o++o

(截至 2022 年 1 月 5 日)

Klaus Benecke

目录

1 愿景

有 4 种基本的单数据运算加法、减法、乘法和除法，分别有几乎标准化的名称 +、 -、×（*）和：（÷）。还有许多其他操作，如 sin、cos、log 等。由于数字化的进展，我们也需要强大的海量数据操作与标准。集合理论操作的集合，如相交(∩)，联合(∪)，集差(\或-)，已经被笛卡尔积(×)，连接(|×|)，投影(π)和选择操作(选择)(σ)所扩展。这些操作隐藏在 SQL 中--关系型数据库的标准数据库语言。当今 80%的数据存储在关系型系统中，如 ORACLE、MYSQL、HANA、DB2、MariaDB 等。...关系模型是由英国数学家 E. F. Codd 发明的。由于关系模型不够强大，无法满足工业应用的要求，计算机科学家们用多集、简单的聚合函数（如 sum、avg 等）、groupby、has、排序操作等来扩展模型。尽管有这些扩展，但结构化表不能用 SQL 处理。然而，有一些非常简单的结构化表格，如

专业，学分 l l

数学 1 2 2 3 1

物理学 2 1 4 1 3 4

这可以让儿童理解。这里我们有一份每个科目的笔记清单（l）。整个表格由(FACH,NOTEl)对的列表(最后 l 个)组成。SQL 甚至 EXCEL 都没有能力处理诸如(FACH,NOTEl l)这样的

方案。因此，我们认为 SQL 将永远无法实现其最初的目标--成为一种终端用户语言。所以我们需要一个新的数据库和文件集的标准。
我们的愿景是，++o 将成为这一标准，用于查询关系型数据库、大数据系统，甚至管理海量文档的系统。它是为终端用户提供结构化表格这一强大的手段。
脸书网 Facebook 和其他公司致力于以这样一种方式准备新闻，它首先吸引了人们的情绪，似乎社会形势还没有足够激烈。我们的方法是不同的。我们正在努力为终端用户提供工具，在计算机的帮助下加强其理性判断。要做到这一点，他必须能够以类似于维基百科的多形式大表格和文件的形式获取广泛的、值得信赖的、可公开获取的信息。 个人应该能够独立评估这些。只有这样，他才能形成自己的判断。
尽管这种语言尽可能地简单，但它需要一定的学习，与 "Facebook 方式 "不同。
这样的编程语言与可信的数据相结合，可以成为社会进一步民主化的一大步。许多错误的信息很容易被任何人通过适当的事实或统计评估而瘫痪。

2 终端用户语言的设计标准

1. 需要有**数学基础。**
2. **方法论-理论和实用性问题**最初面临的是**效率问题。**
3. 终端用户方案应简短易读，以及最重要的关键词。
4. **不要使用循环**（for, while）。循环往往导致程序难以阅读和改变。**一般递归**需要相对较高的抽象水平，因此需要较高的学习努力；这也应该被免除。因此，节目应该简单地从上到下、从左到右进行编写和处理。这需要强大的操作。
5. 最终用户的语言应该是**普遍适用的**。它不仅要能用于关系（扁平的简单表格），而且要能用于结构化表格和结构化文件。它不仅应该适合于对最多样化的系统进行查询，而且还应该适合于多样化的计算。
6. 为了在课堂上适用，它应该**支持**各种**数学子学科，并**为其他学科提供益处。
7. 终端用户语言应该非常**强大**，可以取代其他系统和语言，如电子表格、XQuery、XPath 和 SQL。
8. 从最终用户的角度来看，应该有一个统一的系统，用**统一的语法**（符号）来处理海量数据，就像单一数据处理的操作（+ - * : sqrt sin ... ）是标准化的。

3 用++o 进行计算--不仅仅是一个计算器

o++o 被设计为用户友好型。它的语法简单易记，节目也很短。它的算术运算遵循心算的逻辑，因此运行起来就像在一个简单的袖珍计算器上。
应在 ottops.de 试用以下程序，因为不亲自测试就很难学习编程。
o++o，或者更准确地说，ottoPS（ottoProgrammingLanguage），不仅是一个相当大的扩展的计算器。++o 也允许对结构化的 TABLES 和 DOCUMENTS（tabmente）进行查询。它允许特别简单地制定对表和文件的查询。

方案 3.1：计算 16 的第四根。
```
16 sqrt sqrt
```

结果（点击标签后）。
```
2.
```

你可以看到，个位数的运算被写在输入值（后缀）之后。与已知的符号 sqrt(sqrt(16))相比，这为我们节省了 4 个括号。

方案 3.2：计算 "π 一半 "的正弦。
pi:2 罪

结果。
1.

方案 3.3:计算 30 度的正弦。
30:180*pi sin

结果。
0.5

方案 3.4：有多少个 10 位数的双数？
2 的 10 次方

结果。
1024

方案 3.5：计算体积为 2（2 的三次方根）的立方体的边长。
2 到 1/3 的幂

结果。
1.25992104989

或
2 高 (1:3)

结果。
1.25992104989

1/3 是一个有理数。也就是说，与": "相比，"/"不是一种操作。由于我们总是从左到右计算，2 到 1 的幂：3=（2 到 1 的幂）：3=0.66666667。

方案 3.6：形成一个平均数。
1 3 2 1 3 4 ++:

结果。
2.33333333333

为了节省编写工作，我们可以不使用任何可见的分隔符和任何括号来表示一个数值列表（这里是行的数字）。现在，平均化的操作（++:）被应用于这个列表。也可以形成和（++）、积（**）、数（++1）、最大（max）、正弦（sin）、ln、……，而不是平均。由于 sin 只需要一个输入值，应用正弦操作而不是++：操作的结果是一个输出值的列表。然后你可以再次对这个列表应用++:，只得到一个数字。上面的符号一开始可能需要一些时间来适应，但它比以前的符号更紧凑。
avg([1; 3; 2; 1; 3; 4])
特别是当几个操作连续应用时，这可以节省许多括号，从而减少错误的原因。我们认为++:和这里提到的其他操作是个位数（单数），就像 sin 或 sqrt 一样，并把它们写在输入值后面，因为我们把给定的数字看作**一个**列表（**一个表格**）。

方案 3.7：计算条款的价值
"sqrt（abs（sin（7.1））+abs（cos（8.1）））"。

7.1 孽缘

```
+ 8.1 cos abs

sqrt
```

结果。
0.986160835697

或带一对括号的单线
```
7.1 sin abs + (8.1 cos abs) sqrt
```

结果。
0.986160835697

在三行解决方案中，正如在编程语言中常见的那样，计算是从上到下的。除非省略 "+"号，否则第二行的数值无法计算。因此，第一行和第二行的数值是简单的相加。然后从第二行到第三行的总结果中提取根。

方案 3.8：形成几个数字的乘积。
```
3 5 2 2 **
```

结果。
60

在++o 中，有几个操作是根据非常古老的 "森林原则 "编写的。有一个关于树的词，树 Tree 被称为森林。因此，上面的**取代了 3 个乘法符号。由于这种操作，我们可以不使用阶乘函数。9！可以用１ ...9**来表示 o++o。

方案 3.9：用列表中的每个小数乘以另一个数字。
```
2.40 2.70 7.90 * 1.19
```

结果。
2.856 3.213 9.401

这里，输入列表中的每个数字都要乘以 1.19。如果给定的数字是净价，结果代表相应的毛价。另一方面，如果给定的数字是一种货币的金额，1.19 是转换率，那么结果就代表另一种货币的价值。如果列表中的数字是长方形的长度，那么结果就是 3 个长方形的面积。
我们看到，小数的数字不是用逗号表示，而是用点表示，这是国际上的习惯。

方案 3.10：在不使用许多减号和括号的情况下，形成许多正数和许多负数之和。
```
4 5 3 2 1 8 9 ++

- 7 6 5 4 3 2 1 ++
```

结果。
4

方案 3.11：给定半径，计算几个圆的周长。将结果四舍五入到小数点后 2 位。
```
4 5 6 2 3.7 9.77

*pi*2
```

第 2 轮

结果。
25.13 31.42 37.7 12.57 23.25 61.39

该程序也可以用一行来写。

方案 3.12：给定半径，计算几个圆的周长和面积。

```
Rl:= 4 5 6 2 3.7 9.77

范围:=R*pi*2

FLAT:=R*R*pi

第 1 轮
```

结果。

```
R，SCOPE，FLAT l

4.  25.1     50.3

5.  31.4     78.5

6.  37.7    113.1

2.  12.6     12.6

3.7 23.2     43.

9.8 61.4    299.9
```

通过 R，给定列表的每个元素都得到了名称（标签）R。通过赋值（:=），给定的表被一个新的列所扩展。上面，UMFANG 和 FLAECHE 这两列被一个接一个地加上去，结果是一个 R,UMFANG,FLAECHE l 类型的表。l 代表列表。不幸的是，这很容易与一（1）相混淆。

方案 3.13：计算几个矩形的面积和周长。

```
<TAB!

A, B l

1.23 5.67

7.65 4.32

9.87 6.54

TAB>

范围:=A+B*2

FLAG:=A*B
```

结果。

```
A, B, SURFACE, FLAT l

1.23 5.67 13.8     6.9741

7.65 4.32 23.94   33.048

9.87 6.54 32.82   64.5498
```

TAB 括号（<TAB！, !TAB>）只在系统的程序部分需要。如果 Tabment 存储在一个文件中，系统会通过结尾（如.tab）来识别格式。
在 TAB 显示中，数值必须在相应的列名下开始。

方案 3.14：确定一个简单计算的总价格。

```
<TAB!

文章，价格 l

啤酒 0.61
```

淋浴 0.23

炸肉饼 2.40

TAB>

++

结果。
3.24

这里，总和是在给定的表格（成对的列表（l））中的所有数字上简单形成的。文章的价值是单词，因此对结果没有影响。将++替换为
*1.1

，结果又是一个有 2 列 3 行的表格（集合，图元），其中每个数字都包括 10%的提示。
文章，价格 l

啤酒 0.671

淋浴 0.253

炸猪排 2.64

之后，你可以再次
++

来得出总数（这里是 3,564）。

方案 3.15：确定一个简单表格所给出的略微复杂的计算的总价格。
<TAB!

文章，价格，编号 l

啤酒 0.61 7

淋浴 0.23 3

炸肉饼 2.40 4

TAB>

posprice:=price*number

销售点价格

结果。
14.56

该作业在给定的表中增加了一个新的列，列名为 POS PRICE，将三个价格分别乘以相关数字。为了确定酒馆访问的总和，人们仍然必须给++操作提供第二个输入值。否则，就会形成中间表的所有 9 个数字的总和。方案的两行也可以简单地替换为
++ PRICE*COUNT

可以被替换。在一个程序行的开头，操作的第一个输入值总是前一个程序行的结果。
赋值的返回符号:=要与等号=区分开来。等号以及<, >, <=, in, ...需要用来制定条件。条件服务于选择--删除（结构化）的线条。
例如，如果添加一个条件
与 ARTICLE=啤酒

或在这里简单地
带啤酒

最后的结果只是 7 种啤酒的总价格。如果你想计算其余部分的价格，你可以改用

```
sans ARTICLE=beer
```

或者干脆
无啤酒

插入。
列名（元数据）必须始终大写。关键词（gib, sans, avec, ...）必须始终以小写书写。如果一个人总是用大写**和**小写字母来写主要数据的一个字，程序就会变得容易阅读。

方案 3.16：确定一个长的酒馆访问的总价格。

```
<TABH!
文章，价格，编号 l
啤酒 0.61 7 6 5
                3
淋浴 0.23 3 4
炸肉饼 2.40 4 3 2
TABH>
posprice:=price*number
销售点价格
```

结果。

```
36.02
```

在这里，我们处理的是一个结构化的表格，其中包含每个项目的几个订单。由于我们选择了 TAB 的横向版本 TABH，我们不必将数字条目一个一个地写在下面，这样就可以紧凑地呈现出酒吧的访问。另一方面，如果我们不考虑最终的结果，而只考虑应用赋值的结果，那么 NUMBER 和 POS PRICE 的值必须一个一个地写在下面，否则产生的结构化表格会变得过于混乱。

```
ARTICLE, PRICE, (NUMBER, POS PRICE l)l
啤酒 0.61 7 4.27
                  6       3.66
                  5       3.05
                  3       1.83
淋浴 0.23 3 0.69
                  4       0.92
炸肉饼 2.4 4 9.6
                  3       7.2
                  2       4.8
```

你想先把数值相加，然后再相乘吗？

```
posprice:= number ++ *price
```

其结果又是一个更紧凑的中间表。

```
文章，价格，POS 价格，编号 l m
啤酒 0.61 12.81 7 6 5 3
淋浴 0.23 1.61 3 4
炸肉饼 2.4 21.6 4 3 2
```

这两条程序线也可以再次用++PRICE*NUMBER 代替。
你可以把表格保存在一个名称下，例如扩展名为.tab 或.tabh，然后再调用它们。通过这种方式，表格或文件可以在多个程序中使用。
然后，方案 3.16 可以是这样的。

```
酒吧访问.TABH

++ PRICE*COUNT
```

这个例子可以转移到许多应用中。在保龄球运动中，人们也可以建立一个带有重复组的 kegeln.tabh 表。NAME,WURFl m.
各个投掷物的名称和高度要根据需要输入。
m 缩写为数量，l 缩写为清单。对于每个保龄球手，你就可以通过一项任务确定总和，然后对数据进行降序排序。如果你只想让某些列出现在结果中，而且你还想按这些列排序，你需要一个 gib 子句。

方案 3.17：确定每个人的保龄球总分，并据此对数据进行排序。

```
保龄球.tabh

总计:=WURFl ++

给予总计，NAME m-
```

它也可以更短一点。

```
保龄球.tabh

给予总计，NAME m-

    总数:= 投掷!++
```

这里，聚合的参考（这里是++）来自于所需表格的标题。TOTAL 是每个 NAME 的集合。对于集合（m，m-），总是按照所给的第一列名称进行排序。

方案 3.18：计算 3 名学生的加权平均数和总平均数。

```
<TABH!

名称, KLAl, 注释l l

恩斯特 1 2 1 2 3 1 3 1 1

克拉拉 1 1 3

索菲亚 1 3 1

TABH>

DUR:=KLAl ++:*0.6 +(NOTEl ++: *0.4)

总计:=DURl ++:

第 2 轮
```

结果。

```
总数，（名称，DUR，KLAl，注释l）。

1.66 恩斯特 1.59 1 2 1 2 3 1 3 1 1

         克拉拉 1.8 1 1 3

         索菲亚 1.6 1 3 1
```

方案 3.19：一个带软木塞的瓶子需要 1 马克和 10 马克。这瓶酒的价格比软木塞的价格高一个档次。软木塞的价格是多少？

瓶子:=0 ... 110

软木:=瓶子-100

包括 CORK+BOTTLE = 110

结果。

瓶子，酒瓶 l

105 5

第一个赋值给了从 0 到 110 的每个数字以标签 BOTTLE。通过观察导师的表述，可以最好地看到这一点。

```
<TABM>
  <bottle>0</bottle>
  <bottle>1</bottle>
  <bottle>2</bottle>
  <bottle>3</bottle>
  ...
  <bottle>108</bottle>
  <bottle>109</bottle>。
  <bottle>110</bottle>
</TABM>
```

如果 FLASCHE:=0 ...110 的赋值被错误地写入，标签 FLASCHE 只会出现一次。

<BOTTLE>的情况

0

1

2

3

. ..

108

109

110

</FLASH>

第二个解决方案。

瓶子:=0 ...110

cork1 := bottle - 100

cork2 := 110 - bottle

avec KORK1=KORK2

结果。

瓶子，CORK1，CORK2 l

105 5 5

从方法论的角度来看，这种解决方案是有利的，因为你可以通过点击图像按钮来说明程序的前 3 行。你可以看到，我们在这里处理的是两条直线，其交点是由条件决定的。

方案 3.20：写 20 遍我爱你。
"Je vous aime"。20 次

结果。
文本 l

"Je vous aime"。"Je vous aime"。"Je vous aime"。"Je vous aime"。

"Je vous aime"。"Je vous aime"。"Je vous aime"。"Je vous aime"。

"Je vous aime"。"Je vous aime"。"Je vous aime"。"Je vous aime"。

"Je vous aime"。"Je vous aime"。"Je vous aime"。"Je vous aime"。

"Je vous aime"。"Je vous aime"。"Je vous aime"。"Je vous aime"。

倍是一个二进制操作，像所有其他二进制操作一样，以 infix（在输入值之间）书写。

4 结构化表的模式和 TTD

我们在此插入本章，以澄清简单（平面）表和结构化表之间的区别。
首先，一个表被分为 n 列 A1,A2,...,An。o++o 程序中的列名不得包含小写字母。如果所有的 A1,A2,...,An 都有一个基本类型，一个 A1,A2,...,An 表正好包含一个简单的行，例如。
姓名、地点、工资

Paul Oehna 1000

如果我们附加一个集合符号，例如 l 代表 list，那么这个表可以包含 0、1 或更多的行。例子。
姓名、地点、工资 l

Paul Oehna 1000

索菲亚-达尔戈 900

Claudia Dallgow 2000

Clara Oehna 900

扁平表的 tab 表示（三要素列表）。

名称	地点	工资
保罗	鄂鄂纳	1000
索菲亚	达尔豪斯	900
克劳迪娅	达尔豪斯	2000
克拉拉	鄂鄂纳	900

平面表的图形表示
如果我们用数量符号 m 代替 l，表格就会排序显示。
姓名、地点、薪金 m

Clara Oehna 900

Claudia Dallgow 2000

Paul Oehna 1000

索菲亚-达尔戈 900

如果这个表要按位置排序，只需在相应的语句中交换列，给 LOCATION,NAME,CONTAINMENT m
地点，姓名，工资 m

Dallgow Claudia 2000

达高-索菲亚 900

Oehna Clara 900

Oehna Paul 1000

现在你可以看到，该表包含一些冗余。有了结构化的表格，你可以消除这种情况（给出 LOCATION,(NAME,CONTENT m)m）。
地点，（姓名，工资 m）m

Dallgow Claudia 2000

索菲亚 900

Oehna Clara 900

保罗-1000

(structured_table.tab)

这个表包含 2 个 struples（结构化图元=结构化记录=结构化元素），也就是说，例如，表中（现在的结构化）元素的数量（++1）不再是 4，而是 2。当然，人数仍为 4 人）。)尽管如此，我们还是可以用 O++o 处理这个表，方法与前面的表相同。从初始表，我们也可以用 gib 语句创建 2 或 3 个表。
地点 m, 姓名 m, 薪资 m

达高-克拉拉 900

Oehna Claudia 1000

保罗-2000

索菲亚

(three_collections.tab)

这个完整的表包含了初始表的所有基本数据，但其他信息已经丢失。Dallgow 和 Clara 在这个标签表述中处于同一行，但 Clara 并不住在 Dallgow。因此，你总是要看一下模式。严格来说，我们面对的是 3 组的三倍。
{Dallgow Oehna}, {Clara Claudia Paul Sophia}, {900 1000 2000}.
在这里，逗号代表 "配对"，而空格则是分隔各组元素的。
structured_table.tab 的模式是。
地点,(姓名,薪资 m)m。

在许多情况下，这种元数据已经足够了。 然而，从这一点上还不清楚，例如，如何用哪一列的数值进行计算。这一点在 TTD（Tabment Type Definition）中说得很清楚。
帐单!(LOCATION,(NAME,CONTENT m) m)

薪资！数字

名字的位置!词语

TABMENT 是一个由 TABelle 和 dokuMENT 组成的单词。如果该表以上述名称保存，则 TTD 会相应地扩展。
帐单！ 结构化表

structured_table!(LOCATION,(NAME,CONTENT m) m)

薪资！数字

名字的位置!词语

收集符号的拼写可能需要一些时间来适应。集合符号的写法与其他一位数的操作一样，都是按照计划进行的。如前所述，ORT 计划代表着一个地方。也就是说，一个具有这种模式的++o 表（有这个标题）只能包含一个位置。相反，标题为 ORT 的关系或 EXCEL 表格包含任何数量的位置。在++o 中，人们有三种可能性来表示这些 "关系"。

ORT ORTl ORTb ORTm

达尔高 达尔高 达尔高 达尔高

Oehna Oehna Dallgow Oehna

Dallgow Dallgow Oehna

Oehna Oehna Oehna

EXCEL o++o-List o++o-Bag o++o-Quantity

袋是英语，也是多套的意思。然而，对我们来说，毫米在拼写中太长了。我们认识到，袋是分类的，套是分类的，另外套也不包含重复的。ORTm 和 ORT m 方案表达了同样的内容。只有在有几列的情况下，差异才会变得明显。
地点，名称 m

包含几个(LOCAL,NAME)对，而
地点，名称 m

可以只包含一个位置，但有几个名字。也就是说，在列名后面没有空格的集合符号可以节省一对括号。
地点，名称 m

相当于
地点，(NAME m)

.类型的表格
地点，名称 m m

那么许多不同类型的表格
地点，名称 m

含有。在 gib 指令的上下文中，结果是
给予地点，名称 m m

每个地方只有一次。这允许
地点，名称 m m

Dallgow Claudia

达戈-索菲亚

Oehna Ernst

克拉拉

不是这个 Gib 声明的结果，而是。
地点，名称 m m

Dallgow Claudia

索菲亚

Oehna Ernst

克拉拉

只有当你为每个地方存储进一步的数据时，这种结构化的优势才会真正体现出来，比如居民人数、市长.....内部的 m 说，每个地方都存在一组名字。这再次意味着，在一个地方，任何名字都不能出现超过一次。尽管如此，在达尔古的第二个恩斯特当然也可以输出。
在 O++o 中，每个标签都有一个模式。这甚至适用于用户没有分配模式的单个值。例如，Gerwisch 的基本模式是 WORD，39175 的基本模式是 NUMBER。这使格维希成为一个标签。然而，与"{Gerwisch}"相比，Gerwisch 并不是关系数据模型的一个表。这似乎很有争议，但它对数据模型的架构有很大影响。

5 简单的递归赋值

除了带有相关公式的简单计算（:=），即通过一个新的列来扩展一个表之外，还有所谓的递归赋值。这里，给出了两个公式。第一个公式是针对相应集合的新列的第一个元素，第二个公式是针对其余元素。其余元素的公式可以参考该列的前身。这在以下方案中 BETRAG 预测。

方案 5.1：10 年后，在 9%的增长（利息）下，100 欧元会变成什么？
YEARl:= 0 ... 10

AMOUNT:=前 100 名。下一个 AMOUNT 预测 +% 9，年。

第 2 轮

结果。
年份，金额 l

0	100.
1	109.
2	118.81
3	129.5
4	141.16
5	153.86
6	167.71
7	182.8
8	199.26
9	217.19
10	236.74

方案 5.2：将一个双数（这里是 10111=23）转换成一个十进制数。
BITl:= 1 0 1 1

DEZI:=第一个 BIT 下一个 DEZI pred*2+BIT at BIT

结果。
BIT，DECI l

1	1
0	2
1	5

```
1    11
1    23
```

如果不需要整个发展，而只需要列表中的最后一个元素，可以添加条件 avec BIT pos-=1 或简单的最后一个。如果最后一个位子也不想要，就应该紧接着进行 gib DEZl。

方案 5.3 将一个十进制数（这里是 23）转换成一个二进制数。

```
AVERAGE :=前 23 名，最差 2 名
              下一个 AVERAGE 预测第 1 个 divrest 2
              while AVERAGE ++ > 0
给予 RESTl-
```

结果。

```
1 0 1 1 1
```

在这里，通过递归扩展，在一个名称下引入了 2 个新列，即 DIV 和 REST。divrest 在这里是整数除以整数余数（一对数字的标签为 DIV 和 REST）。

方案 5.4：计算如果每月偿还 900 欧元，年息 2%的房屋建设贷款 110 000 欧元的月度发展。

```
# 初始贷款 110000 欧元
MONLAST:=900 # 月负荷
PROZ:=1.02 到 1/12 的幂 -1# 2 %的年利息
CREDIT,INTEREST,DEPOSIT l:=
     前 110000. , 0. , 0.
          下一个 CREDIT 预测-(DILT 预测), (CREDIT*PROC), (MONLAST INTEREST,CREDIT
min)
     而 CREDIT>0 在 PROZ
YEAR,MON:=KREDIT pos -1 divrest 12+1 leftat KREDIT
proz::=proz*100
第 2 轮
```

结果:

```
MONLAST, PROC, (YEAR, MON, CREDIT, INTEREST, TILGATION l)


900      0.17    1     1   110000.     0.     0.
                 1     2   110000.   181.67 718.33
                 1     3   109281.67 180.49 719.51
                 1     4   108562.16 179.3  720.7
                 1     5   107841.46 178.11 721.89
                 1     6   107119.57 176.92 723.08
                 1     7   106396.49 175.72 724.28
                 1     8   105672.21 174.53 725.47
                 1     9   104946.74 173.33 726.67
                 1    10   104220.06 172.13 727.87
```

```
 1    11   103492.19 170.93 729.07
 1    12   102763.12 169.72 730.28
 2     1   102032.84 168.52 731.48
 2     2   101301.35 167.31 732.69
 2     3   100568.66 166.1  733.9
 2     4    99834.76 164.89 735.11
 ...
11    12     5930.53   9.79 890.21
12     1     5040.33   8.32 891.68
12     2     4148.65   6.85 893.15
12     3     3255.5    5.38 894.62
12     4     2360.88   3.9  896.1
12     5     1464.78   2.42 897.58
12     6      567.2    0.94 567.2
```

更复杂的递归赋值（用++o 数）可以在第 8 章找到。

6 查询

我们现在假设 fluesse.tabh 中的每条河流都有一个 LAENGE 列。然后我们通过以下查询找到所有长度超过 500 公里的河流。

方案 6.1：在给定的表格中选择。
fluesse.tabh

含有 LAENGE > 500

"avec "是法语，意思是 "与"。类似地，我们用 "sans "表示 "没有"。
现在要根据长度对河流进行分类，并与支流一起输出。

方案 6.2：带有后续排序的选择。
来自 fluesse.tabh

与 LAENGE>520

给出长度，流量，流量 m m

结果。
长度、流量、侧向流量 m m

```
544 Main "Fraenkische Saale" Gersprenz Kinzig
                洛尔巴赫-尼达-雷赫滕巴赫-雷格尼茨
                "红色主线 "陶伯 "白色主线
544 摩泽尔阿尔夫-德隆-埃尔兹巴赫-凯尔-利泽-马东-默尔特
                Moselotte Orne Ruwer Saar Salm Sauer Seille
                沃洛涅
544 萨尔-布利斯-尼德-普林斯-罗塞尔
```

866 或 Bartsch Birawka Bober Eilang Faule Obra

Glatzer Neisse Hotzenplotz Ihna Iseritz

卡尔特巴赫-卡茨巴赫-克洛德尼茨-洛河-马拉帕内

米策尔-内塞-奥勒-奥帕-奥斯特拉维扎（Mietzel Neisse Ohle Olsa Oppa）。

Pleiske Raude Roehrike Stober Summina Thue

铁芬巴赫-瓦特柳-威斯特里茨

威斯福特鲶鱼 Zinna

1165 易北河之鹰 Aland Alster Biela Bille Dahle

多尔尼茨-埃格尔-埃斯特-戈特鲁巴-哈维尔

Ilmenau Iser Jahna Jeetze Kamnitz Kirnitzsch

拉赫斯巴赫 洛克维茨 罗克尼茨 模范莫尔道

Mulde Mueglitz Ohre Oste Polzen Priessnitz

萨勒州 "Schwarze Elster" Stepenitz Stoer

坦吉尔-韦瑟里茨-韦瑟里茨 "野生母猪

Zidlina

1320 莱茵河 Aare Ahr Alb 阿尔布吕克附近的 Albbruck Argen Birs

"Bregenzer Ache" Duessel Elde Emscher Erft

Glatt Hinterrhein Ill Kander Kinzig

Kraichbach Lahn Lauter Leiblach Leimbach

利奥波德运河 利普河 摩泽尔河 穆尔格河 纳赫河

内卡-凯奇-鲁尔-舒森-西格-施佩尔巴赫

Thur Toess Vorderrhein Weschnitz Wied Wiese

吴佩尔-乌塔赫

2845 年多瑙河阿本斯-阿尔特穆尔蓝布雷格-布伦兹-布里加赫-瑟纳

多瑙河三角洲 多瑙河运河 Drava Eipel Enns Fischa

Friedberger Ach Grosse Laaber Grosse Muehl

Guenz Hron Iller Ilz Inn Ipel Isar Jantra

Jiu Kamp Small Pair Krems Lauchert Lech

Leitha March Mindel Morava Naab Olt Paar

Pruth Raab Rain Riss Save Schmutter

施韦查特-特梅施-泰斯-蒂莫克-特莱森-特劳恩

维尔斯-瓦赫-沃尔尼茨-Ybbs Zusam

如果你想先输出最长的河流，你只需要用 m-替换外层的 m。

我们的文件 fluesse.tabh 也包含了每个流量的另一个重复组 LAND,BUNDESLANDm m，表示河流流经哪个州和联邦州。因此，在 fluesse.tabh 中，RIVER 的地位高于联邦国家。如果想反过来，也可以简单地通过指定所需表格的标题来实现这一点。

方案 6.3:重组烟道文件。
来自 fluesse.tabh

给予 "联邦",河 "m

结果。
联邦，河流 m

巴登-符腾堡州 多瑙河 伊尔内卡河 莱茵河

巴伐利亚州 阿本斯 多瑙河 古恩兹 伊勒河 伊萨尔酒店

Lech Main Mindel Naab Regen Saale

韦尔塔赫-沃尼茨

柏林 Havel Spree

勃兰登堡 Aland Elde Havel Neisse Oder Spree

斯特佩尼茨-乌克尔

不来梅威悉

汉堡 阿尔斯特河 易北河 埃斯特河 万德斯

黑森州 富尔达 美因河 内卡河 莱茵河 维拉河 威悉河

梅克伦堡-西波美拉尼亚 阿兰德-埃尔德-哈维尔 罗克尼茨 奥德

佩内-斯泰潘尼茨-乌克尔-瓦诺

下萨克森州 易北河 埃斯特 富尔达 耶茨 洛克尼茨

奥斯特-韦尔拉-韦塞尔

北莱茵-威斯特法伦州 莱茵-鲁尔区 威悉区 乌珀区

莱茵兰-巴伐利亚州 摩泽尔州 莱茵州 萨尔州

萨尔州 萨尔

萨克森州 易北河 尼斯河 斯普瑞河

萨克森 安哈尔特 博德 易北河 哈维尔 伊尔姆 萨勒 恩斯特鲁特

石勒苏益格-荷尔斯泰因州阿尔斯特易北河畔的炉子旅行者 Wandse

图林根州 Ilm Saale Unstrut Werra

通过这种自由重组，所有流经每个联邦州的河流都被收集起来。联邦州和联邦州内的河流被分类。在这里，一条河可以发生在几个联邦州。每个联邦州只出现一次，因为我们选择了一个集合作为外部集合。你也可以用 "关闭 "来标记一个节目部分的开始。

方案 6.4：形成一个联盟（其中一个表中包含的所有学生 ID）。
从 examen.tab, projects.tab

给予 STIDm

结果（tabh）。
STIDm

1234 1245 3456 4567 5678

这里，examen.tab 和 projekte.tab 是以下表格。
STID, COURSE, NOTE m,(STID, PROJ, HOURS m)

1234 代数 1 1234 弗里茨 4

1234 历史 1 1234 奥托 2

1234 逻辑学 2 1245 孔子 5

1245 代数 3 1245 明 4

1245 数据库 1 1245 奥托 6

1245 奥托 1 4567 莫奈 10

3456 数据库 1 5678 莫奈 20

3456 OCaml 2

5678 Apel 1

5678 雷平 1

方案 6.5：取平均数（两个表格中包含的所有学生 ID）。
从 examen.tab, projects.tab

igib STIDm

结果。
STIDm

1234 1245 5678

igib（见第 7 节）也可以用来表达 "连接"，从而制定对整个数据库的查询。下面的查询指的是一个数据库 uni.tab，它的第二个表是非结构化的。如果 uni.tab 的所有表都是非结构化的（关系型），则查询不必修改或只需稍作修改。
FAK, DEKAN, BUDGET, STUDKAP m

Inf Reichel 10000 500

艺术定制 2000 600

数学达索 1000 200

菲洛-黑格尔 1000 10

STID, NAME, LOCATION?, STIP, FAK, (COURSE, NOTE m), (PROJ, HOURS m) m

1234 Ernst Oehna 500 数学代数 1 Fritz 4

历史 1 奥托 2

逻辑 2

1245 Sophia Berlin 400 Inf 代数 3 Confuc 5

数据库 1 明 4

奥托 1 号 奥托 6 号

3456 Clara Oehna 450 Inf 数据库 1

OCaml 2

4567 乌尔里克 400 艺术莫奈 10

5678 Kaethe Gerwisch 0 Art Apel 1 Monet 20

转载 1

该数据库由 2 个表组成。表 "STUDENTS "是结构化的，因为每条记录都存储了 7 个组件（记录=结构化元组=学生），最后两个组件本身又是小表。例如，克拉拉的倒数第二部分包含两个考试成绩，最后一个包含空的项目集。表 "FAKS "是关系型的。

方案 6.6：计算大学的总预算。
从 uni.tab

```
++ 预算
```

结果。

```
14000
```

方案 6.7：计算大学的总预算和平均预算。
从 uni.tab

```
给予 SUMBUD,DURBUD
    Sumbud:=预算!++
    durbud:=预算!++:
```

结果。

```
SUMBUD, DURBUD
14000   3500.
```

由于最后两行以超过 3 个空格开始，从逻辑角度看，它们仍然属于前一行。

方案 6.8：大学有多少个院系和学生。

```
uni.tab
++1
```

结果。

```
4     5
```

方案 6.9：给出每门课程的相关学生与成绩。
从 uni.tab

```
给予课程，（名称，注意 b）m
```

结果。

```
课程，（名称，注意 b）m
代数 严肃 1
            索菲亚 3
Apel Kaethe 1
数据库 Clara 1
            索菲亚 1
历史恩斯特 1
逻辑严肃 2
OCaml Clara 2
奥托-索菲亚 1
Repin Kaethe 1
```

方案 6.10：给出所有含有 1234 的集合（struples）。
从 uni.tab

avec 1234

结果为 hsq（分层顺序）输出。
身份证名称 地方规定 FAK

课程说明

项目时间

基金会的预算

1234 Ernst Oehna 500 数学专业

代数 1

故事 1

逻辑 2

弗里茨 4

奥托 2

方案 6.11：在所有包含学生标识符的表中选择学生编号 1234。
uni.tab
avec STID=1234
结果。
身份证名称 地方规定 FAK
课程说明
项目时间
基金会的预算
1234 Ernst Oehna 500 数学专业
代数 1
故事 1
逻辑 2
弗里茨 4
奥托 2
Inf Reichel 10000 500
艺术定制 2000 600
数学达索 1000 200
菲洛-黑格尔 1000 10
方案 6.11 与方案 6.10 的不同之处仅在于 FAKS 表。 在方案 6.10 的结果中，它是空的，而在方案 6.11 中，它完全保持原样。在其他应用中，代替学生标识符（STID），人们也可以选择文章编号或零件编号，....

7 对多个表的请求没有连接（igib）。

方案 7.1：把院长和他的考试交给号码为 1234 的学生。
从 uni.tab

avec STID=1234

igib 名称,DECAN,(CURSE,Note m)m

结果。
名称，DECAN，（课程，注意 m）m

Ernst Dassow 代数 1

故事 1

逻辑 2

这里的 "连接 "是在没有连接条件的情况下实现的。如果使用 gib 而不是 igib，结果集仍然是空的，因为 gib 并没有通过 FAK 在 NAME 和 DEKAN 之间建立连接。

方案 7.2：计算每个院系的 1 的数量。

从 uni.tab

avec NOTE=1

给 FAK,ANZ m

ANZ:=NOTE!++1

结果。

FAK, ANZ m

无 3

艺术 2

数学 2

菲洛 0

方案 7.3: 测试以下方案。

从 uni.tab

与 DEKAN 在[Reichel Dassow]的合作。

给 FAK,DEKAN,(NAME,STIP m)m

结果。

FAK, DEKAN, (NAME, STIP m) m

Inf Reichel

数学达索

方案 7.4：将所有相关的学生交给达索和赖歇尔的学院。

从 uni.tab

与 DEKAN 在[Reichel Dassow]的合作。

igib FAK,DEKAN,(NAME,STIP m)m

结果。

FAK, DEKAN, (NAME, STIP m) m

Inf Reichel Clara 450

索菲亚 400

数学达索-恩斯特 500

方案 7.5：寻求所有拥有至少两个 A 和一个 C 的学生的院系。

从 uni.tab

avec {{1 1 3}} inmath NOTEb

igib FAK,DEKAN m

结果。
FAK, DEKAN m

Inf Reichel

"b "代表包（多套）。

方案 7.6：我们正在寻找所有正好有两个 A 的学生。
从 uni.tab

avec NOTEl = [1 1]

给予学生

结果（hsq）。
身份证名称 地方规定 FAK

课程说明

项目时间

5678 Kaethe Gerwisch 0 艺术

阿佩尔 1

转载 1

莫奈 20

方案 7.7：寻求恩斯特的所有数据（包括他的教员数据）。
从 uni.tab

avec NAME=Ernst

igib

结果。
身份证名称 地方规定 FAK

课程说明

项目时间

基金会的预算

1234 Ernst Oehna 500 数学专业

代数 1

故事 1

逻辑 2

弗里茨 4

奥托 2

数学达索 1000 200

如果 uni.tab 的所有表都是第一正常形式（非结构化），则不需要改变程序。

方案 7.8：给定一个单独的价格表，确定一个较长的酒吧访问的总价格。
<TABH!

文章，编号 l m

啤酒 2 4 5

淋浴 3 2

炸肉饼 4 3

TABH>

, artikels.tab

TOT,(ARTICLE,TOT m) TOT:=NUMBER*PRICE!++

结果。
TOT, (ARTICLE, TOT2 m)

79.3 啤酒 29.7

淋浴 16.

炸肉饼 33.6

artikels.tab。

文章，价格m

啤酒 2.7

淋浴 3.2

炸猪排 4.8

牛排 4.8

方案 7.9：寻找恩斯特的院长。(对关系型数据库的查询)。
非线性.选项卡

avec NAME=Ernst

igib DEKAN

结果。
德勤

达索

unirelational.tab 有以下模式。
帐单!无国籍

无国籍!学生,考试,项目,股票

项目!(STID, PROJ, HOURS m)

学生们!(STID,NAME,PLACE?,STIP,FAK m)

检验!(STID, COURSE, NOTE m)

FAKS!(FAK, DEKAN, BUDGET, STUDKAP m)

方案 7.10：给我恩斯特的所有数据。
非线性.选项卡

avec NAME=Ernst

igib

结果作为 hsq 输出。
身份证名称 地方规定 FAK

STID 课程说明

STID 项目时间

基金会的预算

1234 Ernst Oehna 500 数学专业

1234 代数 1

1234 历史 1

1234 逻辑 2

1234 弗里茨 4

1234 Otto 2

数学达索 1000 200

应该注意的是，恩斯特的所有数据都出现在结果中，没有使用交叉乘积。否则，"Fritz "和"Otto "这两个项目将各出现 3 次。

方案 7.11：我们正在寻找编号为 1234 的学生的考试成绩。
非线性.选项卡

avec STID=1234

igib 名称,DECAN,(CURSE,Note m)m

结果。
名称，DECAN，（课程，注意 m）m

Ernst Dassow 代数 1

故事 1

逻辑 2

应该注意的是，结果包含来自 3 个表的信息，不需要连接条件。

8 一个带有++o 号的零件清单决议

方案 8.1：搜索 Polo 和发动机的所有组件和个别零件。
<TAB!

OT, PROPERTY, (UT, ANZ m) m

布什圆柱形

边缘光滑

Polo 现代轮毂 4

发动机 1

身体 1

高尔夫快速轮 4

备用轮 1

身体 1

克利马尔 1

发动机 1

身体蓝色

气候和健康

KolbRing 轮

活塞灯 KolbRing 2

插座 1

发动机重型活塞 6

螺钉 8

车轮圆形螺丝 5

轮胎 1

边缘 1

备用轮圆螺丝 4

轮胎 1

边缘 1

黑色轮胎

螺钉稳定

TAB>

onrs OTTONR ![Motor Polo] # 引入 O+O 数字（ONR）。

RANZ:=firstonr ANZ nextonr RANZ pred * ANZ at ANZ # ONR 递归

给予 OT,(UT,TOT m) m TOT:=RANZ!++

第一条指令后的中间结果。

OT, PROPERTY, (OTTONR, UT, ANZ m) l

重型马达 1 螺钉 8

2 活塞 6

2.1 KolbRing 2

2.2 插座 1

Polo 现代 1 轮 4 号

1.1 螺钉 5

1.2 轮胎 1

1.3 Rim 1

2 发动机 1

2.1 螺钉 8

2.2 活塞 6

2.2.1 KolbRing 2

2.2.2 插座 1

3 身体 1

应用递归 o++o 数字分配后的中间结果。

OT, PROPERTY, (OTTONR, UT, ANZ, RANZ m) l

重型马达 1 螺钉 8 8

2 活塞队 6 6

2.1 KolbRing 2 12

2.2 插座 1 6

Polo 现代 1 轮 4 4

1.1 螺钉 5 20

1.2 轮胎 1 4

1.3 边缘 1 4

2 发动机 1 1

2.1 螺钉 8 8

2.2 活塞 6 6

2.2.1 KolbRing 2 12

2.2.2 插座 1 6

3 身体 1 1

最终结果。

OT, (UT, TOT m) m

电机插座 6

活塞队 6

KolbRing 12

螺钉 8

Polo 插座 6

边缘 4

身体 1

活塞队 6

KolbRing 12

发动机 1

第 4 轮

轮胎 4

螺钉 28

9 生成图像

方案 9.1：用多种颜色输出三个函数的点。

sin 和 sin 的一阶导数

该表可通过图片按钮进行可视化。

Xl:= -4 ...10 !0.005

SIN:= X sin

扣除:=X+0.001 sin - (X sin) : 0.001

NULL:=X*0

RGB:=绿色 左侧的SIN

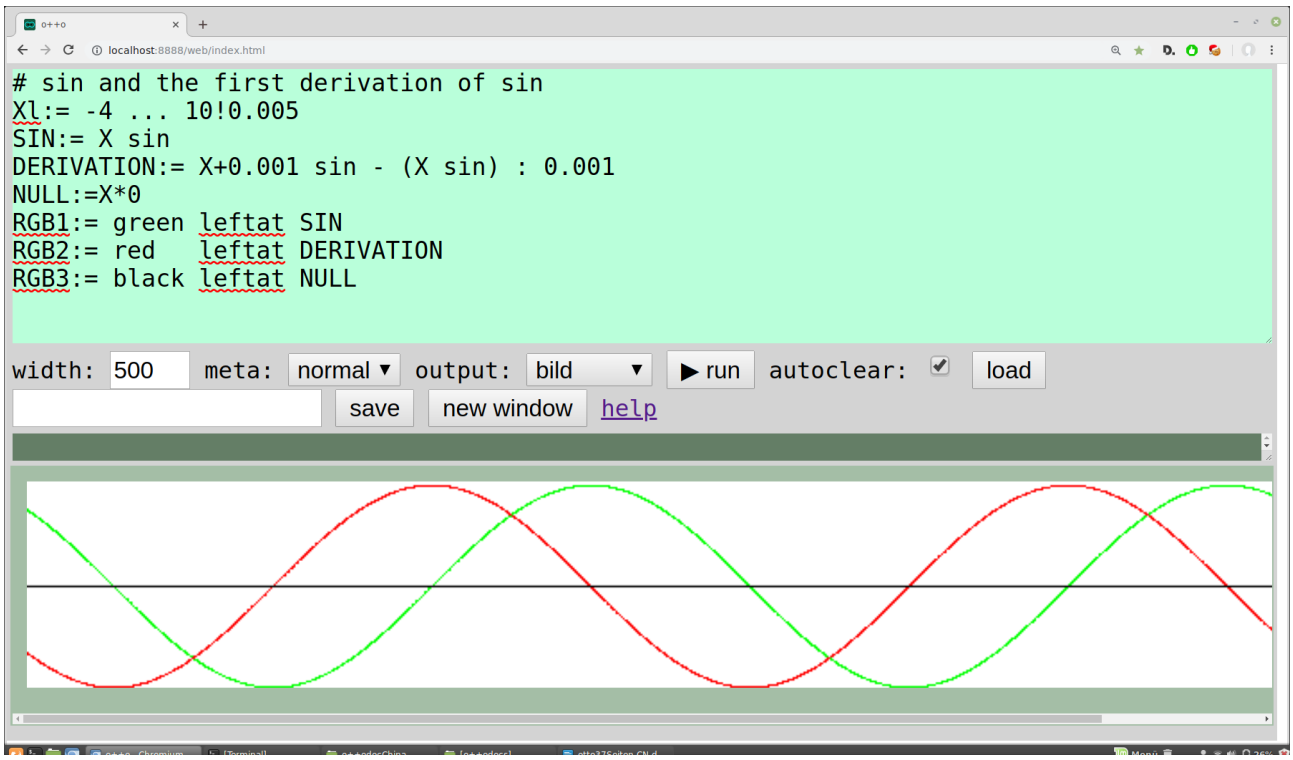
RGB:=红色 左图 ABSTRACT

RGB:=黑色 左侧为NULL

从2800点的结果表中提取（不含颜色值）。

X, SIN, DEDUCTION, NULL l

-4.	0.756802495308	-0.654021913139	-0.
-3.995	0.75352483081	-0.657796099859	-0.
-3.99	0.75022832823	-0.661553841711	-0.
-3.985	0.746913069981	-0.665295044752	-0.
-3.98	0.743579138944	-0.66901961545	-0.
-3.975	0.740226618468	-0.672727460694	-0.
-3.97	0.736855592364	-0.676418487786	-0.
-3.965	0.73346614491	-0.680092604451	-0.
. ..			
9.96	-0.510032040244	-0.859900243999	0.
9.965	-0.514326423954	-0.857337195738	0.
9.97	-0.518607949529	-0.854752714092	0.
9.975	-0.522876509934	-0.852146863673	0.
9.98	-0.527131998452	-0.849519709627	0.
9.985	-0.531374308698	-0.846871317632	0.
9.99	-0.535603334614	-0.844201753898	0.
9.995	-0.539818970475	-0.841511085165	0.



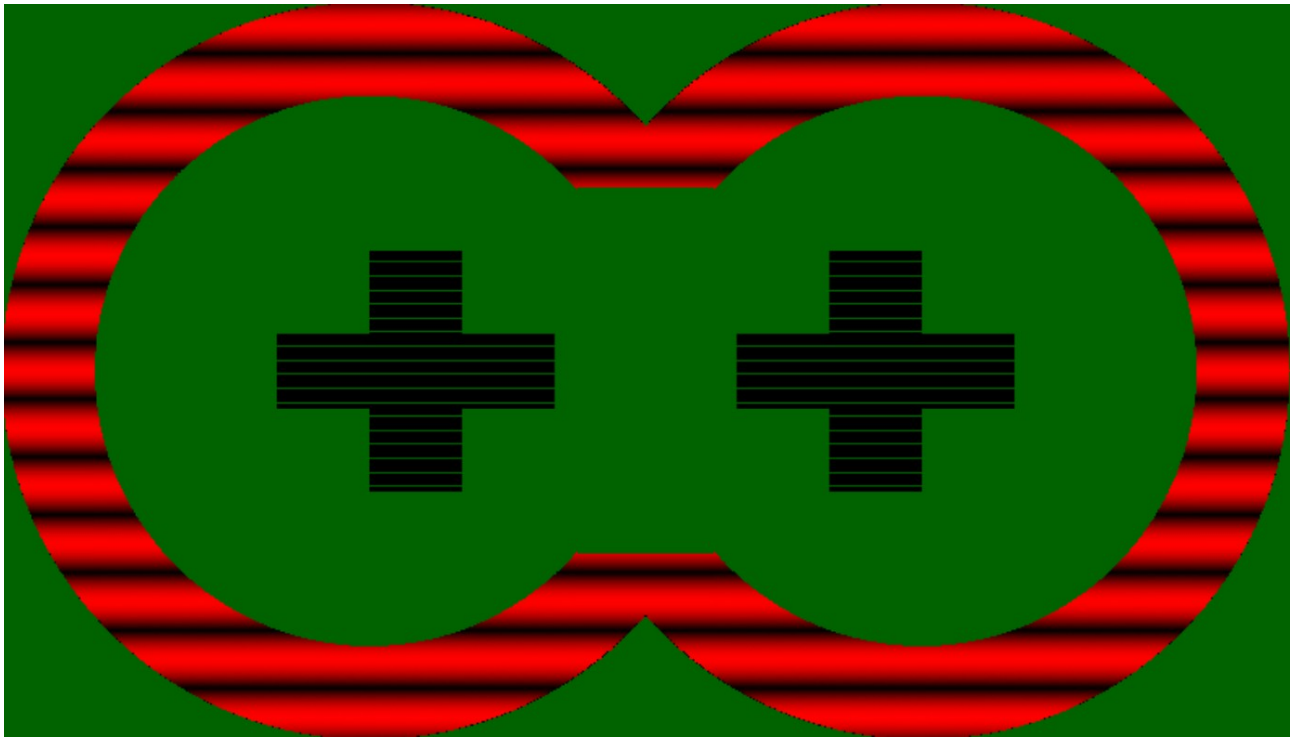
除了颜色值之外，这里除了为 2800 个元素中的每一个设置一个点三次之外，没有做其他事情。每个学生都会学习这个程序--制定一个数值表。在这里，这个简单的方法已经导致了一个可视化。虽然这里只输出了个别的点，但人们对功能的进展有印象。通过合适的++o 程序，不仅可以用图片按钮生成个别功能，还可以生成整个图片，如旗帜和颜色渐变。

方案 9.2：生成一个++o 的标志。

```
RGB:=深绿色

X3l :=-2 ...5!0.02

Y3l:=-2 ...2！0.02在X3

=: $BACKGROUND

从empty_t

Xl:=-2. ...1.5!0.02

Y1:= 4 - (X*X) sqrt

Y2:=如果X<-1.5则为0，否则如果X>1.13则为1，否则2.25-(X*X)sqrt

Zl:=Y2 ...Y1！0.02在Y2

RGB:=10*Z cos abs,,1,,0 sin abs leftat Z

STEP:=if -0.5 <X & X< 0. | (0.5<X & X<1.) then 0.2 else (if -0. <X & X< 0.5
then 0.65)

Wl :=0 ...LEVEL！0.01在LEVEL

RGB:=black leftat W #1,,10*W*X sin abs,0 leftat W

=: $OBENLINKS

$obenrechts:= $obenlinks *(-
1,1,1,1,1,1,1,1,1,1,1,1)+(3,0,0,0,0,0,0,0,0,0,0,0)

$downlinks:=$uplinks * (1,-1,-1,-1,1,-1,1,1,-1)
```

```
$untenrechts:=$obenrechts*(1,-1,-1,-1,1,1,1,-1,1,1,1,-1)
从$BACKGROUND,$OBENLINKS,$OBENRIGHT,$UNTENRIGHT,$UNTENLINKS开始
,<TAB!
X, Y l
-30 -30
TAB>
```

可以用图像按钮再次创建标志。



该程序目前需要相对较长的时间（超过一秒钟），所以你应该只在本地服务器上进行这样的计算。如果你细化步骤的大小，时间就会变得更糟。通过对程序的微小修改，人们可以获得图像的多方面变化。在第 12 节（学校的++o）中，给出了另一个生成图片的例子（12.16）。

10 图示

方案 10.1：计算每个年龄段的平均 BMI 和所有 20 岁以上的人的 BMI 和年龄，以及总体平均数。

```
<TAB!
名称，长度，（年龄，体重l）l
克劳斯 1.68 18 61
                    30     65
                                        56     80
罗尔夫 1.78 40 72
卡蒂 1.70 18 55
```

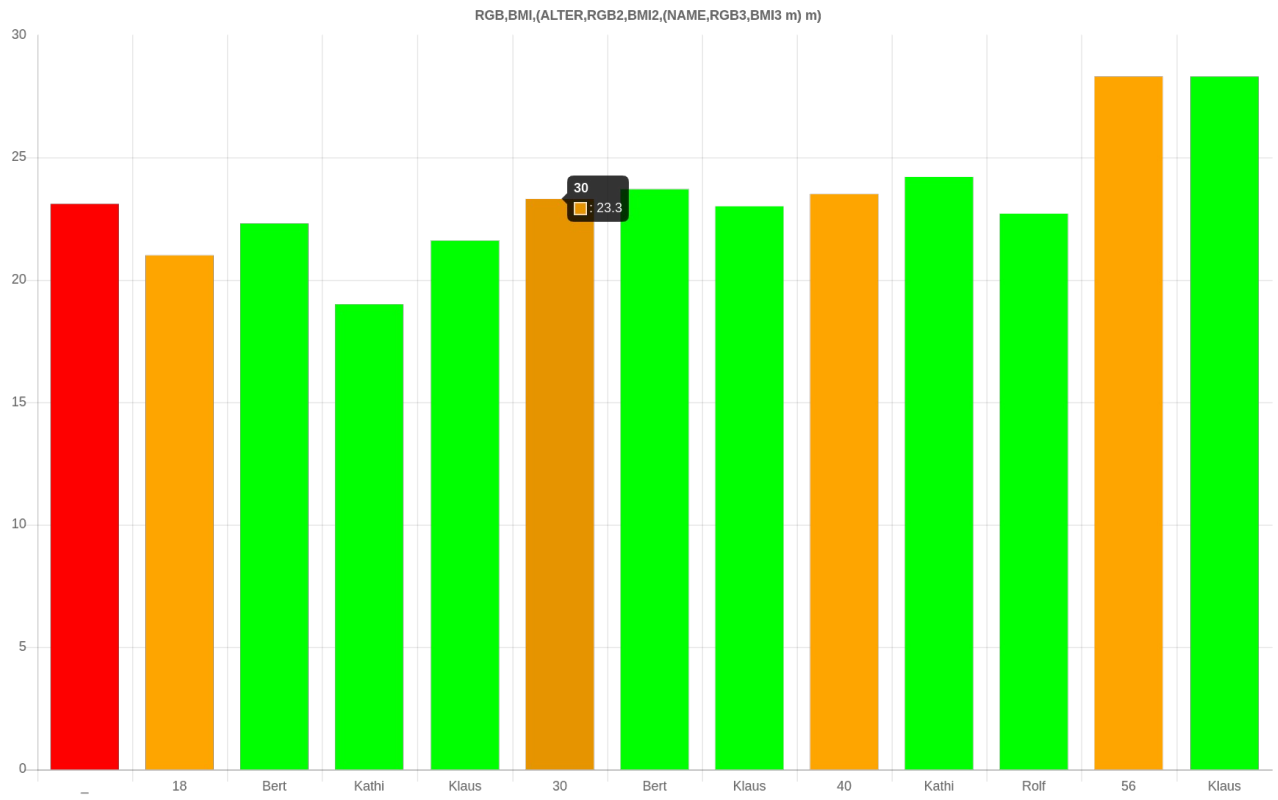
```
                    40     70
Walleri 1.00 3 16
维多利亚州 1.61 13 51
伯特 1.72 18 66
                    30     70
TAB>
avec NAME!20<替代物
给予BMI,(AGE,BMI,(NAME,BMI m) m) BMI:=WEIGHT:LENGTH:LENGTH!++:
第1轮
```

TAB 括号表示所包含的数据与 TAB 表示相对应。
上述条件选择人的记录，即 NAME,LENGTH,(AGE,WEIGHT l) 图元（结构化图元或 struples）。由于一个人有多个 ALTER，所以需要进行量化。名称！"。20<ALTER 因此选择了所有有相应年龄条目的人。也就是说，存在性量词没有被写入，而是属于每个条件。 在这个小例子中，人们当然也可以用手来实现选择。
结果。

```
BMI, (年龄, BMI2, (姓名, BMI3 m) m)
23.1 18 21. 伯特 22.3
                    卡蒂 19.
                    克劳斯 21.6
      30 23.3 Bert 23.7
                    克劳斯23岁。
      40 23.5 卡蒂 24.2
                    罗尔夫 22.7
      56 28.3 克劳斯 28.3
```

这个例子表明，只需指定所需的方案，就可以颠覆一个层次结构。其结果是，名字从属于年龄。
例如，通过两次点击可以将结果显示为柱状图。为了做到这一点，我们为每个级别分配了某种颜色。最后一行是必要的，这样年龄就不会显示为一列，而是显示为一个签名。

```
RGB:=红色左边的BMI
RGB2:=橙色在ALTER
RGB3:=绿色在NAME
ALTER::=ALTER字
```

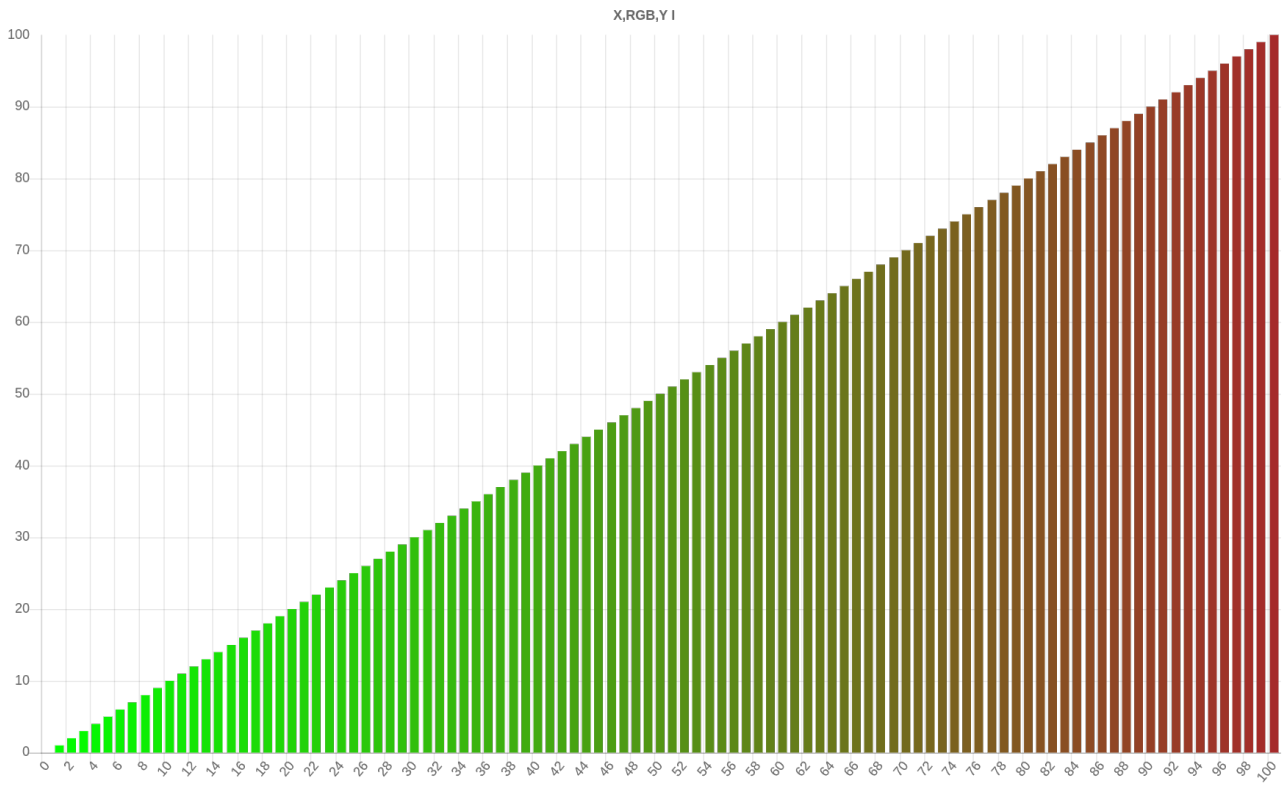


这里你可以看到基于结构化表格的图表的优势。

1. 上级栏目可以被突出显示（第二级为红色；上级第一级为橙色，第二级为绿色）。
2. 列名更短，更清晰，因为更高级别的列的值不必重复（例如，不需要 Bert30 和 Klaus30，只需要输出 Bert 和 Klaus）。

方案 10.2：一个从绿色到棕色的颜色梯度图。

```
Xl:=0 ...100

Y:= X在X

RGB:= (绿*(100-X) :100)+ (棕*X :100) 左图Y

X::=X字
```



11 "Hallo otto" - 噱头

方案 11.1：输出两个字。

你好，奥托

结果是两个词的列表（WORTl）。

方案 11.2： 发出一对两个单词。

你好，奥托

结果是一对 2 个字（WORD,WORD）。逗号将组件和图元分开。元组（对是 2 元组）通常是横向输出的，集合（列表、集、多集）的元素通常是纵向输出的。为了节省空间，简单的集合由 tabh-按钮水平输出。

方案 11.3:输出一个带空格的文本。

"你好，Otto"

结果的类型（模式）是 TEXT。

方案 11.4：连接两个单词。

你好+奥托

其结果是一个词。如果两个输入值都是词或文本，"+"也可以作为文本或词的链接操作（连接）。

方案 11.5：用一个单词表打招呼。

```
问候语:=[你好，奥托]
```

其结果是一个由两个词组成的列表。

方案 11.6：输出两个带有两列名称的字。

```
爱：=HELLO
GREETING:=otto
```

结果是一对（2-tuple）的单列表。该计划是：LIEBER,GRUSS

方案 11.7：输出一个带有标签（列名）的文本。

```
问候语：="你好，奥托"
```

结果。

```
问候

你好，奥托
```

方案 11.8：对一组单词进行分类。

```
GREETING:={otto Hello}
```

结果以导师的形式出现。

```
<<グリート>>

你好

otto

</GRUSS>
```

方案 11.9：对一组单词进行排序。

```
问候语：=奥托你好
```

结果以导师形式出现。

```
<TABM>

   <GRUSS>你好</GRUSS>。

   <问候语>otto</问候语>。

</TABM>
```

集合和多集合总是被排序的；元素的顺序对于列表来说保持不变。

12 o++o 的学校

联邦大法官 A.默克尔曾多次表示。"每个学生除了阅读、算术和写作之外，还应该学习编程"。现代信息社会对学生参与社会和未来工作生活的要求中，编程技能是一个重要方面。我们相信，作为一种易于使用的面向表格的编程语言，++o（详细的说是 ottoPS）是实现这一目标的正确钥匙。

o++o 没有循环。尽管如此，++o 还是很有表现力。你不仅可以用它来进行紧凑的查询，还可以对结构化表格和结构化文件进行各种计算。

o++o 使用并教授许多数学概念，因此我们看到在数学课堂上教授这些概念的主要好处，就像在数学课堂上使用计算器的基本技能一样。++o 特别使用以下概念。集合、多集、列表、平等和包含关系；图元；选择的强大操作；计算；重组、排序和聚合（总和；平均；...），...。
另一方面，EXCEL 等电子表格程序和标准数据库查询语言 SQL 并不了解任何结构化的表格方案。这种差异是++o 的决定性优势。
对学龄前儿童的首次测试表明，用结构表计算比用十进制数字计算更容易。我们希望增加更多的++o 样本程序。前两项对低年级学生来说可能很有趣。

方案 12.1：用单数（|）计算 4 乘 3。

```
KINDl := Ernst Clara Ulrike Sophia
APFELl := | KIND的3倍
++1 苹果
```

你也可以输入[|||]，而不是|乘以 3。这在开始时可能更好，但在更大的数字上你会有问题。结果是 12，但最有趣的是程序的前两行之后的中间结果。

```
儿童，APPLE l

恩斯特|||

Clara | | |

乌尔里克 | |

索菲亚 | |
```

你可以看到，这里的乘法表示的是一个矩形的面积，这不再是小数乘法的情况了。

方案 12.2：用单数计算（3+4）*5。

```
Xl:= |乘以3

Xl:= | 倍 4
Yl:=|在X处乘以5
++1 Y
```

结果。

```
35
```

方案 12.3:如果 X 乘以 X 是 25，就可以确定 X。

```
Xl:=1 ...10

XMALX:= X*X
```

结果。

```
X, XMALX l

 1   1

 2   4

 3   9

 4  16

 5  25

 6  36

 7  49

 8  64
```

```
 9  81

10 100
```

类似于对数表，学生现在可以读出（选择）结果。后来，选择可以通过++o 实现。

```
伴随着 XMALX=25
```

或

```
伴随着 XMALX <= 25

最后一次
```

这种方法可以应用于广泛的问题。

方案 12.4：计算一个简单项的价值。

```
2*3+4
```

结果。

```
10
```

*和+各自有 2 个输入值。首先，计算 2*3（6）。6 是+的第一个输入值，结果是 10。在这里，我们只需从左到右进行计算。

方案 12.5：用 o++o 写出 cos3(sin2(3.14159))这个项。

```
圆周率 sin 到 2 的次幂  cos 到 3 的次幂
```

结果。

```
1.
```

在我们看来，最初的术语对普通人来说是很难读懂的。你从 π 开始，向左移动到 sin；然后向右移动到 2 的幂；现在你再向左移动到 cos，最后向右移动到 3 的幂。实际上，为了不含糊，最初的术语应该有以下外观。

$(\cos((\sin(3.14159))^2))^3$

这当然更难读懂，你从左到右，反之亦然，甚至更频繁。

方案 12.6：用 o+o 写出 sin2(x)+cos3(y)项。

```
X sin 到 2 的次方 + (Y cos 到 3 的次方)
```

结果。

```
空_t
```

人们可以不使用括号，用++o 来写所有术语，但这样一来，某些术语就必须写在多行中，而且必须使用赋值。

没有结果出现，因为 X 和 Y 没有价值。例如，这可以按以下方式进行修改。

```
X:=2

Y:=3

Z:=X sin to the power of 2 + (Y cos to the power of 3)
```

结果。

```
X, Y, Z

2  3  -0.14345512749
```

方案 12.7：如何计算术语 2+3:4*5？

```
2+(3:(4*5))=2 3/20

2+((3:4)*5)=5 ¾
```

o++o：（（2+3）：4）*5=6 1/4

特别是，人们可以看到，"点的计算先于线的计算 "的学校智慧仍然是不够的。也需要 "从左到右 "的规则。

方案 12.8：计算几个音符的平均值。
1 2 3 1 2 ++:

结果。
1.8

从方法论的角度来看，人们仍然可以通过增加列表的括号来改进这个方案。[1 2 3 1 2] ++:
我们现在可以看到，平均运算++:有一个输入值，即一个列表，而++:被放在这个输入值之后。由于用户通常不愿意多打字，我们假设第一种符号在实践中会被更多地使用。

方案 12.9：计算每个科目的结构化表注.tabh 的平均数。
笔记.TABH

DUR:= NOTEl ++:

notes.tabh 可能看起来像这样。
FACH,NOTEl l

马 1 2 1 3 1 2

Phy 4 3 2 1

再次以 "tabh 格式 "请求的结果。
FACH, DUR, NOTEl l

马 1.666666667 1 2 1 3 1 2

Phy 2.4 4 3 2 1

方案 12.10：形成 1 至 100 的数字之和（高斯的任务五年级）。
1 ..100 ++

像加法和乘法一样，"... "有两个输入值（1 和 100）。中间的结果是列表
号码

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43

44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63

64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83

84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

然后将这些数字相加，得出 5050。

方案 12.11：大约计算正弦函数在区间[1，2]中的最大值。
1 ...2！0.001 sin max

„. . ."需要 3 个输入值：1.开始值 1，结束值 2 和步长 0.001，这导致数字 1 1.001 1.002 1.003 ... 1.999 2.
正弦函数被应用于每个数字，结果又是 1001 个数字。然后，max 函数从这个列表中选择最大值。

sin 和 max 各有一个输入值（这里是一个列表），但 sin 的输出值又是一个列表，max 只产生一个数字，因为这是一个聚合函数。三位数运算的第二和第三位输入值（以上 ...）各以"！"隔开。这在++o 中是必要的，因为配对用的逗号已经分配好了，而且空格已经分隔了列表元素。

方案 12.12：近似地计算多项式 "X3+4X2-3X+2 "在区间[0，2]内的最小值和相关的 X 值。

```
Xl:=0 ...2!0.001

Y:=X 聚[1 4 -3 2]

MINI:=Yl min

与 Y=MINI
```

avec 是法语，表示一种选择。一个变量 X 的具体多项式总是只有一个输入值，用于 X.另一方面，第 2 行的 poly 更普遍，有两个输入值。
1.X 的输入值，这里取第一行中产生的所有数字。
2.对应于具体多项式系数的数字列表。
第一行创建了一个数字列表，所有这些数字都被命名为 X。这在 xml 或 ment 表示中得到了最好的认可。

```
<X>0.</X>

<X>0.001</X>

<X>0.002</X>。

…
```

总体结果。

```
MINI, (X, Y l)

1.481482037 0.333 1.481482037
```

方案 12.13：计算余弦函数在区间[1，2]内的一个零点近似值。

```
Xl:=1 ...2!0.0001

在 X cos < 0 的情况下

伴随着 X pos = 1
```

结果。

```
Xl

1.5708
```

在这里，经过第一次选择后，只剩下函数值小于 0 的 X 值，其中第一个值在第二步中被选中。由于我们知道 cos 在所考虑的区间内只有一个零，所以这是个近似的结果。

方案 12.14：给定 5 个年度增长数字，计算总增长。将结果四舍五入到小数点后一位。

```
Wl:=0 1.5 2.1 1.3 0.4 1.2

ACCU:=前 100 名。下一个 ACCU 预测*（W:100+1）在 W

第 1 轮

结果表。
W, ACCU l

0.  100.

1.5 101.5
```

```
2.1 103.6

1.3 105.

0.4 105.4

1.2 106.7
```

第一个 ACCU 值是由第一个（100.）之后的表达式产生的。对于第二个值，值 100.被插入 ACCU 预测，并根据下一步对该术语进行评估。结果是 101.5。这个数字再次被插入 ACCU pred，然后再次计算下一个项（大约 103.6），.....直到达到最后一个 W 值。 pred 是前任。

方案 12.15：计算正弦曲线在区间[0，pi]内的面积近似。

```
0 ...pi！0.0001 sin *0.0001 ++
```

结果。

```
1.99999999867
```

在这里，0 和 π 之间的所有数字都是一个接一个产生的，然后从每个数字中计算出正弦，然后每个数字乘以 0.0001。结果是 10000 个矩形区域，然后将它们加在一起。

方案 12.16：帕斯卡尔三角形

```
Xl:=1 ...9

Y:=第一个 1 下一个 Y 预测,0 + (0,Y 预测) 在 X
```

结果。

```
X, Y l

X Y

1 1

2 1 1

3 1 2 1

4 1 3 3 1

5 1 4 6 4 1

6 1 5 10 10 5 1

7 1 6 15 20 15 6 1

8 1 7 21 35 35 21 7 1

9 1 8 28 56 70 56 28 8 1
```

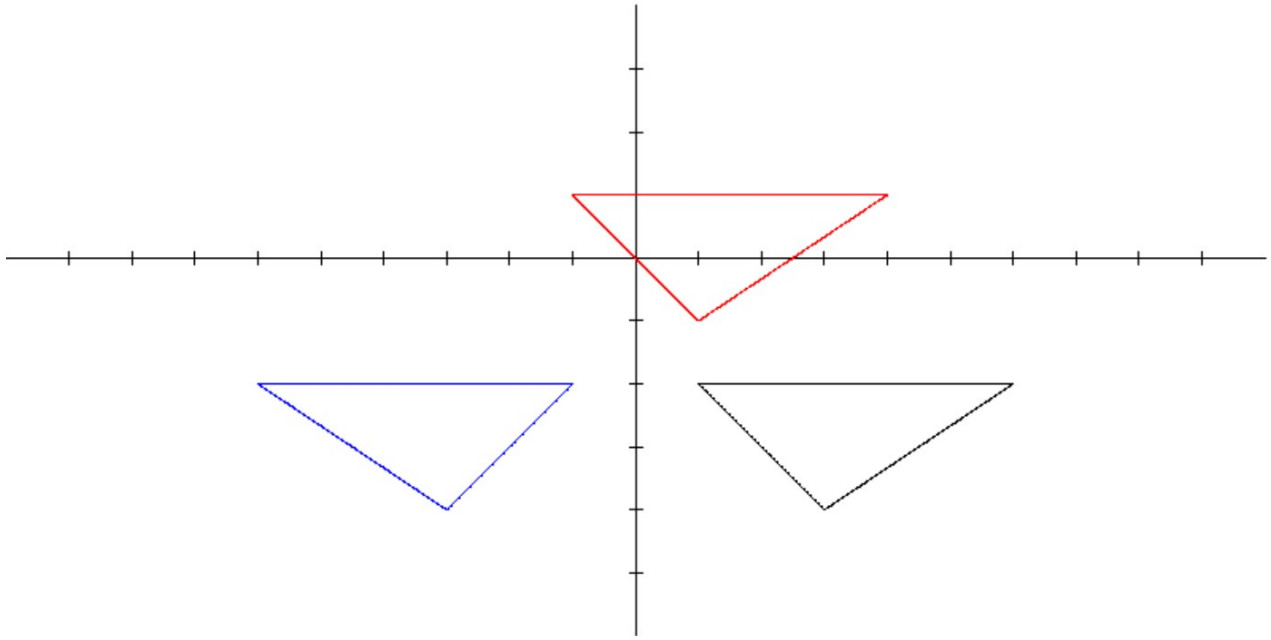
这里的 Y 代表一个元组的值。因此，其结果不再是一个 "正常 "的表。因此，它不能再以标签形式输出。这里选择的是 hsq 输出。

方案 12.17：移动和镜像一个三角形

```
# Class7 p.14 No.5 (red) Saxony-Anhalt Secondary School

# 数学课本

# 请点击图片

Xl:= -10 ...10!0.02

Y:=0*X

X0l:=0

Y0l:=-6 ...4！0.02 在 X0 处
```

```
X1l:= -9 ...9
Y1l:=-0.1 ...0.1！0.01在X1
X2l:=-0.1 ...0.1!0.01
Y2l:=-5 ...3在X2
=: $coordinates
从<TAB!
X3,Y3 l
1  -2
3  -4
6  -2
1  -2
TAB>
航线
=: $TRIANGLE
来自$COORDINATES ,$TRIANGLE
RGBROT:=红色
,$TRIANGLE +(-2,3) #移动
RGBBlue:=蓝色
,$TRIANGLE *(-1,1) # 镜像至Y轴
```

点击图片后的结果。



上述例子特别表明，这些任务不需要微分和积分的知识就可以解决。有了++o，数学课可以在许多方面得到支持。它涉及：自然数、小数、任何大小的有理数的计算，任何函数的零点的近似计算，推导，曲线下的面积，极值（在中学已经可以教了），概率计算，.....。通过++o，可以用简单的方式计算出一些本来只能在理论上处理的事情。因此，对这些概念的理解可以得到明显的改善、扩展和深化。关于++o 的更多信息可在 **ottops.de** 上找到。
我们相信++o 为数学和计算机科学课提供了特别的优势，但也可以在其他学科中得到有益的运用（今后可向维基百科查询）。

13 结束语，引用亚当-里斯的一段话

一个人对他来说，数字(tabment)[1)]是隐藏的
容易被狡猾所诱惑的人
我求你把这句话记在心里
各人教他的孩子计算（程序）[1)]……。

亚当-里斯
1) 作者的补充

14 文学

[AB84] S. Abiteboul, N. Bidot, "Non First Normal Form Relations: An Algebra Allowing Data Restructuring" Rapports de Recherche No347, Institute de Recherce en Informatique et en Automatique, Rocquencourt, France, November 1984.

[AC75] M. M. Astrahan, D. D. Chamberlain, "Implementation of a Structured English Query Language", Communications of ACM 18 10, Oct. 1975 pages 580-587.

[BCFFRS10] **Scott Boag,** Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, XQuery 1.0: An XML Query Language (Second Edition), W3C

Recommendation 14 December 2010 (Link errors corrected 3 January 2011), http://www.w3.org/TR/2010/REC-xquery-20101214/

[Ben80] K. Benecke, "Signature Chains and Operation and Set Forms over Signature Chains", Dissertation A, TH Magdeburg 1980.

[Ben82] K. Benecke."AFUL-- 一种数据库的查询语言"。 In Weiterbildungszentrum für mathematische Kybernetik und Rechentechnik/Informationsverarbeitung, Studientexte Datenbanken TU Dresden Heft 59, Seiten 100 - 107, 1982.

[Ben88] K. Benecke, "Hierarchical Data Structures", Habilitation, Technische Hochschule 马格德堡, 1988

[Ben91] K. Benecke, "A powerful Tool for Object-Oriented Manipulation", in Object-Oriented Databases: Analysis, Design & Construction (DS-4) R.A. Meersmann, W. Kent, S. Khosla (editors), Elsevier Science Publisher B.V. (North Holland) 1991, pp.95-121。

[Ben98] K. Benecke, "Structured Tables - A New Paradigm for Database and Programming Languages", Deutscher Universitätsverlag, ISBN 3-8244-2099-6 Wiesbaden 1998.

[Ben16] K. Benecke, "++oPS The simplest Programming Language", BoD, 2016, ISBN 978-3-7412-4281-6.

[BH11] K. Benecke, A. Hauptmann, "学校需要一种表格计算机语言吗？" http://www.infonomics-society.org/IJDS/Does%20the%20School%20Need%20a %20Tabular%20Computer%20Language.pdf，第 520-527 页，2011 年。

[Cod70] E.F.Codd, "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, Vol.13, No.6 June 1970, pp.377-387.

[Hau10] A. Hauptmann，"OttoQL：非关系型数据库语言的实施问题 （特别考虑到逻辑优化）"。 学生研究项目，马格德堡大学，Beneck-systeme，2010。

[KR71] H. Kaphengst, H. Reichel, "Algebraic Algorithm Theory", VEB Robotron, Wiss. Informationen und Berichte, No. 1/71 Series A, Summer 1971.

[Rei87] Reichel.H., Initial Computability, Algebraic Specifications, and Partial Algebras, Oxford 英国，Claredon 出版社，1987

[Rei90] W. Reichstein."Implementation der Strichlistenoperation Operation stroke in C", Praktikumsbeleg, VE CS Magdeburg, 1990.

[Sch96] D. Schamschurko, "Implementation des Rumpfes des GIB-AUS-MIT-Konstrukts in CAML-Light", Praktikumsbeleg, DeTeCSM Magdeburg, Supervisor: K. Benecke, March 1996, 123 pp。

[SHL75] N. C. Shu, B. C. Housel, V. Y. Lum, "CONVERT--数据转换的高级翻译定义语言", Communications of the ACM, Vol.18 No.10,Oct., 1975, pp.557-567。

[Ull82] J. D. Ullman, "Principles of Database Systems", Computer Science Press, Rockville, Maryland 1982.

[Zem85] H. Zemanek, "Formal Definition the Hard Way", Proc. IFIP TC2 Working Conference, Vienna 1985; North Holland, pp.411-417。