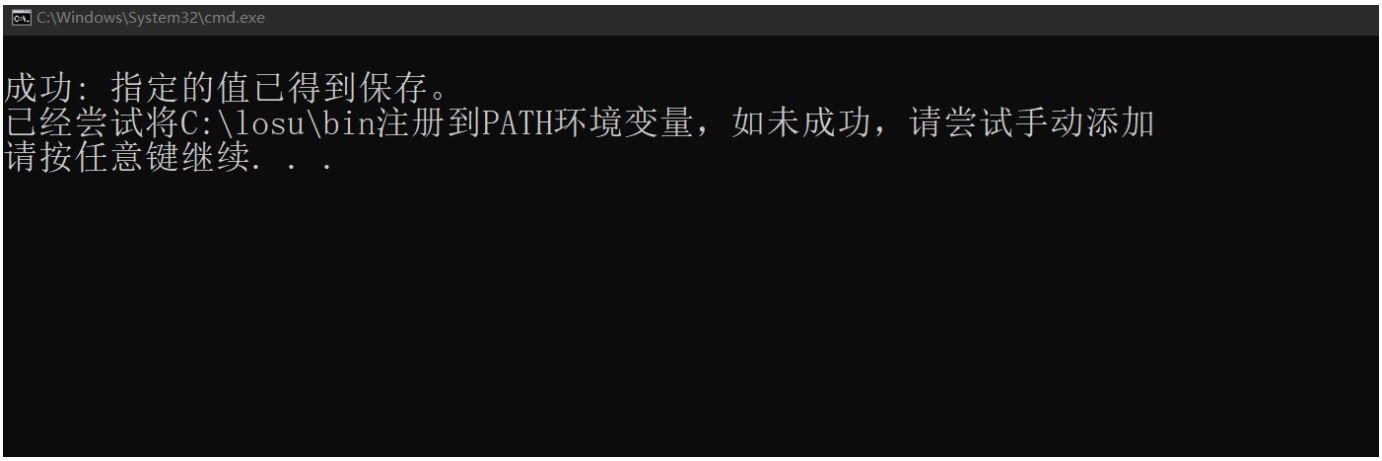
安装流程

1. 下载洛书1.4.2版本，解压到C盘
2. 在losu目录下找到install.bat文件，右键此文件，选择属性选项，解除锁定
3. 再次右键install.bat文件，以管理员身份运行，允许此应用对设备进行更改，出现下图证明安装成功



打开编辑器

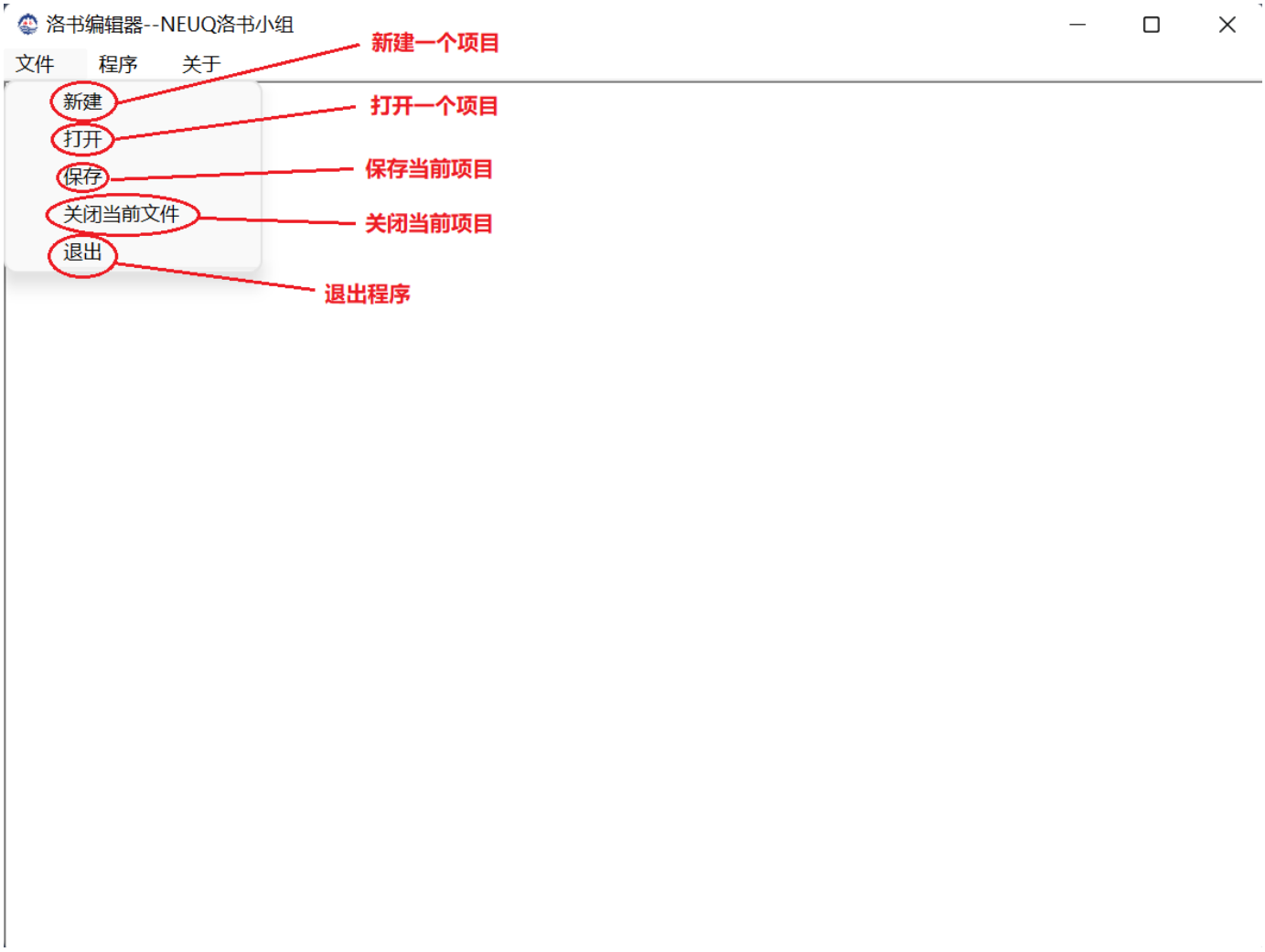
进入C:\losu\bin目录，将洛书编程.exe文件创建桌面快捷方式

打开该文件，会出现以下界面


東北大學秦皇島分校
Northeastern University at Qinhuangdao
洛书编程
自主的中文编程语言设计与研发


洛书编辑器--NEUQ洛书小组
文件
程序
关于

编辑器功能介绍


洛书编辑器--NEUQ洛书小组
文件
程序
关于
新建
打开
保存
关闭当前文件
退出
新建一个项目
打开一个项目
保存当前项目
关闭当前项目
退出程序

第一个程序

点击文件工具栏，新建项目你好洛书

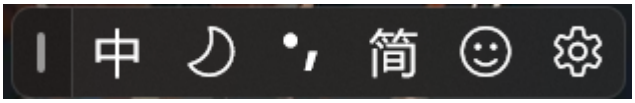
输入以下代码

```
#加载 开始
#导入 洛书/程序
#方法 开始
    程序,输出("你好，洛书\n")
;
```

点击工具栏中的编译运行，就可以运行你的程序了~

如果你的代码没有出错，即可看到下图所示的窗口

注意在程序中除了输出的具体内容外都要使用英文符号可以打开输入法工具切换成中文模式+英文标点(如下)



下面是具体每一行代码的介绍

```
#加载 开始
—设置加载点，表明程序从此开始
```

```
#导入 洛书/程序
—导入洛书的程序模块，该模块包含了输出输出等功能
```

```
#方法 开始
    程序,输出("你好，洛书\n")
;
—在"#方法 开始"与";"之间输入你想实现的代码
  "程序,输出"表明你想在屏幕上输出的内容
  ("")引号中间的内容即是运行成功后将在屏幕上出现的文字
  \n表示输出换行，即输出"你好，洛书"后光标将跳转到下一行
```

课后练习

1.用洛书编辑器输出你好世界，效果如下

```
你好世界
```

2.输出一个字符"*"组成的三角形，效果如下

```
  *
 ***
*****
```

3.输出古诗《静夜思》，效果如下

```
床前明月光，
疑是地上霜。
举头望明月，
低头思故乡。
```

本章拓展阅读

编程语言发展史

1940之前

第一个编程语言比现代的计算机还早诞生。首先，这种语言是种编码(en:code)。

于1801年发明的提花织布机(或称甲卡提花织布机，英文：en:Jacquard loom)，运用打孔卡上的坑洞来代表缝纫织布机的手臂动作，以便自动化产生装饰的图案。 Ada Lovelace在1842年至1843年间花费了九个月，将意大利数学家Luigi Menabrea关于查尔斯·巴贝奇新发表机器分析机的回忆录翻译完成。她于那篇文章后面附加了一个用分析机计算伯努利数方法的细节，被部分历史学家认为是世界上第一个电脑程序。

Herman Hollerith在观察列车长对乘客票根在特定位置打洞的方式后，意识到他可以把资讯编码记载到打孔卡上，随后根据这项发现使用打孔卡来编码并纪录1890年的人口统计资料。

第一个计算机程式码是针对他们的应用面设计的。在20世纪的前十年主要是用十进制来算数，后来人们发现不只是用文字，也可以用数字来表现逻辑。举例来说，阿隆佐·邱奇曾以公式化(formulaic)的方式表达λ演算。图灵机是一种纸带标记(tape-marking)机器(就像电话公司用的那种)操作方法抽象化后的集合。图灵机这种透过有限数字(finite number)呈现机器的方式，奠定了程式如同冯·诺伊曼结构计算机中的资料一样地储存的基础。但不同于λ演算，图灵机的程式码并没有办法成为高阶编程语言的基石，这是是因为它主要的用途是分析算法的复杂度。

就像许多历史上的"第一次"一样，第一个现代编程语言也很难界定。最一开始是因为硬件限制而限定了语言，打孔卡允许80行(column)的长度，但某几行必须用来记录卡片的顺序。FORTRAN则纳入了一些与英文字词相同的关键字，像是"IF"、"GOTO"(原字词为go to)，以及"CONTINUE"。之后采用磁鼓(magnetic drum)作为内存使用，也代表计算机程式也必须插入(interleave)到磁鼓的转动(rotation)中。和现今比较起来，这也让编程语言必须更加依赖硬件(hardware-dependent)。

对部分的人认为必须在"编程语言"的状态确立之前，根据能力(power)以及可读性(human-readability)的程度来决定历史上第一个编程语言是什么语言。提花织布机和查尔斯·巴贝奇所制作的差分机(en:Difference Engine)都具备在大量限制下，简单描述机器应执行行为的语言。也有种并非设计给人类运用的受限特定领域语言(en:domain-specific language)，是将打孔卡运用到自动演奏钢琴(en:player piano)上。

1940年代

最早被确认的现代化、电力启动(electrically powered)的计算机约在1940年代被创造出来。程序员在有限的速度及内存容量限制之下，撰写人工调整(hand tuned)过的组合语言程式。而且很快

就发现到使用组合语言的这种撰写方式需要花费大量的脑力(intellectual effort)而且很容易出错(error-prone)。

Konrad Zuse于1948年发表了他所设计的Plankalkül编程语言的论文[1]。但是在他有生之年却未能将该语言实作，而他原本的贡献也被其他的发展所孤立。

在这段期间被开发出来的重要语言包括有：

- 1943 - Plankalkül (Konrad Zuse)
- 1943 - ENIAC coding system
- 1949 - C-10

1950与1960年代

有三个现代编程语言于1950年代被设计出来，这三者所衍生的语言直到今日仍旧广泛地被采用：

Fortran (1955)，名称取自"FORmula TRANslator"(公式翻译器)，由约翰·巴科斯等人所发明；

LISP，名称取自"LISt Processor"(列举处理器)，由约翰·麦卡锡等人所发明；

COBOL，名称取自"COmmon Business Oriented Language"(通用商业导向语言)，由被葛丽丝·霍普深刻影响的Short Range Committee所发明。

另一个1950年代晚期的里程碑是由美国与欧洲计算机学者针对"算法的新语言"所组成的委员会出版的ALGOL 60报告(名称取自"ALGOrithmic Language"(算法语言))。这份报告强化了当时许多关于计算的想法，并提出了两个语言上的创新功能：

巢状区块结构：可以将有意义的程式码片段群组成一个区块(block)，而非转成分散且特定命名的程序。

词汇范围(lexical scoping)：区块可以有区块外部无法透过名称存取，属于区块本身的变数、程序以及函式。

另一个创新则是关于语言的描述方式：

一种名为巴科斯-诺尔范式 (BNF)的数学化精确符号被用于描述语言的语法。之后的编程语言几乎全部都采用类似BNF的方式来描述程式语法中上下文无关的部分。 Algol 60对之后语言的设计上带来了特殊的影响，部分的语言很快的就被广泛采用。后续为了开发Algol的扩充子集合，设计了一个名为Burroughs(en:Burroughs large systems)的大型系统。

延续Algol的关键构想所产生的成果就是ALGOL 68：

语法跟语意变的更加正交(orthogonal)，采用匿名的历程(routines)，采用高阶(higher-order)功能的递回式输入(typing)系统等等。

整个语言及语意的部分都透过为了描述语言而特别设计的Van Wijngaarden grammar来进行正式的定义，而不仅止于上下文无关的部分。

Algol 68一些较少被使用到的语言功能(如同步与并列区块)、语法捷径的复杂系统，以及型态自动强制转换(coercions)，使得实作者兴趣缺缺，也让Algol 68获得了很难用(diffcult)的名声。尼克劳斯·维尔特就干脆离开该设计委员会，另外在开发出更简单的Pascal语言。

在这段期间被开发出来的重要语言包括有：

- 1951 - Regional Assembly Language
- 1952 - Autocode
- 1954 - FORTRAN
- 1954 - IPL (LISP的先驱)
- 1955 - FLOW-MATIC (COBOL的先驱)
- 1957 - COMTRAN (COBOL的先驱)
- 1958 - LISP
- 1958 - ALGOL 58
- 1959 - FACT (COBOL的先驱)
- 1959 - COBOL
- 1962 - APL
- 1962 - Simula
- 1962 - SNOBOL
- 1963 - CPL (C的先驱)
- 1964 - BASIC
- 1964 - PL/I
- 1967 - BCPL (C的先驱)

1967-1978：确立了基础范式

1960年代晚期至1970年代晚期的期间中，编程语言的发展也有了重大的成果。大多数现在所使用的主要语言范式都是在这段期间中发明的：

Simula，于1960年代晚期由奈加特与Dahl以Algol 60超集合的方式发展，同时也是第一个设计支援面向对象进行开发的编程语言。

C，于1969至1973年间由贝尔实验室的研究人员丹尼斯·里奇与肯·汤普逊所开发，是一种早期的系统程式设计(en:system programming)语言。

Smalltalk，于1970年代中期所开发，是一个完全从零开始(ground-up)设计的面向对象编程语言。

Prolog，于1972年由Colmerauer、Roussel，以及Kowalski所设计，是第一个逻辑程式语言。

ML，于1973年由罗宾·米尔纳所发明，是一个基于Lisp所建构的多型(polymorphic)型态系统，同时也是静态型别函数编程语言的先驱。

这些语言都各自演展出自己的家族分支，现今多数现代编程语言的祖先都可以追朔他们其中至少一个以上。

在1960年代以及1970年代中结构化程式设计的优点也带来许多的争议，特别是在程式开发的过程中完全不使用GOTO。这项争议跟语言本身的设计非常有关系：某些语言并没有包含GOTO，这也强迫程序员必须结构化地编写程式。尽管这个争议在当时吵翻了天，但几乎所有的程序员都同意就算语言本身有提供GOTO的功能，在除了少数罕见的情况下去使用GOTO是种不良的程序风格。结果是之后世代的编程语言设计者发觉到结构化编程语言的争议实在既乏味又令人眼花撩乱。

在这段期间被开发出来的重要语言包括有：

- 1968 - Logo
- 1970 - Pascal
- 1970 - Forth
- 1972 - C语言
- 1972 - Smalltalk
- 1972 - Prolog
- 1973 - ML
- 1975 - Scheme
- 1978 - SQL (起先只是一种查询语言，扩充之后也具备了程式结构)

1980年代：增强、模组、效能

1980年代的编程语言与之前相较显得更为强大。C++合并了面向对象以及系统程式设计。美国政府标准化一种名为Ada的系统编程语言并提供给国防承包商使用。日本以及其他地方运用了大量的资金对采用逻辑编程语言结构的第五代语言进行研究。函数编程语言社群则把焦点转移到标准化ML及Lisp身上。这些活动都不是在开发新的范式，而是在将上个世代发明的构想进一步发扬光大。

然而，在语言设计上有个重大的新趋势，就是研究运用模组或大型组织化的程式单元来进行大型系统的开发。Modula、Ada，以及ML都在1980年代发展出值得注意的模组化系统。模组化系统常拘泥于采用泛型程式设计结构：泛型存在(generics being)、本质(essence)，参数化模组(parameterized modules)。

尽管没有出现新的主要编程语言范式，许多研究人员仍就扩充之前语言的构想并将它们运用到新的内容上。举例来说，Argus以及Emerald系统的语言配合面向对象语言运用到分散式系统上。

1980年代的编程语言实作情况也有所进展。计算机系统结构中RISC的进展假定硬件应当为编译器设计，而非身为人类的组合语言程序员。借由中央处理器速度增快的帮助，编译技术也越来越积极，RISC的进展对高阶语言编译技术带来不小的关注。

语言技术持续这些发展并迈入了1990年代。

在这段期间被开发出来的重要语言包括有：

- 1980 - Ada
- 1983 - C++ (就像有类别的C)
- 1984 - Common Lisp
- 1984 - MATLAB
- 1985 - Eiffel
- 1986 - Objective-C
- 1986 - Erlang
- 1987 - Perl
- 1988 - Tcl
- 1989 - FL (Backus)

1990年代：互联网时代

1990年代未见到有什么重大的创新，大多都是以前构想的重组或变化。这段期间主要在推动的哲学是提升程序员的生产力。许多"快速应用程序开发" (RAD) 语言也应运而生，这些语言大多都有相应的集成开发环境、垃圾回收等机制，且大多是先前语言的衍生语言。这类型的语言也大多是面向对象的编程语言，包含有Object Pascal、Visual Basic，以及C#。Java则是更加保守的语言，也具备垃圾回收机制。与其他类似语言相比，也受到更多的观注。新的脚本语言则比RAD语言更新更好。这种语言并非直接从其他语言衍生，而且新的语法更加开放地(liberal)与功能契合。虽然脚本语言比RAD语言来的更有生产力，但大多会有因为小程式较为简单，但是大型程式则难以使用脚本语言撰写并维护的顾虑。尽管如此，脚本语言还是网络层面的应用上大放异彩。 在这段期间被开发出来的重要语言包括有：

- 1990 - Haskell
- 1991 - Python
- 1991 - Visual Basic
- 1991 - HTML
- 1993 - Ruby
- 1993 - Lua
- 1994 - CLOS (part of ANSI Common Lisp)
- 1995 - Java
- 1995 - Delphi (Object Pascal)
- 1995 - JavaScript
- 1995 - PHP
- 1997 - REBOL
- 1999 - D

现今的趋势

编程语言持续在学术及企业两个层面中发展进化，目前的一些趋势包含有：

在语言中增加安全性与可靠性验证机制：额外的堆叠检查、资讯流(information flow)控制，以及静态执行绪安全。 提供模组化的替代机制：混入(en:mixin)、委派(en:delegates)，以及观点导向。 元件导向(component-oriented)软件开发 元编程、反射或是存取抽象语法树(en:Abstract syntax tree) 更重视分散式及移动式的应用。 与数据库的整合，包含XML及关联式数据库。 支援使用Unicode编写程式，所以源代码不会受到ASCII字元集的限制，而可以使用像是非拉丁语系的脚本或延伸标点符号。 图形化使用者界面所使用的XML(XUL、XAML)。 在这段期间被开发出来的重要语言包括有：

- 2001 - C#
- 2001 - Visual Basic .NET
- 2002 - F#
- 2003 - Scala
- 2003 - Factor
- 2006 - Windows PowerShell
- 2007 - Clojure
- 2009 - Go

编程语言发展史上的杰出人物

- 约翰·巴科斯，发明了Fortran。
- 阿兰·库珀，开发了Visual Basic。
- 艾兹格·迪杰斯特拉，开创了正确运用编程语言(proper programming)的框架。
- 詹姆斯·高斯林，开发了Oak，该语言为Java的先驱。
- 安德斯·海尔斯伯格，开发了Turbo Pascal、Delphi，以及C#。
- 葛丽丝·霍普，开发了Flow-Matic，该语言对COBOL造成了影响。
- 肯尼斯·艾佛森，开发了APL，并与Roger Hui合作开发了J。
- 比尔·乔伊，发明了vi，BSD Unix的前期作者，以及SunOS的发起人，该操作系统后来改名为Solaris。
- 艾伦·凯，开创了面向对象编程语言，以及Smalltalk的发起人。Brian Kernighan，与丹尼斯·里奇合著第一本C程式设计语言的书籍，同时也是AWK与AMPL程式设计语言的共同作者。
- 约翰·麦卡锡，发明了LISP。
- 约翰·冯·诺伊曼，操作系统概念的发起者。丹尼斯·里奇，发明了C。
- 比雅尼·斯特劳斯特鲁普，开发了C++。肯·汤普逊，发明了Unix。
- 尼克劳斯·维尔特，发明了Pascal与Modula。
- 拉里·沃尔，创造了Perl与Perl 6。
- 吉多·范罗苏姆，创造了Python。