Lemon v1.1 Manual

Crash*

Contents

*作者Blog：http://hi.baidu.com/oimaster

1 开发初衷

笔者作为一名OIer，平时做题时经常需要测试自己的程序能否通过所有的测试数据。在Windows平台下，一直使用Cena作为评测软件，并且也很喜欢Cena的界面设计方式。但是Cena本身有一些局限性：比如只能在Windows下使用，不能测试提交答案题等。同时遇到像POI那样的数据文件名添加时也比较头疼。因此笔者决定开发一款跨平台的测试软件，以帮助OIer测试自己的程序。并且笔者也对GUI程序开发产生了一些兴趣，闲暇之余学习了Qt库的基本用法，由于Qt库本身就是跨平台的GUI库，正好符合了我的想法。

Lemon项目就是这样产生的，目前支持Windows平台和Ubuntu平台。

关于Lemon这个名称，其实也没什么特殊含义，是我无聊的时候想到的。因此软件的图标就是下面的大柠檬：

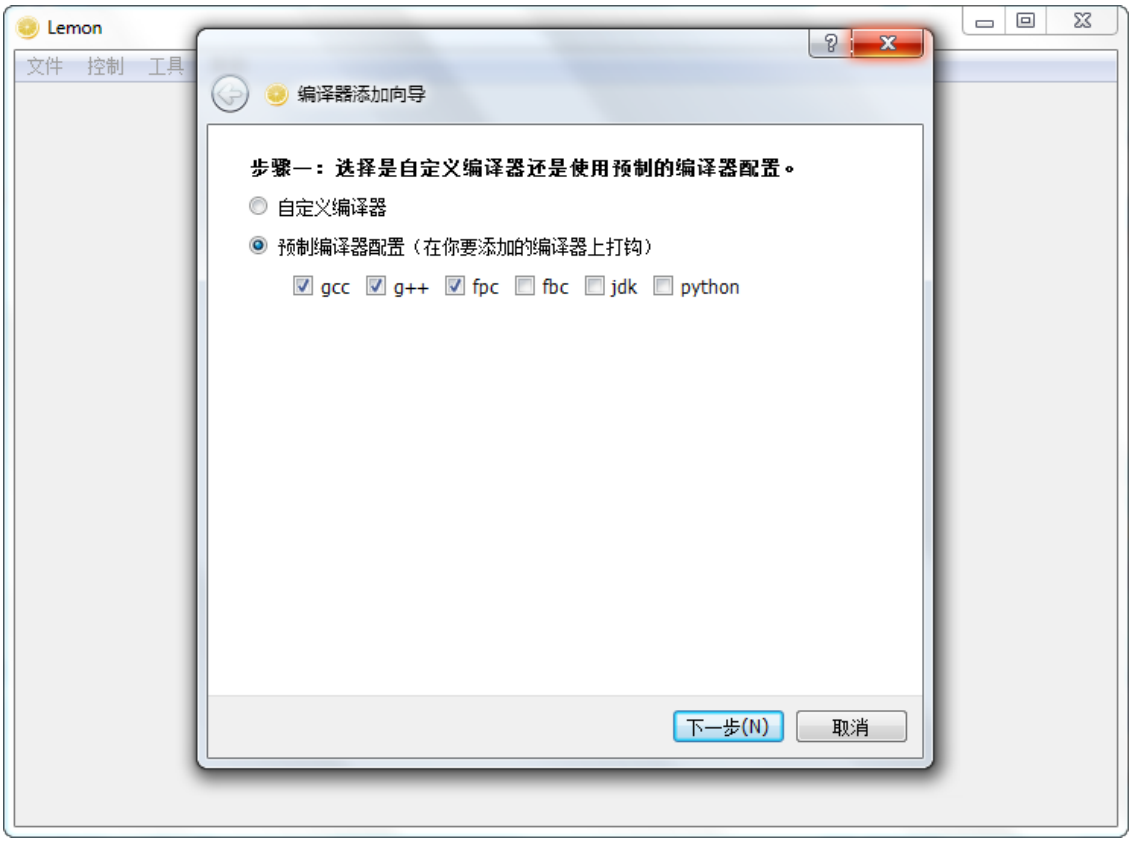
2 安装

Project Lemon在Google Code上有一个页面，不仅用来存储源代码，也用来发布软件的正式版本。

2.1 Windows平台

用浏览器打开http://code.google.com/p/project-lemon页面，进入Downloads，可以选择下载`lemon_v1.1_release_mingw.7z`或`lemon_v1.1_release_msvc2010.7z`。其实这两个版本没什么区别，只是编译软件用的编译器不同，前一个是MinGW g++ 4.4.0，后一个是MSVC 2010。

下载完了之后，直接解压缩到某个目录下，运行lemon.exe就能看到软件的界面。



2.2 Ubuntu平台

Ubuntu平台的话，需要先安装一些工具。打开终端后输入如下命令：

```
sudo apt-get install g++ git qt4-dev-tools
```

然后需要使用git下载源代码，继续输入下面的命令：

```
git clone https://code.google.com/p/project-lemon
```

接着进入project-lemon目录切换到branch v1.1，并进行编译：

```
cd project-lemon
git checkout v1.1
```

```
qmake lemon.pro
make
```

最后通过输入./lemon就能够运行了。

可能在通过git下载源代码时会提示无法连接到服务器，这是由于某个“众所周知”的原因造成。可以按照下面的方式解决：先在终端中输入ping www.google.com，这样就能得到Google服务器的IP，然后修改系统的hosts，在其中将code.google.com的IP指向前面ping到的IP，再重新输入前面的clone命令。如果还是不行请多ping几次，每次ping到的IP可能是不同的。

实在没法用git的话，可以在Downloads里下载lemon_v1.1_release_src.7z，然后解压缩到某个目录，输入前文中的qmake和make命令。

3 配置

由前面的截图可以看到，Lemon第一次运行时会弹出添加编译器的向导，这里可以按照向导的说明设置编译器路径，详细的说明等到后面一节。

3.1 常规配置

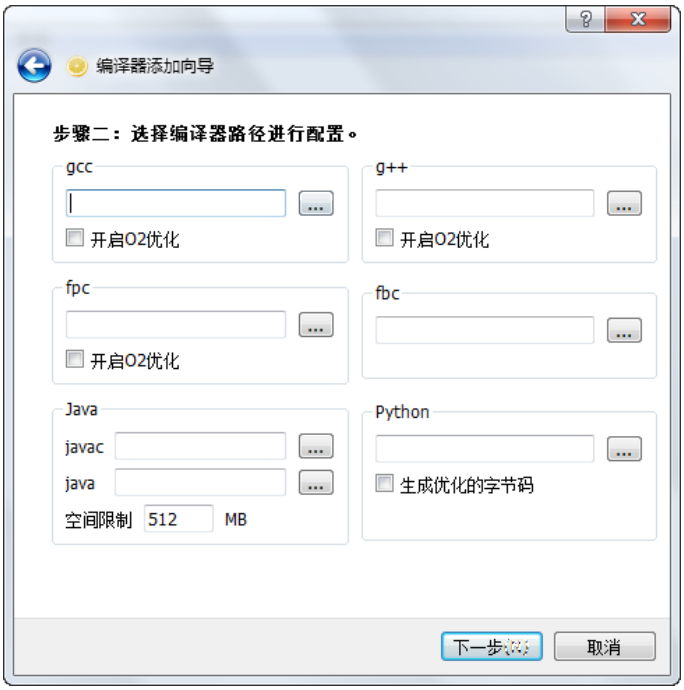
Lemon运行后会弹出打开或新建比赛的对话框，先选择取消关闭这个对话框，然后在工具菜单中选择“选项”，就能够看到下图所示的对话框：

- **默认分值** 新建一个测试点时默认的测试点分值，可以设置的最大值为100。
- **默认时间限制** 新建一个测试点时默认的时间限制，可以设置的最大值为600,000毫秒，即10分钟。
- **默认空间限制** 新建一个测试点时默认的空间限制，可以设置的最大值为1024MB。
- **编译时间限制** 测试时允许编译器运行的最长时间，可以设置的最大值为600,000毫秒。
- **校验器时间限制** 对于使用自定义校验器的试题，测试时允许校验器运行的最长时间，可以设置的最大值为600,000毫秒。
- **源程序大小限制** 测试时可以接受的最大源程序大小，可以设置的最大值为10,240KB，即10MB。
- **测试线程数** 测试时最多允许同时运行的线程数，可以设置的最大值为8。请确保设置的值不超过CPU的核心数，否则对于程序运行速度的影响很大。
- **输入、输出文件扩展名** 在自动添加试题时扫描的输入和输出文件的扩展名。如果有多个请用`';'`隔开。每种扩展名中只能包含英文字母和数字，大小写是敏感的。注意这里的扩展名只供添加测试点时软件搜索文件使用，测试时并不会检查。

3.2 编译器配置

点击上方的“编译器”选项卡就能进入编译器的配置。

按右边的加号就会出现添加编译器的向导，第一步是选择使用预置的编译器配置还是手动配置新的编译器。一般来说，用预置的配置就能够满足大多数需求，第一步只要在需要配置的编译器前打钩，进入下一步后就能看到选择路径的界面。

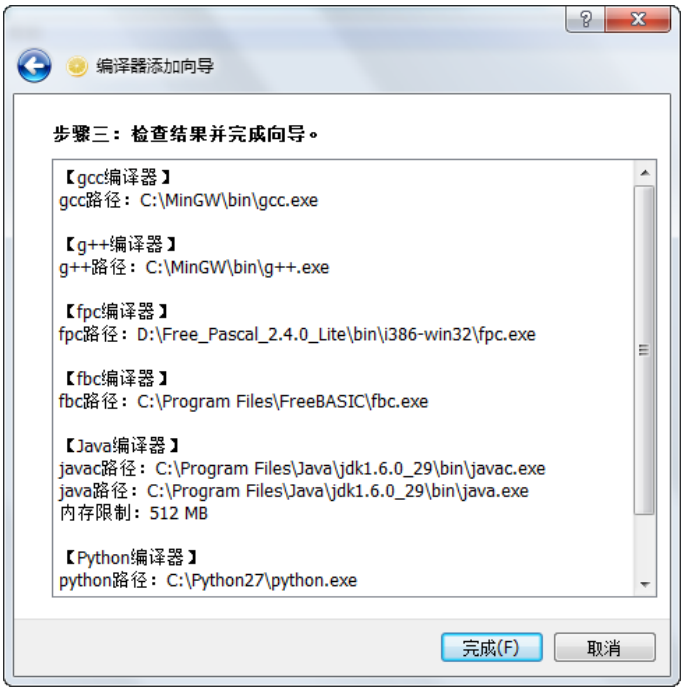


gcc、g++和fpc的O2优化是一种编译器执行的代码优化，可以一定程度提高程序运行效率。不过国内的竞赛中一般不开启这项优化。

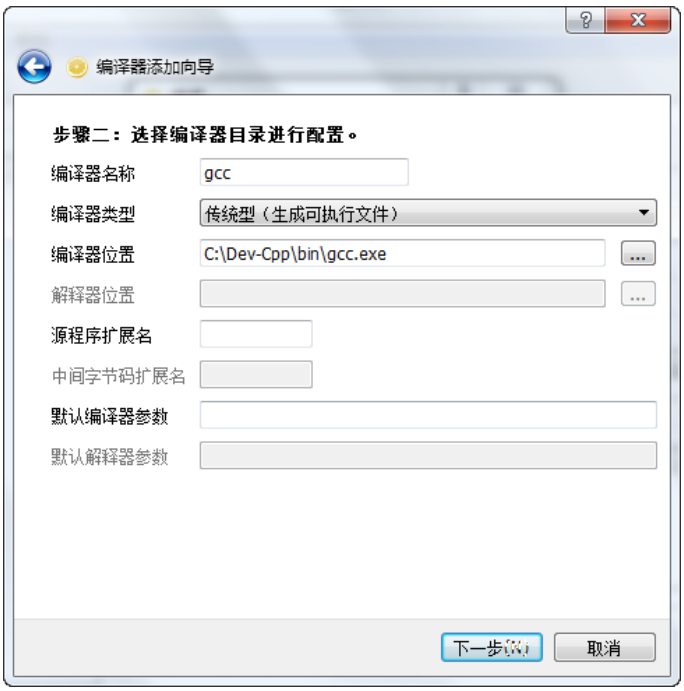
Java的内存限制是由Java虚拟机限制的，你也可以选择不限制内存。

Python解释器可以生成一种中间字节码提高程序加载速度，实际测试中基本没什么优化效果。

最后一步会让你确认编译器路径是否设置正确。



除了使用内置的六种编译器配置，你也可以选择手动配置新的编译器。



- **编译器名称** 编译器在列表中显示的名称。
- **编译器类型** 总共有三种类型可选，括号里给出了相关的解释。C++、Java和Python分别是三种类型的典型代表。

- **编译器位置** 如果编译器的类型是传统型或需要编译的解释型，这里就要选择编译器的位置。对于传统型，编译器会将源代码转直接换成机器代码，而解释型的编译器会生成中间字节码。
- **解释器位置** 如果编译器的类型是解释型，就要选择解释器的位置。解释器用于执行中间字节码或直接解释执行源代码。
- **源程序扩展名** 用于判断哪些扩展名的源程序使用这个编译器编译，如果有多个扩展名请用`';'`隔开。
- **中间字节码扩展名** 解释型编译器生成的中间字节码的扩展名，例如Java的中间字节码扩展名为`.class`。
- **默认编译器参数** 编译时传递给编译器的参数，中用`%s`表示不带扩展名的源程序文件名，`%s.*`表示带扩展名的源程序文件名。例如`gcc`的编译参数为`-o %s %s.*`。
- **默认解释器参数** 运行解释器时传递的参数，表示的方法同编译器参数。

向导完成后，我们再回到编译器管理的选项卡。

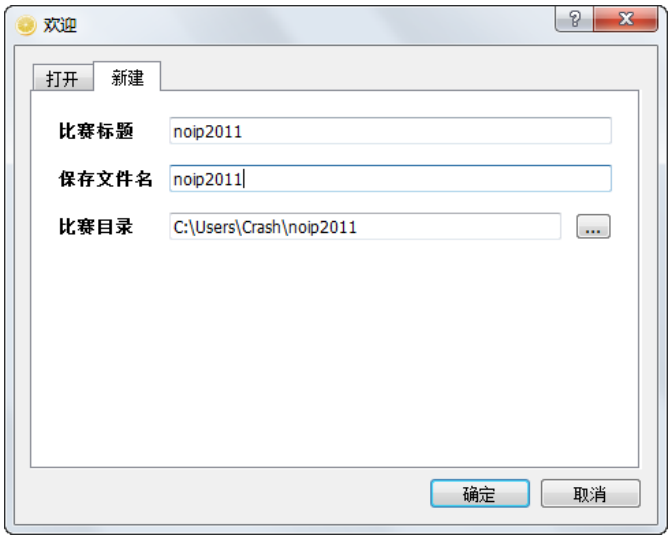
如果要删除当前选择编译器，可以按右边的减号。右边的上下箭头是用来调整编译器优先级的。编译器的优先级是指：如果选手对于同一道题提交了多种扩展名的源程序，排在前面的编译器会被选考虑，同时在源程序扩展名设置中排在前面的扩展名会被先考虑。

注意到下面有高级按钮，可以修改已有编译器的配置。由于不同语言执行效率有差异，因此可以放宽特定语言的时间限制或空间限制，也就是将时间限制或空间限制乘上一个实数，对于空间限制，也可以选择直接取消。每个编译都可以有多个配置，不同配置的编译参数或解释器运行参数不同，一般用于选择不同的优化开关。点击环境变量按钮可以设置编译器和程序运行时额外设置的环境变量，一般用于保证运行所需的动态链接库文件能被找到。

4 新建比赛

运行软件后会弹出“欢迎”对话框，可以选择打开已有的比赛或者新建比赛。如果关闭了这个对话框，也可以在文件菜单中选择打开或新建比赛。

新建比赛的对话框如下图所示：



- **比赛标题** 用来显示在软件标题栏上的比赛标题。
- **保存文件名** 保存比赛文件使用的文件名，这里输入的不需要附带扩展名。
- **比赛目录** 比赛相关文件的存储目录。

点击“确定”后，软件会在指定的比赛目录下创建`data`和`source`目录，同时还有一个扩展名为`cdf`的文件用来保存试题、选手信息。其中`data`下用来存放试题的数据、自定义的校验器等文件，`source`目录下存放每个选手的源程序或答案文件。`source`目录下每个文件夹代表一位选手，文件夹的名称为选手的名称，每位选手的文件夹里直接存放相应的源程序和答案文件，不需要再创建其他的目录。

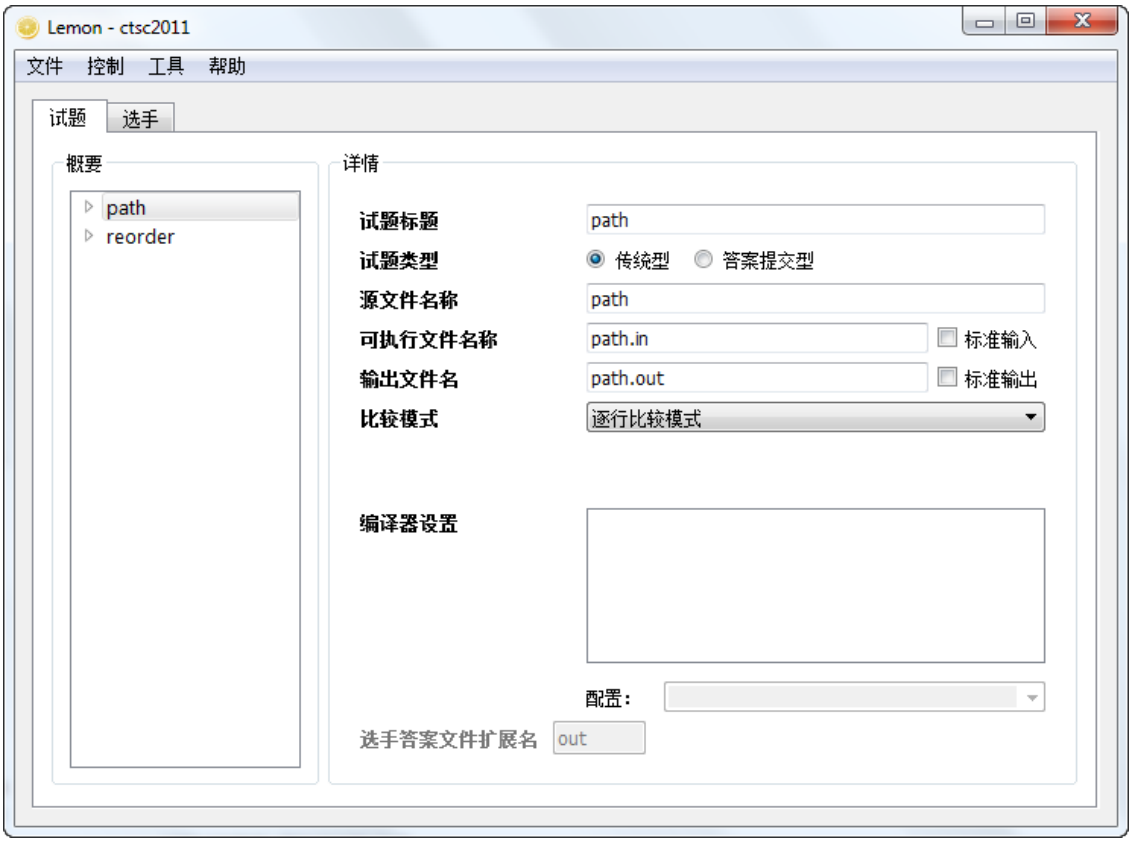
4.1 添加新试题

建立好比赛后，在左边按鼠标右键就可以添加新的试题。然后在右边设置试题相关的信息。

- **试题标题** 试题在列表中显示的名称。
- **试题类型** 试题的类型，目前可用的选择只有传统型和答案提交型。
- **源文件名称** 源程序的文件名，注意不要带扩展名。
- **输入、输出文件名** 输入输出文件名，右边可以选择使用标准输入或输出。
- **比较模式** 比较选手输出和标准输出的方式，目前有五种方式：逐行比较模式、忽略多余空格和制表符的逐行比较模式、外部工具模式、实数比较模式和自定义校验器。逐行比较模式会一行一行比较选手的输出和标准输出是否相同，不同系统平台的换行符不同不会产生影响。逐行比较模式中也可以选择忽略多余的空格和制表符。外部工具模式会调用Linux下的`diff`命令进行比较，Windows下也能使用，但是要将`diff.exe`放在和`lemon.exe`相同的

目录下。实数比较模式会逐一读取选手输出和标准输出中的每一个实数，分别比较误差是否在允许范围内。自定义检验器需要选择一个可执行文件作为校验器，具体的说明请参见下一个章节。

- **编译器设置** 为每个编译器选择配置，也就是选择相应的编译参数，默认会选择“default”配置。
- **选手答案文件扩展名** 对于提交答案题，选手提交的答案文件中，每个文件会和输入文件中除去扩展名后文件名一样的那个配对，这里可以设置选手提交的答案文件的扩展名，默认为`out`。



4.2 自定义校验器说明

自定义检验器需要为一个可执行文件，评测软件通过将一些参数传给校验器，使得校验器获得输出、输出文件等信息，然后将得分的结果写入指定文件中。评测软件会向校验器传入六个参数，按照顺序分别表示标准输入文件、选手输出文件、标准输出文件、本测试点满分、分数输出文件、额外信息文件。其中分数输出文件必须创建，需要向其中写入一个非负整数表示得分。额外的信息文件可以不创建，如果创建了可以写入任何信息，这些信息会显示在结果中。

4.2.1 校验器示例（C语言）

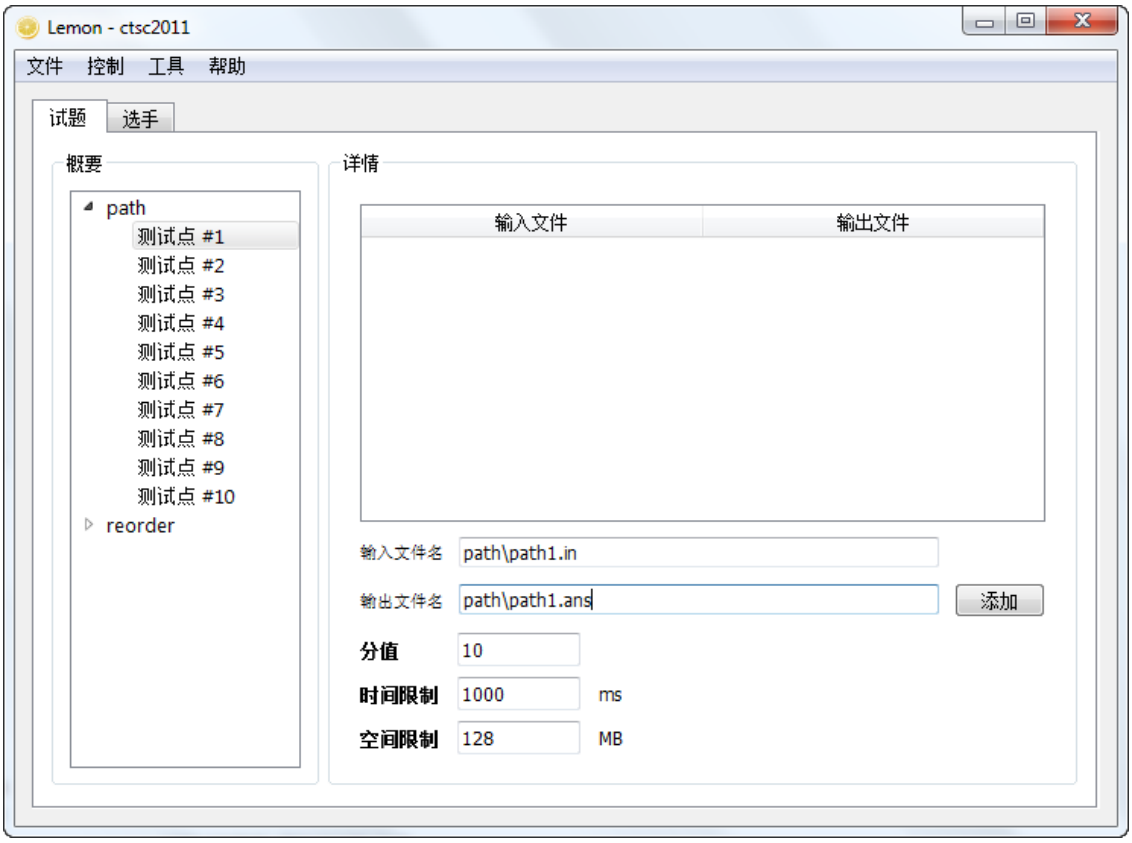
```c
#include <stdio.h>
#include <math.h>

int main(int argc, char *argv[]) {
    FILE *fin = fopen(argv[1], "r");
    FILE *fout = fopen(argv[2], "r");
    FILE *ans = fopen(argv[3], "r");

    FILE *score = fopen(argv[5], "w");
    FILE *msg = fopen(argv[6], "w");

    double yourAnswer, stdAnswer;
    fscanf(fout, "%lf", &yourAnswer);
    fscanf(ans, "%lf", &stdAnswer);

    if (fabs(yourAnswer - stdAnswer) < 1e-4) {
        fprintf(score, "%s\n", argv[4]);
        fprintf(msg, "Correct answer\n");
    } else {
        fprintf(score, "0\n");
        fprintf(msg, "Wrong answer\n");
    }

    fclose(fin);
    fclose(fout);
    fclose(ans);
    fclose(score);
    fclose(msg);

    return 0;
}
```

4.2.2 校验器示例（Pascal语言）

```pascal
program checker;

var
    yourAnswer, stdAnswer : double;
    fin, fout, ans, score, msg : text;

begin
    assign(fin, ParamStr(1));
    reset(fin);
    assign(fout, ParamStr(2));
    reset(fout);
    assign(ans, ParamStr(3));
    reset(ans);
    assign(score, ParamStr(5));
    rewrite(score);
    assign(msg, ParamStr(6));
    rewrite(msg);

    readln(fout, yourAnswer);
    readln(ans, stdAnswer);
    if abs(yourAnswer - stdAnswer) < 1e-4 then
        begin
            writeln(score, ParamStr(4));
            writeln(msg, 'Correct answer');
        end
    else
        begin
            writeln(score, 0);
            writeln(msg, 'Wrong answer');
        end;

    close(fin);
    close(fout);
    close(ans);
    close(score);
    close(msg);
end.
```

4.3 添加新测试点

在左边选中一道试题后右键鼠标出现菜单，选择“添加测试点”即可添加一个新的测试点，右边会变成测试点设置界面。在输入文件名和输出文件名中输入内容后，按添加即可添加一组测试数据。这里的输入输出文件必须在data目录下，并且只要输入data目录之后的路径，如下图所示：



一个测试点可以包含多组输入输出，最终测试点的得分为所有输入输出中得分最低的那一组的分数。

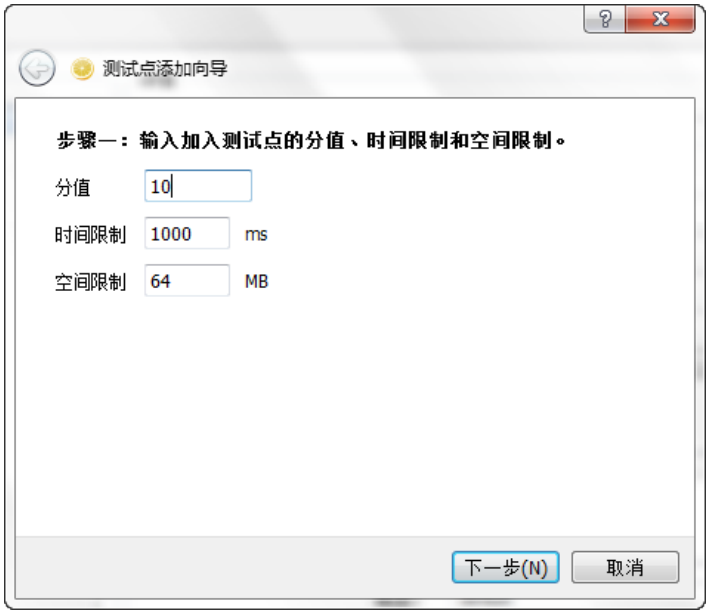
如果要编辑输入输出文件名，直接表格相应位置双击即可修改。如果选中一行或多行后按Delete键，即可删除相应的输入输出文件。

在下面可以设置本测试点的分值、时间限制和空间限制。可以设置的最大分值为100、最大时间限制为600,000毫秒（10分钟）、最大空间限制为1024MB。

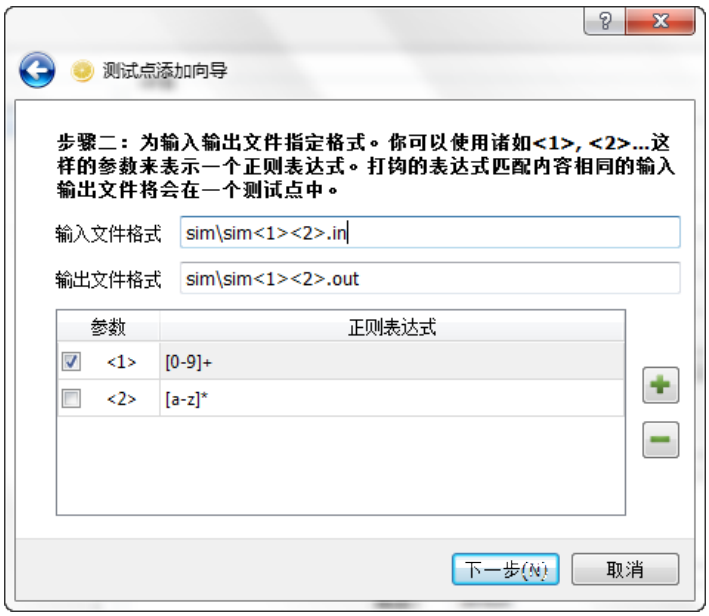
4.4 批量添加测试点

在左边选中一道试题后右键鼠标出现菜单，选择“添加多组测试点...”后会弹出一个向导，用来批量添加测试点。

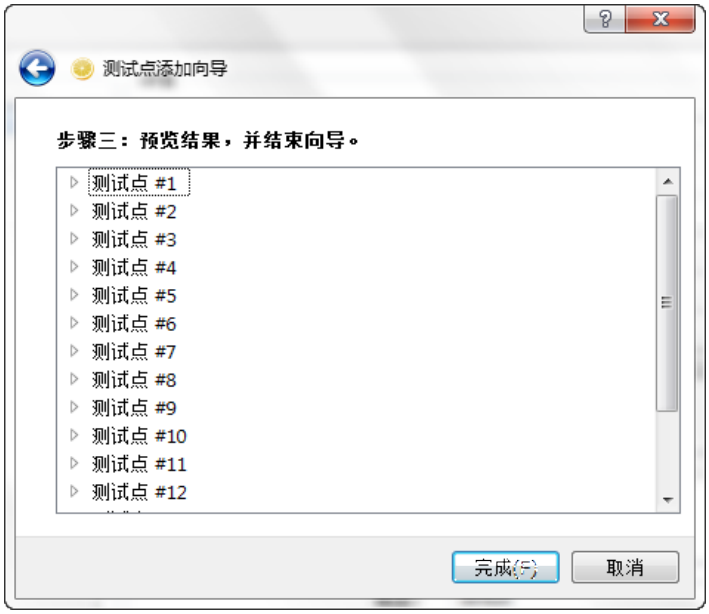
向导的第一步是设置每个测试点的分值、时间限制和空间限制。



第二步是设置如何匹配输入、输出文件名，这里需要会使用简单的正则表达式。输入、输出文件格式中可以使用<1>、<2>、……、<9>来表示一个正则表达式，然后在下面按右边的加号可以新建这样一个参数并指定参数代表的正则表达式。需要注意的是在输入文件格式和输出文件格式中，每个参数分别只能出现一次。对于所有参数匹配内容都相同的文件，会作为一组输入输出。对于打钩参数匹配内容相同的输入输出，会放在同一个测试点中。下图所示的是一道POI的试题。POI的试题常常使用“捆绑测试”的方法，一个测试点拥有多组输入输出文件，一般来说数据文件命名方式是题目英文简称加一个数字再加一个字母，数字用来标记测试点编号，字母用来标记一组输入输出。用图中的方式就能够按照预期的方式添加所有输入输出文件。

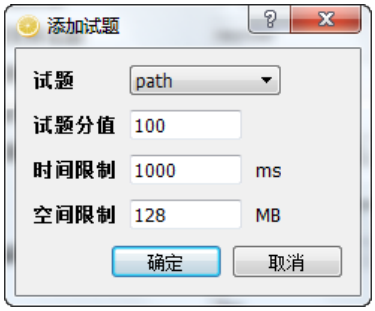


最后一步是预览结果，如果发现结果和预期的不同可以回到上一步修改参数。



4.5 自动添加试题

为了简化试题、测试数据的添加过程，笔者特别设计了自动添加试题的功能。对于一道试题，如果希望能够自动添加，请在data目录下为这个试题创建一个文件夹，将相应的数据文件放入文件夹中。然后在控制菜单中选择“自动添加试题”，会出现一个对话框，给出找到的可添加试题，然后可以设置每道试题所有测试点的总分值、所有测试点的时间限制和空间限制，确定后就会添加相应的试题。

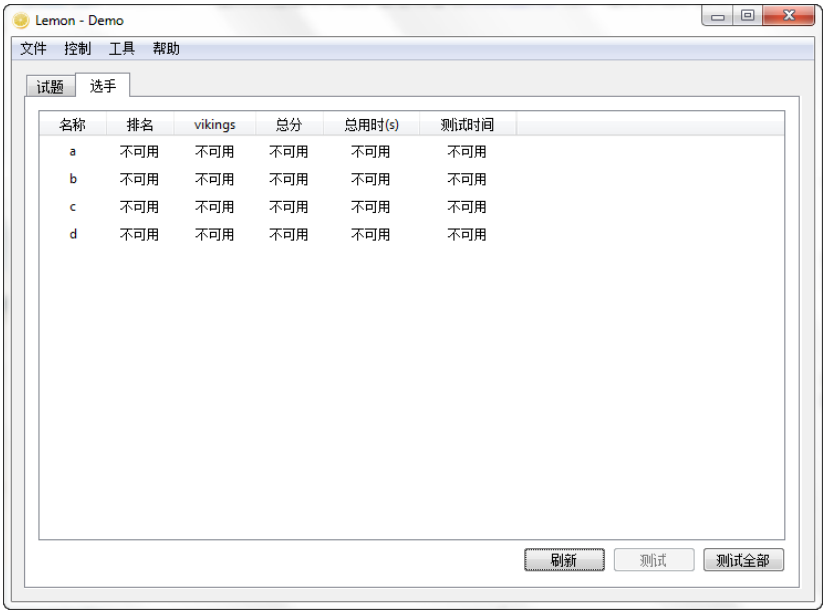


自动添加的试题默认作为传统型试题，添加后的试题标题和源程序名称都是对应试题的文件夹名称，输入输出文件分别再加上扩展名in和out。数据文件的匹配方法是：根据设置中设定的输入输出文件扩展名，选出相应的文件，如果在设置中输入或输出文件扩展名为空，会自动将输入文件扩展名设置为in，输出文件扩展名设置为out;ans。然后对于除扩展名外文件名相同的文件会被作为一组输入输出，并为这一组输入输出创建一个测试点，注意文件名的大小写是敏感的。用“自动添加试题”功能添加的试题，每个测试点只会有一组输入输出。每个测试点的分值会用设置的总分值除以测试点个数，由于分值只能是整数，没法除尽时会向下取整。请尽量保证输入输出文件能够按上述方法唯一配对，否则产生的结果不可预料。

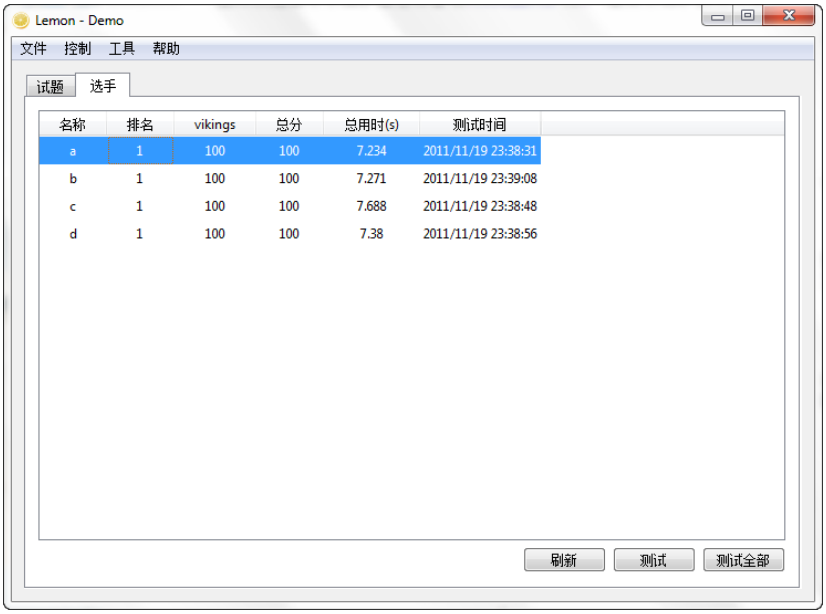
其中可以设置的试题总分值最大为100,000，时间限制最大为600,000毫秒（10分钟），空间限制最大为1024MB。

5 开始测试

点击“选手”选项卡，按下面的“刷新”按钮后，就会根据source目录下的文件夹添加列表中不存在的选手，并从列表中删除已经从source目录下删除的选手。然后单击“测试全部”按钮就能够开始测试。注意选手提交的程序名大小写是敏感的，必须要和试题设置中源文件名称的大小写匹配。



选中单个或多个选手后按“测试”按钮可以仅测试选中的部分选手，按Delete键或右键点删除可以删除选中的选手。测试完成后通过在表头上单击可以按照相应的项目排序。双击一名选手可以查看详细的测试结果。



6 导出成绩

在“控制”菜单中选择“导出成绩”可以将结果导出成HTML文档或表格文件。推荐使用HTML格式，可以导出完整的结果信息，导出成表格的话只能显示选手每道题的得分和总分。表格有两种格式可以选择：CSV格式和XLS格式（仅Windows可用并要安装Excel）。CSV格式是逗号分隔符，多数表格编辑软件都能查看。XLS格式的导出需要利用ActiveX调用Excel，写入速度非常慢，除非特别需要，否则不推荐使用。

此外，“控制”菜单中还提供了建立批处理测试目录的功能，这个功能会在比赛目录下创建`selftest`目录，里面会为每道试题创建一个文件夹，并自动生成相应的批处理测试文件，选手只要将编译好的可执行文件复制到目录中，即可对程序进行简单的测试。