Κεφάλαιο 7
Προγραμματισμός Υπολογιστή

7.1 Η προγραμματιζόμενη μηχανή

Από τις βασικότερες διαφορές ανάμεσα στον υπολογιστή και στις περισσότερες ηλεκτρονικές συσκευές είναι η δυνατότητα προγραμματισμού του.

7.2 Οι χρήστες

Τους χρήστες των υπολογιστικών συστημάτων μπορούμε να τους κατατάξουμε σε δύο μεγάλες κατηγορίες:

α) **τελικοί χρήστες**: χρησιμοποιούν τον υπολογιστή στη δουλειά τους, π.χ. οι αρχιτέκτονες, οι συγγραφείς, οι λογιστές, κλπ.

β) **επαγγελματίες της Πληροφορικής**: προγραμματιστές

7.3 Πρόγραμμα – Γλώσσες προγραμματισμού

7.3.1 Πρόγραμμα

Το πρόγραμμα αποτελείται

α) από μια σειρά οδηγιών, που ονομάζονται **εντολές**, για την εκτέλεση αριθμητικών πράξεων (πρόσθεση, αφαίρεση, κλπ.) και λογικών πράξεων (AND, OR, NOT, κλπ.)

β) σύνολο πρόσθετων **οδηγιών ελέγχου**, που επηρεάζουν τη σειρά με την οποία εκτελούνται οι εντολές (δομές ελέγχου και επανάληψης)

7.3.2 Γλώσσα μηχανής

Ο κάθε τύπος ΚΜΕ έχει διαφορετικό ρεπερτόριο εντολών =>

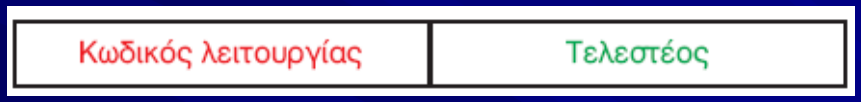
τα προγράμματα που εκτελεί πρέπει να είναι διατυπωμένα στη δική της **γλώσσα μηχανής** και αποτελούνται από ακολουθίες 0 και 1

Οι εντολές που αναγνωρίζει μια τυπική ΚΜΕ ανήκουν σε μια από τις πιο κάτω κατηγορίες:

- εντολές μεταφοράς δεδομένων μεταξύ της κεντρικής μνήμης και των καταχωρητών της ΚΜΕ
- εντολές μεταφοράς δεδομένων μεταξύ των καταχωρητών
- εντολές αριθμητικών πράξεων
- εντολές λογικών πράξεων
- εντολές ελέγχου της ροής εκτέλεσης των εντολών
- διάφορες βοηθητικές εντολές.

7.3.2 Γλώσσα μηχανής

Κάθε εντολή στη γλώσσα μηχανής αποτελείται από δύο τμήματα, τον **κωδικό λειτουργίας** (operation code, OP code) και τον **τελεστέο** (operand).



- Ο κωδικός λειτουργίας προσδιορίζει τη λειτουργία της εντολής, για παράδειγμα «πρόσθεσε στο συσσωρευτή»
- Ο τελεστέος αφορά τα δεδομένα στα οποία δρα η εντολή ή τη διεύθυνση στην οποία βρίσκονται αποθηκευμένα.

Τα πρώτα προγράμματα γράφονταν σε γλώσσα μηχανής

Αναζητήθηκαν τρόποι να γράφονται τα προγράμματα σε γλώσσα πιο προσιτή στον άνθρωπο

```
1110011010110011
1011011010111011
10100111101101101010010
0101011010010011
1110011010110110
11000110101101111010100
1101011000110101
10101110
1100011010110010
(Κωδικός Λειτουργίας/Τελεστέος)
```

7.3.3 Συμβολικές γλώσσες (assembly)

Στις συμβολικές γλώσσες, σε κάθε εντολή της γλώσσας μηχανής αντιστοιχίζεται μια **μνημονική** λέξη η οποία θυμίζει το σκοπό της εντολής.

```
mov   ax,DGROUP
mov   ds,ax
mov   ds:[STKHQQ],0
mov   es:[cdataseg],ax
mov   ax,sp
add   ax,bpregz+pgdata+14h
mov   [_atopsp],ax
mov   bx,seg STACK
sub   bx,DGROUP
mov   cl,4
shl   bx,cl
add   ax,bx
mov   [_abrktb].sz,ax
dec   ax
mov   [_asizds],ax
```

7.3.4 Γλώσσες υψηλού επιπέδου

Τα προγράμματα που γράφονται σε συμβολική γλώσσα ή γλώσσα μηχανής μπορούν να εκτελεστούν μόνο από τον τύπο υπολογιστή για τον οποίο είναι γραμμένα.

Για να αντιμετωπιστούν τέτοιου είδους προβλήματα, άρχισαν να δημιουργούνται και να εξελίσσονται νέες γλώσσες προγραμματισμού πιο απλές και ανεξάρτητες από το συγκεκριμένο τύπο υπολογιστή, που ονομάστηκαν **γλώσσες υψηλού επιπέδου** (high level languages).

Μεταφερσιμότητα προγράμματος: με μικρές πιθανόν αλλαγές στον πηγαίο κώδικα μπορούν να εκτελεστούν στη συνέχεια και σε άλλους τύπους υπολογιστών

7.3.5 Ένα συγκριτικό παράδειγμα

Έστω ότι θέλουμε να προσθέσουμε το περιεχόμενο δύο θέσεων μνήμης και το αποτέλεσμα να το καταχωρίσουμε σε μια τρίτη:

Γλώσσα υψηλού επιπέδου

Εντολή	Περιγραφή
`A:= B + C`	Πρόσθεσε το περιεχόμενο των μεταβλητών B και C και το αποτέλεσμα καταχώρισέ το στη μεταβλητή A

Συμβολική γλώσσα

Εντολή	Περιγραφή
`LDA B`	Μετάφερε στο συσσωρευτή το περιεχόμενο της θέσης μνήμης με όνομα B
`ADD C`	Πρόσθεσε στο περιεχόμενο του συσσωρευτή το περιεχόμενο της θέσης μνήμης με όνομα C
`STA A`	Μετάφερε και αποθήκευσε το περιεχόμενο του συσσωρευτή στη θέση μνήμης με όνομα A

Γλώσσα μηχανής

Εντολή	Περιγραφή
0000001001011010	Μετάφερε στο συσσωρευτή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011010
0000101001011110	Πρόσθεσε στο περιεχόμενο του συσσωρευτή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011110
0000011011011110	Μετάφερε και αποθήκευσε το περιεχόμενο του συσσωρευτή στη θέση μνήμης με διεύθυνση 11011110

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

FORTRAN

1954 IBM, (**FOR**mula **TRAN**slation - Μετάφραση Μαθηματικών Τύπων). Κατάλληλη για επιστημονικές εφαρμογές.

```
      SUBROUTINE RAND(U,N)
      INTEGER N,X,Y,Z
      REAL U(N),V
      COMMON/RANDOM/X,Y,Z
      IF(N.LE.0) RETURN
      DO 1 I=1,N
           X=171*MOD(X,177)-2*(X/177)
           IF(X.LT.0) X=X+30269
           Y=172*MOD(Y,176)-35*(Y/176)
           IF(Y.LT.0) Y=Y+30307
           Z=170*MOD(Z,178)-63*(Z/178)
           IF(Z.LT.0) Z=Z+30323
           V=X/30269.0 + Y/30308.0 + Z/30323.0
1          U(I)=V-INT(V)
      RETURN
      END
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

ALGOL

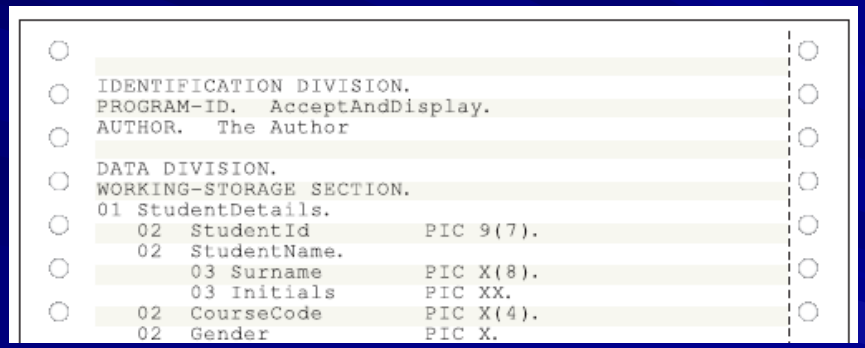
1963 – υλοποίησε τον δομημένο προγραμματισμό

```
BEGIN
REAL X, Y;
    PROCEDURE P;
    BEGIN
    REAL Z;
      . . .
      X := X * Z;
    END P;
    PROCEDURE Q;
    BEGIN
    INTEGER Y, X;
      P;
    END;
    Q;
END
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

COBOL

(**CO**mmon **B**usiness **O**riented **L**anguage - Κοινή Γλώσσα Εμπορικού Προσανατολισμού), σχεδιάστηκε από το Υπουργείο Άμυνας των Η.Π.Α



```
IDENTIFICATION DIVISION.
PROGRAM-ID.   AcceptAndDisplay.
AUTHOR.   The Author

DATA DIVISION.
WORKING-STORAGE SECTION.
01 StudentDetails.
   02  StudentId          PIC 9(7).
   02  StudentName.
       03 Surname         PIC X(8).
       03 Initials        PIC XX.
   02  CourseCode         PIC X(4).
   02  Gender             PIC X.
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

BASIC

(**B**eginner.s **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) αναπτύχθηκε από τους Kemeney και Kurtz στα μέσα της δεκαετίας του 1960, ως μια γλώσσα για τη διδασκαλία προγραμματισμού σε φοιτητές

```
print "Compute: 1+2+3+4+5+...+n"
input "Enter n: ", n
for i=1 to n
  isum = isum + i
next
print "The sum is:"; isum
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

SIMULA

1966 Νορβηγία από τους Dahl και Nygaard. Είναι γλώσσα βασισμένη στην ALGOL με βασικό τομέα εφαρμογής της την προσομοίωση

```
Class Rectangle (Width, Height); Real Width, Height;
 Begin
    Real Area, Perimeter;
    Procedure Update;
    Begin
      Area := Width * Height;
      Perimeter := 2*(Width + Height)
    End of Update;
    Boolean Procedure IsSquare;
      IsSquare := Width=Height;
    Update;
    OutText("Rectangle created: ");
    OutFix(Width,2,6);
    OutFix(Height,2,6); OutImage
 End of Rectangle;
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

LISP

1950 MIT – εφαρμογή στην τεχνητή νοημοσύνη

```
(defun flatten (s-expression)
    (if (atom s-expression)
        s-expression
        (flatten-aux s-expression)
    )
)

(defun flatten-aux (somelist)
    (if (listp (car somelist))
        (append (flatten (car somelist))
           (flatten (cdr somelist))
        )
        (cons (car somelist)
           (flatten (cdr somelist))
        )
    )
)
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

LOGO

Παιδαγωγικές προδιαγραφές με δημιουργία γραφικών και γεωμετρικών σχημάτων

```
to disstack :num
setcursor list (-3 + 5 * :num) 4
type ifelse stackemptyp hidden :num ["| |] ["-]
if stackemptyp shown :num [setcursor list (-4 + 5 * :num) 5
                           type "|    | stop]
localmake "stack (thing shown :num)
localmake "col 5*:num-4
for [i [count :stack] 1] [
    [setcursor list :col :i+4
     carddis pop "stack]
end

to distop :suit
if emptyp top :suit [stop]
if equalp :suit "H [distop1 4 stop]
if equalp :suit "S [distop1 11 stop]
if equalp :suit "D [distop1 18 stop]
distop1 25
end
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

PASCAL

1971 Wirth : αποδοτικός κώδικας κατά τη μεταγλώττιση

```
PROGRAM arrayc (output);
TYPE   days = (sun,mon,tues,weds,thurs,fri,sat);
       dname = string(8);

VAR    d: days;

FUNCTION DayName (fd: days): dname;
  TYPE  abbrevs = ARRAY [days] OF
                  PACKED ARRAY [1..5] OF char;
  CONST DayNames = abbrevs
        [ sun:'Sun';mon:'Mon';tues:'Tues';
          weds:'Weds';thurs:'Thurs';fri:'Fri';
          sat:'Satur' ];
  BEGIN
    DayName := trim(DayNames[fd]) + 'day';
  END {DayName};

BEGIN {program}
  FOR d := fri DOWNTO mon DO writeln(DayName(d));
END.
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

C

1972 AT&T : συγγραφή λειτουργικών συστημάτων

```
void
InitBlock(float *a, float *b, float *c,
          int blk, int row, int col)
{
    int len, ind;
    int i,j;

    srand(pvm_mytid());
    len = blk*blk;
    for (ind = 0; ind < len; ind++) {
       a[ind] = (float)(rand()%1000)/100.0;
       c[ind] = 0.0;
       }
    for (i = 0; i < blk; i++) {
      for (j = 0; j < blk; j++) {
           if (row == col)
                b[j*blk+i] = (i==j)? 1.0 : 0.0;
           else
                b[j*blk+i] = 0.0;
            }
        }
}
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

C++

Εξέλιξη της C – υλοποίηση του αντικειμενοστραφούς προγραμματισμού

```
List&  List::
InsertLast(const Data &d)
{ SHOW("InsertLast(const Data&)");
  Node * const t = new Node(d);

  if (last == NULL) {
    first = t;
  }else {
    last->next = t;
  }
  last = t;
  ++num;
  return *this;
}
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

Smalltalk

Υλοποίηση του αντικειμενοστραφούς προγραμματισμού

```
insetBy: delta
    | newrect |
    newrect _ delta asRectangle.
    ^Rectangle origin: (origin+(newrect origin))
               corner: (corner-(newrect corner))
!

insetOriginBy: originDeltaPoint corner: cornerDeltaPoint
    ^Rectangle origin: origin + originDeltaPoint
               corner: corner + cornerDeltaPoint
!

merge: aRectangle
    | orig corn |
    orig _ Point x: ((origin x) min:(aRectangle origin x))
                 y: ((origin y) min:(aRectangle origin y)).
    corn _ Point x: ((corner x) max:(aRectangle corner x))
                 y: ((corner y) max:(aRectangle corner y)).
    ^Rectangle origin: orig corner: corn
!
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

ADA

Προγραμματισμός Ενσωματωμένων Υπολογιστικών Συστημάτων (Embeded Computer Systems-ECS).

Χαρακτηρίζονται από μεγάλο μέγεθος και υψηλού βαθμού πολυπλοκότητα, ενώ χρειάζονται διαρκή εξέλιξη.

```
for Hour in 1..Num_Hours loop
  Integer_Text_IO.Get (Item => Temp);
  Integer_Text_IO.Put (Item => Temp, Width => 4);

  if Temp < Low then
    Low := Temp;
  end if;

  if Temp > High then
    High := Temp;
  end if;
end loop;
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

Java

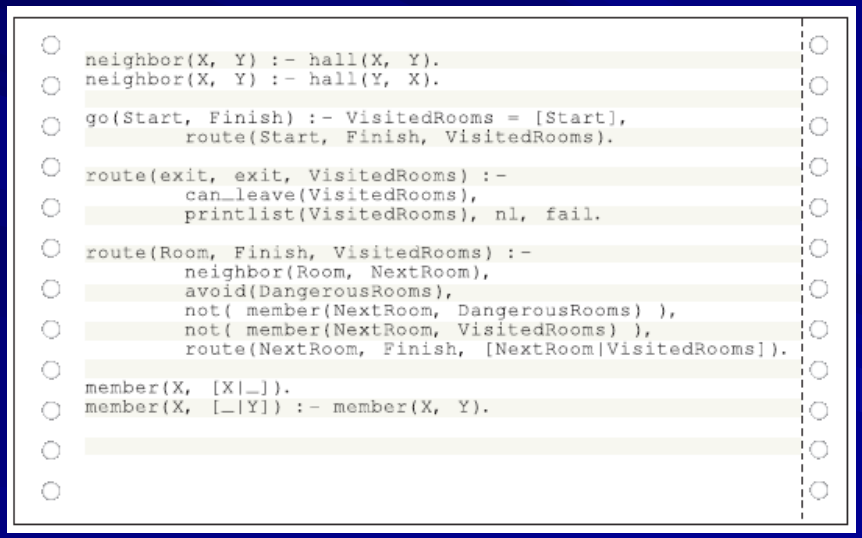
1990 – Sun Microsystems – η γλώσσα του διαδικτύου

```
public class Fibonacci {
  public static void main(String[] args) {
    int current, prev = 1, prevprev = 0;
    for(int i = 0; i < 20; i++) {
      current = prev + prevprev;
      System.out.print(current + " ");
      prevprev = prev;
      prev = current;      }
    System.out.println();
  }
}
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

Prolog

(**PRO**gramming in **LOG**ic) 1973 Γαλλία - εφαρμογές τεχνητής νοημοσύνης.



```
neighbor(X, Y) :- hall(X, Y).
neighbor(X, Y) :- hall(Y, X).

go(Start, Finish) :- VisitedRooms = [Start],
        route(Start, Finish, VisitedRooms).

route(exit, exit, VisitedRooms) :-
        can_leave(VisitedRooms),
        printlist(VisitedRooms), nl, fail.

route(Room, Finish, VisitedRooms) :-
        neighbor(Room, NextRoom),
        avoid(DangerousRooms),
        not( member(NextRoom, DangerousRooms) ),
        not( member(NextRoom, VisitedRooms) ),
        route(NextRoom, Finish, [NextRoom|VisitedRooms]).

member(X, [X|_]).
member(X, [_|Y]) :- member(X, Y).
```

7.3.6 Ιστορία των γλωσσών υψηλού επιπέδου

SQL

(Structured Query Language) - 1974 IBM - γλώσσα επικοινωνίας με τις σχεσιακές βάσεις δεδομένων.

```
CREATE TABLE STATS
(ID INTEGER REFERENCES STATION(ID), MONTH INTEGER CHECK
(MONTH BETWEEN 1 AND 12),
TEMP_F REAL CHECK (TEMP_F BETWEEN -80 AND 150),
RAIN_I REAL CHECK (RAIN_I BETWEEN 0 AND 100),
PRIMARY KEY (ID, MONTH));

SELECT MONTH, ID, RAIN_I, TEMP_F
FROM STATS ORDER BY MONTH,
RAIN_I DESC;
```

7.3.7 Εξάρτηση των γλωσσών από το σκοπό

Γλώσσες ειδικού σκοπού - π.χ. FORTRAN: προσανατολισμένη στις επιστημονικές εφαρμογές

Γλώσσες γενικού σκοπού - π.χ. Java

7.3.8 Μεταφραστές

Στόχος: τα προγράμματα να μεταφραστούν σε γλώσσα μηχανής

- **Συμβολομεταφραστές** (assemblers): μετατροπή των προγραμμάτων από συμβολική γλώσσα σε γλώσσα μηχανής

Υπάρχουν δύο τρόποι για την εκτέλεση από τον υπολογιστή προγραμμάτων που είναι γραμμένα σε γλώσσες υψηλού επιπέδου:

- **Μεταγλωττιστής** (compiler) π.χ. C++:

- **Διερμηνευτής** (interpreter) π.χ. Java: αναλαμβάνει να εκτελέσει μία μία τις εντολές του προγράμματος. Το πρόγραμμα δεν είναι αυτόνομο, δεν μπορεί να εκτελεστεί ανεξάρτητα, εφόσον απαιτείται η παρουσία του αντίστοιχου διερμηνευτή.

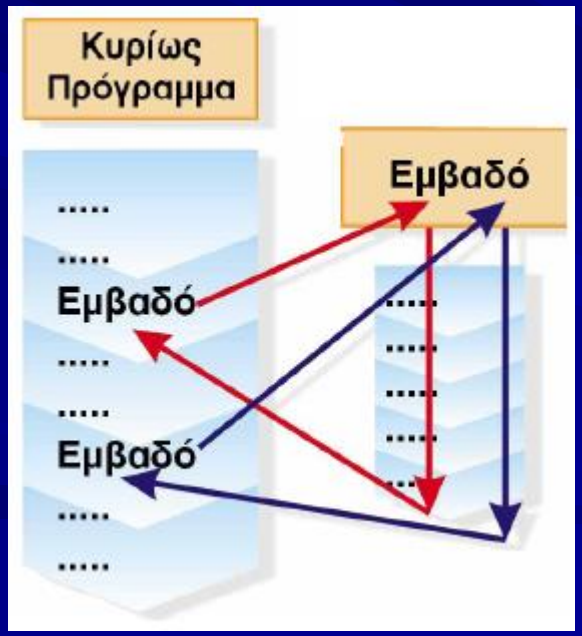
7.4 Αρχές κατασκευής λογισμικού

Τμηματικός προγραμματισμός (modular programming): τεμαχιστός του μεγάλου προγράμματος σε μικρότερα τμήματα, τα οποία θα κατασκευαστούν ανεξάρτητα και στη συνέχεια θα συνδυαστούν

Υπορουτίνα (subroutine): τμήμα κώδικα με ένα όνομα που μπορεί να εκτελεστεί πολλές φορές με αναφορά στο όνομά της

Δομημένος προγραμματισμός (structured programming): διάσπαση των λειτουργιών του προγράμματος σε άλλες, απλούστερες και ανεξάρτητες και υλοποίησή τους με δομές:

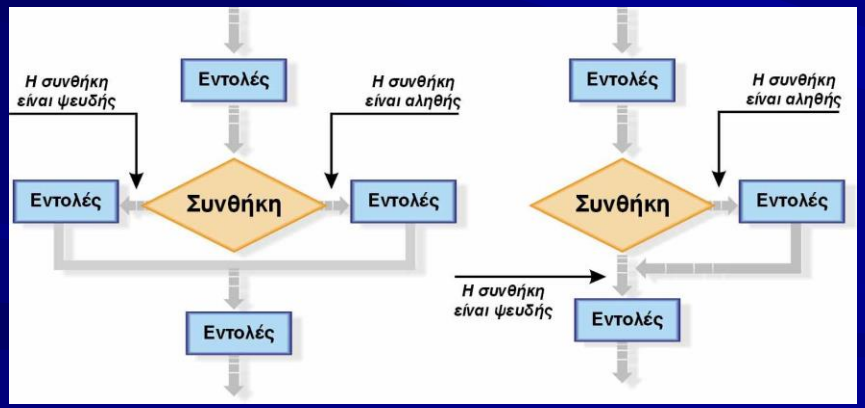
- Η διαδοχή (ακολουθία). Οι εντολές βρίσκονται σε ακολουθία και εκτελούνται με τη σειρά που είναι γραμμένες.
- Η επιλογή. Η εκτέλεση των εντολών εξαρτάται από την τιμή αλήθειας της συνθήκης.
- Η επανάληψη. Οι εντολές επαναλαμβάνονται, εφόσον αληθεύει ή δεν αληθεύει η συνθήκη.

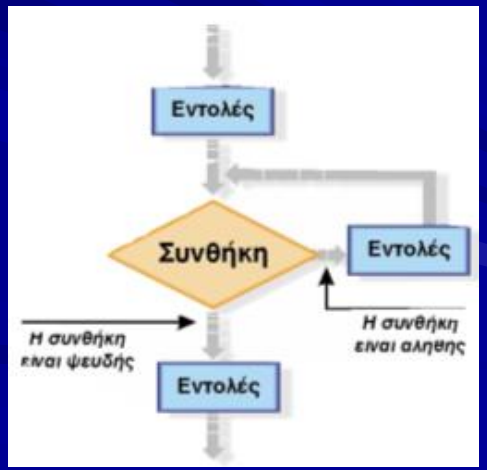


7.5 Πρότυπα προγραμματισμού
7.5.1 Διαδικαστικός προγραμματισμός

Το πρόγραμμα αποτελείται από δύο ξεχωριστά δομικά στοιχεία:

- από εντολές που περιγράφουν βήμα βήμα τη διαδικασία επίλυσης του προβλήματος:
  - Εντολές ανάθεσης (εκχώρησης)
  - Εντολές συνθήκης.
  - Εντολές επανάληψης
  - Εντολές για είσοδο και έξοδο στοιχείων.
- από δομές δεδομένων, στις οποίες αποθηκεύονται τα δεδομένα του προβλήματος τα οποία χειρίζονται οι εντολές.



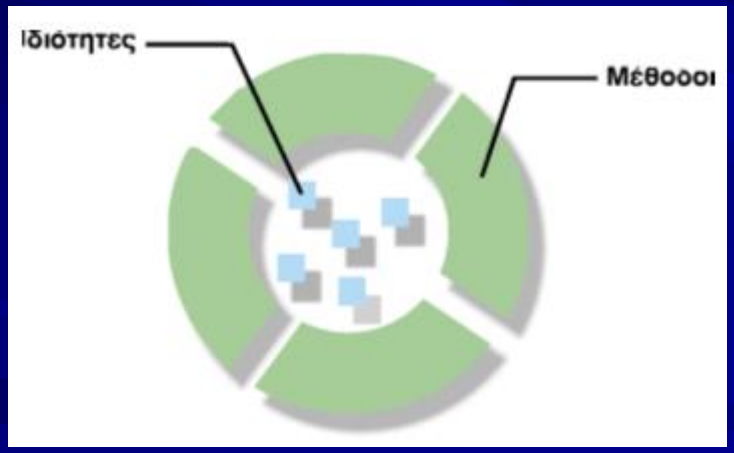


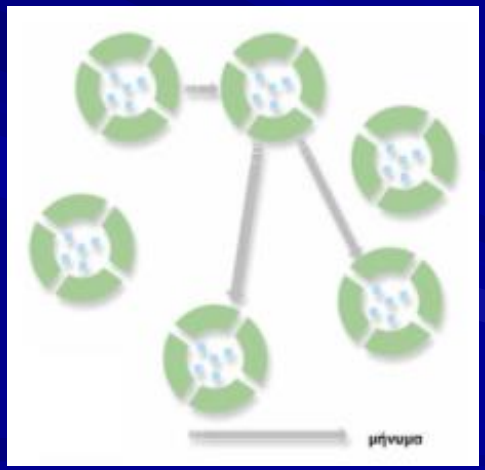
7.5.2 Αντικειμενοστρεφής προγραμματισμός

Αντικειμενοστρεφής προγραμματισμός (object oriented programming): φυσικός τρόπος για να περιγραφεί ένα σύστημα με προσομοίωση με περιγραφή των αντικειμένων από τα οποία αποτελείται, της συμπεριφοράς τους καθώς και του τρόπου με τον οποίο αλληλεπιδρούν μεταξύ τους. Π.χ. C++, Java.

Αντικείμενο (object): προγραμματιστικό «πακέτο», το οποίο αποτελείται από

- **διαδικασίες** (methods)
- **δεδομένα** σχετιζόμενα μεταξύ τους: ιδιότητες (properties)





7.5.3 Λογικός προγραμματισμός

Λογικός προγραμματισμός (logic programming): σύνολο από λογικές προτάσεις και ένα μηχανισμό εξαγωγής συμπερασμάτων, μέσω του οποίου υποβάλουμε ερωτήσεις στο πρόγραμμα και αυτό μας απαντάει αν αληθεύουν ή όχι.

Π.χ. (Prolog) το γενεαλογικό δένδρο των θεών του Ολύμπου:

```
πατέρας(Δίας, Άρης).
πατέρας(Δίας, Διόνυσος).
πατέρας(Άρης, Αρμονία).
πατέρας(Κάδμος, Σεμέλη).
μητέρα(Ήρα, Άρης).
μητέρα(Αρμονία, Σεμέλη).
μητέρα(Σεμέλη, Διόνυσος).
μητέρα(Αφροδίτη, Αρμονία).
αδέλφια(X,Y) αν μητέρα(Z,X) και μητέρα(Z,Y).
αδέλφια(X,Y) αν πατέρας(Z,X) και πατέρας(Z,Y).
```

Μπορούμε να υποβάλουμε την ερώτηση: «είναι ο Δίας πατέρας του Διόνυσου» και να μας απαντήσει «ΝΑΙ» ή να ρωτήσουμε «ποια είναι τα αδέλφια του Άρη» και να απαντήσει «ο Διόνυσος».

7.5.4 Συναρτησιακός προγραμματισμός

Συναρτησιακός προγραμματισμός (functional programming): έχει ως βάση τη συνάρτηση με τη μαθηματική της έννοια, δηλαδή αυτή της απεικόνισης από ένα πεδίο ορισμού σε ένα πεδίο τιμών.

Π.χ. (Miranda) δευτεροβάθμια εξίσωση:

```
quadsolve a b c
     = error "Μιγαδικές ρίζες", if delta < 0
     = [-b/2*a)],if delta = 0
     = [-b/(2*a) + radix/(2*a),
        -b/(2*a) - radix/(2*a)]if delta > 0
where
delta = b*b - 4*a*c
radix = sqrt delta

------------------------------------
sqrt a: είναι συνάρτηση που βρίσκει
        την τετραγωνική ρίζα του a
```

7.6 Προγραμματίζοντας (βήματα)

7.6.1 Καθορισμός προβλήματος

Ο στόχος είναι να εξαλείψουμε τα ασήμαντα στοιχεία και να οδηγηθούμε στην ουσία του προβλήματος

7.6.2 Ανάλυση προβλήματος

- καθορισμός των «εισόδων», δηλαδή των δεδομένων με τα οποία θα δουλέψουμε (inputs)
- καθορισμόσ των «εξόδων», δηλαδή των απαιτουμένων αποτελεσμάτων (outputs)
- αναζήτηση άλλων απαιτήσεων ή περιορισμών

7.6.3 Σχεδιασμός

Σχεδίαση του αλγόριθμου με την «από πάνω προς τα κάτω προσέγγιση» (διάσπαση σε υποπροβλήματα)

7.6 Προγραμματίζοντας (βήματα)

7.6.4 Υλοποίηση

Συγγραφή του προγράμματος με χρήση κάποιας γλώσσας προγραμματισμού

7.6.5 Έλεγχος

Εκτέλεση και έλεγχος του προγράμματος για διάφορα σενάρια

7.6.6 Συντήρηση

Για να διορθωθούν λάθη και για να γίνουν αλλαγές που προέκυψαν λόγω αλλαγών στο περιβάλλον χρήσης του προγράμματος

7.6.7 Τεκμηρίωση

Σύνολο εγγράφων για την περιγραφή του αλγορίθμου (technical report) καθώς και σχόλια εντός του κώδικα

7.6 Προγραμματίζοντας (βήματα)
7.6.7 Τεκμηρίωση

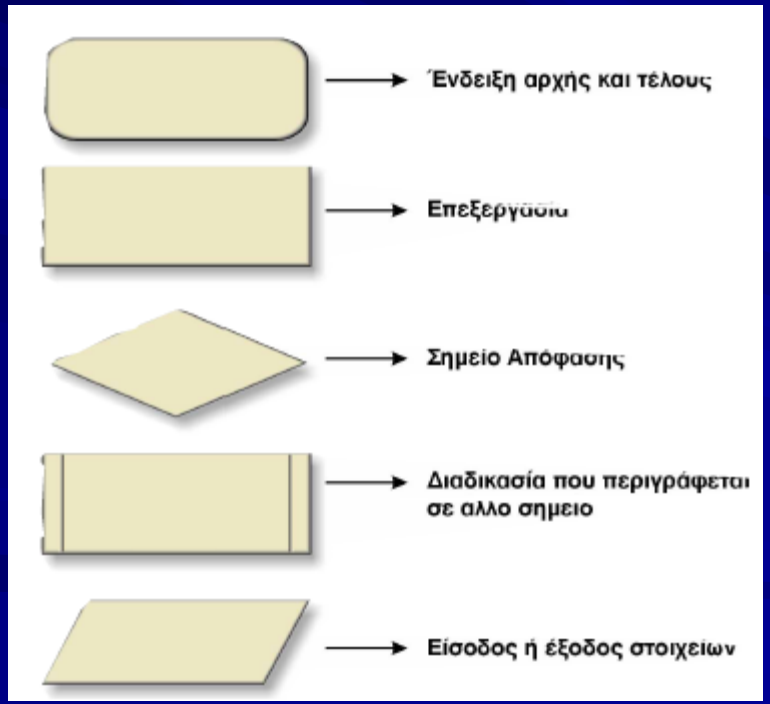
Τρόποι παράστασης αλγόριθμων

Οι αλγόριθμοι μπορούν να παρασταθούν στο χαρτί:

α) με φυσική γλώσσα: σαν απλό κείμενο

β) με ψευδοκώδικα: κάτι σαν γλώσσα υψηλού επιπέδου

γ) με λογικά διαγράμματα (διαγράμματα ροής): γραφική περιγραφή



7.7 Προγραμματιστικά περιβάλλοντα

Προγραμματιστικό περιβάλλον (IDE - integraded development environment): εργαλείο και περιβάλλον λογισμικού για την ανάπτυξη των προγραμμάτων.

Αποτελείται από εργαλεία όπως:

α) Ένα συντάκτη κειμένων, με τον οποίο γράφεται το πηγαίο πρόγραμμα.

β) Μεταφραστικά προγράμματα (assembler, compiler, interpreter).

γ) Εργαλεία εντοπισμού λαθών (debugger).

CASE (computer aided software engineering) **tools**: εργαλεία ανάπτυξης λογισμικού με τη βοήθεια υπολογιστή για την αυτόματη παραγωγή κώδικα μέσω γραφικού περιβάλλοντος.

Οπτικός προγραμματισμός (visual programming):τα αντικείμενα της γραφικής διεπαφής χρήστη κατασκευάζονται από παλέτες εργαλείων (toolbars)

7.8 Παράδειγμα ανάπτυξης προγράμματος

Επίλυση δευτεροβάθμιας

7.8.1 Καθορισμός προβλήματος

Να επιλύει εξισώσεις 2ου βαθμού, για οποιονδήποτε συνδυασμό των σταθερών όρων της

7.8.2 Ανάλυση προβλήματος

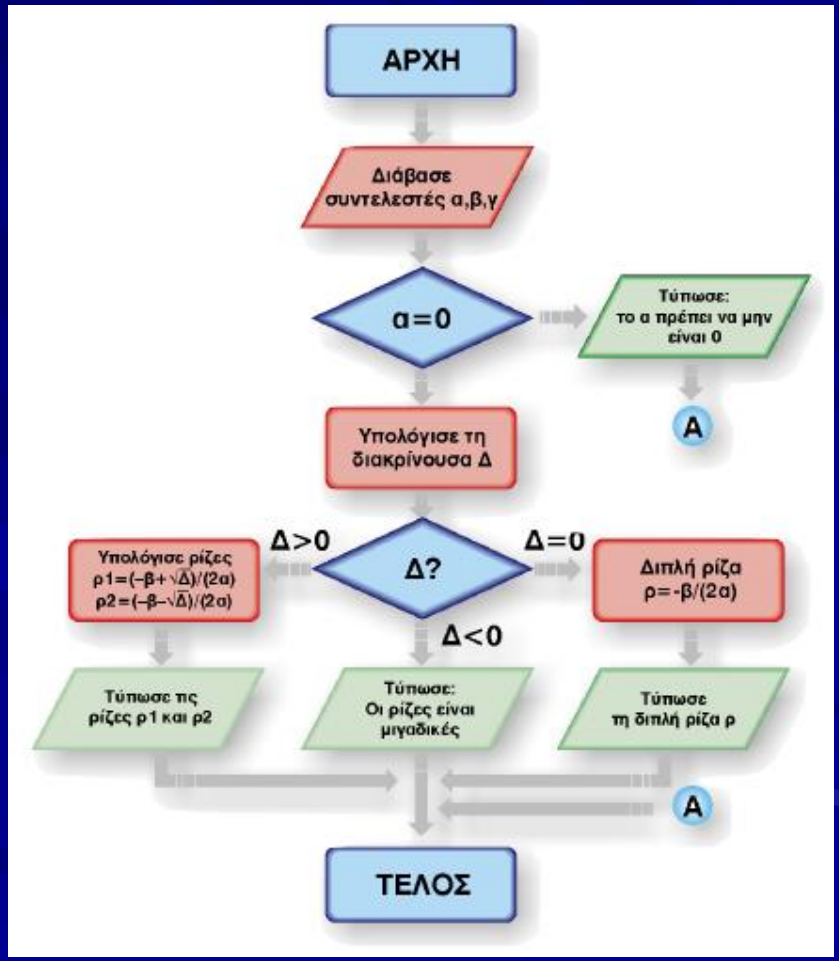
α) Το α μπορεί να είναι μηδέν; (τότε η εξίσωση εκφυλίζεται σε 1ου βαθμού).
β) Θα βρίσκουμε τις μιγαδικές ρίζες;

- Είσοδος: Οι συντελεστές α, β, γ
- Έξοδος: Οι ρίζες της εξίσωσης, μήνυμα για την περίπτωση των μιγαδικών ριζών, μήνυμα λάθους αν α=0
- Σχετικοί τύποι: διακρίνουσα κλπ.

7.8 Παράδειγμα ανάπτυξης προγράμματος

7.8.3 Σχεδιασμός αλγορίθμου

1. Διάβασε τις παραμέτρους της εξίσωσης
2. Έλεγξε την ορθότητα των παραμέτρων
   - 2.1 Αν α=0 τότε
     - Δώσε σχετικό μήνυμα λάθους
     - ΤΕΛΟΣ αλγόριθμου
3. Υπολόγισε τις ρίζες της εξίσωσης
   - 3.1 Υπολόγισε τη διακρίνουσα Δ
   - 3.2 Αν Δ>0 τότε
     - υπολόγισε τις δύο διαφορετικές ρίζες
     - τύπωσε τις ρίζες
   - 3.3 Αν Δ=0 τότε
     - υπολόγισε τη διπλή ρίζα
     - τύπωσε τη ρίζα
   - 3.4 Αν Δ<0 τότε
     - τύπωσε μήνυμα «οι ρίζες είναι μιγαδικές».
4. ΤΕΛΟΣ αλγόριθμου



7.8 Παράδειγμα ανάπτυξης προγράμματος

7.8.4 Υλοποίηση αλγορίθμου (Pascal)

```
program exis_2_vatmoy(input, output);
var
  a, b, c :real;
  diak    :real;
  r1, r2  :real;
begin
readln(a, b, c);
if a=0 then
  writeln('Το a πρέπει να είναι διάφορο από το μηδέν')
  writeln ('εξίσωση α΄ βαθμού')
else
  begin
  diak := b*b-4*a*c;
  if diak>0 then
    begin
    r1 := (-b+sqrt(diak))/(2*a);
    r2 := (-b-sqrt(diak))/(2*a);
    writeln('1η ρίζα:', r1:10:5);
    writeln('2η ρίζα:', r2:10:5)
    end
  else     if diak=0 then
      begin
        r1 := -b/(2*a);
        writeln('Διπλή ρίζα: ', r1:10:5)
      end
  else
    writeln('Οι ρίζες είναι μιγαδικές');
  end
end.
```

7.8 Παράδειγμα ανάπτυξης προγράμματος

7.8.5 Έλεγχος του προγράμματος

α/α	α	β	γ	Κατηγορία εξίσωσης
1	2	4	-5	δύο πραγματικές ρίζες
2	-2	-4	5	δύο πραγματικές ρίζες
3	1	6	9	διπλή ρίζα
4	1	0	0	ειδική περίπτωση (κάποιος συντελεστής 0)
5	1	1	0	ειδική περίπτωση (κάποιος συντελεστής 0)
6	1	0	1	ειδική περίπτωση (κάποιος συντελεστής 0) επίσης μιγαδικές ρίζες
7	0	1	1	εξίσωση α΄ βαθμού
8	2	1	5	μιγαδικές ρίζες

Ενδιαφέρουσες και χρήσιμες διευθύνσεις του Διαδικτύου

http://www.amzi.com

Εταιρεία που ειδικεύεται στην Prolog. Διαθέτει χωρίς κόστος μία καλή έκδοση της γλώσσας.

http://www.brain.uni-freiburg.de/~klaus/fpc/

Μεταφραστής Pascal για DOS και Linux.

http://www.delorie.com/djgpp/

Μεταφραστής C για DOS.

http://www.microsoft.com

Εκτός από τα γνωστά της λειτουργικά συστήματα, η Microsoft διαθέτει περιβάλλοντα ανάπτυξης για διάφορες γλώσσες όπως C, C++ και Basic.

http://www.inspise.com

Θα βρείτε πληροφορίες για τα πολύ επιτυχημένα περιβάλλοντα ανάπτυξης της Borland.

http://www.iecc.com/compilers/crenshaw/

Θα βρείτε γενικές οδηγίες για τον τρόπο κατασκευής ενός μεταγλωττιστή.

http://archie.inesc.pt/free-dir/free-toc.html

Κατάλογος από μη εμπορικούς μεταφραστές για διάφορες γλώσσες προγραμματισμού.

http://www.vlib.org/Computing.html

Γενικές πληροφορίες για τον προγραμματισμό.

http://www.pascal-central.com/

Καλό σημείο εκκίνησης για αναζήτηση πληροφοριών σχετικών με τη γλώσσα Pascal.

http://www.flamingolingo.com/programming.c/

Διαθέτει πολλές πληροφορίες για τη γλώσσα προγραμματισμού C.

http://www.nvg.ntnu.no/~sk/lang/lang.html

Εκτενής αλφαβητικός κατάλογος με γλώσσες προγραμματισμού.