

# CLASSE DATA

Versione 1.0.1 – 12 febbraio 2014

La classe genera oggetti costituiti da un unico valore di tipo `time_t`, come definito nella libreria standard `time.h`. Il valore può essere comodamente impostato sfruttando l'*overloading* dei metodi di inizializzazione e di modifica della classe. Può inoltre essere “letto” in diversi formati per adattarlo al contesto del programma.

## FUNZIONI CREATRICI

---

```
DATA::DATA ( ) ;
```

Crea un oggetto data riferito al giorno corrente (secondo l'orologio di sistema).

```
DATA::DATA( char g, char m, short a ) ;
```

Crea un oggetto data riferito al [g]giorno, [m]ese e [a]nno passati.

```
DATA::DATA( const char *s ) ;
```

Crea un oggetto data secondo i dati rilevati analizzando la stringa `s`. La stringa può contenere (nell'ordine) giorno, mese e anno, oppure solo giorno e mese. L'anno può essere espresso con due sole cifre, nel qual caso si sottintende il XXI secolo. L'anno può essere omesso, nel qual caso viene usato l'anno dell'orologio di sistema.

## FUNZIONI DI IMPOSTAZIONE DELL'OGGETTO

---

```
void DATA::set_data( void ) ;
```

Imposta la data sul giorno corrente (secondo l'orologio di sistema).

```
void DATA::set_data( char g, char m, short a ) ;
```

Imposta la data secondo il [g]giorno, [m]ese e [a]nno passati.

```
void DATA::set_data( const char *s ) ;
```

Imposta la data secondo i dati rilevati analizzando la stringa `s`. La stringa può contenere (nell'ordine) giorno, mese e anno, oppure solo giorno e mese. L'anno può essere espresso con due sole cifre, nel qual caso si sottintende il XXI secolo. L'anno può essere omesso, nel qual caso viene usato l'anno dell'orologio di sistema.

## FUNZIONI DI LETTURA DELLE PROPRIETÀ DEGLI OGGETTI

---

```
time_t DATA::get_time_t( void ) ;
```

Legge la data espressa nel formato della libreria standard del C.

```
void DATA::get_data( char *g, char *m, short *a ) ;
```

Legge la data e ne immette [g]giorno, [m]ese e [a]nno negli appositi parametri passati per indirizzo.

```
const char *DATA::get_data_c( int formato ) ;
```

Legge la data, la formatta in una stringa coerentemente col parametro `formato` e restituisce un puntatore alla stringa. Si tenga presente che la stringa è fisicamente “alloggiata” in un'unica variabile statica condivisa da tutti gli oggetti di classe `DATA`. Il parametro `formato` può essere:

- 11** giorno e mese, ad una o due cifre (a seconda del numero)
- 22** giorno e mese, a due cifre
- 112** giorno, mese e anno (giorno e mese ad una o due cifre, anno a due cifre)
- 222** giorno, mese e anno a due cifre
- xxx** (**xxx** sta per “ogni altro valore”) giorno e mese a due cifre, anno a quattro cifre

```
int DATA::get_giorno_della_settimana( int *giorno );
```

Restituisce un valore numerico che rappresenta il giorno della settimana ( **0** = domenica; **1** = lunedì; **2** = martedì; ...; **6** = sabato; ).

```
const char *DATA::get_giorno_della_settimana_c( int nCar );
```

Restituisce una stringa C contenente il nome del giorno della settimana. Volendo, si può passare un parametro **nCar** che indichi un numero massimo di caratteri da usare nella stringa (tipiche sono le abbreviazioni a tre caratteri come “lun”, “mar”, ecc.).

```
int DATA::rispetto_a( DATA d );
```

Confronta un oggetto data con un altro passato come parametro in **d**. Se l’oggetto precede **d**, restituisce **-1**; se l’oggetto segue **d**, restituisce **1**; se i due oggetti sono equivalenti, restituisce **0**. Ad esempio:

```
DATA d1( 15,8,2013 );
DATA d2( 26,9,2013 );
int esito;

esito = d1.rispetto_a( d2 ); // esito e' -1
esito = d2.rispetto_a( d1 ); // esito e' 1
d1.set_data( 15,8,2013 );
esito = d1.rispetto_a( d2 ); // esito e' 0
```

```
void DATA::estrai_gma( const char *s, int *g, int *m, int *a );
```

È un metodo “esplorativo” da usare qualora si volesse verificare la correttezza di una stringa contenente una data. Analizza la stringa **s** e immette [g]iorno, [m]ese e [a]nno nei parametri passati per indirizzo. L’oggetto che applica il metodo **non** viene modificato.