



## Οδηγός Συγγραφής Σχολίων σε C

Στηριγμένο στο «Google C++ Style Guide, Revision 3.188»

<http://code.google.com/p/google-styleguide/>

Η διαδικασία του να γράφουμε σχόλια σε κώδικα πρέπει να θεωρείτε τόσο σημαντική όσο και το να γράφουμε τον ίδιο το κώδικα. Όταν αναπτύσσετε το δικό σας κώδικα θα πρέπει να έχετε πάντα στο μυαλό ότι σε κάποια φάση θα χρειαστεί να τον ξαναδιαβάσετε και να θυμηθείτε πως λειτουργεί και με πιο σκεπτικό έχει γραφτεί. Αυτό μπορεί να γίνει πολύ σύντομα ή μετά από πολλά χρόνια. Επίσης μπορεί κάποιος άλλος να χρειαστεί να δει τον κώδικα σας και να τον κατανοήσει. Τέλος, τα σχόλια σε ένα κώδικα μπορούν να χρησιμοποιηθούν για τη συγγραφή του εγχειριδίου ενός προγράμματος.

Είναι ξεκάθαρο από τη πιο πάνω σύντομη εισαγωγή ότι θα πρέπει στις προγραμματιστικές εργασίες μας να δίνουμε την ίδια έμφαση στα σχόλια όση δίνουμε και στον υπόλοιπο προγραμματισμό. Τα σχόλια πρέπει να είναι σύντομα και να επεξηγούν τη λειτουργία του κώδικα. Πρέπει να απευθύνονται σε άτομα που γνωρίζουν πολύ καλά τη γλώσσα και τεχνικές προγραμματισμού. Δεν πρέπει με άλλα λόγια να επεξηγούν τη ίδια την γλώσσα προγραμματισμού. **Γράφετε σχόλια (για την λογική του προγράμματος σας) κάνοντας την υπόθεση ότι έχετε απέναντί σας κάποιον που γνωρίζει τη γλώσσα καλύτερα από εσάς.** Πιο κάτω σας δίνουμε τις κύριες γραμμές στις οποίες να βασίζετε τη συγγραφή των σχολίων σας και οι οποίες είναι βασισμένες στα πρότυπα που χρησιμοποιούνται στη Google για προγραμματισμό σε C και C++.

Παρόλη την δυσaréσκεια που μπορεί να προκαλέσει η συνεγγραφή σχολίων σε ένα προγραμματιστή δεν παύουν από το να είναι πολύ σημαντικά και κάνουν τον κώδικα πιο ευανάγνωστο. Οι πιο κάτω κανόνες περιγράφουν το πώς θα πρέπει να είναι τα σχόλια σε ένα πρόγραμμα και το πού θα πρέπει να τοποθετούνται. Αλλά να θυμάστε ότι παρόλο που τα σχόλια είναι σημαντικό κομμάτι του προγράμματος σας, ο τρόπος προγραμματισμού θα πρέπει και αυτός από μόνος του να είναι περιγραφικός. Παραδείγματος χάρη, το να δίνουμε ονόματα με νόημα στις μεταβλητές είναι πάντα καλύτερο από το να δίνουμε ονόματα χωρίς νόημα και να εξηγούμε τη χρήση τους με σχόλια. Ένα άλλο καλό παράδειγμα είναι η χρήση σταθερών αντί για αριθμητικές τιμές π.χ. `#define MAX 10000`

**Όταν γράφετε τα σχόλια να σκέπτεστε αυτόν που θα τα διαβάσει και θα προσπαθήσει να καταλάβει το κώδικα σας. Ίσως κάποιο μέρα αυτός να είστε εσείς!!!**

### Τύπος Σχολίων

Χρησιμοποιήστε είτε `//` ή `/* */` σύνταξη φτάνει να είστε συστηματικοί. Τα `//` είναι πιο διαδεδομένα.



### Σχόλια Αρχείου

Ξεκινήστε κάθε αρχείο πυγαίου κώδικα με σχόλια που να περιγράφουν το κώδικα του αρχείου καθώς και τα στοιχεία του συγγραφέα/προγραμματιστή.

```
/** @file console.h
 *  @brief Function prototypes for the console driver.
 *
 *  This contains the prototypes for the console
 *  driver and eventually any macros, constants,
 *  or global variables you will need.
 *
 *  @author Harry Q. Bovik (hqbovik)
 *  @author Fred Hacker (fhacker)
 *  @bug No known bugs.
 */
```

Όταν γράφετε κώδικα διασπασμένο σε πολλαπλά αρχεία ακολουθείτε το πιο κάτω:

Σε γενικές γραμμές το .h αρχεία περιγράφει τις μεθόδους/σταθερές που ορίζονται στο αρχείο με μια σύντομη περιγραφή του τι είναι και πώς χρησιμοποιούνται. Ένα αρχείο .c πρέπει να περιέχει περισσότερες πληροφορίες σχετικά με τις λεπτομέρειες υλοποίησης και των αλγορίθμων που χρησιμοποιούνται. Ποτέ δεν πρέπει να επαναλαμβάνεις τα σχόλια και στα δυο αρχεία.

### Σχόλια Συναρτήσεων

**Σχόλια** που τοποθετούνται στα **πρότυπα** των συναρτήσεων περιγράφουν τη **χρήση τους**. Σχόλια στο **ορισμό μιας συνάρτησης** περιγράφουν τη **λειτουργία**.

#### i) Δήλωση Συνάρτησης

Κάθε **δήλωση συνάρτησης** θα πρέπει να έχει τα σχόλια αμέσως πριν τη δήλωση τα οποία να περιγράφουν τη λειτουργία αλλά και πώς χρησιμοποιείται. Τα σχόλια περιγράφουν τη λειτουργία της συνάρτησης. Τι πρέπει να περιλαμβάνουν τα σχόλια στον ορισμό της συνάρτησης.

- Ποιες είναι οι είσοδοι και έξοδοι (inputs, outputs).
- Εάν η λειτουργία δεσμεύει μνήμη την οποία ο χρήστης πρέπει να αποδεσμεύσει.
- Εάν κάποιες από τις παραμέτρους μπορούν να NULL.
- Εάν υπάρχουν οποιεσδήποτε επιπτώσεις στην απόδοση.

Εδώ είναι ένα παράδειγμα:

```
/** @brief Prints the string s, starting at the current
 *      location of the cursor.
 *
 *  If the string is longer than the current line, the
 *  string should fill up the current line and then
 *  continue on the next line. If the string exceeds
 *  available space on the entire console, the screen
 *  should scroll up one line, and then the string should
 *  continue on the new line. If '\n', '\r', and '\b' are
```



```
* encountered within the string, they should be handled
* as per putbyte. If len is not a positive integer or s
* is null, the function has no effect.
*
* @param s The string to be printed.
* @param len The length of the string s.
* @return Void.
*/
void putbytes(const char* s, int len);
```

Ωστόσο, δεν υπάρχει λόγος να είναι κανείς λεπτομερής εκεί που κάποια πράγματα είναι αυτονόητα. Στο πιο κάτω παράδειγμα δεν χρειάζεται αν εξηγηθεί και τότε η συνάρτηση επιστρέφει false.

```
// Returns true if the table cannot hold any more entries.
bool IsTableFull();
```

## ii) Ορισμός συνάρτησης

Κάθε **ορισμός συνάρτησης** θα πρέπει να έχει σχόλια που να **περιγράφουν τη λειτουργία της**, ειδικά εάν υπάρχει κάτι δύσκολο για το πώς κάνει τη δουλειά της. Για παράδειγμα, μπορεί να περιγράψετε κάποιες τεχνικές που χρησιμοποιείτε, ώστε να δοθεί μια γενική εικόνα των βημάτων που χρησιμοποιήσατε, ή να εξηγήσει γιατί επιλέξατε να εφαρμόσετε τη συγκεκριμένη μέθοδο και όχι μια άλλη.

*Σημείωση: Δεν πρέπει να επαναλαμβάνετε τις παρατηρήσεις που διατυπώθηκαν με τη δήλωση της συνάρτησης ή οπουδήποτε αλλού. Είναι εντάξει να ανακεφαλαιώνονται εν συντομία ποια είναι η λειτουργία της συνάρτησης, αλλά το επίκεντρο των παρατηρήσεων θα πρέπει να είναι σχετικά με το πώς επιλύει το πρόβλημα.*

## Μεταβλητές

Σε γενικές γραμμές το όνομα της μεταβλητής πρέπει να είναι αρκετά περιγραφικό, και να δίνει μια καλή ιδέα του τι είναι η μεταβλητή και πώς χρησιμοποιείται. Σε ορισμένες περιπτώσεις, απαιτούνται περισσότερα σχόλια.

```
int population_average;           // The average age of the input population
```

## Σχόλια Γραμμής

Οι γραμμές κώδικα που δεν είναι προφανές στο τί κάνουν θα πρέπει έχουν σχόλια στο τέλος τους. π.χ.,

```
// If we have enough memory, mmap the data portion too.
mmap_budget = max<int64>(0, mmap_budget - index_>length());
if (mmap_budget >= data_size_ && !MmapData(mmap_chunk_bytes, mlock))
    return; // Error already logged.
```

Σημειώστε ότι υπάρχουν σχόλια που περιγράφουν τι κάνει ο κωδικός και σχόλια που **αναφέρουν ότι ένα λάθος έχει ήδη καταγραφεί**.

Αν έχετε πολλά σχόλια γραμμών, μπορεί συχνά να είναι πιο ευανάγνωστο να τα ευθυγραμμίζετε:

```
DoSomething();           // Comment here the comments line up.
DoSomethingElseThatIsLonger(); // Comment here are two spaces between
                             // the code and the comment.
```

## NULL, true/false, 1, 2, 3...



Όταν περνούμε NULL, boolean, ή literal integer τιμές σε συναρτήσεις, πρέπει να γράφουμε και σχόλια σχετικά με το τί είναι. Για παράδειγμα:

```
bool success = CalculateSomething(interesting_value,
                                  10,
                                  false,
                                  NULL); // What are these arguments??
```

**Σε αντίθεση με:**

```
bool success = CalculateSomething(interesting_value,
                                  10,          // Default base value.
                                  false,       // Not the first time we're calling this.
                                  NULL);      // No callback.
```

Εναλλακτικά μπορούμε να χρησιμοποιούμε σταθερές με κατάλληλα ονόματα:

```
const int kDefaultBaseValue = 10;
const bool kFirstTimeCalling = false;
Callback *null_callback = NULL;
bool success = CalculateSomething(interesting_value,
                                  kDefaultBaseValue,
                                  kFirstTimeCalling,
                                  null_callback);
```

### Ποτέ

**Ποτέ μην περιγράφετε τον ίδιο το κώδικα. Να θεωρείτε ότι το άτομο που θα τον διαβάσει ξέρει πολύ καλό προγραμματισμό.**

```
// Now go through the b array and make sure that if i occurs,
// the next element is i+1.
...           // Geez. What a useless comment.
```

### Στίξεις, Ορθογραφία και Γραμματική

Δώστε προσοχή σε σημεία στίξης, την ορθογραφία και γραμματική. Είναι πιο εύκολο να διαβάσει κάποιος καλά γραμμένα σχόλια παρά να χρειάζεται χρόνο για να τα κατανοήσει. **ΠΟΤΕ GreekEnglish**

Οι παρατηρήσεις θα πρέπει συνήθως να γραφτούν ως πλήρεις προτάσεις με σωστή κεφαλαιοποίηση. Μικρότερα σχόλια, όπως τα σχόλια στο τέλος της μιας γραμμής, μπορεί μερικές φορές να είναι λιγότερο τυπικά. Πλήρης προτάσεις είναι πιο ευανάγνωστες, και παρέχουν κάποια διαβεβαίωση ότι τα σχόλια είναι πλήρης και δεν είναι μια ημιτελής σκέψη.

### Σχόλια TODO

Χρησιμοποιήστε TODO σχόλια για κώδικα που είναι προσωρινός, αποτελεί βραχυπρόθεσμη λύση, ή αρκετά καλός αλλά όχι τέλειος. Βασιστείτε στα πιο κάτω παραδείγματα.

```
// TODO(kl@gmail.com): Use a "*" here for concatenation operator.
// TODO(Zeke) change this to use relations.
```