

## ΠΡΟΛΟΓΟΣ

---

### ΣΗΜΕΙΩΜΑ ΤΟΥ ΣΥΓΓΡΑΦΕΑ

Κανένας δεν μπορεί να χτίσει για λογαριασμό σου το γεφύρι, απ' όπου χρωστάς να περάσεις το ποτάμι της ζωής. Κανένας εκτός από σένα τον ίδιο. Υπάρχουν, βέβαια, μονοπάτια άπειρα και γεφύρια και ημίθεοι πρόθυμοι να σε περάσουν· μα θα ζητήσουν πληρωμή τον ίδιο σου τον εαυτό.

*Ραμπιντρανάθ Ταγκόρ, Ινδός ποιητής*

Η συγγραφή ενός βιβλίου είναι ιδιαίτερα απαιτητική εργασία. Η συγγραφή ενός βιβλίου για οποιαδήποτε γλώσσα προγραμματισμού έχει επιπλέον δυσκολία, διότι απαιτείται ακρίβεια ώστε τα αποτελέσματα που θα δει μπροστά του ο/η προγραμματιστής/στρια, όταν εφαρμόζει τα όσα διαβάζει στην ανάπτυξη προγραμμάτων, να ταυτίζονται όσο το δυνατόν περισσότερο με όσα είναι τυπωμένα στο χαρτί και δεν αλλάζουν (για τον προγραμματισμό, το χαρτί είναι hardware). Η απαίτηση αυτή αυξάνει την πολυπλοκότητα της συγγραφής, ειδικά αν ληφθεί υπόψη το πλήθος των αρχιτεκτονικών υπολογιστικών συστημάτων και των περιβαλλόντων ανάπτυξης εφαρμογών.

Είναι σαφές ότι η ανάπτυξη προγραμμάτων δεν μαθαίνεται μόνο με το διάβασμα. Είναι ιδιαίτερα σημαντικό κάθε αρχάριος/α προγραμματιστής/στρια να αφιερώσει τον απαιτούμενο χρόνο ώστε να δημιουργήσει ένα λειτουργικό περιβάλλον ανάπτυξης προγραμμάτων και στη συνέχεια να εφαρμόσει ο/η ίδιος/α τις έννοιες που παρουσιάζονται εδώ αναπτύσσοντας προγράμματα. Είναι πολύ σημαντικό όποιος/α επιθυμεί να ασχοληθεί επαγγελματικά με τον προγραμματισμό να ασχοληθεί σε βάθος και να πειραματιστεί με τον τρόπο λειτουργίας των δομών της (εκάστοτε) γλώσσας προγραμματισμού.

Όσο πιο συνοπτικά γίνεται: ο προγραμματισμός μαθαίνεται βάζοντας τα χέρια στο πληκτρολόγιο, γράφοντας, μεταγλωττίζοντας, διορθώνοντας και, τέλος, βελτιστοποιώντας προγράμματα. Όπως δεν γίνεται να μάθει κάποιος/α να κάνει ποδήλατο διαβάζοντας εγχειρίδια, έτσι δεν αρκεί το διάβασμα για να μάθει προγραμματισμό.

Η ενασχόληση με τον προγραμματισμό είναι μια ιδιαίτερα δημιουργική ενασχόληση. Ο επαγγελματίας προγραμματιστής έχει μεγάλη γκάμα από επαγγελματικές επιλογές, ενώ μπορεί να δουλεύει εξ αποστάσεως από οποιοδήποτε μέρος του κόσμου επιθυμεί. Φυσικά, είναι μια επίπονη ενασχόληση, ωστόσο είναι μια από τις πιο καλοπληρωμένες επίπονες ενασχολήσεις που υπάρχουν.

Η ανάπτυξη προγραμμάτων, εκτός από γνώση και δυνατότητα επινόησης αλγορίθμων, απαιτεί σαφήνεια και ακρίβεια ώστε τα προγράμματα να είναι σύμφωνα με το συντακτικό και τις απαιτήσεις των δομών της γλώσσας. Αυτό, ιδιαίτερα στην αρχή της διαδικασίας εκμάθησης, είναι κουραστικό· είναι όμως ο μόνος τρόπος που οδηγεί με ασφάλεια στην κατανόηση της τέχνης του προγραμματισμού.

Είναι σαφές ότι η ανάπτυξη προγραμμάτων είναι μια διαδικασία διαρκούς εξέλιξης και αναζήτησης· οι δε πηγές των αναζητήσεων, εκτός από τα βιβλία, είναι το Διαδίκτυο και τα chatbots της παραγωγικής Τεχνητής Νοημοσύνης. Οι εκπαιδευτικοί παρατηρούμε ότι οι νέοι άνθρωποι προτιμούν τη μάθηση με χρήση video και πηγών του Διαδικτύου παρά με τη μελέτη βιβλίων και εγχειριδίων. Παρά το ότι, κατά τη γνώμη μου, η μελέτη των βιβλίων έχει καλύτερα και μονιμότερα γνωστικά αποτελέσματα σε σχέση με τη μελέτη των πηγών του Διαδικτύου όσον αφορά την κατανόηση πολύπλοκων εννοιών και φαινομένων, είναι σαφές ότι μπορεί και πρέπει να συμπληρώνεται από κατάλληλα διατυπωμένες αναζητήσεις σε πηγές του Διαδικτύου και chatbots και αξιολόγηση των αποτελεσμάτων που επιστρέφονται. Τόσο η σαφής διατύπωση των ερωτημάτων όσο και η αξιολόγηση και χρήση των επιστρεφόμενων αποτελεσμάτων απαιτούν ο/η χρήστης/στρια των μέσων αυτών να είναι εξοικειωμένος με τις αρχές του προγραμματισμού και τις δομές της εκάστοτε γλώσσας προγραμματισμού.

Για να μπορεί η αναζήτηση στο Διαδίκτυο να είναι αποδοτική, θα πρέπει ο/η Έλληνας/ίδα προγραμματιστής/στρια να έχει γνώση των δόκιμων αγγλικών όρων. Ωστόσο, αυτό είναι ένα βιβλίο γραμμένο στα ελληνικά (και όχι σε ελληνικά πασσαλισμένα με αγγλικές λέξεις). Με δεδομένο ότι ο Προγραμματισμός σε C είναι εισαγωγικό μάθημα σε Τμήματα ή κατευθύνσεις σπουδών Πληροφορικής και Τμήματα Θετικών Επιστημών, η προσέγγιση που υιοθετήθηκε είναι η αναγραφή πρώτα της δόκιμης ελληνικής ορολογίας και στη συνέχεια, σε παρένθεση, η παράθεση της αντίστοιχης αγγλικής ορολογίας.

Το ερώτημα που, βιαστικά, κάνουν πολλοί «Γιατί να ασχοληθώ με τη C;» ελπίζω να ξεθωριάζει καθώς ο/η αναγνώστης/στρια διαβάζει το βιβλίο. Αυτός/ή που θα εμβαθύνει, αφιερώνοντας τον ανάλογο χρόνο και κόπο, θα ωφεληθεί με πολλούς τρόπους τόσο στην εκμάθηση βασικών προγραμματιστικών αρχών και τεχνικών όσο και στην κατανόηση του τρόπου με τον οποίο μπορεί να χρησιμοποιείται η C για την ανάπτυξη προγραμμάτων.

Η παρουσίαση της ύλης γίνεται με τέτοιο τρόπο ώστε να αποκαλύπτεται το εύρος των δυνατοτήτων τις οποίες η C παρέχει στους προγραμματιστές. Η συνεισφορά αυτού του βιβλίου είναι, πιστεύω, η σε βάθος παρουσίαση δομών της γλώσσας που ανοίγουν τον δρόμο για τη χρήση της C σε πολλά πεδία εφαρμογών αλλά και για την

ευκολότερη κατανόηση του αντικειμενοστρεφούς προγραμματισμού. Για τον λόγο αυτό, δόθηκε ιδιαίτερη έμφαση στην παρουσίαση εννοιών όπως οι δείκτες, οι δείκτες σε συναρτήσεις, η δυναμική ανάθεση μνήμης, η διαχείριση δεδομένων σε επίπεδο bit και οι τεχνικές δομημένης ανάπτυξης μεγάλων προγραμμάτων.

Με τον τρόπο αυτό ευελπιστώ ότι αποκαλύπτεται η ιδιαίτερη «φύση» της C: πρόκειται για μια γλώσσα ταυτόχρονα υψηλού επιπέδου αλλά και όσο πιο κοντά στο υλικό γίνεται. Η «ανεκτικότητα» της απελευθερώνει τη δημιουργικότητα του προγραμματιστή που έχει εμβαθύνει στα χαρακτηριστικά της. Ωστόσο η «λεπίδα είναι δίκωπη»: η ανεκτικότητα οδηγεί σε προγράμματα που είναι επιρρεπή σε λάθη. Τα λάθη αυτά, σε ορισμένες περιπτώσεις, εντοπίζονται δύσκολα χωρίς υποστήριξη εξειδικευμένου λογισμικού. Εν κατακλείδι, η αξιοποίηση των χαρακτηριστικών της C απαιτεί σοβαρή και σε βάθος μελέτη και κατανόηση των χαρακτηριστικών της γλώσσας και της διεπαφής της με το εκάστοτε λειτουργικό σύστημα.

Η μελέτη του υλικού του βιβλίου αναμένεται «να ανοίξει την όρεξη» του/ης ενδιαφερόμενου/ης για τον προγραμματισμό συστημάτων (system programming) και ενσωματωμένων συστημάτων (embedded systems) που είναι ιδιαίτερα σημαντικά πεδία εφαρμογής της C.

Σκόπιμα δεν περιλήφθηκαν κεφάλαια για την υλοποίηση δομών δεδομένων σε C: οι δομές δεδομένων είναι ένα τα πιο περίπλοκα και ενδιαφέροντα ζητήματα της Επιστήμης και Τεχνολογίας των Υπολογιστών. Η αποσπασματική αναφορά σε αυτές, με έμφαση στην υλοποίησή τους, περισσότερο θα συσκοτίζε ένα ζήτημα κομβικό για τις σπουδές ενός ειδικού της Πληροφορικής, παρά θα βοηθούσε στην κατανόησή του. Άλλωστε, ο προγραμματιστής που θα ολοκληρώσει τη μελέτη του βιβλίου θα κατέχει όλα τα απαραίτητα «εργαλεία» ώστε να αναπτύξει προγράμματα που υλοποιούν αποδοτικά δομές δεδομένων και αλγορίθμους αναζήτησης και ταξινόμησης. Η υλοποίηση δομών δεδομένων και αλγορίθμων θα πρέπει να συνοδεύεται από την ενδελεχή μελέτη των αντίστοιχων αντικειμένων στο πλαίσιο του σχετικού μαθήματος. Σκόπιμα, επίσης, δεν περιλήφθηκε κώδικας για τους βασικούς αλγορίθμους αναζήτησης και ταξινόμησης που είναι γνωστοί στους φοιτητές από την ύλη του πανελλαδικώς εξεταζόμενου μαθήματος της Γ΄ Λυκείου, η δε υλοποίησή τους στη C αφήνεται ως προγραμματιστική εργασία.

Έγινε προσπάθεια ώστε ο κώδικας, όπως βρίσκεται στην τυπωμένη στο βιβλίο μορφή, να είναι απαλλαγμένος από οποιασδήποτε μορφής λάθη. Ωστόσο, είναι πολύ πιθανό να έχουν παρεισφρήσει λάθη, συντακτικά και λογικά, καθώς ο κώδικας μεταφερόταν από το VSCode στο χαρτί. Ο/Η ενδιαφερόμενος/η μπορεί να βρει τον κώδικα του βιβλίου στη δικτυακή θέση <https://www.dardanosnet.gr/product/eisagogi-ston-programmatismo-me-ti-c/> των εκδόσεων Gutenberg και στη διεύθυνση <https://github.com/plamps/Class-Source-Code>.

Ευχαριστώ εκ των προτέρων τους φοιτητές και τις φοιτήτριες που με τις ερωτήσεις τους αποκάλυψαν το εύρος και το βάθος στο οποίο θα πρέπει να καλύπτονται συγκεκριμένες έννοιες, ειδικά αυτές που παραδοσιακά δυσκολεύουν τους/ις

αρχάριους προγραμματιστές/στριες. Ελπίζω ο χρόνος που αφιερώθηκε στη συγγραφή του βιβλίου να «γονιμοποιήσει» τον χρόνο τον οποίο θα αφιερώσουν οι φοιτητές/τριες που επιθυμούν να ασχοληθούν με τον προγραμματισμό σε C.

Κλείνοντας, θα ήθελα να ευχαριστήσω τον φίλο και εξαιρετικό συνάδελφο καθηγητή Σπύρο Λάλη, για την ευγενική διάθεση του υλικού που χρησιμοποιεί για τη διδασκαλία της C: η δομή και τα περιεχόμενα του υλικού του αποτέλεσαν τη βάση για τη συγγραφή του βιβλίου που κρατάτε στα χέρια σας. Ευτυχώς για μένα γνωριστήκαμε στην αρχή της πορείας μου στην τριτοβάθμια εκπαίδευση. Οφείλω να παραδεχτώ ότι οι γνώσεις και το ήθος του καθώς και ο τρόπος που προσεγγίζει τη διδασκαλία και την έρευνα στο αφιλόξενο για αυτά περιβάλλον των ελληνικών ΑΕΙ καθόρισαν και καθορίζουν τον τρόπο με τον οποίο αντιλαμβάνομαι τον ρόλο του λειτουργού της δημόσιας τριτοβάθμιας εκπαίδευσης.

Κλείνοντας, ευχαριστώ τους συνεργάτες μου στις εκδόσεις Gutenberg για την άρτια επεξεργασία του πρωτότυπου και το άψογο τελικό αποτέλεσμα. Για να επισημάνετε λάθη, παραλείψεις ή υποδείξεις παρακαλώ επικοινωνήστε μαζί μου στο [plampsas@uth.gr](mailto:plampsas@uth.gr).

*Πέτρος Λάμψας, Σεπτέμβριος 2024*

## ΚΕΦΑΛΑΙΟ 1

---

# ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

Με την ολοκλήρωση της μελέτης του παρόντος κεφαλαίου, οι φοιτητές/τριες θα είναι σε θέση:

- Να αναφέρουν τα βασικά χαρακτηριστικά της γλώσσας προγραμματισμού C.
- Να γνωρίζουν την εξέλιξη των τυποποιήσεων της C.
- Να αναφέρουν περιοχές της Πληροφορικής όπου η C παραμένει μια βασική επιλογή για τους/ις προγραμματιστές/τίστριες.

### Σύντομη ιστορική αναδρομή

Η C είναι ένα «παραπροϊόν» του UNIX, που αναπτύχθηκε στα Bell Laboratories (AT&T Bell Labs) από τον Ken Thompson, τον Dennis Ritchie και άλλους. Αρχικά, ο Thompson σχεδίασε μια απλή γλώσσα που ονόμασε B και η οποία βασιζόταν στην BCPL, μια γλώσσα προγραμματισμού συστημάτων που αναπτύχθηκε στα μέσα της δεκαετίας του 1960.

Γύρω στο 1971, ο Ritchie ξεκίνησε να αναπτύσσει μια εκτεταμένη έκδοση της B που αρχικά ονόμασε NB (προφερόταν «New B»). Καθώς η γλώσσα απομακρυνόταν όλο και περισσότερο από την B, άλλαξε το όνομά της σε C. Η γλώσσα αυτή ήταν ήδη σταθερή το 1973, με αποτέλεσμα το UNIX, κυρίαρχο τότε λειτουργικό σύστημα για κεντρικούς υπολογιστές (mainframes), να ξαναγραφτεί στη C.

Αναλυτικά ιστορικά στοιχεία για τη C μπορεί ο/η ενδιαφερόμενος/η να εντοπίσει στο σχετικό με τη C άρθρο της Wikipedia.

### Ιδιότητες της C

Η C είναι μια σχετικά απλή γλώσσα προγραμματισμού γενικού σκοπού (**general purpose programming language**). Είναι ενσωματωμένη στα λειτουργικά συ-

στήματα Unix και Linux και στις παραλλαγές τους, μεγάλο μέρος του κώδικα των οποίων είναι γραμμένο σε C.

Είναι μια γλώσσα προστακτικού (imperative) προγραμματισμού με σχετικά λίγες εντολές. Έχει αδύναμο σύστημα ελέγχου τύπων δεδομένων (για τον λόγο αυτό χαρακτηρίζεται ως χαλαρή ή ανεκτική), ενώ η ελάχιστη μονάδα δόμησης κώδικα είναι η συνάρτηση.

Ένα πρόγραμμα C πρώτα υφίσταται **μεταγλώττιση (compilation)**, δηλαδή μετατρέπεται σε εκτελέσιμο, και μετά εκτελείται. Το τελικό εκτελέσιμο μπορεί να δημιουργείται ως αποτέλεσμα σύνθεσης της μεταγλώττισης περισσότερων του ενός αρχείων, κάποια από τα οποία μπορεί να είναι ήδη εκτελέσιμα. Στο Κεφάλαιο 3 παρουσιάζεται αναλυτικά η διαδικασία της μεταγλώττισης και στο Κεφάλαιο 18 η μεταγλώττιση μεγάλων προγραμμάτων τα οποία αποτελούνται από πολλά ξεχωριστά τμήματα λογισμικού που αλληλεπιδρούν.

Η C είναι ανεξάρτητη από ΚΜΕ και μεταφέριμη (Machine Independent/Portable). Υποστηρίζεται από όλες σχεδόν τις αρχιτεκτονικές υπολογιστικών συστημάτων. Είναι η γλώσσα στην οποία αναπτύσσονται λειτουργικά συστήματα, δηλαδή χρησιμοποιείται στην ανάπτυξη τμημάτων του πυρήνα (kernel) ενός λειτουργικού συστήματος, οδηγών συσκευών (drivers) κ.λπ.

Πέρα από τα χαρακτηριστικά που την κάνουν κατάλληλη για προγραμματισμό λειτουργικών συστημάτων, διαθέτει πληθώρα χαρακτηριστικών που την καθιστούν κατάλληλη για ανάπτυξη εφαρμογών. Τα χαρακτηριστικά αυτά είναι:

- Δομημένη και επεκτάσιμη ανάπτυξη προγραμμάτων με συναρτήσεις.
- Δυνατότητες τροποποίησης και επαναχρησιμοποίησης κώδικα.
- Πλούσια **βιβλιοθήκη (library)** με έτοιμες συναρτήσεις.
- Δυνατότητες διαχείρισης μνήμης (memory management).
- Υποστήριξη δεικτών (pointers) για την αλληλεπίδραση με τη μνήμη.
- Υποστήριξη μηχανισμού αναδρομικής κλήσης συναρτήσεων (recursion).
- Υποστήριξη από όλα τα διαδεδομένα αρχιτεκτονικά υπολογιστικά συστήματα και τα λειτουργικά συστήματα.

## Μειονεκτήματα της C

Η χρήση της C για την ανάπτυξη προγραμμάτων έχει τα ακόλουθα μειονεκτήματα:

- Τα προγράμματα σε C είναι επιρρεπή σε λάθη, ειδικά από αρχάριους ή μη ειδικούς στη C προγραμματιστές.
- Οι «ευκολίες» που παρέχει στην πρόσβαση στη μνήμη και η χαλαρότητα είναι εύκολο να οδηγήσουν σε σφάλματα κατά τον χρόνο εκτέλεσης του προγράμματος.

- Τα προγράμματα σε C μπορεί να είναι γραμμένα με τρόπο που δυσχεραίνει την κατανόηση και επαναχρησιμοποίησή τους.

### Συστάσεις για αποδοτική μελέτη και χρήση της C

Όπως θα γίνει σαφές στη συνέχεια, υπάρχουν ορισμένοι βασικοί κανόνες για την αποδοτική χρήση της C. Ο/Η προγραμματιστής/τίστρια σε C θα πρέπει:

- Να υιοθετεί ένα καλά επεξεργασμένο σύνολο κανόνων συγγραφής κώδικα.
- Να αποφεύγει «κόλπα» και τον υπερβολικά πολύπλοκο κώδικα.
- Να χρησιμοποιεί ήδη υπάρχουσες βιβλιοθήκες κώδικα.
- Να χρησιμοποιεί έτοιμα εργαλεία λογισμικού (Integrated Development Environments, debuggers, εργαλεία στατικής ανάλυσης προγραμμάτων) ώστε να διευκολύνεται η ανάπτυξη αξιόπιστων προγραμμάτων.
- Να ακολουθεί πιστά τις τυποποιήσεις και την τεχνογνωσία των ομάδων προγραμματισμού σε C που λειτουργούν στο Διαδίκτυο (π.χ. Stack Overflow).

Η C δεν είναι μια γλώσσα που ανήκει στο μουσείο της Πληροφορικής. Παρά το ότι δεν είναι η πρώτη επιλογή για την ανάπτυξη εμπορικών εφαρμογών και παρά το ότι είναι σχετικά δύσκολη στην εκμάθηση, παραμένει μια ενεργή γλώσσα προγραμματισμού που χρησιμοποιείται σε πληθώρα αντικειμένων. Όπως είναι αναμενόμενο, υπάρχει τεράστια εγκατεστημένη βάση γραμμένων σε C προγραμμάτων καθώς επίσης και ιδιαίτερα μεγάλη εμπειρία στην ανάπτυξη προγραμμάτων σε C. Το Σχήμα 1.1 παρουσιάζει τις περιοχές της Πληροφορικής όπου η C παραμένει μια βασική επιλογή για όσους αναπτύσσουν προγράμματα σε αυτές.

ΣΧΗΜΑ 1.1 Πεδία εφαρμογής της C.



## Τυποποιήσεις της C

Προγράμματα σε C γράφονται εδώ και 5 δεκαετίες. Ο χρόνος αυτός είναι, για την Τεχνολογία των Υπολογιστών, τεράστιος, με αποτέλεσμα πολλές αλλαγές και προσαρμογές να ήταν απαραίτητο να γίνουν στην αρχική έκδοση της C. Στη συνέχεια, αναφέρονται συνοπτικά οι τυποποιήσεις της γλώσσας C από την εμφάνισή της έως την περίοδο συγγραφής του βιβλίου:

- **K&R C**: Είναι η C των Kernighan και Ritchie, δύο εκ των εφευρετών της γλώσσας. Περιγράφεται στο βιβλίο *The C Programming Language*, Kernighan and Ritchie, 1978 (κυκλοφορεί στα ελληνικά από τις εκδόσεις Κλειδάριθμος). Έγινε τυποποίηση εκ των πραγμάτων (de facto standard) και χρησιμοποιήθηκε για πολλά χρόνια.
- **C89/C90**: Η πρώτη επίσημη τυποποίηση για τις ΗΠΑ (με όνομα ANSI X3.159-1989) ολοκληρώθηκε το 1988 και εγκρίθηκε επίσημα τον Δεκέμβριο του 1989. Έγινε **διεθνής τυποποίηση (international standard)** με τίτλο ISO/IEC 9899:1990.
- **C99**: Έγινε διεθνής τυποποίηση με όνομα ISO/IEC 9899:1999. Ενσωματώνει τις αλλαγές από την Τροποποίηση 1 (Amendment 1) της C89/C90 (η Τροποποίηση 1 ονομάστηκε C95). Εισάγονται νέοι τύποι δεδομένων (`long long`, `_Bool`, `_Complex` και `_Imaginary`) και υποστηρίζονται **χαρακτήρες πολλών byte** για Είσοδο/Εξόδο. Η C99 ενσωματώνει πολλά νέα για τη C χαρακτηριστικά, νέα αρχεία επικεφαλίδων, ενώ αφαιρούνται χαρακτηριστικά της C89/C90 που είχαν αποδειχθεί επικίνδυνα.
- **C11**: Έγινε διεθνής τυποποίηση με όνομα ISO/IEC 9899:2011/Cor 1:2012. Αξιοσημείωτα νέα χαρακτηριστικά είναι η βελτιωμένη υποστήριξη Unicode και η υποστήριξη προγραμματισμού με νήματα (multi-threading API) για πολλές πλατφόρμες.
- **C17**: Είναι μια τυποποίηση που λύνει προβλήματα της C11 χωρίς να εισάγει νέα χαρακτηριστικά. Δημοσιεύτηκε το 2018 με επίσημη ονομασία ISO/IEC 9899:2018.
- **C23**: Η τελευταία τυποποίηση της γλώσσας C. Δημοσιεύτηκε στις αρχές του 2024 με επίσημη ονομασία ISO/IEC 9899:2024.

Κομβική θεωρείται η τυποποίηση C99, διότι σε αυτήν προστέθηκαν πολλά νέα στοιχεία και έγιναν αλλαγές ώστε η C να μπορεί να χρησιμοποιηθεί σε μεγαλύτερο πεδίο εφαρμογών. Ιδιαίτερα σημαντική είναι και η τυποποίηση C11. Στο βιβλίο αυτό παρουσιάζουμε τα χαρακτηριστικά της C όπως αυτά έχουν διαμορφωθεί έως την έκδοση C23. Όπου απαιτείται, αναφέρεται στο κείμενο η έκδοση της C η οποία υποστηρίζει τα χαρακτηριστικά που περιγράφονται.

## Γλώσσες που βασίζονται στη C

Η C έχει επηρεάσει τον σχεδιασμό πολλών γλωσσών προγραμματισμού. Ορισμένα παραδείγματα τέτοιων γλωσσών είναι:

- Η C++ περιέχει όλα τα χαρακτηριστικά της C και επιπροσθέτως χαρακτηριστικά που υποστηρίζουν τον αντικειμενοστρεφή προγραμματισμό (object-oriented programming).
- Η Java βασίζεται στη C++ και κατά συνέπεια κληρονομεί πολλά από τα χαρακτηριστικά της C.
- Η C# είναι πιο πρόσφατη από τις προηγούμενες δύο και προέρχεται από τις C++ και Java.
- Η Perl έχει υιοθετήσει πολλά από τα χαρακτηριστικά της C.

## ΚΕΦΑΛΑΙΟ 2

---

### ΔΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΤΗΣ C

Με την ολοκλήρωση της μελέτης του παρόντος κεφαλαίου, οι φοιτητές/τριες θα είναι σε θέση:

- Να γνωρίζουν τη δομή ενός απλού προγράμματος σε C.
- Να αναφέρουν τους βασικούς τύπους δεδομένων της C και το πεδίο τιμών τους.
- Να κατανοούν την υπερχείλιση και υποχείλιση σε αριθμητικά δεδομένα και να εφαρμόζουν τρόπους αντιμετώπισής τους.
- Να δηλώνουν μεταβλητές και να αναθέτουν τιμές σε αυτές.
- Να αναπτύσσουν απλά προγράμματα στη γλώσσα C.

Στο κεφάλαιο αυτό θα παρουσιαστούν τα δομικά στοιχεία της γλώσσας C, αυτά που απαιτούνται για να είναι σε θέση κάποιος, αρχάριος ή προχωρημένος, προγραμματιστής να ξεκινήσει την ανάπτυξη προγραμμάτων σε C. Πιο συγκεκριμένα θα παρουσιαστεί η δομή των προγραμμάτων σε C, οι τύποι δεδομένων που υποστηρίζει η C και ο χώρος που απαιτεί κάθε τύπος δεδομένων για την αποθήκευσή του στη μνήμη, καθώς και η δήλωση μεταβλητών σε προγράμματα και η ανάθεση τιμών σε αυτές.

#### Δομή ενός προγράμματος στη C

Ακόμα και τα απλούστερα προγράμματα σε C δομούνται κάνοντας χρήση τριών βασικών χαρακτηριστικών της γλώσσας: των **οδηγιών (directives)**, των **εντολών (statements)** και των **συναρτήσεων (functions)**. Τα προγράμματα συμπληρώνονται με **σχόλια (comments)** που επεξηγούν τη λειτουργικότητά τους και συμβάλλουν στην αναγνωσιμότητα του προγράμματος.

Η δομή ενός απλού προγράμματος σε C είναι η ακόλουθη:

```

/* Σχόλιο: όνομα προγράμματος, συγγραφέας,
   λειτουργία, ... */
<οδηγίες>

int main(void) {
  <δηλώσεις μεταβλητών>

  <εντολές εισόδου, εξόδου, ελέγχου και
   εντολές επεξεργασίας μεταβλητών>

}

```

## Σχόλια στη C

Σχόλιο είναι το κείμενο που ξεκινά με τους χαρακτήρες /\* και τελειώνει με τους χαρακτήρες \*/. Ένα παράδειγμα σχολίου είναι αυτό που ακολουθεί:

```
/* Αυτό είναι ένα σχόλιο */
```

Σχόλια μπορούν να υπάρχουν σχεδόν οπουδήποτε σε ένα πρόγραμμα, είτε σε ξεχωριστές είτε στις ίδιες γραμμές με εντολές του προγράμματος, ενώ μπορεί να ξεπερνούν τη μία γραμμή. Συνήθως χρησιμοποιούνται στην αρχή ενός προγράμματος για να παρουσιάσουν τη λειτουργικότητά του και στα σημεία του κώδικα που απαιτείται για να διευκρινίζουν τις λειτουργίες τις οποίες επιτελεί ο κώδικας.

Τα σχόλια είναι κείμενο που αφαιρείται στο πρώτο στάδιο της μεταγλώττισης ενός προγράμματος. Παρά το ότι τα σχόλια δεν λαμβάνονται υπόψη για τη δημιουργία του εκτελέσιμου, είναι ιδιαίτερα χρήσιμα και πρέπει να χρησιμοποιούνται όπου ο προγραμματιστής κρίνει ότι απαιτείται προκειμένου να βελτιώνεται η αναγνωσιμότητα του προγράμματος.

Τα σχόλια που ξεκινούν με /\* πρέπει να τερματίζονται προσοχή. Στο ακόλουθο παράδειγμα, το σχόλιο που δεν τερματίστηκε στην πρώτη γραμμή καταργεί, δίχως κάποια ειδοποίηση κατά τη μεταγλώττιση, δύο γραμμές προγράμματος, χωρίς κάτι τέτοιο να είναι στις προθέσεις του προγραμματιστή. Θα πρέπει να είναι σαφές ότι ο ατελής τερματισμός των σχολίων μπορεί να οδηγήσει στην παραγωγή εσφαλμέ-

## 20 Εισαγωγή στον Προγραμματισμό με τη C

νων αποτελεσμάτων, χωρίς κάτι τέτοιο να μπορεί να γίνει άμεσα κατανοητό από τον προγραμματιστή.

```
printf("Please "); /* σχόλιο που δεν τερματίστηκε...  
printf("be ");  
printf("careful "); /* άρα τελειώνει εδώ */  
printf("when using comments");
```

Από τη C99 και μετά, σχόλια που εκτείνονται μόνο σε μία γραμμή μπορούν να γράφονται και με τον ακόλουθο τρόπο:

```
// Αυτό είναι σχόλιο
```

Ένα τέτοιο σχόλιο τερματίζεται αυτόματα στο τέλος της γραμμής και έχει μερικά πλεονεκτήματα σε σχέση με τα σχόλια που αρχίζουν με `/*` και τελειώνουν με `*/`:

- Είναι πιο ασφαλή: δεν υπάρχει περίπτωση ένα μη τερματισμένο σχόλιο να «αδρανοποιήσει» ένα τμήμα του προγράμματος, όπως μπορεί να συμβεί στα σχόλια `/* ... */`.
- Είναι ευκολότερο να εντοπιστούν στον κώδικα.

Θα πρέπει να τονιστεί ότι, εκτός από τα σχόλια, δεν λαμβάνονται υπόψη από τον μεταγλωττιστή κενοί χαρακτήρες, στηλοθέτες (tabs) και κενές γραμμές που βρίσκονται ανάμεσα σε εντολές. Οι χαρακτήρες αυτοί, που χρησιμοποιούνται για τη μορφοποίηση ενός προγράμματος, χρησιμεύουν, όπως και τα σχόλια, μόνο στη βελτίωση της αναγνωσιμότητας του κώδικα.

Τόσο τα σχόλια όσο και η επιλογή ενός στυλ για τη μορφοποίηση του προγράμματος είναι πολύ σημαντικά για την αναγνωσιμότητα και τη συντήρηση/επέκτασή του.

### Οι οδηγίες στη C

Οι οδηγίες είναι ειδικές εντολές που ξεκινούν με `#` και χρησιμοποιούνται για να διευκολύνουν τον **μεταγλωττιστή (compiler)** στη δημιουργία του αρχείου που θα περιέχει τον εκτελέσιμο κώδικα. Εξ ορισμού, κάθε οδηγία καταλαμβάνει μία γραμμή, ενώ δεν πρέπει να τοποθετείται ελληνικό ερωτηματικό ή άλλος ειδικός χαρακτήρας στο τέλος της γραμμής.

Για την επεξεργασία των οδηγιών είναι υπεύθυνος ο **προεπεξεργαστής (pre-processor)** που συνήθως αποτελεί τμήμα του μεταγλωττιστή. Ο προεπεξεργαστής, όπως λέει και το όνομά του, επεξεργάζεται τις οδηγίες του προγράμματος

πριν αυτό δοθεί στον μεταγλωττιστή για τη δημιουργία του εκτελέσιμου, του αρχείου δηλαδή που περιέχει τον κώδικα του προγράμματος σε μορφή που μπορεί να εκτελεστεί σε μια συγκεκριμένη αρχιτεκτονική υπολογιστικού συστήματος. Περισσότερα για τα βήματα δημιουργίας ενός εκτελέσιμου αρχείου στο Κεφάλαιο 3.

## Συναρτήσεις

Ένα πρόγραμμα σε C μπορεί να περιέχει μία ή περισσότερες συναρτήσεις. Μια συνάρτηση είναι μια ακολουθία εντολών που έχουν ομαδοποιηθεί και τους έχει δοθεί ένα όνομα. Ένα πρόγραμμα πρέπει, υποχρεωτικά, να περιέχει τουλάχιστον μία συνάρτηση, την **κύρια συνάρτηση**, που στη C ονομάζεται `main()`. Η εκτέλεση ενός προγράμματος C αρχίζει από την πρώτη εντολή της συνάρτησης `main()`.

Μια συνάρτηση περιέχει ένα **σώμα (body)** εντολών. Με τον όρο σώμα (ή μπλοκ) εντολών αναφερόμαστε σε μια ομάδα μίας ή περισσότερων εντολών που γράφονται ανάμεσα σε `{` (που συμβολίζει την αρχή του σώματος εντολών) και `}` (που συμβολίζει το τέλος του σώματος εντολών).

Η C παρέχει στους προγραμματιστές ως τμήμα της υλοποίησής της έτοιμες **συναρτήσεις βιβλιοθήκης (library functions)**. Οι συναρτήσεις αυτές ομαδοποιούνται σε κατηγορίες (π.χ. εισόδου/εξόδου, μαθηματικές, κ.ά.) και χρησιμοποιούνται από τους προγραμματιστές για να διευκολύνεται η ανάπτυξη κώδικα.

Για να εκτελεστεί ο κώδικας μιας συνάρτησης πρέπει κάποιο σημείο του προγράμματος να την «καλέσει» (**function call**). Μια συνάρτηση μπορεί να καλέσει άλλες συναρτήσεις, που με τη σειρά τους μπορεί να καλέσουν άλλες συναρτήσεις. Όταν κάποιος χρήστης εκτελεί ένα πρόγραμμα C που περιέχει πολλές συναρτήσεις, το λειτουργικό σύστημα ξεκινάει την εκτέλεση από τη συνάρτηση `main()`, ανεξάρτητα από το πόσες συναρτήσεις αποτελούν το πρόγραμμα.

Μια συνάρτηση που υπολογίζει μια τιμή χρησιμοποιεί την εντολή `return` για να «επιστρέψει» την τιμή αυτή στη συνάρτηση που την καλεί. Στο ακόλουθο παράδειγμα η συνάρτηση που καλεί τη `return` επιστρέφει στη συνάρτηση που την κάλεσε, μετά το παράδειγμα την τιμή της μεταβλητής `x` αυξημένη κατά 1.

```
return x + 1;
```

Όταν ολοκληρωθεί η εκτέλεση του προγράμματος, η `main()` «επιστρέφει» στο περιβάλλον που κάλεσε το πρόγραμμα έναν **κωδικό κατάστασης (status code)**. Η επιστροφή κωδικού κατάστασης γίνεται με την εντολή `return` και (συνήθως) η τιμή 0 υποδηλώνει επιτυχημένη ολοκλήρωση του προγράμματος.

## 22 Εισαγωγή στον Προγραμματισμό με τη C

Συμπερασματικά, οι συναρτήσεις είναι αυτοτελή τμήματα κώδικα που βοηθούν στην καλύτερη δόμηση των προγραμμάτων και στην επαναχρησιμοποίηση τμημάτων κώδικα.

### Μεταγλώττιση και εκτέλεση προγραμμάτων

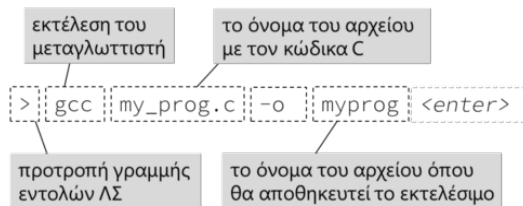
Έστω το ακόλουθο πρόγραμμα σε C που αποθηκεύεται ως `my_prog.c`:

```
#include <stdio.h>

int main(void) {
    printf("hello world\n");
    return 0;
}
```

Τα προγράμματα σε C θα πρέπει να αποθηκεύονται με επέκταση `.c`. Ένα πρόγραμμα σε C, όπως το `my_prog.c`, θα πρέπει να μεταγλωττιστεί προκειμένου να εκτελεστεί. Αν υποθεθεί ότι χρησιμοποιείται ο μεταγλωττιστής `gcc` και η μεταγλώττιση γίνεται με εντολή που πληκτρολογεί ο χρήστης, τότε η μεταγλώττιση του προγράμματος μπορεί να γίνει όπως φαίνεται στο Σχήμα 2.1:

ΣΧΗΜΑ 2.1 Μεταγλώττιση του προγράμματος `my_prog.c`.



Στο Σχήμα 2.1, ο χαρακτήρας `>` είναι η **προτροπή (prompt)** της γραμμής εντολών του λειτουργικού συστήματος. Εκεί δίνεται η εντολή `gcc` η οποία καλεί τον μεταγλωττιστή `gcc`. Μετά την κλήση του μεταγλωττιστή ακολουθεί το όνομα του προγράμματος C που θα μεταγλωττιστεί, στην περίπτωσή μας το `my_prog.c`. Αμέσως μετά ακολουθεί η επιλογή `-o` του `gcc` αμέσως μετά την οποία πρέπει να γραφτεί το όνομα του αρχείου στο οποίο θα τοποθετηθεί το εκτελέσιμο, αν η μεταγλώττιση είναι επιτυχής. Στην περίπτωση της μεταγλώττισης την οποία περιγράφει το Σχήμα 2.1 το εκτελέσιμο ονομάζεται `myprog` (συνήθως στο Linux τα εκτελέσιμα αρχεία δεν έχουν επέκταση, ενώ στα Windows έχουν επέκταση `.exe`).

Αφού δημιουργηθεί το εκτελέσιμο, μπορεί να εκτελεστεί, «χειροκίνητα», ως εξής:

```
>./myprog <enter>
hello world
>
```

Ο παραπάνω τρόπος εκτέλεσης χρησιμοποιείται κατά βάση στο λειτουργικό σύστημα Linux. Το αποτέλεσμα της εκτέλεσης του `myprog` είναι η εμφάνιση της ακολουθίας χαρακτήρων `hello world`. Αμέσως μετά την εκτέλεση του προγράμματος θα εμφανιστεί ξανά η προτροπή του λειτουργικού συστήματος έτοιμη να δεχτεί την επόμενη εντολή.

Στη συνέχεια του βιβλίου, όποτε απαιτείται να παρουσιαστούν τα αποτελέσματα της εκτέλεσης ενός προγράμματος, θα παρουσιάζεται το πρόγραμμα να εκτελείται χειροκίνητα από τη γραμμή εντολών με τον τρόπο που εκτελέστηκε το `myprog`.

Το `myprog.c` χρησιμοποιεί δύο ειδών εντολές:

- Καλεί τη συνάρτηση της βιβλιοθήκης Εισόδου/Εξόδου (E/E) `printf()` για να εμφανίσει την ακολουθία χαρακτήρων `hello world`. (Περισσότερα για τις συναρτήσεις της βιβλιοθήκης Εισόδου/Εξόδου στο Κεφάλαιο 5).
- Επιστρέφει, με την εντολή `return`, τον κωδικό κατάστασης 0 στο περιβάλλον εκτέλεσης του προγράμματος.

Η C απαιτεί στο τέλος κάθε εντολής να υπάρχει ελληνικό ερωτηματικό. Ελληνικό ερωτηματικό δεν απαιτείται στο τέλος των οδηγιών και των **σύνθετων εντολών (compound statements)**. Σύνθετες ονομάζονται οι εντολές που περιέχουν ένα σώμα από μια ή περισσότερες εντολές.

Περισσότερα για τους μεταγλωττιστές και τα βήματα δημιουργίας ενός εκτελέσιμου αρχείου περιγράφονται στο επόμενο κεφάλαιο.

Θα πρέπει να σημειωθεί ότι στις περισσότερες περιπτώσεις η μεταγλώττιση και εκτέλεση προγραμμάτων γίνεται με αυτοματοποιημένο τρόπο, μέσα από **Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης Προγραμμάτων (Integrated Development Environments – IDEs)**, με τα οποία ο χρήστης αλληλεπιδρά μέσα από ένα γραφικό περιβάλλον διεπαφής (Graphical User Interface – GUI).

## Μνήμη και μεταβλητές

Οι περισσότερες από τις συμβατικές μνήμες των υπολογιστικών συστημάτων αποθηκεύουν τα δεδομένα σε δυαδική μορφή. Κατά συνέπεια, ο κώδικας και

τα δεδομένα ενός προγράμματος αποθηκεύονται στην κύρια μνήμη (RAM) ως ακολουθίες από δυαδικά ψηφία ή bits, δηλαδή τιμές 0 ή 1.

Όσον αφορά τα δεδομένα ενός προγράμματος, είναι σαφές ότι δεν μπορεί να είναι όλα του ίδιου τύπου. Μπορεί να απαιτείται να αποθηκευτούν και να επεξεργαστούν χαρακτήρες ή ακολουθίες χαρακτήρων ή αριθμοί, ακέραιοι ή πραγματικοί. Θα πρέπει να είναι σαφές ότι η μετατροπή των δεδομένων από τη μορφή που αντιλαμβάνονται οι χρήστες στη μορφή που αντιλαμβάνεται ένα υπολογιστικό σύστημα απαιτεί κάποιας μορφής **κωδικοποίηση (encoding)**.

Ως ένας γενικός ορισμός, κωδικοποίηση είναι η μετατροπή δεδομένων από ένα σύστημα σε ένα άλλο (π.χ. από το δεκαδικό στο δυαδικό).

Όσο μεγαλύτερο το πεδίο τιμών που επιθυμούμε να καλύψουμε ή/και όσο μεγαλύτερη η **ακρίβεια (precision)** με την οποία επιθυμούμε να αναπαραστήσουμε τις τιμές αυτές, τόσο περισσότερα bits απαιτούνται για την αποθήκευση της ψηφιακής τιμής.

Είναι εύκολο να αποδειχτεί ότι αν διαθέτουμε  $N$  bits, μπορούμε να δημιουργήσουμε  $2^N$  διαφορετικές ακολουθίες 0 και 1 με μέγεθος  $N$  bits, άρα μπορούμε να κωδικοποιήσουμε  $2^N$  διαφορετικές τιμές δεδομένων εισόδου. Η κωδικοποίηση στην περίπτωση αυτή είναι η αντιστοίχιση μιας τιμής δεδομένων εισόδου σε μια ακολουθία από  $N$  bits.

Με βάση τα παραπάνω, αν υπάρχουν διαθέσιμα  $N$  bits για κωδικοποίηση αριθμητικών τιμών μπορεί σε αυτά να αποθηκευτεί οποιοσδήποτε φυσικός αριθμός στο σύνολο  $[0, 2^N - 1]$  ή αντίστοιχα οποιοσδήποτε ακέραιος στο σύνολο  $[-2^{N-1}, 2^{N-1} - 1]$ . Η απόδειξη των παραπάνω είναι ιδιαίτερα απλή και διαισθητική και αφήνεται ως άσκηση στον/ην αναγνώστη/στρια.

## Η θεώρηση της μνήμης για τους προγραμματιστές

Η μνήμη των υπολογιστικών συστημάτων πρέπει να γίνεται αντιληπτή από τους προγραμματιστές ως μια διάταξη από «κουτιά», καθένα από τα οποία μπορεί να χρησιμοποιηθεί για την αποθήκευση 1 bit. Επειδή 1 byte = 8 bits, διευκολύνει τους προγραμματιστές, να σκέφτονται την κύρια μνήμη ως μια συστοιχία από  $N$  συνεχόμενα bytes, ή αλλιώς ένα πίνακα  $N \times 8$ , όπου σε κάθε θέση του πίνακα δύο διαστάσεων είναι αποθηκευμένο ένα bit και σε κάθε γραμμή του είναι αποθηκευμένο 1 byte.

Ένα παράδειγμα μνήμης μεγέθους  $n$  bytes και τα αντίστοιχα δυαδικά δεδομένα που περιέχονται σε ορισμένες από τις θέσεις της φαίνεται στο Σχήμα 2.2. Ο αύξων αριθμός της θέσης στη μνήμη στην οποία αποθηκεύεται ένα byte ονομάζεται **διεύθυνση μνήμης (memory address)**.

ΣΧΗΜΑ 2.2 Διευθύνσεις και περιεχόμενα μνήμης με χωρητικότητα  $n$  bytes.

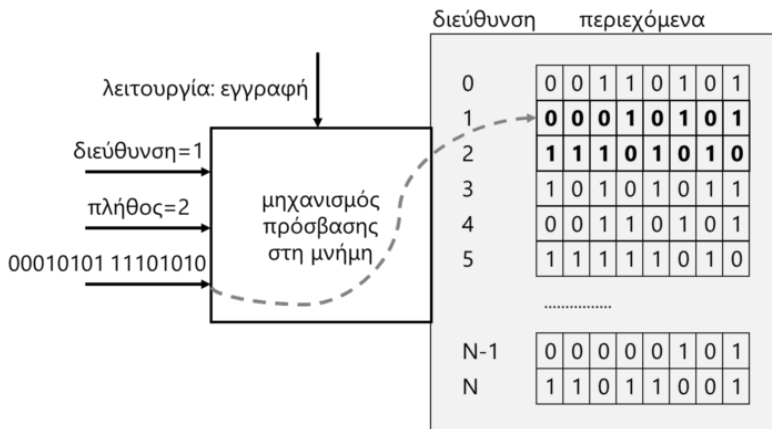
Διεύθυνση	Περιεχόμενα
0	01100001
1	01100011
2	01100010
...	...
$n-1$	01100111

Η μνήμη προσπελάζεται προσδιορίζοντας:

1. Τη διεύθυνση του byte από το οποίο αρχίζει η πρόσβαση.
2. Το πλήθος των bytes στη μνήμη που αφορά η συγκεκριμένη πρόσβαση.

Η πρόσβαση στη μνήμη έχει δύο κύριες λειτουργίες: την **ανάγνωση (read)**, που επιστρέφει τις τιμές των bytes τα οποία έχουν αποθηκευτεί στις θέσεις μνήμης που προσδιορίζονται, και την **εγγραφή (write)**, που καθορίζει τις τιμές των bytes τα οποία πρόκειται να αποθηκευτούν στις θέσεις μνήμης που προσδιορίζονται. Το Σχήμα 2.3 επεξηγεί την πρόσβαση στη μνήμη για εγγραφή.

ΣΧΗΜΑ 2.3 Εγγραφή 2 bytes στις διευθύνσεις μνήμης 1 και 2.



Το μέγεθος της κύριας και της δευτερεύουσας μνήμης μετρείται σε bytes. Ο πίνακας που ακολουθεί παρουσιάζει τις βασικές μονάδες μέτρησης της μνήμης των υπολογιστικών συστημάτων.

ΠΙΝΑΚΑΣ 2.1 Μονάδες μέτρησης μνήμης.

Συμβολισμός	Ερμηνεία
<b>1 KB</b>	1000 bytes = $10^3$ bytes
<b>1 KiB</b>	1024 bytes = $2^{10}$ bytes
<b>1 MB</b>	1000 KB
<b>1 MiB</b>	1024 KiB = $2^{20}$ bytes
<b>1 GB</b>	1000 MB
<b>1 GiB</b>	1024 MiB = $2^{30}$ bytes
<b>1 TB</b>	1000 GB
<b>1 TiB</b>	1024 GiB = $2^{40}$ bytes

### Δεδομένα προγράμματος, μεταβλητές και μνήμη

Η μνήμη ενός υπολογιστικού συστήματος αποθηκεύει τις τιμές των bits που αντιστοιχούν στα δεδομένα του προγράμματος, χωρίς να γνωρίζει τον τύπο τους, δηλαδή το πώς αυτές οι τιμές ερμηνεύονται από το πρόγραμμα. Από την άλλη, για κάθε δεδομένο που χρησιμοποιεί ένα πρόγραμμα, ο προγραμματιστής πρέπει να καθορίζει:

1. τη θέση μνήμης στην οποία θα αποθηκευτεί,
2. το πλήθος των bytes που απαιτούνται για την αποθήκευση του εύρους των τιμών που μπορεί να λάβει, και
3. την κωδικοποίηση που χρησιμοποιείται για την αντιστοίχιση κάθε τιμής που μπορεί να λάβει το δεδομένο αυτό στο δυαδικό σύστημα.

Είναι ιδιαίτερα δύσκολο, αν όχι αδύνατο, για τον προγραμματιστή να θυμάται μια τέτοια αντιστοίχιση, για κάθε δεδομένο που χρησιμοποιεί το πρόγραμμα, κατά τη συγγραφή ή/και ανάγνωση του κώδικα. Είναι προφανές ότι απαιτείται κατάλληλη υποστήριξη από τις γλώσσες προγραμματισμού. Η υποστήριξη την οποία παρέχουν οι γλώσσες είναι ο ορισμός κάποιων βασικών **τύπων δεδομένων (data types)**.

Κάθε βασικός τύπος δεδομένων χρησιμοποιείται για να αποθηκεύει δεδομένα του προγράμματος και έχει:

- ένα εκ των προτέρων καθορισμένο μέγεθος σε bytes, και
- χαρακτηρίζεται από μια συγκεκριμένη δυαδική κωδικοποίηση (και κατά συνέπεια και ένα αντίστοιχο πεδίο τιμών που μπορεί να αποθηκεύσει).

Ο προγραμματιστής μπορεί να ορίσει κάθε δεδομένο του προγράμματος να είναι ένα αντικείμενο ενός τύπου δεδομένων, καθορίζοντας έτσι έμμεσα τόσο το μέγεθος όσο και την κωδικοποίησή του. Για ευκολία, σε κάθε αντικείμενο δεδομένων δίνεται (εκτός από τον τύπο του) και ένα μνημονικό όνομα. Στον κόσμο των γλωσσών προγραμματισμού αυτό το μνημονικό όνομα ονομάζεται **μεταβλητή (variable)**. Το όνομα του αντικειμένου μπορεί να χρησιμοποιείται, αντί της διεύθυνσής του, στις πράξεις ανάγνωσης και αλλαγής των τιμών του.

Όσον αφορά την πραγματική πρόσβαση στη μνήμη, εξακολουθεί να γίνεται με βάση τη θέση μνήμης και το πλήθος των bytes προς ανάγνωση ή αποθήκευση. Η αντιστοίχιση αυτή γίνεται αυτόματα από την εκάστοτε γλώσσα προγραμματισμού.

Εκτός από όνομα και μέγεθος, κάθε μεταβλητή έχει:

- Διάρκεια ζωής: ο χρόνος κατά τον οποίο οι θέσεις μνήμης στις οποίες αποθηκεύεται μια μεταβλητή παραμένουν δεσμευμένες.
- **Εμβέλεια (scope)**: τα σημεία του προγράμματος στα οποία μια μεταβλητή έχει «υπόσταση», άρα μπορεί να χρησιμοποιείται χωρίς να υπάρχει συντακτικό λάθος. Περισσότερα για την εμβέλεια των μεταβλητών αναφέρονται στο Κεφάλαιο 8.

## Οι μεταβλητές στη C

Στη C, όπως στις περισσότερες γλώσσες προγραμματισμού, μεταβλητές ονομάζονται οι οντότητες που χρησιμοποιούν τα προγράμματα κατά τη διάρκεια της εκτέλεσής τους για την προσωρινή αποθήκευση δεδομένων.

Κάθε μεταβλητή πρέπει να έχει έναν τύπο. Ο τύπος της μεταβλητής καθορίζει το μέγεθός της, δηλαδή το πλήθος των συνεχόμενων θέσεων μνήμης που θα καταλάβει αυτή η μεταβλητή κατά την εκτέλεση του προγράμματος.

Η C ορίζει πέντε βασικούς τύπους δεδομένων: τρεις αριθμητικούς, έναν τύπο για αποθήκευση χαρακτήρων και έναν γενικής χρήσης. Το όνομα κάθε τύπου δεδομένων, το είδος των δεδομένων που μπορούν να αποθηκευτούν σε αυτόν καθώς και ο χώρος που καταλαμβάνει στη μνήμη μια μεταβλητή αυτού του τύπου παρουσιάζονται στον Πίνακα 2.2: