Γ Λυκείου

Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Περιεχόμενα
ΒΙ. Κεφάλαιο 6

Κατηφόρης Παναγιώτης
2019-2020

6.1 Η έννοια του Προγράμματος

Επίλυση προβλήματος με τον υπολογιστή

Η **επίλυση** ενός προβλήματος με τον υπολογιστή περιλαμβάνει:

- Τον ακριβή προσδιορισμό του προβλήματος
- Την ανάπτυξη του αντίστοιχου αλγόριθμου
- Τη διατύπωση του αλγόριθμου σε κατανοητή μορφή από τον υπολογιστή

Ο προγραμματισμός (το τελευταίο στάδιο της επίλυσης) είναι η διατύπωση του αλγορίθμου σε μορφή κατανοητή από τον Η/Υ (ώστε να τον εκτελέσει -«τρέξει» όπως λέμε στην ορολογία της Πληροφορικής) το **πρόγραμμα.**
Η διατύπωση γίνεται χρησιμοποιώντας μία **γλώσσα προγραμματισμού**.
Τα προγράμματα (και γενικά το λογισμικό) είναι που προσδίδουν σε ένα υπολογιστικό σύστημα (μηχανή - υλικό – hardware) την αίσθηση της έξυπνης μηχανής.
Μπορούμε, λοιπόν, να συνδέσουμε – συνδυάσουμε τις 3 έννοιες που είδαμε μέχρι τώρα με τη βοήθεια του παρακάτω σχήματος.

Οι γλώσσες προγραμματισμού.

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου (προγραμματιστή) με τη μηχανή (υπολογιστής).

Προσοχή.
Το πρόγραμμα, το οποίο, το οποίο γράφεται σε μια γλώσσα προγραμματισμού, δεν είναι απλά η υλοποίηση του αλγόριθμου, αλλά βασικό του στοιχείο είναι τα δεδομένα και οι δομές δεδομένων επί των οποίων ενεργεί.
Όπως ξέρουμε: ΠΡΟΓΡΑΜΜΑ = ΑΛΓΟΡΙΘΜΟΣ + ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
Δηλαδή, ο αλγόριθμος και οι δομές δεδομένων είναι μια αδιάσπαστη ενότητα.

6.4 Τεχνικές Σχεδίασης Προγράμματος

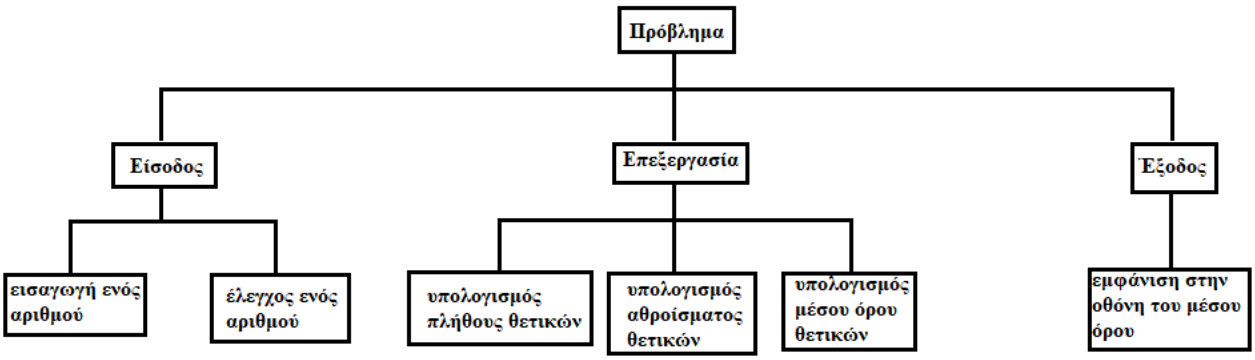
Δεν αρκεί κάποιος να γνωρίζει απλά μία γλώσσα προγραμματισμού. Θα πρέπει να ακολουθήσει και μία τεχνική σχεδίασης του προγράμματός του. Για την σύνταξη σωστών, κομψών, κατανοητών και εύκολα συντηρήσιμων προγραμμάτων ακολουθήθηκαν διάφορες μεθοδολογίες ανάπτυξης που παρουσιάζονται παρακάτω:

6.4.1. Ιεραρχική σχεδίαση ή Top-down σχεδίαση

Η τεχνική αυτή (η οποία λέγεται και σχεδίαση "**από επάνω προς τα κάτω**") περιλαμβάνει στον καθορισμό των βασικών λειτουργιών του προγράμματος σε ανώτερο επίπεδο και στη συνέχεια τη διάσπαση καθεμιάς σε μικρότερες και απλούστερες μέχρι του σημείου να είναι τόσο απλές που μπορούν να επιλυθούν εύκολα.
Η ιεραρχική σχεδίαση ή ιεραρχικός προγραμματισμός χρησιμοποιεί την στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα.
Σκοπός της ιεραρχικής σχεδίασης είναι η διάσπαση του αρχικού προβλήματος σε απλούστερα υποπροβλήματα τα οποία είναι πιο εύκολο να επιλυθούν, οδηγώντας έτσι στη λύση του αρχικού προβλήματος.
Για την υποβοήθηση της ιεραρχικής σχεδίασης χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές.
Παράδειγμα.
Να γραφεί πρόγραμμα το οποίο θα δέχεται από το πληκτρολόγιο 1.000 ακέραιους αριθμούς και θα υπολογίζει τον μέσο όρο μόνο των θετικών

Είσοδος (δεδομένα)	**α (οι αριθμοί που εισάγονται)**
Έξοδος (ζητούμενα)	μο (ο μ.όρος των θετικών)

Ιεραρχική σχεδίαση



6.4.2. Τμηματικός προγραμματισμός

Η ιεραρχική σχεδίαση υλοποιείται με τον τμηματικό προγραμματισμό.

Τμηματικός προγραμματισμός λέμε τη τεχνική σχεδίασης και ανάπτυξης προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.

Κάθε υποπρόβλημα που προκύπτει από την ανάλυση του αρχικού προβλήματος, αντιμετωπίζεται ως ανεξάρτητη **ενότητα.** Γι' αυτή την ενότητα θα γραφτεί το κατάλληλο πρόγραμμα ή τμήμα προγράμματος που αντιμετωπίζει αποκλειστικά αυτή την ενότητα.

Πλεονεκτήματα του τμηματικού προγραμματισμού

- Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος.
- Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.
- Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος.
- Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.

6.4.3. Δομημένος προγραμματισμός

Είναι η μεθοδολογία που έχει επικρατήσει σήμερα. Υποστηρίζεται από όλες, σχεδόν, τις γλώσσες προγραμματισμού.

Παρουσιάστηκε στα μέσα του 1960. Την εποχή εκείνη τα προγράμματα ήταν πολύπλοκα και μπερδεμένα. Απαιτούνταν πολύς χρόνος τόσο για τη συγγραφή του προγράμματος όσο και για τη διόρθωση και συντήρησή τους. Αυτό οφείλονταν στη GOTO η οποία άλλαζε συνεχώς τη ροή του προγράμματος. Η συνεχής χρήση της εντολής GOTO ήταν υπεύθυνη:

- για την δυσκολία στην αρχική σχεδίαση του προγράμματος
- για την παρακολούθηση και κατανόηση του προγράμματος
- στην συντήρηση του προγράμματος

Ο δομημένος προγραμματισμός

- στηρίζεται στη χρήση τριών και μόνο στοιχειωδών λογικών δομών, τη δομή **ακολουθίας**, τη δομή **επιλογής** και τη δομή **επανάληψης**.
- Όλα τα προγράμματα μπορεί να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους.
- Κάθε πρόγραμμα, όπως και κάθε ενότητα προγράμματος έχει μόνο **μία είσοδο** και **μία έξοδο**.

Ο δομημένος προγραμματισμός Αποθαρρύνει τη χρήση της εντολής GOTO.

Πλεονεκτήματα του δομημένου προγραμματισμού

- Δημιουργία απλούστερων προγραμμάτων
- Άμεση μεταφορά του αλγόριθμου σε πρόγραμμα
- Διευκόλυνση της ανάλυσης του προγράμματος σε τμήματα (ενότητες).
- Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος
- Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.
- Ευκολότερη διόρθωση και συντήρηση του προγράμματος

Το πρόγραμμα

Ως ενιαίο σύνολο εντολών	Σε λογικές ενότητες (τμηματικός προγραμματισμός)
ΠΡΟΓΡΑΜΜΑ Παράδειγμα ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: α, πλθ, αθρ ΠΡΑΓΜΑΤΙΚΕΣ: μο ΑΡΧΗ πλθ←0 αθρ←0 ΓΙΑ Ι ΑΠΌ 1 ΜΕΧΡΙ 1000 ΓΡΑΨΕ “Δώσε ακέραιο” ΔΙΑΒΑΣΕ α ΑΝ α>0 ΤΟΤΕ πλθ←πλθ+1 αθρ←αθρ+α ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ΑΝ πλθ<>0 ΤΟΤΕ μο←αθρ/πλθ ΓΡΑΨΕ “Ο μ.όρος είναι”,μο ΑΛΛΙΩΣ ΓΡΑΨΕ “Δεν δόθηκε κανένας θετικός” ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ	ΠΡΟΓΡΑΜΜΑ Παράδειγμα ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: α, πλθ, αθρ ΑΡΧΗ πλθ←0 αθρ←0 ΓΙΑ Ι ΑΠΌ 1 ΜΕΧΡΙ 1000 ΚΑΛΕΣΕ εισαγωγή(α) ΑΝ έλεγχος(α) ΤΟΤΕ ΚΑΛΕΣΕ υπολογισμοί(πλθ,αθρ) ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ΚΑΛΕΣΕ έξοδος(πλθ,αθρ) ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ --- ΔΙΑΔΙΚΑΣΙΑ εισαγωγη(χ) ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: χ ΑΡΧΗ ΓΡΑΨΕ “Δώσε ακέραιο” ΔΙΑΒΑΣΕ χ ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ --- ΣΥΝΑΡΤΗΣΗ έλεγχος(χ):ΛΟΓΙΚΗ ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: χ ΑΡΧΗ ΑΝ χ>0 ΤΟΤΕ έλεγχος←ΑΛΗΘΗΣ ΑΛΛΙΩΣ έλεγχος←ΨΕΥΔΗΣ ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ --- ΔΙΑΔΙΚΑΣΙΑ υπολογισμοί(π,α,χ) ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: π,α,χ ΑΡΧΗ π←π+1 α←α+χ ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ --- ΔΙΑΔΙΚΑΣΙΑ έξοδος(π,α) ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: π,α,χ ΑΡΧΗ ΑΝ π<>0 ΤΟΤΕ ΓΡΑΨΕ “Ο μ.όρος είναι”,α/π ΑΛΛΙΩΣ ΓΡΑΨΕ “Δεν δόθηκε κανένας θετικός” ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

6.5. Αντικειμενοστραφής προγραμματισμός

Ένα πρόγραμμα περιγράφει "**ενέργειες**" (επεξεργασία) που εφαρμόζονται πάνω σε **δεδομένα**. Ένα βασικό ερώτημα που τίθεται είναι αν η δομή του προγράμματος είναι προτιμότερο να στηρίζεται στις "ενέργειες" ή στα δεδομένα. Η απάντηση σε αυτό το ερώτημα προσδιορίζει και τη βασική διαφορά ανάμεσα στον **διαδικασιακό** (ή **διαδικαστικό**) προγραμματισμό και τον **αντικειμενοστραφή** προγραμματισμό.
Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα (objects). Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησι-μοποιήσιμα.
Φυσικά ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

6.3 Φυσικές και Τεχνητές γλώσσες

Τι είναι

Φυσικές γλώσσες: Δημιουργηθεί από κάποιο λαό, επομένως από πολλούς ανθρώπους και στόμα με στόμα. Χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων (π.χ. Ελληνικά, Ιταλικά Αγγλικά κλπ).
Τεχνητές γλώσσες: είναι γλώσσες των οποίων η φωνολογία, η γραμματική και το λεξιλόγιο, έχει επινοηθεί συνειδητά από ένα άτομο ή μια ομάδα, αντί να έχει εξελιχθεί φυσικά. Οι **Γλώσσες προγραμματισμού**: είναι τεχνητές γλώσσες που απευθύνονται σε ανθρώπους που επιθυμούν να επικοινωνήσουν με τον Η/Υ.

Ομοιότητες φυσικών γλωσσών και γλωσσών προγραμματισμού

Κάθε γλώσσα (φυσική ή τεχνική) προσδιορίζεται από:

1. **Το αλφάβητο της γλώσσας**

Ως αλφάβητο ορίζουμε το σύνολο των στοιχείων (χαρακτήρων) που χρησιμοποιείται από τη γλώσσα. Από τους χαρακτήρες αυτούς σχηματίζονται οι λέξεις της γλώσσας.

2. **Το λεξιλόγιο της γλώσσας**

Το λεξιλόγιο μίας γλώσσας περιλαμβάνει όλες τις έγκυρες και αποδεκτές λέξεις. Στην ουσία, είναι ένα υποσύνολο από όλες τις δυνατές ακολουθίες που μπορούμε να σχηματίσουμε από τα στοιχεία του αλφαβήτου. Για παράδειγμα η λέξη ΠΕΡΠΑΤΩ είναι αποδεκτή στην ελληνική γλώσσα, ενώ η λέξη ΠΑΡΤΠΩΕ όχι.

3. **Τη γραμματική της γλώσσας**

Η γραμματική αποτελείται από το *τυπικό* ή *τυπολογικό* και το *συντακτικό*.

- Το ***τυπικό ή τυπολογικό*** είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μια λέξη είναι αποδεκτή. Για παράδειγμα η λέξη ΠΕΡΠΑΤΩ είναι αποδεκτή και στις μορφές ΠΕΡΠΑΤΗΣΑ, ΠΕΡΠΑΤΟΥΝ αλλά όχι στη μορφή ΠΕΡΠΑΤΟΥΣ.
- Το ***συντακτικό*** είναι ένα σύνολο κανόνων που ορίζει το πώς πρέπει να σχηματίζονται οι προτάσεις από τις λέξεις της γλώσσας ώστε οι προτάσεις αυτές να είναι έγκυρες και αποδεκτές. Σε μία γλώσσα προγραμματισμού η γνώση του συντακτικού επιτρέπει την σωστή σύνταξη των εντολών.

4. **Τη σημασιολογία της γλώσσας**

Είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και προτάσεων της γλώσσας. Σε μία γλώσσα προγραμματισμού αυτό καθορίζεται από το δημιουργό της ενώ σε μία φυσική γλώσσα από αυτόν που εκφέρει την πρόταση.

Διαφορές φυσικών γλωσσών και γλωσσών προγραμματισμού

1. **Χρήση**.
   Οι ***φυσικές γλώσσες*** χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπων, ενώ οι ***γλώσσες προγραμματισμού*** χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπου και η/υ.
2. **Εξέλιξη**
   Οι ***φυσικές γλώσσες*** έχουν μεγάλες δυνατότητες εξέλιξης. Οι φυσικές γλώσσες είναι «ζωντανές». Νέες λέξεις μπορεί να εισαχθούν, κανόνες γραμματικής και σύνταξης να αλλάξουν, λέξεις αλλάζουν σημασία ή να καταργούνται .κλπ.
   Αντίθετα
   Οι ***γλώσσες προγραμματισμού*** δεν εξελίσσονται ή έχουν περιορισμένες δυνατότητες εξέλιξης. Τις περισσότερες φορές η εξέλιξη αυτή αφορά την **διόρθωση αδυναμιών** ή την κάλυψη **μεγαλύτερου εύρους εφαρμογών** ή τέλος για να ακολουθήσουν νέες τάσεις στον προγραμματισμό. (π.χ. Basic και Visual Basic).

6.7 Προγραμματιστικά Περιβάλλοντα

Ένα πρόγραμμα που φτιάχνεται σε μία γλώσσα υψηλού επιπέδου δεν είναι άμεσα κατανοητό από τον η/υ. Θα πρέπει να μεταφραστεί σε ισοδύναμο πρόγραμμα σε γλώσσα μηχανής (δυαδική μορφή). Την διαδικασία μετάφρασης την πραγματοποιούν τα *μεταφραστικά προγράμματα* που είναι δύο ειδών οι **μεταγλωττιστές (compilers)** και οι **διερμηνευτές (interpreters)**

Σημαντικοί ορισμοί

Πηγαίο πρόγραμμα (Source program): Το αρχικό πρόγραμμα, το πρόγραμμα που είναι γραμμένο σε μια γλώσσα υψηλού επιπέδου, το πρόγραμμα που συντάσσει ο προγραμματιστής.. Το πηγαίο πρόγραμμα αποτελεί την είσοδο στον μεταγλωττιστή και το διερμηνέα.

Αντικείμενο πρόγραμμα (Object program) : Το πρόγραμμα που παράγεται από το μεταγλωττιστή μετά τη μεταγλώττιση του σε γλώσσα μηχανής. Αντικείμενο πρόγραμμα παράγει μόνο ο μεταγλωττιστής. (Το αντικείμενο πρόγραμμα παρόλο που είναι σε μορφή κατανοητή από τον υπολογιστή δεν είναι σε θέση να εκτελεστεί).

Βιβλιοθήκες (Libraries) : Έτοιμες ενότητες αντικείμενου προγράμματος της γλώσσας, απαραίτητες για την παραγωγή του εκτελέσιμου προγράμματος.

Συνδέτης – Φορτωτής (Linker-Loader) : Ειδικό πρόγραμμα που αναλαμβάνει να συνδέσει το αντικείμενο πρόγραμμα με άλλα τμήματα προγράμματος (που βρίσκονται στις βιβλιοθήκες της γλώσσας ή του προγραμματιστή) και να παράγει το εκτελέσιμο.

Εκτελέσιμο πρόγραμμα (Executable program) : Το πρόγραμμα που παράγεται από το συνδέτη – φορτωτή. Είναι το τελικό πρόγραμμα, αυτό που θα εκτελεστεί από τον υπολογιστή.

Συντάκτης (editor) : Είναι ένα ειδικό πρόγραμμα που χρησιμοποιείται για την αρχική σύνταξη των προγραμμάτων καθώς και για την πιθανή διόρθωση τους. Μοιάζει με επεξεργαστή κειμένου με επιπλέον δυνατότητες που διευκολύνουν την γρήγορη σύνταξη των πηγαίων προγραμμάτων.

Λάθη που παρουσιάζονται στο πρόγραμμα

Κατά τη δημιουργία ενός προγράμματος σχεδόν πάντα ενυπάρχουν λάθη. Τα λάθη τα χωρίζουμε σε τρις κατηγορίες

- **Συντακτικά λάθη.**

  Οφείλονται: σε αναγραμματισμούς ονομάτων εντολών, λάθος σύνταξης εντολών, μη δήλωση δεδομένων.
  Εμφανίζονται: κατά τη διαδικασία της **μεταγλώττισης** ή **διερμήνευσης**. Ο μεταγλωττιστής ή ο διερμηνευτής παρουσιάζει προειδοποιητικά μηνύματα, με τη βοήθεια των οποίων εύκολα και γρήγορα εντοπίζουμε και διορθώνουμε τα λάθη
  Αντιμετώπιση: ο προγραμματιστής επιστρέφει στο πηγαίο πρόγραμμα, διορθώνει τα λάθη και ξαναϋποβάλλει το πηγαίο για μεταγλώττιση
  Παραδείγματα: αντί για το σωστό **ΔΙΑΒΑΣΕ Χ** το **ΔΙΑΒΣΕ Χ**

- **Λογικά λάθη**

  Οφείλονται: σε σφάλματα στη λογική επίλυσης του προβλήματος ή σε λανθασμένη διατύπωση του αλγόριθμου
  Εμφανίζονται: κατά την **εκτέλεση** του προγράμματος. Τα αντιλαμβανόμαστε από τα λαθεμένα αποτελέσματα που παράγει το πρόγραμμα
  Αντιμετώπιση: είναι τα πλέον σοβαρά και δύσκολα στη ανίχνευση τους. Διαπιστώνονται με τη διαδικασία ελέγχου και την ανάλυση των αποτελεσμάτων του προγράμματος
  Παράδειγμα: αντί του σωστού **Α ← Β + Γ** γράφω **Α ← Β * Γ**

- **Λάθη που εμφανίζονται κατά την εκτέλεση του προγράμματος.**

Παράδειγμα: Διαίρεση με το μηδέν, κλειστός εκτυπωτής, αποσύνδεση από το Διαδίκτυο κλπ.

Μεταφραστικά προγράμματα

Ο **Μεταγλωττιστής (Compiler).** δέχεται στην είσοδο ένα πρόγραμμα που έχει γραφεί σε μια γλώσσα υψηλού επιπέδου (**πηγαίο**) και αφού το ελέγξει για ύπαρξη συντακτικών λαθών, παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής (**αντικείμενο**). Το πρόγραμμα που παράγεται (**αντικείμενο**), είναι ανεξάρτητο του πρώτου (**πηγαίο**), και μετά από την «παρέμβαση» το **συνδέτη-φορτωτή**, δημιουργείται το **εκτελέσιμο** πρόγραμμα, το οποίο μπορεί να εκτελεστεί οποιαδήποτε στιγμή από οποιονδήποτε υπολογιστή.
Ο **Διερμηνευτής (Interpreter).** διαβάζει μια – μια τις εντολές του αρχικού προγράμματος (**πηγαίο**) και για κάθε μια εντολή αφού κάμει έλεγχο για συντακτικά λάθη, εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών γλώσσας μηχανής.

Παρατηρήσεις στη χρήση μεταγλωττιστή και διερμηνευτή
Ομοιότητες

- ✓ Και οι δύο μεταφράζουν μεταφράζουν το πηγαίο πρόγραμμα (υψηλού επιπέδου) σε γλώσσα μηχανής
- ✓ Τόσο ο μεταγλωττιστής όσο και ο διερμηνευτής **ανιχνεύουν τα συντακτικά λάθη**, εμφανίζοντας κατάλληλα διαγνωστικά μηνύματα.

Διαφορές

- ✓ Με την χρήση του **μεταγλωττιστή** ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της μεταγλώττισης και σύνδεσης ώστε να παραχθεί το **εκτελέσιμο πρόγραμμα**. Το τελευταίο εκτελείται από κάθε η/υ.
- ✓ Από την άλλη μεριά ο **διερμηνευτής** «τσιμπάει» μία - μία τις εντολές του πηγαίου προγράμματος. Κάθε μία εντολή την μεταφράζει, σε γλώσσα μηχανής, και στην συνέχεια την εκτελεί. Αμέσως μετά επιστρέφει στο πηγαίο για την επόμενη εντολή κ.ο.κ.
- ✓ Η εκτέλεση ενός προγράμματος με τον **διερμηνευτή είναι πιο αργή**, γιατί για να εκτελεστεί το πρόγραμμα, πρέπει *κάθε φορά να ξαναγίνεται η διερμηνεία* από την αρχή, ενώ ο **μεταγλωττιστής** *παράγει μια φορά το αντικείμενο πρόγραμμα* και δεν χρειάζεται ξανά μεταγλώττιση.
- ✓ Για να εκτελεστεί ένα προγράμμα με τον **διερμηνευτή** *είναι απαραίτητη η παρουσία του πηγαίου* προγράμματος ενώ με τον **μεταγλωττιστή** *μόνο την πρώτη φορά*.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με μεικτές υλοποιήσεις. Δηλαδή χρησιμοποιείται:

Διερμηνευτής κατά τη φάση της δημιουργίας του προγράμματος.

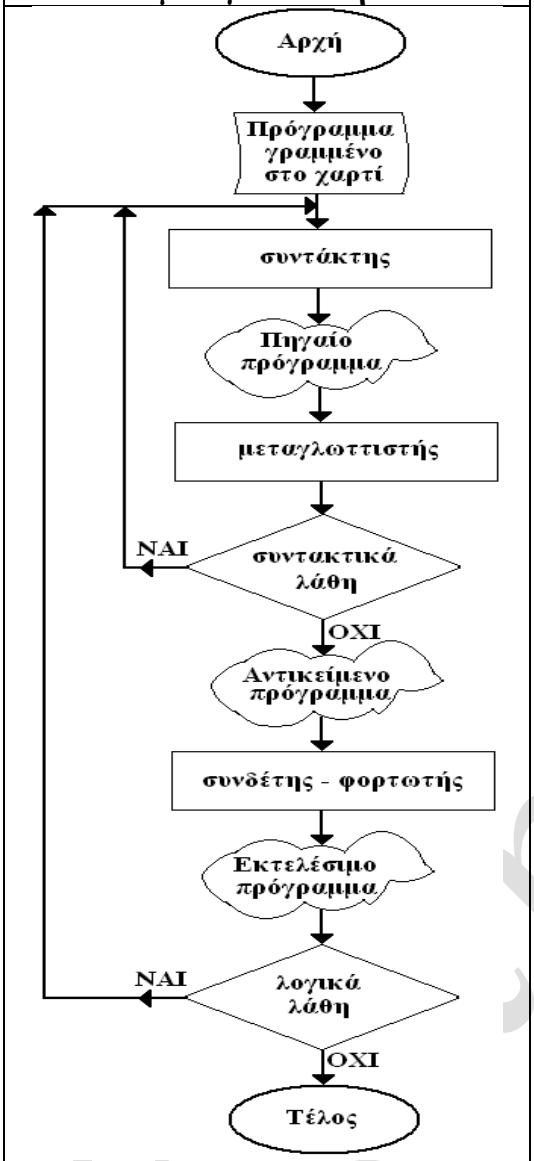
Ο διερμηνευτής αφού εκτελεί τις εντολές μία μία έχει το πλεονέκτημα της άμεσης διόρθωσης των λαθών. Για τον λόγο αυτό χρησιμοποιείται συνήθως κατά την συγγραφή-διόρθωση ενός προγράμματος.

Μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος (εκτελέσιμο πρόγραμμα).

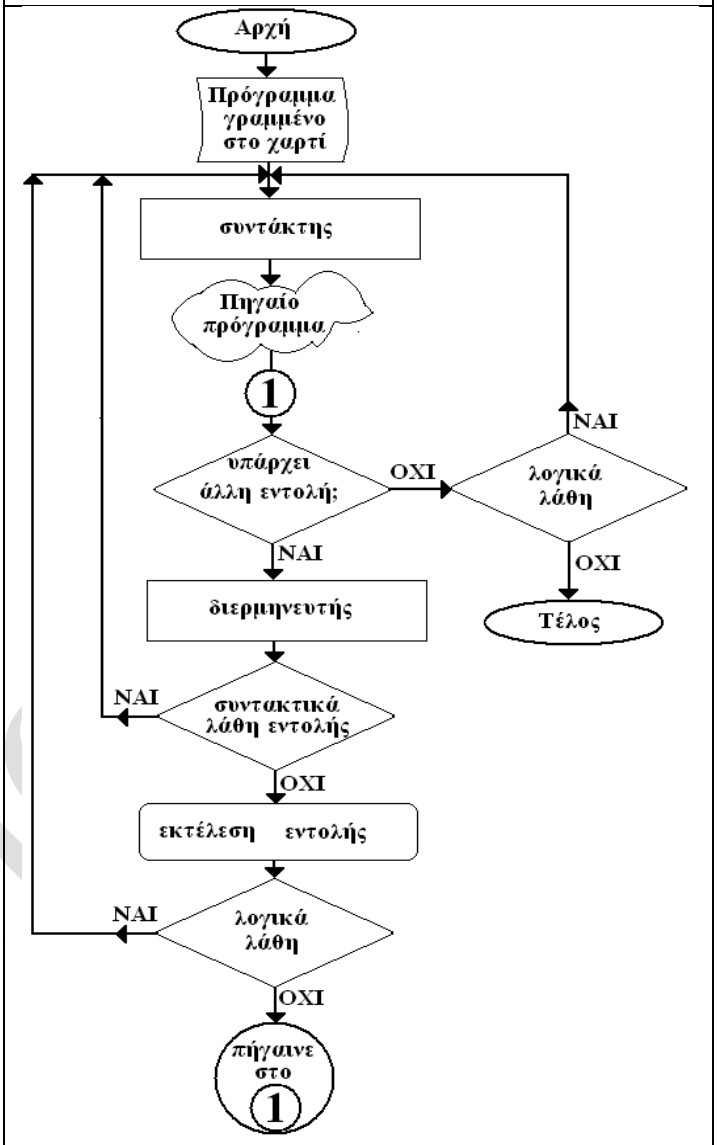
Το *εκτελέσιμο πρόγραμμα* είναι το «προϊόν» που παράγει ο προγραμματιστής, είναι το «προϊόν» που χρησιμοποιεί ο χρήστης και εκτελείται από κάθε υπολογιστή.

ΣΥΓΓΡΑΦΗ ΚΑΙ ΔΙΟΡΘΩΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Διαδικασία **συγγραφής**, **διόρθωσης** και **μεταγλώττισης** ενός προγράμματος με τον **μεταγλωττιστή**.



Διαδικασία **συγγραφής**, **διόρθωσης** και **διερμήνευσης** ενός προγράμματος με τον **διερμηνευτή**.



Σύγχρονα υπολογιστικά περιβάλλοντα

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν, στον προγραμματιστή, όλα εκείνα τα εργαλεία που διευκολύνουν την εύκολη, ταχύτατη σύνταξη, διόρθωση και συντήρηση των προγραμμάτων.

Μεταξύ των εργαλείων αυτών υπάρχουν οπωσδήποτε τα εξής:

- Τον **συντάκτη**, για την συγγραφή και διόρθωση του πηγαίου προγράμματος
- Τον **μεταγλωττιστή** και τον **συνδέτη – φορτωτή** για την παραγωγή του εκτελέσιμου.
- Τον **διερμηνευτή** για την εκτέλεση του προγράμματος γραμμή – γραμμή, ώστε να ανιχνεύονται πιθανά λογικά λάθη ή λάθη χρόνου εκτέλεσης.