Συστήματα Επιχειρηματικής Ευφυίας

CPLEX και OPL: ένα λογισμικό επίλυσης προβλημάτων Γραμμικού Προγραμματισμού

Προβλήματα Γραμμικού Προγραμματισμού

- Πρόβλημα **Γραμμικού Προγραμματισμού** - ΓΠ (Linear Programming – LP) λέγεται κάθε πρόβλημα μεγιστοποίησης ή ελαχιστοποίησης μιας (αντικειμενικής) συνάρτησης, με ταυτόχρονη ικανοποίηση κάποιων περιορισμών, όπου:
  - Κάθε περιορισμός μπορεί να μοντελοποιηθεί ως γραμμική εξίσωση ή ανίσωση
  - Η αντικειμενική συνάρτηση μπορεί να μοντελοποιηθεί ως γραμμικός συνδυασμός των μεταβλητών του προβλήματος.
- Πρόβλημα **Ακέραιου Προγραμματισμού** - ΑΠ (Integer Programming – IP) λέγεται κάθε πρόβλημα ΓΠ όπου επιπλέον κάποιες ή όλες από τις μεταβλητές της λύσης πρέπει να λαμβάνουν οπωσδήποτε Ακέραιες τιμές.
- Πρόβλημα **Δυαδικού Προγραμματισμού** – ΔΠ (Binary Programming – BP) λέγεται κάθε πρόβλημα ΓΠ όπου κάποιες ή όλες από τις μεταβλητές του μοντέλου λαμβάνουν την τιμή 0 ή 1.
- Πρόβλημα **Μεικτού Προγραμματισμού** λέγεται κάθε πρόβλημα ΓΠ όπου είναι ταυτόχρονα ΑΠ και ΔΠ.

Πολυπλοκότητα προβλημάτων ΓΠ

- **Κάθε πρόβλημα ΓΠ επιλύεται σε πολυωνυμικό χρόνο.**
- **Αντίθετα**, τα προβλήματα **Ακέραιου Προγραμματισμού και Δυαδικού Προγραμματισμού** (γενικά) **δεν** επιλύονται σε πολυωνυμικό χρόνο. Με άλλα λόγια **είναι NP-Complete** (Non polynomial Complete).
- **Χαλάρωση** ενός προβλήματος Ακέραιου ή Δυαδικού Προγραμματισμού λέγεται η μέθοδος κατά την οποία επιτρέπουμε σε κάποιες Ακέραιες μεταβλητές (ή και Δυαδικές) να λαμβάνουν Πραγματικές τιμές.
- Η μέθοδος της χαλάρωσης, κάποιες φορές είναι ευεργετική, καθώς ένα πρόβλημα Ακέραιου ή Δυαδικού Προγραμματισμού, που πρακτικά δεν μπορεί να επιλυθεί, τελικά επιλύεται σε αποδεκτό χρόνο.

Γιατί CPLEX; (1)

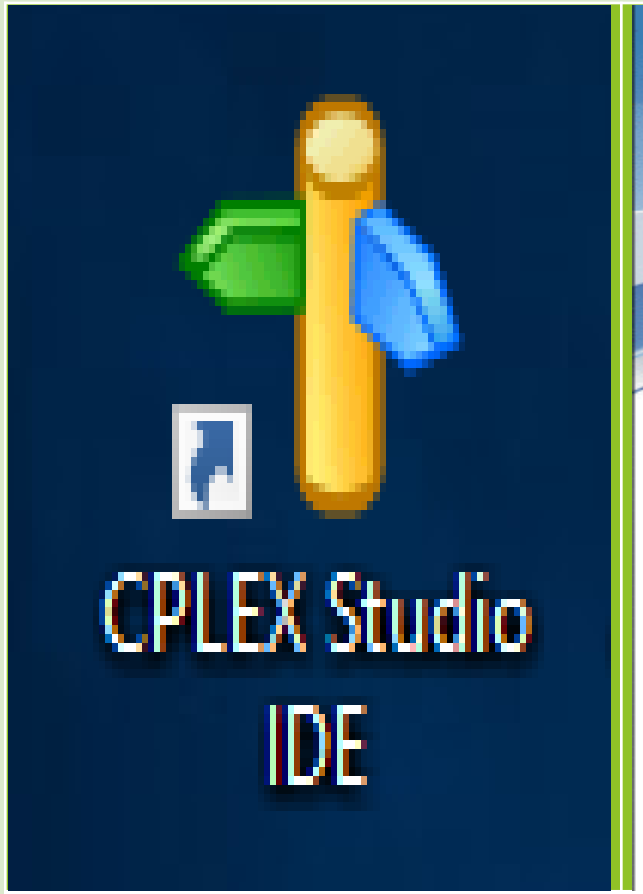
- Πολλά προβλήματα που σχετίζονται με τα Συστήματα Επιχειρηματικής Ευφυΐας μοντελοποιούνται ως προβλήματα Γραμμικού (ή Ακέραιου, ή Δυαδικού ή Μεικτού) Προγραμματισμού.
- Κάποια από τα προβλήματα που διαπραγματεύεται η Υπολογιστική Νοημοσύνη, επίσης μοντελοποιούνται με τον παραπάνω τρόπο. (π.χ. School timetabling, University Course Timetabling, Nurse Rostering κ.ά.)
- Αν και σε κάποιες «εύκολες» περιπτώσεις, τα προβλήματα ΓΠ μπορούν να επιλυθούν από τον άνθρωπο (π.χ. με γραφικό τρόπο – αναπαράσταση των ανισοτήτων στο Καρτεσιανό επίπεδο), γενικά χρησιμοποιούνται εξειδικευμένα Λογισμικά (εργαλεία) για αυτόν τον σκοπό και ΗΥ.

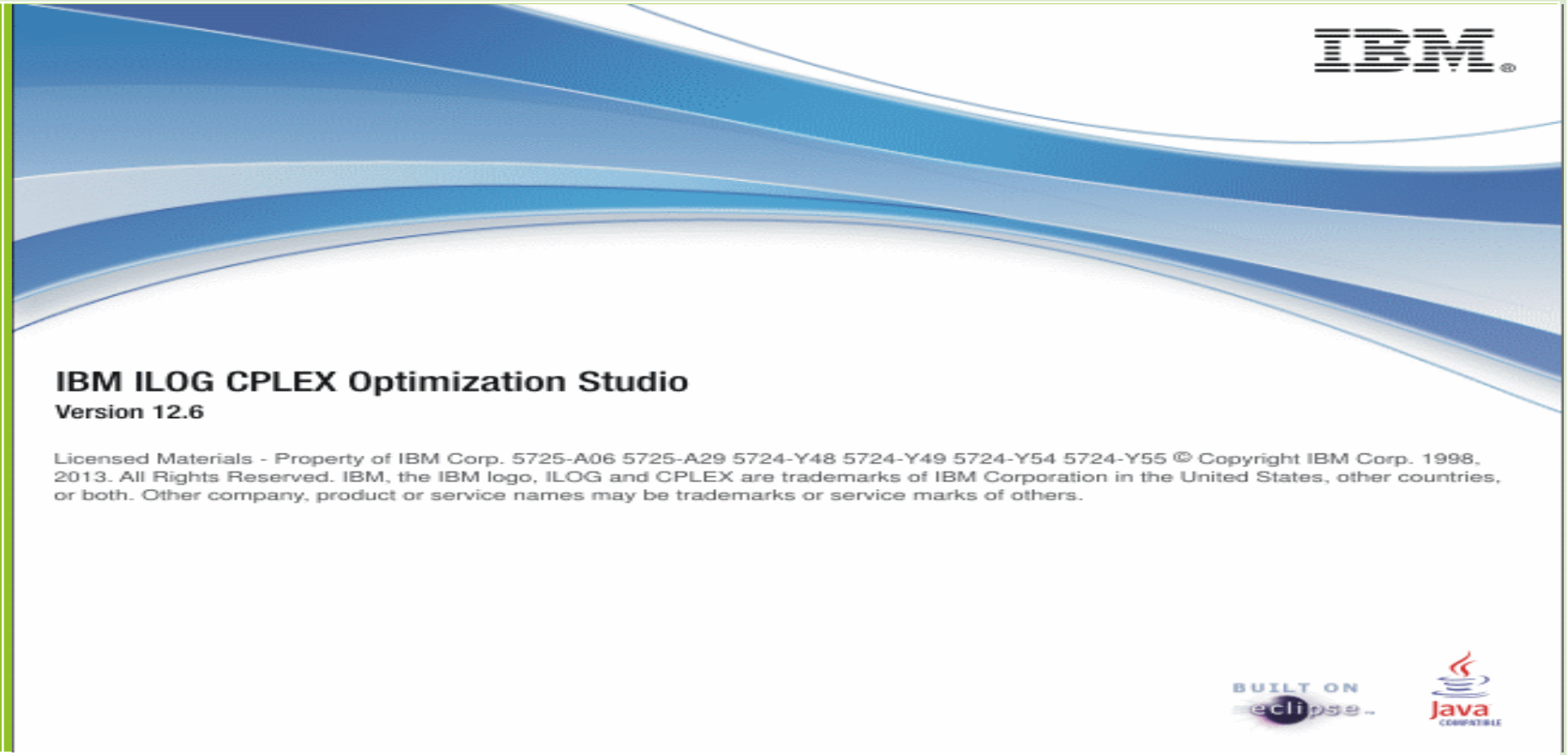
Γιατί CPLEX; (2)

- Υπάρχουν δωρεάν Λογισμικά (π.χ. Glpk) αλλά και εμπορικά (π.χ. CPLEX, Gurobi) τα οποία είναι γενικά καλύτερα από τα δωρεάν, αλλά κοστίζουν μερικές χιλιάδες δολάρια.
- Στην συνέχεια θα μελετήσουμε τα βασικά σημεία χρήσης του CPLEX, διότι από σχετική εργασία της ερευνητικής μας ομάδας (Τασσόπουλος, Μπεληγιάννης) που είναι υπό δημοσίευση, προέκυψαν καλύτερα αποτελέσματα του CPLEX έναντι των Glpk και Gurobi, τουλάχιστον για το πρόβλημα school timetabling. Έτσι, επιβεβαιώνεται η «φήμη» του CPLEX ως ένα από τα πιο ισχυρά Λογισμικά επίλυσης αυτού του είδους των προβλημάτων. Επιπλέον η εκμάθηση του CPLEX είναι σχετικά εύκολη.

Το λογότυπο του CPLEX

Κάνοντας κλικ στη συντόμευση του CPLEX (κάτω - αριστερά), στην επιφάνεια εργασίας, εμφανίζεται για λίγα δευτερόλεπτα η παρακάτω (δεξιά) εισαγωγική οθόνη (λογότυπο).

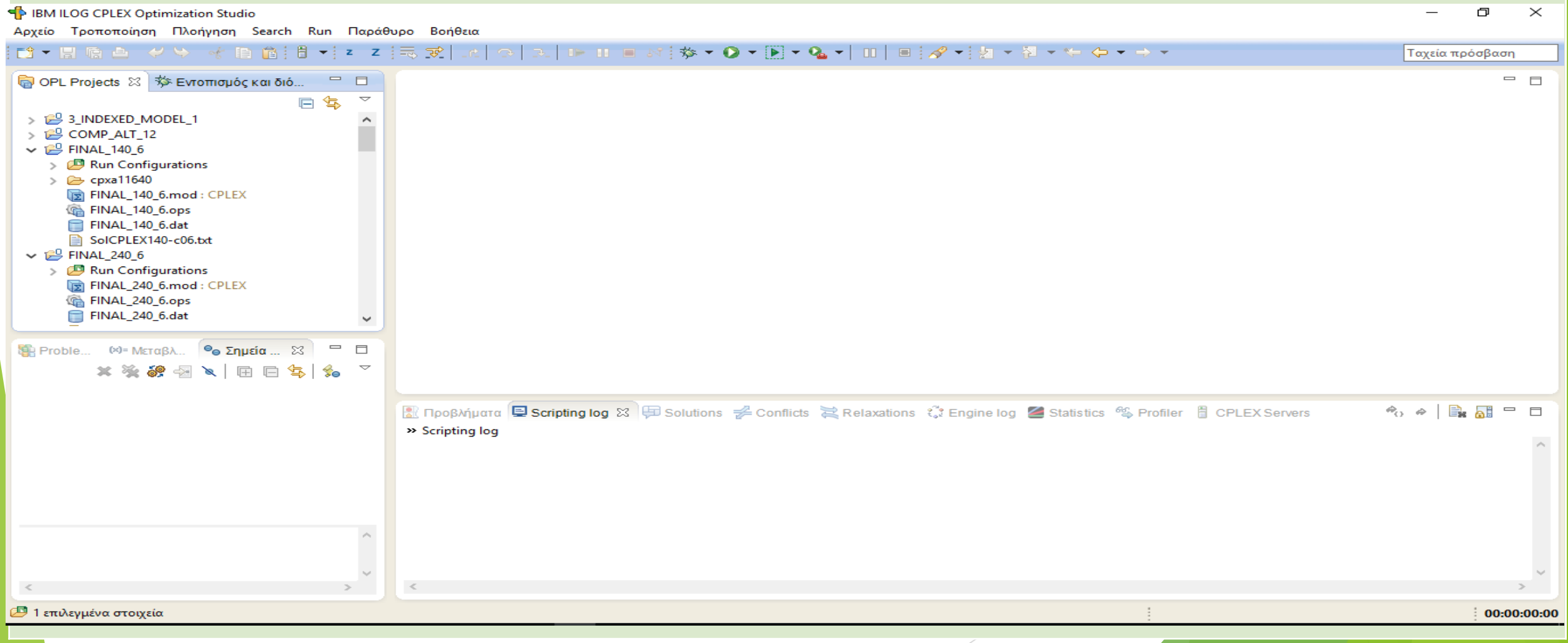




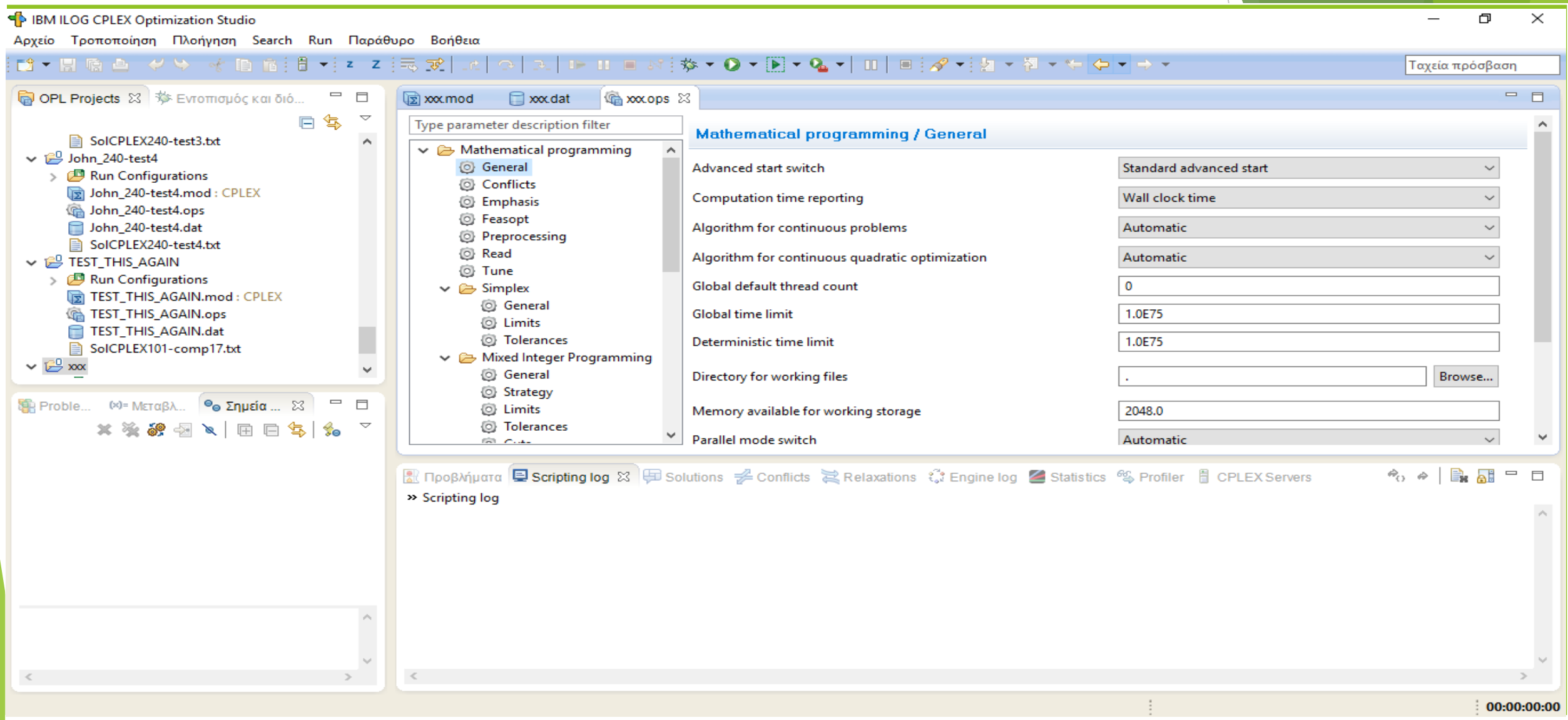
Η συντόμευση του CPLEX στην επιφάνεια εργασίας	Το λογότυπο του CPLEX.

Η διεπαφή (interface) του CPLEX

Πάνω αριστερά: κατάλογος των projects και των αρχείων τους. Πάνω κέντρο-δεξιά: εμφανίζονται τα περιεχόμενα των αρχείων του project. Κάτω κέντρο-δεξιά: εμφανίζονται πληροφορίες που αφορούν στην διάρκεια εκτέλεσης και την τελική λύση.



Το interface μετά τη δημιουργία του project με όνομα xxx.



Περισσότερες λεπτομέρειες για την δημιουργία project και το interface θα δοθούν στο προσεχές εργαστήριο.

Τα 2 αρχεία ενός project και το τμήμα .ops

- Αφού πρώτα δημιουργήσουμε ένα project για το πρόβλημα που επιχειρούμε να λύσουμε, δημιουργούνται 2 αρχεία: Name.mod και Name.dat. Name είναι το εκάστοτε όνομα του project.
- Name.mod: σε αυτό γράφουμε συνήθως μόνο το μοντέλο του προβλήματος, χρησιμοποιώντας ειδική γλώσσα (συντακτικό). Τα πραγματικά δεδομένα, συνήθως, δεν γράφονται εδώ. Γίνεται απλώς η δήλωση της δομής τους.
- Name.dat: εδώ γράφουμε τις τιμές των δεδομένων. Πρέπει να αντιστοιχούν στις δηλώσεις δεδομένων που έγιναν στο αρχείο Name.mod.

- Δημιουργείται επίσης ένα τμήμα πάνω στην επιφάνεια του interface, με όνομα Name.ops.
- Name.ops: περιέχει διάφορες επιλογές (options) και ρυθμίσεις, απαραίτητες για το τρέξιμο της εφαρμογής.

OPL: Optimization Programming Language

- Η γλώσσα OPL είναι μια γλώσσα μοντελοποίησης, την οποία “καταλαβαίνει” ο CPLEX προκειμένου να διαβάσει το μοντέλο του προβλήματος (αρχείο .mod) και τα δεδομένα (αρχείο .dat)
- Ο σκοπός της χρήσης της OPL είναι:
  - η προτυποποίηση στην διατύπωση ενός συγκεκριμένου μοντέλου.
  - Η ακρίβεια στην μαθηματική διατύπωση των μαθηματικών εκφράσεων.
  - Η απλότητα και η συντομία στην γραφή πολυάριθμων ανισοτήτων και εξισώσεων με λίγες μόνο γραμμές κώδικα.

Η βασική δομή ενός μοντέλου OPL (1)

- Τα σχόλια αρχίζουν με /* και τελειώνουν σε οποιαδήποτε γραμμή με */
- Επίσης, κάθε γραμμή που αρχίζει με // θεωρείται γραμμή σχολίων. Τα σχόλια αγνοούνται από τον CPLEX.
- Κάθε αρχείο .mod (στο οποίο περιγράφουμε το μοντέλο του προβλήματος) περιλαμβάνει τουλάχιστον 2 τμήματα: των δηλώσεων των μεταβλητών και της περιγραφής του μοντέλου.
- Όταν δηλώνεται μια μεταβλητή στο αρχείο .mod μπορούμε να της αποδώσουμε και τιμή. Αλλιώς, η τιμή μπορεί να αποδοθεί στο αρχείο .dat.
- Οι μεταβλητές απόφασης δηλώνονται με το πρόθεμα dvar (decision variable)
- Κάθε γραμμή, είτε στο .mod είτε στο .dat τελειώνει με ένα ;

Η βασική δομή ενός μοντέλου OPL (2)

Παράδειγμα:

```
/* This is an OPL model example.
   It is a very simple one                    */
//First there is the variable declaration part.
dvar float+ Gas;    //a positive real decision variable is
                          //declared. Its name is Gas.
dvar float+ Chloride; //another positive real decision
                          //variable with  name Chloride.
//Variable declaration part end.
```

Η Βασική δομή ενός μοντέλου OPL (3)

```
maximize  40 * Gas + 50 * Chloride;     //The objective function
subject to {          //The beginning of constraint part is declared
        ctMaxTotal: //The title of the first constraint.
                Gas + Chloride <= 50;
        ctMaxTotal2:  //The title of the second constraint
                3 * Gas + 4 * Chloride <= 180;
        ctMaxChloride:  //The title of the third constraint
                Chloride <= 40;
}//The title of a constraint is not obligatory but it is recommended
//for reference purposes. The title is followed by the constraint
//itself.
```

Κάποιοι βασικοί Τύποι δεδομένων της OPL

- **Ακέραιοι** (int X; ή int Y = 45; όπου X, Y παράμετροι ακέραιου τύπου)
  - Στην OPL παρέχεται το υποσύνολο των ακεραίων -maxint έως και maxint , όπου maxint ο μεγαλύτερος ακέραιος που μπορεί να αναπαρασταθεί.
- **Πραγματικοί** (float A; ή float B = 3.45; όπου A, B πραγματικές παράμετροι.)
- **Συμβολοσειρές** (string Name; ή string Name = “John”;)
  - Μπορούμε να δηλώσουμε ένα σύνολο από strings: {string} Names = {“John”, “George”, “Natasha”}

Βασικές δομές δεδομένων της OPL (1)

- **Διάστημα (range):**
  - **Ακέραια διαστήματα**

```
int lower = 10;
int upper = 20;
range X = lower .. 2*Upper;
range Y = -lower .. 10;
```

/*Ορίζονται δύο (μεταβλητές) διαστήματα X, Y, με **ακέραιες** τιμές οι οποίες μεταβάλλονται στα αντίστοιχα διαστήματα. */

Βασικές δομές δεδομένων της OPL (2)

- **Πραγματικά διαστήματα**

range float X = 1.0..100.0;

/*Ορίζεται ένα διάστημα από 1 έως 100

dvar float x in X;

/*Ορίζεται μια μεταβλητή απόφασης (decision variable-**dvar**) x **πραγματικού τύπου η** οποία παίρνει πραγματικές τιμές στο διάστημα [1.0, 100.0] */

Βασικές δομές δεδομένων της OPL (3)

- **One-dimensional arrays (μονοδιάστατοι πίνακες)**

int a[1..4] = [10, 20, 30, 40]; /* Ορίζεται ένας πίνακας μιας διάστασης με 4 θέσεις για ακεραίους. Οι ακέραιοι (περιεχόμενο) δίνονται την στιγμή της δήλωσης. Θα μπορούσαν να δοθούν στο αρχείο .dat */

float b[0..10]; /*Ορίζεται πίνακας πραγματικών 11 θέσεων. Οι 11 πραγματικοί δεν δίνονται. Θα δοθούν στο αρχείο .dat */

Βασικές δομές δεδομένων της OPL (4)

- Multi-dimensional arrays (πολυδιάστατοι πίνακες)

int NodesNo = ...; //ορίζεται μια παράμετρος ακέραιου τύπου.

//Η τιμή της δίνεται στο .dat ως : NodesNo = 4;

int TrucksNo = ...; //ορίζεται μια παράμετρος ακέραιου τύπου.

//Η τιμή της δίνεται στο .dat ως : TrucksNo = 5;

{int} C = asSet(1..NodesNo); /*ορίζεται ένα σύνολο ακεραίων, από το 0 έως το NodesNo. */

{int} W = asSet(1..TrucksNo); /* ορίζεται ένα σύνολο ακεραίων, από το 1 έως το TrucksNo. */

Βασικές δομές δεδομένων της OPL (5)

- **Multi-dimensional arrays (πολυδιάστατοι πίνακες)**

float TT[i in C][j in C] = ...;

/*ορίζεται ένας δυσδιάστατος πίνακας πραγματικών, με δείκτες i, j στα αντίστοιχα σύνολα. Οι τιμές του δίνονται στο .dat κατά γραμμές, ως:

TT = [[0, 1.2, 2.3, 1.1], [1.5, 0, 3.4, 1.3], [3.1, 1.2, 0, 1.2], [2.5, 6.7, 3.5, 0]]; */

Βασικές δομές δεδομένων της OPL (6)

- **Tuples (νιάδες)**

tuple triple **{int i; int j; int b;}**

/* ορίζεται μια νιάδα (tuple) με όνομα triple που στην πραγματικότητα είναι διατεταγμένη τριάδα, με μέλη τους ακεραίους i, j , b. */

Βασικές δομές δεδομένων της OPL (7)

- **Σύνολα με στοιχεία νιάδες.**

setof(triple) Triples = {<i,j, b> | i, j in C, b in W};
/* ορίζεται ένα σύνολο με όνομα Triples, όπου τα στοιχεία του είναι διατεταγμένες τριάδες τύπου triples. Εδώ ορίζεται και ο τύπος των μελών της τριάδας. */

Βασικές δομές δεδομένων της OPL (8)

- **Slicing (μερική άθροιση, ή άθροισμα ως προς ένα μέλος της νιάδας)**

setof(int) II = {i | <i,j,b> in Triples};

setof(int) JJ = {j | <i,j,b> in Triples};

setof(int) BB = {b | <i,j,b> in Triples};

/*ορίζονται 3 σύνολα ακεραίων (II, JJ και BB) με συγκεκριμένο τρόπο. Τα σύνολα αυτά είναι απαραίτητα για την **διαδικασία slicing στα αθροίσματα** (βλέπε επόμενες διαφάνειες). */

Μεταβλητές απόφασης (1)

- **Πρώτος τρόπος**
- {int} C = asSet(1..NodesNo);
- {int} W = asSet(1..TrucksNo);
- dvar boolean X[i in C][j in C][b in W] ;
- Αναφερόμαστε στην μεταβλητή ως: X[i][j][b]

/*Ορίζεται μια δυαδική (boolean) μεταβλητή απόφασης τριών διαστάσεων.

Εναλλακτικά, μπορούμε να ορίσουμε την ίδια μεταβλητή απόφασης όπως παρακάτω */

Μεταβλητές απόφασης (2)

- **Δεύτερος τρόπος**
- tuple triple {int i; int j; int b;}
- setof(triple) Triples = {<i,j, b> | i, j in C, b in W};
- dvar boolean X[Triples];
- Αναφερόμαστε στην μεταβλητή ως X[<i, j, b>}

Μερικά αθροίσματα με πολυδιάστατες μεταβλητές (1)

- Έστω η δυαδική μεταβλητή X[i][j][b] με τιμή 1 αν το φορτηγό b μεταβαίνει από τον κόμβο i στον j και τιμή 0 αλλιώς. Υποθέτουμε ότι θέλουμε να εξασφαλίσουμε με κάποιον περιορισμό ότι σε κάθε κόμβο j θα υπάρχει το πολύ ένας κόμβος i ώστε κάθε φορτηγό b να μεταβαίνει από τον i στον j. Ο κατάλληλος περιορισμός γράφεται:

```
forall(b in W, j in C)     //για κάθε φορτηγό b και για
                           //κάθε κόμβο j
  sum(i in C)X[i][j][b] <= 1;    //μερική άθροιση (μερικό
                                 //άθροισμα) ως προς i.
```

Μερικά αθροίσματα με πολυδιάστατες μεταβλητές (2)

Ερωτήσεις

- Τι σημαίνουν οι περιορισμοί:

  forall(b in W, j in C) **sum**(i in C)X[i][j][b] == 1;

- Πώς θα μοντελοποιούσαμε την απαίτηση: από κάθε κόμβο i, κάθε φορτηγό εξέρχεται προς έναν ακριβώς κόμβο j και πως θα μοντελοποιούσαμε την απαίτηση: από κάθε κόμβο i, κάθε φορτηγό εξέρχεται το πολύ μια φορά προς κόμβο j;
- Να μοντελοποιηθεί η απαίτηση: αν ένα οποιοδήποτε φορτηγό εισέλθει στον κόμβο 4 τότε δεν θα εξέλθει.

Slicing (1)

- Αν αντί για την προηγούμενη αναπαράσταση της δυαδικής μεταβλητής X χρησιμοποιήσουμε την ανάλογη αναπαράσταση με τις νιάδες (δηλαδή την X[<i, j, b>] και θέλουμε να γράψουμε τον περιορισμό της προηγούμενης διαφάνειας, δηλαδή να κάνουμε μερική άθροιση ως προς ένα μέλος της διατεταγμένης τριάδας, τότε πρέπει να έχουμε ήδη δηλώσει ένα σύνολο:

  setof(int) **II={i | <i,j,b> in Triples};**

  και μετά να γράψουμε τον περιορισμό ως:

  forall(b in W)

  sum(<i,j,b>in Triples**: i in II**)X[<i,j,b>] <= 1;

Slicing (2)

- Είναι δυνατόν να προσθέσουμε επιπλέον περιορισμούς στην έκφραση **forall** ή / και στο **sum** ενός περιορισμού.

forall(b in W : b != 1) //για κάθε φορτηγό διαφορετικό από το 1

sum(<i,j,b>in Triples**: i in II && (j == 3 || j >= 6)**)X[<i,j,b>] <= 1;

//μερική άθροιση ως προς i, για j = 3 ή j >= 6

Ερώτηση

Πώς ερμηνεύονται οι παραπάνω περιορισμοί;

Πολλαπλά αθροίσματα (1)

- **Ερώτηση:**

  Να μοντελοποιηθεί η απαίτηση: τα φορτηγά που εισέρχονται σε οποιονδήποτε κόμβο από οποιονδήποτε κόμβο είναι το πολύ 5.

- **Απάντηση:**

```
forall(j in C)
    sum(i in C, b in W)X[i][j][b] <= 5;
```

Πολλαπλά αθροίσματα (2)

- **Ερώτηση:**

  Να μοντελοποιηθεί η απαίτηση: τα φορτηγά που εισέρχονται σε οποιονδήποτε κόμβο από οποιονδήποτε κόμβο είναι ακριβώς 5.

- **Απάντηση:**

  (a) **forall**(b in W, j in C)

  sum(i in C)X[i][j][b] <= 1;

  (b) **forall**(j in C)

  sum(i in C, b in W)X[i][j][b] == 5;

Πολλαπλά αθροίσματα (3)

- **Ερώτηση:**

  Στην απάντηση της προηγούμενης διαφάνειας, ποια είναι η σκοπιμότητα των περιορισμών (a);

  Να αιτιολογήσετε αν είναι απαραίτητοι ή όχι.

Το block της main

- Χρησιμοποιείται για να ελέγξει την ροή εκτέλεσης του CPLEX.
- Μπορεί να υπάρχει το πολύ ένα main block στο αρχείο .mod
- Ανεξάρτητα από το σημείο που αναγράφεται στο αρχείο .mod εκτελείται κατά προτεραιότητα.
- Οι ρυθμίσεις που ενδεχομένως υπάρχουν στην main υπερισχύουν των ρυθμίσεων του CPLEX που έχουν γίνει στο αρχείο .ops και αφορούν το περιβάλλον IDE.
- Η μορφή ενός main block είναι: main { ……. }

Παράδειγμα ενός main block

```
main {

thisOplModel.generate();
        if (cplex.solve()) {
          var obj=cplex.getObjValue();
        }
}
```

Η δήλωση thisOplModel είναι μια ειδική μεταβλητή του script που αναφέρεται στο συγκεκριμένο instance (στιγμιότυπο)του μοντέλου.

Η geneate() είναι μια μέθοδος η οποία δημιουργεί το στιγμιότυπο του μοντέλου.

Η cplex είναι μια μεταβλητή του script που αναφέρεται εξ ορισμού στο CPLEX Optimizer instance.
Η solve() καλεί έναν αλγόριθμο βελτιστοποίησης του CPLEX.
Η gerObjValue() αντλεί την τιμή της αντικειμενικής συνάρτησης μετά την επίλυση του μοντέλου.

Παράδειγμα script που καλεί ένα project

```
main {
        var proj = new
IloOplProject("../../../../../opl/mulprod");
          var rc = proj.makeRunConfiguration();
          rc.oplModel.generate();
         if (rc.cplex.solve()) {
            writeln("OBJ = ", rc.cplex.getObjValue());
         }
        else {
             writeln("No solution");
        }
        rc.end();
        proj.end();
}
```

Καλείται το project mulprod.prj
Δημιουργείται το configuration

Δημιουργείται το μοντέλο στην μεταβλητή rc.

Επιχειρείται η επίλυση του μοντέλου.

Κλείνει το μοντέλο και το project

Προ-επεξεργασία (Preprocessing)

- Η προ-επεξεργασία (preprocessing) επιτυγχάνεται με μια ή περισσότερες δομές execute { ……} που πρέπει να αναγράφονται στο αρχείο .mod πριν την δήλωση minimize ή maximize
- Η προ-επεξεργασία, αν υπάρχει, εκτελείται εξ ορισμού πριν την επίλυση του μοντέλου και σκοπό έχει την προετοιμασία των δεδομένων.

Μετά – επεξεργασία (Post processing) (1)

- Η μετά-επεξεργασία (post processing) επιτυγχάνεται με μια ή περισσότερες δομές **execute { ……}** που πρέπει να αναγράφονται στο αρχείο .mod μετά το τμήμα δήλωσης των περιορισμών, δηλαδή μετά το

  subject to { ….. }

- Ο σκοπός της χρήσης της μετά-επεξεργασίας είναι η εκτύπωση των αποτελεσμάτων και διάφορων πληροφοριών.

Μετά – επεξεργασία (Post processing) (2)

- Προφανώς η μετά-επεξεργασία λαμβάνει χώρα μετά την επίλυση του μοντέλου.
- Η μετά-επεξεργασία δεν εκτελείται εξ ορισμού, αλλά αντίθετα χρειάζεται η εντολή **thisOplModel.postProcess();** για να ενεργοποιηθεί, η οποία αναγράφεται στην main { …. }

Προγραμματισμός στην execute και την main

- Οι προγραμματιστικές εντολές που χρησιμοποιούνται στην execute και την main μοιάζουν πολύ με της γλώσσας C.
- Υπάρχει για παράδειγμα η δομή **while** { …}, η δομή **if**( ) **then** { ….} **else** { ….}, η δομή **for**(var i in C){ …. }, η εντολή **break**, κ.ά.

Εκτύπωση στην οθόνη του IDE του CPLEX

- Επιτυγχάνεται γράφοντας σε execute block ή στην main την εντολή **writeln()**;

Παράδειγμα: **writeln**(“objective value is “, obj);

/*θα εκτυπώσει objective value is και στην συνέχεια θα εκτυπώσει την τιμή της μεταβλητής obj. Ακολούθως θα αλλάξει γραμμή. */

Χειρισμός αρχείων και εγγραφή σε αρχείο

var fp = new IloOplOutputFile("Results.txt");	Δημιουργείται ένας «δείκτης» fp προς ένα αρχείο εξόδου με όνομα Results.txt
fp.writeln("This is a text");	Γράφεται στο αρχείο το κείμενο: This is a text και αλλάζει γραμμή.
fp.writeln("Obj value is ", obj);	Γράφεται στο αρχείο το κείμενο: Obj value is, στην συνέχεια γράφεται η τιμή της μεταβλητής obj και αλλάζει γραμμή.
fp.close();	Κλείνει το αρχείο που δείχνει ο δείκτης fp, δηλαδή το Results.txt

Σας ευχαριστώ για την προσοχή σας