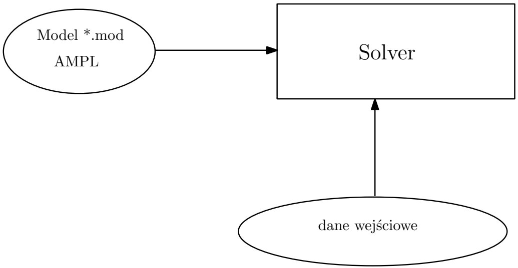
AMPL

AMPL (*A Mathematical Programming Language*) jest jednym z języków służących do zapisywania problemów optymalizacyjnych (idea innych języków jest podobna). Aktualnie pełną wersję z darmowymi solverami można pobrać tutaj: AMPL IDE



Darmowe solvery: HiGHS, CBC, Ipopt. Komercyjne solvery (ograniczony rozmiar modelu): CPLEX, GUROBI

Prosty przykład

$$
\begin{array}{ll}
\min & 3x_1 + 2x_2 + 3x_3 \\
 & 2x_1 + 4.5x_2 + 2x_3 \leq 120 \\
 & 2x_1 + 3x_2 - x_3 \geq 100 \\
 & x_1, x_3 \geq 0 \\
 & x_2 \geq 0 \text{ integer}
\end{array}
$$

Prosty przykład

```
#choose solver
option solver cplex;
#start new model (clear AMPL memory)
reset;
#decision variables
var x1 >=0;
var x2 >=0, integer;
var x3 >=0;
#objective function
minimize cost: 3*x1+2*x2+3*x3;
#constraints
subject to
c1: 2*x1+4.5*x2+2*x3<=120;
c2: 2*x1+3*x2-x3>=100;
#solve the model and display the results
solve;
display x1, x2, x3, cost;
end;
```

Apex TV

Przedsiębiorstwo Apex TV musi zadecydować ile telewizorów 27 i 20 calowych produkować w najbliższym miesiącu w jednej ze swoich fabryk. Badania rynkowe wskazują, że co najwyżej 20 telewizorów 27 calowych i 40 telewizorów 20 calowych może być sprzedanych w ciągu jednego miesiąca. Liczba dostępnych roboczogodzin w jednym miesiącu wynosi 500. Wyprodukowanie jednej sztuki telewizora 27 calowego wymaga 20 roboczogodzin a 20 calowego 10 roboczogodzin. Zysk z jednego telewizora 27 calowego wynosi $120\$$ a z jednego telewizora 20 calowego odpowiednio $80\$$. Firma chce zmaksymalizować całkowity miesięczny zysk.

$$
\begin{array}{ll}
\max & 120x_1 + 80x_2 \\
 & x_1 \leq 20 \\
 & x_2 \leq 40 \\
 & 20x_1 + 10x_2 \leq 500 \\
 & x_1, x_2 \geq 0
\end{array}
$$

Apex TV

```
#choose solver
option solver cplex;
#start new model (clear AMPL memory)
reset;
#decision variables
var x1 >=0; # 27-ich sets
var x2 >=0; # 20-ich sets
#objective function
maximize profit: 120*x1+80*x2;
#constraints
subject to
c1: x1<=20;
c2: x2<=40;
c3: 20*x1+10*x2<=500;
#solve the model and display the results
solve;
display x1, x2, profit;
end;
```

Ogólny model

Budując model w AMPL zawsze staramy się oddzielić dane od modelu. Co się stanie jeżeli dodamy kolejny rodzaj telewizora?

Model musi być uniwersalny i działać poprawnie dla dowolnych poprawnych danych.

Dane mogą być dostarczane w oddzielnych plikach tekstowych. Model zawsze jest taki sam.

Apex TV - ogólny model

Niech n będzie liczbą możliwych typów TV. Popyt, jednostkowy zysk oraz wymagania praca dla typu i-tego wynoszą odpowiednio d_i, p_i i h_i. Liczba dostępnych godzin pracy wynosi h_limit.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} p_i x_i & \\
& \sum_{i=1}^{n} h_i x_i \leq h\_limit & \\
& x_i \leq d_i & i = 1, \ldots, n \\
& x_i \geq 0 & i = 1, \ldots, n
\end{aligned}
$$

Apex TV - model ogólny

```
param n;
param p{1..n};
param d{1..n};
param h{1..n};
param h_limit;
```

$\max \sum_{i=1}^{n} p_i x_i$	`maximize profit: sum{i in 1..n} p[i]*x[i];`
$\sum_{i=1}^{n} h_i x_i \leq h\_limit$	`workload: sum{i in 1..n} h[i]*x[i]<=h_limit;`
$x_i \leq d_i \;\; i = 1, \ldots, n$	`demand{i in 1..n}: x[i]<=d[i];`
$x_i \geq 0 \;\; i = 1, \ldots, n$	`var x{1..n}>=0;`

Apex TV - model ogólny

```
option solver cplex;
reset;
#declaration of parameters
param n;
param p{1..n};
param d{1..n};
param h{1..n};
param h_limit;
#decision variables
var x{1..n} >=0;
#objective function
maximize profit: sum{i in 1..n} p[i]*x[i];
#constraints
subject to
workload: sum{i in 1..n} h[i]*x[i]<=h_limit;
demand{i in 1..n}: x[i]<=d[i];
```

Apex TV - dane wejściowe

```
#provide data
data;
param n:=2;
param p:= [1] 120 [2] 80;
param d:= [1] 20 [2] 40;
param h:= [1] 20 [2] 10;
param h_limit:=500;
#see the model
expand;
#solve the model and display the results
solve;
display x, profit;
end;
```

Apex TV - model ogólny

AMPL powala przedstawić model w bardziej czytelnej postaci za pomocą zbiorów. Załóżmy, że możliwe typy TV należą do zbioru $\boldsymbol{T}$.

$$
\begin{aligned}
\max \quad & \sum_{i \in \boldsymbol{T}} p_i x_i & \\
& x_i \leq d_i & i \in \boldsymbol{T} \\
& \sum_{i \in \boldsymbol{T}} h_i x_i \leq h\_limit & \\
& x_i \geq 0 & i \in \boldsymbol{T}
\end{aligned}
$$

Apex TV - model ogólny

```
set T;
param p{T};
param d{T};
param h{T};
param h_limit;
```

$\max \sum_{i \in T} p_i x_i$	`maximize profit: sum{i in T} p[i]*x[i];`
$\sum_{i \in T} h_i x_i \leq h\_limit$	`workload: sum{i in T} h[i]*x[i]<=h_limit;`
$x_i \leq d_i \;\; i \in T$	`demand{i in T}: x[i]<=d[i];`
$x_i \geq 0 \;\; i \in T$	`var x{T}>=0;`

Apex TV - model ogólny

```
option solver cplex;
reset;
#declaration of parameters
set T;
param p{T};
param d{T};
param h{T};
param h_limit;
#decision variables
var x{T} >=0;
#objective function
maximize profit: sum{i in T} p[i]*x[i];
#constraints
subject to
workload: sum{i in T} h[i]*x[i]<=h_limit;
demand{i in T}: x[i]<=d[i];
```

Apex TV - dane wejściowe

```
#provide data
data;
param: T: p, d, h:= '27-inch' 120, 20, 20
                     '20-ich' 80, 40, 10;
param h_limit:=500;
#see the model
expand;
#solve the model and display the results
solve;
display x, profit;
end;
```

Problem diety (przykład)

Dieta ma być zestawiona z czterech produktów: chleba, mleka, sera i jogurtu. Koszty jednostkowe oraz zawartości składników odżywczych w jednostce produktu podane są w poniższej tabeli:

	Chleb	Mleko	Ser	Jogurt
Koszt jedn. $	1.0	2.5	3.0	4.0
Cukier, g./jedn.	0.5	1	0.2	4
Tłuszcz, g./jedn.	0	5.0	9.0	7.0
Białko, g./jedn.	4.0	11.7	10.0	17.0
Kalorie, cal./jedn.	90	120	106	110

Celem jest ustalenie najtańszej diety zawierającej co najmniej 300 kalorii, 10 g. cukru, 6 g. tłuszczu i 30 g. białka.

Problem diety - model dla przykładu

$$
\begin{array}{ll}
\min & x_1 + 2.5x_2 + 3x_4 + x_4 \\
& 0.5x_1 + x_2 + 0.2x_3 + 4x_3 \geq 10 \\
& 5x_2 + 9x_3 + 7x_4 \geq 6 \\
& 4x_1 + 11.7x_2 + 10x_3 + 17x_4 \geq 30 \\
& 90x_1 + 120x_2 + 106x_3 + 110x_4 \geq 300 \\
& x_1, x_2, x_3, x_4 \geq 0
\end{array}
$$

Problem diety - model ogólny

Dieta ma być zestawiona z produktów należących do zbioru P. Koszt jednostkowy produktu $j \in P$ wynosi c_j. Produkt $j \in P$ zawiera a_{ij} jednostek składnika i należącego do zbioru S. Dieta musi zawierać co najmniej b_i składnika $i \in S$. Należy wyznaczyć najtańszą dietę spełniającą wymagania żywieniowe.

$$
\begin{array}{lll}
\min & \sum_{j \in P} c_j x_j & \\
& \sum_{j \in P} a_{ij} x_j \geq b_i & i \in S \\
& x_j \geq 0 & j \in P
\end{array}
$$

Problem diety - model ogólny

```
set P;
set S;
param c{P};
param b{S};
param a{S,P};
```

$\min \sum_{j \in P} c_j x_j$	`minimize cost: sum{j in P} c[j]*x[j];`
$\sum_{j \in P} a_{ij} x_j \geq b_i \;\; i \in S$	`comp{i in S}: sum{j in P} a[i,j]*x[j]>=b[i];`
$x_j \geq 0 \;\; j \in P$	`var x{P}>=0;`

Problem diety - model ogólny

```
option solver cplex;
reset;
#declaration of parameters
set P;
set S;
param c{P};
param b{S};
param a{S,P};
#decision variables
var x{P} >=0;
#objective function
minimize cost: sum{j in P} c[j]*x[j];
#constraints
subject to
comp{i in S}: sum{j in P} a[i,j]*x[j]>=b[i];
```

Problem diety - dane wejściowe

```
data;
param: P: c:= 'Chleb' 1
              'Mleko' 2.5
              'Ser' 3
              'Jogurt' 4;

param: S: b:= 'Cukier' 10
              'Tluszcz' 6
              'Bialko' 30
              'Kalorie' 300;

param a:    'Chleb' 'Mleko' 'Ser' 'Jogurt' :=
'Cukier'      0.50   1.00   0.20   4.00
'Tluszcz'      0.00   5.00    9.00   7.00
'Bialko'      4.00   11.7    10.0   17.0
'Kalorie'      90     120    106    110;
```

Problem transportowy

Firma musi dostarczyć towar z fabryk należących do zbioru F do sklepów należących do zbioru S. Fabryka $i \in F$ może wysłać s_i jednostek towaru a sklep $j \in S$ potrzebuje d_j jednostek towaru. Jednostkowy koszt transportu od fabryki i do sklepu j wynosi c_{ij}. Należy wyznaczyć najtańszy plan transportu towaru z fabryk do sklepów.

$$
\begin{array}{lll}
\min & \sum_{i\in F}\sum_{j\in S} c_{ij}x_{ij} & \\
& \sum_{j\in S} x_{ij} \le s_i & i \in F \\
& \sum_{i\in F} x_{ij} \ge d_j & j \in S \\
& x_{ij} \ge 0 & i \in F, j \in S
\end{array}
$$

Problem transportowy

```
set F;
set S;
param s{F};
param d{S};
param c{F,S};
```

$\min \sum_{i \in F} \sum_{j \in S} c_{ij} x_{ij}$	`minimize cost: sum{i in F, j in S} c[i,j]*x[i,j];`
$\sum_{j \in S} x_{ij} \leq s_i \; i \in F$	`sp{i in F}: sum{j in S} x[i,j]<=s[i];`
$\sum_{i \in F} x_{ij} \geq d_j \; j \in S$	`dem{j in S}: sum{i in F} x[i,j]>=d[j];`
$x_{ij} \geq 0 \;\; i \in F, j \in S$	`var x{F,S}>=0;`

Zapisz cały model i podaj przykładowe dane.

Problem transportowy - prezentacja wyniku

Polecenie **display** x pokazuje wynik w niezbyt czytelny sposób. Można użyć następującego kodu:

```
for{i in F, j in S}
      if x[i,j]>0 then
         print 'Fabryka', i, 'do sklepu', j, ' ', x[i,j];
```