CHAPTER 3 第3章

IBM ILOG OPL 语言

3.1 IBM ILOG OPL 基本数据类型

IBM ILOG OPL 基本数据类型包括整型(int)、浮点型(float)、字符串型(string)和布尔型(boolean)。IBM ILOG OPL 语言中的常量分为整型常量、浮点型常量、字符串型常量和布尔型常量。在其他编程语言中,代表这些常量的标识符如果允许值被改变,则称为变量,如整形变量、浮点型变量、字符串型变量和布尔型变量;如果不允许值被改变,则称为符号常量。在 IBM ILOG OPL 模型中,代表这些常量的标识符在模型运行过程中值保持不变(通过 IBM ILOG Sricpt 脚本语言修改除外),因此在 IBM ILOG OPL 中这些标识符被称为数据,如整型数据、浮点型数据、字符串型数据和布尔型数据。标识符的命名原则是:以字母(a～z 和 A～Z)、下划线(_)开头,由字母、数字(0～9)和下划线构成的一个符号系列。

1. 整型常量和整型数据

IBM ILOG OPL 的整型常量采用十进制表示。IBM ILOG OPL 预定义了一个整型符号常量 maxint,代表 OPL 中能够存储的最大正整数,因此,IBM ILOG OPL 整型数的范围为－maxint 到 maxint。

【例 3-1】 定义整型数 i,初始化为 25。

```
int i =25;
```

【例 3-2】 定义整型数 n,并把 n 初始化为 3。然后,定义整型数 size,并把 size 初始化为 n^2。

```
int n =3;
int size =n * n;
```

2. 浮点型常量和浮点型数据

IBM ILOG OPL 提供了双精度的浮点型数据类型(符合 IEEE 754 标准)。浮点型常量采用十进制或科学计数法表示。OPL 预定义了一个浮点型符号常量 infinity,代表无穷大。

【例 3-3】 定义浮点数 f,初始化为 3.2。

```
float f =3.2;
```

3. 字符串型常量和字符串型数据

OPL 提供了字符串型数据类型。字符串型常量是用“”括起来的字符的集合。字符串型常量中支持转义字符。

【例 3-4】 定义一个字符串 s,并初始化为“hello world!”。

```
string s="hello world!";
```

字符串主要用于定义字符串集合,做数组下标。

【例 3-5】 定义了一个字符串集合 Tasks 代表工序的集合,并进行了初始化,然后可以定义数组 Duration 存储每道工序需要的时间,集合 Tasks 中的字符串用作数组下标。

```
{string} Tasks={"masonry","carpentry","plumbing","ceiling", "roofing", "
painting", "windows", "facade", "garden","moving"};
int Duration[Tasks]=[10,4,6,3,6,3,2,5,6,7];
```

字符串中允许包含转义字符,转义字符的表示形式和含义如表 3-1 所示。

表 3-1 转义字符的表示形式和含义

转义字符	含义	转义字符	含义
\n	新行	\b	退格
\t	制表符	\f	换页
\\	\	\r	回车
\"	"	\xhh	ASCII 码为十六进制数 hh 的字符
\'	'	\ooo	ASCII 码为八进制数 ooo 的字符

4. 布尔型常量和布尔型数据

IBM ILOG OPL 提供了布尔型数据类型。布尔型常量只有 true 和 false,代表真和假。定义布尔型数据类型时使用关键字 boolean。布尔型数据类型限定数据只能取 true 或 false 两个值。布尔型数据类型主要是用于定义布尔型决策变量,或者在条件约束中用于决定哪些约束条件有效。

【例 3-6】 定义一个布尔型数据 flag,用于条件约束。

```
boolean flag=true;
if (flag==true)
{
    约束条件 1
}
else
{
    约束条件 2
}
```

3.2 IBM ILOG OPL 区间

OPL 提供了区间(range)这个复杂的数据结构,主要用作数组的下标、迭代区间和定义形参的取值范围。

1. 区间的定义

区间的定义是通过给定下限值和上限值的方式定义能够表达区间的数据。

```
range 区间名=区间下限..区间上限
```

【例 3-7】 定义一个区间数据 Rows,代表整数区间[1,10]。

```
range Rows =1..10;
```

区间的下限值和上限值可以使用表达式。

【例 3-8】 用表达式定义区间。

```
int n =8;
range Rows =n+1..2 * n+1;
```

2. 区间的用途

(1) 用途 1：整型区间用作数组下标。

【例 3-9】 定义具有 100 个元素的整型数组 A,数组下标从 1 到 100。

```
range R =1..100;
int A[R];
```

(2) 用途 2：整型区间用作形参的迭代范围。

【例 3-10】 建立一个循环,形参迭代范围是 1 到 100。

```
range R =1..100;
forall(i in R){ 循环体... }
```

(3) 用途 3：整型区间用作整型决策变量的取值范围。

【例 3-11】 定义一个整型决策变量 i,取值在[1,100]。

```
range R =1..100;
dvar int i in R;
```

(4) 用途 4：浮点型区间用于定义浮点型决策变量的取值范围。

【例 3-12】 定义一个浮点数区间[1.0,100.0],作为决策变量 x 的取值范围。

```
range float X=1.0..100.0;
dvar float x in X;
```

3.3　IBM ILOG OPL 集合

OPL 提供集合(set)这个复杂的数据结构,主要用作数组的下标。集合是非索引无重复元素的集合。OPL 允许定义任何类型的集合。

1. 集合的定义

假设 T 是个数据类型,可以使用{T}或者 setof(T)定义集合数据。

【例 3-13】 定义整型集合 setInt。

```
{int} setInt =...;
```

【例 3-14】 定义结构体类型为 Precedence 的集合。

```
setof(Precedence) precedences =...;
```

默认情况下,集合元素是按照创建的顺序排列的。

2. 集合的初始化

OPL 集合的初始化可以分为两种:一种是内部初始化,在模型文件(.mod)中完成;另一种是外部初始化,在数据文件(.dat)中完成。

(1) 集合的内部初始化。在模型文件中,定义集合的同时使用{}对集合进行赋值。这种初始化方式导致数据在脚本代码中不能使用。

【例 3-15】 初始化集合 Week 为"Mon", "Tu", "Wed", "Thu", "Fri", "Sat", "Sun"。

```
{string} Week={"Mon", "Tu", "Wed", "Thu", "Fri", "Sat", "Sun"};
```

使用 asSet 内置函数和区间数据初始化集合。

【例 3-16】 使用区间[1..10]初始化集合 s。

```
{int} s=asSet(1..10) ;
```

s 被初始化为{1,2,..,10}。asSet 功能是将 range 数据转换为集合数据。

【例 3-17】 初始化集合 s 为{1,4,7,10}。

```
{int} s={i| i in 10:i mod 3==1};
```

(2) 集合的外部初始化。在独立的数据文件中使用{}给出集合元素的值。

【例 3-18】 集合的外部初始化。

在模型文件中声明整型集合 a:

```
{int} a=...;
```

在数据文件中进行集合的初始化:

```
a={10, 20, 30, 40, 50, 60};
```

【例 3-19】 使用数据库表初始化集合。

公交车的班次信息存储在 Access 数据库 bus.accdb 的"发车时刻表"中。在 IBM ILOG OPL 模型中定义公交车班次时刻表结构体 shift。定义 shift 类型的 Shifts 结构体集合,数据表中公交车的班次信息将存储于 Shifts 中。

```
tuple shift{
    string ID;
    int hour;
    int minute;
}
{shift} Shifts=…;
```

在数据文件中添加如下语句:

```
DBConnection db("access","bus.accdb");
Shifts from DBRead(db,"select ID,hour,minute from 发车时刻表");
```

【例 3-20】 使用工作表初始化集合。

城区的交通路口信息存储于名为"cumcm2011B. xls"的 Excel 文件的“全市交通路口的路线”工作表中。定义结构体 adj 代表邻接表的结构，定义 adj 类型的 connectionstmp 结构体集合，工作表中数据将存储于 connectionstmp 集合中。

```
tuple adj
{
    int o;
    int d;
    float dist;
}
{adj}connectionstmp=...;
```

在数据文件中添加如下语句：

```
SheetConnection sheet("cumcm2011B.xls");
connectionstmp from SheetRead(sheet,"'全市交通路口的路线'!A2:B929");
```

3.4　IBM ILOG OPL 数组

1. 数组的定义

IBM ILOG OPL 的数组定义必须给出数组的下标范围。下标范围可以是区间，可以是集合，可以是整数集合或字符串集合，也可以是结构体集合。

(1) 一维数组。一维数组是最简单的数组，定义一维数组需要说明数组元素的类型和下标范围。

【例 3-21】 定义具有 4 个元素的整型数组 a，数组元素分别为 10，20，30 和 40，4 个数组元素的引用下标分别为 1，2，3 和 4。

```
int a[1..4] =[10, 20, 30, 40];
```

其中，区间 1..4 是下标值域；10，20，30，40 分别是通过下标值引用的数组元素的值。

【例 3-22】 定义具有 4 个元素的浮点型数组 f，4 个元素分别为 1.2，2.3，3.4 和 4.5，4 个元素的下标分别为 5，6，7 和 8。

```
float f[5..8] =[1.2, 2.3, 3.4, 4.5];
```

其中，区间 5..8 是下标值域；1.2，2.3，3.4，4.5 分别是通过下标值引用的数组元素的值。

【例 3-23】 定义一个字符串数组 d，其数组元素分别为 d[1]= "Monday"和 d[2]= "Wednesday"。

```
string d[1..2] =["Monday", "Wednesday"];
```

【例 3-24】 定义一个整型数组 a，数组元素的值为 a["Mon"]=10，...，a["Sun"]=70。

首先定义一个字符串集合 Week：

```
{string} Week={"Mon", "Tu", "Wed", "Thu", "Fri", "Sat", "Sun"};
```