Κεφάλαιο 14: Συμβουλές προς έναν νέο προγραμματιστή

Φτάσαμε σιγά σιγά στο τέλος του βιβλίου. Αντί για κάποιον επίλογο σκέφτηκα να συλλέξω κάποια πράγματα που θα ήθελα να πω σε κάποιον ο οποίος αρχίζει σιγά σιγά να νιώθει προγραμματιστής. Κάποια από αυτά προκύπτουν από όσα ειπώθηκαν στα προηγούμενα κεφάλαια. Κάποια όχι, και ψάχνουν μία θέση σε αυτό το βιβλίο, μερικές γραμμές να χωρέσουν, αφού είναι θέματα τα οποία, ναι μεν έχει νόημα να γνωρίζει ένας προγραμματιστής, αποτελούν όμως ολόκληρο κλάδο στην Πληροφορική και διδάσκονται σε σχετικά ακαδημαϊκά μαθήματα.

Έτσι, θεώρησα, χρήσιμο, καλό σαν επίλογο και ενδιαφέρον να σημειώσω σε μία λίστα μερικές απλές και εύπεπτες, ελπίζω, συμβουλές, γραμμένες με απλό τρόπο και χωρίς να επιχειρούν να εμβαθύνουν. Ελπίζω να τις βρείτε χρήσιμες.

- **Καταγράφουμε τις απαιτήσεις μας (system requirement specification)**: Πριν ξεκινήσουμε να σχεδιάζουμε κώδικα, πόσω μάλλον πριν ξεκινήσουμε να γράφουμε κώδικα πρέπει να είμαστε απόλυτα σίγουροι ότι έχουμε αποφασίσει τι ακριβώς θέλουμε να φτιάξουμε. Πρέπει να συνομιλήσουμε διεξοδικά με ανθρώπους που γνωρίζουν το πρόβλημα πολύ καλά, πιο καλά ίσως και από εμάς. Αν, για παράδειγμα, θέλουμε να φτιάξουμε ένα πρόγραμμα που να καταγράφει την πίεση ενός ασθενή και να παρουσιάζει κάποια αποτελέσματα πάνω στις καταγραφές σε έναν γιατρό, μήπως θα ήταν καλύτερα να μιλήσουμε πρώτα με έναν γιατρό; Προσπαθούμε, λοιπόν, να καταλάβουμε, να συγκεντρώσουμε τις ανάγκες μας, και αφού βεβαιωθούμε ότι το κάναμε εξαντλητικά και οργανωμένα, όταν δηλαδή είμαστε σίγουροι ότι ξέρουμε το πρόβλημα που πάμε να λύσουμε, στο

βάθος φυσικά που μας αφορά και μας ενδιαφέρει, τότε μπορούμε να αρχίσουμε να σκεφτόμαστε το επόμενο βήμα.

- **Επιλέγουμε την κατάλληλη γλώσσα προγραμματισμού, περιβάλλον εργασίας, πλαίσιο ανάπτυξης (integrated development environment)**: Αν και στο βιβλίο αυτό επιλέξαμε την Python σαν εργαλείο για να μυηθούμε στον κόσμο του προγραμματισμού, δεν σημαίνει ότι θα τη χρησιμοποιήσουμε σε κάθε περίπτωση και για οποιοδήποτε πρόβλημα. Αν, ας πούμε, θέλουμε να αναπτύξουμε έναν πολύ γρήγορο κώδικα που θα αποτελεί μέρος του λειτουργικού συστήματος, πρέπει να ξεχάσουμε την Python και να αρχίσουμε να κοιτάμε προς τη C. Αν θέλουμε να φτιάξουμε κάποια δικτυακή εφαρμογή, υπάρχουν πολλά έτοιμα πλαίσια που θα μας λύσουν τα χέρια. Σε κάθε περίπτωση, μελετάμε καλά τις επιλογές μας πριν πάρουμε τις αποφάσεις μας, και αφού τις πάρουμε, εξετάζουμε και τα υπάρχοντα περιβάλλοντα ανάπτυξης. Εμείς είδαμε το IDLE στην αρχή του βιβλίου αυτού, αρκετά σύντομα. Ίσως κάποιο άλλο να μας είναι περισσότερο χρήσιμο και να μας κάνει την ανάπτυξη πιο ευχάριστη.

- **Σχεδιάζουμε τον κώδικά μας (code design)**: Η σχεδίαση του κώδικα είναι το ίδιο σημαντική όσο η καταγραφή των απαιτήσεων ή τα περισσότερα από τα βήματα που θα ακολουθήσουν. Λάθος επιλογές στη σχεδίαση ή συνειδητοποίηση σε δεύτερο χρόνο λανθασμένων επιλογών που θα μας αναγκάσουν να αλλάξουμε σημεία του κώδικα τα οποία έχουμε γράψει ή που θα μεγαλώσουν χωρίς λόγο τον κώδικα και θα τον κάνουν πιο δύσχρηστο, θα οδηγήσουν σε αύξηση του χρόνου που χρειάζεται η ανάπτυξη αλλά και του κόστους του λογισμικού, αν βέβαια δεν προγραμματίζουμε για τη δόξα αλλά για τον πραγματικό κόσμο και θέλουμε να ζήσουμε από αυτό. Ένα παράδειγμα κακού σχεδιασμού είναι το να μην παρατηρήσουμε από την αρχή ότι ένας κώδικας μπορεί να γραφεί σαν συνάρτηση και να το καταλάβουμε αρκετά πιο αργά. Θα πρέπει να γράψουμε τη συνάρτηση και να γυρίσουμε πίσω σε όλα τα σημεία που θα μπορούσε να είχε χρησιμοποιηθεί αλλά δεν έγινε. Αν κάποιος σκέφτηκε ότι το τελευταίο μπορεί και να αποφευχθεί, αφού έτσι και αλλιώς ο κώδικας λειτουργεί σωστά, ας δει την επόμενη συμβουλή.

- **Αναπτύσσουμε καλογραμμένο κώδικα (code development)**: Ο χρόνος τον οποίο θα δαπανήσουμε για την ανάπτυξη του κώδικά μας αποτελεί μία επένδυση για τη συνέχεια. Σε μία όχι και πολύ μεγάλη εφαρμογή, ο όγκος του κώδικα που αναπτύσσεται μετριέται σε χιλιάδες γραμμές κώδικα. Αν σαν παράμετρο βάλουμε και τον αριθμό των προγραμματιστών που εργά-

ζονται πάνω στον ίδιο κώδικα, τα πράγματα δυσκολεύουν περισσότερο. Κάνουμε ό,τι μπορούμε για να κρατήσουμε τον κώδικά μας όσο το δυνατόν πιο ευανάγνωστο. Το να γυρίσουμε πίσω και να αντικαταστήσουμε τον κώδικα με τη συνάρτηση που αναφέραμε προηγουμένως είναι το λιγότερο. Δομούμε τον κώδικα έτσι ώστε όλα να βρίσκονται στο σωστό σημείο. Δίνουμε ευανάγνωστα ονόματα στις μεταβλητές και στις συναρτήσεις. Προτιμούμε μια μεταβλητή να την ονομάσουμε acceleration αντί για a. Στοιχίζουμε τον κώδικα. Η Python μάς απαγορεύει να μην στοιχίσουμε τον κώδικα, και πολύ καλά κάνει. Σε άλλες γλώσσες όμως η ευθύνη είναι δική μας. Ο κώδικάς μας πρέπει να διαβάζεται και από άλλους. Για να μην πω ότι πρέπει να διαβάζεται και από εμάς τους ίδιους έναν χρόνο μετά (ή τρεις μέρες μετά).

- **Επαναχρησιμοποιούμε κώδικα (code reusability)**: Δεν είναι απαραίτητο να ανακαλύπτουμε κάθε φορά τον τροχό, ούτε να παιδευόμαστε χωρίς λόγο. Ο κώδικας που θέλουμε να φτιάξουμε μπορεί ήδη να έχει φτιαχτεί, σε ένα σημαντικό τουλάχιστον μέρος του, είτε από εμάς παλαιότερα είτε από άλλους και να βρίσκεται κάπου αναρτημένος στο διαδίκτυο. Αν ο κώδικας είναι δικός μας, ακόμα καλύτερα. Συμφωνήσαμε προηγουμένως ότι θα γράφουμε καλό κώδικα. Δεν θα είναι δύσκολο, λοιπόν, να τον επαναχρησιμοποιήσουμε. Αν δεν είναι δικός μας, πρέπει να τον ελέγξουμε προσεκτικά πριν τον ενσωματώσουμε στον δικό μας. Αν μάλιστα είμαστε πιο τυχεροί, αυτό που ψάχνουμε μπορεί να είναι διαθέσιμο μέσα από κάποια ευρέως χρησιμοποιούμενη βιβλιοθήκη. Στην περίπτωση αυτήν, νιώθουμε περισσότερο ασφαλείς, η χρήση του είναι ευκολότερη και τυποποιημένη. Καλό είναι τον κώδικα που γράφουμε και πιστεύουμε ότι θα επαναχρησιμοποιήσουμε να τον οργανώνουμε σε βιβλιοθήκες, όταν αυτό είναι δυνατόν ή έχει νόημα.

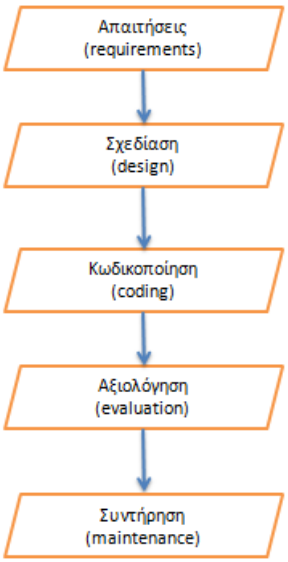
- **Τεκμηριώνουμε τον κώδικα που γράφουμε (code documentation)**: Με άλλα λόγια, του βάζουμε σχόλια. Όσο πιο αναλυτικά, τόσο ευκολότερα θα βρούμε κάποιο σφάλμα μέσα σε αυτόν, τόσο ευκολότερα θα τον καταλάβουμε και θα τον τροποποιήσουμε αν χρειαστεί να τον επαναχρησιμοποιήσουμε στο μέλλον και τόσο πιο εύκολα θα τον κατανοήσει ένας τρίτος, στενός ή από απόσταση συνεργάτης που θα χρειαστεί να το κάνει. Μην υποτιμάτε την ανάγκη της τεκμηρίωσης του κώδικα. Ο χρόνος που δαπανούμε σε αυτό είναι επένδυση για τη συνέχεια.

- **Αποσφαλμάτωνουμε τον κώδικα με επιμονή (code debugging)**: Η αποσφαλμάτωση είναι από τα πιο κουραστικά μέρη της ανάπτυξης. Το να

βρεις και να διορθώσεις όλα τα λάθη του κώδικα απαιτεί συστηματική και προσεκτική εργασία. Ο κώδικάς μας πιθανόν να δοθεί για χρήση σε ένα ευρύ κοινό, μη σχετικό με τη δική μας τεχνογνωσία. Πρέπει να εξαλειφθεί κάθε πιθανότητα οι χρήστες να έρθουν αντιμέτωποι με μία αναπάντεχη κατάσταση. Το κόστος και ο χρόνος της αποσφαλμάτωσης είναι ένα πολύ σημαντικό μέρος της συνολικής ανάπτυξης.

- **Αξιολογούμε και επιστρέφουμε για να διορθώσουμε τις αστοχίες μας (code evaluation)**: Το τελευταίο πράγμα που πρέπει να κάνουμε είναι να αξιολογήσουμε τον κώδικά μας. Φυσικά, αν μέσα από αυτήν τη διαδικασία προκύψουν κάποιες αστοχίες μας πρέπει να επιστρέψουμε και να τις διορθώσουμε. Στη διαδικασία της αξιολόγησης πρέπει να πάρει μέρος και ο ειδικός, αυτός δηλαδή που μας βοήθησε στην πρώτη φάση να συλλέξουμε τις απαιτήσεις μας.

Κάπου εδώ τελειώνουν οι γρήγορες συμβουλές. Μπορείτε να ανατρέξετε σε άλλα βιβλία ή ακαδημαϊκά μαθήματα, ώστε να δείτε όλα τα παραπάνω, όχι σαν συμβουλές, αλλά σαν μεθοδολογία και με τρόπο συστηματικό και επιστημονικό.



Σχήμα 14.1: Βασικές φάσεις ανάπτυξης λογισμικού.

Ας παραθέσουμε για πληρότητα ένα διάγραμμα (Σχήμα 14.1) που δείχνει τις βασικές φάσεις ανάπτυξης του λογισμικού, μια που οι παραπάνω συμβου-

λές το έχουν λάβει υπόψη τους. Στο διάγραμμα αυτό οι φάσεις δεν επικαλύπτονται και η μία ακολουθεί την άλλη. Φυσικά, αν από μία φάση (για παράδειγμα την αξιολόγηση) προκύψει ότι πρέπει να επιστρέψουμε σε κάποια άλλη φάση και να διορθώσουμε τις επιλογές ή τις παραλείψεις μας, αυτό μπορεί να γίνει. Η φυσιολογική ροή των φάσεων είναι: συγκέντρωση απαιτήσεων, σχεδίαση λογισμικού, υλοποίηση λογισμικού, αξιολόγηση λογισμικού και, τέλος, συντήρηση λογισμικού.

Σας ευχαριστώ που διαβάσατε μέρος του ή ολόκληρο αυτό το βιβλίο. Ελπίζω να σας φάνηκε χρήσιμο. Εύχομαι σε όλους καλό ταξίδι στον μαγευτικό κόσμο της Πληροφορικής που έχετε μπροστά σας.