dell'oggetto **totale**.
  - **totale = totale + 1;**
  - totale += 1;
  - totale++;
  - ++totale;

- Decremento del valore di un oggetto. Le seguenti espressioni forniscono tutte lo stesso nuovo valore dell'oggetto **totale**.
  - **totale = totale – 1;**
  - totale –= 1;
  - totale--;
  - --totale;

Esempi

- Modifica del valore di un oggetto: anche in questo caso il valore finale di **costo** è lo stesso

    - costo = costo + 0.2 * costo;
    - costo += 0.2 * costo;
    - costo = costo * 1.2;
    - costo *= 1.2;

Conversioni di tipo

- Possiamo assegnare valori interi (int) a oggetti reali (float, double): viene usato il valore reale corrispondente (promozione)

- Quando invece assegniamo dei reali a degli interi perdiamo la parte frazionaria (troncamento)

```
int j = 1.234;
cout << j;        //    stampa  1 !

int k = -1.99;
cout <<k;         //    stampa –1 !
```

Uso di / e di %

```
// esempio di conversione da secondi a ore, minuti, secondi
int secondi= 3726;                  // tempo totale in secondi
int tOre = secondi / 3600;          // tOre vale 1
// NB per effettuare una divisione il calcolatore promuove
// entrambi gli operandi a numeri reali, quindi  secondi/3600
// è uguale a 3726./3600.
int tMinuti = secondi % 3600 / 60;
// il calcolatore valuta secondi % 3600 che vale 126, poi
// calcola 126/60, che vale 2.1 e lo converte nell'intero 2
// quando lo assegna a tMinuti
int tSecondi = secondi % 3600 % 60;
// il calcolatore valuta secondi % 3600 che vale ancora 126,
// poi 126 % 60 che vale 6
```

Arrotondamenti

- Il troncamento può essere sfruttato per effettuare l'arrotondamento di un numero con la precisione voluta.
- Consideriamo un numero reale r che vogliamo arrotondare con precisione p. Procederemo nel modo seguente:

```
float r,p,ar;
int temp;
temp=r/p + 0.5;
// il troncamento è per difetto, sommando 0.5 si arrotonda all'intero più vicino
ar=temp*p;
```

- esempio: r=10.52 , p=0.1 , 10.52/0.1 + 0.5 vale 105.2+0.5 ovvero 105.7, quindi temp vale 105 e ar vale 10.5
- esempio: r=10.56 , p =0.1, in questo caso otteniamo 105.7+0.5 ovvero 106.2, temp vale 106 e ar vale 10.6

Oggetti costanti

- Si usa **const** nella dichiarazione di oggetti che non debbano essere modificati accidentalmente, ad esempio costanti utili come π .

- Esempi:

```
const float iva=0.2;
const double Pi=3.1415926;
const tasso=0.04;
// Ma la classe di default è int !!!
// E quindi in questo caso tasso viene posto a zero
```

Espressioni booleane

- Prima di proseguire rivedere nel capitolo 4 la definizione di variabili logiche e le operazioni che possono essere effettuate su di esse.

- Un'espressione booleana può assumere solo due valori: vero o falso (true o false).

- In C++ ogni espressione numerica può essere utilizzata come espressione booleana in quanto il C++ interpreta zero come **false** e ogni altro valore numerico come **true**

Operatori booleani

<	minore di
<=	minore o uguale di
>	maggiore di
>=	maggiore o uguale di
==	uguale a (da non confondere con =)
!=	diverso da

Hanno precedenza su

&&	and logico
||	or logico
!	Not logico

Esempi

- 4 < 5 vale 1 (true)
- 2 >= 3 vale 0 (false)
- 4 == 4 vale 1 (true)

double x = 1.5, y = -1.8;

- x > y vale 1 (true)
- x > y + 5 vale 0 (false)
- (4<5)&&7<6 vale 0 (false)

Esercizi

- Valutare le seguenti espressioni dati
  int j=7, k=3, m=12;

  - j - k / m
  - m * j % k
  - 2 * j - k
  - (j - k) * 2

Esercizi

valutare le seguenti espressioni booleane

!(4<5)

3 <= 4 && 5 < 7

2 < 1 || 6 < 8

!(5 ==5) && 3<7

!(x = -56) con x=56

Soluzioni

int j=7, k=3, m=12;

- j - k / m vale 7
- m * j % k vale 0
- 2 * j - k vale 11
- (j - k) * 2 vale 8

Soluzioni

- !(4<5) vale 0
- 3 <= 4 && 5 < 7 vale 1
- 2 < 1 || 6 < 8 vale 1
- !(5 ==5) && 3<7 vale 0
- !(x = -56) con x=56 vale 0 !