# Variabili globali? No, Grazie

*"L'unica sua dote era una specie di futile entusiasmo. In pratica sembrava che usasse l'archetto come una sega, straziando le corde al punto da condurre l'ascoltatore verso pericolosi stati di follia. Non aveva cognizione della natura dello strumento: provava a soffiarci dentro."* (**Mr. A.Torgman**) [maestro di violoncello di Virgil Starkwell, intervistato].

In **Prendi i soldi e scappa** il maestro di violoncello di **Virgil Starkwell** ci descriveva come si sentivano gli sfortunati ascoltatori di **Virgil** (**Woody Allen**) durante le sue esibizioni. Ecco, io ho esattamente le stesse sensazioni quando leggo del codice che usa uno dei capisaldi della **non-programmazione**: le **variabili globali**. Intendiamoci: le variabili globali (come anche il **goto**, per esempio) fanno parte del linguaggio, *quindi esistono*: a volte è possibile e/o necessario usarle. Però, esattamente come il **goto**, è quasi sempre possibile farne a meno, con grandi benefici di stile (leggibilità e manutenibilità, soprattutto), e funzionalità (meno *bugs*): chiamalo poco.



**…la variabile globale: lo strumento sbagliato nel posto**

**sbagliato…**
I punti critici sono moltissimi, ma, visto che non voglio scrivere né un poema, né un libro sull'argomento, ne ho isolati alcuni. Vediamo:

**1) le variabili globali non sono thread-safe**

nella programmazione **multithreading** l'uso delle globali può generare problemi di malfunzionamento subdoli e difficili da trovare. Il caso classico sono le istruzioni di lettura/scrittura di una globale fuori da una zona critica, ossia (per esempio) senza la protezione di un **mutex**: in un grande progetto è sufficiente una (una sola!) dimenticanza, di questo tipo per creare tanti di quei **mal di testa** che ti passerà la voglia di usare le globali per il resto della tua vita di programmatore. Provare per credere. A questo punto uno potrebbe obbiettare: "**ma io scrivo programmi normali, non multithreading**". Va bene: premettendo (e chiarendo nel prossimo paragrafo) che un programma "normale" può essere considerato un programma *multithread* con un solo *thread*, ringrazio per l'obiezione, che mi serve giusto di spunto per illustrare il **punto 2:**

**2) le variabili globali violano il principio di manutenzione/riutilizzazione del Software:**

quando si scrive del codice professionalmente bisogna sempre pensare al lavoro in team, e quindi alle operazioni di manutenzione che potrebbero essere svolte da altri (e, a volte) dopo molto tempo: evidentemente un codice, vasto e pieno di globali, è difficile da manutenere come un codice **No Comment** (ricordate ?), perché la storia di una globale è poco comprensibile, potrebbe essere toccata in molti posti diversi da molte funzioni diverse, e, se per capire un pezzetto di codice bisogna aprire alcune decine di file… avete vinto un altro bel mal di testa! E non ne parliamo di riutilizzare del codice pieno di globali per un altro progetto: se non l'avete mai fatto provate almeno a immaginarvi la difficoltà. E non solo: torniamo al **punto 1** (*thread-safe*): chi mi dice che del

codice “normale” non lo debba riutilizzare (un giorno) in un progetto *multithread* ? Se il codice è *thread-safe* si può fare agevolmente, ma se ci sono delle globali di mezzo… beh, buon lavoro (e buona fortuna).

**3) le variabili globali aumentano la difficoltà del debug e moltiplicano la probabilità di errori di programmazione**

per quanto riguarda il debug vi rimando al **punto 2**: se il valore di una variabile è difficile da seguire a livello di manutenzione, lo sarà anche a livello di *debug*. E ci saranno anche **più malfunzionamenti da debuggare** (fantastico!), perché, oltre a tutti i possibili errori di codificazione ci aggiungiamo anche quelli di **scope**: provate questo codice:

```
int my_var = 0; // globale! // funzione che incrementa la globale
void incrementaMyVar()
{ my_var += 5;
} // funzione main
int main(int argc, char **argv)
{ // faccio mille cose... // ... // incremento my_var incrementaMyVar(); // faccio altre mille cose... // ... // definisco una "nuova" variabile my var e la uso int my_var = 2; // ah, ah, ah: ridefinizione! // ... // faccio ancora mille cose... // ... // incremento e test my_var (oops! quale my_var?) incrementaMyVar(); if (my_var == 2) formatMyHardDisk(); // uh, uh, uh: era quella sbagliata! // ... return 0;
}
```

Quello mostrato sopra era un problema di **scope** con ridefinizione locale (accidentale) di una globale. Il codice che segue è ancora più semplice, mostra una svista su un dettaglio importante:

```
int my_var = 0; // globale! // funzuine lunghissima che fa un sacco di cose
void faccioMilleCose()
{ // faccio mille cose... // ... // incremento my_var (sepolto
```

```
tra mille istruzioni!) my_var += 5; // faccio altre mille cose... // ...
} // funzione main
int main(int argc, char **argv)
{ // faccio un po' di cose... // ... // chiamo faccioMilleCose() faccioMilleCose(); // oops! ho incrementato my_var senza accorgermene // faccio altre cose... // ... // test my_var if (my_var == 5) formatMyHardDisk(); // uh, uh, uh: ho sbagliato qualcosa? // ... return 0;
}
```

bella storia, eh ?

**4) le variabili globali violano il principio di incapsulamento delle variabili**

beh, questo non c'è nemmeno bisogno di spiegarlo, una globale è tutto meno che incapsulata… oops, ma questo è **OOP**, quindi esula un po' l'argomento del post… ah, no: questo articolo vale sia per **C** che per **C++** (*ed anche altri linguaggi, direi…*). Beh, visto che siamo in argomento **OOP**, e quindi **C++**, cito volentieri il grande **M.Cline** che nelle su **C++FAQ dice** (*traduco, eh*):

I nomi delle variabili globali dovrebbero iniziare con //. Ecco il modo ideale per dichiarare una variabile globale: `// int xyz; <-la cosa che rende ideale questa globale è l'iniziale //` Ecco il modo ideale per utilizzare una variabile globale: `void mycode() { ... // fai_qualcosa_con(xyz); <-idem come sopra ... }` Ok, questo è un gioco. Una specie. La verità è che ci sono casi in cui le variabili globali sono meno peggio delle alternative - quando le globali sono il minore dei mali. Ma loro sono sempre malvagie. Quindi lavatevi le mani dopo averle usate. Due volte. (Marshall Cline `C++FAQ sezione 27.15`)

Sante parole.

E, per restare in tema **C++**, aggiungo un piccolo appunto: *una **Duna**, anche se ci attacchi un logo **Ferrari** resta, ahimè, una **Duna*** (*e non me ne vogliano i lettori che usano la Duna, eh!*). Allo stesso modo se, per darti delle arie, chiami **Singleton** una **Variabile Globale Sofisticata** il risultato non cambia,

puoi sofisticarla quanto vuoi (*inizializzazione "lazy", inizializzazione "eager", inizializzazione "oggi-sono-indeciso-e-faccio-un-mix-di-lazy-e-eager", ecc., ecc.*) ma, sotto sotto, ***è sempre una schifezza di Variabile Globale*** (anche se un pelino più accettabile, devo ammetterlo).

(*…e qua bisogna giocare d'anticipo: so che la affermazione qui sopra avrà fatto saltare sulla sedia tutti gli adepti della religione di* ***Design Patterns****, si, quelli che la sera recitano le orazioni davanti alla foto della* ***Gang of Four*** *che tengono sul comodino. Ecco, ho il massimo rispetto* ***per loro****, per la* ***Gang of Four*** *e* ***per il libro*** *che ha, sicuramente, un grande valore tecnico (e chi sono io per negarlo?) ma non è da prendere tutto come oro colato… E vi assicuro che molta gente ha scritto del buon Software anche prima dell'uscita di quel libro (ma come avranno fatto?). Ecco, quel libro è l'equivalente informatico de* ***La corazzata Potëmkin*** *del Cinema, bello fin che vuoi (è un capolavoro) ma ti induce facilmente valutazioni come quella che fece* ***Fantozzi****. Anzi, posso affermare che, ultimamente, considero che il vero grande valore aggiunto che ci fornisce il libro* ***Design Patterns*** *è quello* ***dissuasivo****: supponiamo che avete (o avrete) un/a figlio/a che vuole dedicarsi all'Informatica, ma voi pensate che studiando Giurisprudenza o Economia possa avere un futuro più luminoso. Ecco, fategli trovare (casualmente) una copia di* ***Design Patterns*** *sul letto dicendo "…Ho ritrovato in cantina questo mio vecchio libro… dagli una occhiata, perché presto sarà il tuo pane quotidiano". È probabile che la mattina successiva, verso le 6 (meglio non aspettare) verrete svegliati da vostro figlio/a chiedendovi "Come hai detto che ci si iscrive a Giurisprudenza?"…)*

**ConclusionE**

Che possiamo dire in conclusione? Sicuramente che, se pensate che tutto quanto detto qui sopra sia solo farina del mio sacco, potete provate a interrogare il nostro amico **Google**, chiedendo, ad esempio: "**global variables are evil?**", e vedrete

che valanga di risultati vi verrà fuori. Se vi può interessare una delle pagine più ben fatte la trovate **qui.**

A questo punto penso che quanto detto sia sufficiente. Evidentemente io sono un po' prevenuto perché ho lavorato a lungo su software *multithreading* (*e ho ancora un po' di mal di testa…)*, però vi chiedo di fidarvi e di propagare il messaggio il più possibile. Si, lo so, dire a un programmatore inesperto (*o a un programmatore stanco*) "**non usare le variabili globali!**" è come dire a un bambino "**non mangiare troppe caramelle!**": beh, anche se a malincuore, bisogna farlo: ***la salute prima di tutto.***

Ciao e al prossimo post!

https://italiancoders.it/variabili-globali-no-grazie/