Γενικά περί υπολογιστών

- Υπολογιστής είναι ένα τεχνητό κατασκεύασμα που έχει την ικανότητα να επεξεργάζεται ένα σύνολο από *δεδομένα* που του δίνονται και να παράγει τα απαιτούμενα *αποτελέσματα*.
- Το είδος της επεξεργασίας που πρέπει να γίνει επάνω στα δεδομένα προσδιορίζεται από ένα *πρόγραμμα* με το οποίο έχουμε τροφοδοτήσει τον υπολογιστή και το οποίο αποτελείται από στοιχειώδεις εκτελέσιμες *εντολές* που είναι σε θέση να φέρει σε πέρας ο υπολογιστής.
- Γιατί χρειαζόμαστε υπολογιστές;
  - Για τη γρήγορη εκτέλεση χρονοβόρων αριθμητικών υπολογισμών, για παράδειγμα πόσο είναι το π, αν $\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \frac{1}{6^2} + \frac{1}{7^2} + \frac{1}{8^2} + \ldots$;
  - Για την αποδοτική διαχείριση ογκωδών δεδομένων, για παράδειγμα τραπεζικών λογαριασμών.
  - Και για τα δύο προηγούμενα, ταυτόχρονα, για παράδειγμα στην πρόγνωση καιρού.
- Η συγγραφή προγραμμάτων για υπολογιστές αναφέρεται με τον γενικό όρο *προγραμματισμός υπολογιστών*.

Γενικά περί προγραμματισμού υπολογιστών

- Στη γενική περίπτωση, προγραμματισμός (ενός υπολογιστή) είναι ο σαφής καθορισμός μίας διαδικασίας, σαν ένα σύνολο από εντολές (το πρόγραμμα), που περιγράφει λεπτομερώς τα βήματα που πρέπει να γίνουν για να επιλυθεί ένα πρόβλημα υπολογισμού.
- Τα προγράμματα για ένα υπολογιστή αποτελούν το *λογισμικό* του (software), ενώ η συσκευή του υπολογιστή είναι το *υλικό* (hardware).
- Ένα πρόγραμμα τροφοδοτείται σ΄ ένα υπολογιστή, αυτός το εκτελεί και παράγει το αποτέλεσμα της επίλυσης του προβλήματος.
- Η συγγραφή ενός προγράμματος για την επίλυση ενός προβλήματος προϋποθέτει την επινόηση ενός *αλγορίθμου*, δηλαδή μίας σαφώς καθορισμένης διαδικασίας μέσω ενός συνόλου εκτελέσιμων βημάτων, που είναι εγγυημένο ότι μετά από ένα πεπερασμένο πλήθος βημάτων που θα εκτελεσθούν θα τερματίσει.
- Ένα πρόγραμμα είναι η διατύπωση ενός αλγορίθμου σε μία συγκεκριμένη *γλώσσα προγραμματισμού*.

Ιστορική αναδρομή

- Άβακας, συσκευή προσθαφαιρέσεων με μπίλιες (1000 π.Χ.)
- Ο Μηχανισμός των Αντικυθήρων (75 π.Χ.)
- Muhammad ibn Musa al-Khwarizmi, πρότεινε την έννοια της διαδικασίας για αυτόματους υπολογισμούς (800 μ.Χ.)
- Μηχανικοί υπολογιστές για αριθμητικούς υπολογισμούς (Blaise Pascal, Gottfried Leibniz, 17ος αιώνας)
- Αναλυτική μηχανή του Charles Babbage με πρώτη προγραμματίστρια την Ada Lovelace (19ος αιώνας)
- Alan Turing, θεμελιωτής των εννοιών των αλγορίθμων και του υπολογισμού, η μηχανή Turing (1935)
- MARK I, ο πρώτος ηλεκτρομηχανικός υπολογιστής μεγάλης κλίμακας, IBM, Harvard University (1943)
- John von Neumann, θεμελιωτής της τρέχουσας αρχιτεκτονικής των υπολογιστών (1945)
- ENIAC, ο πρώτος υπολογιστής με λυχνίες κενού, University of Pennsylvania (1946)
- IBM 7090, ο πρώτος υπολογιστής με τρανζίστορ (1961)
- IBM 360, ο πρώτος υπολογιστής με ολοκληρωμένα κυκλώματα ή τσιπ (1964)
- Ο προσωπικός υπολογιστής, το PC (1981)
- Tim Berners-Lee, εισήγαγε την ιδέα του παγκόσμιου ιστού στο διαδίκτυο (1990)

Κάποια υπολογιστικά προβλήματα

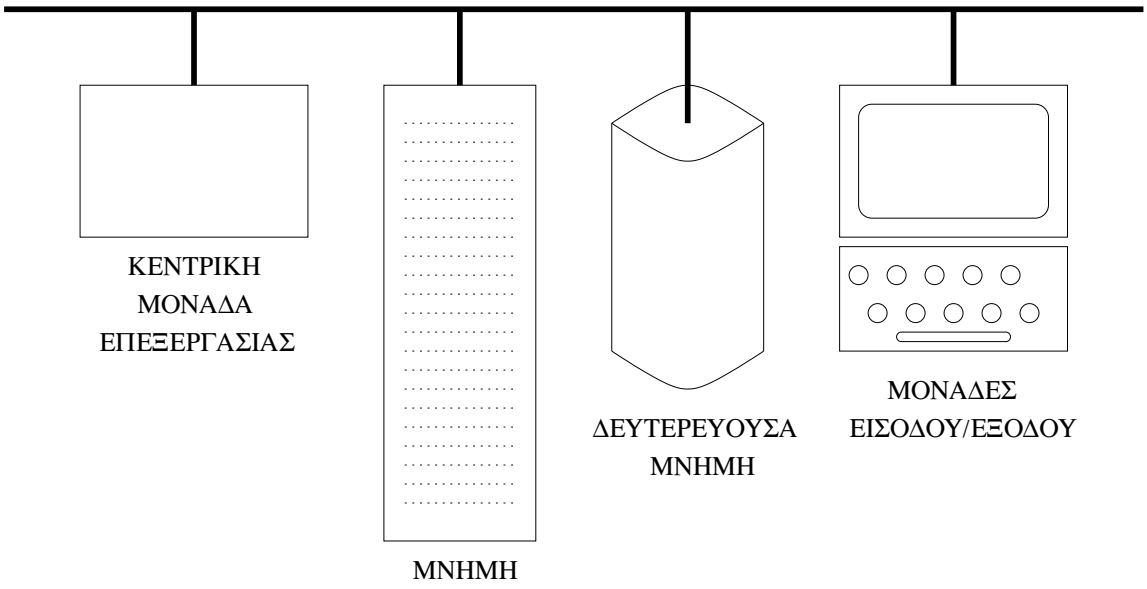
- Πώς προέκυψαν τα αποτελέσματα των πανελληνίων εξετάσεων για την εισαγωγή σε Α.Ε.Ι. και Τ.Ε.Ι.;
- Πώς αποφασίζεται αν ένας φοιτητής του Τμήματός μας έχει εκπληρώσει τις υποχρεώσεις του για λήψη του πτυχίου του;
- Τι φόρο θα πληρώσει η οικογένειά μας εφέτος;
- Πώς υπολογίστηκε το τελικό ποσό πληρωμής στους τελευταίους λογαριασμούς Ο.Τ.Ε και Δ.Ε.Η. που λάβαμε;
- Δεδομένων των αναλυτικών αποτελεσμάτων σε κάθε εκλογικό τμήμα στις εθνικές εκλογές, πόσους βουλευτές εκλέγει το κάθε κόμμα σε κάθε νομό;
- Δεδομένων των αποτελεσμάτων μίας δημοσκόπησης εξόδου (exit poll) σε επιλεγμένα εκλογικά τμήματα, πώς μπορούμε να προβλέψουμε το τελικό ποσοστό ψήφων σε όλη τη χώρα για κάθε κόμμα που συμμετέχει στις εθνικές εκλογές;
- Πώς μπορούμε να προβλέψουμε τον καιρό στην Αθήνα για αύριο ή για τις ερχόμενες ημέρες, δεδομένων όλων των απαιτούμενων μετεωρολογικών στοιχείων;
- Πώς γίνονται οι κρατήσεις αεροπορικών εισιτηρίων;
- Πώς μπορούμε να κατασκευάσουμε το καλύτερο ωρολόγιο πρόγραμμα για τα μαθήματα του τρέχοντος εξαμήνου;
- Το καλύτερο πρόγραμμα εξετάσεων για την ερχόμενη εξεταστική περίοδο;

Πώς να μάθουμε να προγραμματίζουμε;

- Δυστυχώς ο προγραμματισμός δεν διδάσκεται.
- Δεν μπορούμε να μάθουμε να προγραμματίζουμε διαβάζοντας βιβλία για προγραμματισμό.
- Δεν μπορούμε να μάθουμε να προγραμματίζουμε γράφοντας προγράμματα στο χαρτί.
- Ο μόνος τρόπος να μάθουμε να προγραμματίζουμε είναι να γράφουμε προγράμματα στον υπολογιστή, να τα εκτελούμε, να βρίσκουμε τα λάθη που σίγουρα έχουμε κάνει, να τα διορθώνουμε, να εκτελούμε πάλι τα προγράμματα, να ξαναβρίσκουμε λάθη, να τα διορθώνουμε πάλι, μέχρι να καταφέρουμε να κάνουμε τα προγράμματά μας να επιλύουν τα προβλήματά μας όπως πρέπει.
- Επικουρικά, η ανάγνωση και κατανόηση καλογραμμένων, τεκμηριωμένων και ορθών προγραμμάτων είναι πάντοτε χρήσιμη, αλλά από μόνη της δεν αρκεί.

Η δομή του υπολογιστή

- *Κεντρική μονάδα επεξεργασίας*, ο επεξεργαστής
- *Μνήμη*, για προσωρινή αποθήκευση δεδομένων και προγραμμάτων
- *Δευτερεύουσα μνήμη*, για μόνιμη αποθήκευση δεδομένων και προγραμμάτων (δίσκοι, δισκέτες, ταινίες, CDs, DVDs, κλπ.)
- *Μονάδες εισόδου/εξόδου*, για επικοινωνία με τον έξω κόσμο (πληκτρολόγιο, οθόνη, εκτυπωτής, κλπ.)



Η πληροφορία στον υπολογιστή

- Δυαδικά ψηφία 0 και 1 (bits - **b**inary dig**its**)
- Bytes, 1 byte = 8 bits
- Σύμβολα, ASCII κωδικοί (`'A'` = 65, `'B'` = 66, ...)
- Συστήματα αρίθμησης
  - δυαδικό
  - δεκαδικό
  - οκταδικό
  - δεκαεξαδικό

  $(01101101)_2 = (109)_{10} = (155)_8 = (6D)_{16}$
  $0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 109$
- Η μνήμη είναι οργανωμένη σε συνεχή κελιά, που το καθένα χαρακτηρίζεται από τη *διεύθυνσή* του, και μέσα στα οποία ο επεξεργαστής μπορεί να καταχωρήσει, αλλά και να διαβάσει απ΄ αυτά, πληροφορίες (δυαδικούς αριθμούς).
- Στη δευτερεύουσα μνήμη, η οργάνωση στο χαμηλό επίπεδο είναι περίπου όπως και στην (κύρια) μνήμη, αλλά οι καταχωρημένες πληροφορίες προσπελαύνονται από τον χρήστη μέσω *αρχείων* (files) και *καταλόγων* (directories), ή, κατά μία ορολογία, *φακέλων* (folders).
- Μονάδες μέτρησης χωρητικότητας μνήμης
  - 1 Kilobyte (KB) = 2^{10} (= $1024 \simeq 10^3$) bytes
  - 1 Megabyte (MB) = 2^{20} (= $1048576 \simeq 10^6$) bytes
  - 1 Gigabyte (GB) = 2^{30} (= $1073741824 \simeq 10^9$) bytes
  - 1 Terabyte (TB) = 2^{40} (= $1099511627776 \simeq 10^{12}$) bytes

Λογισμικό και γλώσσες προγραμματισμού

- Λογισμικό συστήματος
  - Λειτουργικό σύστημα (Microsoft Windows, Unix, ...)
  - Μεταγλωττιστές (`gcc`, `g++`, ...)
  - Κειμενογράφοι (`pico`, `vim`, ...)
  - ...
- Λογισμικό εφαρμογών
- Γλώσσες προγραμματισμού χαμηλού επιπέδου
  - Γλώσσα μηχανής
  - Assembly
- Γλώσσες προγραμματισμού υψηλού επιπέδου
  - Cobol, Fortran, Algol, Ada
  - Pascal, C
  - C++, Java, C#
  - Visual Basic
  - SQL
  - Prolog
  - Haskell, Common Lisp
  - PHP
  - Perl, Python
  - ...

Τι χρειαζόμαστε από μία γλώσσα προγραμματισμού;

- Μηχανισμούς για είσοδο (δεδομένων) και έξοδο (αποτελεσμάτων)
  Παράδειγμα:

  Διάβασε *Μισθό Υπαλλήλου*
  Υπολόγισε Κρατήσεις
  Υπολόγισε Καθαρό Μισθό
  Εκτύπωσε *Καθαρό Μισθό*

- Διατύπωση ακολουθίας βημάτων προς εκτέλεση
  Παράδειγμα:

  Έστω Ένα Ορθογώνιο Τρίγωνο ΑΒΓ
  με Ορθή Γωνία στο Α
  Βήμα 1*: Υπολόγισε το Τετράγωνο της ΑΒ*
  Βήμα 2*: Υπολόγισε το Τετράγωνο της ΑΓ*
  Βήμα 3*: Πρόσθεσε τα Δύο Τετράγωνα*
  Βήμα 4*: Η Πλευρά ΒΓ Ισούται με την*
  Τετραγωνική Ρίζα του Αποτελέσματος

- Φύλαξη πληροφοριών στη μνήμη και ανάκτησή τους
  Παράδειγμα:

  Μισθός = 1500
  Αύξηση = 0.04
  *Νέος Μισθός = Αύξηση * Μισθός + Μισθός*

- Διαφοροποίηση ροής του ελέγχου σύμφωνα με συνθήκη
  Παράδειγμα:

  Αν *Κατανάλωση < 200*
  Τότε *Κόστος = 20*
  Αλλιώς *Κόστος = 0.3 * Κατανάλωση - 40*

- Δυνατότητα διατύπωσης επαναληπτικών διαδικασιών
  Παράδειγμα:

  Αριθμός = 1
  Άθροισμα = 0
  Ενόσω *Αριθμός < 1001*
  Άθροισμα = Άθροισμα + Αριθμός^2
  Αριθμός = Αριθμός + 1

- Πακετάρισμα συχνά χρησιμοποιούμενων λειτουργιών
  Παράδειγμα:

  Ταξινόμησε*(Πίνακας Α)*
  Ταξινόμησε*(Πίνακας Β)*
  Συγχώνευσε*(Πίνακας Α, Πίνακας Β, Πίνακας Γ)*

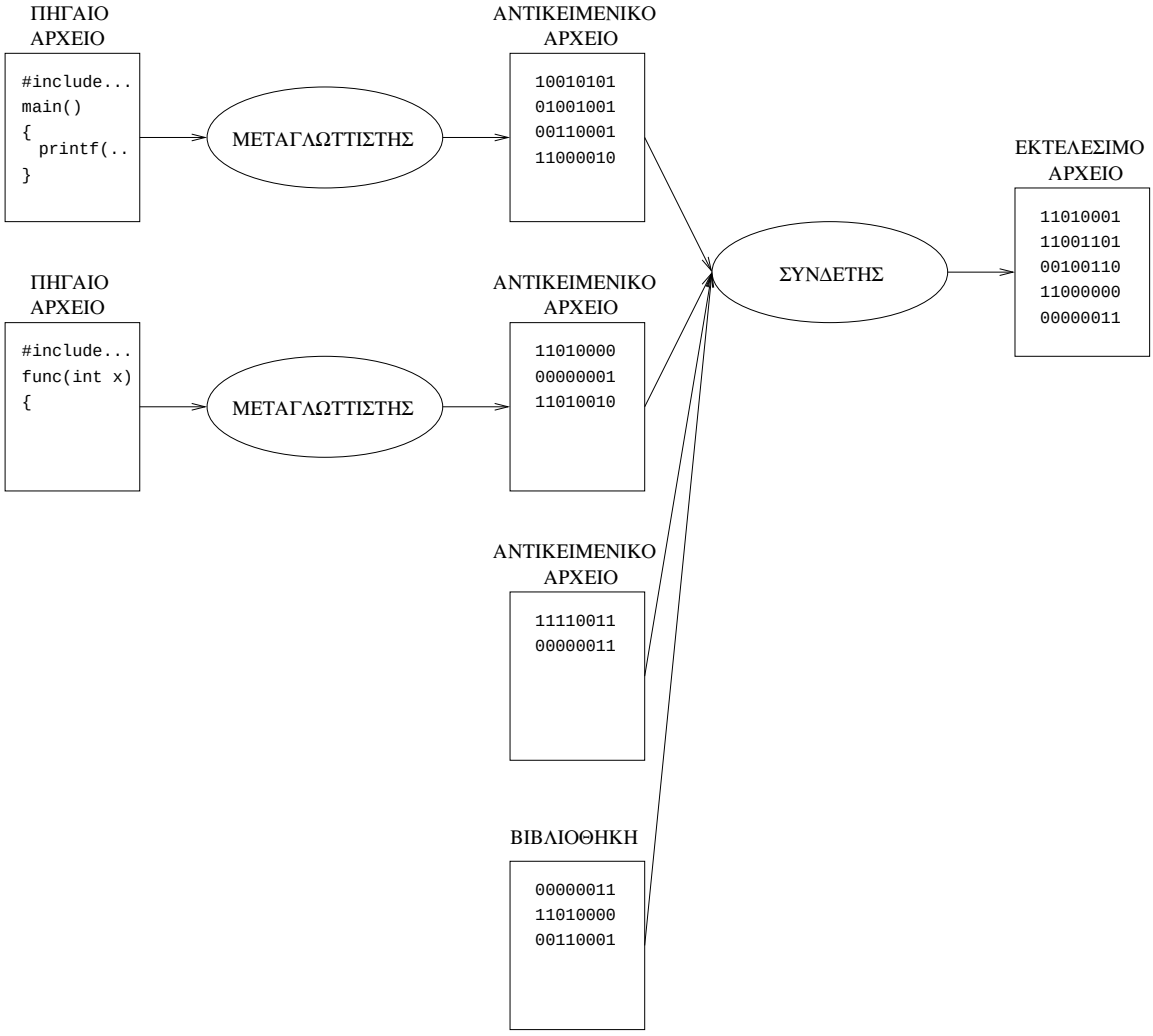
- Μεταφορά της ροής του ελέγχου αλλού (**πολύ σπάνια**)
  Παράδειγμα:

  Βήμα 1*: Αν Χ = 30*
  Τότε ***Πήγαινε*** *στο Βήμα 2*
  Αλλιώς ***Πήγαινε*** *στο Βήμα 4*
  Βήμα 2*: Εκτύπωσε "Τριάντα"*
  Βήμα 3*:* ***Πήγαινε*** *στο Βήμα 5*
  Βήμα 4*: Εκτύπωσε "Οχι τριάντα"*
  Βήμα 5*: ...*

Πώς κατασκευάζουμε εκτελέσιμο πρόγραμμα;

- Συνήθως, γράφουμε τα προγράμματά μας σε κάποια *γλώσσα προγραμματισμού υψηλού επιπέδου*, για παράδειγμα C.
- Για λόγους εύκολης συντήρησης ενός προγράμματος, είναι πιθανόν να το έχουμε διασπάσει σε πολλά *πηγαία αρχεία* (source files), που περιλαμβάνουν *κώδικα* (code) γραμμένο στη γλώσσα προγραμματισμού που έχουμε επιλέξει.
- Ο *μεταγλωττιστής* (compiler) είναι ένα πρόγραμμα που μετατρέπει τα πηγαία αρχεία σε *αντικειμενικά αρχεία* (object files), τα οποία είναι διατυπωμένα στη γλώσσα μηχανής του επεξεργαστή του υπολογιστή μας.
- Τα αντικειμενικά αρχεία δεν είναι άμεσα εκτελέσιμα από τον επεξεργαστή, εκτός του ότι το καθένα απ΄ αυτά μπορεί να μην συνιστά ένα πλήρες πρόγραμμα.
- Ο *συνδέτης* (linker) μετατρέπει ένα σύνολο από αντικειμενικά αρχεία, καθώς και τυχόν *βιβλιοθήκες* (libraries) που θα του δοθούν, σ΄ ένα *εκτελέσιμο αρχείο* (executable file), το οποίο επίσης είναι διατυπωμένο σε γλώσσα μηχανής, αλλά είναι άμεσα εκτελέσιμο από τον επεξεργαστή.

Η διαδικασία της μεταγλώττισης και σύνδεσης



Προγραμματιστικά περιβάλλοντα για την C

- Μεταγλωττιστής `gcc` της C, από το πρόγραμμα ελεύθερου λογισμικού GNU, σε λειτουργικό σύστημα Unix (για παράδειγμα, Solaris σε σταθμούς εργασίας Sun ή Linux σε προσωπικούς υπολογιστές)
- Περιβάλλοντα που υποστηρίζουν τον μεταγλωττιστή `gcc`, κάτω από λειτουργικό σύστημα Microsoft Windows
  - `Dev-C++`
  - `Cygwin`
  - `MinGW`
- Περιβάλλον Microsoft Visual Studio για Windows, αλλά **δεν θα χρησιμοποιηθεί στα πλαίσια του παρόντος μαθήματος**

Παραδείγματα χρήσης του gcc

- `gcc myprog.c`
  Μεταγλώττιση του πηγαίου προγράμματος `myprog.c` σε αντικειμενικό αρχείο και κλήση του συνδέτη για την κατασκευή του εκτελέσιμου προγράμματος `a.out`
- `gcc -o myprog myprog.c`
  Μεταγλώττιση του πηγαίου προγράμματος `myprog.c` σε αντικειμενικό αρχείο και κλήση του συνδέτη για την κατασκευή του εκτελέσιμου προγράμματος `myprog`
- `gcc -c myprog.c`
  Μόνο μεταγλώττιση του πηγαίου προγράμματος `myprog.c` στο αντικειμενικό αρχείο `myprog.o`
- `gcc -o prog myprog1.o myprog2.o -lm`
  Μόνο κλήση του συνδέτη για κατασκευή του εκτελέσιμου προγράμματος `prog` από τα αντικειμενικά αρχεία `myprog1.o` και `myprog2.o` και τη μαθηματική βιβλιοθήκη (`m`)
- Άλλες ενδιαφέρουσες επιλογές του `gcc`, εκτός από τις `-o`, `-c` και `-l`:
  - Για να κληθεί μόνο ο προεπεξεργαστής, `-E`
  - Για να παραχθεί το αποτέλεσμα σε γλώσσα assembly, `-S`