ΕΘΝΙΚΟΝ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟΝ ΠΑΝΕΠΙΣΤΗΜΙΟΝ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΓΕΩΛΟΓΙΑΣ ΚΑΙ ΓΕΩΠΕΡΙΒΑΛΛΟΝΤΟΣ
ΤΟΜΕΑΣ ΓΕΩΦΥΣΙΚΗΣ ΚΑΙ ΓΕΩΘΕΡΜΙΑΣ

«ΠΛΗΡΟΦΟΡΙΚΗ» (ΣΜ002)

Γλώσσα C++ & Τεχνικές Προγραμματισμού

Θεωρία & Ασκήσεις

Δρ. Σπ. Βασιλοπούλου

Δρ. Β. Σακκάς (Επικ. Καθηγητής)

2025

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

Γλώσσα C++ & Τεχνικές Προγραμματισμού

Α. ΘΕΩΡΙΑ

Εγκατάσταση C++

DEV C++ : από www.bloodshed.net, και http://download.cnet.com/Dev-C/3000-2069_4-12686.html.

Codeblocks : από https://www.codeblocks.org/#google_vignette

ΠΛΗΡΟΦΟΡΙΚΗ & ΓΕΩΕΠΙΣΤΗΜΕΣ

Πληροφορική είναι η επιστήμη που ασχολείται με τη συλλογή, την επεξεργασία, την αποθήκευση και την μετάδοση της πληροφορίας. Πλέον επιτυγχάνεται μέσω των ηλεκτρονικών υπολογιστών.
Η **Πληροφορική** απαραίτητη στην **διαχείριση γεωδεδομένων & επίλυση προβλημάτων.**
Γεωπληροφορική είναι η επιστήμη η οποία μέσω τεχνολογιών και τεχνικών πληροφορικής σε συνδυασμό με την θεωρία και τις μεθόδους της γεωδαισίας, φωτογραμμετρίας, τηλεανίχνευσης και την ανάλυση χώρου έχει στόχο την επίλυση προβλημάτων σχετικών με την συλλογή, διαχείριση, επεξεργασία, ανάλυση και απεικόνιση ψηφιακών χωρικών δεδομένων.
Μεγάλες δυνατότητες διαχείρισης και επεξεργασίας της πληροφορίας δίνουν τα **Συστήματα Γεωγραφικών πληροφοριών (ΣΓΠ) ή Geographical Information Systems (GIS).**
Στόχος:

- **Διαχείριση δεδομένων**
- **Εξαγωγή Πληροφορίας**
- **Απάντηση Ερωτημάτων**
- **Αντιμετώπιση Προβλημάτων**
- **Συστήματα Λήψεως Αποφάσεων**

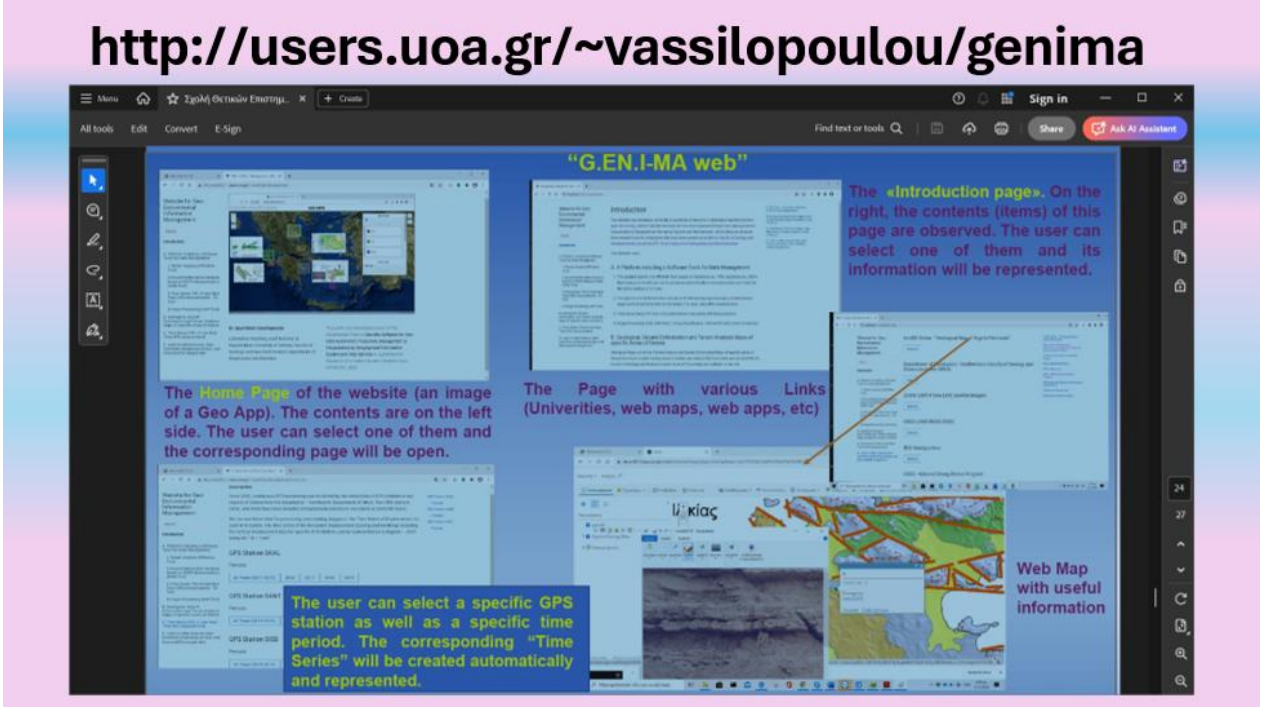
Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

ΑΝΤΙΜΕΤΩΠΙΣΗ ΠΡΟΒΛΗΜΑΤΩΝ

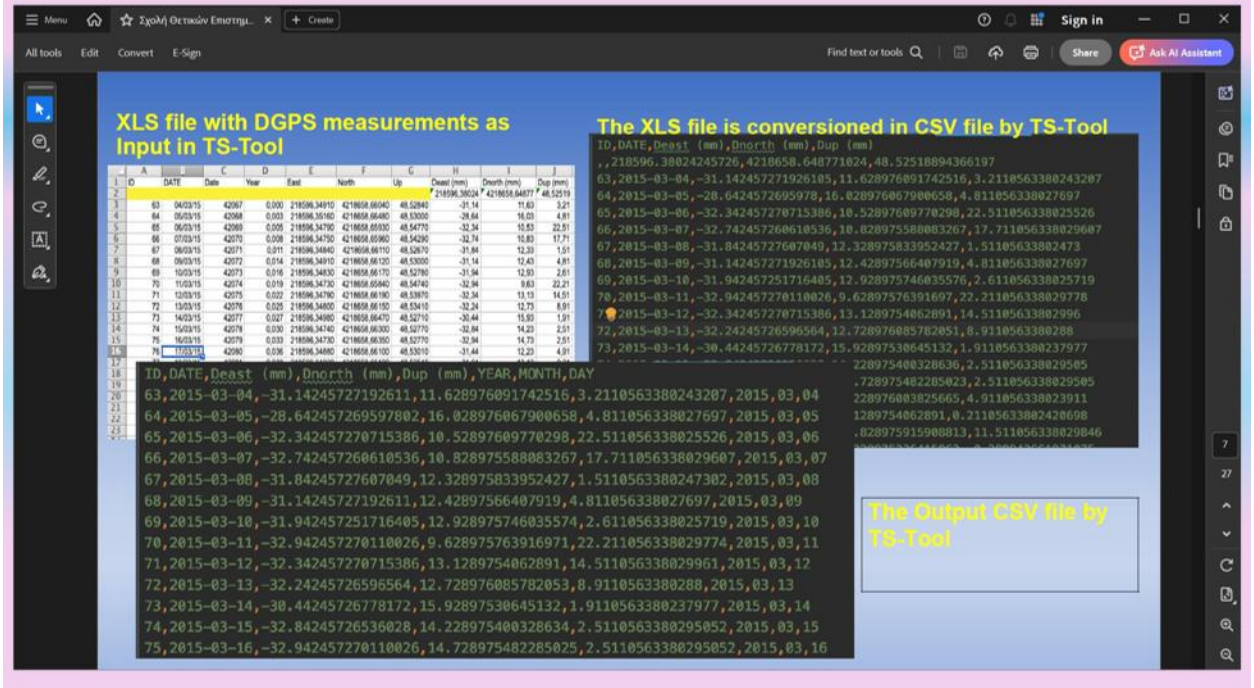
Προκειμένου να αντιμετωπισθεί ένα πρόβλημα απαιτείται αρχικά η κατανόηση του προβλήματος - η ανάλυση – ο σχεδιασμός της επίλυσης, η απόδοσή του σε γλώσσα υπολογιστή, η επαλήθευση και η καταγραφή εμπειρίας για επόμενη χρήση.

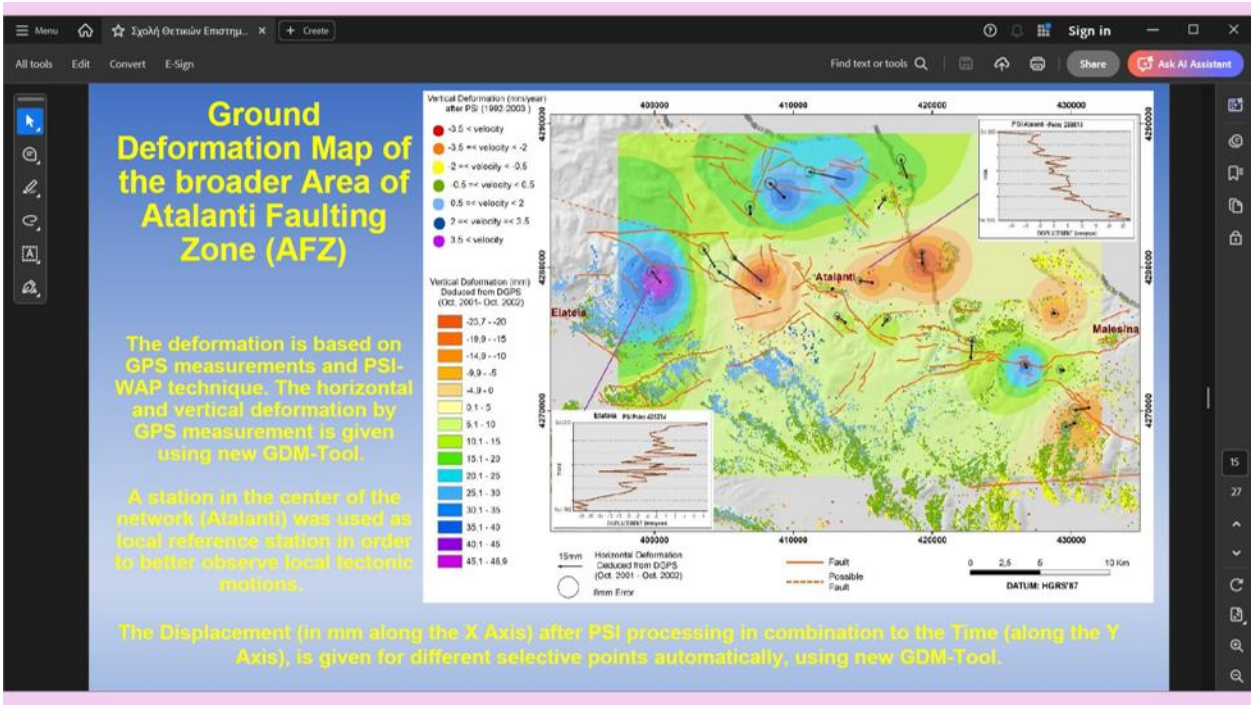
ΑΥΤΟ ΣΗΜΑΙΝΕΙ

Προσαρμογή του προβλήματος, δηλαδή όλων των διαδικασιών που ακολουθούνται με την κλασική μεθοδολογία (χαρτί και μολύβι) σε γλώσσα υπολογιστή. Αυτό γίνεται μέσα από την επιλογή και τον τύπο των δεδομένων, την επιλογή και την τιμή συγκεκριμένων παραμέτρων, την ανάπτυξη - ανάλυση – διαχείριση της βάσης δεδομένων και τα βήματα – διαδικασίες που θα ακολουθηθούν για την εξαγωγή αποτελεσμάτων με αυτοματοποιημένο τρόπο : **ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ**

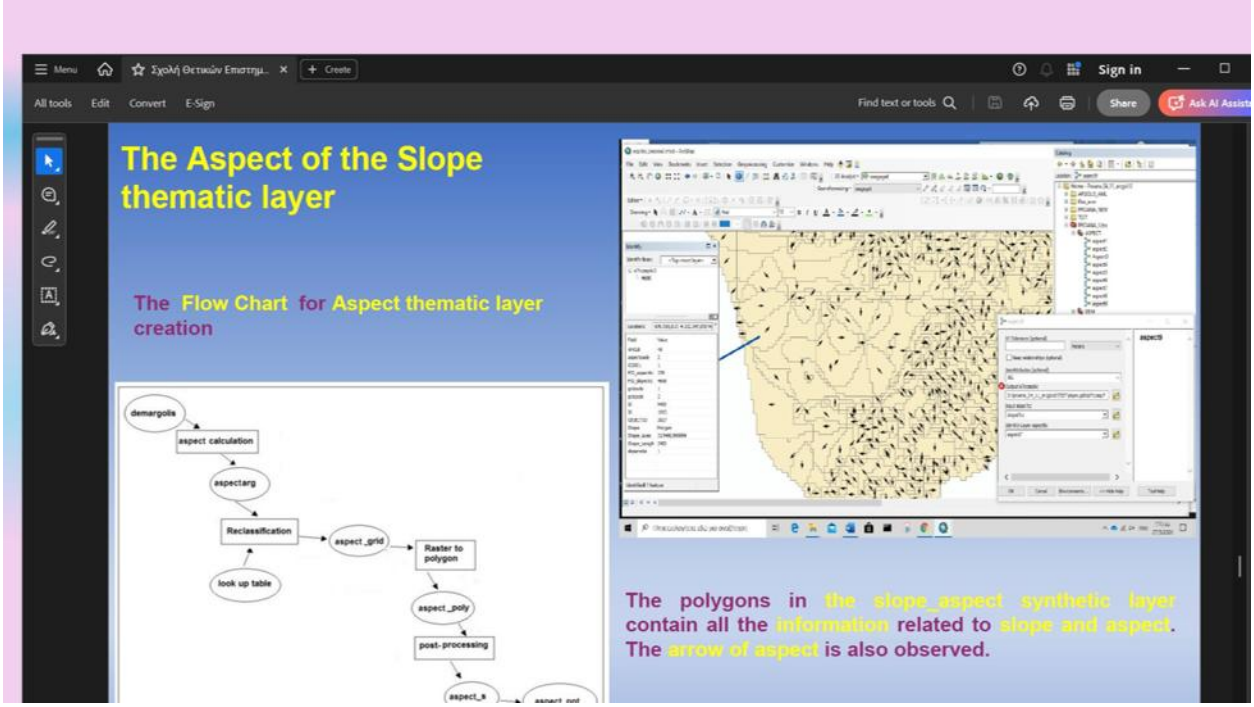


Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.





Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.



ΑΛΓΟΡΙΘΜΟΣ - ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Η περιγραφή της διαδικασίας επίλυσης ενός προβλήματος αποτελεί τον **αλγόριθμο.**

Ο **προγραμματισμός** είναι ο **μετασχηματισμός του αλγορίθμου** σε μορφή κατανοητή από τον ηλεκτρονικό υπολογιστή.

Το **πρόγραμμα** αποτελείται από ένα **σύνολο εντολών** (commands), οι οποίες αποτελούν τις οδηγίες προς τον υπολογιστή.

Το αποτέλεσμα του προγραμματισμού είναι το **λογισμικό** (software).

Χαρακτηριστικά Αλγορίθμου

Τα πέντε κυριότερα χαρακτηριστικά ενός σωστού αλγορίθμου κατά Knuth, 1973 είναι:

Είναι πεπερασμένος. Ένας αλγόριθμος αποτελείται από πεπερασμένο αριθμό βημάτων, αλλά και πρέπει πάντα να τερματίζει μετά από ένα πεπερασμένο αριθμό βημάτων.

Είναι καλά ορισμένος. Κάθε βήμα ενός αλγορίθμου πρέπει να είναι επακριβώς ορισμένο και οι ενέργειες που θα εκτελεστούν σε αυτό πρέπει να περιγράφονται ρητά και αδιαμφισβήτητα.

Είσοδος. Κάθε αλγόριθμος δέχεται μία ή περισσότερες εισόδους. Αφορούν τα δεδομένα που εισέρχονται σε αυτόν πριν αρχίσουν οι κυρίως υπολογισμοί.

Έξοδος. Κάθε αλγόριθμος παράγει μία ή περισσότερες εξόδους. Πρόκειται για πληροφορίες που έχουν μια καθορισμένη σχέση με τις εισόδους.

Αποτελεσματικότητα. Για να χαρακτηριστεί ένας αλγόριθμος ως αποτελεσματικός, πρέπει όλες οι ενέργειες που περιλαμβάνει να είναι αρκετά απλές ώστε να μπορούν να εκτελεσθούν με ακρίβεια σε πεπερασμένο χρόνο από έναν άνθρωπο που χρησιμοποιεί χαρτί και μολύβι.

Μέρη Αλγορίθμου

Κεφαλή (header) αλγορίθμου: περιλαμβάνει το όνομα του αλγορίθμου και τον κατάλογο των δεδομένων εισόδου / εξόδου. Δηλώνεται με την λέξη ΑΛΓΟΡΙΘΜΟΣ

Τμήμα δηλώσεων δεδομένων: Περιλαμβάνει τα δεδομένα, τα οποία ο αλγόριθμος χρησιμοποιεί εσωτερικά. Δηλώνεται με την λέξη ΔΕΔΟΜΕΝΑ. Με αυτά εκτελούνται οι πράξεις, που περιγράφει ο αλγόριθμος.

Κυρίως τμήμα αλγορίθμου (body): περιγράφονται οι πράξεις που εκτελούνται με τα συγκεκριμένα δεδομένα. Η αρχή του τμήματος δηλώνεται με την λέξη ΑΡΧΗ, το τέλος του, με την λέξη ΤΕΛΟΣ.

Μεταβλητές – Σταθερές

Οι υπολογιστές επεξεργάζονται διάφορους τύπους δεδομένων και αποθηκεύουν τιμές.
Τα δεδομένα είναι αριθμοί ή χαρακτήρες.

Π.χ. Η C για την αποθήκευση των δεδομένων χρησιμοποιεί variables (μεταβλητές) και constants (σταθερές).

Η **μεταβλητή** είναι μια περιοχή μνήμης όπου η τιμή που περιέχει μπορεί να αλλάξει κατά τη διάρκεια του προγράμματος.

Η **σταθερά** είναι χώρος μνήμης, όπου η τιμή δεν αλλάζει.

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

Εργαλεία Έκφρασης Αλγορίθμων

Ψευδοκώδικας: Το σημαντικότερο εργαλείο λεκτικής περιγραφής αλγορίθμων. Βασίζεται στη γλώσσα.

Διαγράμματα Ροής: Εργαλεία που βασίζονται σε σχήματα.

Ψευδοκώδικας

Ο **ψευδοκώδικας** μοιάζει με **κώδικα προγραμματισμού**, **αλλά δεν είναι.**

Ο ψευδοκώδικας είναι μια δομημένη γλώσσα που χρησιμοποιεί:

- στοιχεία (συντακτικές δομές) από τις γλώσσες προγραμματισμού
- συμβολισμούς (σημασιολογικά στοιχεία) από τα Μαθηματικά και την Μαθηματική Λογική
- στοιχεία (λεκτικές περιγραφές) από τη φυσική γλώσσα.

Χαρακτηριστικά Ψευδοκώδικα

Ο ψευδοκώδικας συνδυάζει την αυστηρότητα και την ακρίβεια των γλωσσών προγραμματισμού, με την εκφραστική δύναμη της φυσικής γλώσσας

Είναι ανεξάρτητος από τις γλώσσες προγραμματισμού

Εξαρτάται από τη φυσική γλώσσα

Δεν έχει μια τυποποιημένη διάλεκτο

Ο αλγόριθμος που περιγράφεται με ψευδοκώδικα δύναται να γίνει πρόγραμμα σε οποιαδήποτε γλώσσα

Γίνεται κατανοητός από προγραμματιστές αλλά και μη προγραμματιστές
Έχει περιορισμένες εκφραστικές δυνατότητες ως προς τα διαγράμματα ροής

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

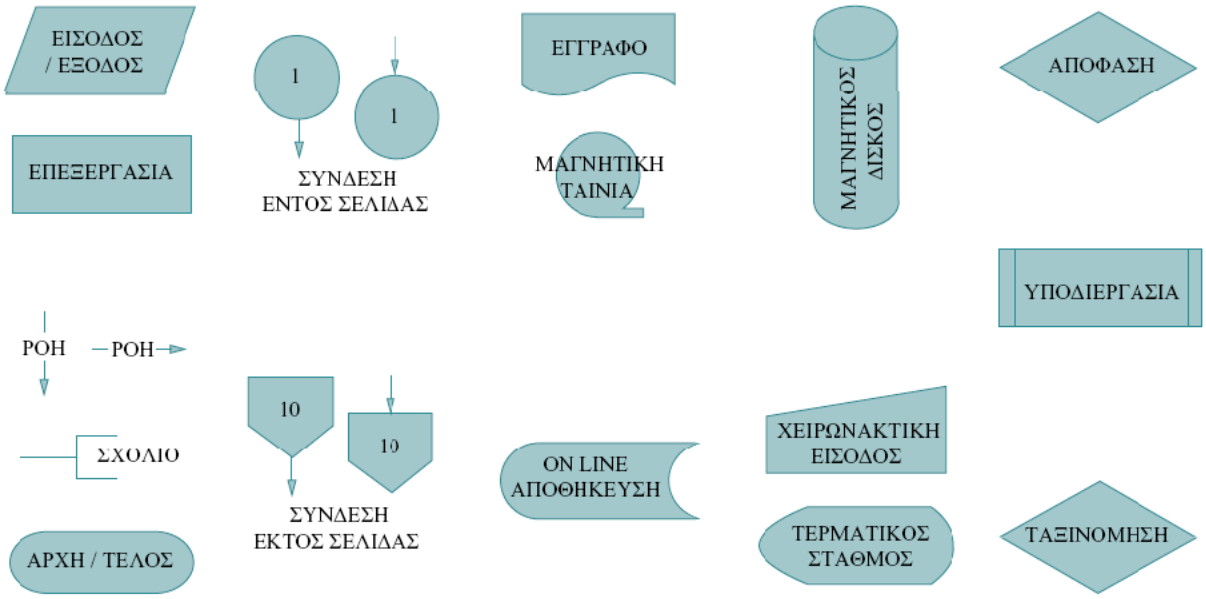
```
ΑΛΓΟΡΙΘΜΟΣ MAX-XYZ
ΔΕΔΟΜΕΝΑ
        X,Y,Z: INTEGER;
ΑΡΧΗ
        ΔΙΑΒΑΣΕ(X,Y,Z);
        ΕΑΝ (X > Y) ΤΟΤΕ
                ΕΑΝ (X > Z) ΤΟΤΕ
                        ΤΥΠΩΣΕ(X)
                ΑΛΛΙΩΣ
                        ΤΥΠΩΣΕ(Z)
                ΕΑΝ-ΤΕΛΟΣ
        ΑΛΛΙΩΣ
                ΕΑΝ (Y > Z) ΤΟΤΕ
                        ΤΥΠΩΣΕ(Y)
                ΑΛΛΙΩΣ
                        ΤΥΠΩΣΕ(Z)
                ΕΑΝ-ΤΕΛΟΣ
        ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ
```

Ψευδοκώδικας για εύρεση του μεγαλύτερου αριθμού από 3 ακ'εραιους αριθμούς. (Καμμέας Α. - Τεχνικές Προγραμματισμού, τ.β', ΕΑΠ, Πάτρα, 2000)

Αρχή ομάδας οδηγιών	ΑΡΧΗ
Τέλος ομάδας οδηγιών	ΤΕΛΟΣ
Καταχώρηση σε μεταβλητή	:=
Διαχωριστής εντολών	;
Είσοδος δεδομένων από συσκευή (προαιρετικά)	ΔΙΑΒΑΣΕ (λίστα ονομάτων μεταβλητών εισόδου) [ΑΠΟ πηγή δεδομένων]
Εξοδος δεδομένων στην οθόνη	ΤΥΠΩΣΕ (μεταβλητές εξόδου, "κείμενο")
Εξοδος δεδομένων στον εκτυπωτή	ΕΚΤΥΠΩΣΕ (μεταβλητές, "κείμενο")

Εξοδος δεδομένων σε άλλη συσκευή	ΓΡΑΨΕ (μεταβλητές, "κείμενο") ΣΕ συσκευή
Απόφαση	ΕΑΝ (συνθήκη) ΤΟΤΕ οδηγία ή ομάδα οδηγιών [ΑΛΛΙΩΣ οδηγία ή ομάδα οδηγιών] ΕΑΝ-ΤΕΛΟΣ
Επανάληψη με μετρητή / δείκτη	ΓΙΑ μετρητής = αρχική τιμή ΕΩΣ τελική τιμή ΕΠΑΝΕΛΑΒΕ οδηγία ή ομάδα οδηγιών ΓΙΑ-ΤΕΛΟΣ
Επανάληψη με συνθήκη στην αρχή	ΕΝΟΣΩ (συνθήκη) ΕΠΑΝΕΛΑΒΕ οδηγία ή ομάδα οδηγιών ΕΝΟΣΩ-ΤΕΛΟΣ
Επανάληψη με συνθήκη στο τέλος	ΕΠΑΝΕΛΑΒΕ οδηγία ή ομάδα οδηγιών ΜΕΧΡΙ (συνθήκη)
Κλήση υποπρογράμματος με μεταβλητές εισόδου (προαιρετικά)	ΥΠΟΛΟΓΙΣΕ όνομα υποπρογράμματος [(λίστα μεταβλητών)]

Δομές και Λέξεις Ψευδοκώδικα (Καμμέας Α. - Τεχνικές Προγραμματισμού, τ.β', ΕΑΠ, Πάτρα, 2000)



Μερικά από τα βασικά κατά ANSI σύμβολα ενός Διαγράμματος Ροής (Καμμέας Α. - Τεχνικές Προγραμματισμού, τ.β', ΕΑΠ, Πάτρα, 2000)

Διάγραμμα Ροής Προγράμματος - ΔΡΠ

Εκτός από την λεκτική αναπαράσταση των αλγορίθμων, διαδεδομένη είναι και η συμβολική (γραφική) αναπαράσταση.

Η συμβολική (γραφική) αναπαράσταση βασίζεται κυρίως στα Διαγράμματα Ροής Προγράμματος - ΔΡΠ (Program Flowcharts)

Το ΔΡΠ για το παλαιότερο, περισσότερο διαδεδομένο, αλλά και περισσότερο αμφισβητημένο εργαλείο αναπαράστασης.

Τα ΔΡΠ θεωρούν έναν αλγόριθμο σαν ένα σύνολο από απλά ή σύνθετα τμήματα κώδικα.

Επειδή για κάθε τέτοιο τμήμα υπάρχει και ένα ειδικό σύμβολο, τελικά ο αλγόριθμος αναπαρίσταται ως ένα ΔΡΠ.

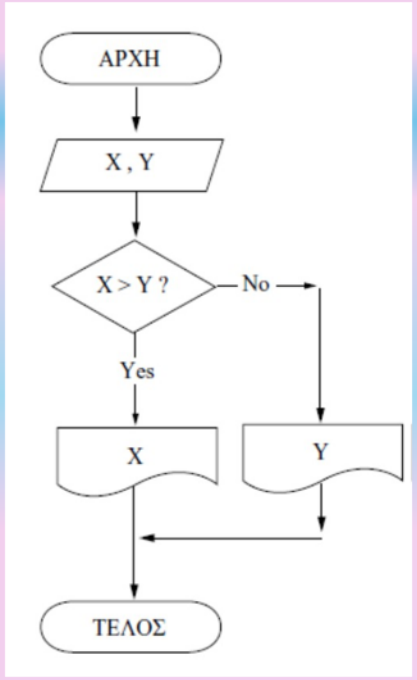
Κανόνες ΔΡΠ

Κατά την κατασκευή ενός ΔΡΠ ακολουθούνται κάποιοι βασικοί κανόνες:

- **Η ροή σε ένα σύστημα που αναπαρίσταται με ΔΡΠ έχει κατεύθυνση από πάνω προς τα κάτω και από αριστερά προς τα δεξιά.**

- **Κάθε σύμβολο του διαγράμματος έχει ένα και μοναδικό όνομα**
- **Δε ονομάζονται οι ροές, αφού δείχνουν απλά την ακολουθία εκτέλεσης .**
- **Πρέπει ένα διάγραμμα να περιέχει τόσα σύμβολα, ώστε να χωρά σε μια σελίδα.**
- **Εάν το ΔΡΠ είναι πολύπλοκο, χρησιμοποιούνται τα σύμβολα διασύνδεσης**

```
ΑΛΓΟΡΙΘΜΟΣ MAX-XY
ΔΕΔΟΜΕΝΑ
    X,Y,MAX: REAL;
ΑΡΧΗ
    ΔΙΑΒΑΣΕ(X,Y);
    ΕΑΝ (X > Y) ΤΟΤΕ
        MAX:=X
    ΑΛΛΙΩΣ
        MAX:=Y
    ΕΑΝ-ΤΕΛΟΣ;
    ΤΥΠΩΣΕ(MAX)
ΤΕΛΟΣ
```



Συγκρίνει δύο αριθμούς και τυπώνει τον μεγαλύτερο

(Καμμέας Α. - Τεχνικές Προγραμματισμού, τ.β', ΕΑΠ, Πάτρα, 2000)

ΔΟΜΗΜΕΝΟΣ & ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Ο δομημένος προγραμματισμός βασίζεται στην οργάνωση του κώδικα σε δομές ελέγχου (επαναλήψεις, διακλαδώσεις)

ο αντικειμενοστραφής (OOP) επικεντρώνεται στην οργάνωση γύρω από "αντικείμενα" που περιέχουν δεδομένα και λειτουργίες. Ο αντικειμενοστραφής προγραμματισμός προσφέρει καλύτερη διαχείριση πολυπλοκότητας, επαναχρησιμοποίηση κώδικα και ευελιξία, καθιστώντας τον πιο κατάλληλο για μεγάλα και πολύπλοκα έργα.

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

Δομημένος προγραμματισμός

Βασική αρχή:
Χρήση δομών ελέγχου (όπως if-else, for, while) για την οργάνωση της ροής εκτέλεσης του κώδικα.

Στόχος:
Να κάνει τον κώδικα πιο κατανοητό και ευκολότερο στον εντοπισμό και διόρθωση σφαλμάτων.

Αντικειμενοστραφής προγραμματισμός

Βασική αρχή:

Οργάνωση του κώδικα σε "αντικείμενα" που ενσωματώνουν δεδομένα (ιδιότητες) και τη λογική (μεθόδους) για την επεξεργασία τους.

Οφέλη:

Επαναχρησιμοποίηση κώδικα: Μπορείτε να επαναχρησιμοποιήσετε υπάρχοντα αντικείμενα σε νέα προγράμματα.

Διαχείριση πολυπλοκότητας: Οργανώνει σύνθετα συστήματα σε μικρότερα, διαχειρίσιμα μέρη.

Ευελιξία και ευκολία συντήρησης: Οι τροποποιήσεις σε ένα μέρος του κώδικα επηρεάζουν λιγότερο τα άλλα μέρη.

Καθαρότερη δομή: Τα αντικείμενα αλληλεπιδρούν μεταξύ τους, δίνοντας μια σαφή δομή στο πρόγραμμα.

Συνοπτικά, ενώ ο δομημένος προγραμματισμός επικεντρώνεται στη ροή της εκτέλεσης, ο αντικειμενοστραφής προγραμματισμός επικεντρώνεται στην οργάνωση του κώδικα γύρω από δεδομένα και τη συμπεριφορά τους, προσφέροντας μεγαλύτερη δομή και ευελιξία σε μεγαλύτερα έργα.

Βασικές Αρχές για τη Δημιουργία Λογισμικού

Τα πολύπλοκα Πληροφοριακά Συστήματα (ΠΣ) δεν είναι δυνατόν να αναπτυχθούν από έναν μόνο άτομο

Ιστορικά η αντιμετώπιση σύνθετων προβλημάτων γίνεται με την διάσπαση τους σε μικρότερα

Η διαδικασία αυτή επαναλαμβάνεται έως ότου κάθε μικρό πρόβλημα είναι κατανοητό και διαχειρίσιμο

Στη συνέχεια γίνεται σύνθεση των επιμέρους λύσεων ώστε να προκύψει η λύση του κυρίως προβλήματος

Η διαδικασία της διάσπασης ενός προβλήματος σε απλούστερα αναφέρεται ως ανάλυση

Η διαδικασία της «συναρμολόγησης» μιας μεγάλης κατασκευής από τα μικρότερα δομικά της στοιχεία λέγεται σύνθεση

Η αναλυτική και συνθετική ικανότητά είναι απαραίτητες σε όποιον ασχολείται με Ανάλυση και Σχεδιασμό ΠΣ (ΑΣΠΣ)

Η ανάγκη για χρήση μοντέλων

Τα τελευταία χρόνια μια σειρά από μοντέλα και τεχνικές μοντελοποίησης έχουν προταθεί όπως η Ενοποιημένη Γλώσσα Μοντελοποίησης – UML (**Unified Modeling Language -** γραφική γλώσσα για δημιουργία διαγραμμάτων, όχι γλώσσα προγραμματισμού)

Γιατί χρησιμοποιούμε μοντέλα;

Για να μην γράφουμε κείμενα.

Τα μοντέλα είναι πιο εύκολα στην κατανόηση

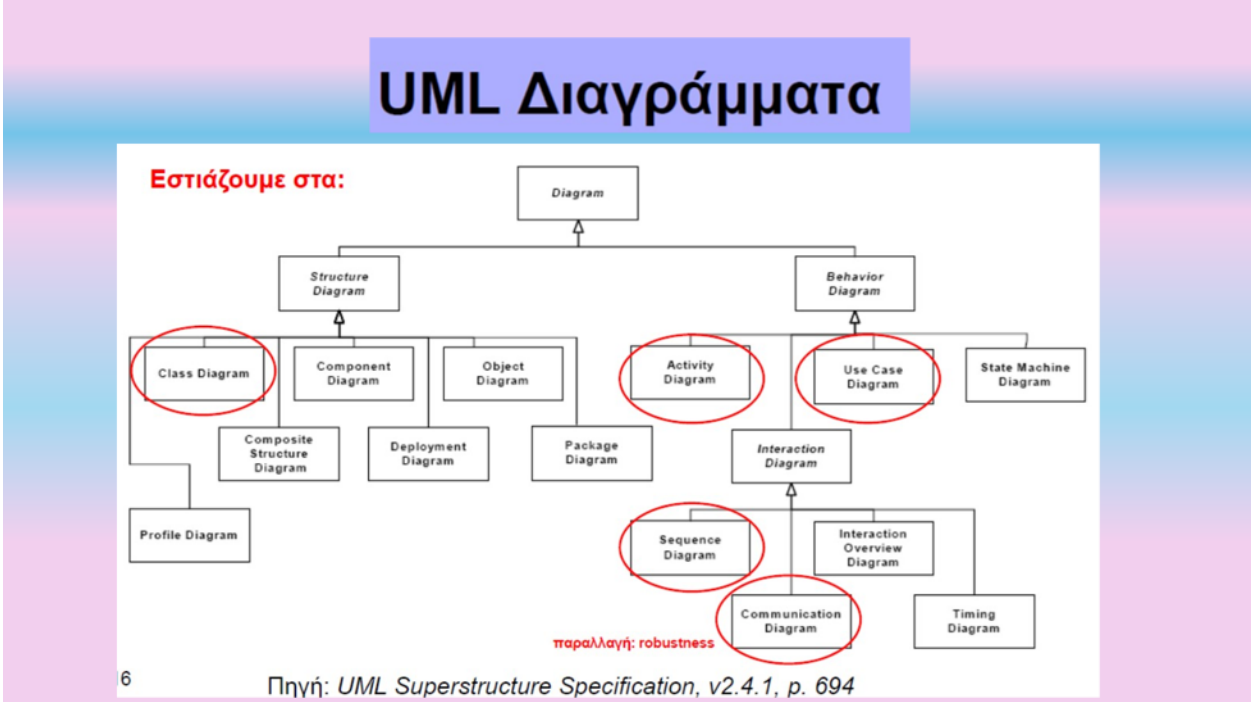
Με τη χρήση μοντέλων μειώνεται σημαντικά η ασάφεια που είναι χαρακτηριστικό του γραπτού λόγου

Τα μοντέλα βοηθούν στη καλύτερη επικοινωνία

Λογισμικό

Το λογισμικό **CASE (Computer Aided Software Engineering)** έχει αναπτυχθεί σχεδόν ταυτόχρονα με την Τεχνολογία Λογισμικού. Περιλαμβάνει εργαλεία και μεθόδους που βοηθούν στην ανάπτυξη λογισμικού.

Η γλώσσα μοντελοποίησης **UML (Unified Modeling Language)** υποστηρίζεται από πλήθος εργαλείων.



Βασικοί ορισμοί στον Αντικειμενοστρεφή Προγραμματισμό

Κλάση (Class)

Ομάδα στοιχείων – δεδομένων - αντικειμένων με την ίδια δομή και την ίδια συμπεριφορά.

Πεδίο – Γνώρισμα – Χαρακτηριστικό (Field – Attribute)

Μεταβλητή, που προσδίδει κάποιο χαρακτηριστικό – περιγραφική πληροφορία σε κάθε στοιχείο της κλάσης.

Αντικείμενο

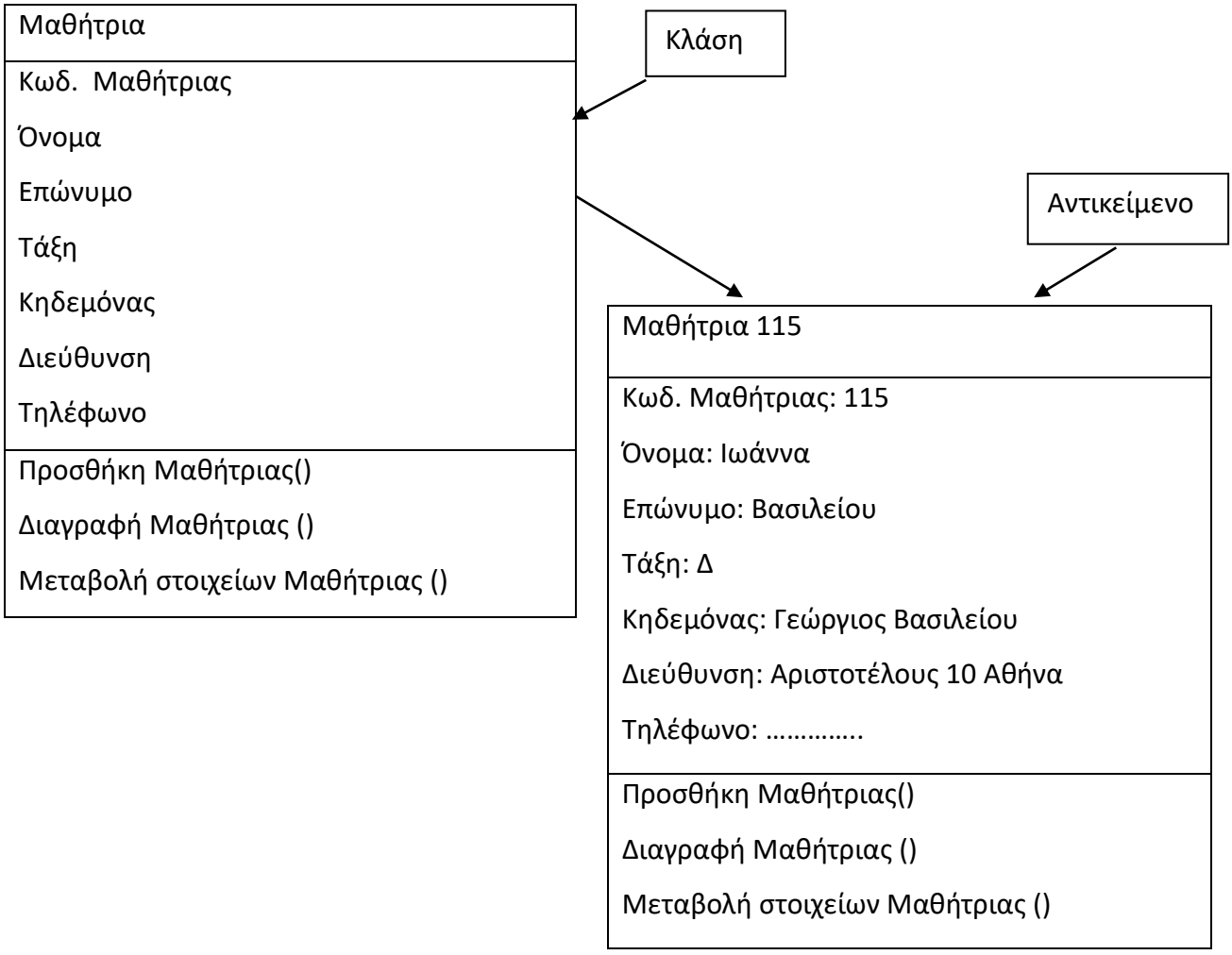
Κάθε στοιχείο μιας κλάσης αποτελεί αντικείμενο της κλάσης. Έχει τα ίδια χαρακτηριστικά (πεδία) της κλάσης αλλά οι τιμές διαφέρουν σε καθένα.

Στιγμιότυπο – Εκδοχή (Instance)

Κάθε αντικείμενο αποτελεί ένα μοναδικό στιγμιότυπο – εκδοχή της κλάσης.

Μέθοδος (Method)

Αποτελεί ένα ενεργό συστατικό του λογισμικού, είναι συνάρτηση ή διαδικασία, η οποία υλοποιεί ένα μέρος της συμπεριφοράς του αντικειμένου. Το σύνολο όλων των μεθόδων ενός αντικειμένου καθορίζει την συμπεριφορά του.



Παράδειγμα με **κλάση (Μαθήτρια)**, με **πεδία** (Κωδ. Μαθήτριας, Όνομα, Επώνυμο κ.λ.π.) και **μεθόδους** (Προσθήκη Μαθήτριας(), Διαγραφή Μαθήτριας (), Μεταβολή στοιχείων Μαθήτριας ()). Το **αντικείμενο (Μαθήτρια 115)** της κλάσης έχει συγκεκριμένες τιμές στα χαρακτηριστικά-πεδία.

..

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

Β. ΑΣΚΗΣΕΙΣ

1. Δώστε τον Ορισμό του **Αλγορίθμου** και των Εργαλείων Έκφρασης του Αλγορίθμου (**Ψευδοκώδικας & Διαγράμματα Ροής**)

2. Δώστε τον Ορισμό του **Δομημένου Προγραμματισμού**

3. Δώστε τον ορισμό του **Αντικειμενοστρεφούς Προγραμματισμού**

4. Με βάση τις ανωτέρω δομές και λέξεις δημιουργείστε Ψευδοκώδικα. Με βάση τα σύμβολα δημιουργείστε Διάγραμμα Ροής:

   α. Συγκρίνετε 2 αριθμούς Χ και Υ και δώστε να τυπώνεται ο μεγαλύτερος

   β. Κατόπιν δώστε τον κώδικα σε γλώσσα C++

5. α. Δημιουργείστε ψευδοκώδικα και διάγραμμα Ροής, ώστε να τυπώνονται, τα ακόλουθα:

   I am a student

   My name is ……

   I am learning C++

   β. Εν συνεχεία μετατρέψτε σε γλώσσα C++

6. Δημιουργείστε 1 κλάση **Γεωλογικός Σχηματισμός** και 1 κλάση **Τεκτονική Επαφή** με τα χαρακτηριστικά τους και τις ιδιότητές τους. Επίσης, δημιουργείστε από 1 **αντικείμενο** για την καθεμιά.

7. Γράψτε κώδικα σε C++ για τον υπολογισμό εμβαδού Ορθογωνίου Παραλληλογράμμου. Δώστε επεξήγηση-σχόλια στον κώδικα.

8. Δώστε από το πληκτρολόγιο, 2 ακέραιους αριθμούς. Στη συνέχεια υπολογίστε και δώστε να εμφανίζεται το άθροισμά τους. Δώστε επεξήγηση-σχόλια στον κώδικα.

9. Ελεγχόμενη ροή προγράμματος: Η ροή ενός προγράμματος, κατευθύνεται με πολλούς τρόπους. Κάθε τρόπος είναι για συγκεκριμένο σκοπό.

Παραδείγματα:

Cin_f.cpp

```
#include <iostream>
using namespace std;

int main() {
  int x ;

  cout << "type x = ";

          if (cin >> x) {

    cout << "x = " << x;
  }
  return 0;
}
```

Cin_f2.cpp

```
#include <iostream>
using namespace std;


int main() {
  int x ;
  int y ;
  cout << "type x: ";
  cin >> x;
  cout << "type y: ";
  cin >> y;

          if (cin) {

  cout << "x = " << x << "\n";
  cout << "y = " << y << "\n";
  }
  return 0;
}
```

If.cpp

```
#include <iostream>
using namespace std;

int main(){

int x = 10;
int y = 5;
if (x > y) {
cout << "x > y";
}
else if (x = y){
cout << "x = y";
}
else {
cout << "y > x";
}
return 0;
}
```

Cin_f3.cpp

```
#include <iostream>
using namespace std;

int main() {
  int x ;
  int y ;
  cout << "type x: ";
  cin >> x;
  cout << "type y: ";
  cin >> y;

          if (cin) {

  cout << "x = " << x << "\n";
  cout << "y = " << y << "\n";
  }

  if (x > y) {
          cout << "x > y" ;
```

```
  } else if (x == y){
        cout << "x = y" ;
  } else {
        cout << "x < y" ;
  }

 return 0;
}
```

c.cpp

```
#include <iostream>
using namespace std;

int main() {
int x;
cout << "type x = ";
cin >> x;
if (cin) {
        if (x == 10)
cout << "x is ten" << endl;
else if (x == 20) {
cout << "x is twenty" << endl;
} else if (x == 30) {
cout << "x is thirty" << endl;
} else if (x == 40) {
cout << "x is fourty" << endl;
} else {
cout << "x is not 10, 20, 30 or 40." << endl;
}

}
return 0;
}
```

10. Δώστε πλήρη κώδικα με σχόλια για εντολές και χρήση των ***if - else if - else***

A.

```
if (x = 10) {
cout << "x is ten" << endl;
} else if (x = 20) {
```

```
cout << "x is twenty" << endl;
} else if (x = 30) {
cout << "x is thirty" << endl;
} else if (x = 40) {
cout << "x is fourty" << endl;
} else {
cout << "x is not 10, 20, 30 or 40." << endl;
}
```

Β.

```
int x = 10;
int y = 5;
if (x > y) {
cout << "x > y";
}
else if (x = y) {
cout << "x = y";
}
else {
cout << "y > x";
```

11. Η εντολή ***switch*** χρησιμοποιείται για πολλαπλές επιλογές ή τιμές για μια μεταβλητή που θα θέλαμε να εκτελεστούν διαφορετικές εντολές για κάθε τιμή, όπως στο παραδειγμα:

……..

```
int day = 5;
switch (day) {
  case 1:
    cout << "Monday";
    break;
  case 2:
    cout << "Tuesday";
    break;
  case 3:
    cout << "Wednesday";
    break;
  case 4:
    cout << "Thursday";
```

```
    break;
  case 5:
    cout << "Friday";
    break;
  case 6:
    cout << "Saturday";
    break;
  case 7:
    cout << "Sunday";
    break;
}
// Outputs "Friday" (day 5)
```

***Οι εντολές break/continue**

Πολλές φορές θέλουμε να διακόψουμε την εκτέλεση των εντολών σε ένα loop πριν από την προκαθορισμένη στιγμή.

```
for (int i = 1; i =< 10; i += 3) {
if ( i*i > 100)
break;
cout << i << endl;
}
```

Το αποτέλεσμα θα είναι:

1
4
7
10

Η εντολή **break** χρησιμοποιείται σε όλα τα loops: **for, while, do/while.**

12. Η εντολή **for** χρησιμοποιείται για επανάληψη (loop) υπό συνθήκη κάποιων εντολών. Θεωρείται ίσως η πιο δυνατή και ευέλικτη εντολή τύπου for, σε οποιαδήποτε γλώσσα (έχει ακριβώς την ίδια λειτουργία με την αντίστοιχη της C). Η σύνταξή της είναι η εξής:

for (εντολή αρχικοποίησης; συνθήκη; εντολή επανάληψης) {
εντολές;
}

```
#include <iostream>
using namespace std;
int main() {
  for (int x = 0; x <= 50; x = x + 5) {
    cout << x << "\n";
  }
  return 0;

}
```

```
#include <iostream>
using namespace std;

int main() {
  // Outer loop
  for (int i = 1; i <= 2; ++i) {
    cout << "Outer: " << i << "\n";  // Executes 2 times

    // Inner loop
    for (int j = 1; j <= 3; ++j) {
      cout << " Inner: " << j << "\n";  // Executes 6 times (2 * 3)
    }
  }
  return 0;
}
```

Outer: 1
Inner: 1
Inner: 2
Inner: 3
Outer: 2
Inner: 1
Inner: 2
Inner: 3

13. Η εντολή **while** χρησιμοποιείται αντί της **for** όταν **δεν μπορούμε να προβλέψουμε εύκολα** πόσες φορές θέλουμε να εκτελεστούν οι εντολές, ή όταν δεν έχει σημασία ο αριθμός των επαναλήψεων αλλά η ικανοποιήση ή όχι της συνθήκης:

```
while (συνθήκη) {
εντολές;
}
```

Παράδειγμα:

```
bool exit_from_loop = false;
while (exit_from_loop == false) {

// υποθετική ρουτίνα που διαβάζει δεδομένα από ένα αρχείο
read_bytes(file1);

// δεύτερη ρουτίνα που γράφει δεδομένα σε ένα άλλο αρχείο
write_bytes(file2);
if (file_empty(file1) == true)
exit_from_loop = true;
```

14. Η εντολή **do/while** είναι παρόμοια με την **while,** με μία διαφορά, οι εντολές εντός του loop θα εκτελεστούν τουλάχιστον μια φορά ανεξαρτήτως αν είναι αληθής η συνθήκη ή όχι.

```
do {
εντολές;
} while (συνθήκη);
```

Παράδειγμα:

```
#include <iostream>
using namespace std;
int main() {
  int i = 0;
  do {
    cout << i << "\n";
    i++;
  }
  while (i < 5);
 return 0;
}
```

15. **C++ Arrays (Πίνακες – Σειρές)**

Τα **Arrays** χρησιμοποιούνται για να αποθηκεύουν πολλές μεταβλητές σε μια τιμή, αντί να δηλώνονται διαφορετικές μεταβλητές για κάθε τιμή. Παραδείγματα δήλωσης για **string** και **integer**:

```
string cars[5];
string cars[5] = {"Fiat", "Mini", "Ford","Toyota", “Autobianchi”};
```

int mynum[5] = {10, 20, 30, 40, 50};

16. **Πρόσβαση ατα στοιχεία (Elements) ενός πίνακα (Array)**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string cars[4] = {"Fiat", "Mini", "Ford", "Toyota"};
  cout << cars[0];
  return 0;
}
```

Fiat

17. **Αντικατάσταση στοιχείου (Element) σε πίνακα (Array)**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string cars[4] = {"Fiat", "Mini", "Ford", "Toyota"};
  cars[0] = "Mercedes";
  cout << cars[0];
  return 0;
}
```

Mercedes

18. **Βρόχος σε Πίνακα (for)**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string cars[5] = {"Fiat", "Mini", "Ford", "Toyota", “Autobianchi”};
  for (int i = 0; i < 5; i++) {
    cout << cars[i] << "\n";
  }
  return 0;
}
```

Fiat
Mini
Ford
Toyota

Autobianchi

19. Εξαγωγή του δείκτη και της τιμής

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string cars[5] = {"Fiat", "Mini", "Ford", "Toyota", “Autobianchi”;
  for (int i = 0; i < 5; i++) {
    cout << i << " = " << cars[i] << "\n";
  }
  return 0;
}
```

0 = Fiat
1 = Mini
2 = Ford
3 = Toyota
4 = Autobianchi

20. Βρόχος σε Πίνακα ακεραίων αριθμών

```
#include <iostream>
using namespace std;
int main() {
  int myNumbers[5] = {100, 200, 300, 400, 500};
for (int i = 0; i < 5; i++) {
   cout << myNumbers[i] << "\n";
  }
return 0;
}
```

100
200
300
400
500

21. Μέγεθος Πίνακα (sizeof())

```
#include <iostream>
```

```
        using namespace std;
    int main() {
          int myNumbers[5] = {100, 200, 300, 400, 500};
          cout << sizeof(myNumbers);
           return 0;
    }
```

20

Η συνάρτηση **sizeof()** επιστρέφει το μέγεθος σε **bytes**.

Ένας ακέραιος (**int) έχει μέγεθος** 4 bytes, οπότε *4 x 5 (4 bytes x 5 elements)* = ***20 bytes***.

22. **Πόσα στοιχεία (elements) έχει ένας πίνακας (Διαιρούμε το μέγεθος του πίνακα με το μέγεθος του πρώτου στοιχείου του πίνακα**

```
#include <iostream>
using namespace std;
int main() {
  int myNumbers[5] = {100, 200, 300, 400, 500};
  int getArrayLength = sizeof(myNumbers) / sizeof(myNumbers[0]);
  cout << getArrayLength;
  return 0;
}
```

5

23. Βρόχος σε πίνακα με sizeof()

```
#include <iostream>
using namespace std;
int main() {
  int myNumbers[5] = {100, 200, 300, 400, 500};
  for (int i = 0; i < sizeof(myNumbers) / sizeof(myNumbers[0]); i++) {
    cout << myNumbers[i] << "\n";
  }
  return 0;
}
```

100
200
300

400
500

24. Πολλών Διαστάσεων (Multidimentional) Πίνακες

```
#include <iostream>
using namespace std;
int main() {
  string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
  };
    cout << letters[0][2];
  return 0;
}
```
C

25. Αλλαγή στοιχείου σε Πίνακα

```
#include <iostream>
using namespace std;
int main() {
  string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
  };
  letters[0][0] = "Z";
    cout << letters[0][0];
  return 0;
}
```
Z

26. Βρόχος σε πολυδιάστατο πίνακα

Χρειάζεται ένα βρόχος για κάθε διάσταση.

```
#include <iostream>
using namespace std;
int main() {
```

```
  string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
  };
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 4; j++) {
      cout << letters[i][j] << "\n";
    }
  }
  return 0;
}
```

Για κάθε γραμμή τυπώνονται όλες οι στήλες

Τυπώνονται 1η γραμμή (0) όλα τα στοιχεία (όλες οι στήλες 0,1,2,3,)

2η γραμμή (1) όλα τα στοιχεία (όλες οι στήλες)

A
B
C
D
E
F
G
H

27. C++ Structure (struct)

Δομή ή Structure (ή struct) είναι ο τρόπος ώστε να ομαδοποιηθούν παρόμοιες μεταβλητές.

Κάθε **μεταβλητή στη δομή** αποτελεί **μέλος -** member σε αυτήν.

Αντιθέτως με τον πίνακα - array, η **δομή - structure** δύναται να **περιλαμβάνει** διαφορετικού τύπου δεδομένα: **int, string, bool, etc.**

<u>Δημιουργία Structure</u>

```
struct {            // Structure declaration
  int myNum;        // Member (int variable)
  string myString;  // Member (string variable)
} myStructure;      // Structure variable
```

Πρόσβαση στα μέλη (Members) της Structure

Χρησιμοποιείται το σύμβολο της τελείας (.):

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  struct {
    int myNum;
    string myString;
  } myStructure;
  myStructure.myNum = 100;
  myStructure.myString = "My name is Helen!";
  cout << myStructure.myNum << "\n";
  cout << myStructure.myString << "\n";
  return 0;
}
100
My name is Helen!
```

28. One Structure in Multiple Variables

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  struct {
    string brand;
    string model;
    int year;
  } myCar1, myCar2; // We can add variables by separating them with a comma
here
  // Put data into the first structure
  myCar1.brand = "BMW";
  myCar1.model = "X5";
  myCar1.year = 1999;

  // Put data into the second structure
  myCar2.brand = "Ford";
  myCar2.model = "Mustang";
  myCar2.year = 1969;
  // Print the structure members
  cout << myCar1.brand << " " << myCar1.model << " " << myCar1.year << "\n";
```

```
  cout << myCar2.brand << " " << myCar2.model << " " << myCar2.year << "\n";
  return 0;
}
BMW X5 1999
Ford Mustang 1969
```

29. C++ Απαριθμήσεις (Enumerations / Enums)

Μια **Απαρίθμηση (Enum)** είναι ένας ιδιαίτερος τύπος που αναπαριστά μια ομάδα από σταθερές (constants / unchangeable values).
Μια **enum** δημιουργείται με τη χρήση του **enum keyword**, ακολουθούμενο από το όνομα που θα έχει (π.χ. στο παράδειγμα **Level)** . Τα πεδία **(LOW, MEDIUM, HIGH)** χωρίζονται με κόμμα. Μετά το τελευταίο πεδίο δεν χρειάζεται κόμμα.
Η προσπέλαση σε **enum**, γίνεται μέσω μεταβλητών (στο παράδειγμα **(myVar):**

Δημιουργία enum:

```
#include <iostream>
using namespace std;
 enum Level {
  LOW,
  MEDIUM,
  HIGH
};
int main() {
  enum Level myVar = MEDIUM;
  cout << myVar;
  return 0;

}
```

```
#include <iostream>
using namespace std;
 enum Level {
  LOW,
  MEDIUM,
  HIGH
};
int main() {
  enum Level myVar = MEDIUM;
  cout << myVar;
  return 0;
}
1
```

Είναι ορισμένο (by default), το πρώτο πεδίο (LOW) να παίρνει την τιμή 0, το δεύτερο (MEDIUM) την τιμή 1, κ.λ.π.

30. C++ Αναφορές (References)

Μια **αναφορά** σε μεταβλητή (reference variable) είναι ένα ψευδώνυμο (**alias**) για μια υπάρχουσα μεταβλητή. Δημιουργείται με το **(&) operator**:

```
string food = "Pizza";  // food variable
string &meal = food;    // reference to food
```

Χρησιμοποιώντας το **food** ή **meal** αναφερόμαστε στην ίδια τιμή:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string food = "Pizza";
  string &meal = food;
  cout << food << "\n";
  cout << meal << "\n";
  return 0;
}
Pizza
Pizza
```

31. Ενημέρωση μέσω References

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string food = "Pizza";      // food variable
  string &meal = food;        // reference to food

  meal = "Burger";            // change the reference
  cout << food << "\n";       // Outputs Burger
  cout << meal << "\n";       // Outputs Burger
  return 0;
}
```

Burger
Burger

32. Διεύθυνση Μνήμης (Memory Address)

Όταν δημιουργείται μια μεταβλητή, αυτή καταλαμβάνει συγκεκριμένη θέση στην μνήμη του υπολογιστή και όταν της προσδίδεται τιμή, καταλαμβάνει συγκεκριμένο χώρο. Με το σύμβολο (&) μας δίδεται η διεύθυνση μνήμης μιας μεταβλητής.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string food = "Pizza";
  cout << &food;
  return 0;
}
```
0x6dfed4

- **Η μνήμη δίδεται στο δεκαεξαδικό σύστημα αρίθμησης.**
- **Η μνήμη και η διεύθυνση μνήμης έχουν μεγάλη σπουδαιότητα στην C++, διότι δίνουν τη δυνατότητα διαχείρισης των δεδομένων στην μνήμη του υπολογιστή.**

33. C++ Δείκτες (Pointers)

Ο **Δείκτης** είναι μια **μεταβλητή,** η οποία **αποθηκεύει την διεύθυνση μνήμης.** Δημιουργείται με το σύμβολο **(*).** Προκειμένου να δηλωθεί ο δείκτης υπάρχουν τρεις τρόποι:

string* mystring; // Προτιμάται συνήθως
string *mystring;
string * mystring;

Παράδειγμα

```
#include <iostream>
#include <string>
```

```
using namespace std;
int main() {
  string food = "Pizza";  // A string variable
  string* ptr = &food;  // A pointer variable that stores the address of food
  // Output the value of food
  cout << food << "\n";
  // Output the memory address of food
  cout << &food << "\n";
  // Output the memory address of food with the pointer
  cout << ptr << "\n";
  return 0;
}
```

Pizza
0x6dfed4
0x6dfed4

34. C++ Dereference

Χρησιμοποιείται ο δείκτης ώστε να εξαχθεί η τιμή της μεταβλητής:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string food = "Pizza";  // Variable declaration
  string* ptr = &food;    // Pointer declaration
  // Reference: Output the memory address of food with the pointer
  cout << ptr << "\n";
  // Dereference: Output the value of food with the pointer
  cout << *ptr << "\n";
  return 0;
}
```

0x6dfed4
Pizza

35. C++ Μεταβάλλοντας την τιμή των δεικτών (Modify Pointers)

Θα αλλάξει και η τιμή της αρχικής μεταβλητής:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
```

```
  string food = "Pizza";
  string* ptr = &food;
  // Output the value of food
  cout << food << "\n";
  // Output the memory address of food
  cout << &food << "\n";
  // Access the memory address of food and output its value
  cout << *ptr << "\n";
    // Change the value of the pointer
  *ptr = "Hamburger";
    // Output the new value of the pointer
  cout << *ptr << "\n";
    // Output the new value of the food variable
  cout << food << "\n";
  return 0;
}
```

Pizza
0x6dfed4
Pizza
Hamburger
Hamburger

36. Μέγεθος Μνήμης (sizeof())

```
#include <iostream>
using namespace std;
int main() {
  int myInt;
  float myFloat;
  double myDouble;
  char myChar;
  cout << sizeof(myInt) << "\n";     // 4 bytes (typically)
  cout << sizeof(myFloat) << "\n";   // 4 bytes
  cout << sizeof(myDouble) << "\n";  // 8 bytes
  cout << sizeof(myChar) << "\n";    // 1 byte
  return 0;
}
```

4
4
8
1

37. C++ new and delete

Με **new** διαχειριζόμαστε την μνήμη:

```
#include <iostream>
using namespace std;
int main() {
  int* ptr = new int;
  *ptr = 35;
  cout << *ptr;
  return 0;
}
```
35

Επεξήγηση:

- **new int** δημιουργεί μνήμη για έναν **integer**
- **ptr** αποθηκεύει την διεύθυνση μνήμης
- ***ptr = 35**; Αποθηκεύει τον αριθμό 35
- **cout << *ptr**; Τυπώνει την τιμή

The delete Keyword

Μετά την εντολή **new** και όταν ολοκληρωθεί η εργασία μας πρέπει να δώσουμε **delete, ώστε να μην καταναλώνεται μνήμη, να καθαρίσει η μνήμη και να τερματίσει το πρόγραμμα :**

```
#include <iostream>
using namespace std;
int main() {
  int* ptr = new int; // Create memory
  *ptr = 35;  // Store a value
  cout << *ptr; // Print the value
  delete ptr; // Free the memory
  return 0;
}
```
35

38. C++ Συναρτήσεις (Functions)

Μια συνάρτηση αποτελεί ένα τμήμα κώδικα (block of code) που τρέχει όταν κλειθεί.

```
void myFunction() {
  // code to be executed
}
```

- **myFunction()** : Όνομα συνάρτησης
- **void :** εννοείται ότι η συνάρτηση δεν θα επιστρέψει τιμή.
- Εσωτερικά της συνάρτησης (the body), προσθέτουμε των κώδικα που ορίζει τι θα κάνει η συνάρτηση.

Κλήση Συνάρτησης (Call a Function)

```
#include <iostream>
using namespace std;
void myFunction() {
  cout << "I am learning C++!";
}

int main() {
  myFunction();
  return 0;
}
I am learning C++!
```

Επιστροφή τιμής (Return Values)

Με **(void)** η συνάρτηση δεν επιστρέφει τιμή. Προκειμένου να επιστρέψει τιμή πρέπει να δηλωθεί ένας τύπος δεδομένων (**int, string, etc**.) και η εντολή (**return)** μέσα στην συνάρτηση:

```
#include <iostream>
using namespace std;
int myFunction(int x) {
  return 5 + x;
}
int main() {
  cout << myFunction(3);
  return 0;
}
8
```

```
#include <iostream>
using namespace std;
int myFunction(int x, int y) {
  return x + y;
}
int main() {
  cout << myFunction(5, 3);
  return 0;
}
8
```

```
#include <iostream>
using namespace std;
int myFunction(int x, int y) {
  return x + y;
}
int main() {
  int z = myFunction(5, 3);
  cout << z;
  return 0;
}
8
```

```
#include <iostream>
using namespace std;

int doubleGame(int x) {
  return x * 2;
}
int main() {
  for (int i = 1; i <= 5; i++) {
    cout << "Double of " << i << " is " << doubleGame(i) << endl;
  }
  return 0;
}
Double of 1 is 2
Double of 2 is 4
Double of 3 is 6
Double of 4 is 8
```

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

39. Κλάσεις και Αντικείμενα (Class & Objects)

Class	Objects
Geological Formation	Limestones, Flysch, Alluvial
Tectonic Lines	Fault, Thrust, Overthrust

Δημιουργία Κλάσης

```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
```

- Με class keyword δημιουργείται η κλάση MyClass.
- Το public keyword είναι ένα **access specifier**, που ορίζει ότι τα μέλη της κλάσης (attributes and methods) είναι δημόσια, δηλαδή προσπελάσιμα και εκτός κλάσης.
- Μέσα στην κλάση υπάρχουν οι μεταβλητές myNum myString. Οι οποίες λέγονται **attributes**.
- Η κλάση κλείνει με το ελληνικό ερωτηματικό(;)

Δημιουργία Αντικειμένου στην κλάση

Το αντικείμενο δημιουργείται δίνοντας το όνομα της κλάσης, ακολουθούμενο με το όνομα του αντικειμένου.
Η προσπέλαση στα πεδία της κλάσης γίνεται δίνοντας το όνομα του αντικειμένου, ακολουθούμενο από τελεία και το όνομα του πεδίου.

```
#include <iostream>
#include <string>
using namespace std;
```

```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
int main() {
  MyClass myObj;  // Create an object of MyClass
  // Access attributes and set values
  myObj.myNum = 225;
  myObj.myString = "This is a text";
  // Print values
  cout << myObj.myNum << "\n";
  cout << myObj.myString;
  return 0;
}
```

225
This is a text

Πολλά Αντικείμενα

```
#include <iostream>
#include <string>

using namespace std;
class Formation {
  public:
    string name;
    string year;
};

int main() {
  Formation formationObj1;
  formationObj1.name = "limestones";
  formationObj1.year = " Cretaceous";

  Formation formationObj2;
  formationObj2.name = "Conglomerates";
  formationObj2.year = "Pleio-Pleiocene";
  cout << formationObj1.name << " " << formationObj1.year << " " << "\n";
  cout << formationObj2.name << " " << formationObj2.year << " " << "\n";
  return 0;
}
```

Limestones Cretaceous

Conglomerates Pleio-Pleiocene

Μέθοδοι Κλάσης

Οι μέθοδοι (**methods**) είναι συναρτήσεις (**functions)** που ανήκουν στην κλάση (**class**). Με δύο τρόπους ορίζονται συναρτήσεις που ανήκουν σε κλάση:

- Δήλωση μέσα στην κλάση
- Δήλωση έξω από την κλάση

Δήλωση Μεθόδου μέσα στην Κλάση

```
#include <iostream>
using namespace std;
class MyClass {         // The class
  public:               // Access specifier
    void myMethod() {   // Method/function
      cout << "Hello World!";
    }
};
int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```

Δήλωση Μεθόδου έξω από την Κλάση

Μερικές φορές είναι καλύτερο να δηλώνεται η μέθοδος στην κλάση και να καθορίζεται αργότερα (ειδικά σε μεγάλα προγράμματα).

Αυτό επιτυγχάνεται, ορίζοντας το όνομα της κλάσης ακολουθούμενο με **(: :)** και το όνομα της συνάρτησης.

```
#include <iostream>
using namespace std;
class MyClass {         // The class
  public:               // Access specifier
    void myMethod();    // Method/function declaration
};
// Method/function definition outside the class
```

```
void MyClass::myMethod() {
  cout << "Hello World!";
}
int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```

Παράμετροι (Parameters)

Μπορούμε να δώσουμε τιμές στις μεθόδους όπως στις κανονικές συναρτήσεις:

```
#include <iostream>
using namespace std;
class Car {
  public:
    int speed(int maxSpeed);
};
int Car::speed(int maxSpeed) {
  return maxSpeed;
}
int main() {
  Car myObj;
  cout << myObj.speed(200);
  return 0;
}
```

200

Στην C++ έχουμε:

- **Public Methods (Δημόσιες Μεθόδους)** – Τα μέλη της κλάσης (συναρτήσεις) θα είναι προσβάσιμα μέσα και έξω από την κλάση.
- **Private Methods (Ιδιωτικές Μεθόδους)** – Τα μέλη της κλάσης δεν είναι προσβάσιμα (ή ορατά) από συναρτήσεις ή κώδικα, έξω από την κλάση.
- **Protected Methods (Προστατευμένες Μεθόδους)** – Τα μέλη είναι προσβάσιμα μόνο από την κλάση στην οποία δηλώνονται και από οποιαδήποτε άλλη κλάση – παιδί της κλάσης (κλάση που κληρονομεί).

Ενθυλάκωση (Encapsulation)

Η έννοια της Ενθυλάκωσης αναφέρεται στο ότι τα "sensitive" δεδομένα είναι κρυμμένα από τους χρήστες. Δηλαδή, αναφέρεται στην απόκρυψη πληροφοριών - προστασία της εσωτερικής κατάστασης ενός αντικειμένου από μη εξουσιοδοτημένη εξωτερική

πρόσβαση ή τροποποίηση. Στην Ενθυλάκωση, τα δεδομένα είναι κρυμμένα και μόνον «trusted methods» μπορούν να έχουν πρόσβαση ή να τροποποιήσουν αυτά.

- Δηλώνονται οι μεταβλητές / χαρακτηριστικά (class variables/attributes) ως private.
- Πρόσβαση (διάβασμα ή τροποποίηση) στην τιμή ενός private member επιτυγχάνεται μέσω των public «**get»** και «**set»** methods.

C++ Friend Functions

Οι **friend functions (φίλες συναρτήσεις)** στην C++ είναι εξωτερικές συναρτήσεις που, παρόλο που δεν είναι μέλη μιας κλάσης, έχουν το ειδικό προνόμιο να έχουν πρόσβαση στα **ιδιωτικά (private)** και **προστατευμένα (protected)** μέλη αυτής της κλάσης.

Βασικά Χαρακτηριστικά

- **Δεν είναι μέλη κλάσης**: Οι φίλες συναρτήσεις ορίζονται εκτός του πεδίου εφαρμογής της κλάσης και δεν καλούνται χρησιμοποιώντας τους τελεστές επιλογής μέλους (. ή ->).
- **Πρόσβαση**: Η κλάση "παραχωρεί" την πρόσβαση σε μια συγκεκριμένη συνάρτηση δηλώνοντάς την ως friend μέσα στο σώμα της με τη δεσμευμένη λέξη friend.
- **Ενθυλάκωση**: Η χρήση τους επιτρέπει την πρόσβαση σε ιδιωτικά δεδομένα, κάτι που υπό κανονικές συνθήκες παραβιάζει την αρχή της ενθυλάκωσης (data hiding). Για αυτόν τον λόγο, θα πρέπει να χρησιμοποιούνται με σύνεση.
- **Ευελιξία**: Χρησιμοποιούνται συχνά για λειτουργίες που απαιτούν πρόσβαση σε ιδιωτικά δεδομένα, αλλά δεν ταιριάζουν λογικά ως μέθοδοι της κλάσης, όπως για παράδειγμα η υπερφόρτωση του τελεστή εισόδου/εξόδου (<< και >>).
- **Δεν κληρονομούνται**: Η ιδιότητα του "φίλου" δεν κληρονομείται στις παράγωγες κλάσες.

Σύνταξη και Παράδειγμα

Μια **φίλη συνάρτηση** δηλώνεται μέσα σε μια κλάση χρησιμοποιώντας τη λέξη-κλειδί **friend.** Η δήλωση μπορεί να τοποθετηθεί σε οποιοδήποτε τμήμα της κλάσης

(public, private ή protected), καθώς η θέση της δεν επηρεάζει τα δικαιώματα πρόσβασης.

```
#include <iostream>
using namespace std;
class Employee {
  private:
    int salary;
  public:
    Employee(int s) {
      salary = s;
    }
    // Declare friend function
    friend void displaySalary(Employee emp);
};
void displaySalary(Employee emp) {
  cout << "Salary: " << emp.salary;
}
int main() {
  Employee myEmp(50000);
  displaySalary(myEmp);
  return 0;
}
```

Salary: 50000

Κληρονομικότητα (Inheritance)

Μια κλάση (παιδί) κληρονομεί χαρακτηριστικά από μια άλλη κλάση (γονέας).
Για την κλάση-παιδί χρησιμοποιείται το σύμβολο (:) .

Στο παράδειγμα η κλάση **Car** (child) κληρονομεί χαρακτηριστικά και μεθόδους από την κλάση **Vehicle** (parent):

```
#include <iostream>
#include <string>
using namespace std;
// Base class
class Vehicle {
  public:
    string brand = "Ford";
    void honk() {
      cout << "Tuut, tuut! \n" ;
    }
};
```

```
// Derived class
class Car: public Vehicle {
  public:
    string model = "Mustang";
};
int main() {
  Car myCar;
  myCar.honk();
  cout << myCar.brand + " " + myCar.model;
  return 0;
}
```

Tuut, tuut!
Ford Mustang

Πολυμορφισμός (Polymorphism)

Πολυμορφισμός σημαίνει «πολλές μορφές» και απαντάται όταν έχουμε πολλές κλάσεις που σχετίζονται μεταξύ τους με την κληρονομικότητα. Με τον πολυμορφισμό έχουμε την δυνατότητα να χρησιμοποιήσουμε τις μεθόδους και να πάρουμε διαφορετικά αποτελέσματα.

Στο παράδειγμα με την **κλάση-γονέας Animal**, η μέθοδος **makeSound()** δίνει διαφορετικές τιμές-αποτελέσματα για τις **κλάσεις-παιδιά: Pigs, Cats, Dogs, Birds,** κ.λπ. Για κάθε ζώο χρησιμοποιείται η μέθοδος **"make a sound",** αλλά οι ήχοι σε κάθε ζώο είναι διαφορετικοί:

- **Pig:** wee wee
- **Dog:** bow wow
- **Bird:** tweet tweet

```
#include <iostream>
#include <string>
using namespace std;
// Base class
class Animal {
  public:
    void animalSound() {
      cout << "The animal makes a sound \n";
    }
};
```

```
// Derived class
class Pig : public Animal {
  public:
    void animalSound() {
      cout << "The pig says: wee wee \n";
    }
};
// Derived class
class Dog : public Animal {
  public:
    void animalSound() {
      cout << "The dog says: bow wow \n";
    }
};
int main() {
  Animal myAnimal;
  Pig myPig;
  Dog myDog;
  myAnimal.animalSound();
  myPig.animalSound();
  myDog.animalSound();
  return 0;
}
```

The animal makes a sound
The pig says: wee wee
The dog says: bow wow

C++ Πρότυπα (Templates)

Τα Πρότυπα δίνουν την δυνατότητα να γράψουμε συνάρτηση ή κλάση, η οποία χρησιμοποιείται με διαφορετικό τύπο δεδομένων. Ο κώδικας επαναχρησιμοποιείται και τα προγράμματα είναι ευέλικτα.

C++ Πρότυπα Συναρτήσεων (Function Templates)

Μια **function template** δημιουργείται με το **template** keyword:

```
template <typename T>
return_type function_name(T parameter) {
  // code
}
```

- T is a placeholder for a data type (like int, float, etc.).
- You can use any name instead of T, but T is common.

Παράδειγμα:

```
#include <iostream>
using namespace std;
template <typename T>
T add(T a, T b) {
  return a + b;
}
int main() {
  cout << add<int>(5, 3) << "\n";
  cout << add<double>(2.5, 1.5) << "\n";
  return 0;
}
```

8
4

C++ Πρότυπα Κλάσεων (Class Templates)

```
template <typename T>
class ClassName {
  // members and methods using T
};
```

40. C++ Αρχεία (Files)

Η βιβλιοθήκη **fstream** χρησιμοποιείται για αρχεία - files.
Η βιβλιοθήκη **fstream** περιλαμβάνει τις βιβλιοθήκες standard **<iostream>** και **<fstream>** για header file:

```
#include <iostream>
#include <fstream>
```

Υπάρχουν τρεις Κλάσεις (classes) στην βιβλιοθήκη fstream, που χρησιμοποιούνται ώστε να δημιουργήσουν, γράψουν ή να διαβάσουν αρχεία:

Class	Description
`ofstream`	Creates and writes to files
`ifstream`	Reads from files
`fstream`	A combination of ofstream and ifstream: creates, reads, and writes to files

Δημιουργία και εγγραφή σε Αρχείο (File)

Για εγγραφή σε αρχείο χρησιμοποιούμε το τελεστή **(<<).**

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
  // Create and open a text file
  ofstream MyFile("filename.txt");
  // Write to the file
  MyFile << "Files can be tricky, but it is fun enough!";
  // Close the file
  MyFile.close();
}
```

Είναι απαραίτητο να κλείνουμε τα αρχεία ώστε καθαρίζεται και απελευθερώνεται και ο χώρος της μνήμης.

Διάβασμα από Αρχείο (File)

Για να διαβάσουμε από ένα αρχείο χρησιμοποιούμε τις **ifstream** or **fstream class**, και το όνομα του αρχείου.

Όταν έχουμε ένα **while loop** χρησιμοποιούμε και την συνάρτηση **getline()** που ανήκει στην **ifstream class**, ώστε να διαβαστεί το αρχείο γραμμή – γραμμή και να τυπώσει το περιεχόμενό του :

```
// Create a text string, which is used to output the text file
string myText;

// Read from the text file

ifstream MyReadFile("filename.txt");

// Use a while loop together with the getline() function to read the file line by line

while (getline (MyReadFile, myText)) {
  // Output the text from the file
  cout << myText;
}

// Close the file

MyReadFile.close();
```

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main () {
  // Create a text file
  ofstream MyWriteFile("filename.txt");
  // Write to the file
  MyWriteFile << "Files can be tricky, but it is fun enough!";
   // Close the file
  MyWriteFile.close();
  // Create a text string, which is used to output the text file
  string myText;
  // Read from the text file
  ifstream MyReadFile("filename.txt");
  // Use a while loop together with the getline() function to read the file line by line
  while (getline (MyReadFile, myText)) {
```

```
    // Output the text from the file
    cout << myText;
  }
  // Close the file
  MyReadFile.close();
}https://www.w3schools.com/
```

Files can be tricky, but it is fun enough!

41. Date and Time

Η βιβλιοθήκη **<ctime>** έχει πλήθος συναρτήσεων και δίνει την δυνατότητα να υπολογίσουμε ημερομηνίες και χρόνο-ώρα.

Παράδειγμα:

#include <ctime> // Import the ctime library

Display Current Date and Time

<ctime>: για υπολογισμό ημερομηνίας και χρόνου-ώρας.

time() function: δίνει ένα **timestamp** που αντιπροσωπεύει την τρέχουσα ημερομηνία και ώρα. representing the current date and time.

```
#include <iostream>
#include <ctime>  // Import the ctime library
using namespace std;
 int main () {
  // Get the timestamp for the current date and time
  time_t timestamp;
  time(&timestamp);
  // Display the date and time represented by the timestamp
  cout << ctime(&timestamp);
  return 0;
}
```

Mon Nov 24 09:44:18 2025

Data Types

Δύο τρόποι χρησιμοποιούνται ώστε να αποθηκευθεί η ημερομηνία και χρόνος: **time_t** για **timestamps** και **struct tm** για **datetime structures**.

Τα **Timestamps** αντιπροσωπεύουν μια στιγμή του χρόνου ως ένα απλό αριθμό, ώστε ο υπολογιστής να κάνει εύκολα τους υπολογισμούς.
Οι **Datetime structures** είναι δομές που αντιπροσωπεύουν διαφορετικά στοιχεία της ημερομηνίας και χρόνου ως μέλη. Είναι ευκολότερο στον χρήστη να προσδιορίσει τις ημερομηνίες και χρόνο:

- tm_sec - The seconds within a minute
- tm_min - The minutes within an hour
- tm_hour - The hour within a day (from 0 to 23)
- tm_mday - The day of the month
- tm_mon - The month (from 0 to 11 starting with January)
- tm_year - The number of years since 1900
- tm_wday - The weekday (from 0 to 6 starting with Sunday)
- tm_yday - The day of the year (from 0 to 365 with 0 being January 1)
- tm_isdst - Positive when daylight saving time is in effect, zero when not in effect and negative when unknown

Τα συστατικά αντιπροσωπεύουν:

- Hours are represented in 24-hour format. 11pm would be represented as **23**.
- The months go from 0 to 11. For example, December would be represented as **11** rather than 12.
- Years are represented relative to the year 1900. The year 2024 would be represented as **124** because 124 years have passed since 1900.

Δημιουργώντας Timestamps

Η **time()** μπορεί να δημιουργήσει ένα **timestamp** μόνο για την τρέχουσα ημερομηνία, αλλά για οποιαδήποτε ημερομηνία χρησιμοποιούμε την **mktime() function.**
Η **mktime() function** μετατρέπει μια **datetime structure** σε ένα **timestamp.**

```
#include <iostream>
#include <ctime>  // Import the ctime library
using namespace std;
 int main () {
  struct tm datetime;
  time_t timestamp;
  datetime.tm_year = 2023 - 1900; // Number of years since 1900
  datetime.tm_mon = 12 - 1; // Number of months since January
  datetime.tm_mday = 17;
  datetime.tm_hour = 12;
  datetime.tm_min = 30;
  datetime.tm_sec = 1;
  // Daylight Savings must be specified
  // -1 uses the computer's timezone setting
  datetime.tm_isdst = -1;
```

```
  timestamp = mktime(&datetime);
  cout << ctime(&timestamp);
  return 0;

}
```

Sun Dec 17 12:30:01 2023

- Η **mktime() function** χρειάζεται αυτά τα μέλη: **tm_year, tm_mon, tm_mday, tm_hour, tm_min, tm_sec and tm_isdst.**

Δημιουργώντας Datetime Structures

```
#include <iostream>
#include <ctime>  // Import the ctime library
using namespace std;
 int main () {
  // Create the datetime structure and use mktime to fill in the missing members
  struct tm datetime;
  datetime.tm_year = 2023 - 1900; // Number of years since 1900
  datetime.tm_mon = 12 - 1; // Number of months since January
  datetime.tm_mday = 17;
  datetime.tm_hour = 0; datetime.tm_min = 0; datetime.tm_sec = 0;
  datetime.tm_isdst = -1;
  mktime(&datetime);
  string weekdays[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"};
  cout << "The date is on a " << weekdays[datetime.tm_wday];
  return 0;
}
```

The date is on a Sunday

The localtime() and gmtime() functions can convert timestamps into datetime structures.
The localtime() function returns a pointer to a structure representing the time in the computer's time zone.
The gmtime() function returns a pointer to a structure representing the time in the GMT time zone.
These functions return a **pointer** to a datetime structure. If we want to make sure its value does not change unexpectedly we should make a copy of it by dereferencing the pointer.

Get a datetime structure and output the current hour:

```
#include <iostream>
#include <ctime>  // Import the ctime library
```

```
using namespace std;
 int main () {
  time_t timestamp = time(&timestamp);
  struct tm datetime = *localtime(&timestamp);
  cout << datetime.tm_hour;
  return 0;
}
// Note: This example displays the server's local time, which may differ from your local time.
```

9

Display Dates

So far we have been using the ctime() function to display the date contained in a timestamp. To display dates from a datetime structure we can use the asctime() function.

Display the date represented by a datetime structure:

```
time_t timestamp = time(NULL);
#include <iostream>
#include <ctime>  // Import the ctime library
using namespace std;
 int main () {
  time_t timestamp = time(NULL);
  struct tm datetime = *localtime(&timestamp);
  cout << asctime(&datetime);
  return 0;
}
// Note: This example displays the server's local time, which may differ from your local time.
```

```
Mon Nov 24 09:47:09 2025
```

Note: The **asctime()** function does not correct invalid dates. For example, if you set the day of the month to 32 it will display 32. The **mktime()** function can correct these kinds of errors:

Correct a date before displaying it:

```
#include <iostream>
#include <ctime>  // Import the ctime library
using namespace std;
 int main () {
  // Create the datetime structure and use mktime to correct mistakes
  struct tm datetime;
  datetime.tm_year = 2022 - 1900; // Number of years since 1900
```

```
  datetime.tm_mon = 0; // 0 is January
  datetime.tm_mday = 32;
  datetime.tm_hour = 0; datetime.tm_min = 0; datetime.tm_sec = 0;
  datetime.tm_isdst = -1;
  mktime(&datetime);
  cout << asctime(&datetime);
  return 0;
}
```

Tue Feb 1 00:00:00 2022

The **ctime()** and **asctime()** functions allow us to display the date but they do not allow us to choose how it is displayed.
To choose how a date is displayed we can use the **strftime() function**.

Η τρέχουσα ημερομηνία με διαφορετικούς τρόπους:

```
#include <iostream>
#include <ctime>  // Import the ctime library
using namespace std;
 int main() {
  time_t timestamp = time(NULL);
  struct tm datetime = *localtime(&timestamp);
  char output[50];
  strftime(output, 50, "%B %e, %Y", &datetime);
  cout << output << "\n";
  strftime(output, 50, "%I:%M:%S %p", &datetime);
  cout << output << "\n";
  strftime(output, 50, "%m/%d/%y", &datetime);
  cout << output << "\n";
  strftime(output, 50, "%a %b %e %H:%M:%S %Y", &datetime);
  cout << output << "\n";
  return 0;
}
// Note: This example displays the server's local time, which may differ from your local time.
```

November 24, 2025
09:48:34 AM
11/24/25
Mon Nov 24 09:48:34 2025

Σπ. Βασιλοπούλου, Β. Σακκάς - θεωρία & Ασκήσεις για το Σεμιναριακό Μάθημα «Πληροφορική», (ΣΜ002), 2025.

Βιβλιογραφικές Αναφορές

https://www.w3schools.com/cpp/

https://apothesis.eap.gr/handle/repo/49120

http://users.uoa.gr/~vassilopoulou/genima

Βεσκούκης, Β., 2000. - Τεχνολογία Λογισμικού Ι, ΕΑΠ, Πάτρα.

Βεσκούκης, Β., 2001. - Τεχνολογία Λογισμικού ΙΙ, ΕΑΠ, Πάτρα.

Θραμπουλίδης, Κ., 2001. - Γλώσσες Προγραμματισμού ΙΙ (Αντικειμενοστρεφής Προγραμματισμός). – ΕΑΠ, Πάτρα.

Καμμέας Α. 2000. - Τεχνικές Προγραμματισμού, τ.β', ΕΑΠ, Πάτρα.

Καρύδης, Ι., Σιούτας, Σ., Τζήμας, Ι., 2015. - Σύγχρονες Μέθοδοι Προγραμματισμού, τ.Γ'. - ΕΑΠ, Πάτρα.

Μαυρομμάτης, Γ., Σακκόπουλος, Ε., 2015. – Γλώσσες Προγραμματισμού, τ. Β'. - ΕΑΠ, Πάτρα.

Μαργαρίτης Κωνσταντίνος, Τσιμπίκας, Γιάννης, 2004. - Εισαγωγή στη C++ - eBooks4Greeks.gr

Ντόκας, Γ. - Εισαγωγή στη Γλώσσα Προγραμματισμού C++ - Εργαστηριακές Σημειώσεις. Τμήμα Διαχείρισης Πολιτισμικού Περιβάλλοντος & Νέων Τεχνολογιών, Παν/μιο Δυτικής Ελλάδας.

Χατζηγεωργίου Α., 2008. – Ανάπτυξη Συστήματος Λογισμικού βάσει της μεθοδολογίας ICONIX. –ΕΑΠ, ΠΑΤΡΑ.