fChart 6 使用手册

https://fchart.github.io

* 安装、设定与启动 fChart
* 建立第一个fChart 流程图
* fChart 流程图的直线、L 型和 U 型连接线
* 开启与执行fChart 流程图
* fChart 分类 150 个流程图项目
* 编辑 fChart 流程图项目
* fChart 流程图符号的对话框
* fChart 程序代码编辑器与Blockly 积木程序

2 fChart 程序设计教学工具 - https://fchart.github.io

安装、设定与启动 **fChart**

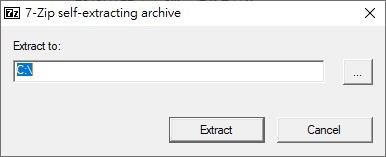
fChart 6.0 版不只支持多国语言的使用接口，更重新改写流程图的连接线功能，在新版本共支持五种连接线：直线、先水平再垂直 L 型线、先垂直再水平 L 型线、顺时钟和反时钟的 U 型线。

请注意！虽然新版兼容旧版项目，但新版建立的流程图项目，在旧版 **fChart** 开启时不会错误，但是因为只支持直线，所有 **L** 或 **U** 型线都只能显示直线，其显示的流程图可能会有很大的差异。

fChart 程序语言教学工具是一套程序设计的教学工具，也是一套轻量级的整合开发环境（内建 Blockly 积木程序），主要分成两大工具： 流程图直译器和程序代码编辑器。

# 安装fChart

fChart 新版本已经改为 7-Zip 格式的自解压缩文件，以标准版下载的fChart6.exe 为例，只需执行下载的执行文件，即可将相关程序档案解压缩至指定目录（工具并不需安装，解压缩后即可使用），如下图所示：



上述对话框是 7-Zip 自解压缩对话框，因为压缩文件已经包含

「fChart6 」目录， 在字段只需输入硬盘【C:\】或【D:\】即可， 按

【Extract】钮即可解压缩至此硬盘下的「fChart6」目录，以 C:\为例，

fChart 6 使用手册 3

标准版是「C:\fChart6」目录；Python 版预设是「C:\fChartPython6」；

Node 版是「C:\fChartNode6」，在目录下的主要档案说明，如下所示：

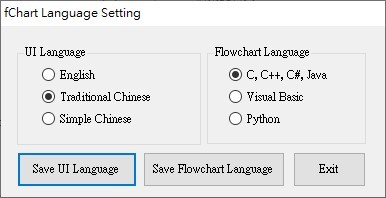
* **RunfChart.exe**：使用系统管理员身份启动 fChart 流程图直译器。
* **FlowProgramming\_Edit.exe**：fChart 流程图直译器的执行档，因为权限问题，直接执行档案，需在档名上执行【右】键快捷菜单的

【以系统管理员身份执行】命令，使用系统管理员身份来执行。

* **fChartSetting.exe**：fChart 使用接口和流程图语言的设定程序。
* **fChartCodeEditor.exe**：fChart 程序代码编辑器的执行档。

# 设定fChart 接口语言和流程图的程序语言

fChart 支持多国语言使用接口，默认接口是繁体中文，流程图是C、C++和 Java 语言。请执行安装目录下的【fChartSetting.exe】来设定使用的接口语言，支持繁体中文、简体中文和英文，如下图所示：



在上述对话框的左边是使用接口语言，在选择后，请按【Save UI Language】钮储存接口语言设定；右边是流程图使用的语言（影响运

4 fChart 程序设计教学工具 - https://fchart.github.io

算子），在选择程序语言后，请按【Save Flowchart Language】钮储存设定，在完成设定后，请按【Exit】钮结束设定程序。

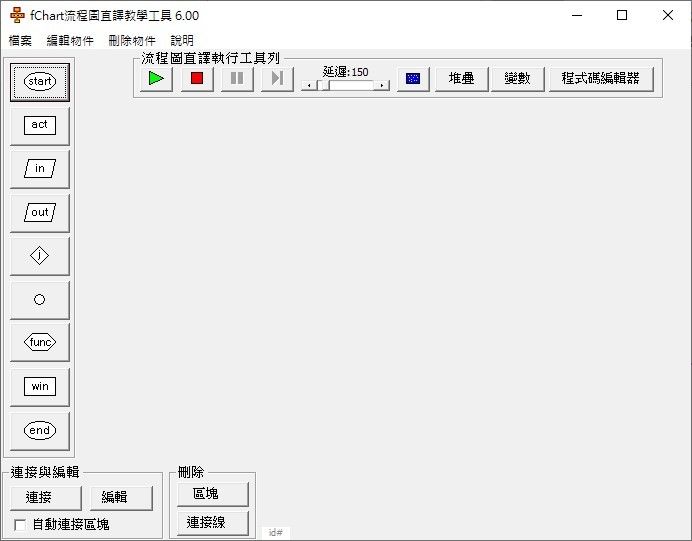
# 启动fChart 流程图直译器

在解压缩和设定 fChart 后，我们可以马上在 Windows 操作系统执行 fChart 流程图直译器（从此工具可以启动 fChart 程序代码编辑器），其步骤如下所示：

Step 1：请开启 fChart 所在的「\fChart6」目录，执行【RunfChart.exe】后，按【是】钮启动 fChart 流程图直译器。

Step 2：在成功启动 fChart 流程图直译器后，可以看到流程图编辑的使用接口。

菜单



流程图符号图标钮

流程图编辑区域 执行工具栏

连接、编辑和删除按钮

fChart 6 使用手册 5

上述图例是 fChart 流程图直译器的使用接口，在上方是菜单，菜单下方是执行工具栏，可以执行我们绘出的流程图，左边是建立流程图符号图标的按钮工具栏，在下方是连接、编辑和删除图标符号的按钮，位在中间区域就是编辑建立流程图的编辑区域。

# 结束fChart 流程图直译器

请执行「档案>结束」命令结束 fChart 流程图直译器，或是按窗口右上角【X】钮关闭流程图直译器。

建立第一个 **fChart** 流程图

在启动 fChart 流程图直译器后，我们可以马上开始绘制第一个流程图，fChart 流程图直译器提供十分容易的方式来绘制流程图。

# 流程图的基本绘制原则

为了绘制良好的流程图，一些绘制流程图的基本原则，如下所示：

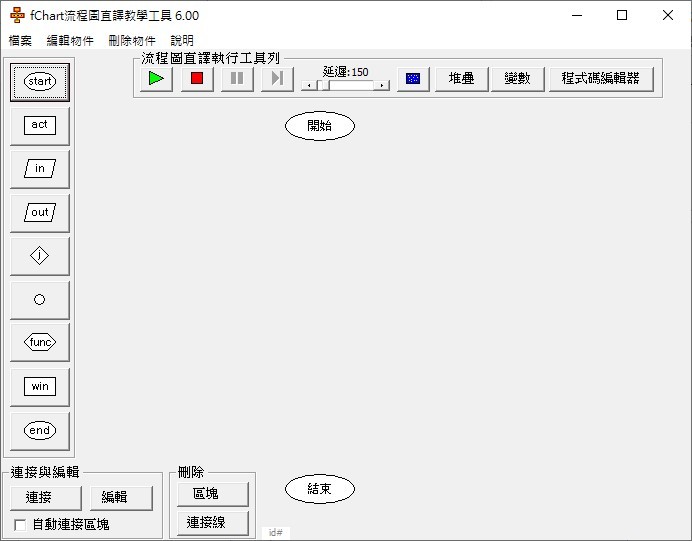
* 流程图请使用标准的图标符号，以方便阅读、沟通和小组讨论。
* 在每一个流程图符号的说明文字需力求简洁、扼要和明确可行。
* 流程图只能有一个起点，和至少一个终点。
* 流程图的绘制方向是从上而下；从左至右。
* 决策符号有两条出去连接线的流程符号；终止符号不允许有出去连接线的流程符号。
* 流程图连接线的流程符号应避免交叉或太长，如果有多条进入的连接线，可以使用连接符号来进行连接。

6 fChart 程序设计教学工具 - https://fchart.github.io

# 建立第一个 fChart 流程图

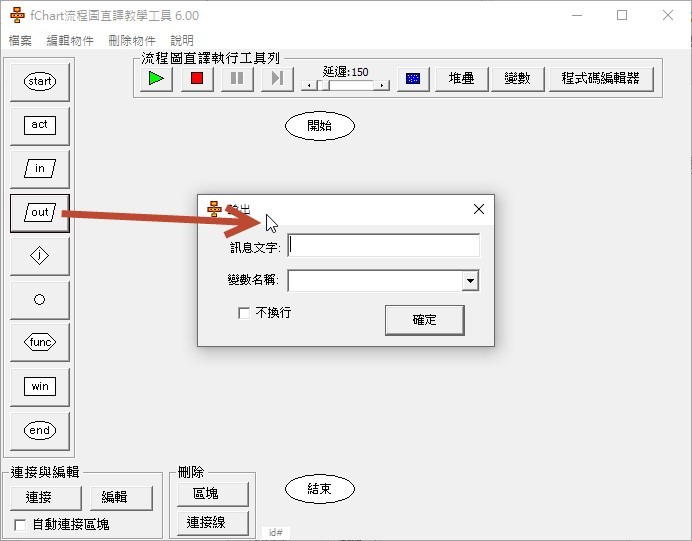
我们准备建立第 1 个 fChart 流程图来显示一段文字内容，即传统程序语言最常见的 Hello World 程序，其步骤如下所示：

Step 1：请启动fChart 流程图直译器，执行「档案>新增流程图项目」命令，可以看到新增的流程图项目，预设新增开始和结束 2 个符号。

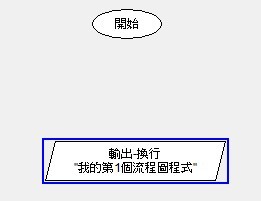
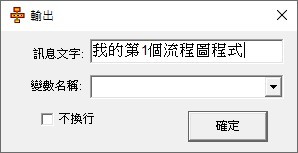


Step 2：在左边垂直工具栏，选第 4 个 out 输出符号后，拖拉至插入位置，点选一下，开启「输出」对话框。

fChart 6 使用手册 7

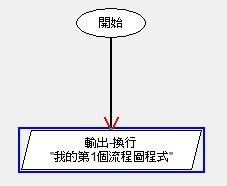
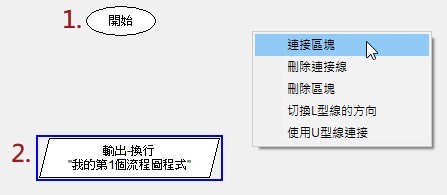


Step 3：在【消息正文】栏输入欲输出的文字内容【我的第 1 个流程图程序】，如果有输出变量值，请在下方【变量名称】字段输入或选择变量名称，按【确定】钮，可以看到新增的输出符号。

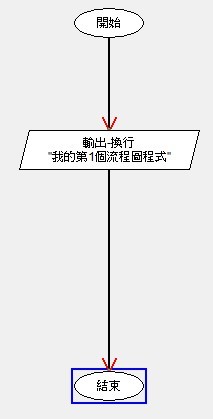
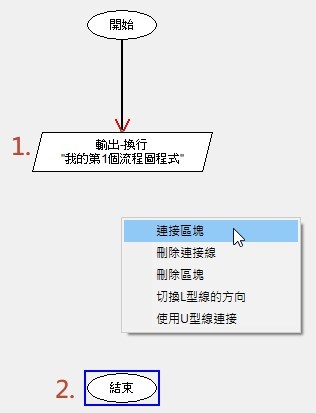


Step 4：接着连接流程图符号，请先点选「开始」符号，然后是「输出」符号，在没有符号的区域，执行鼠标【右】键快捷菜单的【连接区块】命令，可以新增开始至输出符号间的连接线，红色箭头是执行方向。

8 fChart 程序设计教学工具 - https://fchart.github.io

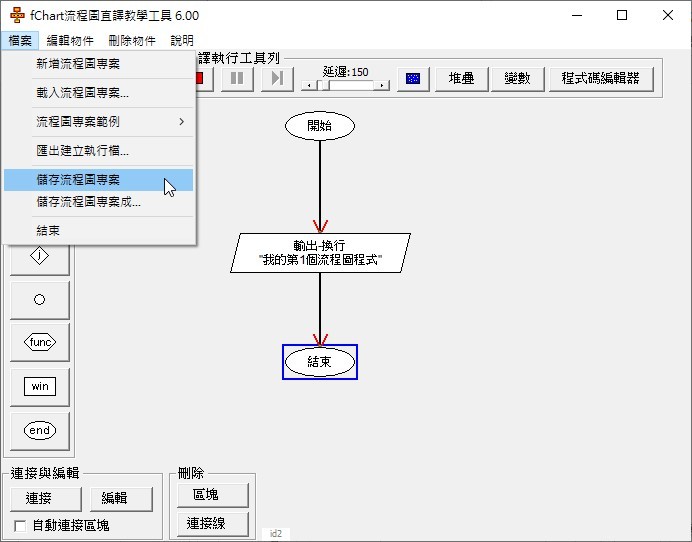


Step 5：然后点选输出符号，再选结束符号，在没有符号区域，执行鼠标【右】键快捷菜单的【连接区块】命令，新增输出至结束符号间的连接线，红色箭头是执行方向。



Step 6：在拖拉调整流程图符号的位置后，即可完成 fChart 流程图的绘制。

fChart 6 使用手册 9



Step 7：请执行「档案>储存流程图项目」命令储存流程图项目，可以看到 「 另 存 新 檔 」 对 话 方 块 ， 请 切 换 路 径 和 输 入 檔 案 名 称

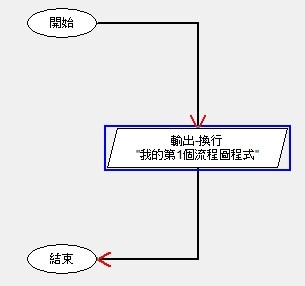
【FirstProgram.fpp】后，按【存档】钮储存项目，扩展名是.fpp。

**fChart** 流程图的直线、**L** 型和 **U** 型连接线

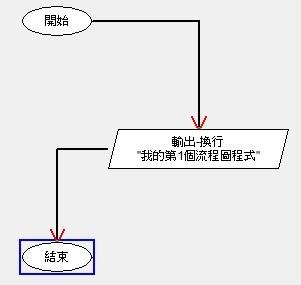
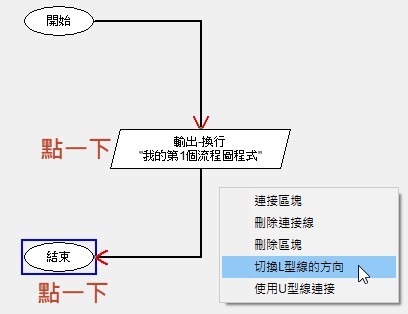
fChart 默认能够自动依据符号位置来调整是否使用 L 型连接线（自动 L 型线），如果符号太接近，空间不足是使用直线；空间足够使用 L 型线，当在起点坐标已经有其他连接线（fChart 预设不允许出去和进入连接线是重迭线；多条进入线是可以是重迭线），就自动切换成相反方向的 L 型线，如下所示：

* 自动 **L** 型线：当 2 个符号位置如果是终点符号是位在起点符号的右下方；L 型线预设是先水平再垂直 90 度；如果是位在右上方；左下方；左上方，预设是先垂直再水平 90 度的L 型线，如下图所示：

10 fChart 程序设计教学工具 - https://fchart.github.io



* 切换 **L** 型线的方向：在连接线的 2 个符号各点选一下（顺序没有关系）后，执行「编辑对象>切换 L 型线的方向」命令, 或【右】键快捷菜单的【切换 L 型线的方向】命令，可以切换成先水平再垂直L 型线或先垂直再水平的 L 型连接线，如下图所示：

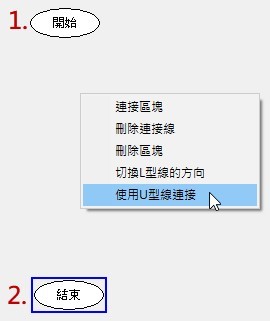


请注意！**L** 型连接线会自动判断符号的起点是否已经有其他进入的连接线（有重迭连接线），如果有，就会自动切换至反方向的 **L** 型线， 此时，因为反方向 **L** 型线的起点有重迭连接线，**fChart** 预设不允许出去和进入连接线是重迭线，所以不能手动切换 **L** 型线的方向，如需切换方向，请先调整其他符号的连接线，移开造成相同点的其他连接线后，即可自动更新或手动切换 **L** 型连接线的方向。

fChart 6 使用手册 11

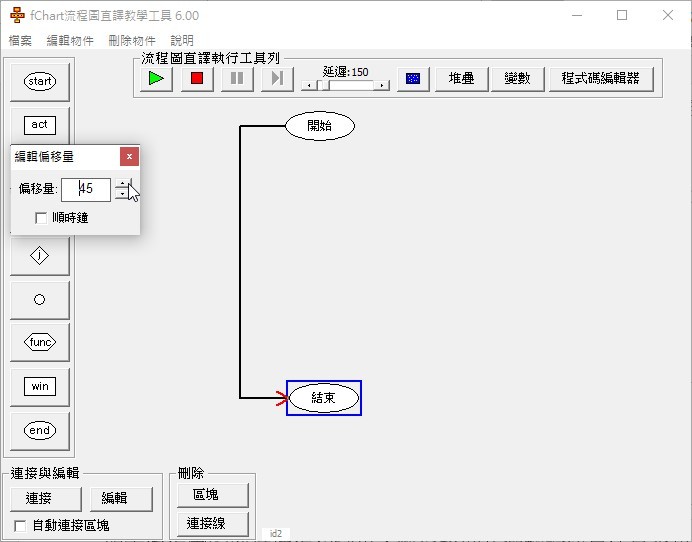
* **U** 型线： 如同其他连接线， 在欲连接的 2 个符号各点选一下

（顺序是先开始；然后结束）后，执行「编辑对象>使用 U 型线连接」命令，或【右】键快捷菜单的【使用 U 型线连接】命令, 预设建立使用偏移量 40 建立反时钟连接的 U 型连接线，同时显示对话框来调整偏移量，最小值是 20，如下图所示：



U 型线预设是使用反时钟方向来连接 2 个符号，同时显示「编辑偏移量」对话框，我们可以调整 U 型线的偏移量，如果勾选下方【顺时钟】的复选框，可以切换成顺时钟方向的 U 型线连接，如下图所示：

12 fChart 程序设计教学工具 - https://fchart.github.io



在建立 U 型线后，如需编辑 U 型线，我们可以在连接线的 2 个符号各点选一下（顺序没有关系）后，执行「编辑对象>编辑 U 型线的偏移量」命令来更改偏移量和 U 型线的连接方向。

* 设定使用直线：如果一定需要使用直线，不使用 L 型或 U 型线，请在欲设定的 2 个符号各点选一下（顺序没有关系）后，执行「编辑对象>使用直线的连接线」命令，就可以改用直线的连接线来连接 2 个符号，
* 将 **U** 型线或直线重设为自动 **L** 型线：如果连接线已经指定成 U 型或直线，欲重设成预设的自动 L 型线，请在欲设定连接的 2 个符号各点选一下（顺序没有关系）后，执行「编辑对象>切换 L 型线的方向」命令，可以再切换成预设的自动 L 型连接线。

fChart 6 使用手册 13

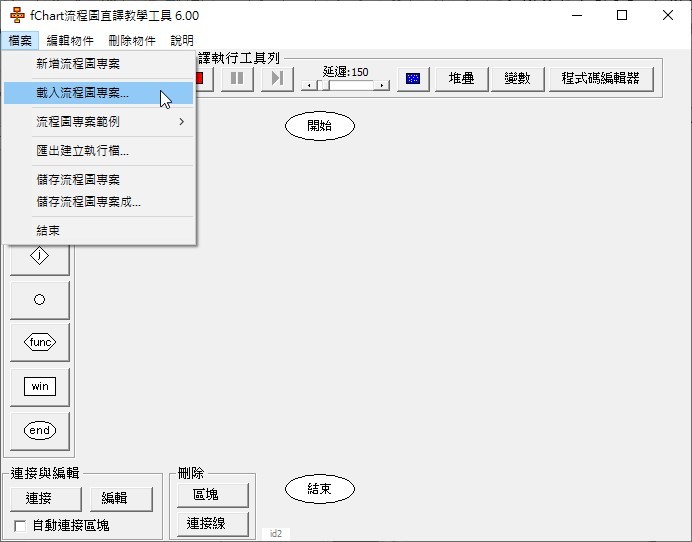
开启与执行 **fChart** 流程图

fChart 流程图直译器不只可以编辑绘制流程图，还可以使用动画来完整显示流程图的执行过程和结果，轻松帮助我们验证算法是否可行，和训练初学程序设计者的程序逻辑。

# 开启fChart 项目执行流程图

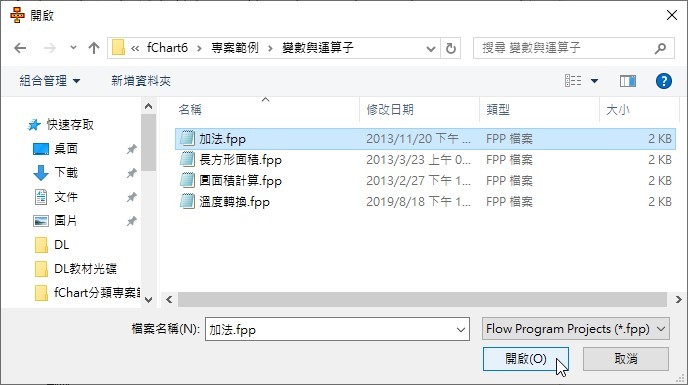
对于已经存在的 fChart 流程图项目，我们可以加载流程图项目来执行，其步骤如下所示：

Step 1：请启动 fChart 流程图直译器，可以进入流程图编辑的使用接口。

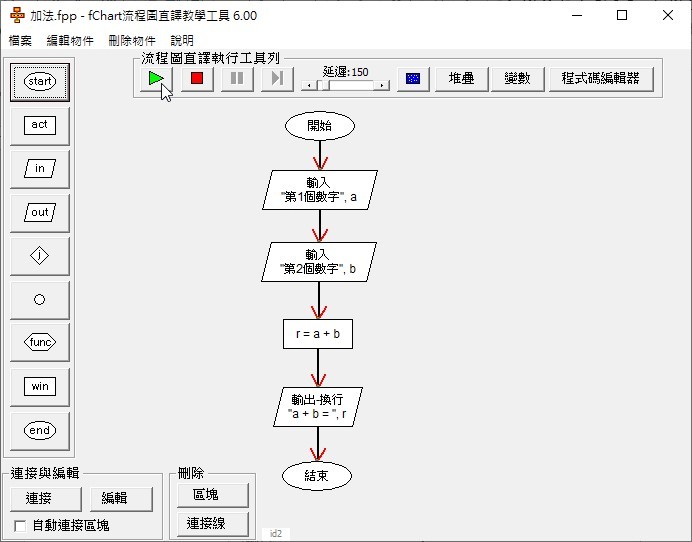


Step 2：请执行「档案>加载流程图项目」命令，可以看到「开启」对话框。

14 fChart 程序设计教学工具 - https://fchart.github.io

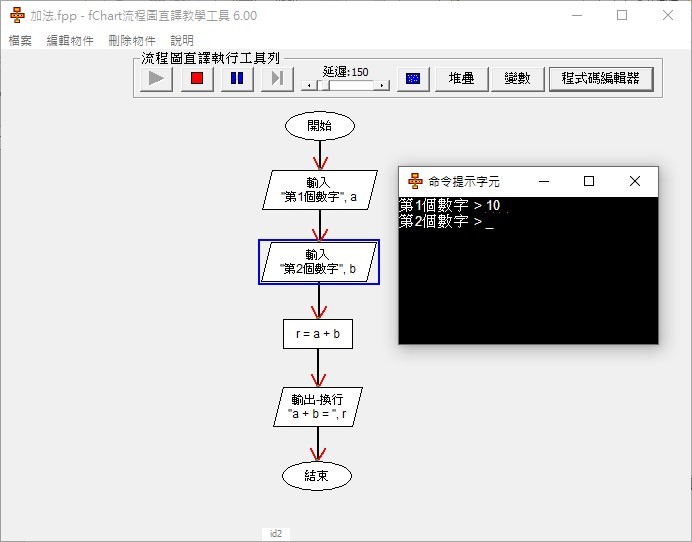


Step 3 ： 切换至「\fChart6\ 项目范例\ 变量与运算符」路径， 选【加法.fpp】，按【开启】钮加载流程图。



Step 4：按上方执行工具栏的第 1 个【执行】钮，可以看到动画移动蓝色框来执行流程图，因为执行到输入符号，所以显示「命令提示字符」窗口，和输入第 1 个数字的提示文字。

fChart 6 使用手册 15



Step 5：请输入 10，按 键，可以看到输入第 2 个数字，请输入

Enter

15，按 键，可以看到流程图继续执行，和显示执行结果 25。

Enter



16 fChart 程序设计教学工具 - https://fchart.github.io

Step 6：我们可以再次执行流程图，并且输入不同值，就可以看到不同的执行结果。

# 流程图直译执行工具栏

fChart 流程图直译器是使用上方执行工具栏按钮来控制流程图的执行，我们可以调整执行速度和显示相关辅助信息窗口，如下图所示：

暂停 调整延迟时间



执行 停止

逐步执行

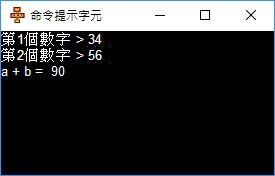
显示命令提示字符窗口

上述执行工具栏按钮从左至右的说明，如下所示：

* 执行：按下按钮开始执行流程图，这是使用延迟时间定义的间隔时间来一步一步自动执行流程图，如果流程图需要输入数据，就会开启「命令提示字符」窗口让用户输入数据（在输入数据后，请按

键）例如：【加法.fpp】，如下图所示：

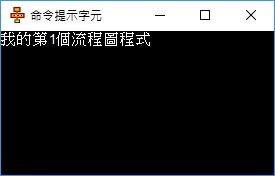
Enter



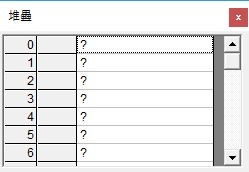
* 停止：按此按钮停止流程图的执行。

fChart 6 使用手册 17

* 暂停：当执行流程图时，按此按钮暂停流程图的执行。
* 逐步执行：当延迟时间卷动轴调整至最大时，就是切换至逐步执行模式，此时按【执行】钮执行流程图，就是一次一步来逐步执行流程图，请重复按此按钮来执行流程图的下一步。
* 调整延迟时间：使用卷动轴调整执行每一步骤的延迟时间，如果调整至最大，就是切换成逐步执行模式。
* 显示命令提示字符窗口： 按下此按钮可以显示「命令提示字符」窗口的执行结果，例如：FirstProgram.fpp，如下图所示：

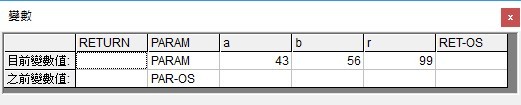


* 显示堆栈窗口：在「堆栈」窗口显示函数呼叫保留的局部变量值， 如下图所示：



18 fChart 程序设计教学工具 - https://fchart.github.io

* 显示变量窗口：在「变量」窗口显示执行过程的每一个变量值，包含目前和之前上一步的变量值，例如：【加法.fpp】，如下图所示：

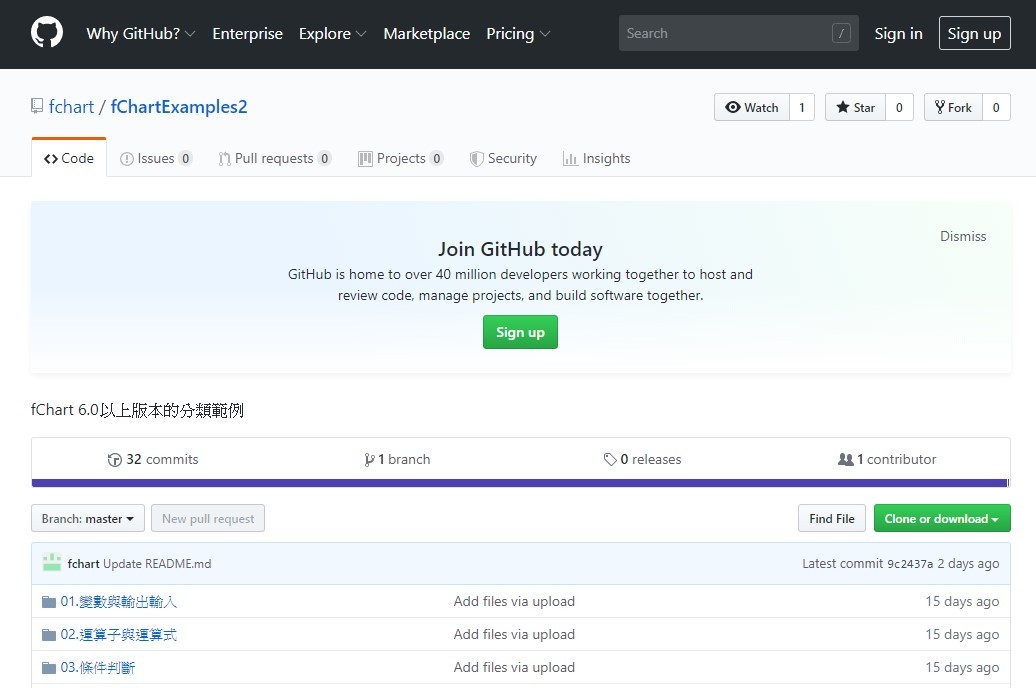


* 程序代码编辑器：启动fChart 程序代码编辑器。

**fChart** 分类 **150** 个流程图项目

fChart 在 GitHub 档案库提供有 150 个分类范例，请启动浏览器进入 6.0 版 fChart 分类 150 个流程图项目的 GitHub 档案库，如下所示：

* https://github.com/fchart/fChartExamples2



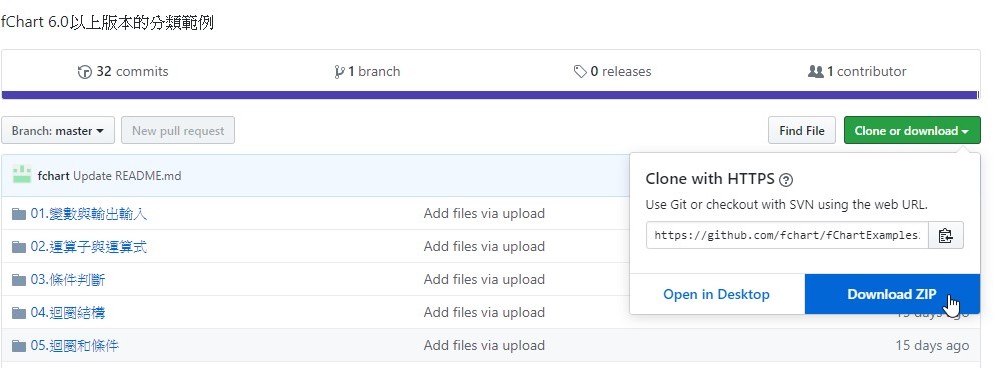
fChart 6 使用手册 19

上述 GitHub 档案库提供分成十类约 150 个 fChart 流程图项目，可以作为程序设计的学习单、作业或考试题目，因为提供的是可执行的流程图， 即程序的算法， 所以适用各种程序语言： C/C++ 、C# 、Python、Java、JavaScript、VB 和PHP 等，如下图所示：



# 下载整个GitHub 档案库

请在档案库首页，点选【Clone or download】钮，再选【Download ZIP】，即可下载整个GitHub 档案库，如下图所示：



20 fChart 程序设计教学工具 - https://fchart.github.io

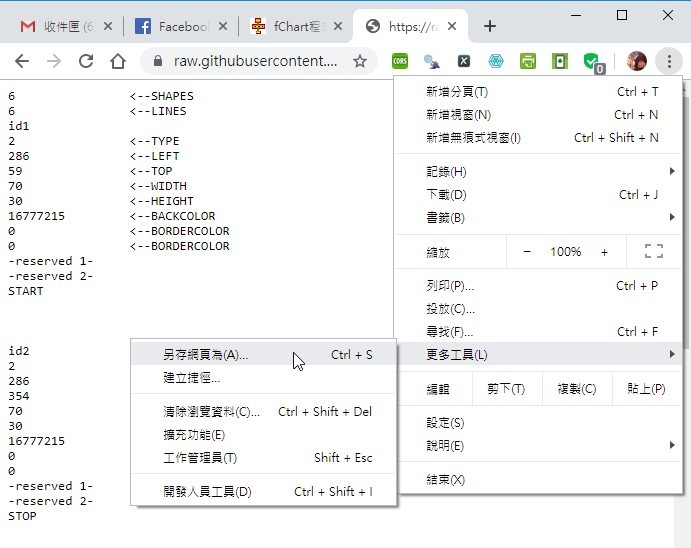
# 下载指定的 fChart 流程图项目

如果是下载特定的 fChart 流程图项目， 例如：【01.单选\_绝对值.fpp】，请点选目录切换至此.fpp 档案，点选【Raw】钮，如下图所示：



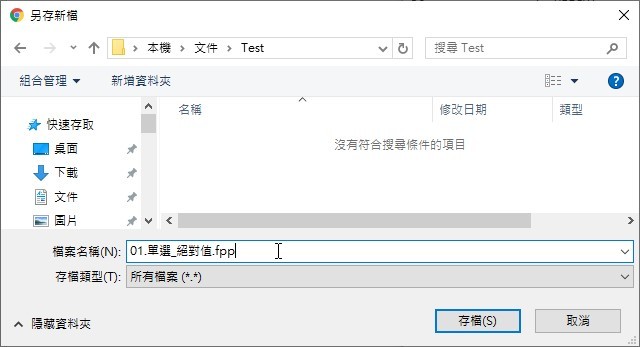
在显示.fpp 的档案内容（.fpp 文件格式是文本文件）， 请执行

Chrome 浏览器的「更多工具>另存网页为」命令，如下图所示：

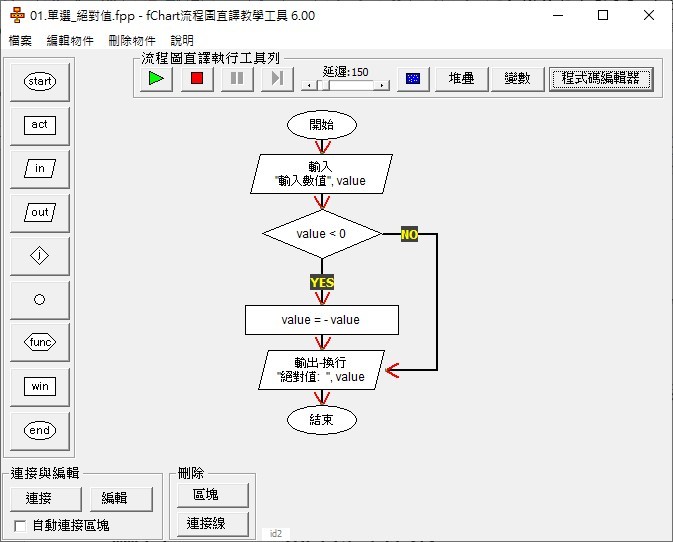


fChart 6 使用手册 21

在「另存新文件」对话框切换至储存路径，【存盘类型】栏选【所有档案 (\*.\*)】，在【文件名】栏删除最后的【.txt】，按【存档】钮储存成扩展名.fpp 的fChart 流程图项目档，如下图所示：



在成功下载 GitHub 档案库的 fChart 流程图项目档.fpp 后，我们就可以在fChart 开启此项目档，如下图所示：



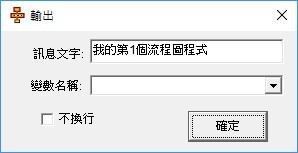
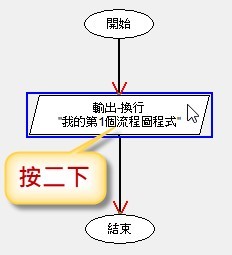
22 fChart 程序设计教学工具 - https://fchart.github.io

编辑 **fChart** 流程图项目

我们只需开启已经建立的 fChart 流程图项目，即可重新编辑项目的流程图。

# 编辑流程图符号

在流程图编辑区域建立的流程图符号，只需按二下符号图标，就可以开启符号的编辑对话框，重新编辑流程图符号，如下图所示：

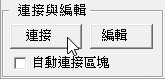


# 连接两个流程图符号

在 fChart 新增连接 2 个流程图符号之间的连接线，请在欲连接的 2 个符号各点选一下（顺序是先点选开始符号，然后是结束符号）后，我们共有二种方式来建立 2 个符号之间的连接线，红色箭头是执行方向， 如下所示：

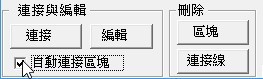
* 请按左下方「连接与编辑」框的【连接】钮来新增连接线，如下图所示：

fChart 6 使用手册 23



* 在没有符号区域，执行鼠标【右】键快捷菜单的【连接区块】命令来新增连接线（执行【使用 U 型线连接】命令是使用U 型线）。

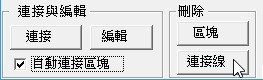
如果在左下方「连接与编辑」框勾选【自动连接区块】，在新增符号图标后，就会自动新增符号图标之间的连接线，如下图所示：



# 删除符号间的连接线

删除连接线请分别点选一下连接线两端的流程图符号（顺序没有关系），我们共有三种方式来删除连接线，如下所示：

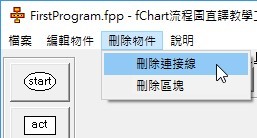
* 按左下方「删除」框的【连接线】钮删除之间的连接线。



* 在没有符号区域，执行鼠标【右】键快捷菜单的【删除连接线】命令来删除连接线。

24 fChart 程序设计教学工具 - https://fchart.github.io

* 执行「删除对象>删除连接线」命令删除连接线，如下图所示：



# 删除流程图符号

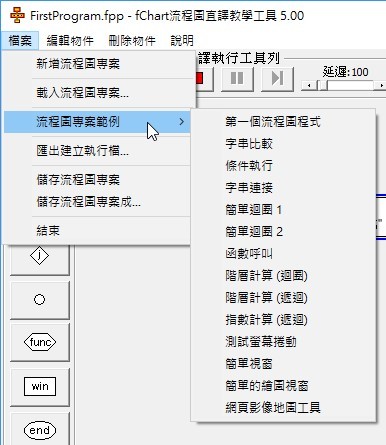
当流程图符号没有任何连接线时，我们才可以删除流程图符号，请点选一下欲删除符号后，我们共有三种方式删除流程图符号，如下所示：

* 按左下方「删除」框的【区块】钮来删除流程图符号。
* 在没有符号区域，执行鼠标【右】键快捷菜单的【删除区块】命令。
* 执行「删除对象>删除区块」命令。

# 流程图项目范例

在「档案>流程图项目范例」菜单的子选单提供多个内建流程图项目的范例，如下图所示：

fChart 6 使用手册 25



请直接执行选项命令，就可以马上加载范例的 fChart 流程图项目， 在 fChart 安装目录下的「项目范例」子目录提供更多流程图范例项目。

**fChart** 流程图符号的对话框

在 fChart 流程图直译器左边工具栏点选欲新增的流程图符号后，移动符号图标至编辑区域的欲插入位置，点选一下，可以开启编辑符号的对话框来编辑符号内容，各种符号对话框的说明，如下所示：

# 输出符号

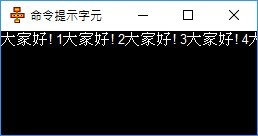
输出符号是用来显示程序的执行结果，请在「输出」对话框的

【消息正文】栏输入欲输出的文字内容，在下方【变量名称】字段可以输入或选择输出的变量值，例如：运算结果的变量 a，如下图所示：

26 fChart 程序设计教学工具 - https://fchart.github.io



上述图例没有勾选【不换行】，如果勾选，可以看到输出不会换行，如下图所示：



在输出符号的【消息正文】字段的输出字符串可以使用{0}符号（位在中间是数字 0）标示下方【变量名称】栏变量值显示的位置，换句话说，变量值不只可以显示在最后，还可以显示在字符串之中，例如：

输出{0}元

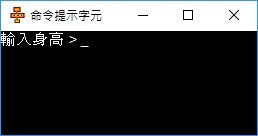
第一节: {0} 分

# 输入符号

输入符号可以让用户输入数据，我们是在「输入」对话框的

【提示文字】栏输入提示说明文字，下方【变量名称】字段输入或选择输入的变量名称，例如：让用户输入的数据储存至下方变量 height， 如下图所示：

fChart 6 使用手册 27



# 动作符号

动作符号可以定义变量值、数组变量，或建立拥有 2 个操作数的算术和字符串表达式，如下所示：

* 定义变量：在【定义变量】卷标新增变量和指定初值，我们可以在

【变量名称】栏输入新增的变量名称（或选择目前已经新增过的变量），【变量值】栏输入变量值（也可以是其他变量名称 b，即将其他变量值指定给变量，例如：a = b），如下图所示：



* 变数值是表达式：除了指定变量的值是常数值或其他变量值外，我们还可以在【变量值】字段输入完整的算术表达式，即指定变量是此表达式的运算结果，支持「+」、「-」、「\*」、「/」、「\」

（整数除法）、「^」（指数）、「%」（余数）运算符和「( )」括

28 fChart 程序设计教学工具 - https://fchart.github.io

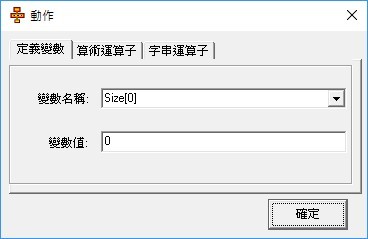
号，操作数可以使用整数、浮点数或数学函数。目前支持的数学函数，如下表所示：

|  |  |
| --- | --- |
| 数学函数 | 说明 |
| abs(exp) | 绝对值函数 |
| int(exp)、fix(exp) | 取得整数值 |
| sin(rad) 、 cos(rad) 、 tan(rad) 、  atn(rad) | 三 角 函 数 ， 参 数 是 径 度  deg\*3.1415926/180 |
| sqr(exp) | 开平方根 |
| factorial(exp) | 阶乘函数 |



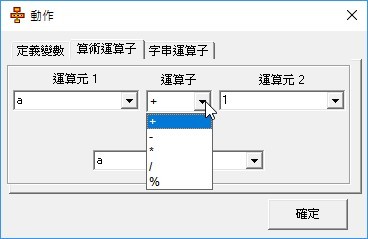
* 建立数组变量：在【定义变量】卷标可以建立数组变量和指定初值，fChart 并不用宣告数组，可以马上使用数组变量（因为 fChart 将数组变量视为是一个拥有索引的变量），请使用方框「[ ]」建立，在方框中也可以使用表达式，如下图所示：

fChart 6 使用手册 29



* 建立二元表达式：在【算术运算符】标签可以使用选择方式来建立二元的算术表达式，在中间可以选择使用的运算符：「+」（加）、

「-」（减）、「\*」（乘）、「/」（除）、和「%」（余数），如下图所示：



* 建立字符串表达式：在【字符串运算符】卷标可以建立字符串表达式，可以使用「&」（字符串连接）和「COMP」（字符串比较）运算符，如下图所示：

30 fChart 程序设计教学工具 - https://fchart.github.io

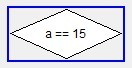
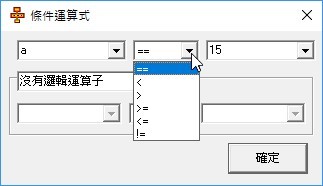


# 决策符号

决策符号是用来建立条件表达式，可以建立 2 个操作数的比较表达式，在中间可以选择条件运算符：「==」（等于）、「<」（小于）、

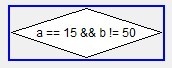
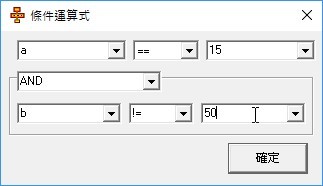
「>」（大于）、「>=」（大于等于）、「<=」（小于等于）和「!=」

（不等于），如下图所示：

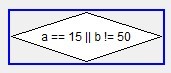
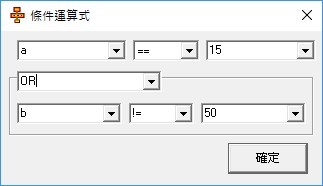


fChart 流程图直译器支持逻辑运算符 AND 和 OR，可以建立 2 个比较表达式作为操作数的逻辑表达式。首先是 AND 逻辑运算符，如下图所示：

fChart 6 使用手册 31



然后是 OR 逻辑运算符，如下图所示：



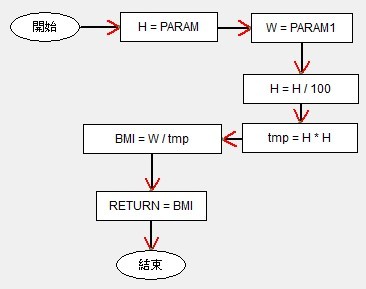
# 函数符号

fChart 流程图的.fpp 文件可以是主程序；也可以是一个同名的函数， 我们可以传递预设名为 PARAM 和 PARAM1 两个参数进入函数，然后指定 RETURN 变量值来回传值，如下图所示：



32 fChart 程序设计教学工具 - https://fchart.github.io

上述函数名称是指【BMI 函数.fpp】档，传递 2 个参数，和指定回传值至变量 BMI，【BMI 函数.fpp】的流程图，如下图所示：



上述传入函数的第 1 个参数是 PARAM；第 2 个是 PARAM1，最后指定 RETURN 的值，这就是回传的 BMI 值。

**fChart** 程序代码编辑器与 **Blockly** 积木程序

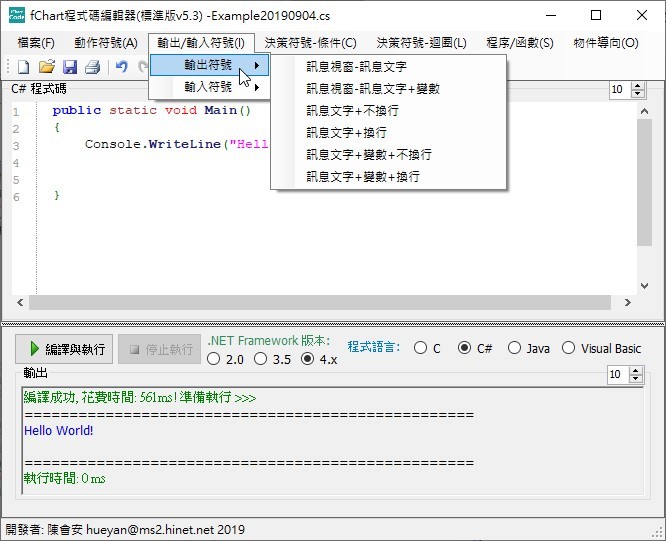
fChart 程序设计教学工具内建轻量级整合开发环境和 Blockly 积木程序编辑器，当我们使用 fChart 流程图直译器了解流程图的程序逻辑后，可以马上拼出或写出算法的程序代码，各版本支持的程序语言有： C/C++、Java、C#、VB、Python、JavaScript 和Node.js。

# fChart 程序代码编辑器

在 fChart 流程图直译器的执行工具栏，按最后【程序代码编辑器】钮，可以启动 fChart 程序代码编辑器的整合开发环境，程序代码编辑器的接口，如下图所示：

fChart 6 使用手册 33

菜单



程序代码编辑窗口

插入代码段

编译执行按钮

输出框

切换程序语言

上述执行画面上方是菜单，这些菜单命令是对应流程图符号， 可以帮助初学者建立程序代码和流程图符号的对应关系，在下方是程序代码编辑窗口，右下方选择使用的程序语言；左下方是编译和执行按钮，位在最下方是输出框，可以输出编译讯息或 VB 和C#程序的执行结果。

在 fChart 各版本的程序代码编辑器都内建 TCC 的 C 语言编译程序，可以马上编辑、编译和执行 C 程序，也可以编译执行 C#和 VB 程序。以C 语言为例，我们除了自行使用键盘输入 C 程序代码外，也可以使用菜单命令来快速插入 C 代码段后，然后小部分修改来完成 C 程序代码的输入，其步骤如下所示：

Step 1：请在 fChart 流程图直译器的上方工具栏，按最后【程序代码编辑器】钮启动 fChart 程序代码编辑器，然后在右下方选项选【C】，切换成C 程序语言，如下图所示：

34 fChart 程序设计教学工具 - https://fchart.github.io



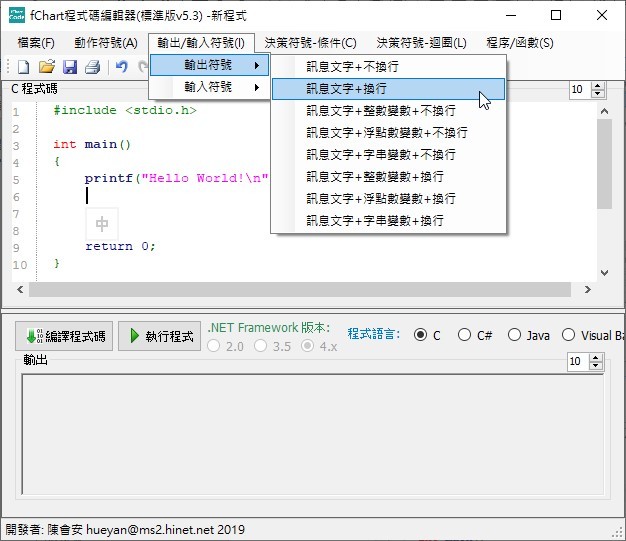
上述接口位在程序代码编辑区域右上角的数字是目前的字型尺寸，我们可以使用箭头马上调整程序代码显示的字型尺寸。

Step 2：因为流程图是使用输出符号输出一段消息正文，请先在 main()

函数程序区块中点一下作为插入点。

Step 3：请执行「输出/输入符号>输出符号/消息正文+换行」命令，可以插入 C 语言 printf()函数的输出程序代码，"\n"是换行。

fChart 6 使用手册 35



Step 4：请将原来的字符串内容「Hello World!」改为「我的第 1 个流程图程序」。

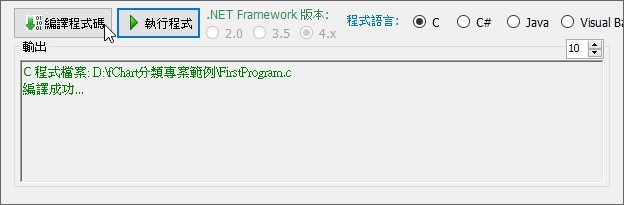


Step 5：执行「档案>储存」命令储存档案，可以开启「另存新文件」对话框，请切换路径后，在【文件名】栏输入档名 FirstProgram.c，按

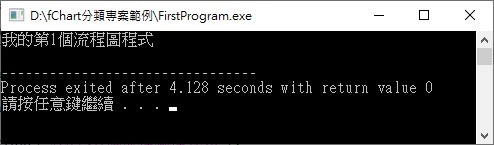
【存盘】钮储存成 C 程序档案（执行「档案>开启」命令可以开启存在的 C 程序档案）。

36 fChart 程序设计教学工具 - https://fchart.github.io

Step 6：请按下方【编译程序代码】钮编译 C 程序，如果没有错误，可以在下方显示成功编译的绿色消息正文；错误是红色的错误消息正文。



Step 7：然后按【执行程序】钮执行 C 程序，可以开启「命令提示字符」窗口来显示执行结果。



# Blockly 积木程序编辑器

Blockly 是 Google 公司支持的开源项目，一套在客户端执行的可视化积木程序编辑器，源于 Scratch 和 App Inventor 的积木观念，可以让用户不用撰写文字内容的程序代码，直接拖拉积木来组合出积木程序。fChart 工具支持 2 种 Blockly 积木程序编辑器，如下所示：

* Cake Blockly for C：基于 Blockly 的积木程序编辑器，原是韩国

Joshua 团队开发的 C 语言版本（ https://github.com/cra16/cake-

fChart 6 使用手册 37

core），笔者已经中文化、修正错误和增强功能，可以帮助初学程序者轻松拖拉积木来学习C 程序设计，如下图所示：



* Blockly 积木程序：Python 和 Node 版支持，这是修改源于"云林SONG"的精简版本，只保留 Code 部分，目前版本已经修改支持整数和Python 3（不再支持 Python 2），如下图所示：

