Κεφάλαιο 1

ΓΝΩΡΙΜΙΑ ΜΕ ΤΗ ΓΛΩΣΣΑ C

1.1. Εισαγωγή

Η *επιστήμη των υπολογιστών* αποτελεί την πιο ταχέως αναπτυσσόμενη περιοχή έρευνας και εφαρμογών των τελευταίων ετών. Δραστηριοποιείται σε πολλούς κλάδους με έμφαση στην παραγωγή αποτελεσμάτων μετά από πολλές πράξεις.

Ο προγραμματισμός των υπολογιστών αποτελεί το κυριότερο εργαλείο της επιστήμης των υπολογιστών για να επιλύει σύνθετα υπολογιστικά προβλήματα και για να επιτύχει αυτή την πολύπλοκη αποστολή του, εφαρμόζει ανάλογα με τη φύση του προβλήματος και διαφορετικές γλώσσες προγραμματισμού.

Λέμε *γλώσσα προγραμματισμού* (computer language) μια νέα τεχνητή γλώσσα η οποία μπορεί να χρησιμοποιηθεί για τον έλεγχο και τη λειτουργία του υπολογιστή. Μια γλώσσα προγραμματισμού ορίζεται από ένα συγκεκριμένο και αυστηρό σύνολο συντακτικών και εννοιολογικών κανόνων οι οποίοι ορίζουν τη δομή της γλώσσας.

Οι γλώσσες προγραμματισμού γενικά, διευκολύνουν την οργάνωση και τη διαχείριση των πληροφοριών.

Υπάρχουν πάρα πολλές και διαφορετικές γλώσσες προγραμματισμού οι οποίες δημιουργήθηκαν για να ικανοποιήσουν τις απαιτήσεις και τις ανάγκες των πολυάριθμων προβλημάτων των σύγχρονων επιστημών και η κατηγοριοποίηση τους δεν είναι εύκολη υπόθεση.

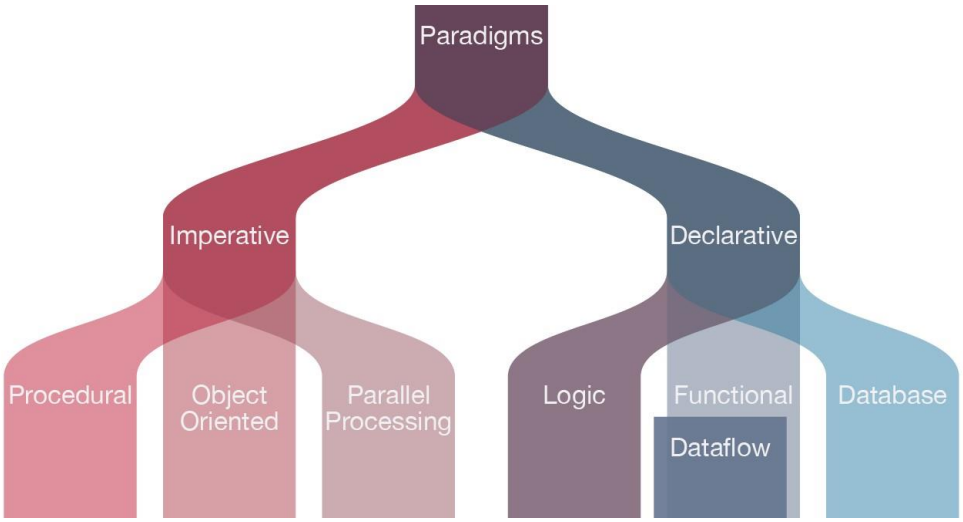
Η πρώτη γλώσσα προγραμματισμού ήταν ένας αλγόριθμος που δημιουργήθηκε από την Ada Lovelace το 1883. Η Ada Lovelace
δημιούργησε αυτόν τον αλγόριθμο για τον αναλυτικό κινητήρα του Charles Babbage. Ο σκοπός αυτού του αλγορίθμου ήταν να υπολογίσει τους αριθμούς Bernoulli.

Το 1936, για πρώτη φορά, οι κωδικοί υπολογιστών εξειδικεύτηκαν από τον Church Alonzo και τον Alan Turing όπου εξέφρασαν το λάμδα λογισμό (Lambda calculus) με τυποποιημένο τρόπο.

Οι γλώσσες προγραμματισμού υπολογιστών χρησιμοποιούνται στη μετάδοση οδηγιών σε έναν υπολογιστή για τον έλεγχο και τη λειτουργεία του ή για την έκφραση αλγορίθμων. Βασίζονται σε ορισμένους αυστηρούς συντακτικούς (syntax) και

εννοιολογικούς κανόνες (semantics), οι οποίοι ορίζουν την έννοια κάθε δομής γλώσσας προγραμματισμού.

Στο πεδίο του υπολογιστή, πολλές γλώσσες προγραμματισμού χρησιμοποιούν δηλωτική μορφή ενώ άλλες πρέπει να δηλώνονται σε προστακτική μορφή.



Εικόνα 2.3 Διαφορετικοί τύποι γλωσσών προγραμματισμού
https://www.typesnuses.com/types-of-programming-languages-withdifferences/

Σε αντίθεση με τον δηλωτικό προγραμματισμό, ο οποίος περιγράφει "τι" πρέπει να πραγματοποιήσει ένα πρόγραμμα, ο προστακτικός προγραμματισμός λέει ρητά στον υπολογιστή "πώς" να το επιτύχει (εικόνα 1.0)

Ο δηλωτικός προγραμματισμός (declarative programming) είναι ένα πρότυπο προγραμματισμού που εκφράζει τη λογική ενός υπολογισμού χωρίς να περιγράφει τη ροή ελέγχου του.

Οι κοινές δηλωτικές γλώσσες περιλαμβάνουν:

Τις γλώσσες βάσης δεδομένων (database languages) (π.χ. SQL),

Τις γλώσσες λογικού προγραμματισμού (logic programming languages) (π.χ. Prolog)

Τις γλώσσες λειτουργικού προγραμματισμού (functional programming languages) (π.χ. Huskell)

Τις γλώσσες ροής δεδομένων (dataflow languages) (π.χ. Max/MSP)

Ο προστακτικός προγραμματισμός (imperative programming) είναι ένα πρότυπο προγραμματισμού στο οποίο το πρόγραμμα περιγράφει μια ακολουθία βημάτων που αλλάζουν την κατάσταση του υπολογιστή.

Οι κοινές προστακτικές γλώσσες περιλαμβάνουν:

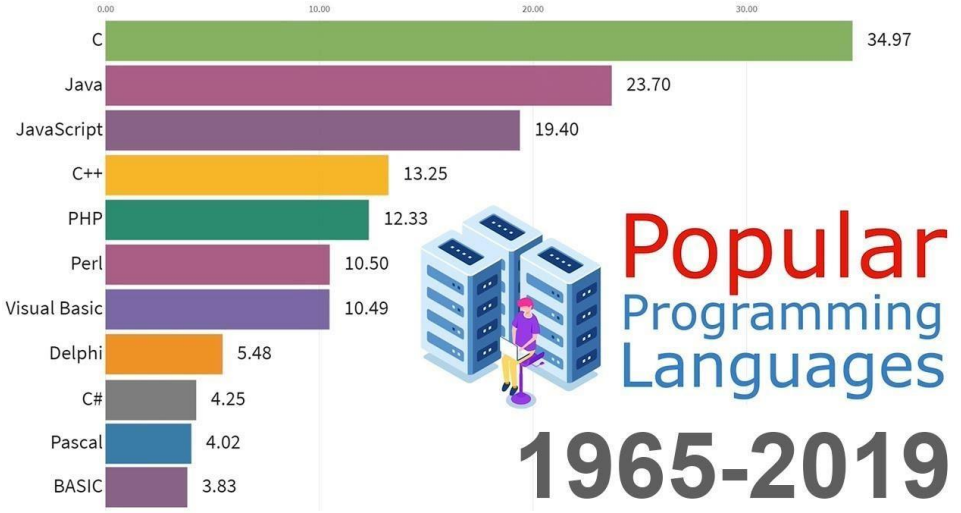
Τις γλώσσες διαδικαστικού προγραμματισμού (procedure programming languages) (π.χ. C)

Τις γλώσσες αντικειμενοστραφή προγραμματισμού (object-oriented programming languages) (π.χ. Smalltalk)

Τις γλώσσες παράλληλου προγραμματισμού (parallel computing languages) (π.χ. SequenceL)

Οι περισσότερες γλώσσες προγραμματισμού εκτός από ένα «αγνό» (pure) πρότυπο προγραμματισμού υποστηρίζουν πολλαπλά πρότυπα προγραμματισμού (multi-paradigm).

Όταν το ζητούμενο αποτέλεσμα θα υπολογίζεται περιγράφοντας μόνο τις επιθυμητές ιδιότητές του τότε, έχουμε την ανάγκη μιας γλώσσας της οικογένειας του *δηλωτικού προγραμματισμού* (declarative programming). Παραδείγματα γλωσσών δηλωτικού προγραμματισμού είναι οι γλώσσες Haskell, Prolog, Lisp και HTML.



Σχήμα 1.1. Οι πιο δημοφιλείς γλώσσες προγραμματισμού
(https://www.youtube.com/watch?v=Og847HVwRSI)

Όταν το ζητούμενο αποτέλεσμα θα υπολογίζεται αλλάζοντας την κατάσταση της μνήμης του υπολογιστή με τη βοήθεια εντολών τότε, έχουμε την ανάγκη μιας γλώσσας της οικογένειας του *προστακτικού προγραμματισμού* (imperative programming).

Παραδείγματα γλωσσών προστακτικού προγραμματισμού είναι κυρίως οι γλώσσες προγραμματισμού: Pascal, C, Fortran κ.τ.λ.

Διαχρονικά, η πιο δημοφιλής γλώσσα προγραμματισμού είναι η γλώσσα C όπως αποτυπώνεται στην εικόνα 1.1.

Ο διαδικαστικός προγραμματισμός βασίζεται στην έννοια της κλήσης διαδικασιών. *Διαδικασία* (procedure) είναι ένα αυτοτελές σύνολο εντολών προς εκτέλεση. Μια διαδικασία είναι γνωστή και με τις λέξεις *ρουτίνα*, *υπορουτίνα, μέθοδος*, *συνάρτηση (function).*

Το αυτοτελές σύνολο των εντολών μιας διαδικασίας μπορούμε να το διαχειριστούμε ευκολότερα και με μεγαλύτερη ασφάλεια, από ότι το σύνολο των εντολών ενός προγράμματος. Μπορούμε επίσης, να το συντηρήσουμε πιο εύκολα και ταχύτερα και να διορθώσουμε τα ορθογραφικά και συντακτικά λάθη. Οι ενέργειες για τη διόρθωση των λαθών ενός προγράμματος είναι γνωστές με τον όρο *αποσφαλμάτωση* (debugging). Η έννοια της διαδικασίας αποτελεί για τη γλώσσα προγραμματισμού C το βασικό συστατικό της, το οποίο επιτρέπει στους προγραμματιστές να αναπτύξουν τις εφαρμογές τους με ταχύτητα και απλότητα.

Ο δομημένος προγραμματισμός ο οποίος εμφανίστηκε στα τέλη της δεκαετίας του '60 και πολύ γρήγορα έγινε δημοφιλής στους προγραμματιστές, δημιουργήθηκε για να βελτιώσει και να συστηματοποιήσει τις τεχνικές του παλαιότερου διαδικαστικού προγραμματισμού. Έτσι,, στην έννοια της διαδικασίας προστέθηκε η αναγκαιότητα της ανάλυσης ενός προγράμματος σε θεμελιώδεις διαδικασίες.

Με την ομαδοποίηση των διαδικασιών και τη δημιουργία αρχείων διαδικασιών δημιουργούνται οι *βιβλιοθήκες* (library) δηλαδή, μια συλλογή από έτοιμα υποπρογράμματα τα οποία χρησιμοποιούνται πολύ εύκολα για την ταχύτερη και ασφαλέστερη ανάπτυξη νέων εφαρμογών.

Οι βιβλιοθήκες περιέχουν επαναχρησιμοποιούμενο κώδικα και δηλώσεις οι οποίες μπορούν να χρησιμοποιηθούν άμεσα από τους προγραμματιστές μέσα στα νέα προγράμματα τα οποία αναπτύσσουν. Αυτή η τεχνική επιτρέπει το διαμοιρασμό και τη χρήση κώδικα και δεδομένων με πιο αποτελεσματικό τρόπο.

Η γλώσσα C, υιοθέτησε τις αρχές του δομημένου προγραμματισμού και προσαρμόστηκε γρήγορα στις απαιτήσεις της εξελισσόμενης επιστήμης των υπολογιστών με αποτέλεσμα να γίνει πολύ γρήγορα δημοφιλής στους προγραμματιστές των υπολογιστών αφού παρείχε όλα τα στοιχεία για ασφαλή και αποδοτικό προγραμματισμό.

Είναι μια γλώσσα προγραμματισμού γενικού σκοπού, προσφέρει οικονομία στην έκφραση, μοντέρνο έλεγχο της ροής του προγράμματος, πλήρεις δομές δεδομένων καθώς και ένα πλούσιο σύνολο τελεστών. Η απουσία περιορισμών και η γενικότητά της την κάνουν ιδιαίτερα εύχρηστη και αποτελεσματική για την υλοποίηση μεγάλου εύρους εφαρμογών.

Επιπλέον, παρουσιάζει πολλά και χαρακτηριστικά πλεονεκτήματα:

- **Συντήρηση**: Τα προγράμματα είναι εύκολο να συντηρηθούν.
- **Αναγνωσιμότητα**: Τα προγράμματα διαβάζονται εύκολα.
- **Μεταφερσιμότητα**: Τα προγράμματα εύκολα μεταφέρονται σε διαφορετικά λειτουργικά συστήματα και υπολογιστές.

Τέλος, η γλώσσα C, διαθέτει όλες τις χαρακτηριστικές ιδιότητες των γλωσσών μέσου επιπέδου και με τη δύναμη της δομής της αποτελεί υπόδειγμα για τη δημιουργία πολλών άλλων νέων γλωσσών προγραμματισμού.

Η γλώσσα C είναι ίσως η πιο δημοφιλής γλώσσα προγραμματισμού για την ανάπτυξη λογισμικού. Οι εταιρείες που δημιουργούν λειτουργικά συστήματα, μεταγλωττιστές γλωσσών, βάσεις δεδομένων και οδηγούς δικτύου (drivers) στηρίζονται σε μεγάλο βαθμό στη γλώσσα C.

Η εκμάθηση της γλώσσας C κρίνεται απαραίτητη γιατί αποτελεί τη βάση πολλών σύγχρονων γλωσσών προγραμματισμού και σύντομα θα γίνει ο καλύτερος φίλος σας.

1.1.1. Παράδειγμα προγράμματος

Ας ξεκινήσουμε με το απλό αλλά και κλασσικό παράδειγμα ενός προγράμματος σε γλώσσα C, όπως εμφανίζεται και στο πρώτο βιβλίο για τη γλώσσα C, των Kernigham and Ritchie [1] δηλαδή, αυτό του Περιγράμματος 1.1. το οποίο εμφανίζει στην οθόνη του υπολογιστή τη φράση **HELLO WORLD**.

```
#include <stdio.h>
main( )
   {
      printf("HELLO WORLD\n");
   }
```

Περίγραμμα 1.1. Ένα απλό πρόγραμμα σε γλώσσα C

Βασικές παρατηρήσεις.

Ένα πρόγραμμα της γλώσσας C είναι η συλλογή μιας ή περισσότερων συναρτήσεων (functions). Για να γράψουμε ένα πρόγραμμα πρώτα δημιουργούμε τις συναρτήσεις και μετά τις ενώνουμε.

Στο παράδειγμα του Περιγράμματος 1.1. έχουμε μόνο μια συνάρτηση, τη **main**.

Συνάρτηση είναι ένα μικρό κομμάτι προγράμματος (υποπρόγραμμα) το οποίο περιέχει μία ή περισσότερες εντολές και εκτελεί μία ή περισσότερες αποστολές δηλαδή, υπολογίζει τα αποτελέσματα των πράξεων του προγράμματος ή εκτελεί διαδικασίες εισόδου/εξόδου.

Ένα πρόγραμμα της γλώσσας C, θεωρείται ότι είναι σύμφωνο με τους κανόνες της γλώσσας όταν κάθε συνάρτηση εκτελεί μόνο μία συγκεκριμένη δουλειά, αν και αυτό δεν συμβαίνει συχνά.

Κάθε συνάρτηση έχει ένα συγκεκριμένο και διακεκριμένο όνομα το οποίο ακολουθείται από μία λίστα παραμέτρων, μέσα σε δύο παρενθέσεις. Τις τιμές των παραμέτρων θα δεχθεί η συνάρτηση και θα τις χρησιμοποιήσει κατά τη διάρκεια των πράξεων σύμφωνα με τον αλγόριθμο επίλυσης του προβλήματος.

Γενικά, μπορούμε να δώσουμε σε μια συνάρτηση οποιοδήποτε επιτρεπτό από τους κανόνες της γλώσσας όνομα, με εξαίρεση το όνομα **main**, το οποίο δεσμεύεται για την ειδική συνάρτηση με την οποία αρχίζει η εκτέλεση των πράξεων ενός προγράμματος της γλώσσας C.

Για να δηλώσουμε μια νέα συνάρτηση αρκεί να χρησιμοποιηθεί μια παραδοχή η οποία έχει γίνει αποδεκτή όταν γράφουμε προγράμματα στη γλώσσα C. Η παραδοχή ορίζει ότι οι συναρτήσεις θα έχουν παρενθέσεις μετά το όνομά τους.

Π.χ. αν το όνομα μιας συνάρτησης θέλουμε να είναι το **max**, τότε πρέπει να γράψουμε **max( ).**

Αυτή η γραφή είναι χρήσιμη και βοηθά πολύ για να ξεχωρίζουμε τα ονόματα των μεταβλητών του προγράμματος από τα ονόματα των συναρτήσεων.

Στο πρόγραμμα του Περιγράμματος 1.1. τόσο το όνομα **main( )** όσο και το όνομα **printf( )** είναι συναρτήσεις.

Η συνάρτηση **main( ),** είναι η πρώτη συνάρτηση η οποία εκτελείται αυτόματα όταν το πρόγραμμα αρχίζει να *τρέχει* (run) στον υπολογιστή.

Η συνάρτηση **printf( )** δεν αποτελεί γνήσιο τμήμα της γλώσσας C, είναι ένα υποπρόγραμμα γραμμένο σε γλώσσα C το οποίο βρίσκεται σε μια από τις πολλές τυποποιημένες βιβλιοθήκες της γλώσσας C. Οι βιβλιοθήκες αυτές συνοδεύουν αναγκαστικά όλες τις εκδόσεις της γλώσσας και αναφέρονται σαν τυπικές βιβλιοθήκες (standards library) της γλώσσας C. Η συνάρτηση **printf( )** βρίσκεται αποθηκευμένη στην τυπική βιβλιοθήκη **stdio** (STandarD Input Output). Για να γνωρίζει ο μεταγλωττιστής της γλώσσας C, ποια βιβλιοθήκη πρέπει να χρησιμοποιήσει για τη δημιουργία του τελικού εκτελέσιμου προγράμματος πρέπει να προστεθεί η *οδηγία*

(directive) **#include** η οποία λέγεται και οδηγία του προ-επεξεργαστή της γλώσσας C.

Τοποθετούμε όλες τις οδηγίες **#include** στην αρχή του προγράμματος και πριν από οποιαδήποτε άλλη εντολή της γλώσσας C.

Η συνάρτηση **printf( )** εμφανίζει την τιμή της παραμέτρου της στην οθόνη του υπολογιστή.

Στο πρόγραμμα του Περιγράμματος 1.1., η μοναδική παράμετρος της συνάρτηση **printf( )** είναι η σειρά των χαρακτήρων μέσα στην παρένθεση δηλαδή, το σύνολο των χαρακτήρων "HELLO WORLD\n".

Εκτός από τη γνήσια σειρά των χαρακτήρων HELLO WORLD, το ζεύγος **\n** είναι ένα σύμβολο το οποίο χρησιμοποιεί η γλώσσα C για να μεταφέρει την επόμενη εμφάνιση των αποτελεσμάτων της εκτέλεσης του προγράμματος στην επόμενη γραμμή. Δηλαδή, αν υπάρχει μια επόμενη εντολή εξόδου στο πρόγραμμα τότε θα ξεκινήσει η εμφάνιση των αποτελεσμάτων της στην επόμενη γραμμή της οθόνης.

Αν δεν υπάρχει το ζεύγος **\n** τότε, τα νέα αποτελέσματα από μια επόμενη εντολή εξόδου, θα εμφανιστούν στην ίδια γραμμή της οθόνης και ακριβώς μετά το τέλος της τελευταίας εμφάνισης.

Ο κώδικας μιας συνάρτησης δηλαδή, οι εντολές της, οριοθετούνται από το ζεύγος των αγκυλών { και }. Η αγκύλη { υποδεικνύει το αρχικό σημείο των εντολών της συνάρτησης και η αγκύλη } υποδεικνύει το τέλος των εντολών της συνάρτησης.

1.1.2. Ιστορικά στοιχεία

Η γλώσσα C αναπτύχθηκε από τον Dennis Ritchie στα AT&T Bell Labs (Murray Hill, New Jersey, USA) από το 1969 μέχρι το 1973. Ονομάστηκε
"γλώσσα C" επειδή πολλά χαρακτηριστικά και ιδέες προήλθαν από μια παλαιότερη γλώσσα προγραμματισμού, τη γλώσσα "B".

Το 1978, ο Dennis Ritchie και ο Brian Kernighan εκδίδουν το βιβλίο με τίτλο *"The C Programming Language"* το οποίο για πολλά χρόνια θεωρείτο ως ο ανεπίσημος ορισμός της γλώσσας.

Η έκδοση της γλώσσας C του βιβλίου αυτού αναφέρεται συνήθως ως "K&R C" .

Το 1983, το American National Standards Institute (ANSI) στην προσπάθεια εναρμόνισης των διαφορετικών εκδόσεων της γλώσσας και δημιουργίας μιας έκδοσης η οποία θα επέτρεπε τη φορητότητα των προγραμμάτων της γλώσσας C, ορίζει μια επιτροπή (με κωδικό X3J11), και με στόχο την κανονικοποίηση της γλώσσας. Το πρώτο *πρότυπο* (standard) ολοκληρώθηκε το 1989 και είναι γνωστό ως ANSI X3.159-1989 "Programming Language C".

Το 1990, το πρότυπο ANSI υιοθετήθηκε και από τον *Οργανισμό Διεθνών Προτύπων* (International Organization for Standardization (ISO) με τον κωδικό ISO/IEC 9899:1990 και έτσι προέκυψε η τυποποιημένη γλώσσα C90. Ουσιαστικά, οι εκφράσεις "C89" και "C90" αναφέρονται στην ίδια έκδοση της γλώσσας C και με τα ίδια χαρακτηριστικά.

Το πρότυπο της γλώσσας C89 κατά τη διάρκεια της δεκαετίας του '90, βελτιώθηκε σημαντικά μετά από τις εύστοχες παρατηρήσεις των διαρκώς αυξανόμενων χρηστών της γλώσσας και τις εξελίξεις της επιστήμης των υπολογιστών.

Οι βελτιώσεις αυτές προετοίμασαν την έκδοση του νέου προτύπου το έτος 1999 με τον κωδικό ISO 9899:1999. Το πρότυπο αυτό αναφέρεται ως "**C99**" και υιοθετήθηκε

ως πρότυπο του ANSI το Μάρτιο του 2000. Επειδή τα πρότυπα της γλώσσας συνεχώς ενημερώνονται και βελτιώνονται, η επόμενη έκδοση ήταν η γλώσσα "**C11**" (**ISO/IEC 9899:2011).** Το τρέχον διεθνές πρότυπο που καθορίζει τη γλώσσα προγραμματισμού C είναι το **ISO / IEC 9899: 2018** - ονομάζεται επίσης **C17** και **C18**.
Αυτή η έκδοση αντιμετωπίζει πολλά ελαττώματα που αναφέρθηκαν για την έκδοση **C11** και δεν εισάγει νέα γλωσσικά χαρακτηριστικά.

Οι σύγχρονοι μεταγλωττιστές της γλώσσας C, υποστηρίζουν όλα τα χαρακτηριστικά των προτύπων **C89** και **C99** και τα περισσότερα χαρακτηριστικά των νέων προτύπων **C11** και **C18**.

Έτσι, για να εξασφαλίσουμε, προς το παρόν, τη φορητότητα (portability) στα προγράμματά μας δηλαδή, να μπορούν να μεταγλωττίζονται χωρίς αλλαγές σε όλους τους μεταγλωττιστές, θα πρέπει να επιμένουμε και να εμπιστευόμαστε περισσότερο τα τελευταία πρότυπα **C89** και **C99**.

Η γλώσσα C ενέπνευσε και αποτέλεσε βασικό οδηγό για πάρα πολλές νεότερες γλώσσες προγραμματισμού και πιο ειδικά για τις βασικές γλώσσες προγραμματισμού τις προσανατολισμένες στο διαδίκτυο, όπως είναι οι γλώσσες: **C++, C# , Java, JavaScript, PHP** κτλ.

Η καλή γνώση και εμπειρία του προγραμματισμού με τη γλώσσα C, μπορεί να βοηθήσει σημαντικά τους προγραμματιστές στην ομαλή μετάβασή τους σε όλες αυτές τις νεότερες γλώσσες προγραμματισμού.

Σημείωση. *Το* ***Παράρτημα Α,*** *στο τέλος του βιβλίου, περιέχει όλες τις οδηγίες για την τυποποιημένη έκδοση της γλώσσας C. Αυτή η έκδοση της γλώσσας C ονομάστηκε ANSI C, και αργότερα* ***C89*** *(για να διαχωρίζεται από την επόμενη έκδοση, τη* ***C99).***

1.2. Βασικοί ορισμοί

Πηγαίο πρόγραμμα (Source code). Λέμε *πηγαίο πρόγραμμα* ή *πηγαίο κώδικα*, το αρχικό κείμενο ενός προγράμματος δηλαδή, τις εντολές τις οποίες γράφει ο προγραμματιστής και τις οποίες μπορεί να διακρίνει και να επεξεργαστεί εύκολα κάθε χρήστης του προγράμματος στην οθόνη του υπολογιστή με τη βοήθεια ενός *διορθωτή κειμένου* (editor). Το πηγαίο πρόγραμμα είναι ένα αρχείο κειμένου (text file) το οποίο αποθηκεύεται με επέκταση το γράμμα c ή C, σε κάποιο βοηθητικό αποθηκευτικό μέσο χρήστη (σκληρό δίσκο, δισκέτα, USB Flash κ.τ.λ.) Π.χ. Το πρόγραμμα του Περιγράμματος 1.1. μπορούμε να το αποθηκεύσουμε με όνομα **first.c**

Αντικειμενικό πρόγραμμα (Object Code). Λέμε *αντικειμενικό πρόγραμμα* ή *αντικειμενικό κώδικα*, το αποτέλεσμα της μετάφρασης του πηγαίου προγράμματος σε κώδικα μηχανής. Ένα αντικειμενικό πρόγραμμα έχει επέκταση το γράμμα : **o**, ή τρία γράμματα: **obj** και διατηρεί το όνομα του πηγαίου προγράμματος. Το αντικειμενικό πρόγραμμα χρησιμοποιείται για τη δημιουργία του τελικού εκτελέσιμου προγράμματος. Π.χ. το αντικειμενικό πρόγραμμα του προγράμματος **first.c** μετά τη μεταγλώττισή του θα έχει όνομα **first.obj**.

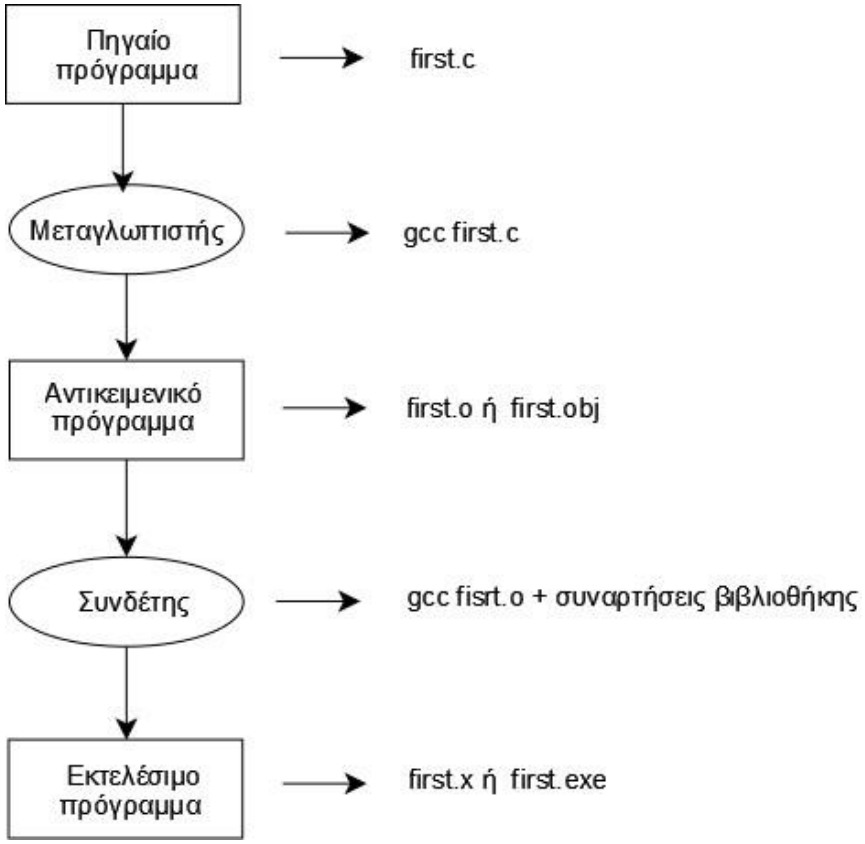
Μεταγλωττιστής (Compiler). Λέμε *μεταγλωττιστή*, το ειδικό πρόγραμμα το οποίο ελέγχει τα ορθογραφικά και συντακτικά λάθη του πηγαίου προγράμματος και ετοιμάζει

το αντικειμενικό πρόγραμμα. Για τη γλώσσα C, διατίθενται πολλές εκδόσεις προγραμμάτων μεταγλώττισης όπως: GCC, MSVC, Borland C, Watcom C. Στο λειτουργικό σύστημα των Windows, ο μεταγλωττιστής της γλώσσας C, είναι ενσωματωμένος στο περιβάλλον του Visual Studio. Στο **Παράρτημα Γ** του βιβλίου, αναφέρονται οι πιο γνωστοί μεταγλωττιστές της γλώσσας C καθώς και οι διευθύνσεις διάθεσής τους από το διαδίκτυο, έτσι ώστε να μπορούν να διαλέξουν οι προγραμματιστές ανάλογα με το περιβάλλον εργασίας τους, τον πιο κατάλληλο για την εκμάθηση της γλώσσας C και για την ανάπτυξη των προγραμμάτων τους. Μερικοί από αυτούς τους μεταγλωττιστές διατίθενται εντελώς δωρεάν, υπάρχουν όμως και μεταγλωττιστές οι οποίοι διατίθενται με κάποιο χρηματικό κόστος.

Συνδέτης (linker, ή loader, ή bilder). Λέμε *συνδέτη*, ένα ειδικό βοηθητικό πρόγραμμα το οποίο αναλαμβάνει να συνδέσει ξεχωριστά μεταγλωττισμένες συναρτήσεις τις οποίες γράφει ο προγραμματιστής καθώς και τις συναρτήσεις βιβλιοθήκης της γλώσσας C, σε ένα τελικό εκτελέσιμο πρόγραμμα. Το αποτέλεσμα του συνδέτη είναι ένα αρχείο το οποίο για το λειτουργικό σύστημα των Windows διατηρεί το αρχικό όνομα του αρχείου και έχει επέκταση .**exe**, ενώ για τα συστήματα Unix και Linux έχει όνομα **a.out,** εκτός και αν οριστεί από τον προγραμματιστή ένα άλλο, διαφορετικό, όνομα.

Βιβλιοθήκη (library). Λέμε *βιβλιοθήκη*, ένα αρχείο το οποίο περιέχει μια ή και περισσότερες συναρτήσεις, οι οποίες μπορούν να χρησιμοποιηθούν σε ένα πρόγραμμα. Συνήθως έχει επέκταση το γράμμα **h**.

Χρόνος εκτέλεσης (Run time). Λέμε *χρόνο εκτέλεσης*, τη χρονική περίοδο κατά την οποία ένα πρόγραμμα εκτελείται στον υπολογιστή. Συνήθως, κατά το χρόνο εκτέλεσης λέμε ότι το πρόγραμμα "τρέχει" (run). Αν συμβεί κατά τη διάρκεια του χρόνου εκτέλεσης του προγράμματος ένα λάθος (run time error), αυτό αποτελεί ένα σφάλμα της εκτέλεσης του προγράμματος το οποίο αναγνωρίζεται από τον υπολογιστή αλλά οφείλεται συνήθως στον προγραμματιστή π.χ. λανθασμένη απόδοση του αλγορίθμου ή λανθασμένη χρήση τελεστών. Στην περίπτωση αυτή σταματά η εκτέλεση των υπολοίπων εντολών του προγράμματος και στην οθόνη εμφανίζεται το σχετικό μήνυμα λάθους το οποίο ενημερώνει το χρήστη για την αιτία της διακοπής του προγράμματος.

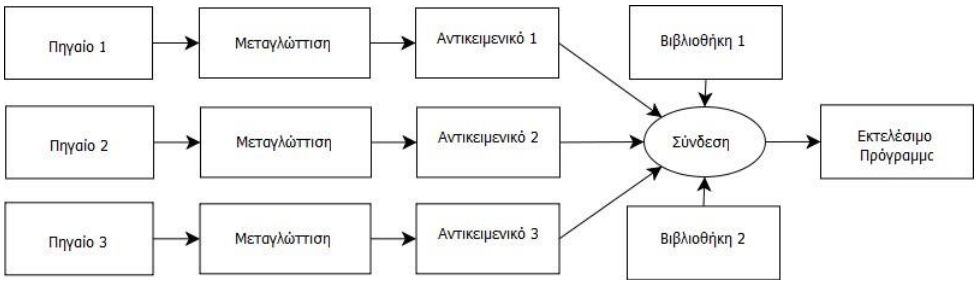


Σχήμα 1.1. Παράσταση των διαδοχικών φάσεων ενός πηγαίου προγράμματος

Στο Σχήμα 1.1. διακρίνουμε τις τρεις διαδοχικές φάσεις τις οποίες πρέπει να "περάσει" υποχρεωτικά το πηγαίο πρόγραμμα (**first.c**) για να είναι έτοιμο να "τρέξει" δηλαδή, να εκτελέσει τον αλγόριθμο του προβλήματος για το οποίο έχει αναπτυχθεί.

Ένας τυπικός προγραμματιστής δεν ενδιαφέρεται παρά μόνο για την πρώτη φάση. Τη φάση δηλαδή, της σύνταξης του πηγαίου κώδικα του προγράμματος. Τις υπόλοιπες δύο φάσεις (μεταγλώττιση και σύνδεση) αναλαμβάνει είτε το ολοκληρωμένο περιβάλλον στο οποίο εργάζεται ο προγραμματιστής είτε το ίδιο το λειτουργικό σύστημα, για τον έλεγχο και τη δημιουργία του εκτελέσιμου προγράμματος (αρχείο **first.exe**).

Επίσης, η γλώσσα C, επιτρέπει τη εύκολη διαμέλιση ενός μεγάλου σε όγκο εντολών προγράμματος σε μικρές ενότητες ανεξάρτητων προγραμμάτων τα οποία αποθηκεύονται σε διαφορετικά αρχεία και σε διαφορετικά αποθηκευτικά μέσα αλλά με τη βοήθεια του συνδέτη μπορούν να φορτωθούν όλα μαζί για το σχηματισμό του τελικού εκτελέσιμου αρχείου του προγράμματος.



Σχήμα 1.2. Σύνδεση χωριστά μεταγλωττισμένων πηγαίων προγραμμάτων

Η ωφέλεια από αυτή τη διαδικασία εμφανίζεται όταν θέλουμε να τροποποιήσουμε μόνο ένα τμήμα του πηγαίου κώδικα το οποίο βρίσκεται σε ένα συγκεκριμένο αρχείο.

Στην περίπτωση αυτή, δεν απαιτείται εκ νέου μεταγλώττιση όλου του κώδικα του προγράμματος παρά μόνο του αρχείου στο οποίο βρίσκεται το τροποποιημένο τμήμα του πηγαίου κώδικα.

Στη συνέχεια, ο συνδέτης θα αναλάβει τη δημιουργία ενός εκτελέσιμου προγράμματος όπως χαρακτηριστικά εμφανίζεται στο σχήμα 1.2. όπου τρία διαφορετικά πηγαία αρχεία με τη βοήθεια της διαδικασίας της σύνδεσης (Linking) και δύο πρόσθετων βιβλιοθηκών δημιουργούν το τελικό εκτελέσιμο αρχείο του προγράμματος.

1.3. Συναρτήσεις (functions)

Ο γενικός απλός τύπος δήλωσης μιας νέας συνάρτησης στη γλώσσα C, είναι:

όνομα_συνάρτησης (λίστα παραμέτρων) δήλωση

του τύπου των παραμέτρων της λίστας ;

{ εναρκτήριο άγκιστρο με το οποίο αρχίζει η συνάρτηση

. . .

το σώμα της συνάρτησης ; . .

.

} τελικό άγκιστρο με το οποίο τελειώνει η συνάρτηση

Για τη σύνταξη μιας συνάρτησης πρωταρχικό στοιχείο είναι το όνομά της. Μέσα στις παρενθέσεις οι οποίες ακολουθούν το όνομα της συνάρτησης υπάρχει η λίστα των παραμέτρων.

Λέμε **παράμετρο,** μια μεταβλητή ή μια σταθερή τιμή την οποία μπορεί να χρησιμοποιήσει η συνάρτηση κατά τη διάρκεια των πράξεων.

Αμέσως μετά, στην επόμενη σειρά, ακολουθεί η δήλωση της λίστας των παραμέτρων, η οποία ενημερώνει το μεταγλωττιστή για τον τύπο των τιμών των παραμέτρων. Ο τύπος αυτός θα χρησιμοποιηθεί από το μεταγλωττιστή έτσι ώστε να δεσμευτεί ο απαραίτητος χώρος στη μνήμη του υπολογιστή για την ασφαλή και άνευ λαθών εκτέλεση των πράξεων της συνάρτησης.

Οι νέες εκδόσεις της γλώσσας C, συνιστούν ταυτόχρονα με τη δήλωση των μεταβλητών στη λίστα των παραμέτρων να δηλώνεται και ο τύπος τους. Στη συνέχεια, τα άγκιστρα περιβάλλουν το σώμα της συνάρτησης.

Το σώμα της συνάρτησης αποτελείται από εντολές της γλώσσας C οι οποίες ορίζουν τις πράξεις τις οποίες πρέπει να εκτελέσει η συνάρτηση.

Η γλώσσα C έχει και μια ειδική εντολή τη **return**, η οποία αναγκάζει τη συνάρτηση να επιστρέψει μετά την κλήση της, μία και μόνο μία τιμή.

Όταν δεν εμφανίζεται μια εντολή **return** στο σώμα της συνάρτησης τότε η συνάρτηση δεν επιστρέφει καμία τιμή, απλά τελειώνουν οι πράξεις όταν συναντηθεί το τελικό άγκιστρο.

1.3.1. Η συνάρτηση main( )

Η συνάρτηση **main( )** είναι μια πολύ ειδική συνάρτηση της γλώσσας C, επειδή είναι η πρώτη συνάρτηση η οποία θα κληθεί αυτόματα όταν το πρόγραμμα θα ξεκινήσει την εκτέλεσή του στον υπολογιστή (run).

Σε πολλές γλώσσες προγραμματισμού η εκτέλεση των εντολών αρχίζει από την κορυφή (από την πρώτη γραμμή) του προγράμματος, ενώ τα προγράμματα σε γλώσσα C αρχίζουν την εκτέλεση των πράξεων με κλήση στη συνάρτηση **main( ).**

Η συνάρτηση **main( )** μπορεί να βρίσκεται οπουδήποτε μέσα στο συνολικό πρόγραμμα (αρχή, μέση, τέλος) συνήθως όμως για λόγους σαφήνειας είναι η πρώτη στη σειρά ή η τελευταία συνάρτηση. Οπωσδήποτε πρέπει να υπάρχει μια συνάρτηση **main( )** κάπου στο πρόγραμμα ούτως ώστε ο μεταγλωττιστής της γλώσσας C, να μπορεί να αποφασίσει από ποιο σημείο θα αρχίσει τη μεταγλώττιση (μετάφραση) για τη δημιουργία του εκτελέσιμου κώδικα του προγράμματος.

Η συνάρτηση **main( )** έχει ακριβώς τις ίδιες ιδιότητες όπως οποιαδήποτε άλλη συνάρτηση της γλώσσας C, εκτός από το ότι το τελικό άγκιστρό της δείχνει και το τέλος των πράξεων του προγράμματος. Όταν οι πράξεις συναντήσουν το τελικό άγκιστρο, το πρόγραμμα τελειώνει και ο έλεγχος περνά αυτόματα στο λειτουργικό σύστημα.

Προσοχή**, σ' ένα πρόγραμμα πρέπει να υπάρχει **μόνο μια** συνάρτηση **main( ).
*Αν υπάρχουν περισσότερες, το πρόγραμμα δεν μπορεί να διακρίνει από ποια συνάρτηση **main( )** πρέπει να αρχίσει την εκτέλεση των πράξεων. Οι μεταγλωττιστές εντοπίζουν ένα τέτοιο σφάλμα πριν ακόμα φτάσει το στάδιο της εκτέλεσης του προγράμματος και προειδοποιούν με την εμφάνιση μηνύματος.*

1.3.2. Παράμετροι συναρτήσεων (arguments)

Η συνάρτηση **printf( )** στο παράδειγμα του Περιγράμματος 1.1. έχει μια παράμετρο: τη σειρά των χαρακτήρων η οποία θα εμφανιστεί στην οθόνη του υπολογιστή. Οι συναρτήσεις στη γλώσσα C μπορεί να έχουν από καμία μέχρι αρκετές παραμέτρους.

***Σημείωση1.** Το ανώτατο όριο των επιτρεπτών παραμέτρων μιας συνάρτησης προσδιορίζεται από το χρησιμοποιούμενο μεταγλωττιστή και περιγράφεται στο εγχειρίδιο χρήσης του (manual). Η γλώσσα ANSI C αποδέχεται μέχρι 32565 παραμέτρους.*

***Σημείωση2.** Στις συναρτήσεις της γλώσσας C, οι παράμετροι ξεχωρίζουν μεταξύ τους με υποδιαστολές. Η λίστα παραμέτρων (argument list) για τη συνάρτηση add3( ) είναι οι μεταβλητές x, y και z.*

Παράμετρος είναι μια τιμή η οποία μεταφέρεται μέσα στη συνάρτηση. Όταν ορίζεται μια συνάρτηση πρέπει επίσης να ορισθούν και οι μεταβλητές οι οποίες θα δεχθούν τις τιμές των παραμέτρων.

Αυτές οι μεταβλητές ονομάζονται *τυπικές παράμετροι* (formal parameters) της συνάρτησης.

Για παράδειγμα, η συνάρτηση:

```
add3 (x, y, z)      int x, y, z;
                {           return (x + y + z);
            }
```

θα επιστρέψει το άθροισμα των τριών ακέραιων παραμέτρων x, y και z. Η κύρια αποστολή της εντολής **return** είναι να μεταβιβάζει το αποτέλεσμα, εδώ του αθροίσματος των τριών ακεραίων παραμέτρων, από τη συνάρτηση πίσω στο σημείο της κλήσης της.

Όταν καλείται η συνάρτηση **add3( ),** αθροίζονται οι τιμές των τριών παραμέτρων της x, y και z. Όταν οι παράμετροι x, y και z της συνάρτησης λάβουν συγκεκριμένες τιμές τότε ονομάζονται *συγκεκριμένες παράμετροι* (actual parameters). Οι συγκεκριμένες παράμετροι δέχονται τις συγκεκριμένες τιμές όπως ορίζονται κατά την κλήση της συνάρτησης.

Το Περίγραμμα 1.2. παρουσιάζει ένα σύντομο πρόγραμμα το οποίο χρησιμοποιεί τη συνάρτηση **add3()** για τον υπολογισμό του αθροίσματος των τιμών 10, 15 και 7 και στη συνέχεια του αθροίσματος των τιμών 25, 30, και 15.

Οι τιμές των μεταβλητών x1, x2, x3, y1, y2 και y3 δεν τροποποιούνται από την κλήση της συνάρτησης **add3( ).** Η συνάρτηση **add3( )** αθροίζει την πρώτη φορά τις τιμές των μεταβλητών x1, x2 και x3 ενώ τη δεύτερη φορά τις τιμές των μεταβλητών y1, y2 και y3. Με αυτή τη απλή μορφή η συνάρτηση **add3( )** δεν επιστρέφει καμιά πληροφορία στις παραμέτρους της συνάρτησης.

Όταν καλείται μια συνάρτηση οι παράμετροι πρέπει να είναι σταθερές τιμές ή πρέπει να είναι μεταβλητές, όπως στο παράδειγμα της συνάρτησης **add3( ).**

Στη συνάρτηση **add3( )** οι μεταβλητές των παραμέτρων είναι οι x, y και z. Αυτές είναι οι *τυπικές παράμετροι* της συνάρτησης και θα περιέχουν τις πληροφορίες (τις συγκεκριμένες τιμές) όταν κληθεί η συνάρτηση.

Αντίθετα από ότι συμβαίνει σε πολλές άλλες γνωστές γλώσσες προγραμματισμού όπως π.χ. στη FORTRAN, η γλώσσα C δεν αναγνωρίζει το τέλος της γραμμής της εντολής και σαν το τέλος της εντολής.

Αυτό σημαίνει ότι η γλώσσα C δεν έχει περιορισμούς ως αναφορά τη θέση μιας εντολής, γεγονός το οποίο επιτρέπει να ομαδοποιηθούν πολλές εντολές για μεγαλύτερη σαφήνεια και ευελιξία.

Το τέλος μιας εντολής υποδεικνύει το σύμβολο ; (ελληνικό ερωτηματικό).

```
#include <stdio.h>
        main( )
    {
      int p, x1, x2, x3, y1, y2, y3;
 x1=10;          x2=15;
 x3=7;
        p=add3 (x1, x2, x3) ;     /*  1η κλήση της συνάρτησης  */
printf("%d \n", p);          /* εμφάνιση του αποτελέσματος
σε μορφή ακεραίου αριθμού */              y1=25;
y2=30;      y3=15;
        p=add3(y1, y2, y3);      /*  2η κλήση της συνάρτησης  */
        printf("%d", p);          /* εμφάνιση του αποτελέσματος
                                        σε μορφή ακέραιου */
        }
    add3(x, y, z)
int x, y, z;
         {
           return(x + y + z);    /* επιστρέφει το αποτέλεσμα του
                                  αθροίσματος των τριών αριθμών */
        }
```

*Περίγραμμα 1.2. Ένα πρόγραμμα το οποίο χρησιμοποιεί τη συνάρτηση **add3( )***

Π.χ. αν γράψουμε τις εντολές: **x=12;**
y=1; z=-3 add3(x, y, z);

Τότε, το σύνολο αυτών των τεσσάρων γραμμών μπορεί να γραφτεί και με τον ακόλουθο τρόπο:

x=12; y=1; z=-3; add3(x, y, z) ;

Το πρόγραμμα του Περιγράμματος 1.2. θα εμφανίσει μετά την εκτέλεσή του δυο αριθμούς στην οθόνη, τον αριθμό 32 μετά την πρώτη κλήση της συνάρτησης **add3( )** και τον αριθμό 70 μετά από τη δεύτερη κλήση της συνάρτησης **add3( ).**

Προσοχή, *όλες οι εντολές δήλωσης μεταβλητών, όπως και όλες οι εντολές στη γλώσσα C, τελειώνουν με το ελληνικό ερωτηματικό (;).*

1.3.3. Η συνάρτηση printf( )

Η συνάρτηση **printf( )** δεν αποτελεί γνήσιο σύνολο της γλώσσας C**,** είναι μία συνάρτηση βιβλιοθήκης της γλώσσας C, η οποία χρησιμοποιείται για την έξοδο στην οθόνη των αποτελεσμάτων. Ήδη έχουμε δει τον τρόπο της εμφάνισης μίας σειράς χαρακτήρων στην οθόνη του υπολογιστή στο πρόγραμμα του Περιγράμματος 1.1.

Το πρόγραμμα του Περιγράμματος 1.3. εμφανίζει τον αριθμό 715 στην οθόνη.

Όλα τα παραδείγματα των προγραμμάτων του βιβλίου τα οποία θα παράγουν μια έξοδο στην οθόνη, θα χρησιμοποιούν τη συνάρτηση **printf( ).**

```
#include <stdio.h>      main( )
{
  printf("%d \n",715);
}
```

Περίγραμμα 1.3. Ένα πρόγραμμα το οποίο εμφανίζει τον αριθμό 715 στην οθόνη.

Στο πρόγραμμα του Περιγράμματος 1.3. η συνάρτηση **printf( )** καλείται με δύο παραμέτρους.

Η πρώτη παράμετρος, η **"%d",** δείχνει στη συνάρτηση **printf( )** τον τρόπο με τον οποίο θα μεταχειρισθεί τη δεύτερη παράμετρο δηλαδή, την τιμή 715.

Ο τρόπος ή η μορφή εμφάνισης των αποτελεσμάτων λέγεται μορφοποίηση (format).

Το σύμβολο με την ποσοστιαία ένδειξη **%** δείχνει ότι ο επόμενος χαρακτήρας είναι υπόδειξη ελέγχου (προδιαγραφή) δηλαδή, υποδεικνύει τον τρόπο (format) με τον οποίο πρέπει να εμφανιστούν ή να τυπωθούν τα δεδομένα. Το γράμμα **d** σημαίνει ότι τα επόμενα δεδομένα, δηλαδή η τιμή 715, πρέπει να εμφανιστεί σαν ακέραιος αριθμός.

Στη συνάρτηση **printf( )** αν θέλουμε να μορφοποιήσουμε τα δεδομένα, η πρώτη παράμετρος πρέπει πάντα να υποδεικνύει τον τρόπο εμφάνισης των δεδομένων.

Ο γενικός τύπος της συνάρτησης **printf( )** είναι:

printf (σειρά χαρακτήρων ελέγχου, λίστα παραμέτρων);

Με τη σειρά των χαρακτήρων ελέγχου ορίζουμε τις κατάλληλες προδιαγραφές οι οποίες θα χρησιμοποιηθούν από την **printf( )** για την εμφάνιση στην οθόνη των τιμών της λίστας των παραμέτρων.

Η συνάρτηση **printf( )** δέχεται και αναγνωρίζει τις σειρές των χαρακτήρων ελέγχου (προδιαγραφές) οι οποίες εμφανίζονται στον Πίνακα 1.1.

*Πίνακας 1.1. Οι χαρακτήρες ελέγχου της συνάρτησης **printf( )***

Χαρακτήρας	***Αποτέλεσμα***
c	απλός χαρακτήρας
d	ακέραιος
e	επιστημονικός συμβολισμός
f	πραγματικός ή δεκαδικός (Floating point)
g	χρήση των %e ή %f (το συντομότερο)
s	σειρά χαρακτήρων
u	ακέραιος χωρίς πρόσημο
x	εμφάνιση στο δεκαεξαδικό σύστημα

o	εμφάνιση στο οκταδικό σύστημα
b	εμφάνιση στο δυαδικό σύστημα

Η σειρά των χαρακτήρων ελέγχου μπορεί να περιέχει είτε μια σειρά χαρακτήρων οι οποίοι θα εμφανιστούν αυτούσιοι στην οθόνη ή/και οδηγίες (προδιαγραφές) για την εμφάνιση των τιμών των παραμέτρων οι οποίες υπάρχουν στη λίστα των παραμέτρων.

Όταν καλείται για να εκτελεστεί η συνάρτηση **printf( ),** τότε ανιχνεύεται η σειρά των χαρακτήρων ελέγχου από αριστερά προς τα δεξιά της λίστας.

Οι χαρακτήρες οι οποίοι δεν προηγούνται από το σύμβολο της εκατοστιαίας αναλογίας **%** εμφανίζονται στην οθόνη όπως ακριβώς είναι.

Όταν συναντηθεί μία οδηγία (προδιαγραφή ή format), η οποία προηγείται από το σύμβολο της εκατοστιαίας αναλογίας **%**, η συνάρτηση **printf( )** θα θυμηθεί να τη χρησιμοποιήσει για την εμφάνιση της σχετικής παραμέτρου. Το πλήθος των προδιαγραφών στη σειρά των χαρακτήρων ελέγχου, ορίζει πόσες παραμέτρους πρέπει να περιμένει.

Αν θέλουμε να εμφανίσουμε το σύμβολο της εκατοστιαίας αναλογίας **%** σαν χαρακτήρα, πρέπει να χρησιμοποιήσουμε δύο διαδοχικά σύμβολα **%** το ένα δίπλα στο άλλο δηλαδή, να γράψουμε **%%**.

Υπάρχουν δύο περιπτώσεις τις οποίες πρέπει να διακρίνουμε με πολύ μεγάλη προσοχή όταν χρησιμοποιούμε τη συνάρτηση **printf( )** για την εμφάνιση των χαρακτήρων στην οθόνη.

Πρώτον, οι μεμονωμένοι χαρακτήρες οι οποίοι χρησιμοποιούν το χαρακτήρα ελέγχου **%c** πρέπει να περικλείονται από μονά (απλά) εισαγωγικά, π.χ.

'λ', 'A', '+', '*'

Δεύτερον, οι σειρές χαρακτήρων οι οποίες χρησιμοποιούν το χαρακτήρα ελέγχου **%s** πρέπει να βρίσκονται ανάμεσα σε διπλά εισαγωγικά, π.χ.

"this is a string" ή

"Αυτά είναι τα αποτελέσματα του προγράμματος"

Ο Πίνακας 1.2. περιέχει χαρακτηριστικά παραδείγματα χρήσης της συνάρτησης **printf( ).**

*Πίνακας 1.2. Παραδείγματα χρήσης της συνάρτησης **printf( )***

Εντολή	***Αποτελέσματα***
printf("%s , "my home");	my home
printf("My address is%d", 100);	My address is 100
printf("The number %d is integer and the number %f is real.", 27, 16.789328);	The number 27 is integer and the number 16.789328 is real.

printf("%c %s %d-%x",'A',"number in decimal and hex: ",10,10);	A number in decimal and hex: 10-A
printf("%s","HELLO \n WORLD ";	HELLO WORLD

Προσοχή, *κάθε παράμετρος στη λίστα των παραμέτρων μιας συνάρτησης πρέπει να ξεχωρίζει από την προηγούμενη με μία υποδιαστολή.*

Οι προδιαγραφές μπορούν να περιέχουν και τροποποιητές δηλαδή, εκφράσεις οι οποίες θα ορίζουν τον τρόπο της εμφάνισης του ακέραιου και του δεκαδικού μέρους ενός πραγματικού αριθμού και στην περίπτωση των σειρών χαρακτήρων, θα ανιχνεύουν τα κενά.

Αυτές οι λεπτομέρειες, όπως και πολλές άλλες, για τις δυνατότητες της συνάρτησης εξόδου **printf( )** θα αναλυθούν στο 2ο κεφάλαιο του βιβλίου (§ 2.9.3) όπου θα αναπτυχθούν αναλυτικά όλες οι δυνατότητες εισόδου και εξόδου των δεδομένων.

Προσοχή, *πρέπει πάντα να υπάρχει το ίδιο πλήθος παραμέτρων και προδιαγραφών στη σειρά των χαρακτήρων ελέγχου της συνάρτησης εξόδου* ***printf( )***.

1.4. Μεταβλητές

Το όνομα μιας μεταβλητής στη γλώσσα C μπορεί να αποτελείται από ένα χαρακτήρα μέχρι και αρκετούς χαρακτήρες. Δεν υπάρχει προκαθορισμένο ανώτερο όριο για το πλήθος των χαρακτήρων του ονόματος μιας μεταβλητής, εν τούτοις μόνο οι 32 πρώτοι χαρακτήρες είναι γενικά σημαντικοί.

Συνιστάται να μη χρησιμοποιούμε πολλούς χαρακτήρες για να δηλώσουμε το όνομα μιας μεταβλητής. Συνήθως ένα όνομα μιας μεταβλητής αποτελείται από ένα μέχρι το πολύ 10 χαρακτήρες.

Έτσι, το όνομα της μεταβλητής πρέπει να αρχίζει υποχρεωτικά με ένα γράμμα της λατινικής αλφαβήτου ή με το χαρακτήρα υπογράμμισης δηλαδή, το _ (underscore). Στη συνέχεια, μπορούν να εμφανίζονται: γράμματα ή αριθμοί και η υπογράμμιση.

Η υπογράμμιση μπορεί να χρησιμοποιηθεί για να επιτείνει τη δυνατότητα ανάγνωσης και αναγνώρισης του ονόματος μιας μεταβλητής

π.χ. my_name, The_greater_number.

Αντίθετα απ' ότι ισχύει σε πολλές άλλες γλώσσες προγραμματισμού, όπως π.χ. στη γλώσσα FORTRAN όπου δεν υπάρχει διάκριση ανάμεσα σε πεζά και κεφαλαία γράμματα, τα κεφαλαία γράμματα διαφέρουν από τα μικρά γράμματα (πεζά), όταν χρησιμοποιούνται σε μεταβλητές της γλώσσας C.

Έτσι, οι λέξεις **count** και **COUNT** είναι ξεχωριστές λέξεις όπως και οι λέξεις **Top** και **top.**

Οι χαρακτήρες οι οποίοι χρησιμοποιούνται για να δηλώσουν το όνομα μιας μεταβλητής στη γλώσσα C, χαρακτηρίζονται ως *ευαίσθητοι* (case sensitive) επειδή

κάθε χαρακτήρας είναι μοναδικός και διαφορετικός από τον αντίστοιχο κεφαλαίο ή πεζό χαρακτήρα.

Μερικά παραδείγματα αποδεκτών ονομάτων στη γλώσσα C, ως ονόματα μεταβλητών είναι:

find, Color
Add_sum
Telos top_of_file name23
_temp1
Mmmm

Δεν μπορούμε να χρησιμοποιήσουμε σαν όνομα μιας μεταβλητής καμία από τις δεσμευμένες λέξεις της γλώσσας C. *Δεσμευμένες λέξεις* είναι οι λέξεις οι οποίες ανήκουν στο λεξιλόγιο της γλώσσας προγραμματισμού C και τις οποίες θα μελετήσουμε διεξοδικά στην επόμενη παράγραφο.

Πίνακας 1.3. Οι ενσωματωμένοι τύποι δεδομένων και οι λέξεις κλειδιά

Τύπος δεδομένων	***Η αντιστοιχία στη C***
χαρακτήρας	**char**
ακέραιος	**int**
βραχύς ακέραιος	**short int**
μακρύς ακέραιος	**long int**
ακέραιος χωρίς πρόσημο	**unsigned int**
Πραγματικός αριθμός	**float**
Πραγματικός αριθμός διπλής ακρίβειας	**double**

Επίσης, θα πρέπει να φροντίζουμε να μη δίνουμε (να αντιστοιχούμε) στις δικές μας συναρτήσεις (αυτές που θα δημιουργήσουμε) το ίδιο όνομα το οποίο έχουν οι συναρτήσεις της γλώσσας C. Εξ ορισμού, μια νέα συνάρτηση την οποία συντάσσει ένας προγραμματιστής για να χρησιμοποιήσει στο πρόγραμμά του, έχει προτεραιότητα σε σχέση με τις άλλες συναρτήσεις οι οποίες βρίσκονται στις βιβλιοθήκες της γλώσσας C και έχουν το ίδιο όνομα.

Αν χρησιμοποιηθεί το ίδιο όνομα, μπορεί να δημιουργήσει προβλήματα τόσο σε επίπεδο προγράμματος-προγραμματιστή, όσο και σε κάποιους μεταγλωττιστές και για αυτό το λόγο πρέπει να αποφεύγεται ακριβώς η ίδια ονομασία.

Π.χ. αν θέλουμε να δώσουμε το όνομα **printf** σε μια νέα δική μας συνάρτηση θα πρέπει να γράψουμε: **new_printf** ή **printfn** κ.τ.λ.

Πέρα από αυτούς τους περιορισμούς, ο σωστός προγραμματισμός απαιτεί, χωρίς να είναι υποχρεωτικό, να χρησιμοποιούμε ονόματα μεταβλητών τα οποία θα αντικατοπτρίζουν τη σημασία ή τη χρήση των μεταβλητών.

Έτσι, όταν θέλουμε να χρησιμοποιήσουμε μια μεταβλητή η οποία θα εκφράζει θερμοκρασίες μπορούμε να γράψουμε:

count_temperature *ή*
t_counter *ή*
TempCount

Η γλώσσα ANSI C (C89) έχει ενσωματωμένους επτά βασικούς τύπους δεδομένων (Πίνακας 1.3), οι οποίοι έχουν τις αντίστοιχες λέξεις κλειδιά στη γλώσσα C.

Εκτός από αυτούς τους προκαθορισμένους τύπους μπορούμε επίσης, να δημιουργήσουμε ομάδες νέων τύπων μεταβλητών γνωστές ως *δομή* (structure) και *ένωση* (union) τις οποίες θα μελετήσουμε αναλυτικά στο 7ο κεφάλαιο. Επίσης, με την έκδοση **C99** έχει προστεθεί ο τύπος **long long** τον οποίο θα μελετήσουμε αναλυτικά στο 2ο κεφάλαιο.

Οι μεταβλητές οι οποίες θα είναι γνωστές και θα μπορούν να χρησιμοποιηθούν σε όλες τις συναρτήσεις ενός προγράμματος λέγονται *γενικές μεταβλητές* ή *εξωτερικές μεταβλητές* ή ακόμη και *καθολικές μεταβλητές* (global variables).

Οι μεταβλητές οι οποίες θα είναι γνωστές μόνο στη συνάρτηση η οποία τις χρησιμοποιεί ονομάζονται *τοπικές μεταβλητές* ή *αυτόματες μεταβλητές* (local variables).

Βασικός κανόνας του δομημένου προγραμματισμού. *Όλες οι μεταβλητές πρέπει να δηλωθούν πριν να χρησιμοποιηθούν.*

Η διαδικασία δήλωσης δεν ενημερώνει μόνο το μεταγλωττιστή ποιο θα είναι το όνομα της μεταβλητής αλλά ορίζει και τον τύπο των δεδομένων.

Για παράδειγμα, αν θέλουμε να δηλώσουμε τις μεταβλητές **max**, και **min** σαν ακέραιες, θα πρέπει να γράψουμε :

```
int max, min;
```

Στην αρχή τοποθετούμε τον επιθυμητό τύπο της μεταβλητής και μετά παραθέτουμε τη λίστα των μεταβλητών οι οποίες θέλουμε να ανήκουν σε αυτό τον τύπο.

Οι μεταβλητές μπορούν να δηλωθούν σε οποιοδήποτε σημείο του προγράμματος. Συνήθως, δηλώνονται στην αρχή μιας συνάρτησης, ακριβώς μετά το αρχικό άγκιστρο.

Για παράδειγμα, στη συνάρτηση **test**, δηλώνεται μια μεταβλητή από κάθε έναν τύπο:

```
symbol_test ( )
 {          int max;          char
character;          float value;
short int top1;          long int top2;
double big_number;
unsigned int telos;
        . . .
        . . .
/*  Οι υπόλοιπες εντολές της συνάρτησης */
        . . .
        . . .
       }
```

Στο παράδειγμα αυτό, οι μεταβλητές θα είναι γνωστές μόνο στη συνάρτηση **symbol_test ( )** δηλαδή, είναι τοπικές μεταβλητές για τη συγκεκριμένη συνάρτηση.

Αν θέλουμε να χρησιμοποιήσουμε μια μεταβλητή σε κάποια συνάρτηση ενός προγράμματος δηλαδή, η τιμή της μεταβλητής αυτής να είναι γνωστή και αξιοποιήσιμη σε όλες τις συναρτήσεις του προγράμματος πρέπει να τη δηλώσουμε έξω από κάθε άλλη συνάρτηση του προγράμματος. Στην περίπτωση αυτή η μεταβλητή αυτή θα είναι γενική μεταβλητή.

Όταν σε ένα πρόγραμμα, θέλουμε να χρησιμοποιείται η ακέραια μεταβλητή **counter** σαν γενική μεταβλητή δηλαδή, η μεταβλητή αυτή να ισχύει κατά τη διάρκεια ολόκληρου του προγράμματος (σε οποιοδήποτε σημείο του), πρέπει να δηλωθεί η μεταβλητή πριν ακόμη ορίσουμε τη συνάρτηση **main( ).**

Π.χ. γράφουμε: **int counter;**

```
main ( ) {
 . . . . .
  counter = 36;
     . . . . .          /* Εντολές του προγράμματος  */

  printf("Μετρητής = %d", counter);
}

  test1( ) {  . . . . .
. . . . .
  printf("Μετρητής = %d", counter);
. . . . .
. . . . .    /*  Εντολές της συνάρτησης test1  */          }

  test2( ) {
 . . . . .
  printf("Μετρητής = %d", counter);
. . . . .
. . . . .     /*  Εντολές της συνάρτησης test2  */
     . . . . .
}
```

Στο παράδειγμα αυτό, η μεταβλητή **counter** είναι γενική και θα είναι γνωστή σε όλες τις συναρτήσεις του προγράμματος.

Κάθε νέα τιμή της μεταβλητής **counter** μεταφέρεται αυτόματα σε κάθε άλλη συνάρτηση η οποία τη χρησιμοποιεί δηλαδή, και οι τρεις κλήσεις στη συνάρτηση **printf()** θα εμφανίσουν την ίδια ακριβώς τιμή, την οποία έχει λάβει η μεταβλητή κατά την έναρξη της εκτέλεσης της συνάρτησης **main( ).**

***Προσοχή**, δεν πρέπει να χρησιμοποιούνται γράμματα της ελληνικής αλφαβήτου (κεφαλαία ή πεζά) ανάμεσα στους χαρακτήρες ορισμού του ονόματος μιας μεταβλητής της γλώσσας C, παρά μόνο λατινικοί χαρακτήρες.*

1.5. Δεσμευμένες λέξεις της γλώσσας C

Η λεπτομερής περιγραφή της γλώσσα C, αποτελεί κύριο στόχο αυτού του βιβλίου και βασίζεται στην τυποποίηση του ANSI X3.159 δηλαδή, στη διεθνή τυποποίηση ISO/IEC 9899.1990 η οποία είναι γνωστή και ως **Standard C89**.

Οι δεσμευμένες λέξεις της έκδοσης **C89**, εμφανίζονται στον Πίνακα 1.4. Οι δεσμευμένες λέξεις της γλώσσας C λέγονται και *λέξεις-κλειδιά* (keywords). Οι λέξεις αυτές αποτελούν το βασικό λεξιλόγιο για τη δημιουργία των εντολών της γλώσσας C. Όταν συνδυασθούν με το τυπικό συντακτικό της γλώσσας C, σχηματίζουν τη γλώσσα προγραμματισμού C.

Πίνακας 1.4. Κατάλογος των δεσμευμένων λέξεων της γλώσσας C89

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**signed**	**unsigned**
continue	**for**	**sizeof**	**void**
default	**goto**	**short**	**volative**
do	**if**	**static**	**while**

Η έκδοση **C89** αναγνωρίζεται από όλους τους διαθέσιμους μεταγλωττιστές της γλώσσας C ενώ οι νέες επεκτάσεις της ANSI **C99** δεν έχουν ενσωματωθεί σε όλους τους μεταγλωττιστές της γλώσσας C .Η τυποποίηση **C99** περιέχει εκτός των άλλων και τις πέντε νέες δεσμευμένες λέξεις:

inline, restrict, _Bool, _Complex, _Imaginary

Προσοχή, *οι δεσμευμένες λέξεις χρησιμοποιούνται μόνο με πεζά γράμματα. Για παράδειγμα, η λέξη* ***RETURN*** *δεν είναι η ίδια με τη λέξη-κλειδί* ***return****.*

1.6. Κλιμακωτές εντολές και σχόλια

Σε όλα τα παραδείγματα των προγραμμάτων του βιβλίου, ορισμένες εντολές αρχίζουν πιο μέσα από την προηγούμενη γραμμή. Αυτό δεν είναι τυχαίο. Η γλώσσα C έχει μεγάλη ελευθερία σε αυτό το θέμα επειδή δεν έχει σημασία σε ποιο σημείο της γραμμής τοποθετούμε τις εντολές.

Με την πάροδο του χρόνου έχει εξελιχθεί και έχει διαμορφωθεί ένας κοινά παραδεκτός τύπος αρχής παραγράφου στο δομημένο προγραμματισμό και σχεδόν έχει τυποποιηθεί. Αυτός ο απλός τρόπος γραφής των εντολών ενός προγράμματος καθιστά τον πηγαίο κώδικα πιο ευανάγνωστο και πιο προσιτό στις μεταβολές, τους ελέγχους καθώς και στις διορθώσεις από τον προγραμματιστή.

```
#include <stdio.h>     /* Ενσωμάτωση Βιβλιοθήκης  */     main()
/* Αρχή του προγράμματος */
  {
        /* Αυτό το πρόγραμμα εμφανίζει τη λέξη
HELLO WORLD στην οθόνη */
             printf ("HELLO WORLD\n");    /*  Εντολή εξόδου */
   }     /* Εδώ τελειώνει το πρόγραμμα */
```

Περίγραμμα 1.4. Το πρόγραμμα HELLO με σχόλια

Αυτός ο άτυπος τύπος γραφής των προγραμμάτων ακολουθείται σε όλα τα παραδείγματα και συστήνεται να εφαρμόζεται από όλους όσους θέλουν να γράφουν σωστά και αποδοτικά προγράμματα στη γλώσσα C.

Χρησιμοποιώντας αυτόν τον άτυπο τύπο γραφής, γράφουμε πιο μέσα και παρακάτω από το αρχικό άγκιστρο κλιμακωτά και συνεχίζουμε αρχίζοντας κάθε νέα γραμμή της ίδιας ενότητας, στην ίδια κάθετο. Η λογική ενότητα τελειώνει με την τοποθέτηση του τελικού άγκιστρου στην ίδια κατακόρυφη ευθεία με το αρχικό άγκιστρο.

Υπάρχουν μερικές εντολές της γλώσσας C οι οποίες ενθαρρύνουν από τη φύση τους περισσότερες βαθμίδες δηλαδή, η αρχή της νέας γραμμής να τοποθετείται πιο μέσα από την προηγούμενη γραμμή.

Στη γλώσσα C τα σχόλια του προγραμματιστή μπορούν να τοποθετηθούν οπουδήποτε στο πρόγραμμά του και προσδιορίζονται από δυο χαρακτηριστικά σημάδια. Το σημάδι αρχής σχολίου είναι το ζεύγος των χαρακτήρων ***/**** και το σημάδι τέλους σχολίου είναι το ζεύγος των χαρακτήρων ****/***. Τα σχόλια μπορούν να επεκταθούν και σε περισσότερες από μία γραμμές του κώδικα.

Προσοχή, *η χρήση των δύο διαδοχικών πλάγιων γραμμών // ισχύει μόνο από την έκδοση της γλώσσας* ***C99.***

Επίσης, σχόλια μπορούν να τοποθετηθούν σε μια μόνο γραμμή αμέσως μετά την παρουσία δύο διαδοχικών πλάγιων γραμμών ***//*** (slash).

Τα σωστά τοποθετημένα σχόλια βοηθούν σημαντικά τόσο τον ίδιο τον προγραμματιστή όσο και τους συνεχιστές ή αναγνώστες του πηγαίου κώδικα του προγράμματος κυρίως στη γρήγορη κατανόηση του αλγόριθμου ή των ιδιαιτεροτήτων οι οποίες έχουν εφαρμοστεί στο συγκεκριμένο σημείο του προγράμματος και συντελούν στην αποφυγή εσφαλμένων επιλογών και μελλοντικών λαθών.

Η πρόσθεση σχολίων στο πρόγραμμα του Περιγράμματος 1.1. εμφανίζεται στο Περίγραμμα 1.4.

```
/* Η συνάρτηση mul υπολογίζει το γινόμενο δύο αριθμών */
mul(x,y)            int x,y;        // Δήλωση του τύπου των
παραμέτρων
        {
           return(x*y);             /* επιστρέφει το αποτέλεσμα του
                                       γινομένου των δύο παραμέτρων
της συνάρτησης */
         }  /* Τέλος της συνάρτησης */
```

*Περίγραμμα 1.5. Η συνάρτηση **mul( )** με σχόλια*

Το Περίγραμμα 1.5. υποδεικνύει το σωστό τρόπο συγγραφής μιας συνάρτησης για τον υπολογισμό του γινομένου δύο αριθμών με στόχο την αποθήκευσή της σε κάποιο βοηθητικό χώρο για μελλοντική χρήση. Η προσθήκη των σχολίων καθιστά τη συνάρτηση κατανοητή από όλους τους μελλοντικούς χρήστες οι οποίοι θα μπορούν πολύ εύκολα να προχωρήσουν σε μικρές ή μεγάλες αλλαγές βασιζόμενοι στην αρχική μορφή της συνάρτησης.

Τα σχόλια σε ένα πρόγραμμα, δεν επιβαρύνουν το μέγεθος του εκτελέσιμου κώδικα επειδή ο μεταγλωττιστής πριν από τη δημιουργία του αντικειμενικού κώδικα αντικαθιστά κάθε γραμμή σχολίου με ένα κενό διάστημα (blanc).