OPL

OPTIMIZATION PROGRAMMING LANGUAGE

Luiz Henrique Cherri

O que é? Por que usar?

- Linguagem de programação
- Desenvolvida pela IBM ILOG

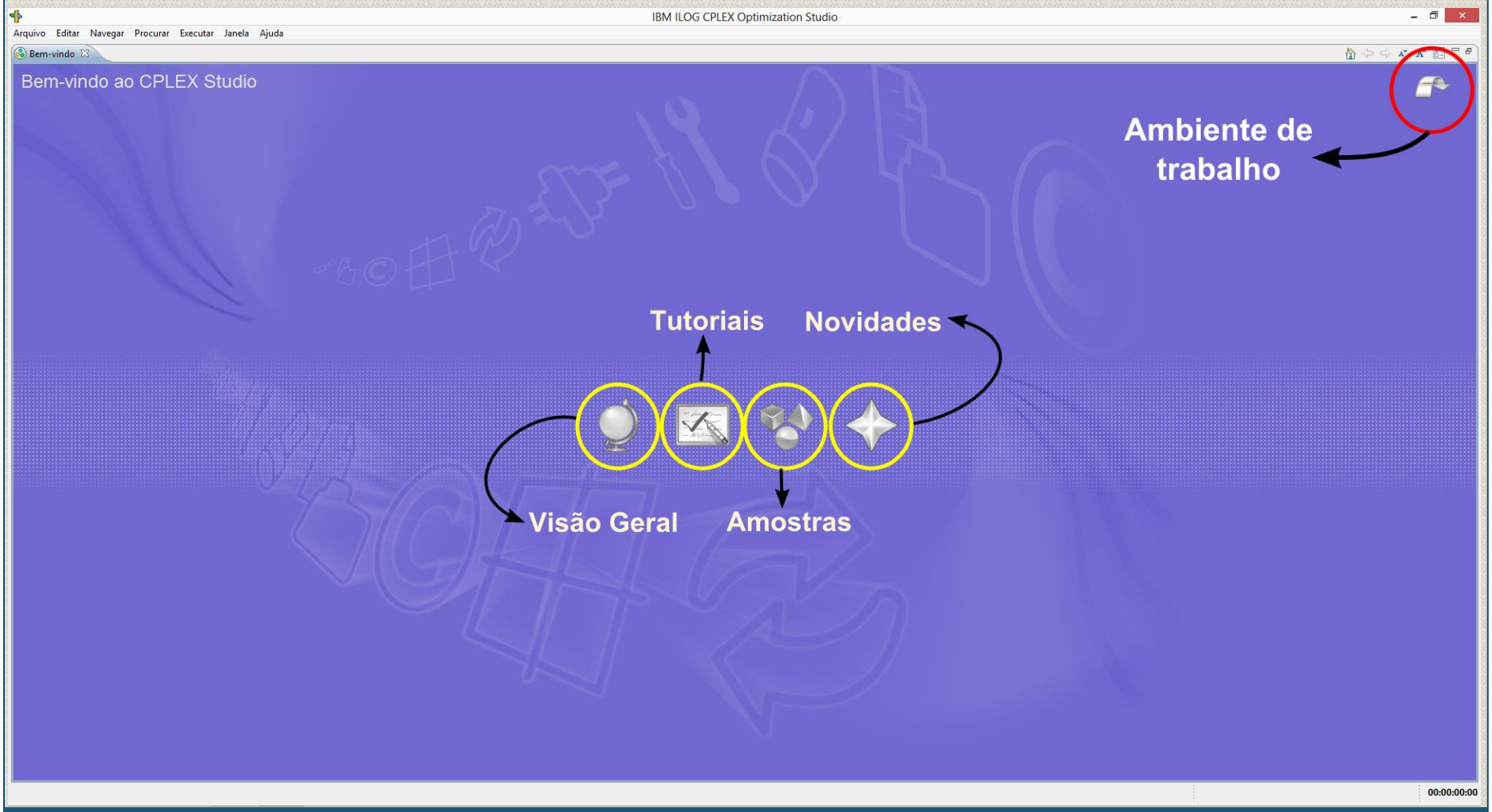
- Independe de outras linguagens de programação
- Implementação de modelos é fácil
- Possibilita a implementação de rotinas complexas

Como instalar?

- A instalação é simples ...

- O OPL não é um software livre!
  - A IBM disponibiliza licenças grátis para a universidade
  - O uso dessas licenças deve ser estritamente acadêmico

Instalação concluída... E agora?

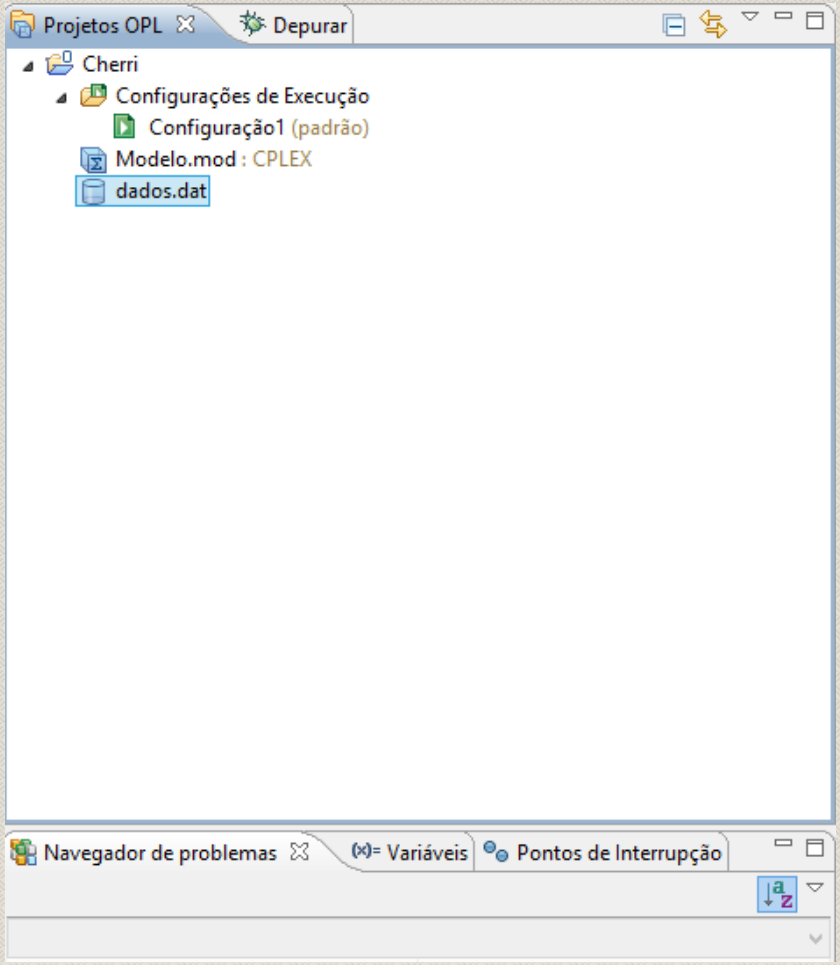


Iniciando um projeto

- Criar um novo projeto
- Incluir no projeto as componentes:
  - Modelo – Onde o modelo será implementado
  - Dados – Os quais serão usados para a criação do modelo
  - Execução – Componente que integra o modelo aos dados
  - Configurações – Permite alterar facilmente alguns parâmetros importantes
- Tudo isso pode ser feito no menu superior, em "Arquivo"

Iniciando um projeto

- Após criado o projeto e seus componentes a aba “Projeto OPL” terá a seguinte cara:



Tipos de dados

- float: Declara um número real
  - Ex: float valor1;
- int: Declara um número inteiro
  - Ex: int valor2;
- range: Declara uma sequencia de números a serem iterados
  - Ex: range it = 1..valor2;
- E para declarar um vetor?
  - → int vet[it];

Organização do código

- Declaração variáveis e leitura de dados
- Função objetivo
- Restrições
- Funções de impressão

Declarando variáveis

- É só utilizar o identificador dvar antes do tipo!
  - Exemplos:
    - dvar float custo_inicial;
    - range N = 1..n;
    - dvar int vetor[N];
    - dvar float matriz[N][N];

Lendo de arquivos

- Imagine que você possui um arquivo (.dat) com as seguintes informações:

```
N = 5;
Demanda = [20, 45, 50, 55, 75];
```

- Para fazer a leitura dessas variáveis somente é necessária a seguinte sintaxe:
  - int N = ...; ← Vai receber o valor 5 do arquivo
  - range it = 1..N; ← Cria o iterador;
  - int Demanda[it] = ...; ← Faz a leitura do vetor no arquivo

Programando restrições

- $\sum_{i=1}^{100} x_i \leq 1000$
  - range N = 1 , .. , 100;
  - sum (i in N) x[i] <= 1000;

- $x_i \geq 0, \forall i = 1, \ldots, 100$
  - range N = 1 , .. , 100;
  - forall (i in N)
  - x[i] >= 0

- Não há problemas em combinar expressões!

Arquivo de configuração

- É uma maneira fácil para ajuste de parâmetros como:
  - Tempo limite de execução
  - Foco da busca
  - Geração do LP
- Não é necessário para executar o algoritmo!

Comandos adicionais

- Com a tag

  execute <NOME> {

  }

- É possível utilizar as funções disponíveis para C++ e Java.

- A programação é muito similar ao Java!
- Pode ser muito útil para impressão de soluções

Exemplo para fixação!

```
dvar int x;
dvar int y;

minimize x - y;

subject to{
     x >= 0;
     y >= 0;
     -x+2*y <= 15;
     x <= 10;
     y <= 10;
}
```

Exercício

Minimizar: $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} * x_{ij}$

Sujeito a:

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1, \dots, n$$

$$\sum_{j=1}^{n} r_{ij} * x_{ij} \leq b_i \qquad i = 1, \dots, m$$

$$x_{ij} \in \{0,1\} \qquad i = 1, \dots, m, \qquad j = 1, \dots, n$$