2. Τύποι δεδομένων και Τελεστές

Εκπαιδευτικοί στόχοι:

- *Κατανόηση της έννοιας των μεταβλητών και σταθερών.*
- *Τελεστές εκχώρησης και αριθμητικών πράξεων.*
- *Τελεστές αύξησης και μείωσης.*
- *Συγκριτικοί και σχεσιακοί τελεστές.*
- *Λογικοί τελεστές.*
- *Προτεραιότητα τελεστών.*

2.1 Μεταβλητές και σταθερές

Οι *μεταβλητές* και οι *σταθερές* αποτελούν δύο βασικές οντότητες σε κάθε γλώσσα προγραμματισμού. Και οι δύο αντιπροσωπεύουν συμβολικά ονόματα στη μνήμη του υπολογιστή, στα οποία αποθηκεύονται τα δεδομένα (δηλαδή οι τιμές) των προγραμμάτων. Αν και οι μεταβλητές μοιάζουν με τις σταθερές, εντούτοις υπάρχει μια βασική διαφορά μεταξύ τους αναφορικά με τη δυνατότητα

τροποποίησης της τιμής αυτών. Συγκεκριμένα, ενώ σε ένα πρόγραμμα μπορούμε να τροποποιήσουμε μία ή περισσότερες φορές την τιμή μιας μεταβλητής, κάτι τέτοιο δεν είναι εφικτό για τις σταθερές, οι τιμές των οποίων, όπως δηλώνει και το όνομα τους, πρέπει να παραμένουν αμετάβλητες. Σε διαφορετική περίπτωση, όπως θα δούμε στη συνέχεια, εμφανίζεται μήνυμα σφάλματος.

Οι βασικές παράμετροι που χαρακτηρίζουν μια μεταβλητή ή μια σταθερά ενός προγράμματος της ANSI C είναι: το *όνομα*, η *τιμή* και η *διεύθυνση* αυτών στη μνήμη του υπολογιστή (γνωστή και ως *αναφορά*, όπως θα μάθουμε σε επόμενο κεφάλαιο). Αναφορικά με την επιλογή του ονόματος μιας μεταβλητής ή σταθεράς, αυτό θα πρέπει να είναι μοναδικό σε κάθε πρόγραμμα. Ως εκ τούτου, δεν επιτρέπεται να χρησιμοποιούμε το ίδιο όνομα για μεταβλητές ή σταθερές ακόμη και αν αυτό συμβαίνει σε διαφορετικά πηγαία αρχεία που ανήκουν όμως στο ίδιο πρόγραμμα. Επιπλέον, τα υποστηριζόμενα ονόματα τόσο των μεταβλητών, όσο και των σταθερών ενός προγράμματος μπορεί να αποτελούνται από συνδυασμούς πεζών και κεφαλαίων γραμμάτων του λατινικού αλφαβήτου (a-z, A-Z), αριθμητικά ψηφία (0-9), καθώς επίσης και τον χαρακτήρα "_". Σε κάθε περίπτωση πάντως, ο πρώτος χαρακτήρας των ονομάτων θα πρέπει υποχρεωτικά να μην είναι αριθμός. Στο σημείο αυτό αξίζει να τονίσουμε πως η γλώσσα ANSI C κάνει διάκριση μεταξύ πεζών και κεφαλαίων χαρακτήρων. Συνεπώς, τα ακόλουθα ονόματα μεταβλητών ή σταθερών είναι διαφορετικά μεταξύ τους, και επομένως μπορούν να συνυπάρχουν στον ίδιο κώδικα.

- **`Counter`**
- **`counter`**
- **`CoUnTeR`**
- **`COUNTer`**
- **`COUNTER`**

> Αν και δεν υπάρχει περιορισμός στο πλήθος χαρακτήρων από τους οποίους μπορεί να αποτελείται το όνομα μιας μεταβλητής ή μιας σταθεράς, εντούτοις συνιστάται αυτό να μην ξεπερνά τους 31 χαρακτήρες, διότι μπορεί να δημιουργήσει σφάλματα σε ορισμένους μεταγλωττιστές.

Μιας και οι μεταβλητές/σταθερές θα χρησιμοποιούνται κατά κόρον στα προγράμματά μας, είναι χρήσιμο να ακολουθούμε ορισμένες "καλές πρακτικές" για την επιλογή των ονομάτων αυτών, έτσι ώστε να έχουμε καλογραμμένους και ευανάγνωστους κώδικες. Προς την κατεύθυνση αυτή, συνιστάται τα ονόματα τόσο των σταθερών, όσο και μεταβλητών, να είναι περιγραφικά και να αντιστοιχούν στην πληροφορία που θα αποθηκευτεί σε αυτά. Για παράδειγμα, στην περίπτωση κατά την οποία θέλουμε να χρησιμοποιήσουμε μια μεταβλητή για την αποθήκευση του διαστήματος που διένυσε ένα όχημα κινούμενο στον αυτοκινητόδρομο, είναι προτιμότερο να επιλέξουμε ως όνομα αυτής το **`distance`** παρά

κάποιο από τα **diametros**, **i**, **test**, κτλ. Τέλος, θα πρέπει να θυμόμαστε πως τα ονόματα των μεταβλητών και των σταθερών δε θα πρέπει να ταυτίζονται με κάποια από τις δεσμευμένες λέξεις της ANSI C, όπως αυτές έχουν παρουσιαστεί στον Πίνακα 1.1.

Ερώτηση 2.1 Ποια από τα επόμενα ονόματα μεταβλητών είναι έγκυρα;

1. **counter**
2. **_result**
3. **my-variable**
4. **01value**
5. **value02**
6. **flag!**
7. **$home;**

2.1.1 Δήλωση μεταβλητών

Πριν χρησιμοποιήσουμε μια μεταβλητή στον κώδικα ενός προγράμματος, αυτή θα πρέπει να έχει δηλωθεί. Σύμφωνα με το πρότυπο της ANSI C, οι δηλώσεις των μεταβλητών πραγματοποιούνται στην αρχή της εκάστοτε συνάρτησης (π.χ. στην αρχή της συνάρτησης **main()**) και οπωσδήποτε πριν οι μεταβλητές αυτές χρησιμοποιηθούν στα πλαίσια του προγράμματος για την αποθήκευση δεδομένων. Στη συνέχεια, παρουσιάζεται το συντακτικό δήλωσης μιας μεταβλητής, στο οποίο ο *τύπος_δεδομένων* καθορίζει το είδος της πληροφορίας που θα περιέχει η εκάστοτε μεταβλητή. Αξίζει να παρατηρήσουμε πως στο τέλος των δηλώσεων υπάρχει το ελληνικό ερωτηματικό, το οποίο σηματοδοτεί και την ολοκλήρωση αυτής.

τύπος_δεδομένων *όνομα_μεταβλητής*;

Στην περίπτωση που θέλουμε να δηλώσουμε περισσότερες από μία μεταβλητές ιδίου τύπου, αυτό μπορεί να γίνει είτε επαναλαμβάνοντας πολλαπλές φορές την προηγούμενη εντολή (με διαφορετικό όνομα μεταβλητής κάθε φορά) ή να τροποποιήσουμε ελαφρά το συντακτικό αυτής, όπως φαίνεται στη συνέχεια. Σύμφωνα με τον δεύτερο τρόπο, τα ονόματα των μεταβλητών δηλώνονται στην ίδια γραμμή και διαχωρίζονται μεταξύ τους με το σύμβολο ”**,**”. Αυτό που θα πρέπει να προσέξουμε στο συγκεκριμένο τρόπο δήλωσης αφορά το γεγονός ότι όλες οι μεταβλητές που ορίζονται κατά τη συγκεκριμένη δήλωση θα πρέπει να αντιστοιχούν στον ίδιο τύπο δεδομένων (θα μάθουμε περισσότερα για τους

Πίνακας 2.1: Υποστηριζόμενοι τύποι δεδομένων στη γλώσσα ANSI C αναφορικά με ένα τυπικό 32-bit σύστημα.

Τύπος δεδομένων	Πλήθος bit	Υποστηριζόμενο εύρος τιμών
char **signed char**	8	-128 έως 127
unsigned char	8	0 έως 255
short int	16	-32,768 έως 32,767
unsigned short int	16	0 έως 65,535
int **long int**	32	-2,147,483,648 έως 2,147,483,647
unsigned int **unsigned long int**	32	0 έως 4,294,967,295
float	32	1.2e-38 έως 3.4e+38
double	64	1.7e-308 έως 1.7e+308
long double	80	3.4e-4932 έως 1.1e+4932

τύπους δεδομένων στη συνέχεια της τρέχουσας ενότητας).

τύπος_δεδομένων *μεταβλητή1*, *μεταβλητή2*, *μεταβλητή3*, ...;

Η γλώσσα προγραμματισμού ANSI C υποστηρίζει μια πληθώρα τύπων δεδομένων, οι κυριότεροι εκ των οποίων αναφορικά με ένα 32-bit σύστημα συνοψίζονται στον Πίνακα 2.1. Στον ίδιο πίνακα παρουσιάζεται επίσης ο χώρος (αριθμός από bit) που καταλαμβάνει ο καθένας εξ αυτών στη μνήμη του υπολογιστή, καθώς επίσης και το εύρος τιμών που μπορεί να αποθηκεύσει. Παρατηρήστε πως καθώς μεταβαίνουμε σε τύπους δεδομένων που απαιτούν περισσότερα δυαδικά ψηφία (bit) για την αποθήκευση της πληροφορίας, μπορούμε να χειριστούμε αριθμούς με ολοένα και μεγαλύτερα εύρη τιμών. Μη σας προβληματίζει η πληθώρα διαφορετικών τύπων δεδομένων που παρουσιάζονται στον συγκεκριμένο πίνακα. Στην πλειονότητα των περιπτώσεων, καθώς και στις ασκήσεις του βιβλίου, οι συνηθέστερα χρησιμοποιούμενοι τύποι δεδομένων είναι οι **char**, **int**, **float** και **double**.

> Αν και ενδέχεται να υπάρχουν διαφοροποιήσεις ως προς το εύρος τιμών που υποστηρίζει ο κάθε τύπος δεδομένων (π.χ. ανάλογα με τον μεταγλωττιστή ή το λειτουργικό σύστημα), το πρότυπο της ANSI C καθορίζει ότι το μέγεθος του τύπου **long** θα πρέπει να είναι μεγαλύτερο από το αντίστοιχο του τύπου **int**, το οποίο με τη σειρά του θα πρέπει να είναι μεγαλύτερο από αυτό του **short**.

Ερώτηση 2.2 Δηλώστε μια μεταβλητή προκειμένου να αποθηκεύσουμε σε αυτή την τιμή της σταθεράς π=3.14159265358979323846.

Το πλήθος των bit που καταλαμβάνει μια μεταβλητή αντιστοιχεί στο πλήθος των δυαδικών ψηφίων που χρησιμοποιούνται για την αναπαράσταση ενός δεδομένου (αριθμού ή αλφαριθμητικού) στο δυαδικό σύστημα αρίθμησης. Ο επόμενος πίνακας παρουσιάζει τις αντιστοιχίες μεταξύ των συνηθέστερα χρησιμοποιούμενων μονάδων για την αποθήκευση δεδομένων. Θα πρέπει να δείξουμε ιδιαίτερη προσοχή στο γεγονός ότι τα σύμβολα *K*, *M*, *G* και *T* αντιστοιχούν σε πολλαπλάσια του 1024 (σε αντίθεση με το δεκαδικό σύστημα όπου τα συγκεκριμένα σύμβολα δηλώνουν πολλαπλάσια του 1000).

1 byte	→	8 bit
1 Kbyte	→	2^{10} = 1024 bytes
1 Mbyte	→	2^{10} = 1024 Kbytes
1 Gbyte	→	2^{10} = 1024 Mbytes
1 Tbyte	→	2^{10} = 1024 Gbytes

Γιατί όμως υπάρχει ανάγκη για τόσους πολλούς τύπους δεδομένων; Δε θα μπορούσαμε να κάνουμε όλους τους υπολογισμούς χρησιμοποιώντας μόνο μεταβλητές τύπου **`double`**, μιας και αυτός ο τύπος δεδομένων μπορεί να αποθηκεύσει το μεγαλύτερο εύρος τιμών; Η απάντηση στο συγκεκριμένο ερώτημα έχει ήδη δοθεί μέσω του Πίνακα 2.1, στον οποίο παρουσιάζεται το απαιτούμενο πλήθος δυαδικών ψηφίων για κάθε έναν από τους υποστηριζόμενους τύπους δεδομένων. Όπως έχουμε ήδη αναφέρει, το πλήθος των bit αντιστοιχεί στον χώρο που καταλαμβάνει μια μεταβλητή τέτοιου τύπου στη μνήμη του υπολογιστή. Έτσι, το προηγούμενο ερώτημα μπορεί να αναδιατυπωθεί ως εξής: *Ποιος είναι ο βέλτιστος τύπος δεδομένων για κάθε μεταβλητή του προγράμματος;* Η απάντηση στην ερώτηση αυτή δίνεται συνήθως από τον προγραμματιστή λαμβάνοντας υπόψιν τις απαιτήσεις που θέτει ο ίδιος ο αλγόριθμος. Αυτό που θα πρέπει να γνωρίζουμε όμως είναι πως δεν υπάρχει ένας τύπος δεδομένων, ο οποίος μπορεί να χρησιμοποιηθεί (με βέλτιστο τρόπο) καθολικά για όλες τις μεταβλητές ενός προγράμματος.

Ακολούθως, παρατίθενται ορισμένα χαρακτηριστικά παραδείγματα επιλογής *βέλτιστου* τύπου δεδομένων, προκειμένου να καταστεί σαφής ο τρόπος τον

τρόπο με τον οποίο λαμβάνει τις αποφάσεις του ο προγραμματιστής. Έστω πως θέλουμε να αποθηκευτεί στη μεταβλητή με όνομα **temperature** η θερμοκρασία ενός δωματίου. Πιστεύετε πως θα μπορούσαμε να ορίσουμε τη συγκεκριμένη μεταβλητή ως **short int**, **unsigned short int** ή **int**; Η απάντηση είναι όχι, διότι οι μεταβλητές αυτές θα υποστήριζαν μόνο ακέραιες τιμές. Τι γίνεται όμως στην περίπτωση που η θερμοκρασία πρέπει να αποθηκευτεί με ακρίβεια δεκαδικού ψηφίου; Επιπλέον, η τιμή της θερμοκρασίας θα μπορούσε να έχει και αρνητικές τιμές. Με βάση όσα έχουν ήδη αναφερθεί, συμπεραίνουμε πως για την αποθήκευση των τιμών θερμοκρασίας είναι προτιμότερο να χρησιμοποιήσουμε έναν από τους τύπους **float**, **double** και **long double**, οι οποίοι υποστηρίζουν θετικούς και αρνητικούς πραγματικούς αριθμούς. Ποιος όμως από αυτούς είναι καταλληλότερος για το συγκεκριμένο πρόβλημα; Για να απαντήσουμε την ερώτηση θα πρέπει να συμβουλευτούμε πάλι τον Πίνακα 2.1 λαμβάνοντας υπόψιν, όχι μόνο το ζητούμενο εύρος τιμών, αλλά και τον χώρο (δηλαδή το πλήθος των bit) που καταλαμβάνει καθεμιά από αυτές στη μνήμη του υπολογιστή. Με βάση τα περιεχόμενα του πίνακα συμπεραίνουμε πως οι μεταβλητές τύπου **float** απαιτούν 32 bit (ή 4 bytes), ενώ οι μεταβλητές τύπου **double** και **long double** θέλουν 64 και 80 bit, αντίστοιχα. Από τη στιγμή που και οι δύο τύποι δεδομένων μπορούν να αποθηκεύσουν το ζητούμενο εύρος τιμών, αυτός που τελικά επιλέγεται είναι εκείνος με τις μικρότερες απαιτήσεις σε πόρους μνήμης. Κατά συνέπεια, αναφορικά με το συγκεκριμένο παράδειγμα, ο αποτελεσματικότερος τύπος δεδομένων για την αποθήκευση των τιμών θερμοκρασίας είναι ο **float**, ενώ η σχετική δήλωση της μεταβλητής **temperature** πραγματοποιείται, ως ακολούθως:

Απόσπασμα Κώδικα 2.1

```
1 // δήλωση πραγματικής μεταβλητής με όνομα temperature
2 float temperature;
```

> Η επιλογή του σωστού τύπου δεδομένων αποτελεί μια διαδικασία στην οποία θα πρέπει να δίνουμε ιδιαίτερη προσοχή κατά την ανάπτυξη των προγραμμάτων μας, προκειμένου να διασφαλίζεται αφενός μεν η ορθή λειτουργία του προγράμματος για όλες τις αποδεκτές τιμές εισόδου που εισάγει ο χρήστης, αφετέρου δε να απαιτεί την ελάχιστη δυνατή μνήμη του υπολογιστικού συστήματος.

Όμοια, αν σε κάποιο πρόγραμμα απαιτείται η δήλωση μιας μεταβλητής προκειμένου να αποθηκεύουμε το πλήθος των θεατών ενός ποδοσφαιρικού αγώνα, τότε αυτό θα μπορούσε να γίνει ως εξής:

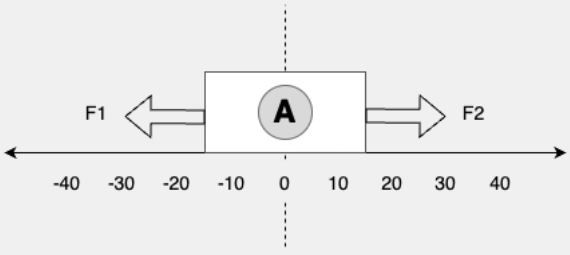
Απόσπασμα Κώδικα 2.2

```
1 // δήλωση ακέραιας μεταβλητής με όνομα number_people
2 int number_people;
```

Ερώτηση 2.3 Θα μπορούσαμε να επιλέξουμε κάποιον άλλο τύπο δεδομένων για τη μεταβλητή `number_people`; Αν ναι, ποιον (να αιτιολογήσετε την απάντησή σας); Υπάρχουν περιορισμοί για τη συγκεκριμένη επιλογή;

Ερώτηση 2.4 Δώστε τη δήλωση μιας μεταβλητής η οποία μπορεί να χρησιμοποιηθεί για την αποθήκευση του αριθμού που προκύπτει από τη ρίψη ενός ζαριού. Να αιτιολογήσετε την απάντησή σας για την επιλογή του συγκεκριμένου τύπου δεδομένων.

Ερώτηση 2.5 Έχετε στη διάθεσή σας ένα σώμα στο οποίο ασκούνται δύο αντίρροπες δυνάμεις *F1* και *F2*, όπως φαίνεται στο ακόλουθο σχήμα. Ανάλογα με τα μέτρα των συγκεκριμένων δυνάμεων, το κέντρο βάρους του σώματος μετακινείται πάνω στο επίπεδο ως προς το σημείο αναφοράς κατά μία τιμή που ανήκει στο διάστημα [-40, 40] μέτρα. Γνωρίζοντας πως η μετατόπιση του σώματος μπορεί να λάβει μόνο ακέραιες τιμές, ζητείται να δηλώσετε μια μεταβλητή κατάλληλου τύπου για την αποθήκευση της μετατόπισης του σώματος.



Ερώτηση 2.6 Ποιος θα ήταν ο καταλληλότερος τύπος δεδομένων για τη μεταβλητή που αποθηκεύει τη μετατόπιση του σώματος στην Ερώτηση 2.5, αν αυτή αντιστοιχούσε σε πραγματικό αριθμό;

Ερώτηση 2.7 Σε ένα πρόγραμμα απαιτείται η αποθήκευση του τηλεφωνικού αριθμού 4 ατόμων. Δώστε τη δήλωση των σχετικών μεταβλητών αιτιολογώντας την επιλογή του τύπου δεδομένων αυτών.

Ερώτηση 2.8 Δίνεται πως σε ένα πρόγραμμα χρησιμοποιείται μια μεταβλητή τύπου **`float`**. Δώστε τις προϋποθέσεις που πρέπει να ισχύουν προκειμένου να αντικαταστήσουμε τον συγκεκριμένο τύπο δεδομένων με αυτόν του **`int`**. Να αιτιολογήσετε την απάντησή σας.

Ερώτηση 2.9 Μια γραμμή παραγωγής αποτελείται από 18 διακριτά στάδια, σε καθένα από τα οποία ο μηχανικός επιτελεί κάποια εργασία. Δηλώστε μια μεταβλητή (με όνομα της επιλογής σας) προκειμένου να αποθηκεύουμε σ' αυτή το στάδιο που βρίσκεται κάθε φορά ο μηχανικός. Να αιτιολογήσετε την απάντησή σας για την επιλογή του συγκεκριμένου τύπου δεδομένων.

2.1.2 Εκχώρηση τιμών σε μεταβλητές

Όπως έχουμε ήδη αναφέρει, οι μεταβλητές χρησιμοποιούνται για την προσωρινή αποθήκευση δεδομένων (π.χ. τιμή θερμοκρασίας, πλήθος ατόμων, απόσταση σημείων, κτλ.) και η τιμή τους μπορεί να τροποποιηθεί όσες φορές απαιτείται κατά την εκτέλεση του προγράμματος. Προκειμένου να πραγματοποιηθεί εκχώρηση τιμής σε μια μεταβλητή χρησιμοποιείται ο τελεστής **=**, στην αριστερή μεριά του οποίου βρίσκεται το όνομα της μεταβλητής, ενώ δεξιά της ισότητας υπάρχει η τιμή προς εκχώρηση. Τέλος, σύμφωνα με το πρότυπο της γλώσσας ANSI C, η εκχώρηση τιμών σε μια μεταβλητή μπορεί να γίνει είτε κατά τη δήλωση της μεταβλητής, ή ακόμη και αργότερα.

Στα επόμενα παραδείγματα μπορούμε να δούμε τον τρόπο δήλωσης μιας μεταβλητής τύπου **`float`** με όνομα **`temperature`** και την εκχώρηση της τιμής 24.3 σ' αυτή. Συγκεκριμένα, στο πρώτο παράδειγμα, η ανάθεση τιμής πραγματοποιείται κατά τη δήλωση της μεταβλητής **`temperature`**, ενώ στη δεύτερη περίπτωση αυτό πραγματοποιείται στο κυρίως πρόγραμμα. Αξίζει να σημειωθεί, πως στην πλειονότητα των περιπτώσεων εφαρμόζεται ένας υβριδικός τρόπος εκχώρησης τιμών, σύμφωνα με τον οποίο κατά τη δήλωση των μεταβλητών πραγματοποιείται η αρχικοποίηση αυτών (δηλαδή τους ανατίθεται μια αρχική τιμή), ενώ στη συνέχεια του προγράμματος η τιμή τους μπορεί να τροποποιηθεί μία ή περισσότερες φορές.

Απόσπασμα Κώδικα 2.3

```
1 #include <stdio.h>
2 int main()
3 {
4 float temperature = 24.3;  // δήλωση και αρχικοποίηση μεταβλητής
5
6 // ακολουθούν οι υπόλοιπες εντολές του προγράμματος
```

```
7
8 return 0;
9 }
```

Απόσπασμα Κώδικα 2.4

```
1 #include <stdio.h>
2 int main()
3 {
4 float temperature;  // δήλωση μεταβλητής χωρίς αρχικοποίηση
5
6 // εντολές του προγράμματος
7
8 temperature = 24.3;  // εκχώρηση τιμής στη μεταβλητή
9
10 // ακολουθούν οι υπόλοιπες εντολές του προγράμματος
11
12 return 0;
13 }
```

> Σύμφωνα με το πρότυπο της ANSI C, η δήλωση των μεταβλητών πρέπει να πραγματοποιείται στην αρχή των συναρτήσεων (π.χ. στην αρχή της συνάρτησης **`main()`**), ενώ η τιμή αυτών μπορεί να τροποποιείται (μία ή περισσότερες φορές) κατά την εκτέλεση του προγράμματος.

Ας δούμε ένα ακόμη παράδειγμα στο οποίο η αρχική τιμή της μεταβλητής **`temperature`** μεταβάλλεται από το 24.3 σε 22.7, και εν συνεχεία στην τιμή 18.5. Επομένως, η τελική τιμή της μεταβλητής **`temperature`** κατά την ολοκλήρωση του προγράμματος είναι 18.5. Μέσω αυτού θα κατανοήσετε τη σειριακή ροή εκτέλεσης ενός προγράμματος γραμμένου στην ANSI C, σύμφωνα με την οποία η σειρά εμφάνισης των εντολών στον κώδικα καθορίζει και τη σειρά με την οποία αυτές εκτελούνται.

Απόσπασμα Κώδικα 2.5

```
1 #include <stdio.h>
2
3 int main()
4 {
5 // δήλωση μεταβλητής και αρχικοποίηση αυτής
6 float temperature = 24.3;
7
8 // εμφάνιση της τιμής για τη μεταβλητή temperature στην οθόνη
```

```
 9 printf("Η θερμοκρασία είναι = %f\n", temperature);
10
11 // εκχώρηση νέας τιμής στη μεταβλητή και εμφάνιση αυτής στην οθόνη
12 temperature = 22.7;
13 printf("Η θερμοκρασία είναι = %f\n", temperature);
14
15 // εκχώρηση τελικής τιμής στη μεταβλητή και εμφάνιση στην οθόνη
16 temperature = 18.5;
17 printf("Η θερμοκρασία είναι = %f\n", temperature);
18
19 return 0;
20 }
```

Ερώτηση 2.10 Ποιος είναι ο τελεστής ανάθεσης στη γλώσσα ANSI C; Περιγράψτε συνοπτικά τη λειτουργία του.

Για την εκχώρηση πραγματικών αριθμητικών, οι οποίοι διαθέτουν ακέραιο και δεκαδικό μέρος, σε μεταβλητές κατάλληλου τύπου (π.χ. **`float`** ή **`double`**) χρησιμοποιείται το σύμβολο της τελείας "**`.`**"'.

Όπως μάθαμε στο προηγούμενο κεφάλαιο, ορισμένα είδη σφαλμάτων (π.χ. λογικά σφάλματα) δεν ανιχνεύονται από τον μεταγλωττιστή. Ας δούμε μια τέτοια περίπτωση με στόχο να κατανοήσουμε τη συσχέτιση που υπάρχει μεταξύ μιας λανθασμένης επιλογής τύπου δεδομένων αναφορικά για μια μεταβλητή και τη συγκεκριμένη κατηγορία σφαλμάτων. Για το σκοπό αυτό θεωρήστε τον κώδικα του Παραδείγματος 2.6, στον οποίο μία από τις δύο τιμές που αναθέτουμε στις μεταβλητές τύπου **`short`** είναι εκτός του υποστηριζόμενου εύρους τιμών (σύμφωνα με τον Πίνακα 2.1). Έστω λοιπόν ότι επιχειρούμε να εκχωρήσουμε στις μεταβλητές με ονόματα **`value1`** και **`value2`**, τις τιμές 24000 και 45000, αντίστοιχα. Όπως θα μάθουμε στο επόμενο κεφάλαιο του βιβλίου μας, η συνάρτηση **`printf()`** που χρησιμοποιείται στο παράδειγμα αυτό, εμφανίζει στην οθόνη του υπολογιστή το περιεχόμενο της εκάστοτε μεταβλητής που περιέχεται εντός παρενθέσεων. Συνεπώς, η πρώτη κλήση της συνάρτησης **`printf()`** θα οδηγήσει στο να εμφανιστεί στην οθόνη του υπολογιστή το μήνυμα *"Η τιμή της μεταβλητής value1 = ..."*, όπου οι τελείες (...) αντιπροσωπεύουν την τιμή που περιέχει η μεταβλητή **`value1`** στη γραμμή 16 του κώδικα.

Απόσπασμα Κώδικα 2.6

```
1 #include <stdio.h>
2 int main()
3 {
4 // δήλωση μεταβλητών
```

```
 5 short int value1, value2;
 6
 7 // Εκχώρηση επιτρεπτής τιμής στη μεταβλητή value1.
 8 // Η τιμή είναι εντός ορίων.
 9 value1 = 24000;
10
11 // Εκχώρηση μη επιτρεπτής τιμής στη μεταβλητή value2.
12 // Η τιμή είναι εκτός ορίων και προκύπτει λογικό σφάλμα.
13 value2 = 45000;
14
15 printf("Η τιμή της μεταβλητής value1 = %d\n", value1);
16 printf("Η τιμή της μεταβλητής value2 = %d\n", value2);
17 return 0;
18 }
```

Αν δοκιμάσουμε να τρέξουμε τον συγκεκριμένο κώδικα, θα διαπιστώσουμε πως δε λειτουργεί έτσι όπως θα περιμέναμε με την πρώτη ματιά. Πιο αναλυτικά, ενώ η τιμή που εμφανίζεται στην οθόνη για τη μεταβλητή **`value1`** είναι η σωστή, κάτι ανάλογο δεν ισχύει για τη μεταβλητή **`value2`**. Αναλυτικότερα, η τιμή της δεύτερης μεταβλητής του προγράμματος είναι το -20536, και όχι αυτή που αναθέσαμε εμείς στον κώδικα (45000). Γιατί όμως συμβαίνει κάτι τέτοιο; Η απάντηση του ερωτήματος προκύπτει αν ανατρέξουμε στο εύρος τιμών που μπορεί να αναπαραστήσει ο τύπος δεδομένων **`short int`**. Όπως έχουμε ήδη αναλύσει με τη βοήθεια του Πίνακα 2.1, το υποστηριζόμενο εύρος τιμών για τις μεταβλητές αυτού του τύπου είναι μεταξύ -32768 έως 32767. Στην περίπτωση που η τιμή προς αποθήκευση (στο συγκεκριμένο παράδειγμα το 45000) υπερβαίνει το άνω όριο του επιλεχθέντος τύπου δεδομένων, τότε η διαφορά που προκύπτει από τη μέγιστη επιτρεπτή τιμή μειωμένη κατά 1, προστίθεται στην ελάχιστη επιτρεπτή τιμή του συγκεκριμένου τύπου δεδομένων. Στο παράδειγμα που μελετάμε, εφόσον ο αριθμός 45000 υπερβαίνει τη μέγιστη τιμή 32767, η διαφορά αυτών μειωμένη κατά 1 (δηλαδή το 45000-32767-1=12233) προστίθεται στην ελάχιστη τιμή που υποστηρίζει ο τύπος δεδομένων **`short int`**, προκειμένου να προκύψει η τιμή 12233 + (-32768) = -20536 που εμφανίζεται στην οθόνη του υπολογιστή.

> Για τη σωστή λειτουργία ενός προγράμματος θα πρέπει να αποφεύγονται λάθη στον κώδικα που σχετίζονται με την παραβίαση του εύρους τιμών τις οποίες μπορούμε να αποθηκεύσουμε στον επιλεχθέντα τύπο δεδομένων. Με άλλα λόγια, ο προγραμματιστής πρέπει να εγγυάται πως οι τιμές που αποθηκεύονται σε όλες τις μεταβλητές και σταθερές ενός προγράμματος είναι εντός των επιτρεπτών ορίων, διότι διαφορετικά οδηγούμαστε σε λανθασμένα αποτελέσματα (λογικό σφάλμα), για τα οποία ο μεταγλωττιστής δεν εμφανίζει κάποιο σχετικό μήνυμα ενημέρωσης.

Ερώτηση 2.11 Μπορείτε να υπολογίσετε θεωρητικά τις τιμές που θα περιέχουν οι ακόλουθες μεταβλητές;

1. `short int counter = 73000;`
2. `char value = 85;`
3. `unsigned char houses = 300;`

Ερώτηση 2.12 Τι θα συμβεί αν εκχωρήσουμε ως τιμές στις μεταβλητές **`value1`** και **`value2`** του Αποσπάσματος Κώδικα 2.6 τα 38532 και 102342, αντίστοιχα; Μπορείτε να υπολογίσετε θεωρητικά τα αναμενόμενα αποτελέσματα που θα εμφανιστούν στην οθόνη; Επαληθεύστε την απάντησή σας με τη βοήθεια του υπολογιστή τροποποιώντας κατάλληλα τον κώδικα του παραδείγματος.

Ερώτηση 2.13 Τι θα συμβεί αν εκχωρήσουμε ως τιμές στις μεταβλητές **`value1`** και **`value2`** στο Απόσπασμα Κώδικα 2.6 τα -38532 και -102342, αντίστοιχα; Μπορείτε να υπολογίσετε θεωρητικά τα αναμενόμενα αποτελέσματα που θα εμφανιστούν στην οθόνη; Επαληθεύστε την απάντησή σας με τη βοήθεια του υπολογιστή τροποποιώντας κατάλληλα τον κώδικα του παραδείγματος.
Υπόδειξη: Ακολουθήστε την αντίστροφη διαδικασία από αυτή που παρουσιάστηκε στην επεξήγηση του Αποσπάσματος Κώδικα 2.6.

Θα πρέπει να σημειωθεί πως η γλώσσα ANSI C επιτρέπει την εκχώρηση τιμών στις μεταβλητές αριθμητικού τύπου **`int`**, **`float`**, **`long`**, **`double`**, καθώς και στις αντίστοιχες **`unsigned`** μορφές αυτών, χρησιμοποιώντας την επιστημονική σημειογραφία του **`e`** ή **`E`**. Ακολούθως παρουσιάζονται ορισμένα ενδεικτικά παραδείγματα με τη χρήση της επιστημονικής σημειογραφίας.

Απόσπασμα Κώδικα 2.7

```
1 // Η τιμή της μεταβλητής a ισούται με 5*10^3
2 int a = 5e3;
3
4 // Η τιμή της μεταβλητής b ισούται με 2*10^19
5 double b = 2e19;
6
7 // Η τιμή της μεταβλητής c ισούται με 8.5*10^-6
8 float c = 8.5e-6;
```

Τέλος, στην περίπτωση που επιθυμούμε να αρχικοποιήσουμε πολλαπλές τιμές

μεταβλητών ταυτόχρονα με τη δήλωση αυτών, κάτι τέτοιο μπορεί να πραγματοποιηθεί ως εξής:

τύπος_δεδομένων *μεταβλητή1=τιμή1*, *μεταβλητή2=τιμή2*, ..., *μεταβλητή*$_N$*=τιμή*$_N$**;**

Θα πρέπει να θυμόμαστε πως η δήλωση, και κατά συνέπεια η αρχικοποίηση, πολλαπλών μεταβλητών σε μία εντολή προϋποθέτει ότι όλες είναι του ιδίου τύπου.

Για λόγους εξάσκησης, καθεμιά από τις επόμενες εντολές ορίζει και αναθέτει αρχική τιμή σε τέσσερις μεταβλητές τύπου **`int`**, **`float`** και **`double`** αντίστοιχα.

Απόσπασμα Κώδικα 2.8

```
1 int a=4, b=15, c=−1, e=0;
2 float q=2.3, w=−9.3e−7, y=−9, r=3.5e12;
3 double z=1.2e14, x=3e−29, m=−100.3e49, p=0.5e22;
```

2.1.3 Χρήση της τιμής μιας μεταβλητής

Έπειτα από τη δήλωση και εκχώρηση τιμής σε μια μεταβλητή, αυτή μπορεί να χρησιμοποιηθεί σε κάποιο πρόγραμμα στα πλαίσια υπολογισμών. Προκειμένου να κατανοήσουμε καλύτερα τον ρόλο που έχουν οι μεταβλητές, στο Παράδειγμα 2.1 αναθέτουμε την τιμή 3.14 στην πραγματική μεταβλητή με όνομα **`pi`** και εν συνεχεία, δηλώνουμε πως η τιμή της ακτίνας του κύκλου, η οποία με τη σειρά της αποθηκεύεται στη μεταβλητή πραγματικού τύπου με όνομα **`radius`**, ισούται με το 8.5. Προσέξτε ότι για τις δύο αυτές μεταβλητές πραγματοποιείται δήλωση και ανάθεση τιμής στις γραμμές 7–8 και 11–12, αντίστοιχα. Ακολούθως, χρησιμοποιούμε τις τρέχουσες τιμές των μεταβλητών **`pi`** και **`radius`**, προκειμένου να υπολογίσουμε την περίμετρο και το εμβαδόν του συγκεκριμένου κύκλου. Πού όμως θα αποθηκευτούν τα αποτελέσματα αυτών; Όπως μπορούμε να δούμε στον κώδικα του Παραδείγματος 2.1, η αποθήκευση πραγματοποιείται σε δύο νέες μεταβλητές με ονόματα **`perimetros`** και **`embado`**, αντίστοιχα. Θυμηθείτε ότι κάθε φορά που χρειαζόμαστε στον κώδικα ενός προγράμματος να χρησιμοποιήσουμε μια νέα μεταβλητή, αυτή θα πρέπει προηγουμένως να έχει οριστεί στην αρχή της εκάστοτε συνάρτησης – στο συγκεκριμένο παράδειγμα η δήλωση των νέων μεταβλητών πραγματοποιείται στη γραμμή 8 της κύριας συνάρτησης **`main()`**. Επιπλέον, όπως έχουμε μάθει, η εκχώρηση τιμών σε μεταβλητές πραγματοποιείται με τη βοήθεια του τελεστή "**`=`**", όπου στο δεξί μέρος της ισότητας βρίσκεται η τιμή προς ανάθεση και στο αριστερό μέρος αυτής η μεταβλητή στην οποία θα αποθηκευτεί το αποτέλεσμα. Τέλος, όμοια με τα προηγούμενα πα-

ραδείγματα, έτσι και εδώ, η συνάρτηση **`printf()`** εμφανίζει στην οθόνη του υπολογιστή τις τιμές των μεταβλητών που περιέχονται στο εσωτερικό των παρενθέσεων (δηλαδή τις τιμές που έχει η περίμετρος και το εμβαδόν του κύκλου).

Λυμένο Παράδειγμα 2.1 Γράψτε ένα πρόγραμμα στο οποίο, αφού δηλώσετε και αποθηκεύσετε σε μια μεταβλητή κατάλληλου τύπου την ακτίνα ενός κύκλου, το πρόγραμμα να υπολογίζει την περίμετρο και το εμβαδόν αυτού, και εν συνεχεία να εμφανίζει τα αποτελέσματα με τη χρήση κατάλληλων μηνυμάτων στην οθόνη του υπολογιστή.

Ενδεικτική Λύση:

```
1  #include <stdio.h>
2  int main()
3  {
4  // δήλωση μεταβλητών
5  int radius;
6  float pi, perimetros, embado;
7
8  // εκχώρηση τιμών στις μεταβλητές
9  pi = 3.14;
10 radius = 8.5;
11
12 // υπολογισμός περιμέτρου και εμβαδού
13 perimetros = 2*pi*radius;
14 embado = pi*radius*radius;
15
16 // εμφάνιση αποτελεσμάτων στην οθόνη
17 printf("Περίμετρος = %f\n", perimetros);
18 printf("Εμβαδόν = %f\n", embado);
19 return 0;
20 }
```

Ερώτηση 2.14 Επαληθεύστε με τη βοήθεια του υπολογιστή τον κώδικα του Παραδείγματος 2.1. Εμφανίζονται στην οθόνη τα αναμενόμενα αποτελέσματα;

Άσκηση 2.1 Γιατί επιλέχθηκε ως τύπος δεδομένων για την ακτίνα του κύκλου στο Παράδειγμα 2.1 το **`float`**; Τι θα συνέβαινε αν αντί αυτού η μεταβλητή **`radius`** ήταν τύπου **`int`**; Επαληθεύστε την απάντησή σας τροποποιώντας κατάλληλα τον κώδικα του Παραδείγματος 2.1.
Υπόδειξη: Αντικαταστήστε τον τύπο δεδομένων όλων των μεταβλητών που επηρεάζονται με αυτόν που ζητάει η άσκηση (δηλαδή **`int`**). Επιπλέον,

για να εμφανιστούν στην οθόνη του υπολογιστή τα αποτελέσματα τύπου **int**, πρέπει να τροποποιήσετε τις γραμμές 19–20 του κώδικα, ως εξής:

```
19 printf("Περίμετρος = %d\n", perimetros);
20 printf("Εμβαδόν = %d\n", embado);
```

Άσκηση 2.2 Τροποποιήστε κατάλληλα τον κώδικα του Παραδείγματος 2.1, έτσι ώστε η ακτίνα του κύκλου να έχει τιμή 4.1. Στη συνέχεια, επαληθεύστε τα αποτελέσματα που εμφανίζονται στην οθόνη του υπολογιστή.

2.2 Αριθμητικές μετατροπές

Ιδιαίτερη προσοχή θα πρέπει να δοθεί στο γεγονός πως η επιλογή του τύπου μιας μεταβλητής μπορεί να επηρεάσει την τιμή του αποτελέσματος που αποθηκεύεται σ' αυτή. Χαρακτηριστικό παράδειγμα αποτελούν τα προβλήματα *αποκοπής* και *προσαρμογής* δεδομένων κάθε φορά που πραγματοποιείται ανάθεση τιμών σε μεταβλητές *λάθος τύπου*. Η σπουδαιότητα ενός τέτοιου είδους σφάλματος γίνεται εμφανής με τη βοήθεια του ακόλουθου παραδείγματος. Έστω λοιπόν ότι θέλουμε να αποθηκευτεί το αποτέλεσμα μιας διαίρεσης σε μεταβλητή τύπου **int**. Από τη στιγμή που ο συγκεκριμένος τύπος δεδομένων δε διαθέτει δεκαδικό μέρος, ο μεταγλωττιστής θα κρατήσει μόνο το ακέραιο τμήμα του αποτελέσματος, στρογγυλοποιώντας την τιμή του αριθμού προς την πλησιέστερη ακέραια μονάδα. Με αυτή τη λογική, το αποτέλεσμα του υπολογισμού είναι διαφορετικό από αυτό που θα αποθηκευτεί τελικά στη μεταβλητή. Θα πρέπει να σημειωθεί πως το συγκεκριμένο είδος λάθους συμβαίνει συχνά στους κώδικες των προγραμμάτων μας, και γι' αυτό θα πρέπει ο προγραμματιστής να είναι ιδιαίτερα προσεκτικός, προκειμένου να διασφαλίζει την ορθή εκτέλεση των υπολογισμών.

Ακολούθως, μπορείτε να βρείτε ορισμένα αντιπροσωπευτικά παραδείγματα δήλωσης και εκχώρησης τιμών σε μεταβλητές. Στις περιπτώσεις κατά τις οποίες συμβαίνει κάποια αριθμητική προσαρμογή στην τιμή που αποθηκεύεται, παραθέτουμε επιπλέον και τη σχετική αιτιολόγηση του σφάλματος.

Απόσπασμα Κώδικα 2.9

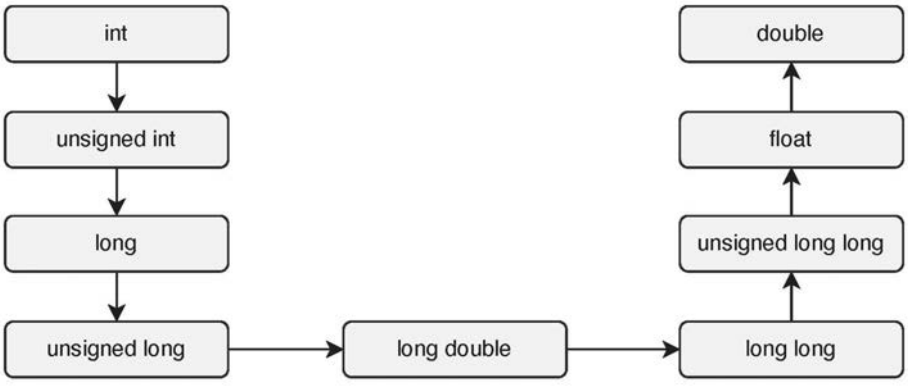
```
1 // σωστή δήλωση
2 int a=2;
3
4 // Λανθασμένη δήλωση, διότι συμβαίνει σφάλμα αποκοπής.
5 // Η τιμή της μεταβλητής b ισούται με το 3.
6 int b=3.14;
7
```

```
 8 // Η τιμή της μεταβλητής c ισούται με το 5.0.
 9 double c=5;
10
11 // Η τιμή της μεταβλητής d ισούται με το 8.0.
12 float d=8;
```

Κάτι αντίστοιχο συμβαίνει και στην περίπτωση που σε ένα πρόγραμμα πραγματοποιηθούν υπολογισμοί, ή εκχωρήσεις τιμών, μεταξύ μεταβλητών διαφορετικού τύπου. Για να καταστεί δυνατός ο υπολογισμός των συγκεκριμένων πράξεων, ο μεταγλωττιστής παρέχει ένα σύνολο από αυτοματοποιημένους μηχανισμούς προσαρμογής του τύπου μεταβλητών, οι οποίοι βασίζονται στην αρχή πως ο τύπος δεδομένων του τελεστέου με το μικρότερο εύρος τιμών μετατρέπεται στον αντίστοιχο τύπο του τελεστέου με το μεγαλύτερο εύρος τιμών. Στη συνέχεια, μπορείτε να βρείτε ορισμένες από τις βασικότερες αυτοματοποιημένες μετατροπές μεταξύ τελεστέων που υποστηρίζει ο μεταγλωττιστής της ANSI C.

- Αν σε κάποια αριθμητική πράξη υπάρχουν μεταβλητές τύπου **`float`**, **`double`** και **`long double`**, τότε όλες μετατρέπονται αυτόματα σε **`long double`**, σύμφωνα με την ακόλουθη σειρά: **`float`** → **`double`** → **`long double`**.
- Στην περίπτωση που κάποια αριθμητική πράξη περιλαμβάνει μεταβλητές τύπου **`float`** και **`int`**, τότε όλες μετατρέπονται στον τύπο **`float`**.
- Στην περίπτωση που η αριθμητική πράξη περιλαμβάνει μεταβλητές τύπου **`int`**, **`char`** και **`short`**, τότε όλες μετατρέπονται στο τύπο δεδομένων **`int`**. Ακολούθως, ενδέχεται να λάβουν χώρα επιπλέον μετατροπές σύμφωνα με την αλυσίδα: **`int`** → **`unsigned int`** → **`long int`** → **`unsigned long int`**.

Το Σχήμα 2.1 συνοψίζει την τυπική σειρά με την οποία λαμβάνουν χώρα οι μετατροπές των τύπων δεδομένων στη γλώσσα προγραμματισμού ANSI C.



Σχήμα 2.1: Αυτόματη προσαρμογή στους τύπους δεδομένων μεταβλητών στη γλώσσα ANSI C.

Αν και η αυτόματη προσαρμογή του τύπου δεδομένων οδηγεί συχνά σε κώδικες που μεταγλωττίζονται επιτυχώς, εντούτοις, συνιστάται η αποφυγή της συγκεκριμένης δυνατότητας διορθώνοντας τις όποιες προειδοποιήσεις (warnings) εμφανίζονται κατά τη διάρκεια της μεταγλώττισης του κώδικα, προκειμένου να διασφαλιστεί η ορθότητα του προγράμματος.

Ερώτηση 2.15 Ποια είναι η τιμή των μεταβλητών **a** έως **f** έπειτα από τις επόμενες εντολές εκχώρησης; Να αιτιολογήσετε την απάντησή σας.

1. `float a = 18.5;`
2. `int b = 2*a;`
3. `double c = a;`
4. `double d = a/5;`
5. `int e = a/5;`
6. `int f = 3.14159;`

2.3 Δήλωση σταθεράς

Πέρα από τις μεταβλητές, στα προγράμματα της ANSI C υπάρχει η δυνατότητα να ορίζουμε *σταθερές*. Όπως φανερώνει και το όνομά αυτών, το περιεχόμενο (δηλαδή η τιμή) μιας σταθεράς δεν μπορεί να τροποποιηθεί καθ' όλη τη διάρκεια της εκτέλεσης ενός προγράμματος. Μπορείτε να σκεφτείτε τι είδους δεδομένα θα μπορούσανε να αποθηκευτούν σε μια σταθερά; Πέρα από τις τυπικές περιπτώσεις, όπως είναι η τιμή του π=3.14 και το g=9.83 m/sec^2, οι σταθερές περιέχουν τιμές που θέλουμε να μείνουν αμετάβλητες κατά τη ροή εκτέλεσης ενός προγράμματος. Για παράδειγμα, αν απαιτείται ο υπολογισμός του διαστήματος που καλύπτει ένας ποδηλάτης ο οποίος εκτελεί ευθύγραμμη ομαλή κίνηση με σταθερή ταχύτητα **u** για διαφορετικούς χρόνους κίνησης, τότε θα μπορούσαμε να αναθέσουμε την τιμή της ταχύτητας (εκμεταλλευόμενοι τη διατύπωση "*σταθερή ταχύτητα*") σε μια σταθερά κατάλληλου τύπου.

Ερώτηση 2.16 Μπορείτε να επιχειρηματολογήσετε για το βέλτιστο τύπο δεδομένων που θα πρέπει να έχει η σταθερά της ταχύτητας στο προηγούμενο παράδειγμα;

Αναφορικά με την επιλογή ονόματος για μια σταθερά, ισχύουν όλα όσα έχουμε ήδη μάθει για τις μεταβλητές. Ως εκ τούτου, αυτό θα πρέπει να αποτελείται από συνδυασμούς πεζών και κεφαλαίων γραμμάτων του λατινικού αλφαβήτου (a-z,

A-Z), αριθμητικά ψηφία (0-9), καθώς επίσης και τον χαρακτήρα "_", λαμβάνοντας υπόψιν ότι ο πρώτος χαρακτήρας των σταθερών θα πρέπει υποχρεωτικά να είναι γράμμα του λατινικού αλφαβήτου ή ο χαρακτήρας "_". Επιπλέον, τα ονόματα των σταθερών θα πρέπει να είναι μοναδικά σε κάθε πρόγραμμα και να μη συμπίπτουν με κάποιο από τα χρησιμοποιούμενα ονόματα των μεταβλητών. Θυμηθείτε ότι η γλώσσα ANSI C κάνει διάκριση μεταξύ πεζών και κεφαλαίων στα ονόματα των σταθερών και των μεταβλητών.

Προκειμένου να εκχωρήσουμε κάποια τιμή σε μια σταθερά, αυτό μπορεί να πραγματοποιηθεί είτε στην αρχή μιας συνάρτησης (όμοια με τη δήλωση των μεταβλητών), ή εναλλακτικά με τη χρήση της οδηγίας **`#define`**. Συγκεκριμένα, σύμφωνα με τον πρώτο τρόπο, απαιτείται να χρησιμοποιήσουμε τη δεσμευμένη λέξη **`const`** πριν από τον επιλεγμένο τύπο δεδομένων της σταθεράς, ενώ στην περίπτωση που η δήλωση πραγματοποιηθεί με την οδηγία **`#define`**, τότε αυτή λαμβάνει χώρα αμέσως μετά τις δηλώσεις των βιβλιοθηκών του προγράμματος και πάντα εκτός των συναρτήσεων. Τα επόμενα δύο παραδείγματα ανάθεσης σταθερών παρουσιάζουν το συντακτικό των δύο εναλλακτικών τρόπων δήλωσης μιας σταθεράς σε ένα πρόγραμμα.

```
// 1ος τρόπος δήλωσης σταθεράς
const τύπος_σταθεράς όνομα_σταθεράς = τιμή;

// 2ος τρόπος δήλωσης σταθεράς
#define όνομα_σταθεράς τιμή;
```

Οι επόμενες εντολές θα μας βοηθήσουν στο να εμβαθύνουμε στους δύο εναλλακτικούς τρόπους δήλωσης μιας σταθεράς.

Απόσπασμα Κώδικα 2.10

```
1  // Ανάθεση της τιμής 3.1415 στη σταθερά PI
2  const float PI = 3.1415;
3
4  // Ανάθεση της τιμής 8 στη σταθερά u
5  const int u = 8;
6
7  // Ανάθεση της τιμής 9.6*10^20 στη σταθερά max_value
8  const double max_value = 9.6e20;
9
10 // Ανάθεση της τιμής 9.84 στη σταθερά G
11 #define G = 9.84;
12
13 // Ανάθεση της τιμής 4 στη σταθερά t
14 #define t = 4;
15
```

```
// Ανάθεση της τιμής 2.1*10^3 στη σταθερά max_value
#define min_value = 2.1e3;
```

Ερώτηση 2.17 Έχει νόημα να δηλώσετε μια σταθερά σε ένα πρόγραμμα χωρίς να της δώσετε τιμή; Τεκμηριώστε την απάντησή σας.

Ερώτηση 2.18 Δώστε τη δήλωση και την αρχικοποίηση σε μια αυθαίρετη τιμή των επόμενων μεταβλητών και σταθερών ενός προγράμματος, επιλέγοντας κάθε φορά τον κατάλληλο τύπο δεδομένων:

1. Μεταβλητή για την αποθήκευση του χρόνου κίνηση ενός οχήματος στη διαδρομή Αθήνα - Θεσσαλονίκη.
2. Μεταβλητή για την αποθήκευση της βαθμολογίας στο μάθημα του Προγραμματισμού.
3. Μεταβλητή για την αποθήκευση της κατανάλωσης ενέργειας σε kWh.
4. Μεταβλητή για την αποθήκευση του πλήθους φοιτητών σε ένα αμφιθέατρο.
5. Σταθερά για την αποθήκευση του μήκους των πλευρών ενός γεωμετρικού σχήματος.
6. Σταθερά για την αποθήκευση της τιμής του φορτίου ενός ηλεκτρονίου.
7. Σταθερά για την αποθήκευση της τιμής του π=3.14159.

Λυμένο Παράδειγμα 2.2 Γράψτε δύο εναλλακτικές υλοποιήσεις του Παραδείγματος 2.1, στις οποίες η δήλωση και η αρχικοποίηση της σταθεράς **`pi`** θα πραγματοποιείται με δύο διαφορετικούς τρόπους. Στη συνέχεια, το πρόγραμμα αφού αναθέσει στη μεταβλητή **`radius`** που δηλώνει την ακτίνα ενός κύκλου την τιμή 5.4, να εμφανίζει στην οθόνη του υπολογιστή την περίμετρο και το εμβαδόν αυτού.

Ενδεικτική Λύση:
1ος τρόπος λύσης

```
#include <stdio.h>

int main()
{
// δήλωση σταθεράς
const float pi = 3.14;
```

```
 7
 8 // δήλωση μεταβλητής για την ακτίνα του κύκλου
 9 float r = 5.4;
10
11 // δήλωση μεταβλητών για την περίμετρο και το εμβαδό
12 float perimetros, embado;
13
14 // υπολογισμός περιμέτρου και εμβαδού του κύκλου
15 perimetros = 2*pi*r;
16 embado = pi*r*r;
17
18 //  εμφάνιση αποτελεσμάτων στην οθόνη του υπολογιστή
19 printf("Η περίμετρος του κύκλου είναι %f\n", perimetros);
20 printf("Το εμβαδόν του κύκλου είναι %f\n", embado);
21
22 return 0;
23 }
```

2ος τρόπος λύσης

```
 1 #include <stdio.h>
 2 #define  pi  3.14   // δήλωση σταθεράς
 3
 4 int main()
 5 {
 6 // δήλωση μεταβλητής για την ακτίνα του κύκλου
 7 float r = 5.4;
 8
 9 // δήλωση μεταβλητών για την περίμετρο και το εμβαδό
10 float perimetros, embado;
11
12 // υπολογισμός περιμέτρου και εμβαδού του κύκλου
13 perimetros = 2*pi*r;
14 embado = pi*r*r;
15
16 //  εμφάνιση αποτελεσμάτων στην οθόνη του υπολογιστή
17 printf("Η περίμετρος του κύκλου είναι %f\n", perimetros);
18 printf("Το εμβαδόν του κύκλου είναι %f\n", embado);
19
20 return 0;
21 }
```

Επεξήγηση: Στο παράδειγμα αυτό, η συνάρτηση **`printf()`** χρησιμοποιείται για να εμφανίσει στην οθόνη του υπολογιστή των τιμών που έχουν τα αποτελέσματα της περιμέτρου και του εμβαδού, όπως αυτά αποθηκεύονται στις αντίστοιχες μεταβλητές τύπου **`float`**. Αυτό που αξίζει να παρατηρήσουμε στους συγκεκριμένους κώδικες είναι οι εναλλακτικοί τρόποι

δήλωσης της σταθεράς π. Σύμφωνα με την πρώτη προσέγγιση, αυτό πραγματοποιείται με τη βοήθεια της εντολής **const** μαζί με τις δηλώσεις των μεταβλητών. Αντίθετα, στην περίπτωση που χρησιμοποιούμε την οδηγία **#define**, τότε η δήλωση της σταθεράς λαμβάνει χώρα έπειτα από τις δηλώσεις των βιβλιοθηκών του προγράμματος.

2.4 Τελεστές

2.4.1 Τελεστής εκχώρησης =

Στα παραδείγματα του βιβλίου που έχουμε μελετήσει έως τώρα χρησιμοποιούμε κατά κόρον τον τελεστή "**=**" για να εκχωρούμε τιμές στις μεταβλητές και σταθερές των προγραμμάτων. Σύμφωνα με τον ορισμό του συγκεκριμένου τελεστή, η τιμή που βρίσκεται στη δεξιά πλευρά της ανάθεσης αποθηκεύεται στη μεταβλητή (ή τη σταθέρα) που δηλώνεται αριστερά της ισότητας, όπως χαρακτηριστικά αυτό παρουσιάζεται με τη βοήθεια των επόμενων παραδειγμάτων:

Απόσπασμα Κώδικα 2.11

```
// Εκχώρηση της τιμής 10 στη μεταβλητή τύπου ακέραιου counter
int counter = 10;

// Εκχώρηση της τιμής 5 στη μεταβλητή radius
float radius = 5.1;

// Εκχώρηση της τιμής 3.14 στη σταθερά pi
const float pi = 3.14;

// Ανάθεση της τιμής 9.1 στη μεταβλητή τύπου float με όνομα value.
// Η δήλωση της μεταβλητής value πρέπει να έχει προηγηθεί.
value = 9.1;
```

Επιπλέον, ο μεταγλωττιστής της γλώσσας ANSI C υποστηρίζει την εκχώρηση της τιμής μιας μεταβλητής ή σταθεράς σε μια άλλη, όπως αυτό φαίνεται στο ακόλουθο Απόσπασμα Κώδικα. Πρέπει να σημειωθεί πως ο συγκεκριμένος τρόπος είναι αυτός που χρησιμοποιείται κατά κόρον στους αλγορίθμους των προγραμμάτων διότι επιτρέπει την υλοποίηση πολύπλοκων αριθμητικών και λογικών πράξεων, οι οποίες είναι παραμετρικές ως προς την τιμή των μεταβλητών.

Θα πρέπει να καταστεί σαφές πως η εκχώρηση τιμών με τη βοήθεια του τελεστή **=** πραγματοποιείται πάντα από τα δεξιά προς τα αριστερά. Εφαρμόζοντας επαναληπτικά τη συγκεκριμένη διαδικασία μπορούμε να πετύχουμε την πολλαπλή εκχώρηση μιας τιμής σε περισσότερες από μια μεταβλητές.