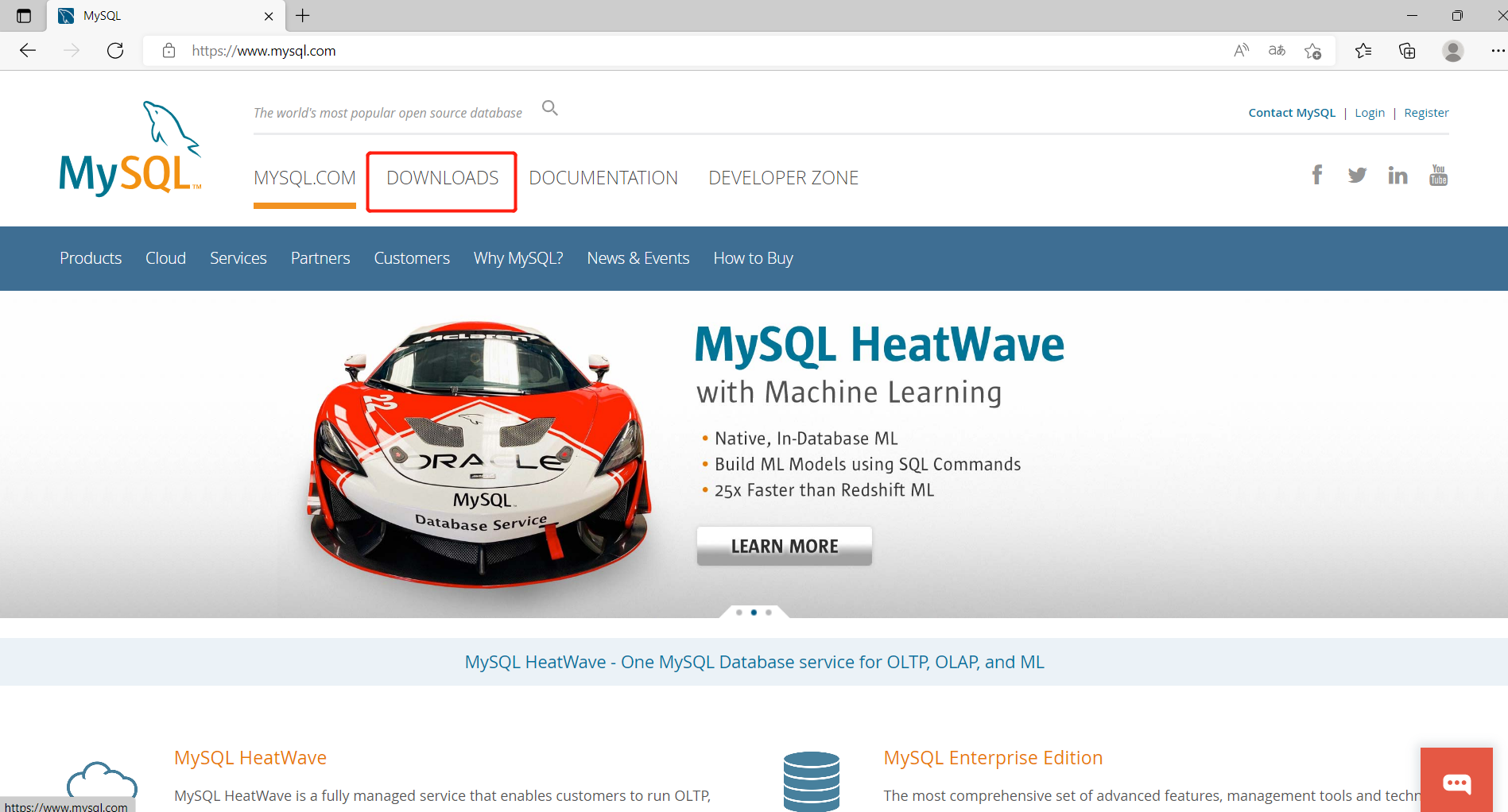
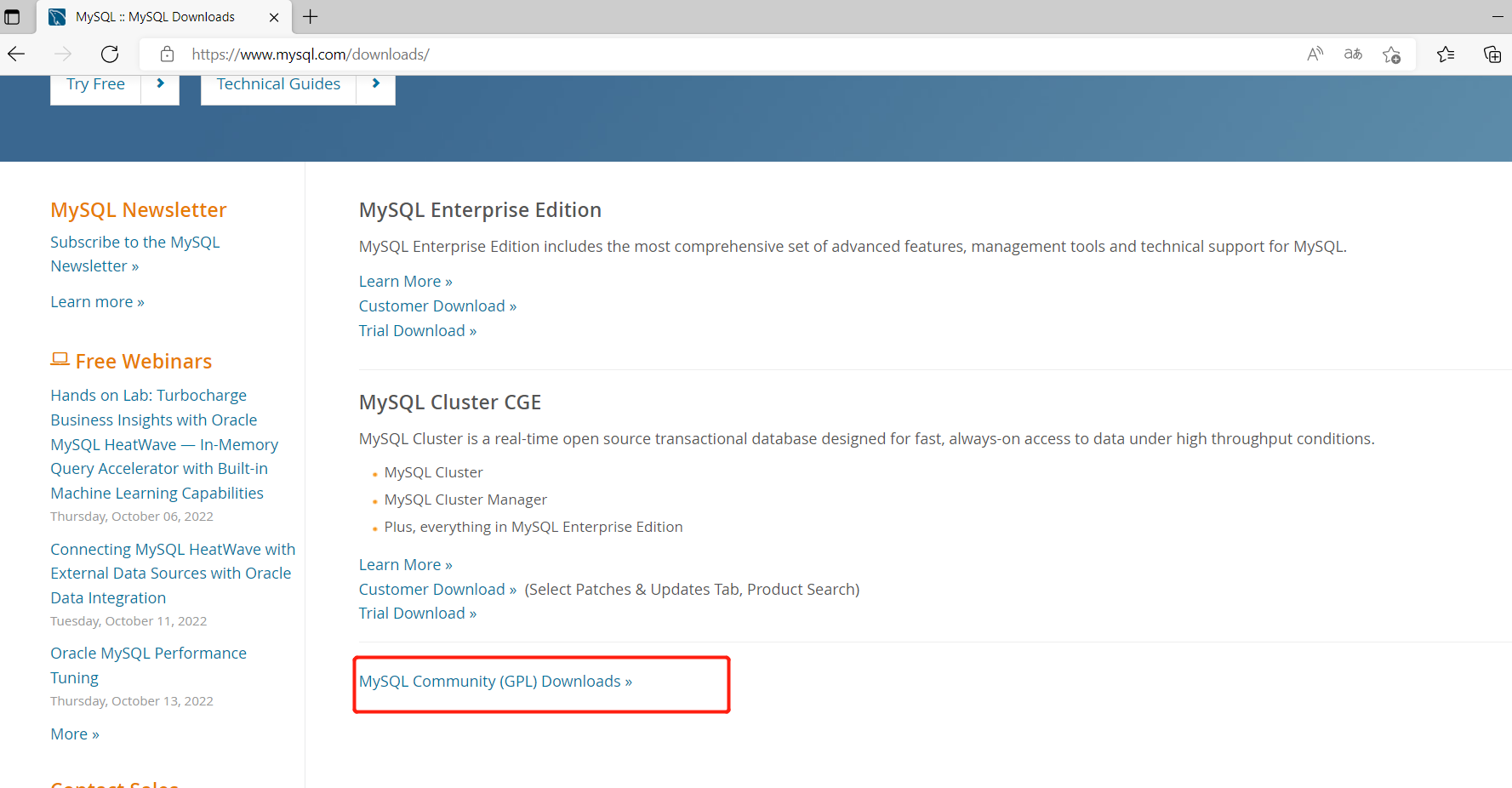
1. Mysql 数据库下载

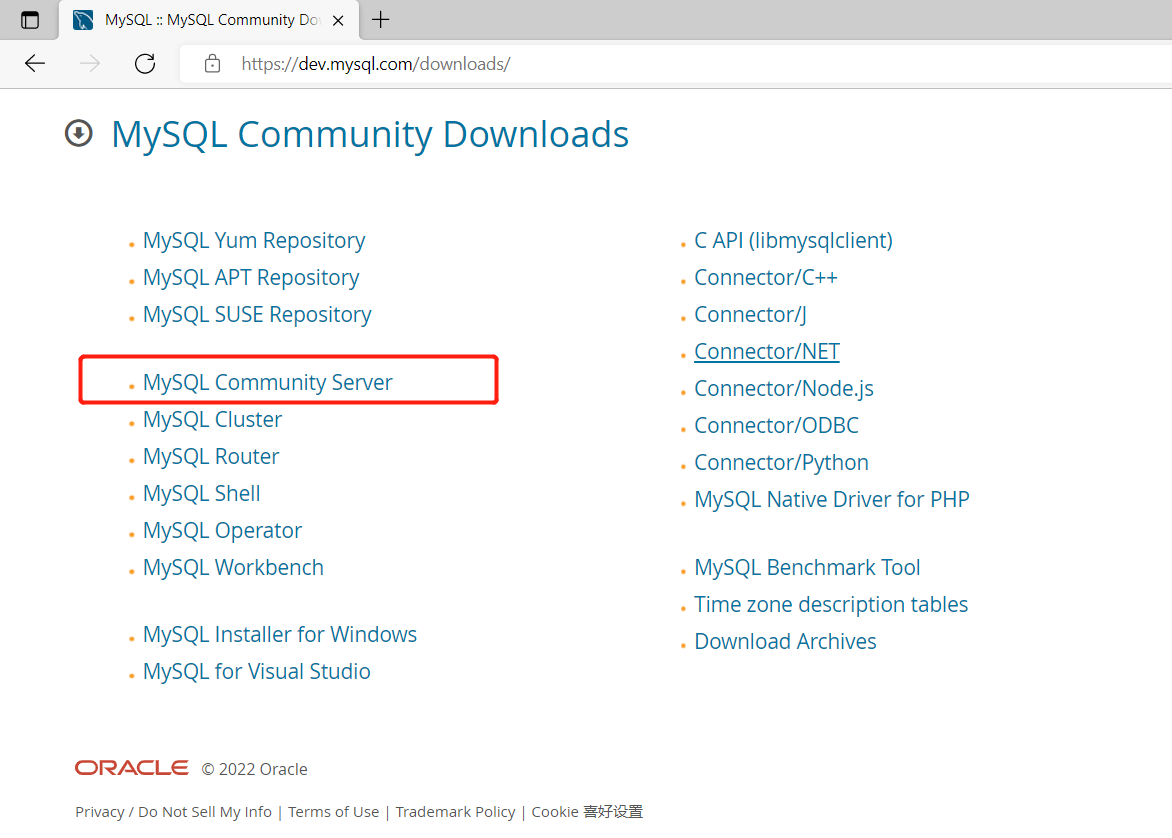
1）进入Mysql 官网，进入下载界面



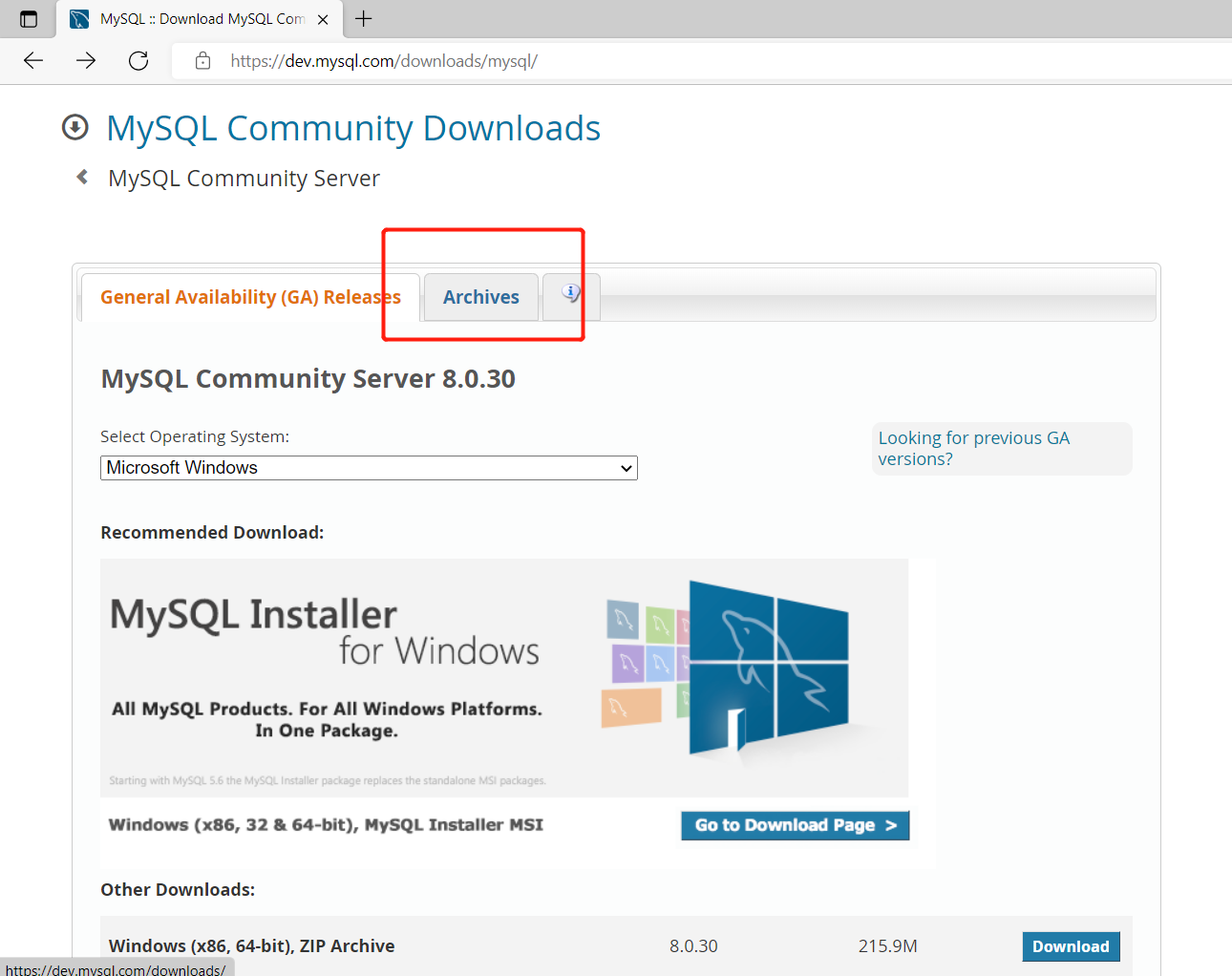
2）选择Mysql Community Downloads



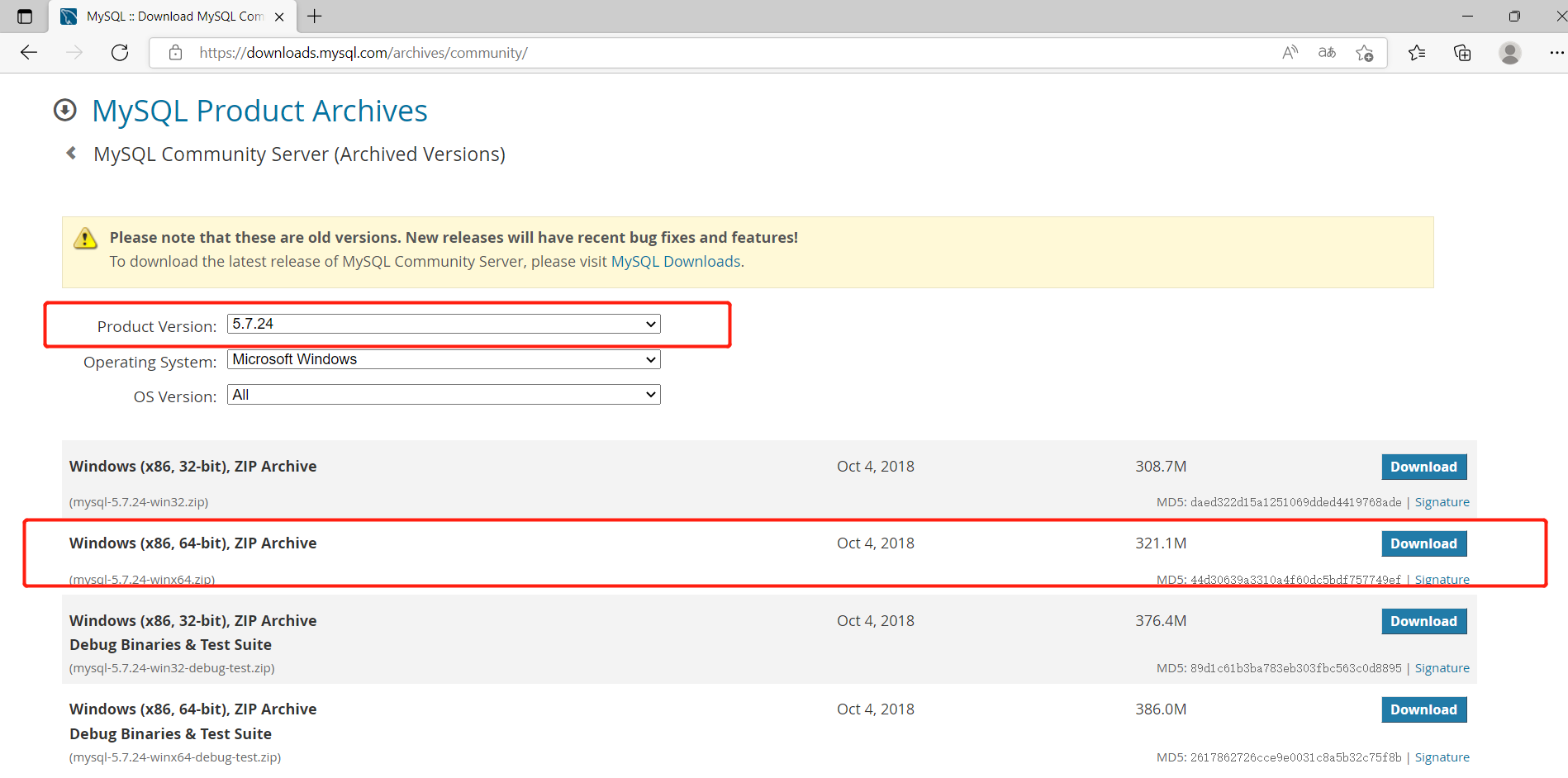
3) 选择Mysql Community Server



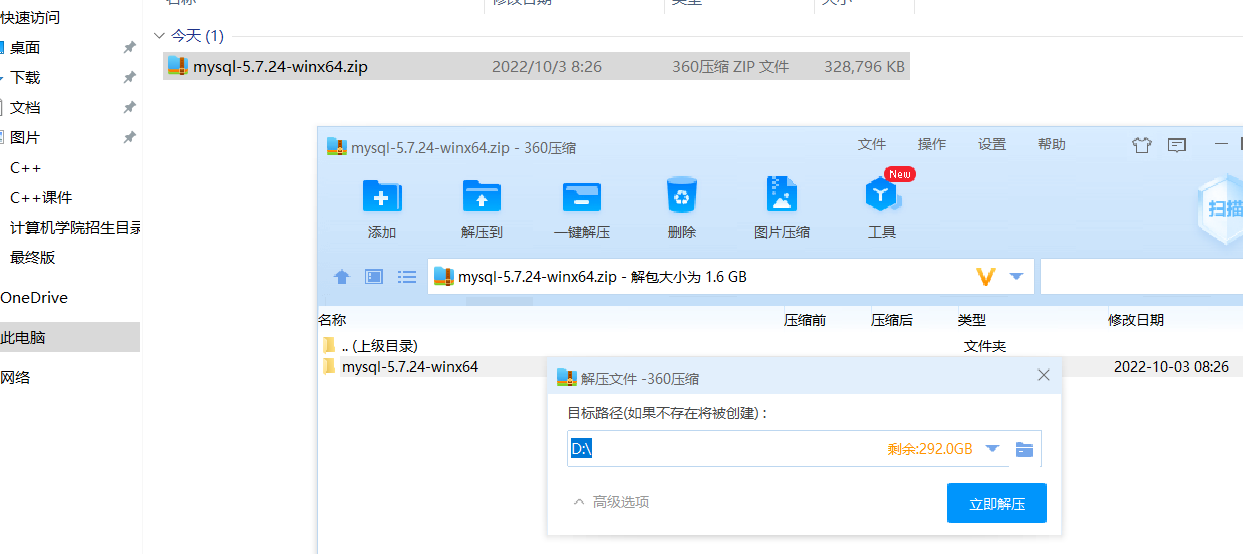
4）可以选择之前版本

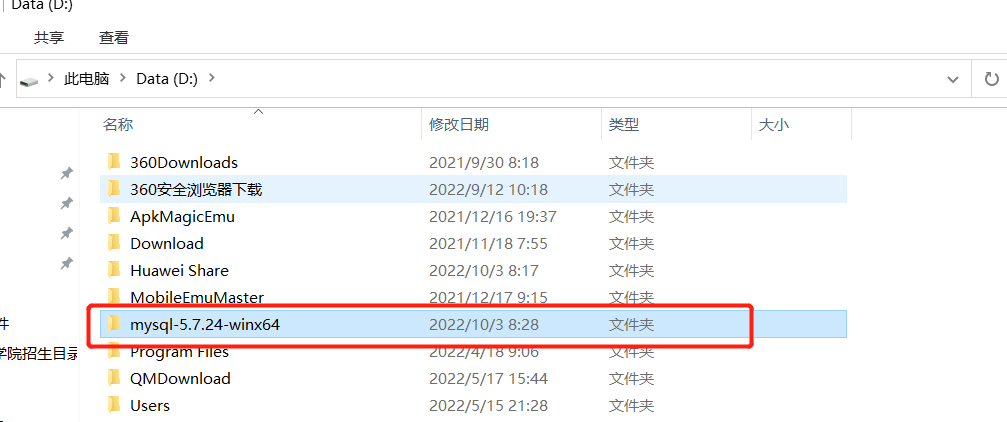


5）开始下载



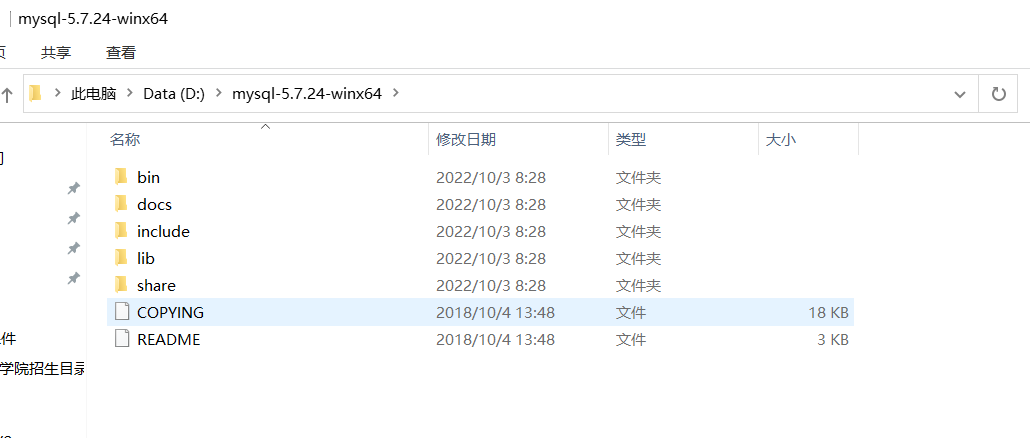
6）将下载好的压缩包解压缩





7）设置配置文件

a）解压缩后没有配置文件，需要自定义



b）定义配置文件，同时新建一个文件夹，命名为data，与配置文件中数据存放位置定义的名称一致。

[mysql]

# 设置 MySQL 客户端默认字符集

default-character-set=utf8

[mysqld]

#设置 MySQL 端口号

port = 3306

#skip-grant-tables

# 设置 MySQL 的安装目录

basedir=D:/mysql-5.7.24-winx64

# 设置 MySQL 数据库的数据的存放目录

datadir=D:/mysql-5.7.24-winx64/data

# 允许最大连接数

max\_connections=200

# 服务端使用的字符集

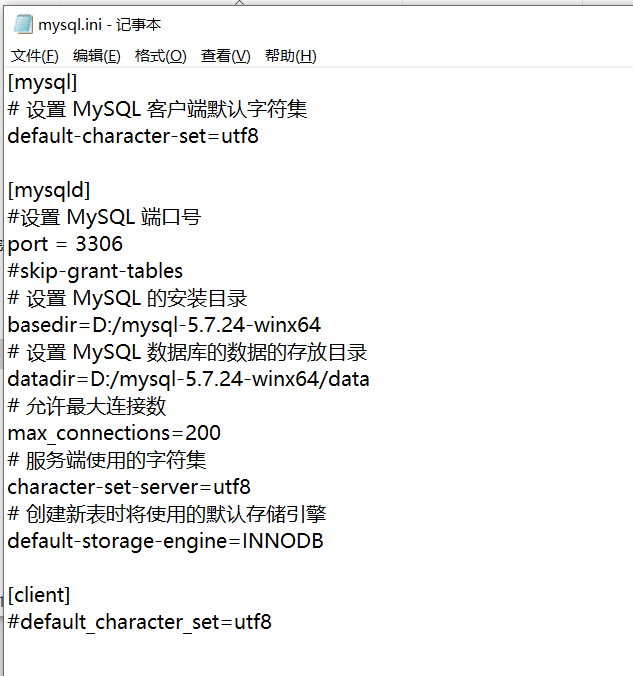
character-set-server=utf8

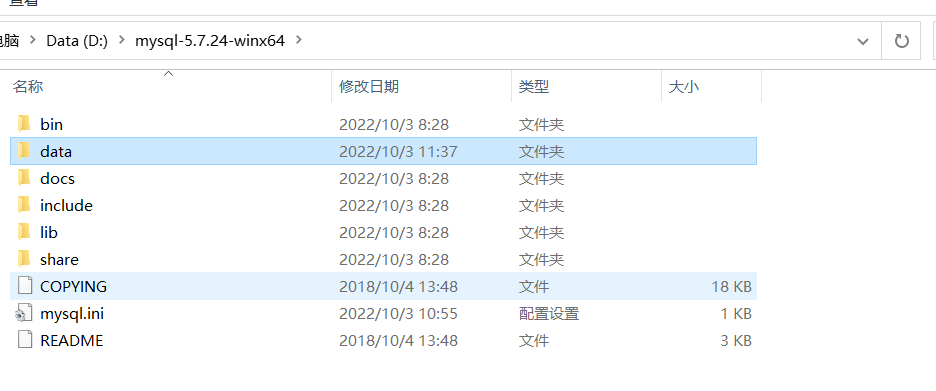
# 创建新表时将使用的默认存储引擎

default-storage-engine=INNODB

[client]

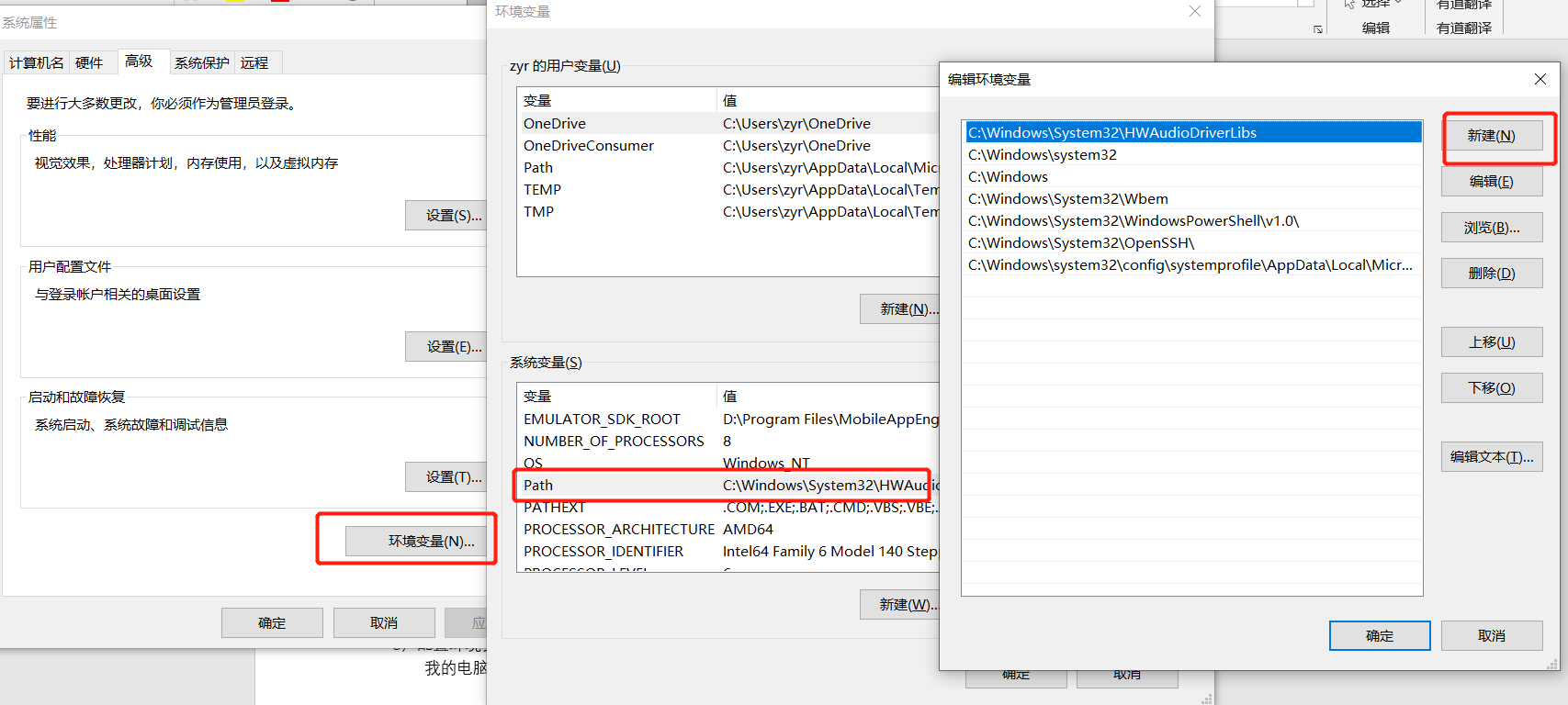
#default\_character\_set=utf8



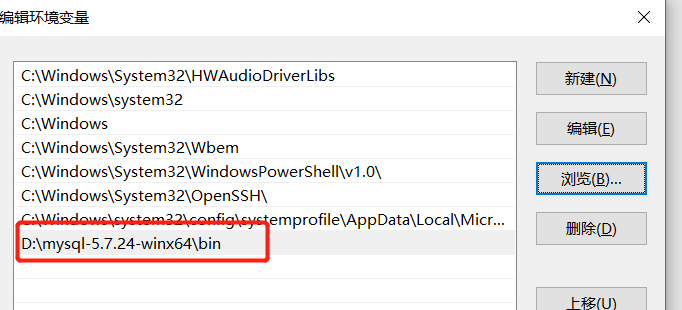


8）配置环境变量

我的电脑 -- 属性 – 高级环境设置 – 环境变量 – 系统变量 – 选择PATH – 新建

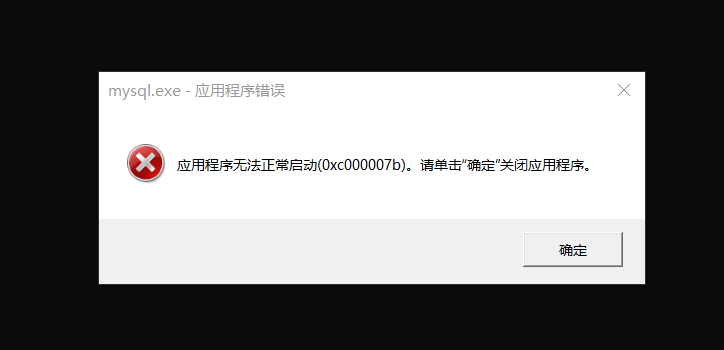


将Mysql 文件夹中的bin导入。



9）重启电脑

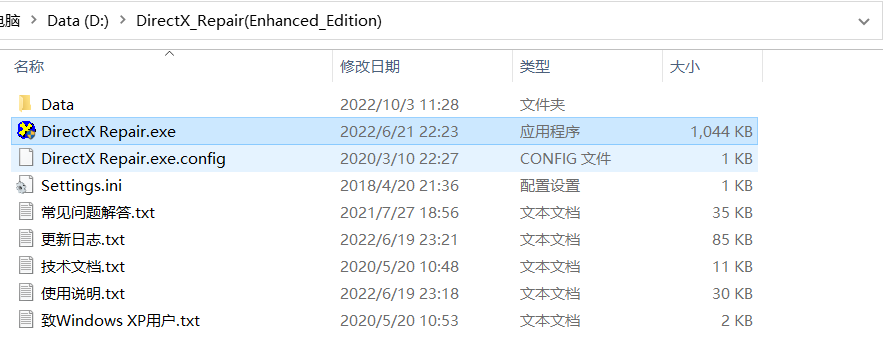
10）如果打开mysql.exe未出现下图错误，则跳过该步骤

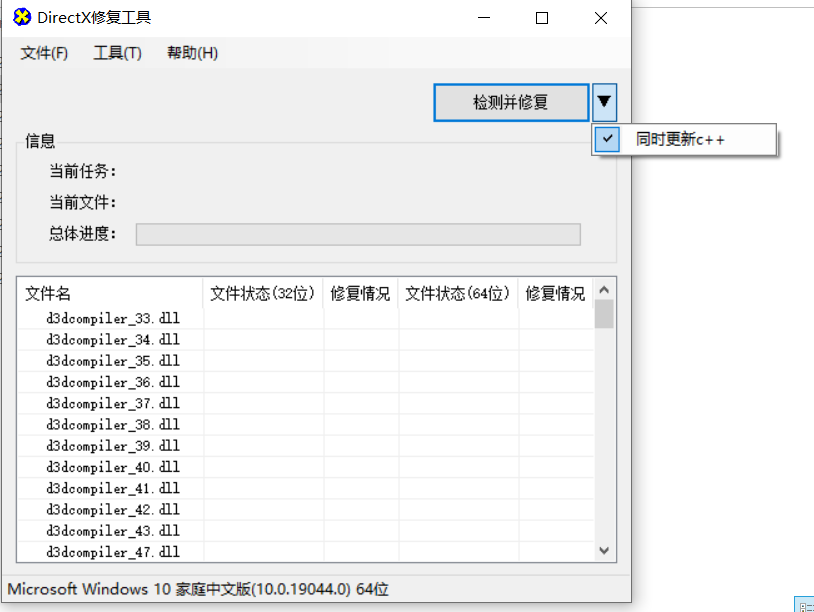


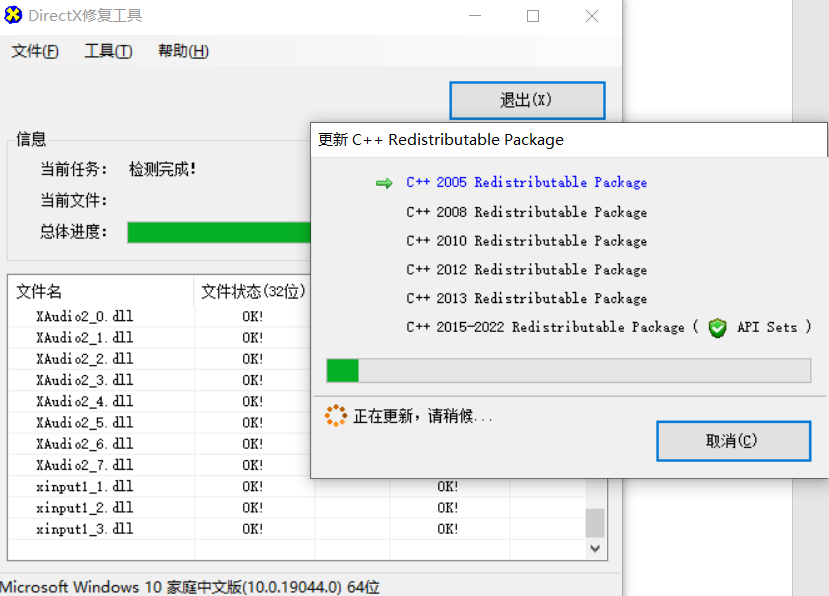
10-1）下载DirectX 修复工具

网址： <https://zhangyue667.lanzouh.com/DirectXRepairEnhanced>

10-2）运行程序

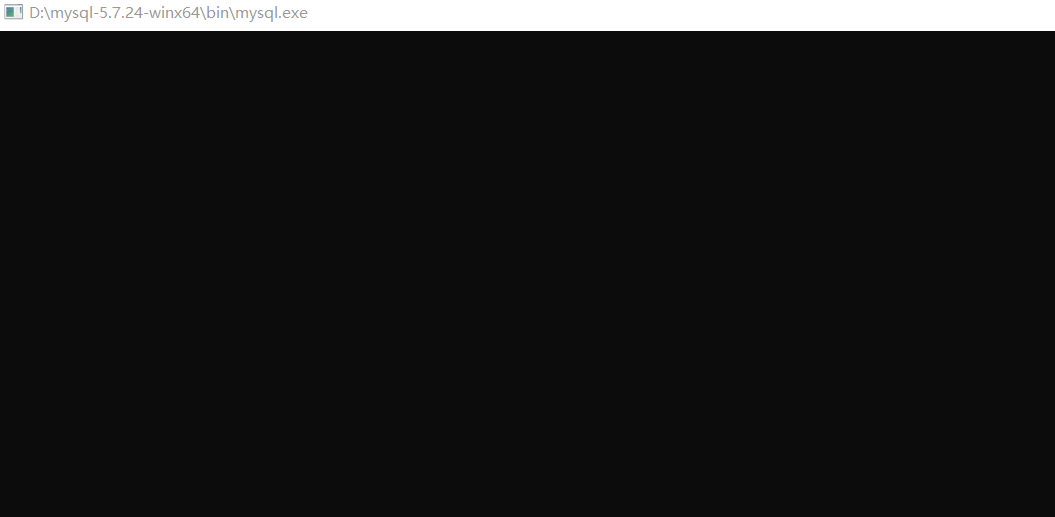






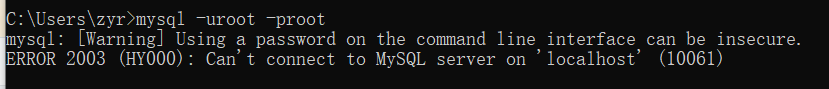
10-3）重启电脑

10-4）修复成功，启动mysql.exe测试



11）验证环境变量是否配置正确

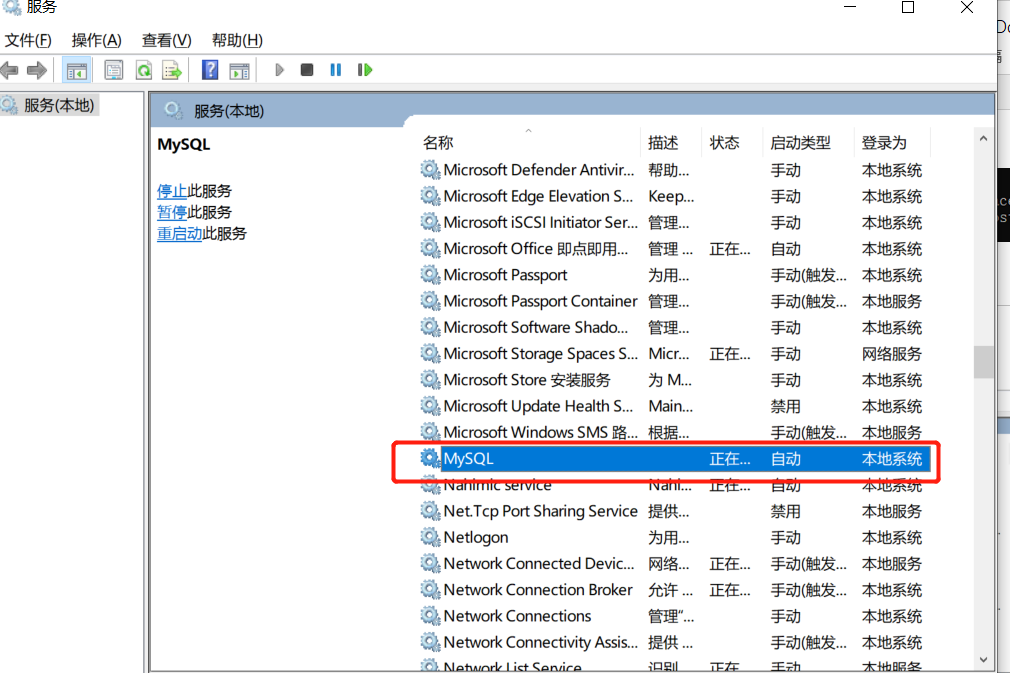
在cmd窗口中输入 mysql -uroot -proot



说明配置没有问题

12）修复上述出现的问题

12-0) 首先确保服务中启动了Mysql



若未启动出现问题，则先启动。

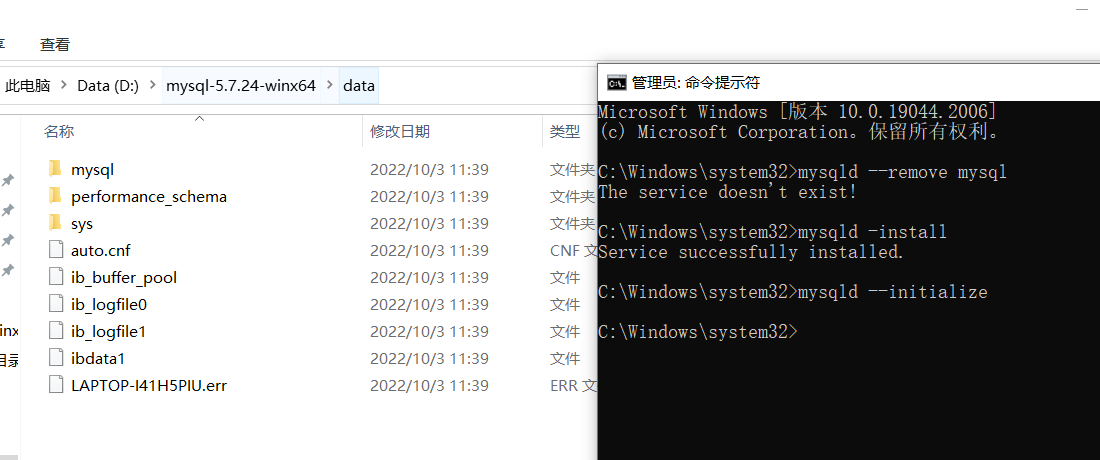
12-1）以管理员身份启动[cmd](https://so.csdn.net/so/search?q=cmd&spm=1001.2101.3001.7020)，要不然服务禁止访问

12-2）在cmd中输入 mysqld --remove mysql来移除服务。

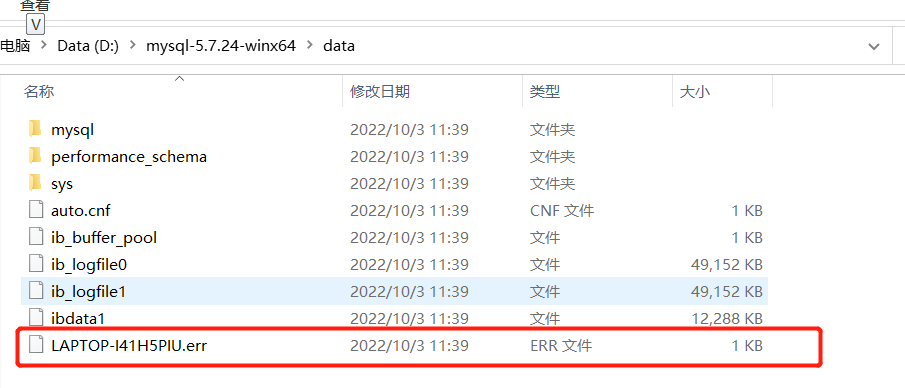
12-3）进入MySQL安装目录中找到data文件，清空其中全部文件。我的目录是D:\mysql-5.7.24-winx64\data，没有data文件自己建一个。

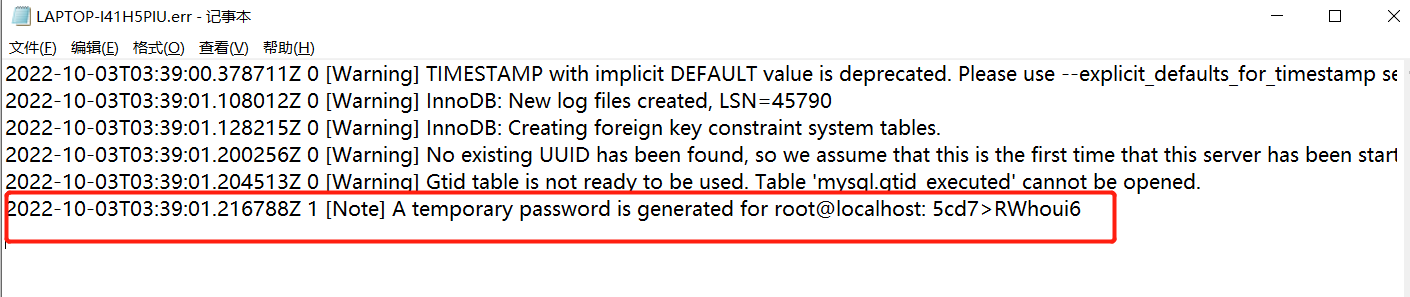
12-4）接下来注册服务，mysqld -install。

12-5）然后开始初始化，mysqld --initialize，然后可以看到data文件夹中文件正在初始化中（生成文件中）



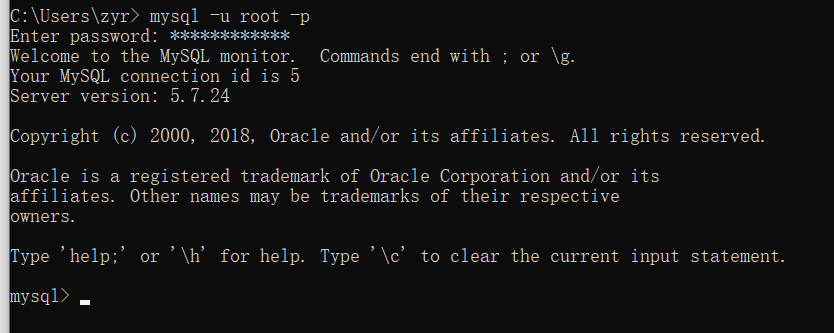
12-6）初始化过程中会把登录密码初始化，可以在data文件中找到以.err结尾的文件





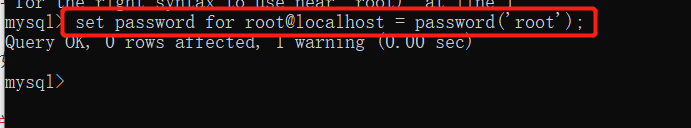
12-7）在root@localhost:后面即初始密码，记住初始密码

12-8) mysql -u root -p 后出现Enter password:输入密码即可

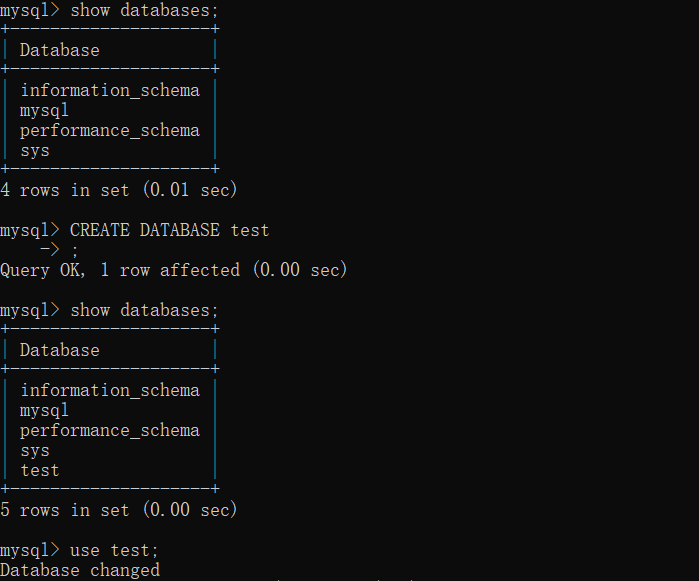


12-9）修改密码

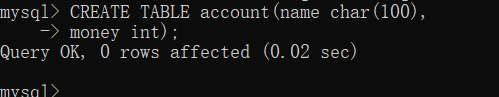
set password for username@localhost = password(‘newpwd’)



1. 创建数据库
2. 进入mysql数据库，创建一个test数据库



1. 创建一个account表

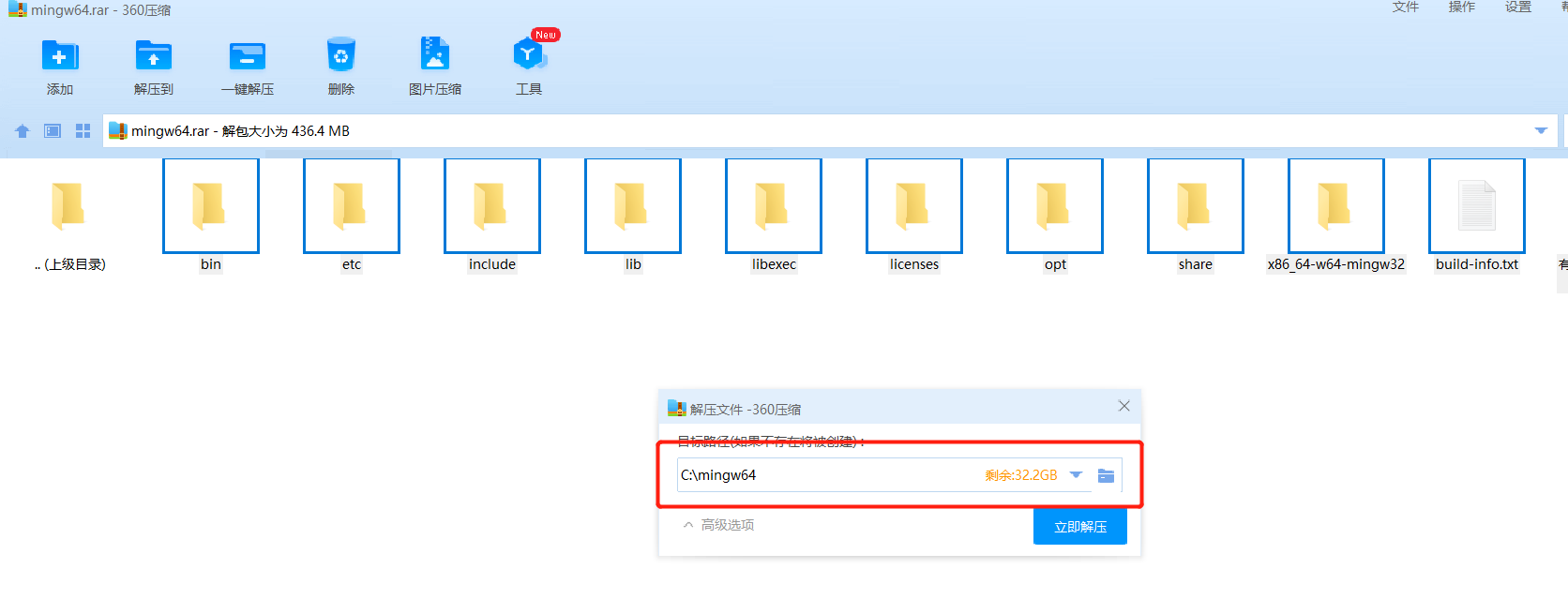


1. 展示表中数据

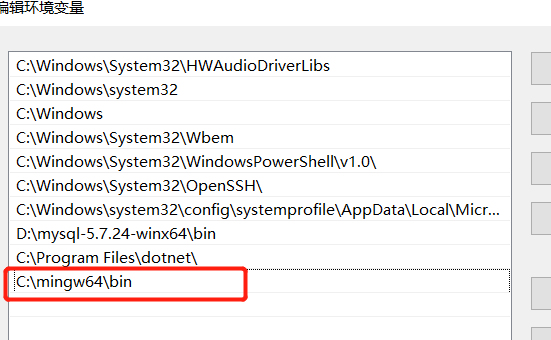


**后续会用到这个表**

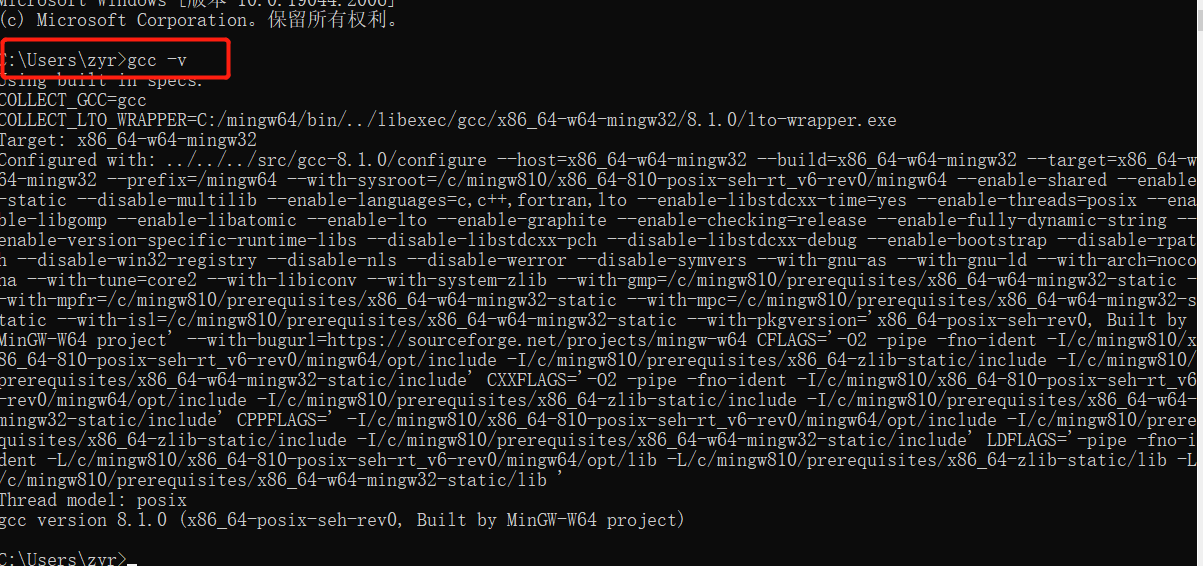
1. C++编译软件
2. mingw编译软件已经挺久不更新了，建议使用mingw64
3. 下载地址：<https://www.onlinedown.net/soft/10045442.htm>
4. 将压缩包解压至C盘



1. 将mingw64配置到环境变量中



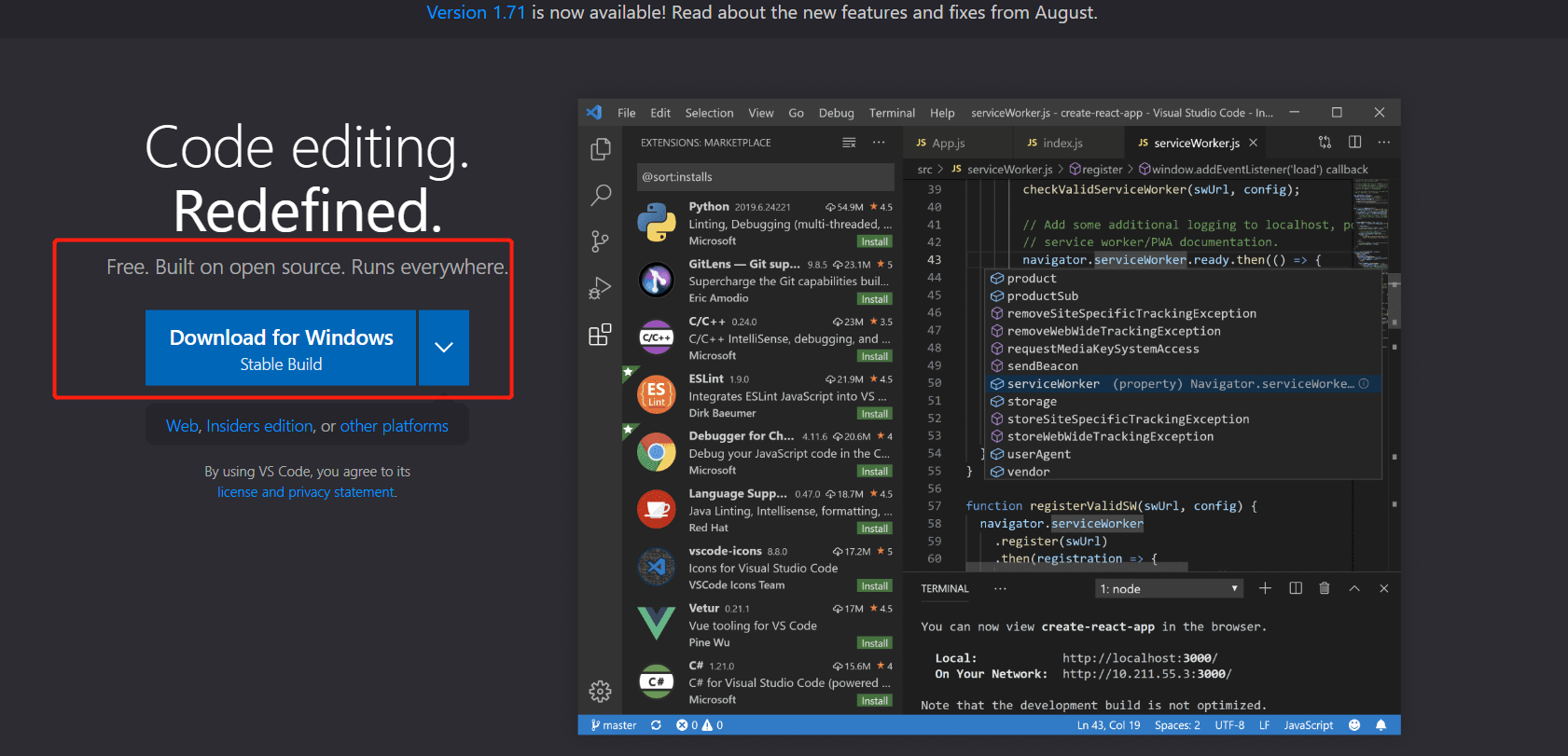
1. 验证



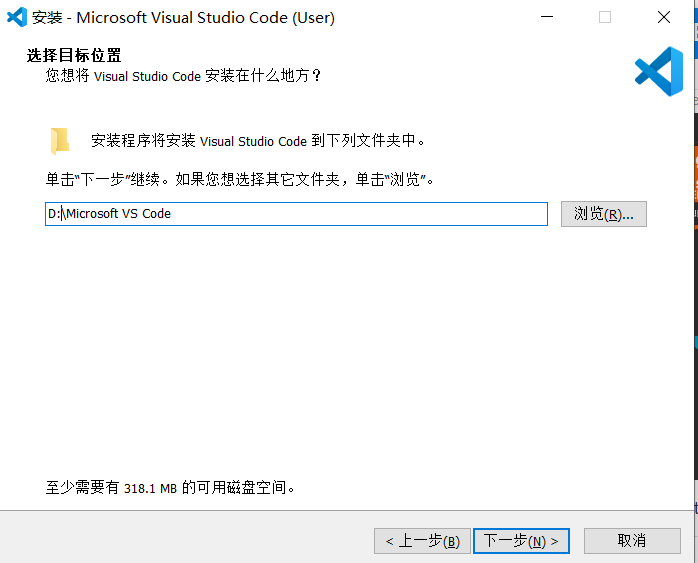
三、C++开发工具下载

1）VSCode 安装

1-1）官网 <https://code.visualstudio.com/>

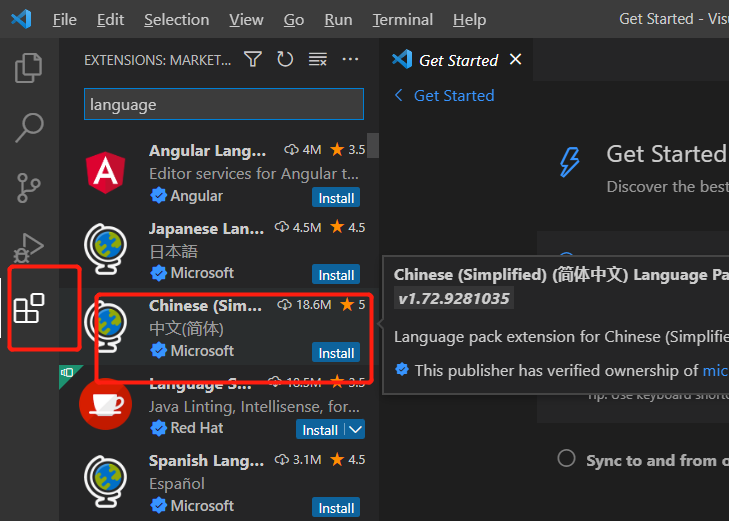


1-2）安装



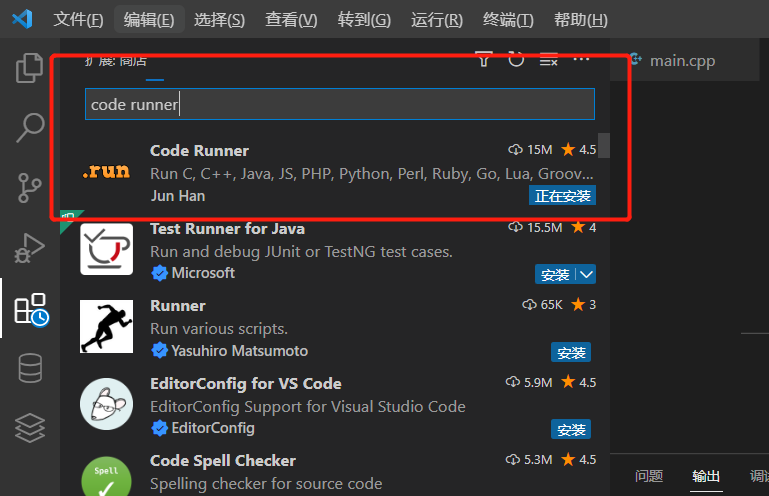
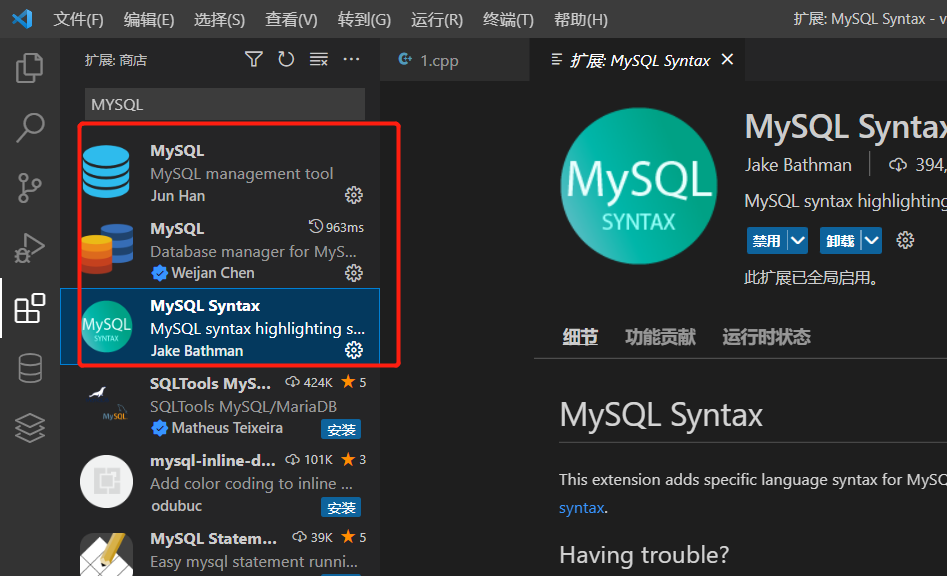
1-3）安装扩展插件

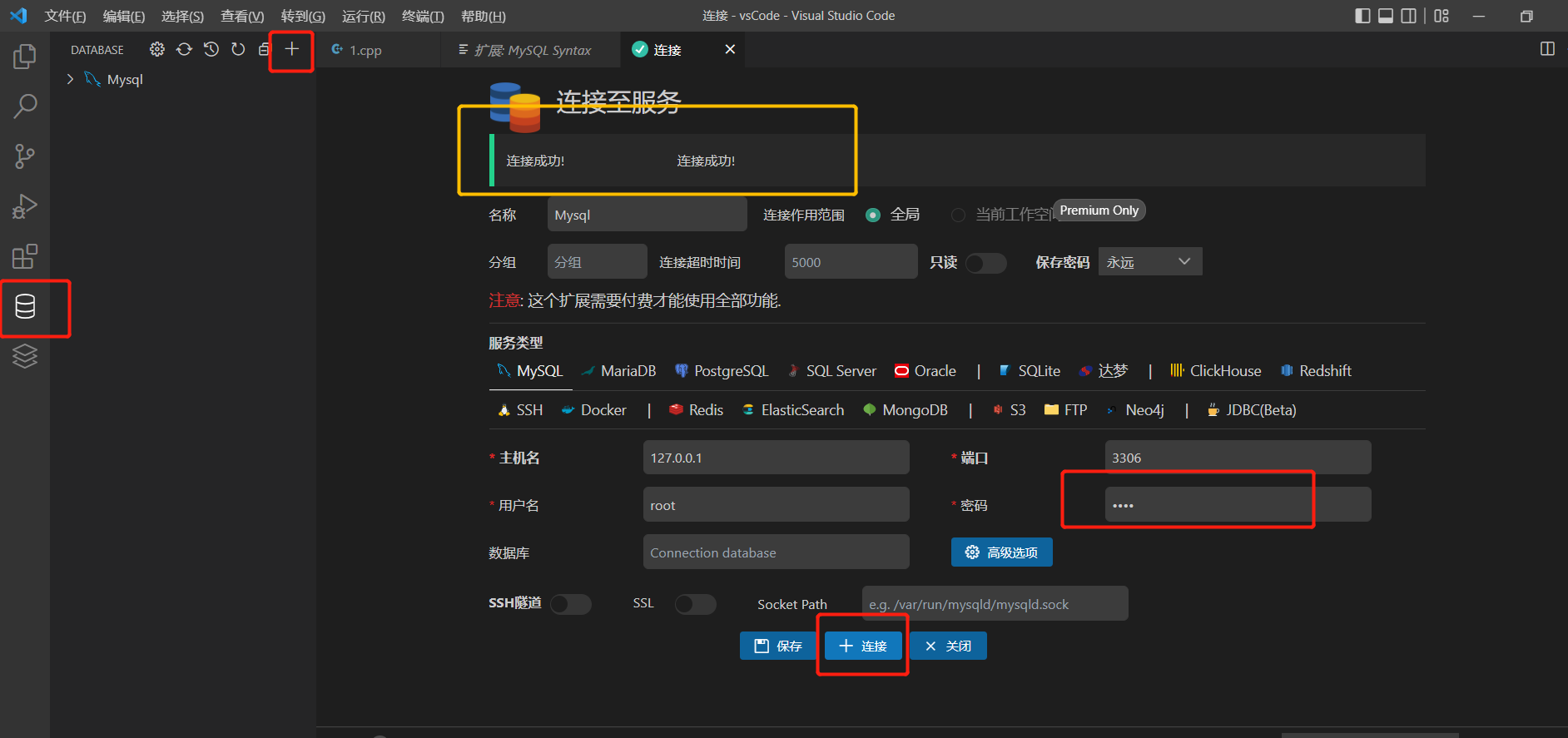
首先设置为中文环境



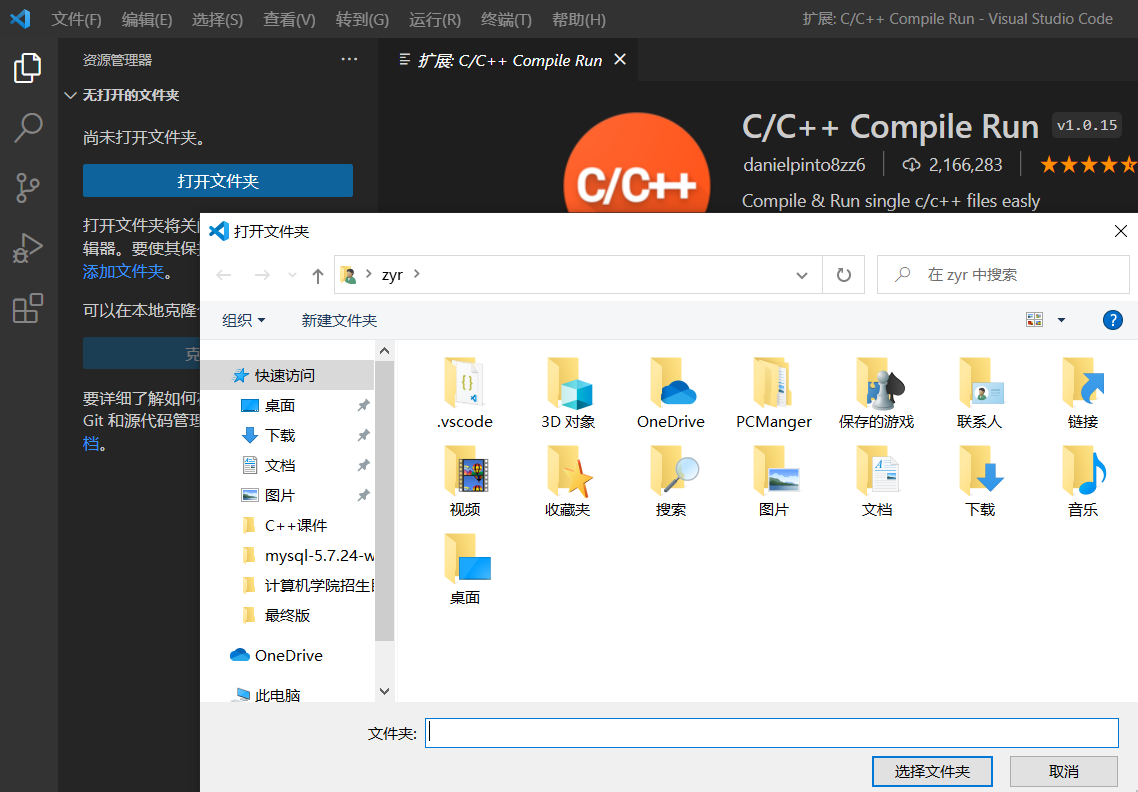
1-4）安装C++扩展



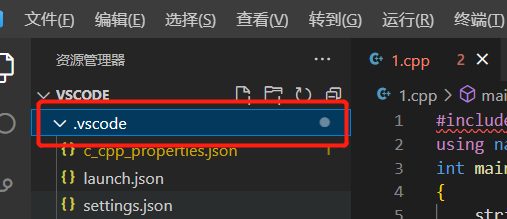
 



1-5）创建一个文件夹（用于存放代码）



1-6）

1-6）在VSCODE里面创建一个 .vscode 文件夹（如果先创建代码，系统会自动创建） 

1-7）修改里面的配置文件（如果有文件则直接覆盖，如果没有则创建）

文件：c\_cpp\_properties.json

{

    "configurations": [

      {

        "name": "Win32",

        "includePath": [

          "${workspaceFolder}/\*\*",

          "D:/mysql-5.7.24-winx64/include/\*\*"

        ],

        "defines": [

          "\_DEBUG",

          "UNICODE",

          "\_UNICODE"

        ],

        "compilerPath": "C:\\mingw64\\bin\\gcc.exe",

        "cStandard": "gnu17",

        "cppStandard": "gnu++14",

        "intelliSenseMode": "windows-gcc-x64"

      }

    ],

    "version": 4

  }

文件：launch.json

// https://code.visualstudio.com/docs/cpp/launch-json-reference

{

    "version": "0.2.0",

    "configurations": [{

        "name": "(gdb) Launch", // 配置名称，将会在启动配置的下拉菜单中显示

        "type": "cppdbg", // 配置类型，对于C/C++可认为此处只能是cppdbg，由cpptools提供；不同编程语言不同

        "request": "launch", // 可以为launch（启动）或attach（附加）

        "program": "${fileDirname}\\bin\\${fileBasenameNoExtension}.exe", // 将要进行调试的程序的路径

        "args": [], // 程序调试时传递给程序的命令行参数，一般设为空

        "stopAtEntry": false, // 设为true时程序将暂停在程序入口处，相当于在main上打断点

        "cwd": "${workspaceFolder}", // 调试程序时的工作目录，此为工作区文件夹；改成${fileDirname}可变为文件所在目录

        "environment": [], // 环境变量

        "externalConsole": false, // 使用单独的cmd窗口，与其它IDE一致；为false时使用内置终端

        "internalConsoleOptions": "neverOpen", // 如果不设为neverOpen，调试时会跳到“调试控制台”选项卡，你应该不需要对gdb手动输命令吧？

        "MIMode": "gdb", // 指定连接的调试器，可以为gdb或lldb。但我没试过lldb

        "miDebuggerPath": "gdb.exe", // 调试器路径，Windows下后缀不能省略，Linux下则不要

        "setupCommands": [

            { // 模板自带，好像可以更好地显示STL容器的内容，具体作用自行Google

                "description": "Enable pretty-printing for gdb",

                "text": "-enable-pretty-printing",

                "ignoreFailures": false

            }

        ],

        "preLaunchTask": "Compile" // 调试前执行的任务，一般为编译程序。与tasks.json的label相对应

    }]

}

文件：settings.json

{

    "files.defaultLanguage": "c++", // ctrl+N新建文件后默认的语言

    "editor.formatOnType": true,  // 输入分号(C/C++的语句结束标识)后自动格式化当前这一行的代码

    "editor.suggest.snippetsPreventQuickSuggestions": false, // clangd的snippets有很多的跳转点，不用这个就必须手动触发Intellisense了

    "editor.acceptSuggestionOnEnter": "off", // 我个人的习惯，按回车时一定是真正的换行，只有tab才会接受Intellisense

    // "editor.snippetSuggestions": "top", // （可选）snippets显示在补全列表顶端，默认是inline

    "code-runner.runInTerminal": true, // 设置成false会在“输出”中输出，无法输入

    "code-runner.executorMap": {

        // "c": "gcc '$fileName' -o '$fileNameWithoutExt.exe' -Wall -m64 -lm -static-libgcc -fexec-charset=GBK -D\_\_USE\_MINGW\_ANSI\_STDIO && &'./$fileNameWithoutExt.exe'",

       // "cpp": "g++ '$fileName' -o '$fileNameWithoutExt.exe' -Wall -m64 -static-libgcc -fexec-charset=GBK && &'./$fileNameWithoutExt.exe'"

        // "cpp": "g++ '$fileName' -o '$fileNameWithoutExt.exe' -Wall -m64 -static-libgcc -fexec-charset=GBK && &'./$fileNameWithoutExt.exe'"

        "c": "gcc $fileName -o $fileNameWithoutExt.exe -Wall -O2 -m64 -lm -static-libgcc -fexec-charset=GBK -D\_\_USE\_MINGW\_ANSI\_STDIO && $dir$fileNameWithoutExt.exe",

        "cpp": "g++ $fileName -o $fileNameWithoutExt.exe -I D:/mysql-5.7.24-winx64/include -L D:/mysql-5.7.24-winx64/lib -llibmysql -Wall -O2 -m64 -static-libgcc -fexec-charset=GBK && $dir$fileNameWithoutExt.exe"

    }, // 右键run code时运行的命令；未注释的仅适用于PowerShell（Win10默认）和pwsh，文件名中有空格也可以编译运行；注释掉的适用于cmd（win7默认）、PS和bash，但文件名中有空格时无法运行

    "code-runner.saveFileBeforeRun": true, // run code前保存

    "code-runner.preserveFocus": true,     // 若为false，run code后光标会聚焦到终端上。如果需要频繁输入数据可设为false

    "code-runner.clearPreviousOutput": false, // 每次run code前清空属于code runner的终端消息，默认false

    "code-runner.ignoreSelection": true,   // 默认为false，效果是鼠标选中一块代码后可以单独执行，但C是编译型语言，不适合这样用

    "code-runner.fileDirectoryAsCwd": true, // 将code runner终端的工作目录切换到文件目录再运行，对依赖cwd的程序产生影响；如果为false，executorMap要加cd $dir

    "C\_Cpp.clang\_format\_sortIncludes": true,

    "files.associations": {

        "\*.tcc": "cpp",

        "string": "cpp",

        "istream": "cpp",

        "iostream": "cpp",

        "iosfwd": "cpp",

        "array": "cpp",

        "atomic": "cpp",

        "hash\_map": "cpp",

        "bitset": "cpp",

        "cctype": "cpp",

        "cfenv": "cpp",

        "charconv": "cpp",

        "chrono": "cpp",

        "cinttypes": "cpp",

        "clocale": "cpp",

        "cmath": "cpp",

        "codecvt": "cpp",

        "complex": "cpp",

        "condition\_variable": "cpp",

        "csetjmp": "cpp",

        "csignal": "cpp",

        "cstdarg": "cpp",

        "cstddef": "cpp",

        "cstdint": "cpp",

        "cstdio": "cpp",

        "cstdlib": "cpp",

        "cstring": "cpp",

        "ctime": "cpp",

        "cuchar": "cpp",

        "cwchar": "cpp",

        "cwctype": "cpp",

        "deque": "cpp",

        "forward\_list": "cpp",

        "list": "cpp",

        "unordered\_map": "cpp",

        "unordered\_set": "cpp",

        "vector": "cpp",

        "exception": "cpp",

        "algorithm": "cpp",

        "functional": "cpp",

        "iterator": "cpp",

        "map": "cpp",

        "memory": "cpp",

        "memory\_resource": "cpp",

        "numeric": "cpp",

        "optional": "cpp",

        "random": "cpp",

        "ratio": "cpp",

        "regex": "cpp",

        "set": "cpp",

        "string\_view": "cpp",

        "system\_error": "cpp",

        "tuple": "cpp",

        "type\_traits": "cpp",

        "utility": "cpp",

        "fstream": "cpp",

        "future": "cpp",

        "initializer\_list": "cpp",

        "iomanip": "cpp",

        "limits": "cpp",

        "mutex": "cpp",

        "new": "cpp",

        "ostream": "cpp",

        "scoped\_allocator": "cpp",

        "shared\_mutex": "cpp",

        "sstream": "cpp",

        "stdexcept": "cpp",

        "streambuf": "cpp",

        "thread": "cpp",

        "typeindex": "cpp",

        "typeinfo": "cpp",

        "valarray": "cpp",

        "getopt.h": "c",

        "alignedvector3": "cpp",

        "hash\_set": "cpp",

        "slist": "cpp"

    },

    "C\_Cpp.errorSquiggles": "Enabled", // 格式化时调整include的顺序（按字母排序）

}

注！：主要起作用的就是这条语句“"cpp": "g++ $fileName -o $fileNameWithoutExt.exe -I D:/mysql-5.7.24-winx64/include -L D:/mysql-5.7.24-winx64/lib -llibmysql -Wall -O2 -m64 -static-libgcc -fexec-charset=GBK && $dir$fileNameWithoutExt.exe"

“

如果熟悉Linux下编程的同学可以发现，这就是将mysql里面的include包include进来（-I 指令），在将链接下依赖库“libmysql “。至于其他语法可以学习Makefile。

文件名：tasks.json

{

    "version": "2.0.0",

    "tasks": [

      {

        "type": "cppbuild",

        "label": "C/C++: cpp.exe 生成活动文件",

        "command": "C:\\mingw64\\bin\\cpp.exe",

        "args": [

          "-g",

          "${file}",

          "-o",

          "${fileDirname}\\${fileBasenameNoExtension}.exe"

        ],

        "options": {

          "cwd": "${workspaceFolder}"

        },

        "problemMatcher": [

          "$gcc"

        ],

        "group": "build",

        "detail": "编译器: C:\\mingw64\\bin\\cpp.exe"

      }

    ]

  }

文件名：main.cpp

#include <iostream>

#include "mysql.h"

using namespace std;

int main()

{

    cout << "Hello, world!" << endl;

    // q1();

    MYSQL conn;

    int res;

    mysql\_init(&conn);

    if (mysql\_real\_connect(&conn, "127.0.0.1", "root", "root", "test", 3306, NULL, CLIENT\_FOUND\_ROWS))

    {

        printf("connect success!\n");

        res = mysql\_query(&conn, "insert into account values('use111r','123456')");

        if (res)

        {

            printf("error\n");

        }

        else

        {

            printf("OK\n");

        }

        mysql\_close(&conn);

    }

    else

    {

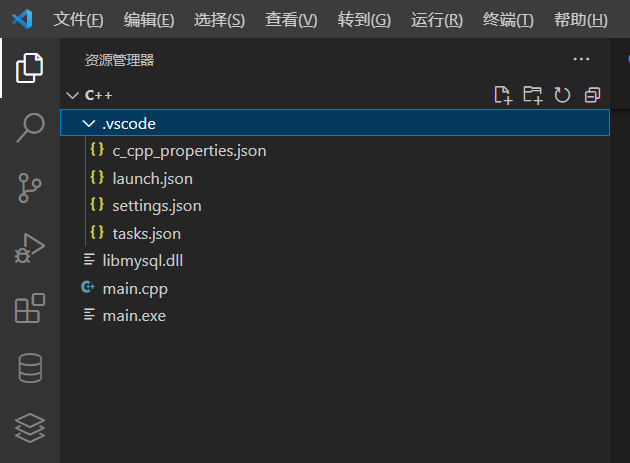
        printf("connect errot!\n");

    }

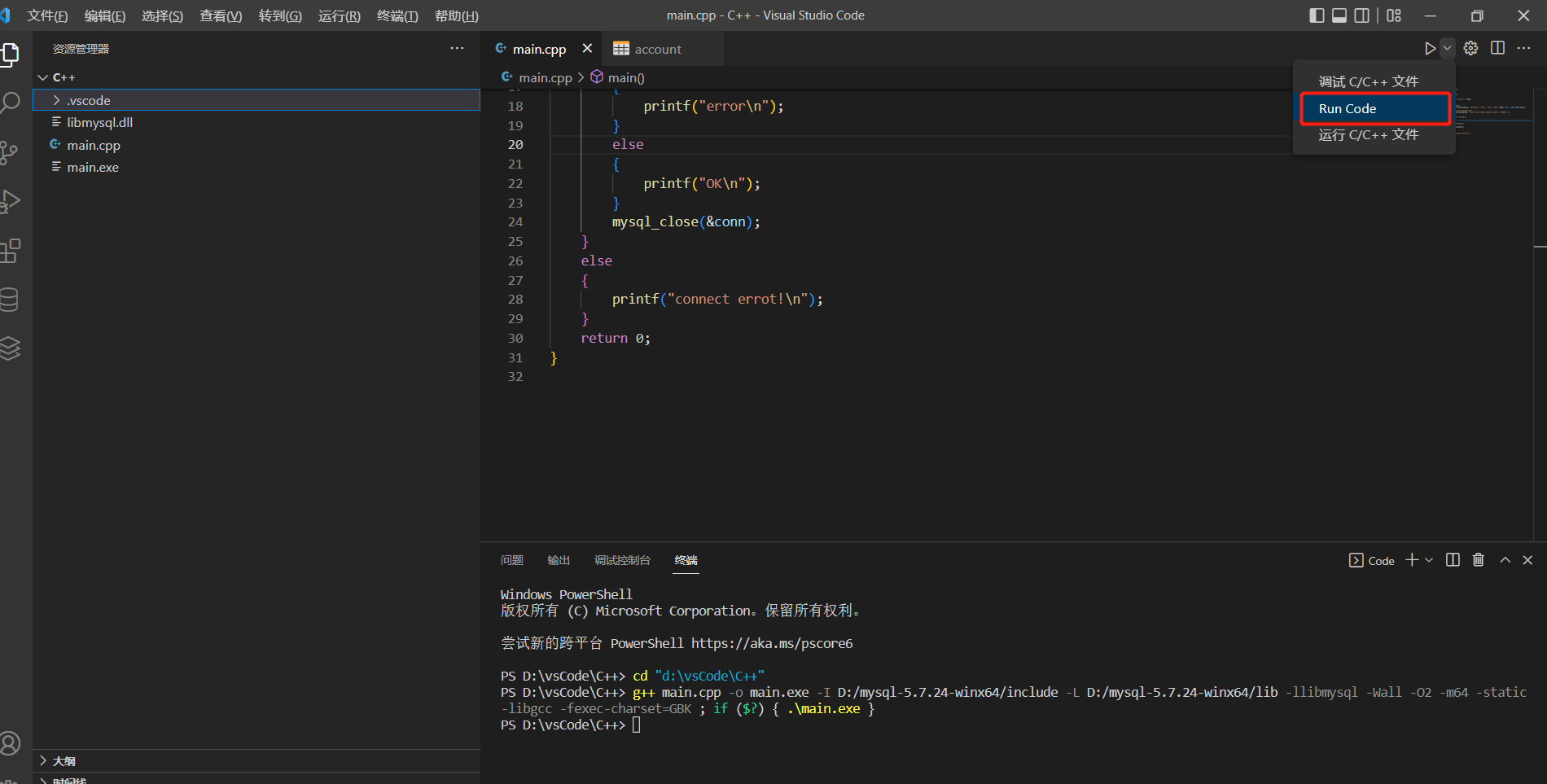
    return 0;

}

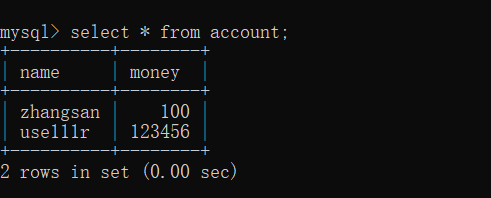
1-8）将libmysql.dll文件复制到当前目录下



1-9）运行main.cpp 代码

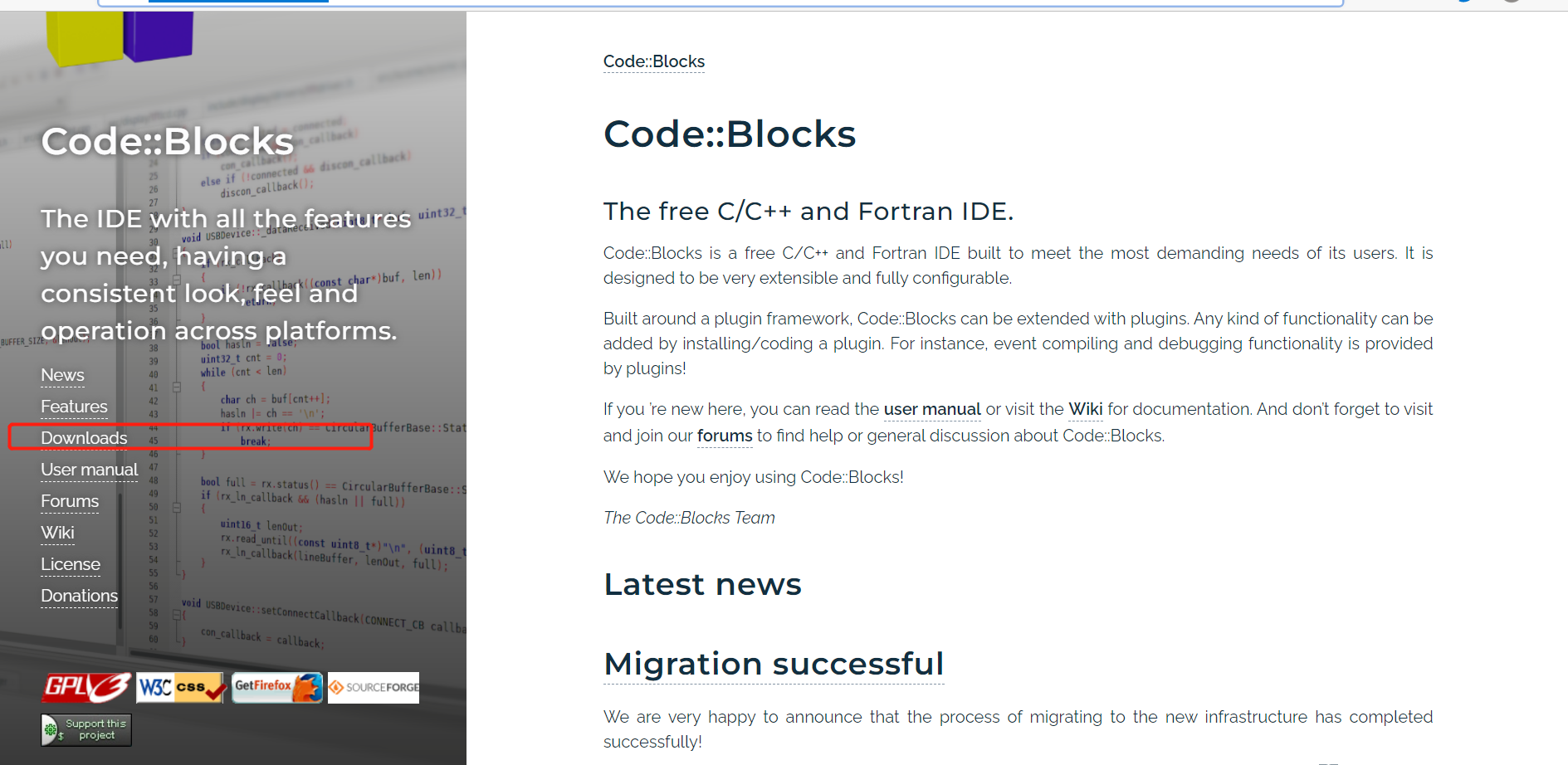


1-10）查看数据库 运行结果

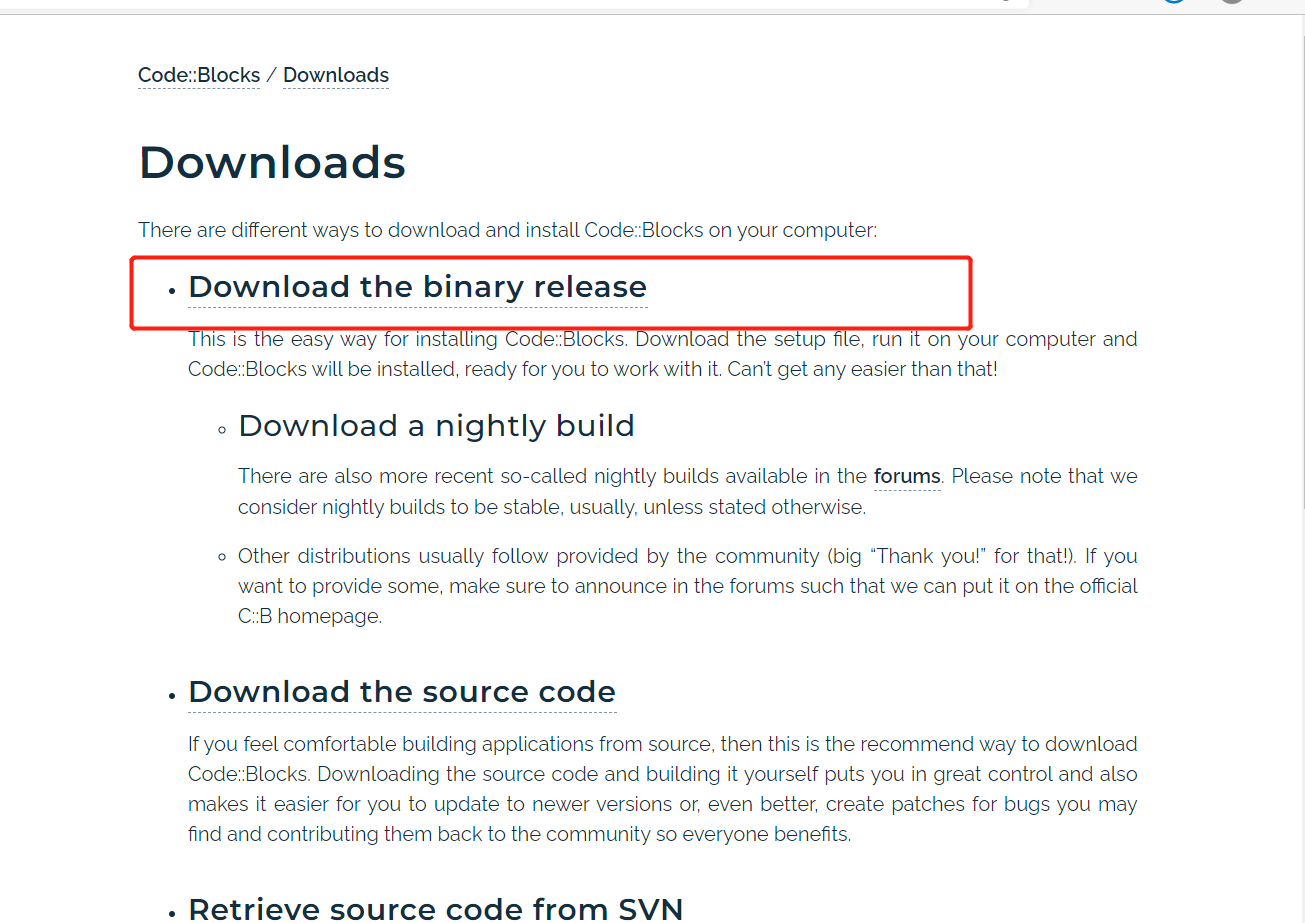


2）codeblock 安装

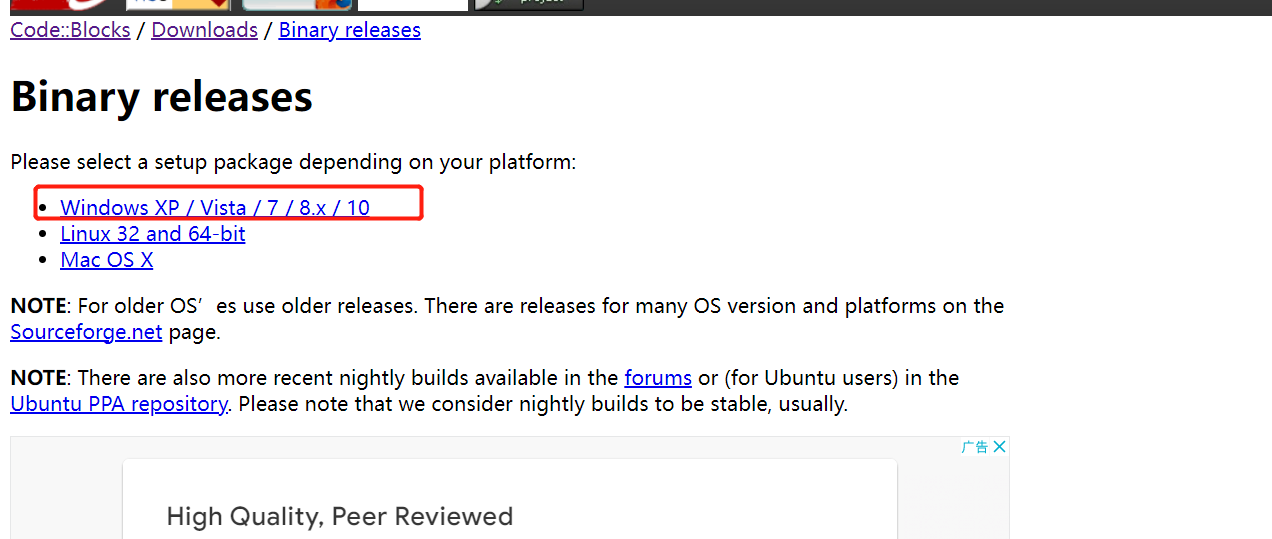
2-1）官网 <https://www.codeblocks.org/>

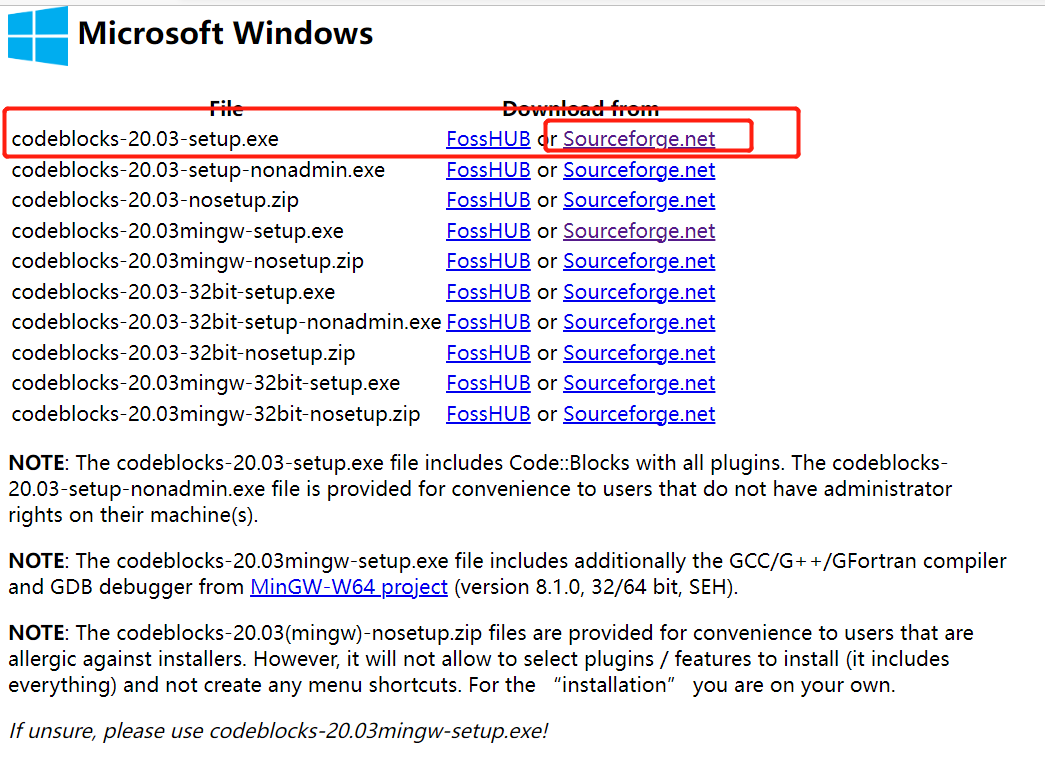


2-2）点击下载

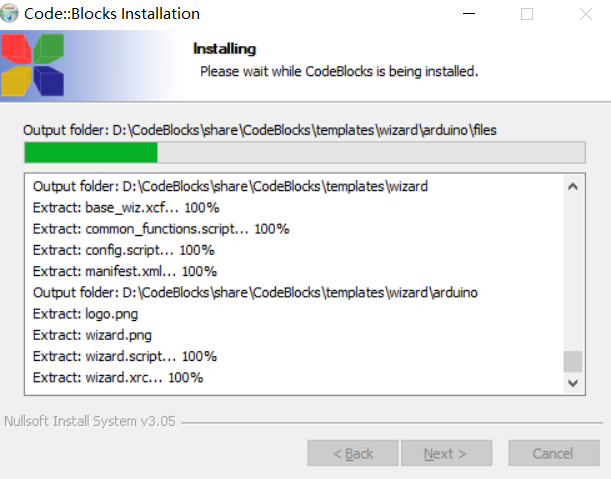


2-3）选择对应平台

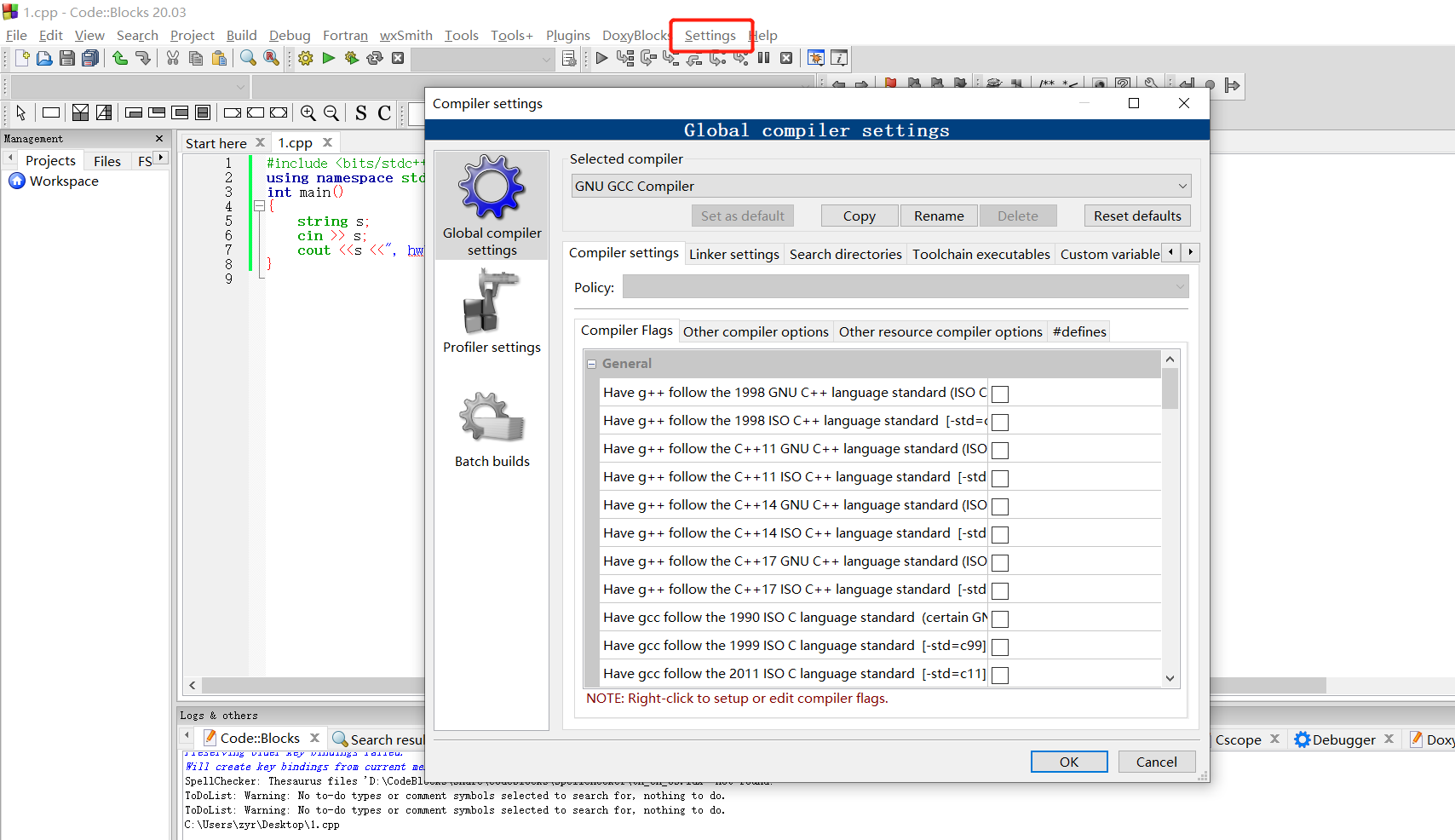




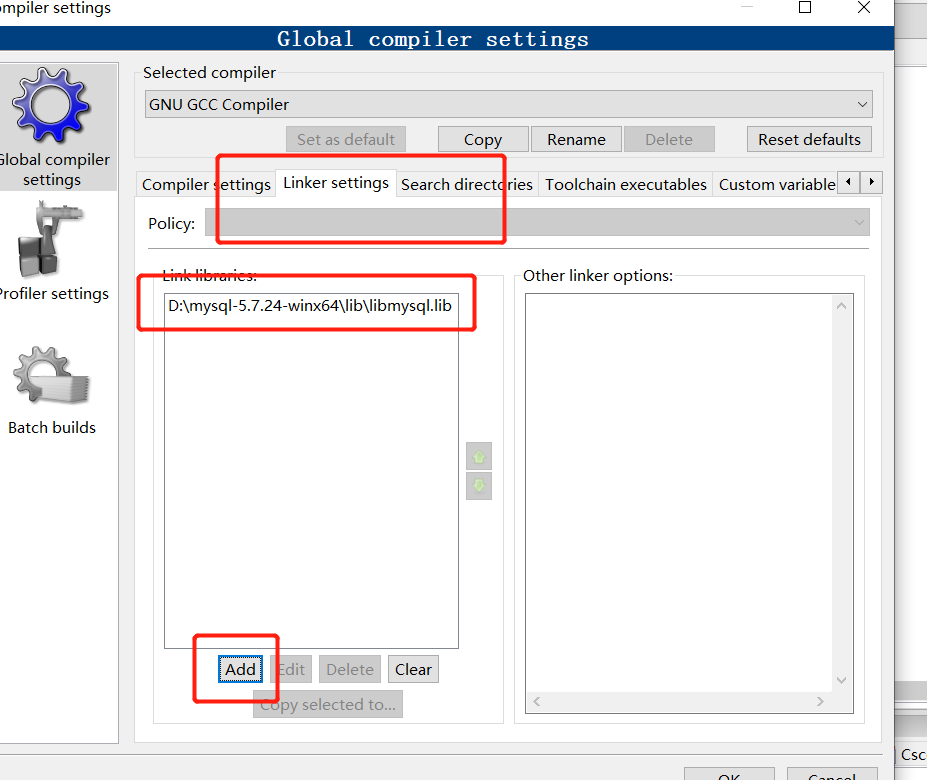
2-4）安装即可

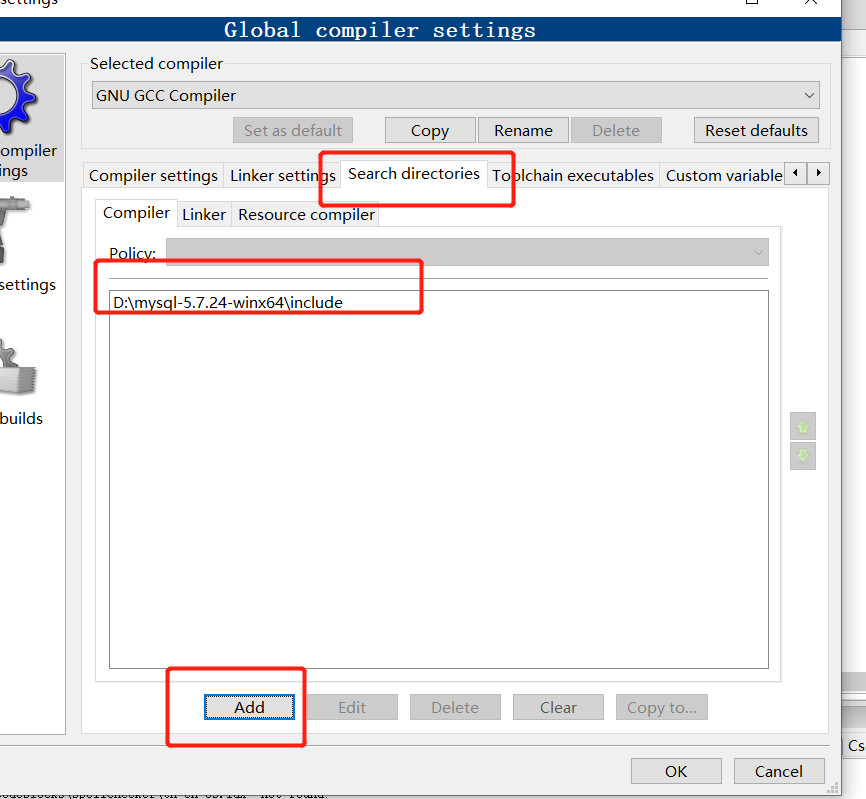


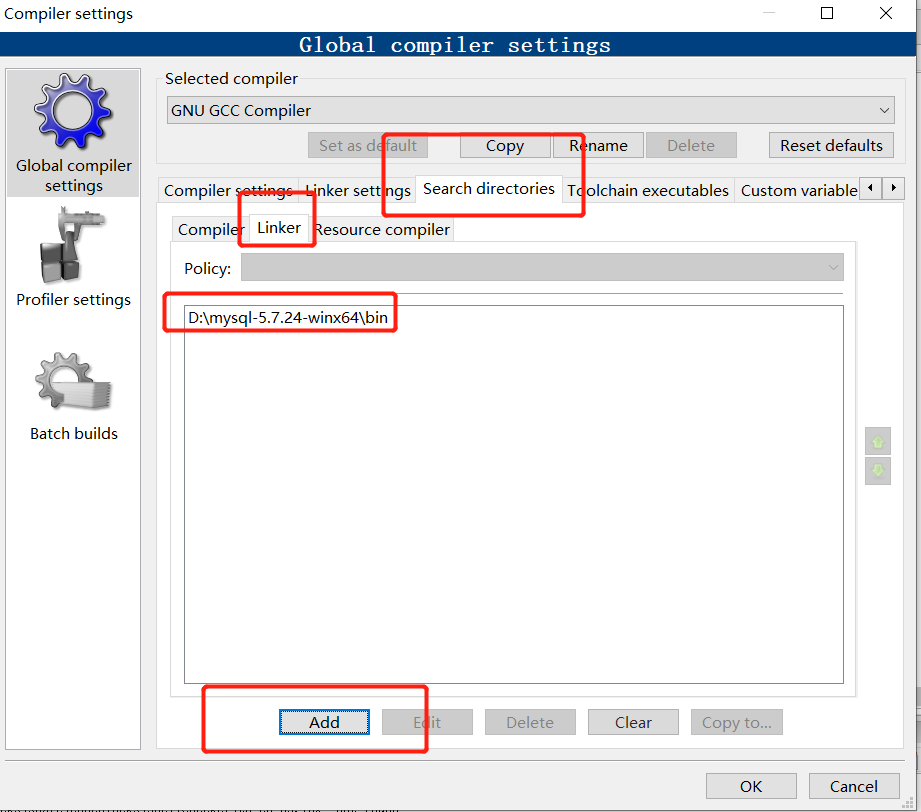
2-5）配置依赖库和include库



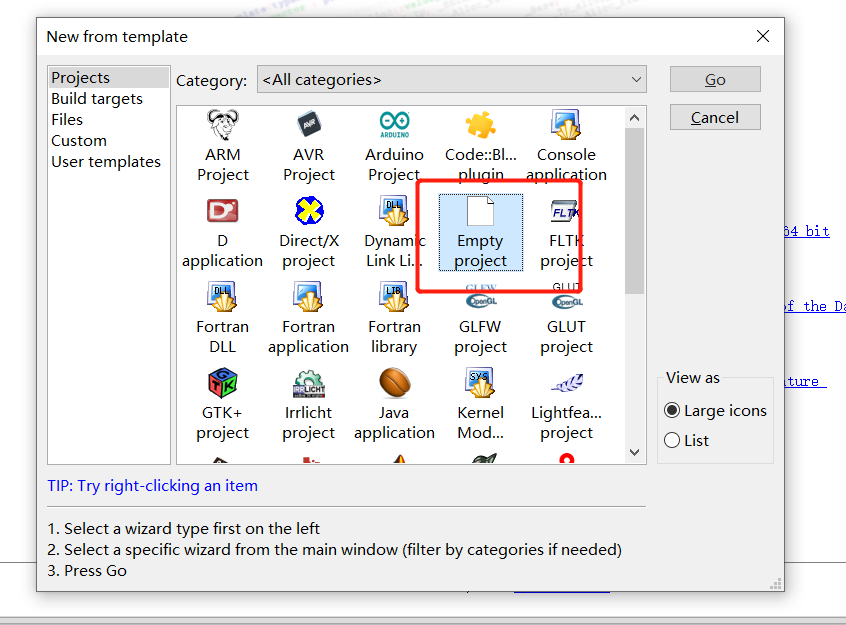
Setting中第三个Compiler

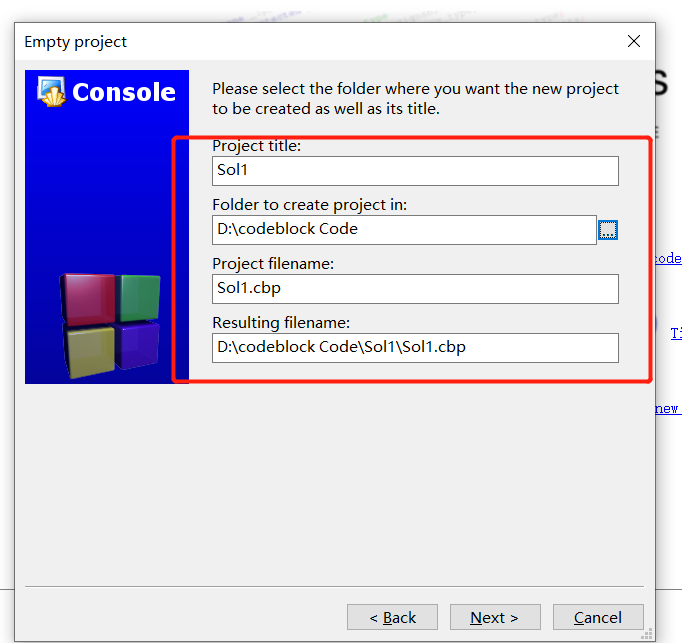




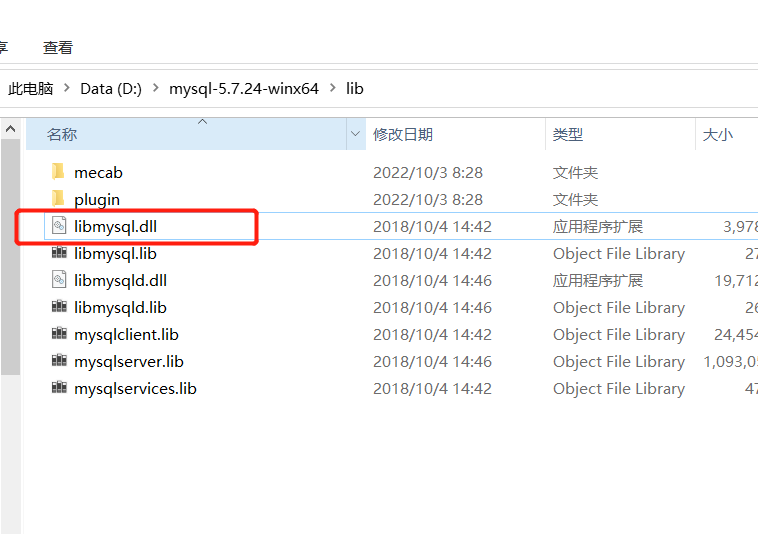


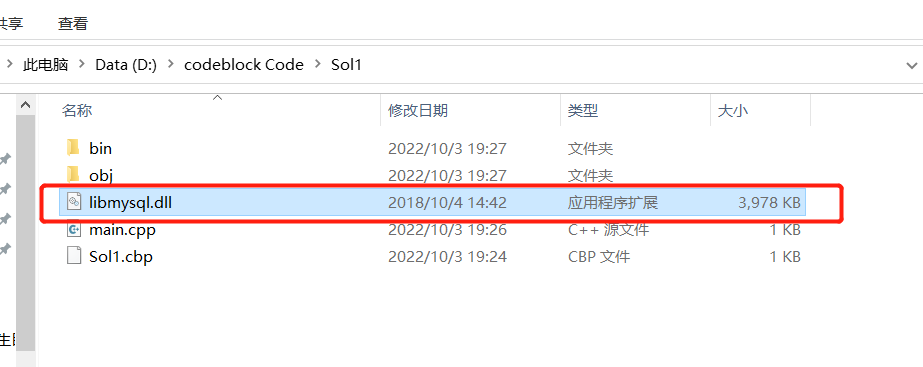
2-6）创建工程



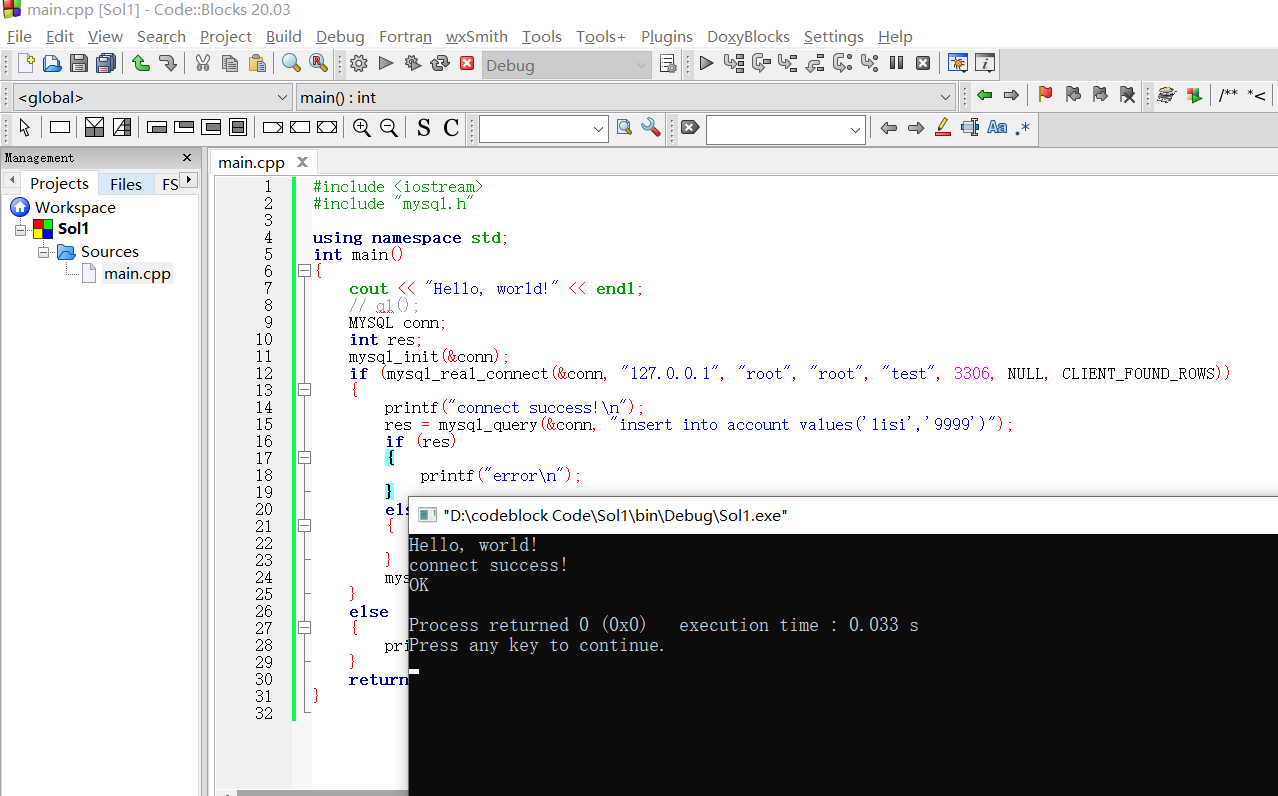


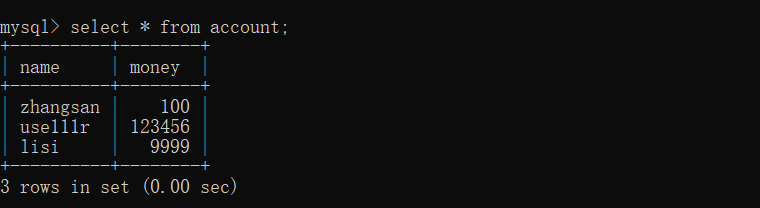
2-7）将libmysql.dll复制到工程中





2-8）运行结果





文件：main.cpp

#include <iostream>

#include "mysql.h"

using namespace std;

int main()

{

cout << "Hello, world!" << endl;

// q1();

MYSQL conn;

int res;

mysql\_init(&conn);

if (mysql\_real\_connect(&conn, "127.0.0.1", "root", "root", "test", 3306, NULL, CLIENT\_FOUND\_ROWS))

{

printf("connect success!\n");

res = mysql\_query(&conn, "insert into account values('lisi','9999')");

if (res)

{

printf("error\n");

}

else

{

printf("OK\n");

}

mysql\_close(&conn);

}

else

{

printf("connect errot!\n");

}

return 0;

}

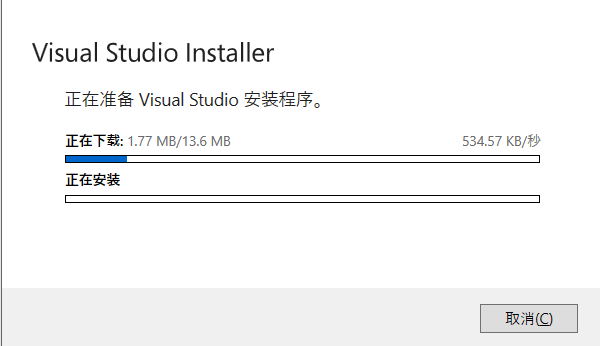
3）visual studio 安装

3-1）官网 <https://visualstudio.microsoft.com/zh-hans/free-developer-offers/>



社区版足够了。

3-2）安装



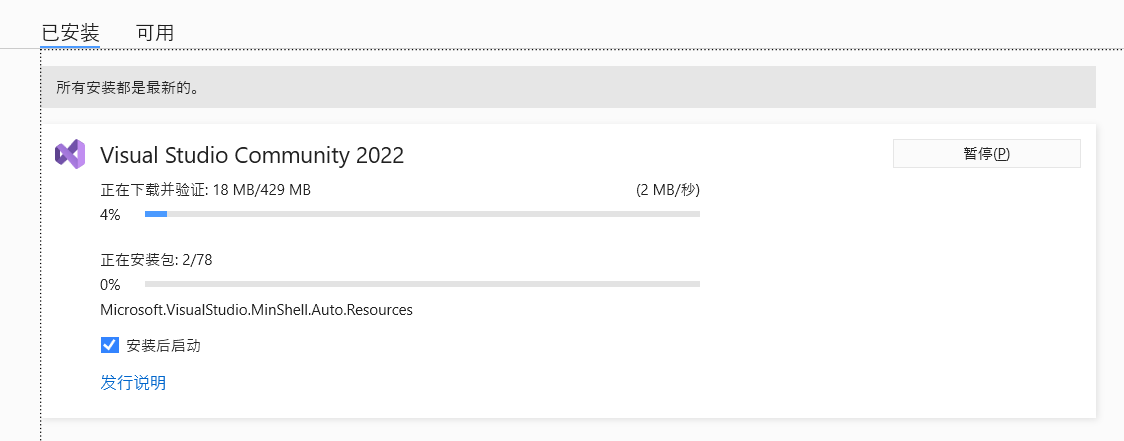
3-3）修改安装位置



3-4）增加组件



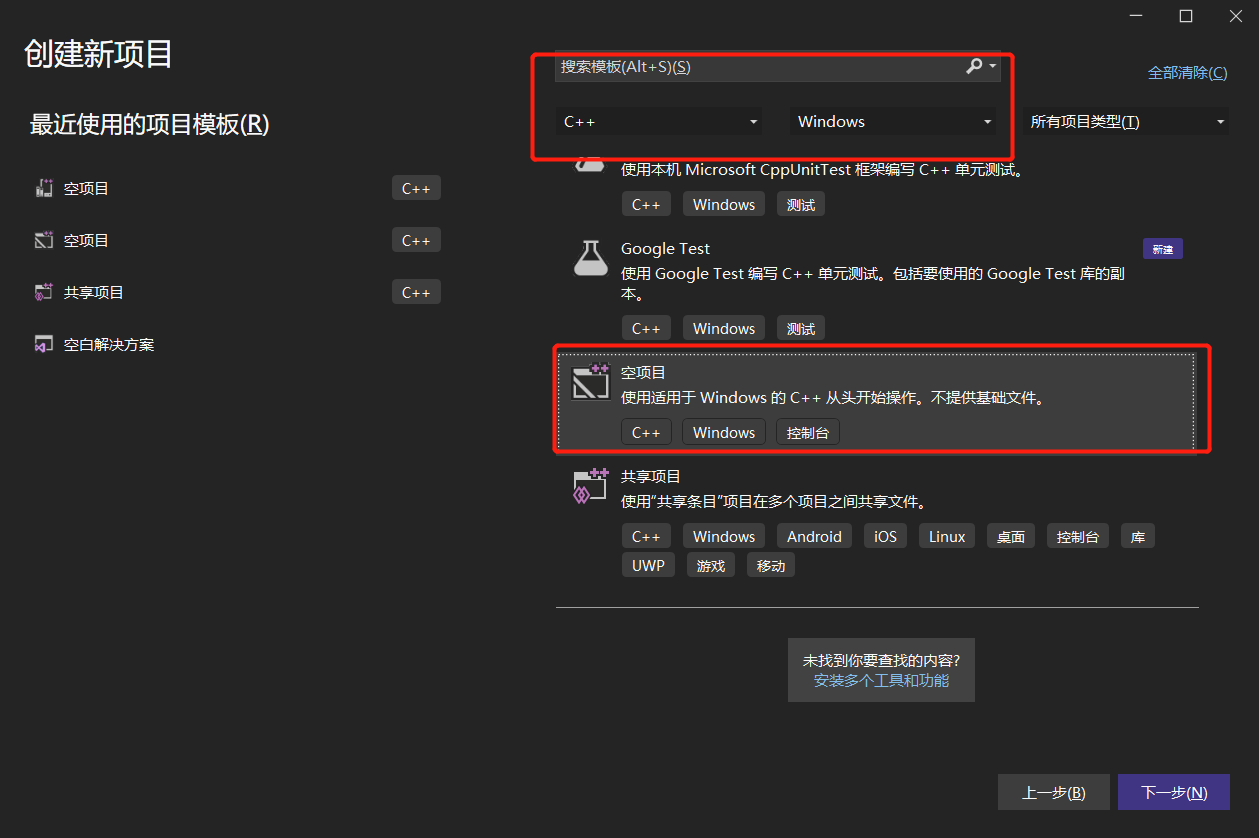
3-5）开始安装



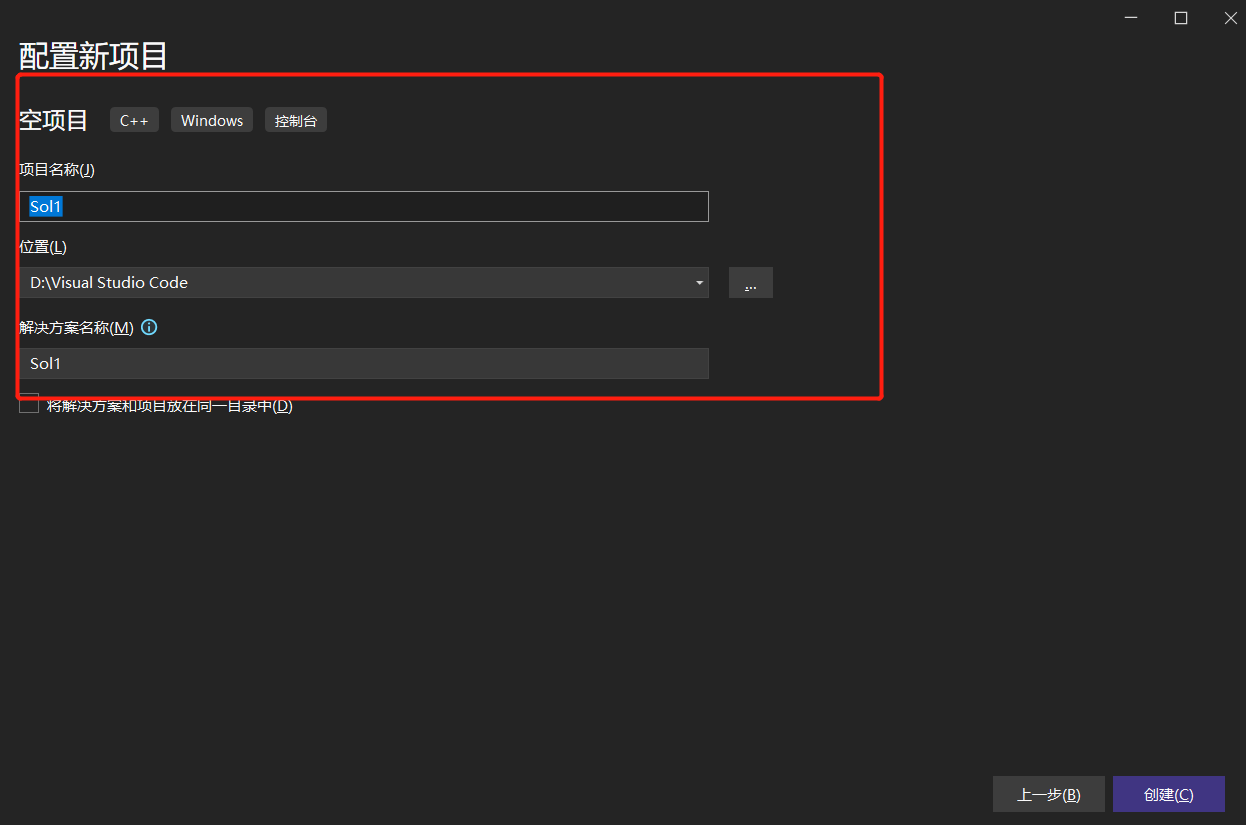
3-6）创建新的环境



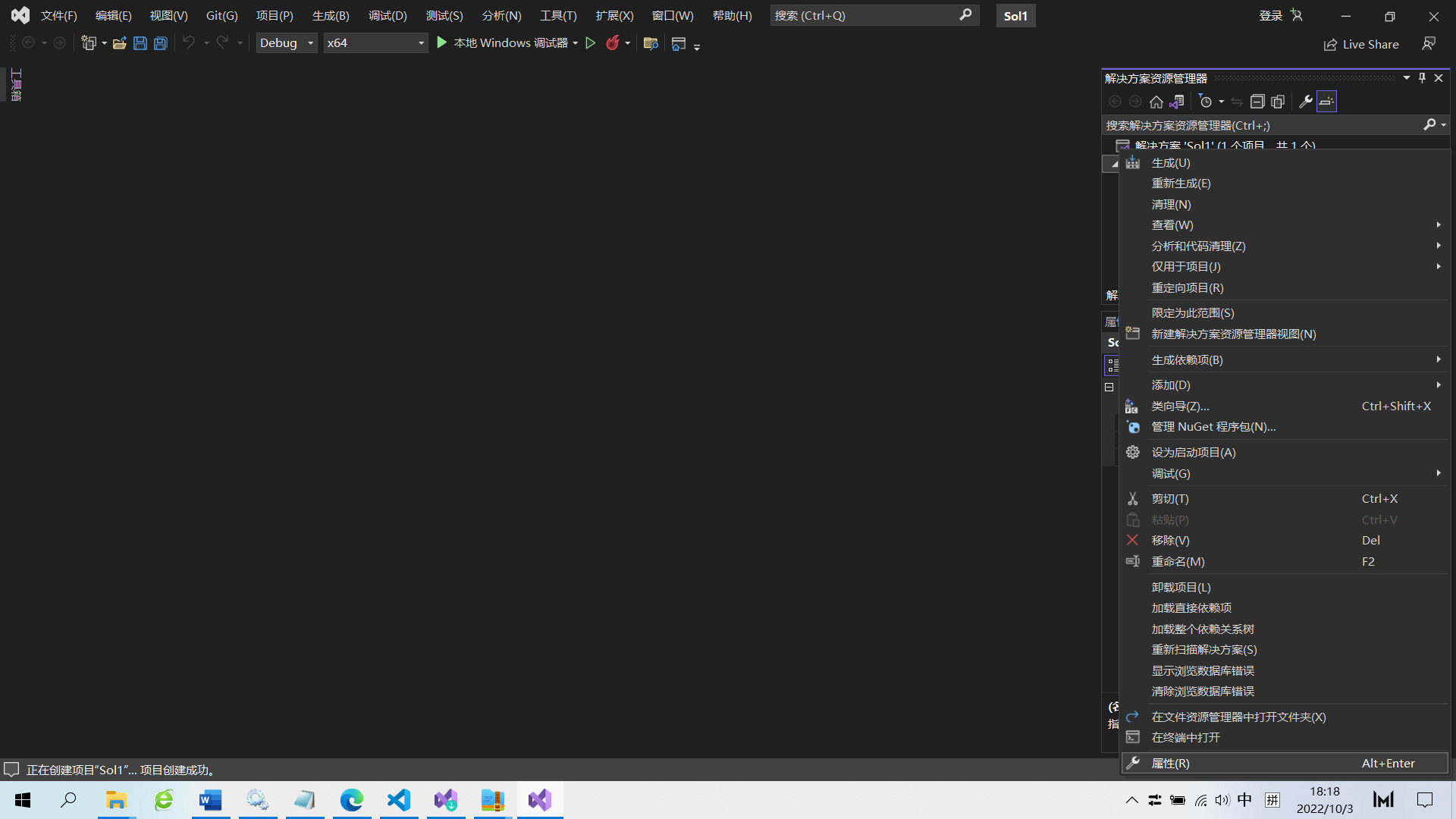
3-7）选择C++

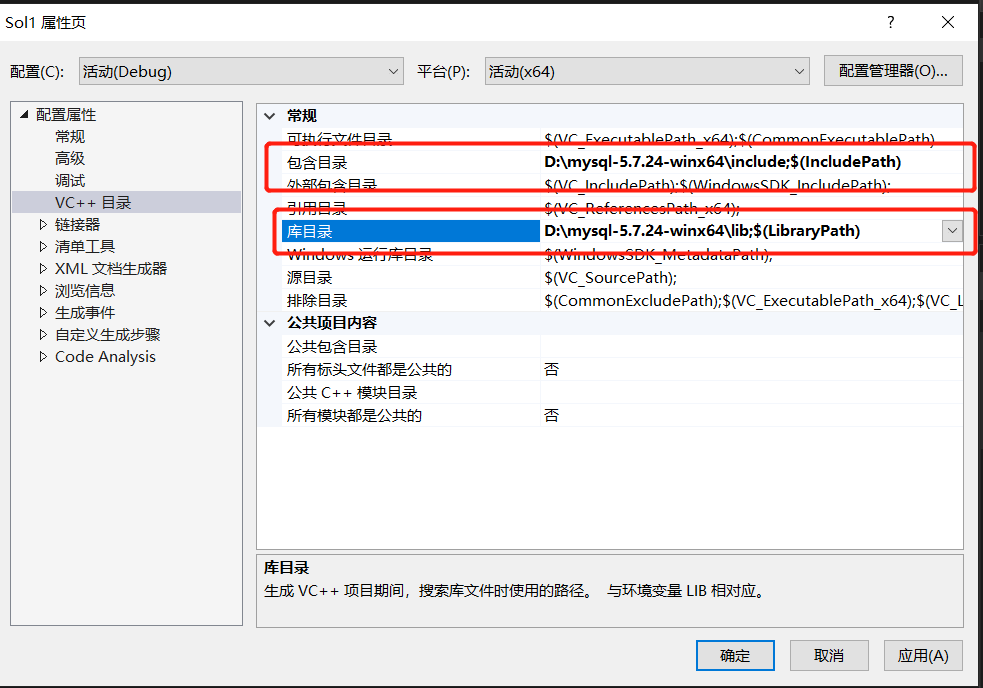


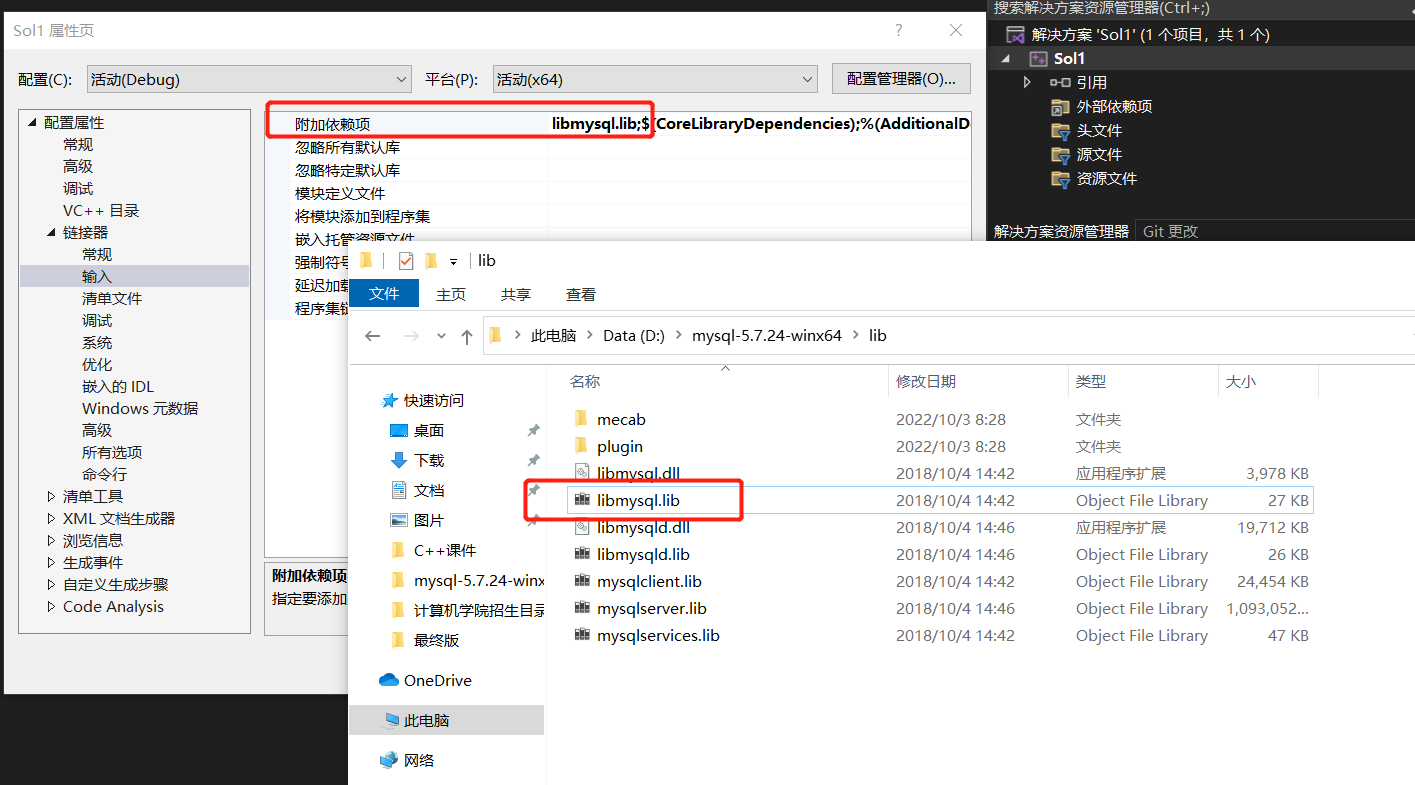
3-8）修改位置



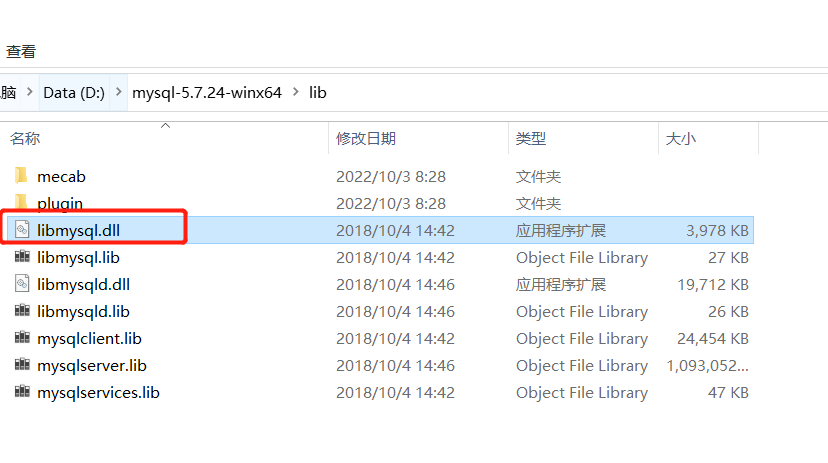
3-9）对项目修改属性，添加Mysql依赖和Include包

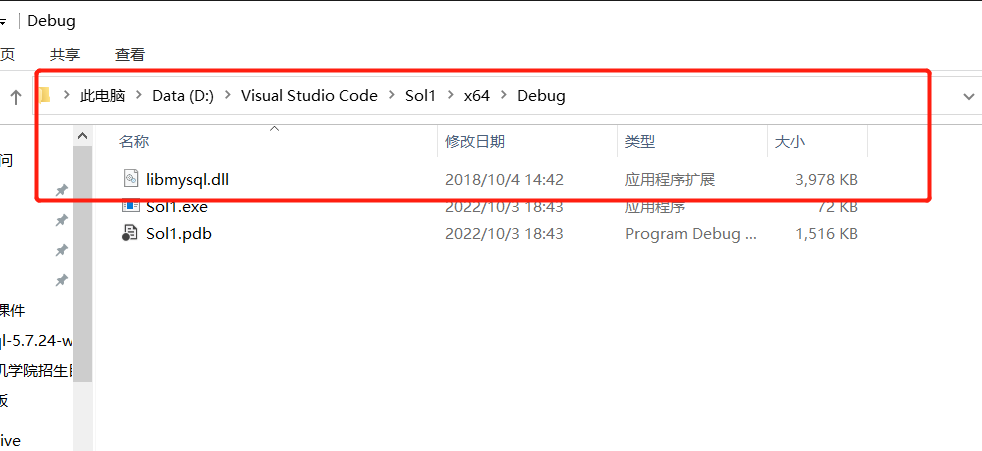






3-10）将下属文件复制到项目中





3-11）写代码

文件名：mysql.h

#pragma once

#ifndef MY\_SQL\_H

#define MY\_SQL\_H

#include<iostream>

#include<Windows.h>

#include<WinSock.h>

#include<mysql.h>

class DataBase

{

public:

DataBase();

~DataBase();

//连接数据库 参数为ip 用户名 密码 数据库名 端口

bool Connect(const char\* ip, const char\* name, const char\* cypher, const char\* database\_name, const int port);

//获取表内的字段数

int GetTableField(const char\* table\_name);

//查询表 参数为表名

bool Query(const char\* table\_name);

//自由执行指令

bool Implement(const char\* sentence);

private:

bool \_state;//连接状态 true为已连接

MYSQL\* \_mysql;//mysql连接

MYSQL\_FIELD\* \_fd;//字段列数组

char \_field[32][32];//存字段名二维数组

MYSQL\_RES\* \_res;//这个结构代表返回行的一个查询结果集

MYSQL\_ROW \_column;//一个行数据的类型安全(type-safe)的表示，表示数据行的列

char \_query[150];//查询语句

};

#endif // !MY\_SQL\_H

文件名：mysql.cpp

#include "mysql.h"

DataBase::DataBase()

{

\_state = false;

\_mysql = new MYSQL;

\_fd = nullptr;

memset(\_field, NULL, sizeof(\_field));

\_res = nullptr;

\_column = nullptr;

memset(\_query, NULL, sizeof(\_query));

}

DataBase::~DataBase()

{

}

bool DataBase::Connect(const char\* ip, const char\* name, const char\* cypher, const char\* database\_name, const int port)

{

if (true == \_state)

{

printf("Database connected\n");

return false;

}

//初始化mysql

mysql\_init(\_mysql);

//返回false则连接失败，返回true则连接成功

if (!(mysql\_real\_connect(\_mysql, ip, name, cypher, database\_name, port, NULL, 0))) //中间分别是主机，用户名，密码，数据库名，端口号（可以写默认0或者3306等），可以先写成参数再传进去

{

printf("Error connecting to database:%s\n", mysql\_error(\_mysql));

return false;

}

else

{

\_state = true;

printf("Connected succeed\n\n");

return true;

}

return true;

}

int DataBase::GetTableField(const char\* table\_name)

{

if (false == \_state)

{

printf("Database not connected\n");

return -1;

}

//查询内容

sprintf\_s(\_query, "desc %s", table\_name); //desc 语句获取字段数

//设置编码格式（SET NAMES GBK也行），否则cmd下中文乱码

mysql\_query(\_mysql, "set names gbk");

//返回0 查询成功，返回1查询失败

if (mysql\_query(\_mysql, \_query)) //执行SQL语句

{

printf("Query failed (%s)\n", mysql\_error(\_mysql));

return false;

}

//获取结果集

if (!(\_res = mysql\_store\_result(\_mysql))) //获得sql语句结束后返回的结果集

{

printf("Couldn't get result from %s\n", mysql\_error(\_mysql));

return false;

}

//数据行数即为字段个数

return mysql\_affected\_rows(\_mysql);

}

bool DataBase::Query(const char\* table\_name)

{

if (false == \_state)

{

printf("Database not connected\n");

return false;

}

//获取字段数

int field = GetTableField(table\_name);

//查询内容

sprintf\_s(\_query, "select \* from %s", table\_name); //执行查询语句

//设置编码格式（SET NAMES GBK也行），否则cmd下中文乱码

mysql\_query(\_mysql, "set names gbk");

//返回0 查询成功，返回1查询失败

if (mysql\_query(\_mysql, \_query)) //执行SQL语句

{

printf("Query failed (%s)\n", mysql\_error(\_mysql));

return false;

}

else

{

printf("query success\n");

}

//获取结果集

if (!(\_res = mysql\_store\_result(\_mysql))) //获得sql语句结束后返回的结果集

{

printf("Couldn't get result from %s\n", mysql\_error(\_mysql));

return false;

}

//打印数据行数

printf("number of dataline returned: %lld\n", mysql\_affected\_rows(\_mysql));

//获取字段的信息

char\* str\_field[32]; //定义一个字符串数组存储字段信息

for (int i = 0; i < field; i++) //在已知字段数量的情况下获取字段名

{

str\_field[i] = mysql\_fetch\_field(\_res)->name;

}

for (int i = 0; i < field; i++) //打印字段

{

printf("%10s\t", str\_field[i]);

}

printf("\n");

//打印获取的数据

while (\_column = mysql\_fetch\_row(\_res)) //在已知字段数量情况下，获取并打印下一行

{

for (int i = 0; i < field; i++)

{

printf("%10s\t", \_column[i]); //column是列数组

}

printf("\n");

}

return true;

}

bool DataBase::Implement(const char\* sentence)

{

if (false == \_state)

{

printf("Database not connected\n");

return false;

}

//查询内容

sprintf\_s(\_query, "%s", sentence); //desc 语句获取字段数

//设置编码格式（SET NAMES GBK也行），否则cmd下中文乱码

mysql\_query(\_mysql, "set names gbk");

//执行SQL语句

if (mysql\_query(\_mysql, \_query))

{

printf("Query failed (%s)\n", mysql\_error(\_mysql));

return false;

}

return true;

}

文件名：main.cpp

#include"mysql.h"

int main()

{

DataBase\* d1 = new DataBase;

//连接

d1->Connect("localhost", "root", "root", "test", 3306);

printf("\n");

//查询表

d1->Query("account");

printf("\n");

//添加内容

d1->Implement("insert into account values('张三', 100)");

printf("\n");

//查询表

d1->Query("account");

printf("\n");

return 0;

}

3-12）执行结果

