Ελληνικό Μεσογειακό Πανεπιστήμιο
Σχολή Μηχανικών / Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

HMY01K06
Επιστημονικός Προγραμματισμός με Python

Διάλεξη 01
Προβλήματα / Αλγόριθμοι / Γλώσσες Προγραμματισμού

Φθινόπωρο 2026

Προβλήματα και Αλγόριθμοι

Πρόβλημα

Μια κατάσταση η οποία είναι *μη αποδεκτή ως έχει*.

Απαιτεί λύση (ή απάντηση) η οποία δεν είναι *ούτε γνωστή ούτε προφανής*.

Λύση / Απάντηση του προβλήματος

Μια σειρά από ενέργειες ή σκέψεις που μετασχηματίζουν την αρχική κατάσταση σε μία νέα αποδεκτή κατάσταση

Προϋποθέσεις κατανόησης και επίλυσης

- σωστή διατύπωση από μέρους του δημιουργού του προβλήματος
- σωστή ερμηνεία από την πλευρά του λύτη

Κατηγορίες προβλημάτων

- **Επιλύσιμα**

  Προβλήματα για τα οποία η *λύση* έχει *βρεθεί* και έχει *διατυπωθεί*

  Η επίλυση δευτεροβάθμιας εξίσωσης $(\alpha x^2+\beta x+\gamma = 0)$

- **Μη επιλύσιμα**

  Προβλήματα για τα οποία έχει *αποδειχτεί*, ότι *δεν επιδέχονται λύση*

  Ο τετραγωνισμός του κύκλου με κανόνα και διαβήτη

  Η εύρεση των λύσεων οποιασδήποτε ακέραιας πολυωνυμικής εξίσωσης (πχ $6x + 15y = 4$)

- **Ανοικτά**

  Προβλήματα για τα οποία η λύση *τους δεν έχει ακόμα βρεθεί*,
  ενώ ταυτόχρονα *δεν έχει αποδειχτεί, ότι δεν επιδέχονται λύση*

  Το πρόβλημα της ενοποίησης των τεσσάρων πεδίων δυνάμεων

  Η εικασία του Goldbach (κάθε άρτιος μπορεί να γραφτεί ως άθροισμα δύο πρώτων αριθμών)

Στάδια αντιμετώπισης ενός προβλήματος

Σωστή και πλήρης *αποσαφήνιση* των *δεδομένων* και των *ζητούμενων* του προβλήματος

Διάσπαση του αρχικού πρόβλημα τος σε άλλα επιμέρους *απλούστερα* προβλήματα

Υλοποίηση της λύσης μέσω της *επίλυσης* των *επιμέρους προβλημάτων* και της *σύνθεσης* των αποτελεσμάτων τους

Υπολογιστικό Πρόβλημα

Οποιοδήποτε (επιλύσιμο) που πρόβλημα μπορεί να λυθεί *και* με τη βοήθεια του Η/Υ

Παραδείγματα

- Η επίλυση της *δευτεροβάθμιας εξίσωσης* (τριώνυμο).
- Η *ταξινόμηση* ονομάτων/λέξεων σε *αλφαβητική σειρά*.
- Ο *γεωμετρικός μετασχηματισμός* μιας *εικόνας* (ανάκλαση /περιστροφή)
- Η εύρεση της *συντομότερης διαδρομής* που θα κάνει κάποιος για να επισκεφθεί *ένα πλήθος διαφορετικών πόλεων* και να επιστρέψει *πίσω απ' όπου ξεκίνησε* περνώντας *μόνο μία φορά* από κάθε πόλη(*) (πρόβλημα περιοδεύοντος πωλητή – travelling salesman problem)

(*) *με βάση έναν δεδομένο χάρτη των πόλεων και των δρόμων που τις συνδέουν*

Διαδικασίες και Αλγόριθμοι

Διαδικασία

- Ενας *καθιερωμένος* ή *επίσημος* τρόπος για να επιτευχθεί ένα συγκεκριμένο αποτέλεσμα.
- Περιγράφεται ως ενα σύνολο *ενεργειών* οι οποίες πρέπει να εκτελεστούν με *συγκεκριμένη σειρά* ή *τρόπο* για την επίτευξη του αποτελέσματος.
- Οι ενέργειες αυτές ονομάζονται *βήματα* της διαδικασίας.

Παραδείγματα διαδικασιών στην καθημερινότητά μας

Συνταγές μαγειρικής, οδηγίες χρήσης, βήματα χορού…

Διαδικασίες και Αλγόριθμοι

Αλγόριθμος

- Μια διαδικασία με *συγκεκριμένες ιδιότητες* (*) που την κάνουν να ξεχωρίζει με ειδικό τρόπο.

- Η ονομασία προέρχεται από το όνομα του μαθηματικού, αστρονόμου και γεωγράφου *Mohammad Ibn Musa* ***al'Khowârizmî*** που έζησε στην Περσία τον 9ο αιώνα μ.χ. και θεωρείται ως ο πατέρας της σύγχρονης *άλγεβρας.*
- *Η λέξη «al'Khowârizmî » σημαίνει «από το Khowarazm», που είναι η σημερινή πόλη Khiva του Ουζμπεκιστάν*

(*) *θα τις δούμε στη συνέχεια*

Βασικές Ιδιότητες ενός Αλγόριθμου

Finiteness *Χρήση πεπερασμένων πόρων*

- Οι ανάγκες ενός αλγορίθμου σε *χώρο* και *χρόνο* πρέπει να είναι πεπερασμένες

  Ένας αλγόριθμος πρέπει να υλοποιείται μέσα σε *πεπερασμένο χώρο* και να τερματίζει μετά από *πεπερασμένο αριθμό βημάτων.*

- *Υπολογιστικές διαδικασίες που στερούνται αυτή την ιδιότητα – ενώ έχουν όλες τις υπόλοιπες- ονομάζονται* ***Υπολογιστικές Μέθοδοι*** *(computational methods)*

Βασικές Ιδιότητες ενός Αλγόριθμου

Definiteness Σαφήνεια / Καθοριστικότατα

- Τα βήματα ενός αλγόριθμου πρέπει να είναι σαφώς καθορισμένα και λεπτομερή.

Κάθε βήμα του αλγόριθμου πρέπει να περιγράφεται με τέτοιο τρόπο ώστε αφ' ενός να είναι πλήρως κατανοητός ο τρόπος εκτέλεσής του και αφ' εταίρου να μην αποδέχεται διαφορετικές ερμηνείες από διαφορετικούς ανθρώπους

Έτσι πχ όταν σε κάποια σημεία μιας συνταγής μαγειρικής διαβάζουμε φράσεις όπως "…λίγο αλάτι…" ή "…όσο αλεύρι χρειαστεί…" η διαδικασία αυτή δεν είναι σαφής, επομένως δεν είναι αλγόριθμος.

Βασικές Ιδιότητες ενός Αλγόριθμου

Effectiveness *Εφικτότητα / Αποτελεσματικότητα*

- Όλα τα βήματα ενός αλγορίθμου πρέπει να μπορούν να αναπαραχθούν με *ακρίβεια* και σε *πεπερασμένο χρονικό διάστημα και* εν τέλει να παράγουν το επιθυμητό αποτέλεσμα

- Κάθε βήμα πρέπει να είναι *επαρκώς βασικό* ώστε να μπορεί να αναπαραχθεί (με ακρίβεια και σε πεπερασμένο χρόνο) από κάποιον πχ που χρησιμοποιεί μόνο *μολύβι και χαρτί*

Βασικές Ιδιότητες ενός Αλγόριθμου

Input / Output *Είσοδος / Έξοδος*

Ένας αλγόριθμος τροφοδοτείται ***πάντα*** με *δεδομένα* και παράγει ***πάντα*** *αποτελέσματα*

Τα δεδομένα μπορεί είτε να *προϋπάρχουν ενσωματωμένα* στον ίδιο τον αλγόριθμο, είτε να *εισάγονται δυναμικά* κατά την εκτέλεσή του.

Τα αποτελέσματα είναι οι ποσότητες που *προκύπτουν* από τα *δεδομένα εισόδου* μετά από *επεξεργασία*.

Ο αλγόριθμος για να φτιάξεις ένα φλυτζάνι *ζεστό* τσάι

1. Βάλε **1 *φακελάκι* τσάι** σε ένα φλιτζάνι
2. Γέμισε το βραστήρα με **300 *ml* νερό**
3. Βράσε το νερό στο βραστήρα
4. Ρίξε το νερό από το βραστήρα στο φλιτζάνι
5. Πρόσθεσε **25 *ml* γάλα** στο φλιτζάνι
6. Πρόσθεσε **2 *κουταλάκια* ζάχαρη** στο φλιτζάνι
7. Ανακάτεψε το φλυτζάνι και περίμενε **3 *λεπτά***
8. Το τσάι είναι έτοιμο να το πιείς

- Παρόλο που τα βήματα πρέπει να ακολουθηθούν με *συγκεκριμένη σειρά*, κάποια βήματα μπορούν να *αλλάξουν σειρά χωρίς πρόβλημα* *(πχ τα βήματα 5 και 6 μπορούν να αναστραφούν)*
- Αν *αφαιρέσουμε* κάποια βήματα τότε ο αλγόριθμός μας *πιθανόν να εξακολουθήσει να δουλεύει* αλλά να παράγει *αποτελέσματα διαφορετικά από τα αναμενόμενα* *(πχ αν παραλείψουμε το βήμα 3 θα έχουμε φτιάξει κρύο – και όχι ζεστό τσάι)*

Παραδείγματα

Είναι αλγόριθμος (ναι ή όχι;)

- Οι 10 εντολές (*δεν είναι αλγόριθμος - δεν είναι καν διαδικασία - τι είναι;)*
- Μια συνταγή μαγειρικής *(εξαρτάται - από τι;)*
- Οι οδηγίες για την ύφανση ενός χαλιού με σχέδια
- Οι οδηγίες συναρμολόγησης ενός επίπλου
- Ο πολλαπλασιασμός δύο δεκαδικών αριθμών
- Η ταξινόμηση μιας λίστας αριθμών κατ' αύξουσα ή φθίνουσα τάξη *(εξαρτάται - από τι;)*
- Η εύρεση *όλων* των όρων μιας γεωμετρικής προόδου
- Ο υπολογισμός του πρώτου ενός δισεκατομμυρίου ψηφίων του αριθμού π *(3.14159…)*

Μελέτη Αλγορίθμων

Αποτελεί βασικό μέρος της επιστήμης των υπολογιστών

Δοθέντος ενός προβλήματος

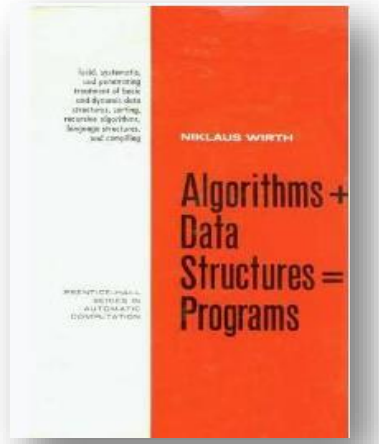
- Υπάρχει αλγόριθμος για την επίλυσή του;
- Αν κάποιος προτείνει έναν αλγόριθμο για την επίλυση
  - Είμαστε βέβαιοι ότι ο αλγόριθμος λειτουργεί για *όλα τα πιθανά δεδομένα*;
  - *Πόσο διαρκεί* ο αλγόριθμος για να τρέξει; Πόση *μνήμη* χρειάζεται;
  - Είναι αυτός ο αλγόριθμος *ο καλύτερος δυνατός*; Μπορεί το πρόβλημα να επιλυθεί *πιο γρήγορα*;

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Algorithms + Data Structures = Programs

Niklaus Wirth
Πατέρας της πρώτης δομημένης γλώσσας προγραμματισμού ***Pascal***

Οι *αλγόριθμοι*
συνυφασμένοι με τις απαραίτητες *δομές*
σε μία *αδιάσπαστη ενότητα*
αποτελούν τη βάση κάθε προγράμματος



Εκδ.1976

Τα σημαντικότερα βήματα στην διαδικασία του προγραμματισμού

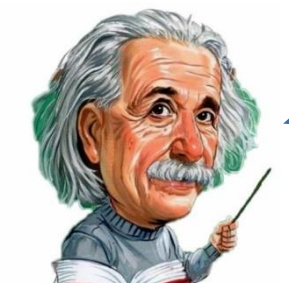
- Διατύπωση του Προβλήματος
- Γενική περιγραφή της λύσης
- Επιλογή του κατάλληλου αλγορίθμου
- Γράψιμο του κώδικα
- Διόρθωση συντακτικών λαθών (debugging)
- Έλεγχος ορθότητας
- Τεκμηρίωση
- Συντήρηση / Αναβάθμιση

Περισσότερο απαραίτητα σε μεγάλα εμπορικά ή/και επιστημονικά προγράμματα (συστήματα)

Διατύπωση του Προβλήματος

Το πρώτο πράγμα που πρέπει να γίνει είναι να ορίσουμε σωστά και με σαφήνεια το πρόβλημα που θέλουμε να λύσουμε.

Παρόλο που κάτι τέτοιο φαίνεται αρκετά προφανές για απλά και μικρά προβλήματα, το βήμα αυτό είναι απαραίτητο όταν έχουμε να κάνουμε με μεγαλύτερα και πιο σύνθετα προβλήματα



"Αν είχα στη διάθεσή μου μια ώρα για να σώσω τον πλανήτη, θα αφιέρωνα τα 59 λεπτά για να ορίσω το πρόβλημα και ένα λεπτό για να το λύσω"

Ένα καλά διατυπωμένο πρόβλημα συχνά περιέχει την ίδια τη λύση μέσα του, και αυτή η λύση είναι συνήθως αρκετά προφανής και απλή.

Γενική περιγραφή της λύσης

- Απαραίτητη επειδή ως γνωστόν ο υπολογιστής δεν παίρνει καμία πρωτοβουλία από μόνος του και επομένως περιμένει από εμάς να του παρέχουμε τις ακριβείς οδηγίες για τη λύση του συγκεκριμένου προβλήματος

- Γίνεται σε φυσική γλώσσα

- Χρησιμοποιείται ιδιαίτερα αυστηρή διατύπωση ώστε να εξασφαλίζεται η μέγιστη δυνατή σαφήνεια

- Τυχόν εναλλακτικές μέθοδοι επίλυσης, διατυπώνονται η καθεμιά τους ξεχωριστά

Επιλογή του κατάλληλου αλγορίθμου (1/2)

(και διατύπωσή του με κατάλληλο τρόπο)

- Διαφορετικές διατυπώσεις της λύσης → διαφορετικοί αλγόριθμοι
- Επιλογή κατάλληλου ≠ επιλογή βέλτιστου (εν γένει)
- Εξαρτάται από την προσδοκώμενη χρήση και τους διαθέσιμους πόρους

Παράδειγμα: Επιλογή καταλληλότερου αλγόριθμου ταξινόμησης

Κριτήρια	Καταλληλότερος Αλγόριθμος Ταξινόμησης
Μόνο λίγα στοιχεία	Insertion Sort
Συνήθως τα περισσότερα στοιχεία είναι ήδη ταξινομημένα	Insertion Sort
Ανησυχία για τα χειρότερο σενάριο	Heap Sort
Ενδιαφέρον για ένα καλό μέσο αποτέλεσμα	Quicksort
Επιθυμία για όσο το δυνατόν μικρότερο κώδικα	Insertion Sort

Επιλογή του κατάλληλου αλγορίθμου (2/2)

Εναλλακτικές μορφές διατύπωσης αλγορίθμων

Φυσική γλώσσα

Ο αλγόριθμος εκφράζεται σε απλή καθομιλούμενη γλώσσα, στην οποία οι προτάσεις έχουν χωριστεί σε παραγράφους-βήματα και έχουν αριθμηθεί.

Τύπος

Ο αλγόριθμος εκφράζεται με τη βοήθεια ενός (ή συνδυασμού περισσότερων) μαθηματικών τύπων.

Ψευδοκώδικας

Ο αλγόριθμος εκφράζεται σε μια υποθετική γλώσσα με στοιχεία από κάποιες γλώσσες προγραμματισμού, παραλείποντας λεπτομέρειες που δεν είναι ουσιαστικές για την ανθρώπινη κατανόηση του αλγορίθμου.

Διαγράμματα ροής

Ο αλγόριθμος εκφράζεται γραφικά με μια σειρά σχημάτων που συνδέονται μεταξύ τους με συγκεκριμένο τρόπο. Τα διαφορετικά περιγράμματα των σχημάτων υποδηλώνουν διαφορετικά είδη εντολών / πράξεων , ενώ ο τρόπος διασύνδεσης υποδηλώνει την σειρά εκτέλεσης των συγκεκριμένων εντολών / πράξεων.

Γράψιμο του κώδικα

Από το βήμα αυτό και μετά, η χρήση του Η/Υ καθίσταται απαραίτητη

Εξαρτάται κυρίως από παράγοντες όπως

- το *είδος* του προβλήματος (μαθηματικό, λογικό, επεξεργ. πληροφορίας)
- τις *διαθέσιμες γλώσσες* προγραμματισμού
- τους *διαθέσιμους πόρους* του Η/Υ στον οποίο πρόκειται να εκτελεστεί (μνήμη, ταχύτητα κλπ.)
- την *εμπειρία* του προγραμματιστή

Διόρθωση συντακτικών λαθών (Debugging)

Συντακτικά Λάθη:

- Προκύπτουν κατά *διατύπωση* ενός προγράμματος σε μια *γλώσσα προγραμματισμού* υψηλού επιπέδου
- Αιτία τους η εγγενής *αδυναμία των ανθρώπων* να προσαρμοστούν στους *πολύ αυστηρούς συντακτικούς κανόνες* μιας *γλώσσας προγραμματισμού* υψηλού επιπέδου
- *Εντοπίζονται αυτόματα* από το πρόγραμμα μεταγλωττισμού (compiler) και *επισημαίνονται* στον προγραμματιστή
- Ένα πρόγραμμα *δεν μετατρέπεται σε εκτελέσιμο κώδικα* αν δεν έχουν *διορθωθεί* προηγουμένως *όλα τα συντακτικά του λάθη* (*)

(*) σε ειδικές περιπτώσεις, ένα πρόγραμμα με συντακτικά λάθη μπορεί να ξεκινήσει να εκτελείται μέχρι να "χτυπήσει" στο πρώτο συντακτικό λάθος που θα συναντήσει (βλ. interpreter παρακάτω)

Έλεγχος ορθότητας

Το πρόγραμμα ελέγχεται για την *ορθότητα της λειτουργίας του* δίνοντάς του να τρέξει προβλήματα με *αποτελέσματα* που είναι *από πριν γνωστά*

Με τον τρόπο αυτό εντοπίζονται και διορθώνονται **λογικά λάθη** που πιθανόν να υπάρχουν στο πρόγραμμα και πιθανόν να οφείλονται:

- είτε σε *εγγενή σφάλματα του ίδιου του αλγορίθμου* *(garbage-in / garbage-out)*
- είτε σε λάθη που προέκυψαν κατά την *διατύπωση* του αλγορίθμου ή/και την *υλοποίησή* του σε πρόγραμμα

- Ο *εντοπισμός* και η διόρθωση των λογικών λαθών είναι μια *επίπονη διαδικασία* (ιδιαίτερα στα μεγαλύτερα / πιο σύνθετα προγράμματα)

- Θεωρητικά δεν μπορεί *ποτέ* να δοθεί *εγγύηση* για την *απόλυτη ορθότητα* ενός προγράμματος

Τεκμηρίωση

- Τεκμηρίωση των ***λειτουργικών δυνατοτήτων*** του προγράμματος

  Εγχειρίδια λειτουργίας του συστήματος (manuals) σε *διάφορα επίπεδα*

  - *Τελικού* χρήστη (end user)
  - Χρήστη *αυξημένων προνομίων* (super user)
  - *Διαχειριστή* (administrator)
  - …

  Υποσύστημα παροχής *άμεσης βοήθειας* (online help)

  Υποσύστημα παροχής *στοχευμένης βοήθειας* (context sensitive help)

- Τεκμηρίωση του ***πηγαίου κώδικα*** του προγράμματος

  - Λειτουργική περιγραφή των *επί μέρους τμημάτων* του προγράμματος (υπορουτίνες, συναρτήσεις, βιβλιοθήκες κλπ) τόσο με *σχόλια* πάνω στον ίδιο τον κώδικα όσο και με *ειδικά εγχειρίδια* (manuals)

Συντήρηση / Αναβάθμιση

Λάθη / αβλεψίες που εντοπίζονται κατά την χρήση ενός προγράμματος καθώς και *βελτιώσεις / νέες δυνατότητες* που επιλέγεται να υλοποιηθούν οδηγούν σε συχνές *αναβαθμίσεις* του από νεότερες εκδόσεις

- Ο αριθμός έκδοσης ενός προγράμματος συνήθως διαιρείται σε *σύνολα αριθμών*, χωρισμένα με δεκαδικά σημεία.

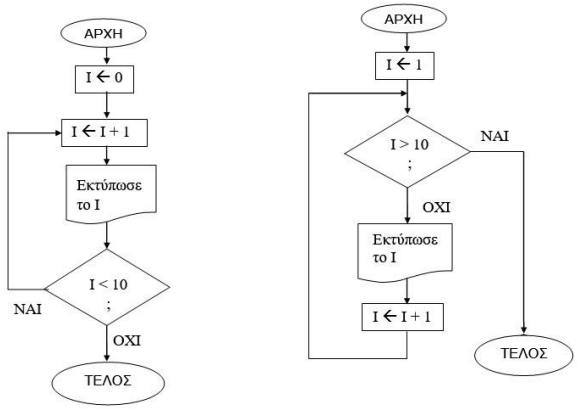
π.χ. ***Chrome.3.9.2.45***

- Μεταβολή του *πρώτου* (αριστερού) αριθμού υποδεικνύει συνήθως μια *σημαντική αλλαγή* του προγράμματος (μεταξύ άλλων και την αναβάθμιση της διεπαφής)
- Μεταβολή του *τελευταίου* αριθμού υποδεικνύει συνήθως μια *μικρή αλλαγή* που παραμένει απαρατήρητη σε κανονικές συνθήκες λειτουργίας του προγράμματος
- Οι μεταβολές των άλλων *ενδιάμεσων* αριθμών αντιπροσωπεύουν συνήθως διάφορες *αλλαγές ενδιάμεσης σημασίας*

Διαγράμματα Ροής

Αποτελούνται από μια σειρά **σχημάτων** που **συνδέονται** μεταξύ τους με συγκεκριμένο τρόπο.

- Τα *διαφορετικά περιγράμματα* των σχημάτων υποδηλώνουν *διαφορετικά είδη εντολών / πράξεων*
- ο *τρόπος διασύνδεσης* υποδηλώνει την *σειρά εκτέλεσης* των συγκεκριμένων εντολών / πράξεων.



Ένα διάγραμμα ροής μας δίνει μια **συνολική εποπτεία** του αλγορίθμου γιαυτό και δεν θα πρέπει να είναι αρκετά περίπλοκο ή μεγάλο σε έκταση

Γενικά δεν θα πρέπει να εκτείνεται σε περισσότερες από μια σελίδες, έτσι ώστε να μη χάνεται η συνολική εποπτεία.

Διαγράμματα Ροής

Βασικά Σχήματα (1/2)

Αρχή / Τέλος αλγορίθμου: Το σχήμα αυτό περιέχει είτε την λέξη ΑΡΧΗ είτε την λέξη ΤΕΛΟΣ και χρησιμοποιείται για να σηματοδοτήσει αντίστοιχα τα σημεία έναρξης και λήξης του διαγράμματος ροής

Απλές πράξεις: Μέσα στο παραλληλόγραμμο αυτό αναγράφονται μια ή περισσότερες απλές πράξεις που εκτελούνται διαδοχικά

Προκαθορισμένη διαδικασία: Μέσα στο παραλληλόγραμμο αυτό αναγράφεται το όνομα μιας διαδικασίας (συνήθως αρκετά σύνθετης) η οποία περιγράφεται αναλυτικά με δικό της ξεχωριστό διάγραμμα ροής

Έλεγχος / Απόφαση: Μέσα στον ρόμβο αναγράφεται μια συνθήκη (π.χ. A>B) η οποία μπορεί να πληρείται ή όχι ανάλογα με τις τιμές των δεδομένων. Η πορεία που θα ακολουθηθεί αποφασίζεται ανάλογα με την έκβαση του ελέγχου.

Διαγράμματα Ροής

Βασικά Σχήματα (2/2)

Είσοδος (ανάγνωση) δεδομένων: Το σχήμα αυτό χρησιμοποιείται για να περιγράψει την είσοδο δεδομένων στον αλγόριθμο από μια εξωτερική πηγή δεδομένων του ηλεκτρονικού υπολογιστή όπως το πληκτρολόγιο

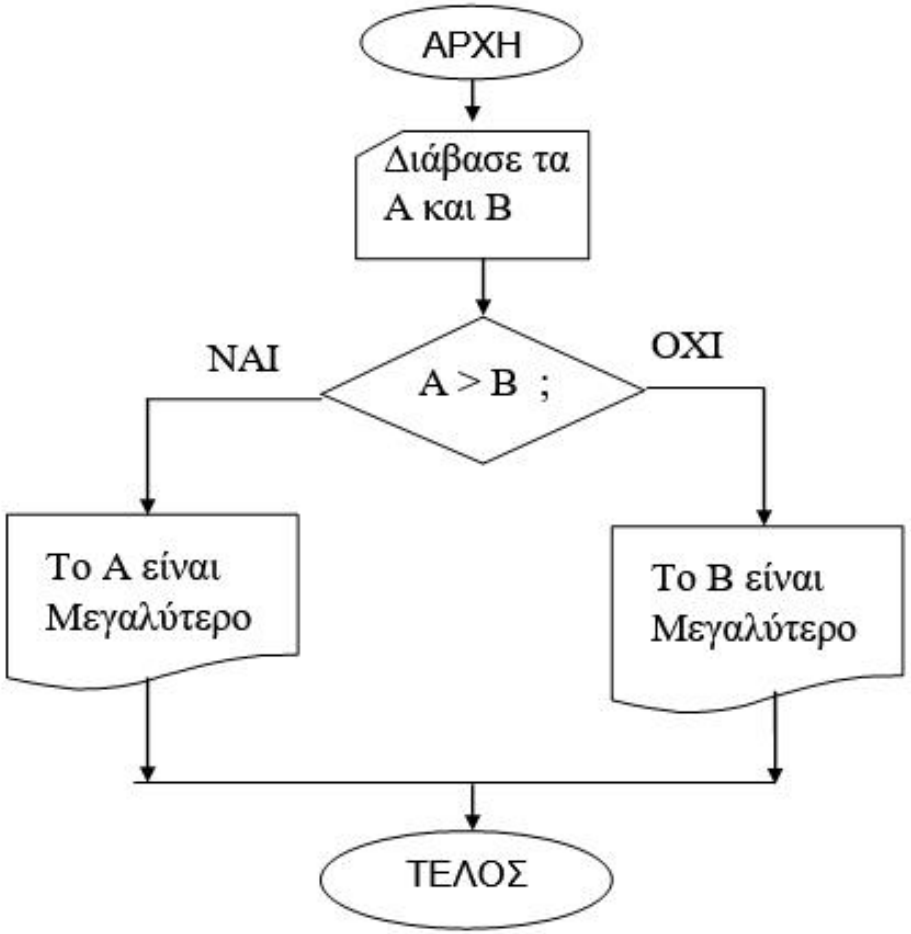
Έξοδος (εμφάνιση) αποτελεσμάτων: Το σχήμα αυτό χρησιμοποιείται για να περιγράψει την έξοδο αποτελεσμάτων από τον αλγόριθμο σε μια συσκευή εξόδου του ηλεκτρονικού υπολογιστή όπως η οθόνη ή ο εκτυπωτής

Επικοινωνία με αρχείο: Το σχήμα αυτό χρησιμοποιείται για να περιγράψει την ανάγνωση ή την γραφή δεδομένων από/προς ένα αρχείο.

Σύνδεση με άλλο σημείο του διαγράμματος (στην ίδια ή σε άλλη σελίδα)

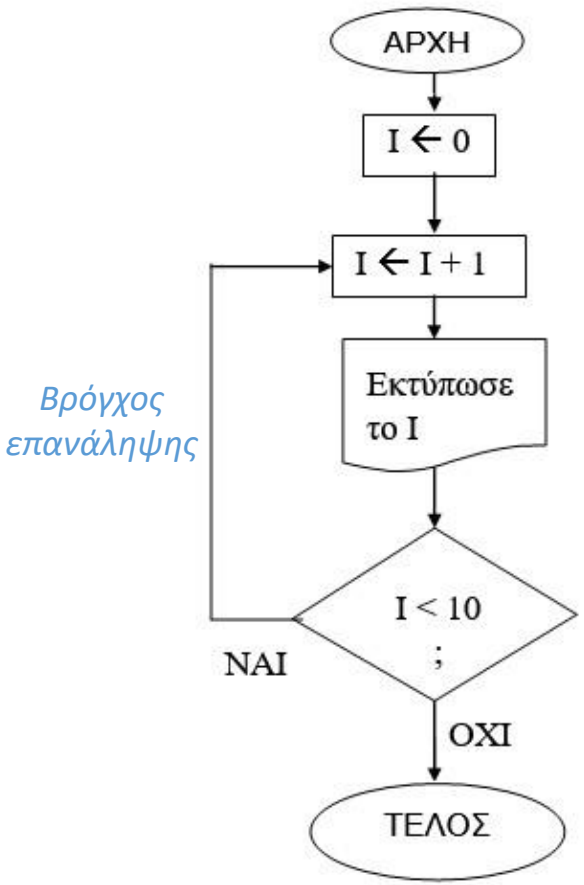
Διαγράμματα Ροής - Παραδείγματα

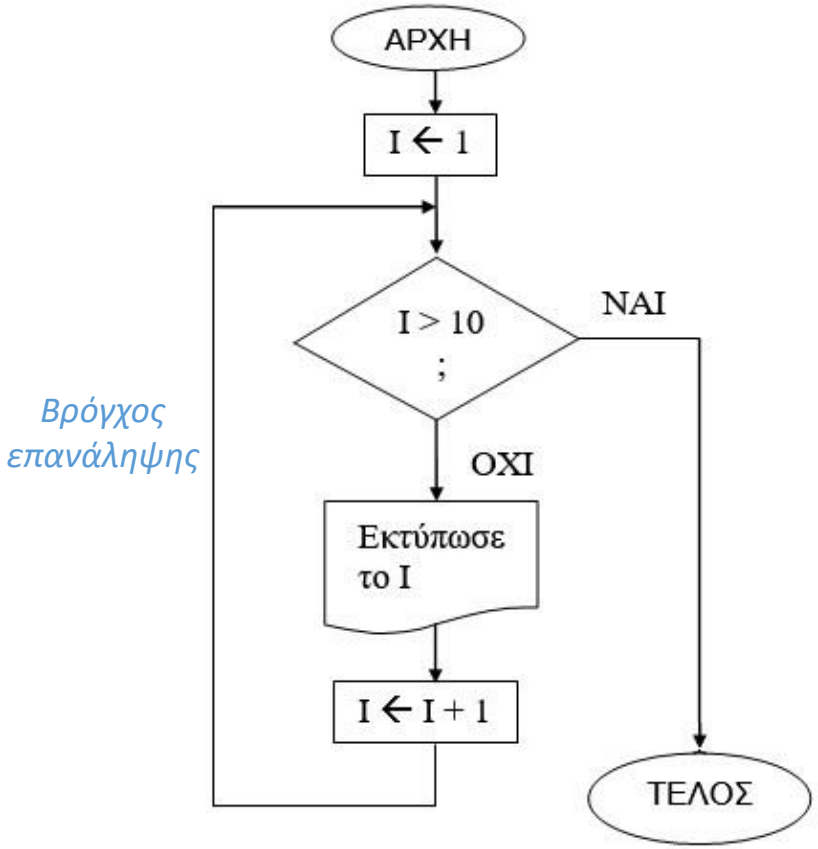
Παράδειγμα 1: Εύρεση του μεγαλύτερου από δυο αριθμούς



Διαγράμματα Ροής - Παραδείγματα

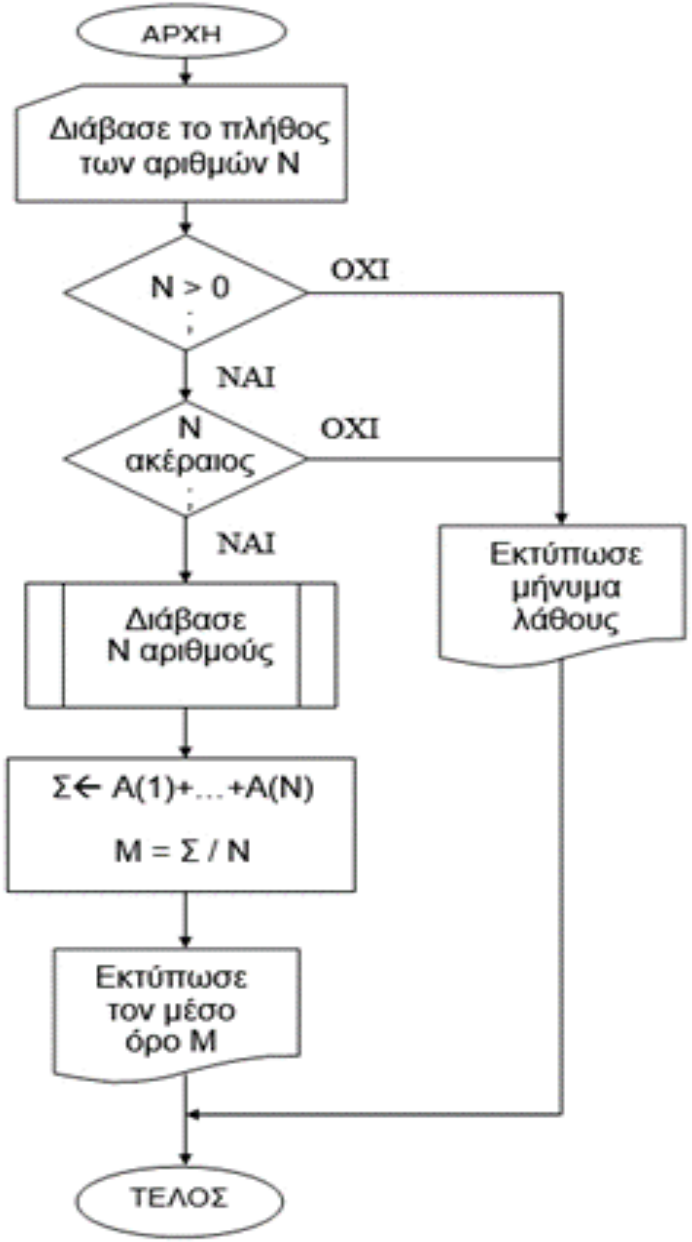
Παράδειγμα 2: Εκτύπωση των αριθμών από το 1 ως το 10. (Δυο διαφορετικές εκδοχές)



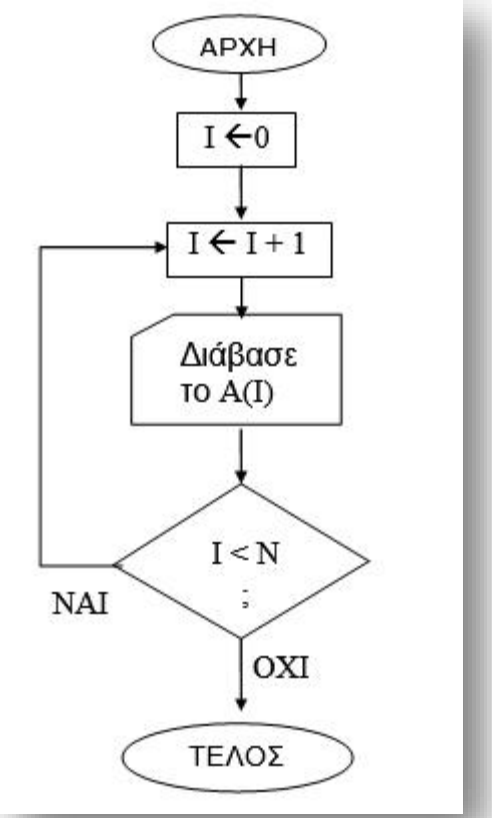


Διαγράμματα Ροής - Παραδείγματα

Παράδειγμα 3: Υπολογισμός του μέσου όρου ***γνωστού*** πλήθους αριθμών.

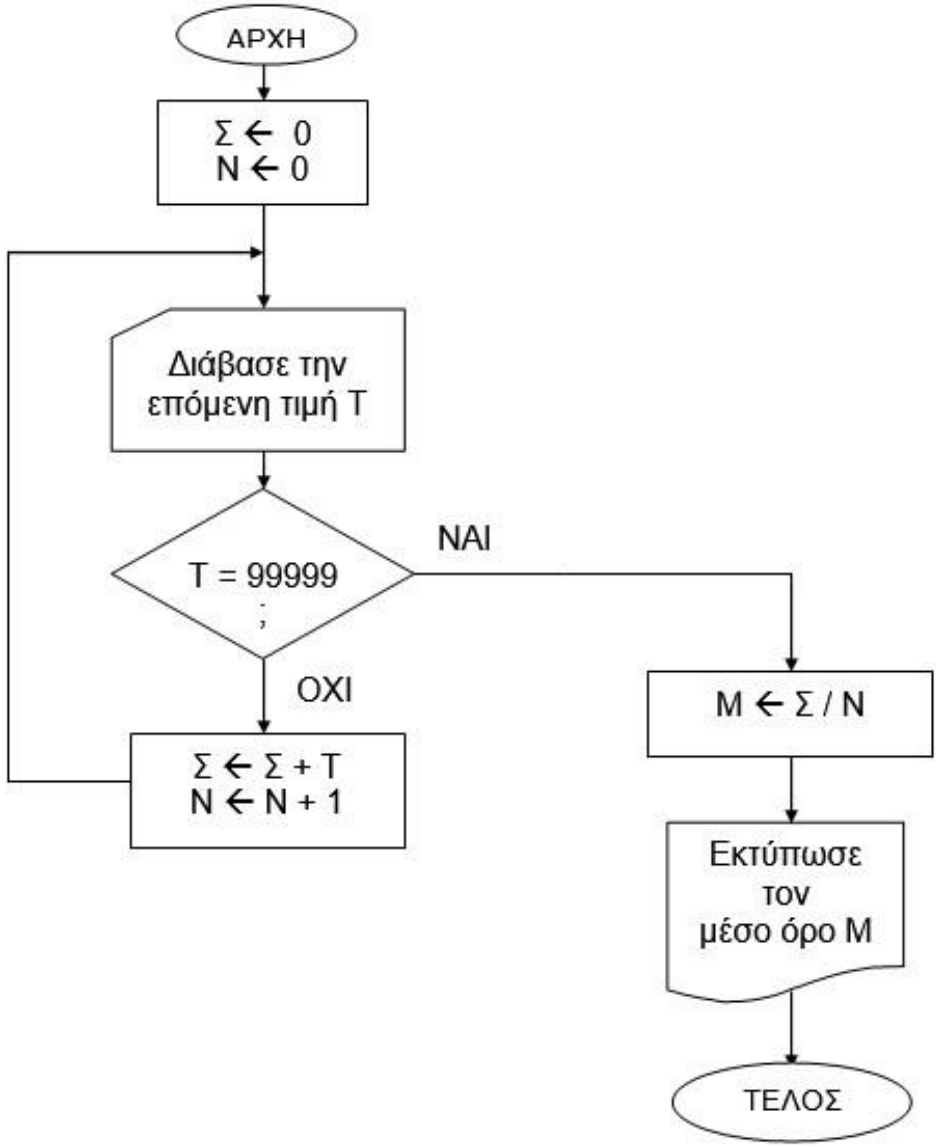


Προκαθορισμένη διαδικασία "Διάβασε N αριθμούς"



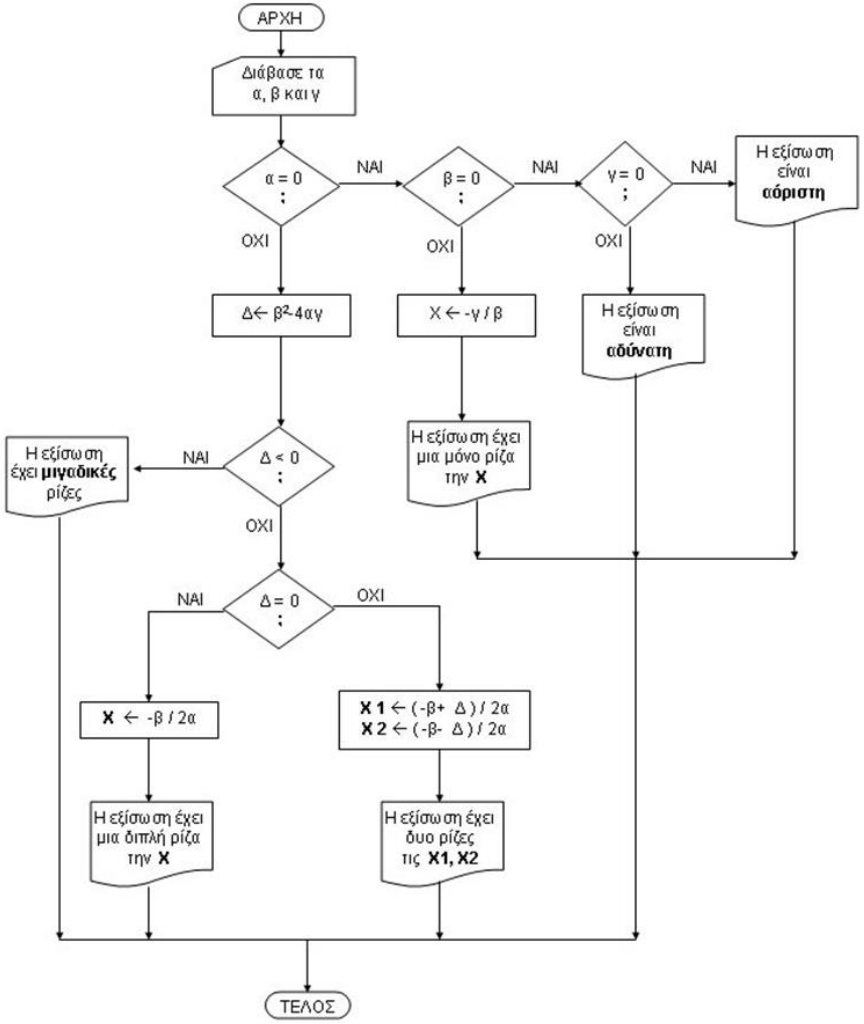
Διαγράμματα Ροής - Παραδείγματα

Παράδειγμα 4: Υπολογισμός του μέσου όρου ***άγνωστού*** πλήθους αριθμών



Διαγράμματα Ροής - Παραδείγματα

Παράδειγμα 5: Επίλυση της τριωνυμικής εξίσωσης $\alpha x^2+\beta x+\gamma=0$



Ανάπτυξη Προγράμματος

Γλώσσα Μηχανής

- Κάθε τύπος Κεντρικής Μονάδας Επεξεργασίας έχει το δικό της *ρεπερτόριο εντολών* (instruction set) τις οποίες αναγνωρίζει και εκτελεί.
- Οι εντολές διατυπώνονται σε *δυαδική* (πχ 00101011) ή *δεκαεξαδική* μορφή (πχ F7)
- Το *σύνολο των εντολών* αυτών μαζί με τους *πολύ αυστηρούς κανόνες σύνταξής* τους αποτελούν την λεγόμενη γλώσσα μηχανής (machine language) του συγκεκριμένου ηλεκτρονικού υπολογιστή.

Δυαδική μορφή διατύπωσης	***Δεκαεξαδική μορφή διατύπωσης***
10001001 11011001	89 D9
10111000 00000001 00000000	B8 01 00
10000011 11111001 00000000	83 F9 00
01110100 00000101	74 05
11110111 11100001	F7 E1
01001001	49
11101011 11110110	EB F6

1000	1001	1101	1001
8_{16}	9_{16}	D_{16}	9_{16}

Ανάπτυξη Προγράμματος

Γλώσσα Assembly

Αποτελεί το επόμενο βήμα εξέλιξης μετά την γλώσσα μηχανής

- *Μνημονικό* σύστημα για την φιλικότερη αναπαράσταση προγραμμάτων σε γλώσσα μηχανής
- Εγγενής εξάρτηση από την αρχιτεκτονική της εκάστοτε ΚΜΕ
  - *Ακριβής αντιστοιχία* μεταξύ των *εντολών της μηχανής* και των *εντολών assembly*
  - Ο προγραμματιστής πρέπει να *σκέφτεται* σε *γλώσσα μηχανής*
- Χρήση *μνημονικών ονομάτων* για την αναπαράσταση όλων των *εντολών μηχανής* και όλων των *καταχωρητών*
- Χρήση *περιγραφικών ονομάτων* για τις *θέσεις μνήμης* που χρησιμοποιούνται από το πρόγραμμα
  - *Επιλέγονται από τον προγραμματιστή (ονόματα μεταβλητών)*
- Η *μετάφραση σε γλώσσα μηχανής* γίνεται από ένα *πρόγραμμα* που ονομάζεται **assembler**

Παράδειγμα προγράμματος σε γλώσσα assembly

Πρόγραμμα που αθροίζει δυο μεταβλητές και καταχωρεί το αποτέλεσμα σε μια τρίτη

Γλώσσα μηχανής	Γλώσσα assembly	Ερμηνεία
89 05 D9	LD R5, Price	Φόρτωσε στον καταχωρητή *R5* το περιεχόμενο της θέσης μνήμης *Price*
B8 06 C3	LD R6, Tax	Φόρτωσε στον καταχωρητή *R6* το περιεχόμενο της θέσης μνήμης *Tax*
83 00 05 06	ADD R0, R5, R6	Πρόσθεσε τα περιεχόμενα των *R5* και R*6* και βάλε το άθροισμα στον *R0*
74 00 7F	ST R0, TotalCost	Αποθήκευσε το περιεχόμενο του καταχωρητή *R0* στη θέσης μνήμης *TotalCost*
49	HLT	Σταμάτα την εκτέλεση του προγράμματος

Ο ίδιος κώδικας σε μια γλώσσα προγραμματισμού υψηλού επιπέδου θα διατυπωνόταν ως μια και μόνο εντολή:

TotalCost = Price + Tax

Ρεπερτόριο εντολών του επεξεργαστή 8086

OPCODE		DESCRIPTION
AAA		ASCII adjust addition
AAD		ASCII adjust division
AAM		ASCII adjust multiply
AAS		ASCII adjust subtraction
ADC	dt,sc	Add with carry
ADD	dt,sc	Add
AND	dt,sc	Logical AND
CALL	proc	Call a procedure
CBW		Convert byte to word
CLC		Clear carry flag
CDL		Clear direction flag
CLI		Clear interrupt flag
CMC		Complement carry flag
CMP	dt,sc	Compare
CMPS	[dt,sc]	Compare string
CMPSB	"	" bytes
CMPSW	"	" words
CWD		Convert word to double word
DAA		Decimal adjust addition
DAS		Decimal adjust subtraction
DEC	dt	Decrement
DIV	sc	Unsigned divide
ESC	code,sc	Escape
HLT		Halt
IDIV	sc	Integer divide
IMUL	sc	Integer multiply
IN	ac,port	Input from port
INC	dt	Increment
INT	type	Interrupt
INTO		Interrupt if overflow
IRET		Return from interrupt
JA	slabel	Jump if above
JAE	slabel	Jump if above or equal
JB	slabel	Jump if below
JBE	slabel	Jump if below or equal
JC	slabel	Jump if carry
JCXZ	slabel	Jump if CX is zero
JE	slabel	Jump if equal
JG	slabel	Jump if greater
JGE	slabel	Jump if greater or equal
JL	slabel	Jump if less
JLE	slabel	Jump if less or equal
JMP	label	Jump
JNA	slabel	Jump if not above
JNAE	slabel	Jump if not above or equal
JNB	slabel	Jump if not below
JNBE	slabel	Jump if below or equal
JNC	slabel	Jump if no carry
JNE	slabel	Jump if not equal
JNG	slabel	Jump if not greater
JNGE	slabel	Jump if not greater or equal
JNL	slabel	Jump if not less
JNLE	slabel	Jump if not less or equal
JNZ	slabel	Jump if not zero
JNO	slabel	Jump if not overflow
JNP	slabel	Jump if not parity
JNS	slabel	Jump if not sign
JO	slabel	Jump if overflow
JPO	slabel	Jump if parity odd
JP	slabel	Jump if parity
JPE	slabel	Jump if parity even
JS	slabel	Jump if sign
JZ	slabel	Jump if zero
LAHF		Load AH from flags
LDS	dt,sc	Load pointer using DS
LEA	dt,sc	Load effective address
LES	dt,sc	Load pointer using ES
LOCK		Lock bus
LODS	[sc]	Load string
LODSB	"	" bytes
LODSW	"	" words
LOOP	slabel	Loop
LOOPE	slabel	Loop if equal
LOOPZ	slabel	Loop if zero
LOOPNE	slabel	Loop if not equal
LOOPNZ	slabel	Loop if not zero
MOV	dt,sc	Move
MOVS	[dt,sc]	Move string
MOVSB	"	" bytes
MOVSW	"	" words
MUL	sc	Unsigned multiply
NEG	dt	Negate
NOP		No operation
NOT	dt	Logical NOT
OR	dt,sc	Logical OR
OUT	port,ac	output to port
POP	dt	Pop word off stack
POPF		Pop flags off stack
PUSH	sc	Push word onto stack
PUSHF		Push flags onto stack
RCL	dt,cnt	Rotate left through carry
RCR	dt,cnt	Rotate right through carry
REP		Repeat string operation
REPE		Repeat while equal
REPZ		Repeat while zero
REPNE		Repeat while not equal
REPNZ		Repeat while not zero
RET	[pop]	Return from procedure
ROL	dt,cnt	Rotate left
ROR	dt,cnt	Rotate right
SAHF		Store AH into flags
SAL	dt,cnt	Shift arithmetic left
SHL	dt,cnt	Shift logical left
SAR	dt,cnt	Shift arithmetic right
SBB	dt,sc	Subtract with borrow
SCAS	[dt]	Scan string
SCASB	"	" byte
SCASW	"	" word
SHR	dt,cnt	Shift logical right
STC		Set carry flag
STD		Set direction flag
STI		Set interrupt flag
STOS	[dt]	Store string
STOSB	"	" byte
STOSW	"	" word
SUB	dt,sc	Subtraction
TEST	dt,sc	Test (logical AND)
WAIT		Wait for 8087
XCHG	dt,sc	Exchange
XLAT	table	Translate
XLATB	"	"
XOR	dt,sc	Logical exclusive OR

Notes:
dt - destination
sc - source
label - may be near or far address
slabel - near address

Τμήμα πραγματικού κώδικα σε γλώσσα assembly

Διευθύνσεις μνήμης

```
08048918   pushl  %ebp
08048919   movl   %esp,%ebp
0804891b   subl   $0x4,%esp
0804891e   movl   $0x0,0xfffffffc(%ebp)
08048925   cmpl   $0x63,0xfffffffc(%ebp)
08048929   jle    08048930
0804892b   jmp    08048948
0804892d   nop
0804892e   nop
0804892f   nop
08048930   movl   0xfffffffc(%ebp),%eax
08048933   pushl  %eax
08048934   pushl  $0x8049418
08048939   call   080487c0 <printf>
0804893e   addl   $0x8,%esp
08048941   incl   0xfffffffc(%ebp)
08048944   jmp    08048925
08048946   nop
08048947   nop
08048948   xorl   %eax,%eax
0804894a   jmp    0804894c
0804894c   leave
0804894d   ret
```

Κώδικας assembly

Γλώσσες προγραμματισμού υψηλού επιπέδου

- *Τεχνητές* γλώσσες που μπορούν να χρησιμοποιηθούν για τον έλεγχο μιας μηχανής, συνήθως ενός υπολογιστή
- Χρησιμοποιούνται για να διευκολύνουν την οργάνωση και διαχείριση πληροφοριών αλλά και την *ακριβή διατύπωση αλγορίθμων*
- Ακολουθούν τις *βασικές έννοιες και αρχές της γλωσσολογίας* και προσεγγίζουν πολύ περισσότερο τον ανθρώπινο τρόπο σκέψης και διατύπωσης
- Χαρακτηρίζονται από ένα σύνολο *συντακτικών* και *εννοιολογικών* κανόνων, που ορίζουν αντίστοιχα τη *δομή* και το *νόημα* των *προτάσεων* της γλώσσας

Γλώσσες προγραμματισμού υψηλού επιπέδου

Μια γλώσσα προγραμματισμού υψηλού επιπέδου προσδιορίζεται από 4 στοιχεία:

Αλφάβητο - Λεξιλόγιο - Συντακτικό - Σημασιολογία

Αλφάβητο. Το σύνολο των *αποδεκτών συμβόλων* - δηλαδή εκείνων που *αναγνωρίζει* και *κατανοεί* η συγκεκριμένη γλώσσα.

A…Z a…z + - * / = < > ! @ # $ % ^ & ?

Λεξιλόγιο Το σύνολο των *αποδεκτών συνδυασμών των συμβόλων* της γλώσσας δηλαδή των *λέξεων* που αναγνωρίζει και κατανοεί η συγκεκριμένη γλώσσα.

Οι περισσότερες από αυτές τις λέξεις είναι συνήθως Αγγλικά ρήματα στην προστακτική τους (εντολές της γλώσσας).

read, print, if, else, for, pause…

Γλώσσες προγραμματισμού υψηλού επιπέδου

Συντακτικό (syntax) Το σύνολο των *κανόνων* που διέπουν την *σύνταξη* κειμένων (προγραμμάτων) στην συγκεκριμένη γλώσσα (συντακτικοί κανόνες).

Αφορούν την ορθή σύνταξη των φράσεων και την φυσική διάταξη των συμβόλων της γλώσσας.

var *<όνομα μεταβλητής>* **as** *<τύπος δεδομένων>* **;**

if *<συνθήκη>* **then** *<έκφραση 1>* **else** *<έκφραση 2>*

Σημασιολογία (semantics) Προσδιορίζει την *έννοια των εκφράσεων* δηλαδή το *τι ακριβώς σημαίνει η κάθε έκφραση* όταν το πρόγραμμα εκτελεστεί σε κάποιον υπολογιστή.

Η έκφραση **var** *Test* **as integer** ; σημαίνει ότι κατά την εκτέλεση του προγράμματος θα πρέπει να διατεθεί ένα τμήμα της κεντρικής μνήμης του υπολογιστή για την αποθήκευση μιας ακέραιας μεταβλητής που θα ονομάζεται *Test*.

Υπολογισμός του N! σε διάφορες γλώσσες προγραμματισμού (1/2)

FORTRAN

```
C Υπολογισμός  του N! σε FORTRAN
C
      FUNCTION IFACTORIAL(N)
      INTEGER N
      INTEGER IFACTORIAL
      INTEGER IRESULT, I

      IRESULT = 1
      DO I = 1, N
        IRESULT = IRESULT * I
      END DO
      IFACTORIAL = IRESULT

      END
```

C / C++

```
/* Υπολογισμός  του N! σε  C*/

Int factorial (int n)
{
        int result;
        int count;

        count = n;
        result = 1;
        while (count > 0) {
                result = result *
count;
                count = count -
1;
        }
        return (result);
}
```

Pascal

```
 (* Υπολογισμός  του N! σε
Pascal*)

function factorial(N : Integer) :
Integer;

var
  Count, Result : Integer;

begin
     Count := N;
     Result := 1;
     While Count > 0 Do
     begin
        Result := Result * Count;
        Count := Count - 1
     end;
     Factorial := Result
end (* Factorial *);
```

Υπολογισμός του N! σε διάφορες γλώσσες προγραμματισμού (2/2)

Prolog

```
% Υπολογισμός του N! σε Prolog

factorial(0, 1).

factorial(N, N_Factorial) :-
        N > 0,
        M is N - 1,
        factorial(M,
M_Factorial),
        N_Factorial is
M_Factorial * N.
```

Java

```
// Υπολογισμός  του N! σε Java

public int factorial (int N) {
        int result;
        int counter;

       counter= N;
       result= 1;
       while (counter> 0) {
         result= result*
                      counter;
         counter= counter- 1;
       }
        return (result);
```

Python

```
# Υπολογισμός  του N! σε Python

def factorial(n):

res=1
for i in range(2,n+1):
     res=res*i
return res
```

Δείτε τον κώδικα για την συνάρτηση N παραγοντικό σε περισσότερες από 260 διαφορετικές γλώσσες προγραμματισμού στο https://www.rosettacode.org/wiki/Factorial

Υπολογισμός του N! σε Assembly

```
FACTO    CSECT
         USING  FACTO,R13
SAVEAREA B      STM-SAVEAREA(R15)
         DC     17F'0'
         DC     CL8'FACTO'
STM      STM    R14,R12,12(R13)
         ST     R13,4(R15)
         ST     R15,8(R13)
         LR     R13,R15          base register and savearea pointer
         ZAP    N,=P'1'          n=1
LOOPN    CP     N,NN             if n>nn
         BH     ENDLOOPN         then goto endloop
         LA     R1,PARMLIST
         L      R15,=A(FACT)
         BALR   R14,R15          call fact(n)
         ZAP    F,0(L'R,R1)      f=fact(n)
DUMP     EQU    *
         MVC    S,MASK
         ED     S,N
         MVC    WTOBUF+5(2),S+30
         MVC    S,MASK
         ED     S,F
         MVC    WTOBUF+9(32),S
         WTO    MF=(E,WTOMSG)
         AP     N,=P'1'          n=n+1
         B      LOOPN
ENDLOOPN EQU    *
RETURN   EQU    *
         L      R13,4(0,R13)
         LM     R14,R12,12(R13)
         XR     R15,R15
         BR     R14
```

```
FACT     EQU    *                function FACT(l)
         L      R2,0(R1)
         L      R3,12(R2)
         ZAP    L,0(L'N,R2)      l=n
         ZAP    R,=P'1'          r=1
         ZAP    I,=P'2'          i=2
LOOP     CP     I,L              if i>l
         BH     ENDLOOP          then goto endloop
         MP     R,I              r=r*i
         AP     I,=P'1'          i=i+1
         B      LOOP
ENDLOOP  EQU    *
         LA     R1,R             return r
         BR     R14              end function FACT
         DS     0D
NN       DC     PL16'29'
N        DS     PL16
F        DS     PL16
C        DS     CL16
II       DS     PL16
PARMLIST DC     A(N)
S        DS     CL33
MASK     DC     X'40',29X'20',X'212060'  CL33
WTOMSG   DS     0F
         DC     H'80',XL2'0000'
WTOBUF   DC     CL80'FACT(..)=.............................. '
L        DS     PL16
R        DS     PL16
I        DS     PL16
         LTORG
         YREGS
         END    FACTO
```

Η έννοια του Compiler (Μεταγλωττιστή)

Compiler ***συγκεκριμένης*** γλώσσας προγραμματισμού για ***συγκεκριμένο*** τύπο Η/Υ

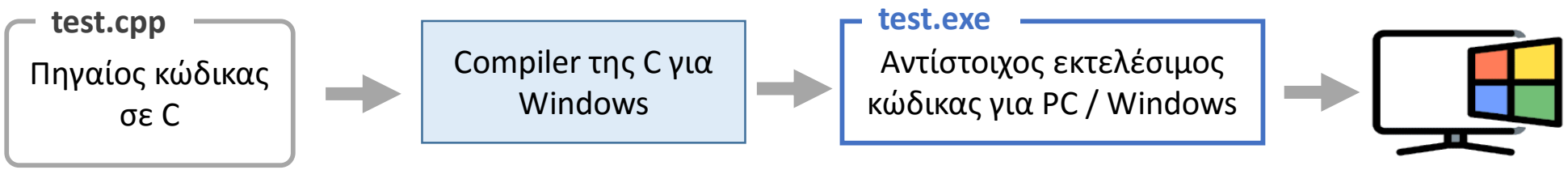
Ειδικό πρόγραμμα που μεταγλωττίζει / μετατρέπει

- *προγράμματα (κείμενα)* διατυπωμένα στην παραπάνω γλώσσα προγραμματισμού

  πηγαίος κώδικας – source code

- σε αντίστοιχο *δυαδικό κώδικα* της γλώσσας μηχανής αυτού του Η/Υ

  εκτελέσιμος κώδικας – machine code



Compiler και συντακτικά λάθη

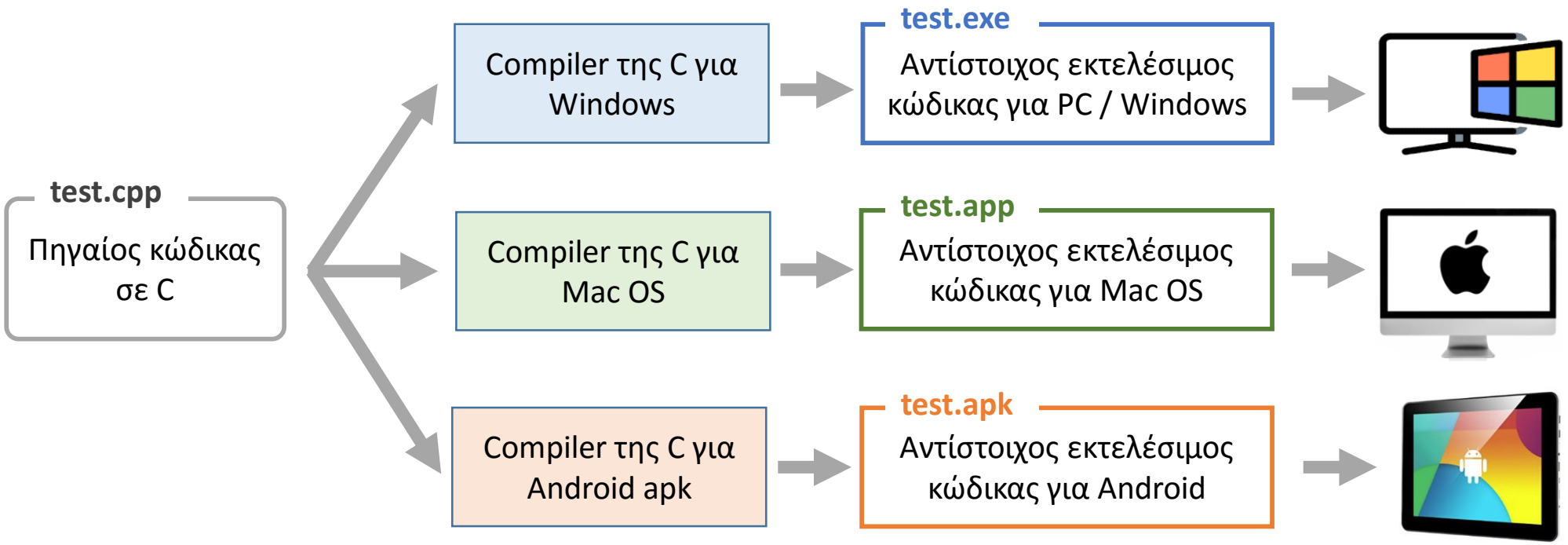
Πριν τη μεταγλώττιση ενός προγράμματος ο *compiler* ελέγχει αν η διατύπωση του προγράμματος είναι *συντακτικά σωστή* (εντοπίζει τα συντακτικά λάθη).

- **Αν δεν υπάρχουν συντακτικά λάθη**
  ο compiler προχωρά στην δημιουργία του δυαδικού κώδικα.
- **Αν υπάρχουν συντακτικά λάθη**
  μας επιστρέφει μια λίστα με τα λάθη που πρέπει να διορθώσουμε.

```
>ch -u ./hello.c
ERROR: missing ')'
ERROR: syntax error before or at line 4 in file C:\Ch\dem
  ==>:      printf("Hello, world\n";
  BUG:      printf("Hello, world\n";<== ???
ERROR: cannot execute command 'C:\Ch\demos\bin\hello.c'
>Exit code: -1
li=4 co=28 INS (CR+LF)
```

Διαφορετικοί compilers για την ίδια γλώσσα προγραμματισμού

Από τον *ίδιο*(*) *πηγαίο κώδικα* μπορούμε να δημιουργήσουμε *διαφορετικά εκτελέσιμα προγράμματα* για διαφορετικούς τύπους ηλεκτρονικών υπολογιστών

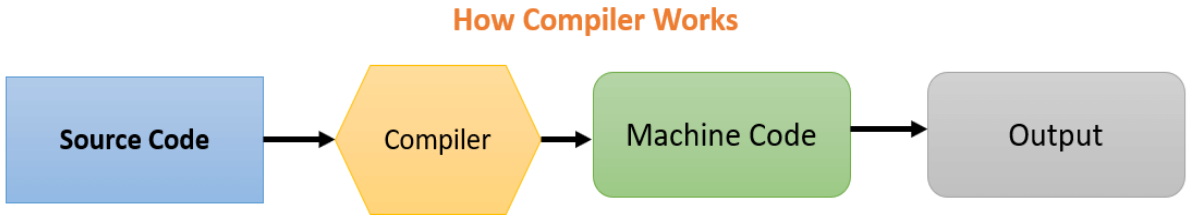


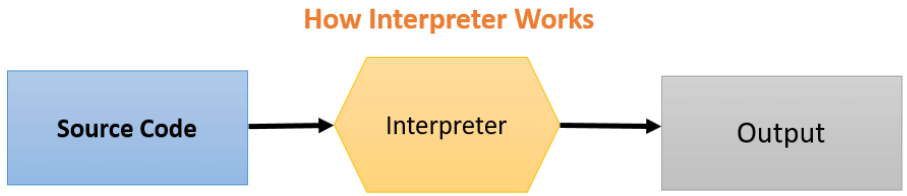
(*) *ανάλογα με τον τύπο του υπολογιστή ενδέχεται να υπάρχουν μικροδιαφορές στον πηγαίο κώδικα*

Η έννοια του Interpreter (Διερμηνέα)

Interpreter ***συγκεκριμένης*** γλώσσας προγραμματισμού για ***συγκεκριμένο*** τύπο Η/Υ

- Ειδικό πρόγραμμα που μεταγλωττίζει / μετατρέπει ***μία-μία*** τις εντολές μιας γλώσσας προγραμματισμού υψηλού επιπέδου σε γλώσσα μηχανής *και στη συνέχεια τις εκτελεί*. *Αν ο interpreter συναντήσει μια εντολή που δεν είναι συντακτικά σωστή, τότε η εκτέλεση του προγράμματος διακόπτεται*
- Τόσο ο compiler όσο και ο interpreter κάνουν βασικά την *ίδια δουλειά*, δηλαδή τη μετατροπή της γλώσσας προγραμματισμού υψηλού επιπέδου σε γλώσσα μηχανής
- Ωστόσο, ένας *compiler* μετατρέπει ***συνολικά*** τον κώδικα σε γλώσσα μηχανής *(δημιουργία αρχείου .exe , .app ή .apk ανάλογα με το είδος του υπολογιστή)* *πριν* από την εκτέλεσή του, ενώ ενας *interpreter* μετατρέπει ***μια-μια*** τις εντολές σε γλώσσα μηχανής και στη συνέχεια τις εκτελεί



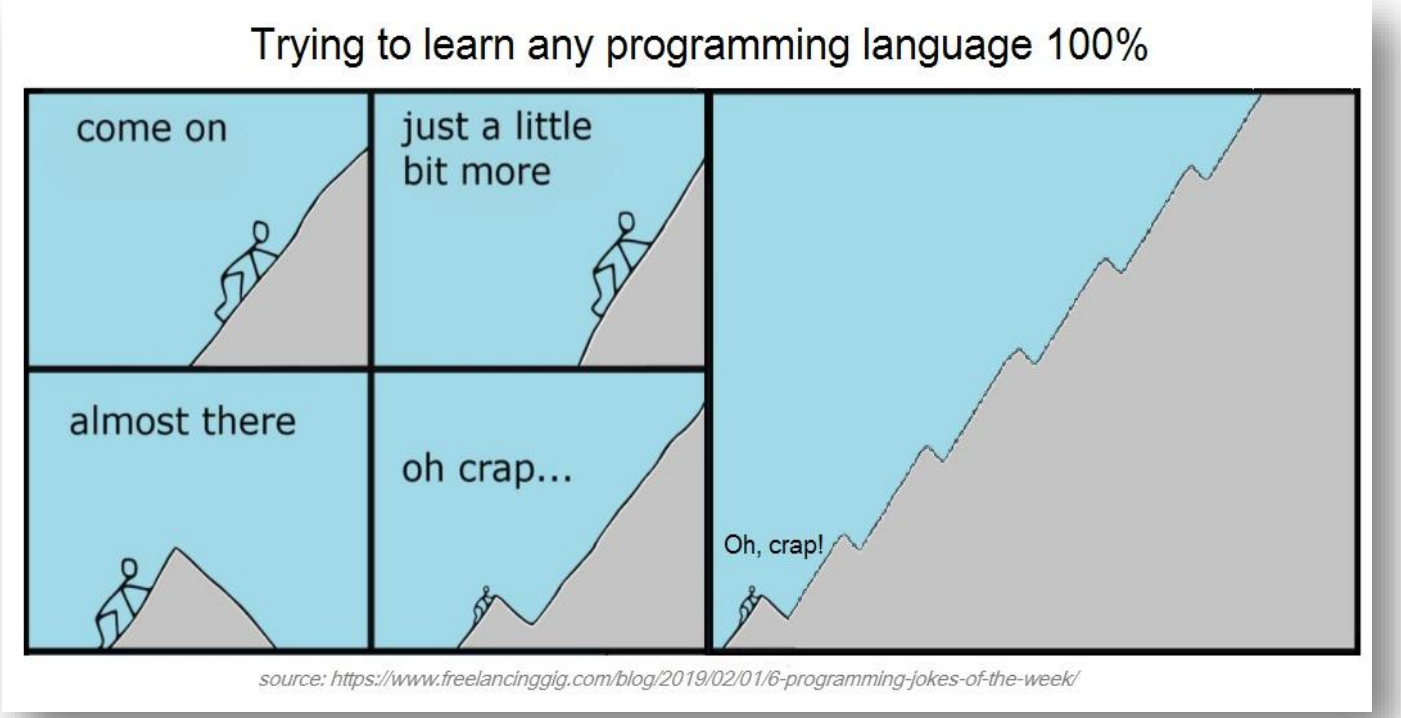


Διαφορές ανάμεσα σε compiler και interpreter

Compiler (Μεταγλωττιστής)	Interpreter (Διερμηνέας)
Σαρώνει όλο το πρόγραμμα και μεταφράζει το σύνολο του σε κώδικα μηχανής	Μεταφράζει μία-μία τις εντολές ενός προγράμματος
Χρειάζεται μεγαλύτερο χρόνο για την ανάλυση του πηγαίου κώδικα, αλλά ο συνολικός χρόνος εκτέλεσης είναι συγκριτικά ταχύτερος	Χρειάζεται μικρότερος χρόνος για την ανάλυση του πηγαίου κώδικα, αλλά ο συνολικός χρόνος εκτέλεσης είναι πιο αργός.
Δημιουργείται ενδιάμεσος κώδικας (object code) επομένως απαιτεί περισσότερη μνήμη	Δεν δημιουργείται ενδιάμεσος κώδικας (object code), επομένως είναι αποτελεσματικός στη μνήμη
Δημιουργεί συνολική λίστα σφαλμάτων μόνο μετά τη σάρωση ολόκληρου του προγράμματος (δυσκολότερο debugging)	Συνεχίζει τη μετάφραση *και εκτέλεση* του προγράμματος μέχρι να εντοπίσει το πρώτο σφάλμα, οπότε σταματά. (ευκολότερο debugging)
Γλώσσες προγραμματισμού όπως οι C, C++, FORTRAN, PASCAL χρησιμοποιούν μεταγλωττιστές	Γλώσσες προγραμματισμού όπως η Python και η Ruby χρησιμοποιούν διερμηνείς

Τέλος Διάλεξης

Ερωτήσεις;



Τμήματα αυτής της διάλεξης περιέχουν πληροφορίες από πηγές που είναι ελεύθερες στο διαδίκτυο όπως η Βικιπαίδεια και ανοιχτές σημειώσεις παρεμφερών διαλέξεων.