COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

WEB DEVELOPER
Fondamenti di Programmazione
Massimo PAPA

I TIPI AGGREGATI

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

I TIPI AGGREGATI

- Sino a ora abbiamo trattato solo variabili di tipo fondamentale (a parte l'oggetto string)
- I tipi fondamentali si suddividono in
  - **Tipi integrali**
  - **Tipi a virgola mobile**
- Questi rappresentano un unico valore
- Ci si riferisce a loro anche come **tipi scalari**

I TIPI AGGREGATI

- Nell'analisi di un problema e nella successiva ricerca di una soluzione, l'uso dei soli tipi scalari contrasta con la naturale tendenza dell'uomo a ragionare in termini di categorie e insiemi.
- L'utilizzo dei dati aggregati semplifica notevolmente la modellizzazione di un problema.
- Inoltre consente una codifica più efficace e ottimizzata

I TIPI AGGREGATI

- In c++ abbiamo la possibilità di aggregare tipi scalari per rappresentare dati più complessi.
- Abbiamo questi tipi:
  - Enumerazione
  - Vettore (array)
  - Struttura
  - Unione
  - Classe

ENUMERAZIONI

- Per definire una costante si utilizza la parola chiave **`const`**
- Si usa la seguenti sintassi:

```
const tipo nome = valore ;
```

- Esempi:

```
const int aliquota = 27 ;
const char risposta = 's' ;
const string sigla = "TO";
```

ENUMERAZIONI

- Le enumerazioni sono invece una serie di valori interi che vengono associati a nomi di costanti
- Si usa la seguenti sintassi:

```
enum identificatore {lista costanti};
```

- identificatore deve essere *parlante*
- Esempi:

```
enum lista {E1,E2,E3,E4};
enum MESI {AGO=8,SET=9,OTT=10};
```

ENUMERAZIONI

- Se non viene specificato, il primo valore dell'enumerazione è posto uguale a 0
- L'enumerazione in un certo qual modo consente di definire un nuovo tipo di dato
- L'enumerazione è quindi un primo esempio di *programmer-defined type*
- Implementiamo un programma di esempio che utilizzi **`enum`**

Esempio ENUMERAZIONI

Calcolare il giudizio finale di un test sulla base del punteggio conseguito

Una prova scritta è formata da 20 quesiti. Le risposte ai diversi quesiti sono valutate in decimali: 0 per una risposta errata, 0,5 per una risposta parzialmente corretta e 1 per una risposta esatta.

La valutazione complessiva non riporta il punteggio, ma raggruppa i punti totalizzati nella prova secondo la tabella:

da	a	risultato
0	6	negativo
7	13	incerto
14	20	positivo

Il programma potrebbe utilizzare:

```
enum valutazione{POSITIVO,INCERTO,NEGATIVO};
```

Per presentare il risultato utilizza una funzione come la seguente:

```
void presenta(valutazione val)
```

Esempio ENUMERAZIONI

Un altro esempio interessante è che produce un insieme iterabile.

Questo è vero in quanto i nomi costanti vengono *mappati* su tipi interi.

Consideriamo il seguente codice:

```
...
enum Giorni {LUN=1, MAR, MER, GIO, VEN, SAB, DOM};
int i;
for(i=LUN; i<=SAB;i++)
    cout << i;
...
```

In output si ottiene la stampa da 1 a 7

Note sulle ENUMERAZIONI (1/3)

ATTENZIONE: se si dichiara una variabile o un nuovo enumeratore con lo stesso nome di un enumeratore già dichiarato, da quel punto in poi si perde la visibilità del precedente enumeratore.

Esempio:

```
enum Giorni {lu, ma, me, gi, ve, sa, do} ;

enum PrimiGiorni {do, lu, ma, gi} ;

// da qui in poi non si vedono più gli enumeratori

// lu, ma, gi e do del tipo Giorni
```

Note sulle ENUMERAZIONI (2/3)

ATTENZIONE: il tipo enumerato è totalmente ordinato.
Su un dato di tipo enumerato sono applicabili tutti gli operatori relazionali.

Esempio:

```
lu < ma → vero

lu >= sa → falso

rosso < giallo → vero
```

Note sulle ENUMERAZIONI (3/3)

ATTENZIONE: si possono inizializzare a piacimento le costanti

Esempio:

```
enum Mesi {gen=1, feb, mar, ... } ;

// Implica: gen = 1, feb = 2, mar = 3, ...

enum romani { i=1, v = 5, x = 10, c = 100 } ;
```