Variabili

Una variabile è uno spazio di memoria in cui archiviare un dato.
In c/C++ una variabile ha un nome, valore ed un tipo.
Ad esempio, questa affermazione (chiamata dichiarazione):

```
int pin = 13;
```

crea nella memoria SRAM una variabile il cui nome è : **pin** il cui valore è **13** e il cui tipo è **int**
Se nel programma si fa riferimento alla variabile in questo modo:

```
pinMode(pin, OUTPUT);
```

è il valore della variabile pin (13) che verrà passato alla funzione pinMode().
Si sarebbe potuto anche scrivere:

```
pinMode(13, OUTPUT);
```

Il vantaggio di una variabile in questo caso è che si deve specificare il numero effettivo del pin solo una volta, ma puoi usarlo molte volte evitando il rischio di insidiosi errori di esecuzione dovuti ad errori di editing.
Ad esempio:

```
digitalWrite(pin, HIGH);
```

invece che:

```
digitalWrite(13, HIGH);
```

Se in un secondo tempo si vuole passare dal pin 13 al pin 12 basta cambiare la dichiarazione della variabile senza modificare l'intero programma.
Quindi, se in seguito decidi di passare dal pin 13 al pin 12, devi solo cambiare un punto nel codice.
L'impiego principale delle variabili però è l'allocazione temporanea di dati letti dall'ingresso o elaborati dal programma.
Ad esempio:

```
int value=0;
...
value =digitalRead(pin);
```

La variabile value viene dichiarata ed inizializzata a 0 ma nel corso del programma il valore viene sostituito dal valore letto da un pin di ingresso (che può essere HIGH=1 oppure LOW=0)
Il valore di value è quindi cambiato nel tempo di esecuzione e la variabile tiene traccia di tale cambiamento.

Ambiente (Scope) di una variabile

Per una variabile è anche importante determinare l'ambiente in cui è definita.
Nel linguaggio C/C++ (a differenza da Java) esiste un "ambiente globale" in cui sono allocate le variabile dichiarate al di fuori di qualsiasi funzione, classe o metodo. Le variabili dichiarate in questo modo sono visibile da ogni punto del programma.
In alternativa le variabili possono essere dichiarate all'interno di funzioni o metodi e in tale caso vengono create all'esecuzione della funzione o metodo e distrutte al termine della funzione o metodo con visibilità solo interna alla funzione o al metodo, oppure possono essere dichiarate come attributi di classe ed in tale caso la loro visibilità dipende dalla visibilità assegnata agli attributi della classe ed alla visibilità degli oggetti istanziati di quella classe.
E' bene non eccedere nell'uso di variabili globali perchè le variabili globali, essendo accessibili da ogni punto del programma sono soggette frequenti errori di programmazione dovuti ad effetti collaterali.
Tuttavia se la variabile ha il significato di variabile di stato è indispensabile che sia persistente per tutta la durata dell'esecuzione del programma (dall'accensione allo spegnimento).
Le possibili soluzioni sono l'utilizzo di variabili globali o meglio l'utilizzo di oggetti statici globali dichiarati a partire da classi che implementano l'information hiding.

Ultime modifiche: domenica, 6 febbraio 2022, 13:20

?

?