项目说明文档——考试报名系统

姓名：吴桐欣

学号：1652677

同济大学 软件学院 软件工程专业

目录

项目说明文档——考试报名系统 1

目录 2

1.项目概述 3

1.1项目简介 3

1.2文件目录 3

1.3操作指南 3

1.4注意事项 3

2.思路与设计 3

2.1基本思路 3

2.2设计 4

3.具体实现 4

3.1添加考生信息 4

3.2输入考生信息 4

3.3输出考生信息 4

3.4查询考生信息 4

3.5修改考生信息 4

3.6删除考生信息 4

4.测试 4

4.1功能测试 4

4.2出错测试 4

1.项目概述

1.1项目简介

考试报名工作给各高校报名工作带来了新的挑战，给教务管理部门增加了很大的工作量。本项目是对考试报名管理的简单模拟，用控制台选项的选择方式完成多项功能。

**1.1.1功能分析**

本项目要求实现功能

1. 输入考生信息；
2. 输出考生信息；
3. 查询考生信息；
4. 添加考生信息；
5. 修改考生信息；
6. 删除考生信息。

1.2文件目录

（1）P01\_1652677\_吴桐欣\_说明文档.docx（本文档）

（2）P01\_1652677\_吴桐欣.exe（可执行文件）

（3）P01\_1652677\_吴桐欣.cpp（源文件）

（4）P01\_1652677\_吴桐欣.h（头文件）

（5）输入样例.txt（测试数据）

1.3操作指南

（1）运行程序后，将获得程序提示“请输入考生人数：”

用户输入一个数字表示考生人数。

（2）随后程序提示“请依次输入考生的考号，姓名，性别，年龄及报考类别：”

用户需要按照要求依次输入信息。

（3）程序提示“操作数表(1-插入；2-删除；3-查找；4-修改；5-统计；0-取消)

请选择您要进行的操作：”

用户根据自身需要输入操作数。

[输入样例]

1 甲 女 32 设计

2 乙 男 25 开发

3 丙 男 30 测试

4 丁 女 19 测试

1.4注意事项

（1）用户需按要求输入考生信息，信息按顺序一一对应

（2）输入人数、考号、年龄、操作数这类数字信息时，不能输入除数字以外的字符

2.思路与设计

2.1基本思路

利用链表对考生信息进行存储，并用一个信息系统的类，用其成员函数来实现各项功能。

2.2设计

**2.2.1数据结构**

用双向链表结构。一位考生的信息用一个类封装起来，并带有分别指向上一位考生和下一位考生的指针。还有一个是信息系统的类，设为考生类的友元类，封装实现功能的函数。

**2.2.2成员与成员函数**

class student {

friend class system; //声明友元类

private:

int \_id;

string \_name;

string \_gender;

int \_age;

string \_type;

student\* \_last;

student\* \_next;

public:

//两个构造函数

student(student\* last=NULL, student\* next=NULL){\_last=last, \_next=next;};

student(const int& ID, const string& name, const string& gender, const int& age, const string& type, student\* last=NULL, student\* next=NULL) {

\_id=ID;

\_name=name;

\_gender=gender;

\_age=age;

\_type=type;

\_last=last;

\_next=next;

}

friend istream& operator>>(istream& is, student& rhs);//重载输入符号

friend ostream& operator<<(ostream& os, const student& rhs);//重载输出符号

};

student内数据包括考号，姓名，性别，年龄，报考类别，以及两个指向前、后一位考生的student指针。构造函数有两个，一个仅初始化指针，一个初始化所有数据。为了方便，重载了输入与输出符号。

class system {

private:

student\* \_data;

public:

//构造函数

system(){\_data = new student;}

//析构函数

~system(){

student\* toDelete=NULL;

while(toDelete!=NULL){

toDelete = \_data->\_next;

\_data->\_next = toDelete->\_next;

delete toDelete;

}

}

void insertStu(int pos, student\* newStu);//插入考生

void deleteStu(student\* stu);//删除考生

student\* searchStu(int ID);//根据考号查找考生

student\* findPos(int pos);//根据位置查找将插入的考生的位置

void reset(int item, student\* stu);//修改信息

void display();//输出所有信息，即统计

};

3.具体实现

3.1添加考生信息

核心代码

student\* newStu = new student;

cout<<"请依次输入要插入的考生的考号，姓名，性别，年龄及报考类别："<<endl;

cin>>\*newStu;

testSys->insertStu(pos, newStu);

insertStu(int pos, student\* newStu)函数代码

void system::insertStu(int pos, student\* newStu){

student\* pre = findPos(pos-1);

if (pre) {

newStu->\_next = pre->\_next;

pre->\_next = newStu;

newStu->\_last=pre;

if (newStu->\_next) {

newStu->\_next->\_last=newStu;

}

} else {

cout<<"该数字超过当前考生人数，无法完成操作"<<endl;

}

}

findPos(int pos)函数代码

student\* system::findPos(int pos){

student\* cur = \_data;

for (int i=0; i<pos; i++) {

if (cur->\_next) {

cur = cur->\_next;

} else {

return NULL;

}

}

return cur;

}

说明

从链表表头开始用for循环，依据用户所给的pos信息确定循环次数，直到找到该位置，若未找到，则无法完成操作。找到位置后，修改插入位置前后的考生所拥有的指针，以及插入的考生的指针，完成插入。

3.2输入考生信息

核心代码

说明

cout<<"请依次输入考生的考号，姓名，性别，年龄及报考类别："<<endl;

for (int i=0; i<n; i++) {

student\* newStu = new student;

cin>>\*newStu;

testSys->insertStu(i+1, newStu);

}

直接利用insertStu函数，循环操作，一个个录入考生信息。

3.3输出考生信息

核心代码display()函数

void system::display(){

cout.width(10);

cout<<"考号";

cout.width(10);

cout<<"姓名";

cout.width(10);

cout<<"性别";

cout.width(10);

cout<<"年龄";

cout.width(10);

cout<<"报考类别"<<endl;

student\* current = \_data->\_next;

while (current) {

cout<<\*current;

current = current->\_next;

}

}

说明

用current指针指向链表中的元素，从表头开始走到表尾

3.4查询考生信息

核心代码

cout<<"请输入你要查找的考生的考号"<<endl;

int ID;

cin>>ID;

student\* find=testSys->searchStu(ID);

if (find) {

cout<<"以下是你所查找的考生信息："<<endl;

cout.width(10);

cout<<"考号";

cout.width(10);

cout<<"姓名";

cout.width(10);

cout<<"性别";

cout.width(10);

cout<<"年龄";

cout.width(10);

cout<<"报考类别"<<endl;

cout<<\*find;

} else {

cout<<"考生不存在！"<<endl;

}

searchStu(int ID)函数代码

student\* system::searchStu(int ID){

student\* current = \_data;

while(current) {

if (current->\_id == ID) {

return current;

break;

}

current = current->\_next;

}

return NULL;

}

说明

current指针指向链表中的元素，从链表表头开始，一一比较用户所给考号ID及每位考生的考号，当二者一致时，找到考生。若直到表尾都没找到，则不存在该考号的考生。

3.5修改考生信息

核心代码

void system::reset(int item, student\* stu){

switch (item) {

case 1:

{

int ID;

cout<<"请输入新的考号：";

cin>>ID;

stu->\_id = ID;

break;

}

case 2:

{

string name;

cout<<"请输入新的姓名：";

cin>>name;

stu->\_name = name;

break;

}

case 3:

{

string gender;

cout<<"请输入新的性别：";

cin>>gender;

stu->\_gender = gender;

break;

}

case 4:

{

int age;

cout<<"请输入新的年龄：";

cin>>age;

stu->\_age = age;

break;

}

case 5:

{

string type;

cout<<"请输入新的报考类别：";

cin>>type;

stu->\_type = type;

break;

}

default:

break;

}

}

说明

依据用户所指示的修改项目号item，执行操作。

3.6删除考生信息

核心代码

void system::deleteStu(student\* stu){

cout<<"你删除的考生的信息是："<<endl;

cout<<\*stu;

stu->\_next->\_last = stu->\_last;

stu->\_last->\_next = stu->\_next;

delete stu;

}

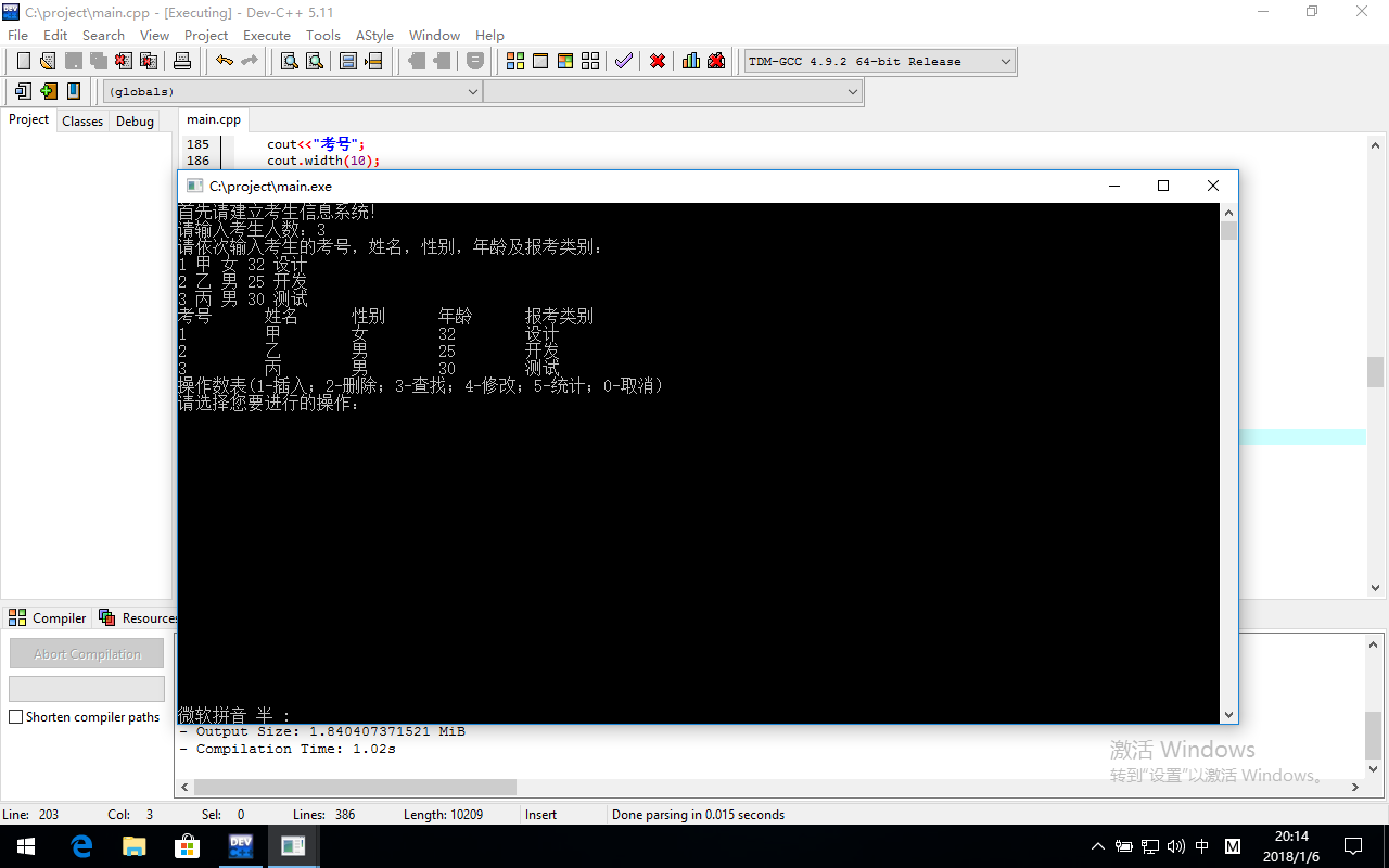
说明

修改被删除考生位置前后的考生所拥有的指针，并释放该考生所占的空间，完成删除。

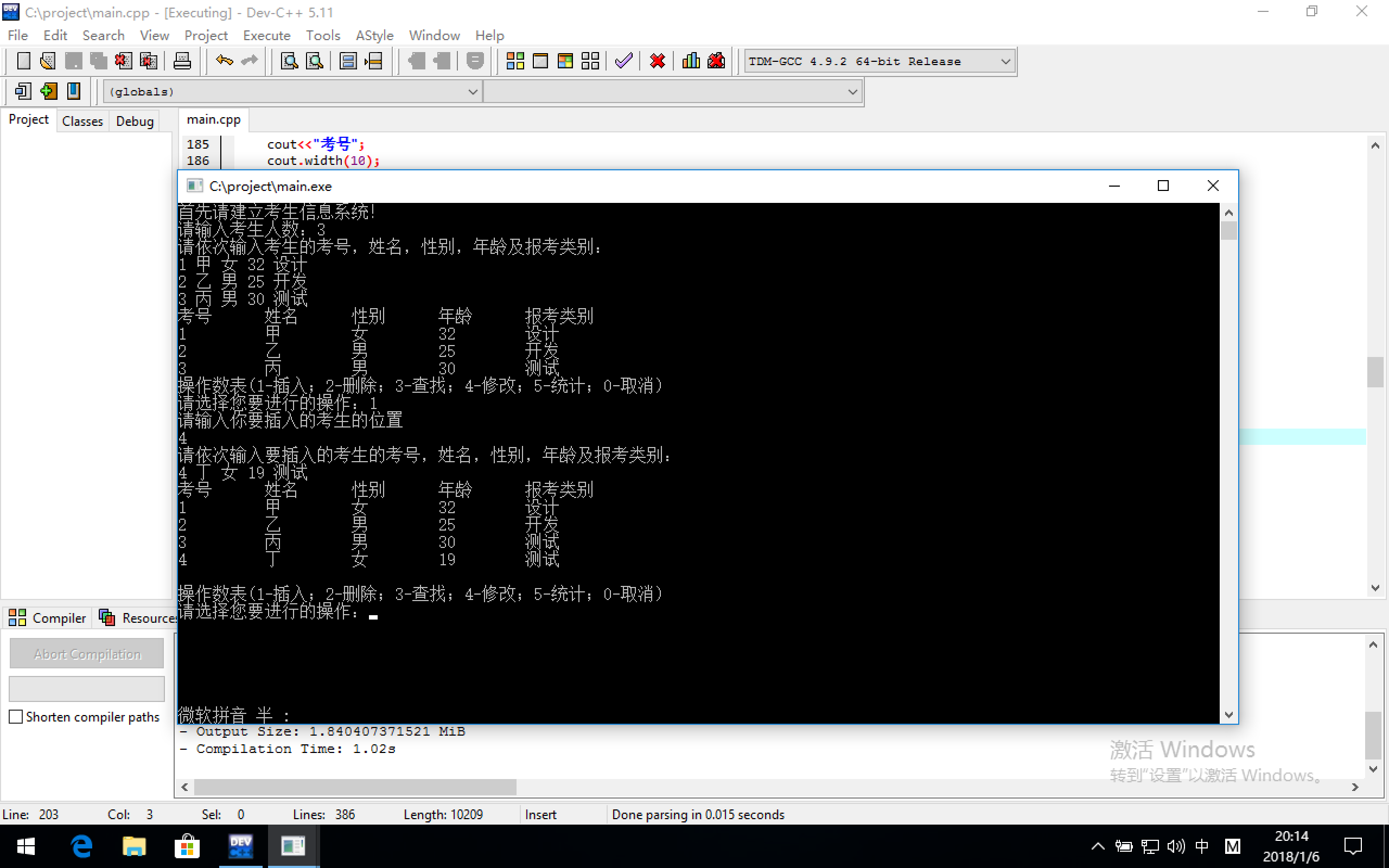
4.测试

4.1功能测试

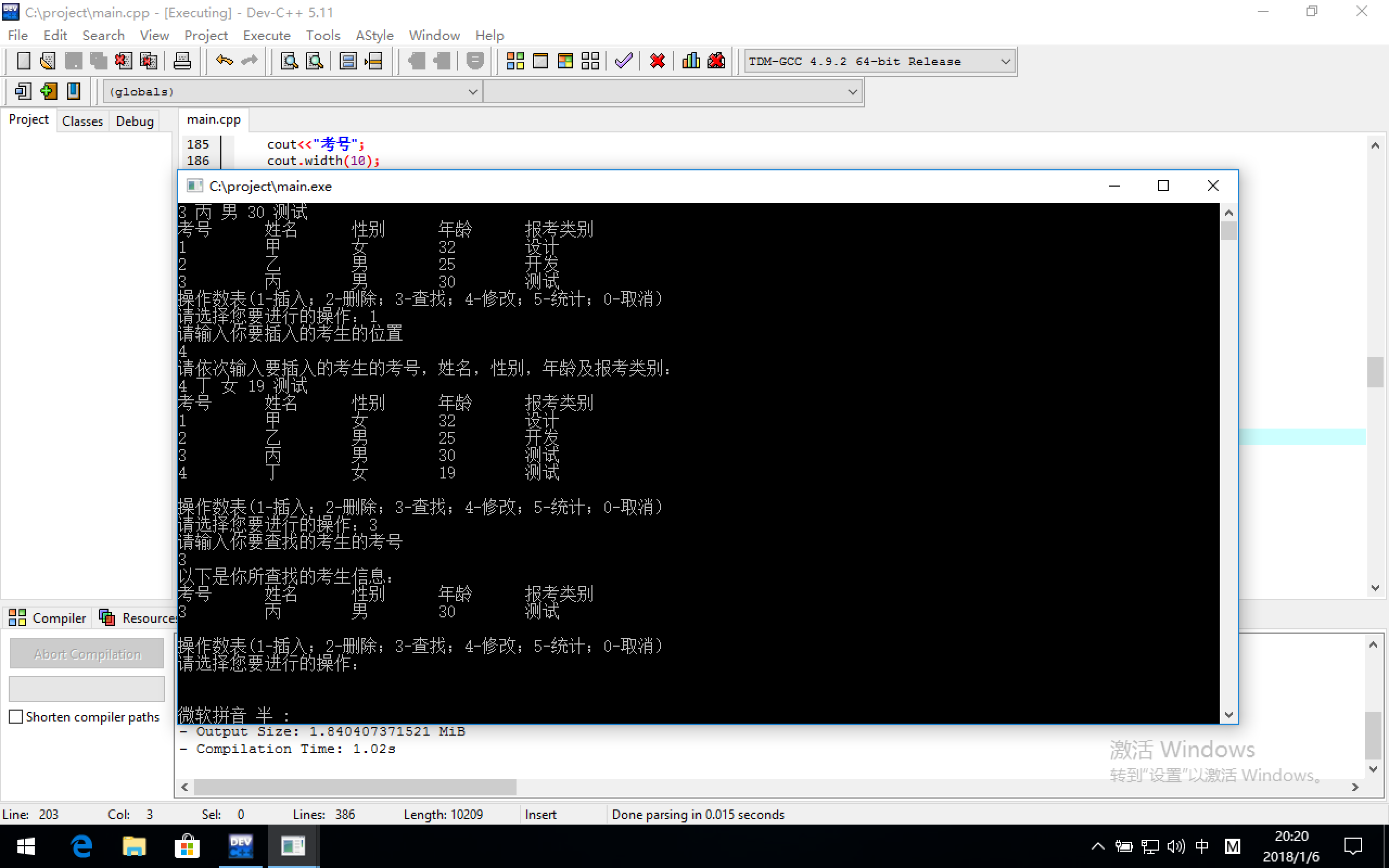
4.1.1输入考生信息



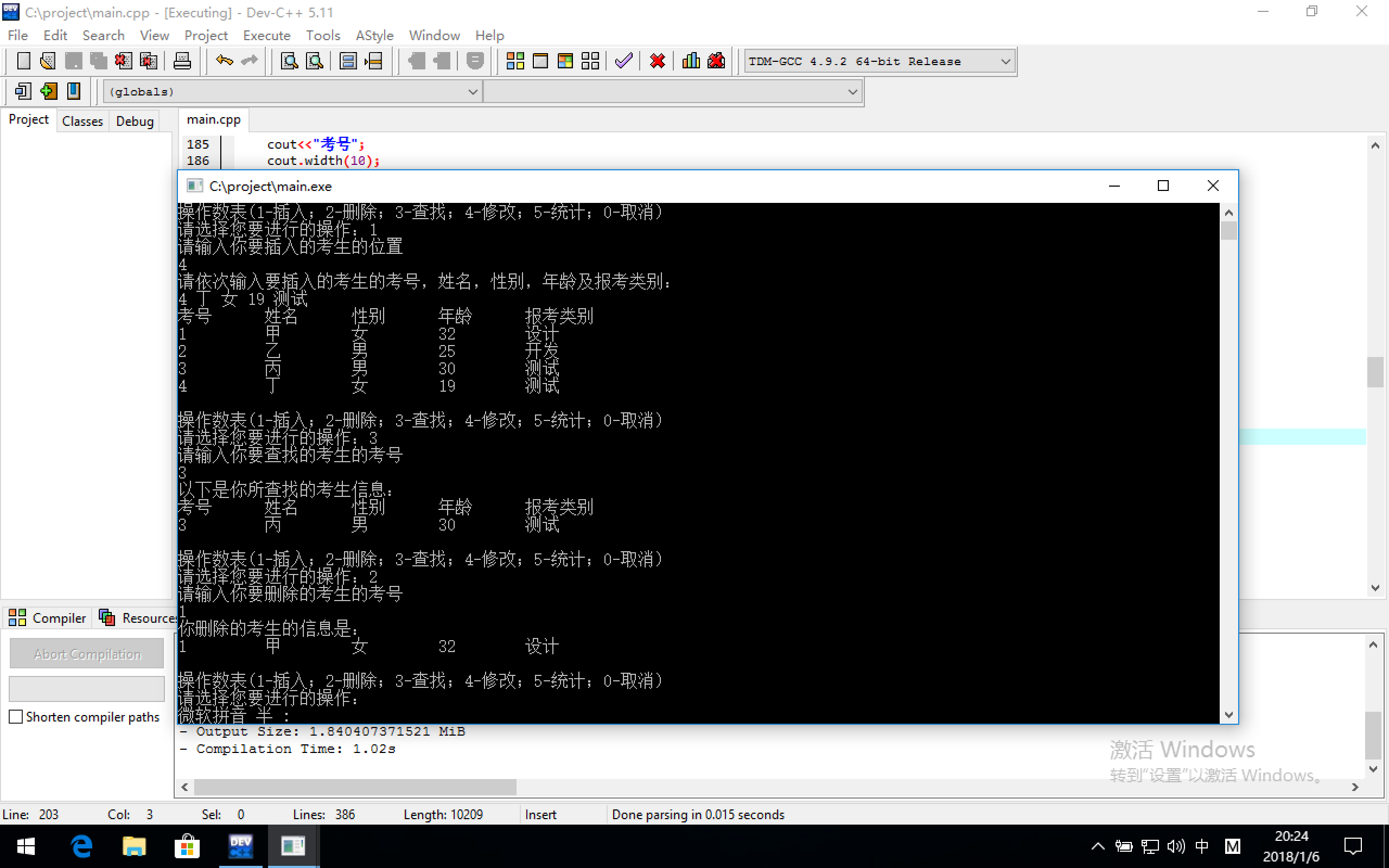
4.1.2添加考生信息



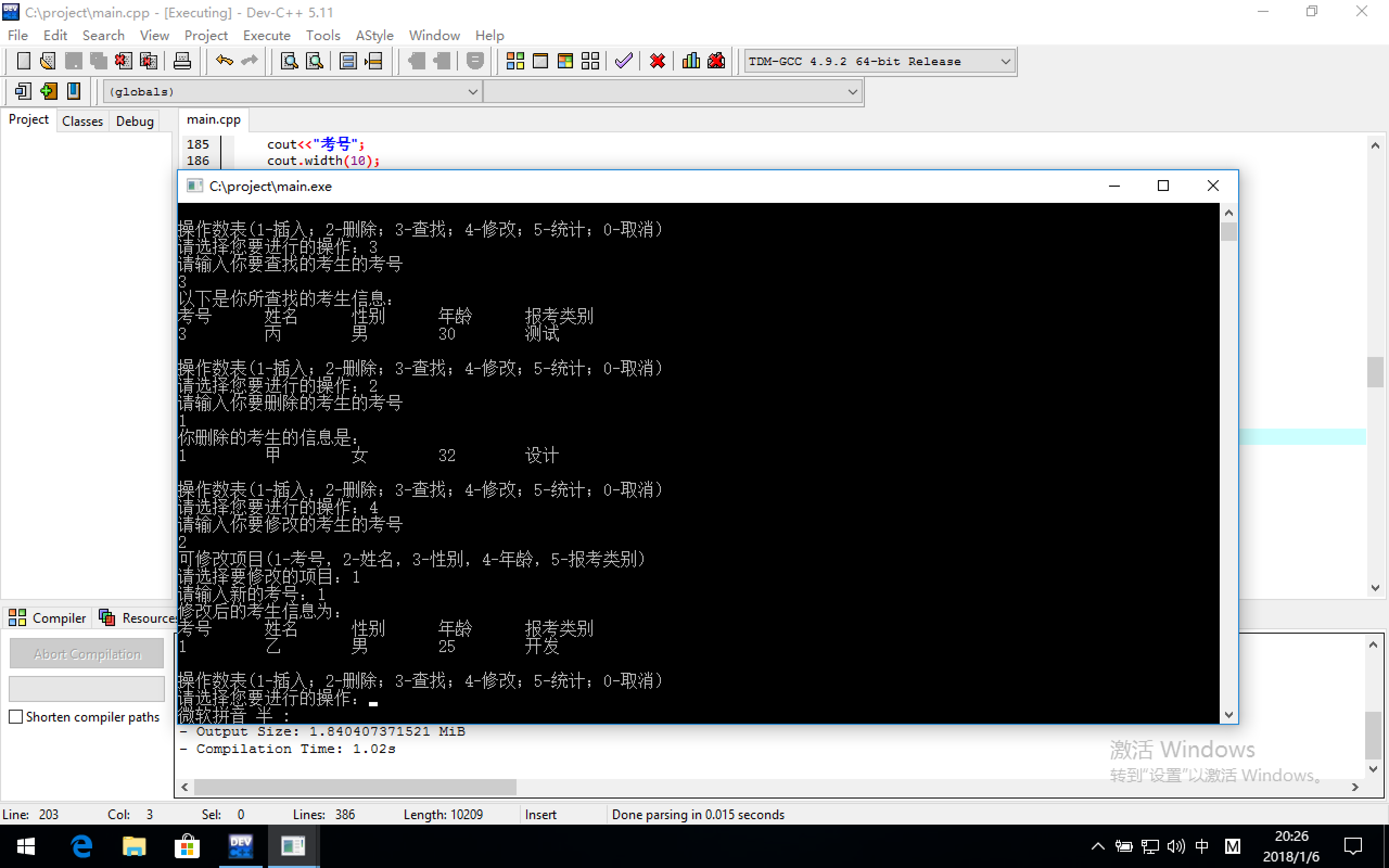
4.1.3查询考生信息



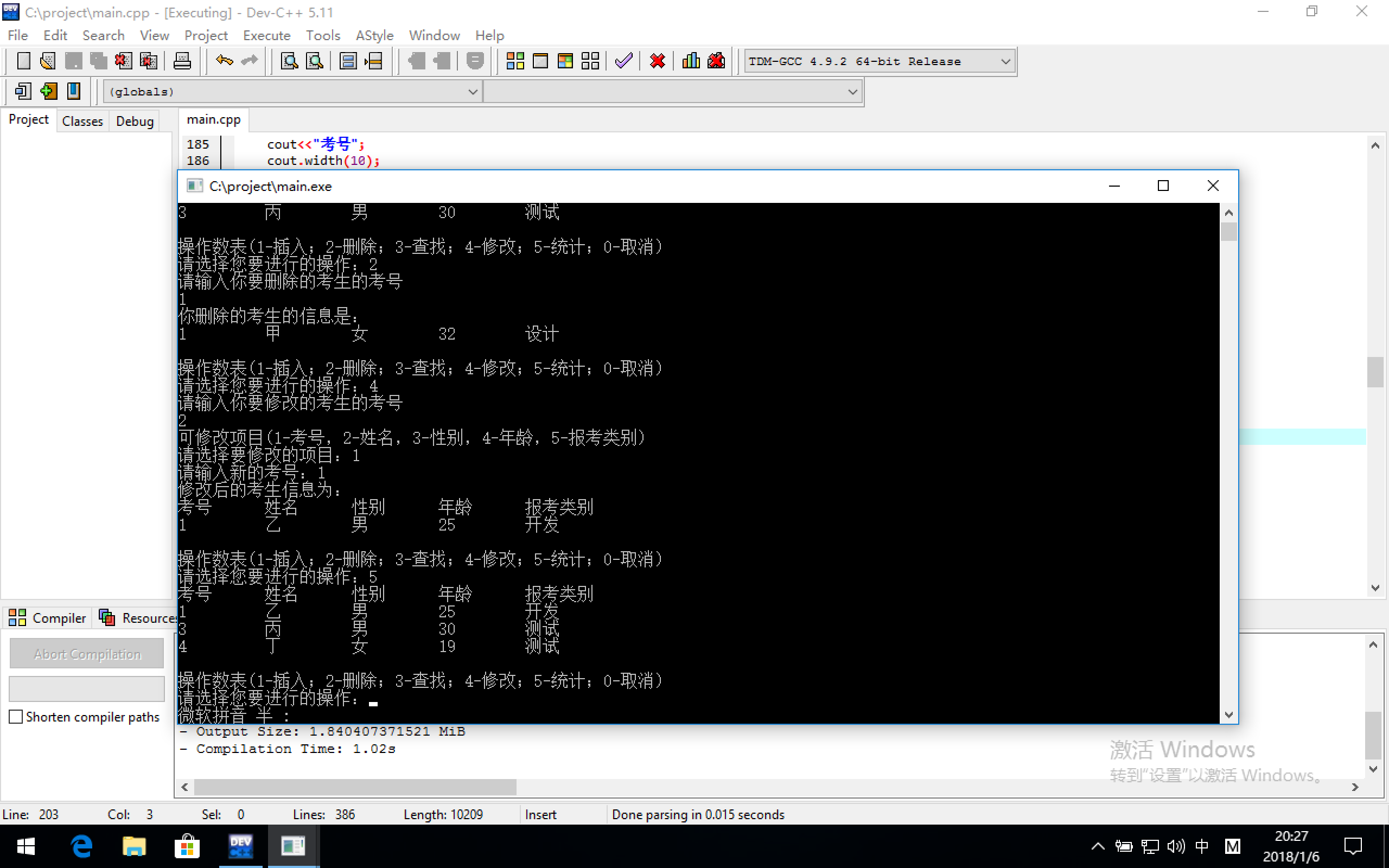
4.1.4删除考生信息



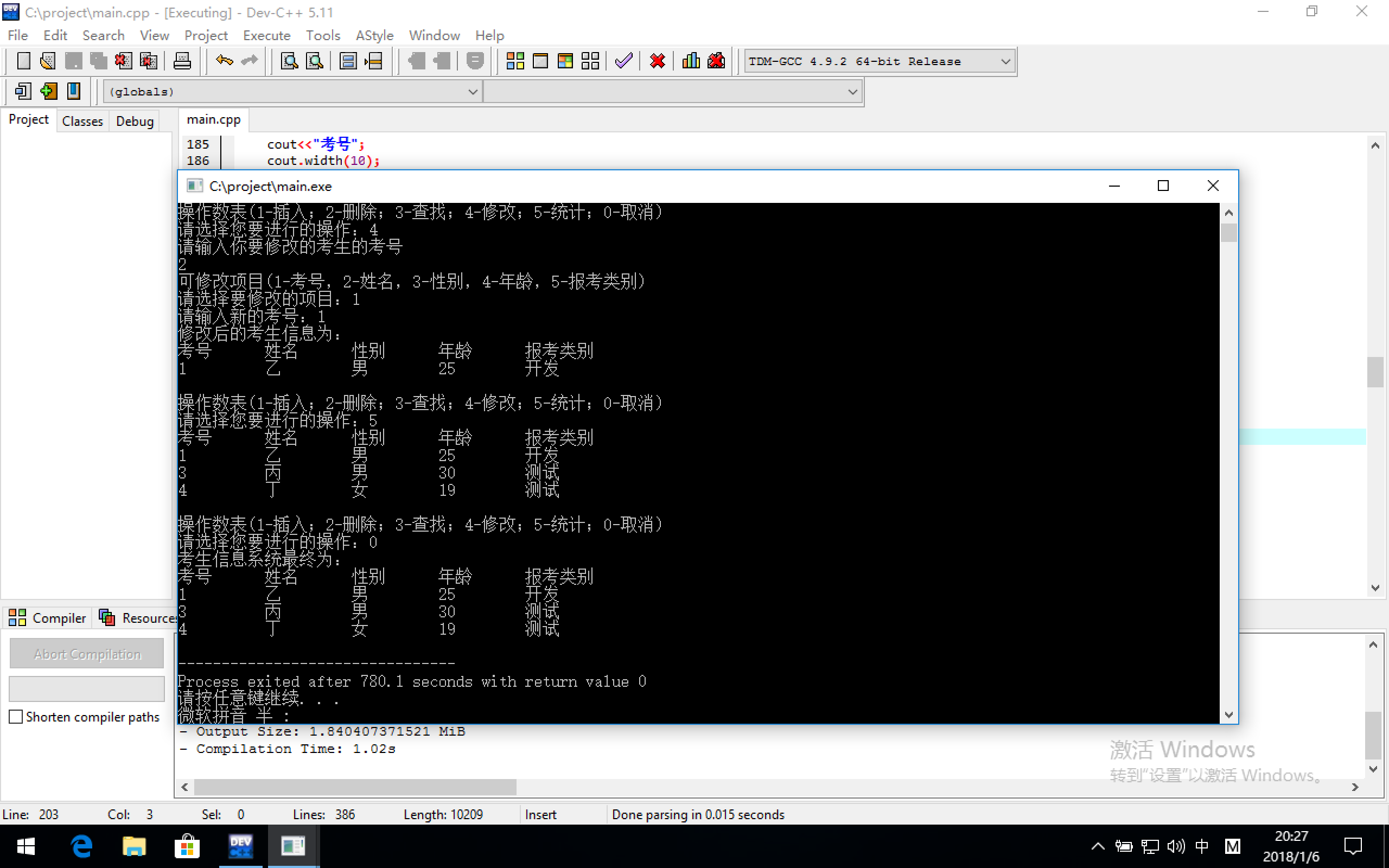
4.1.5修改考生信息



4.1.6输出考生信息

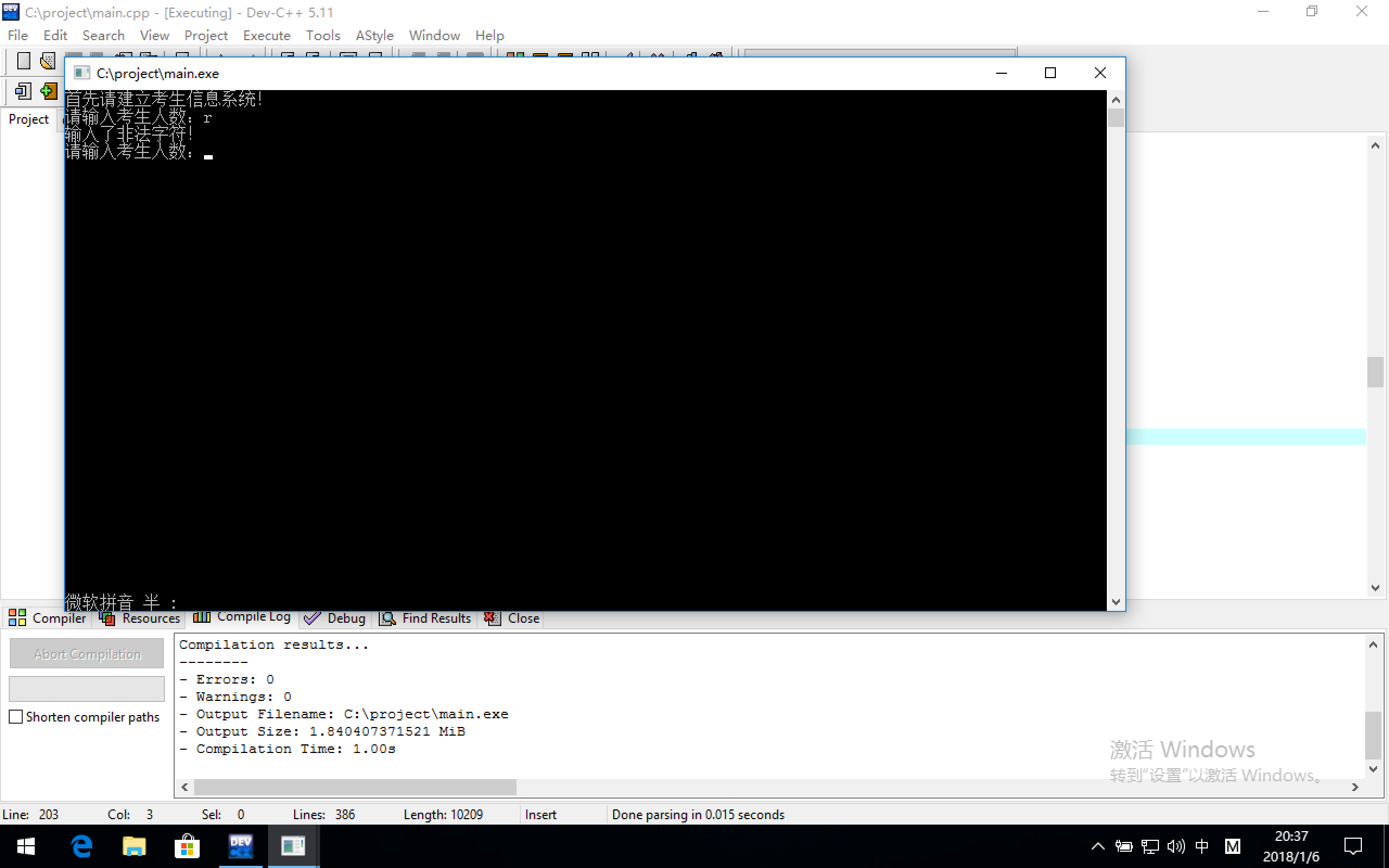


4.1.7退出系统

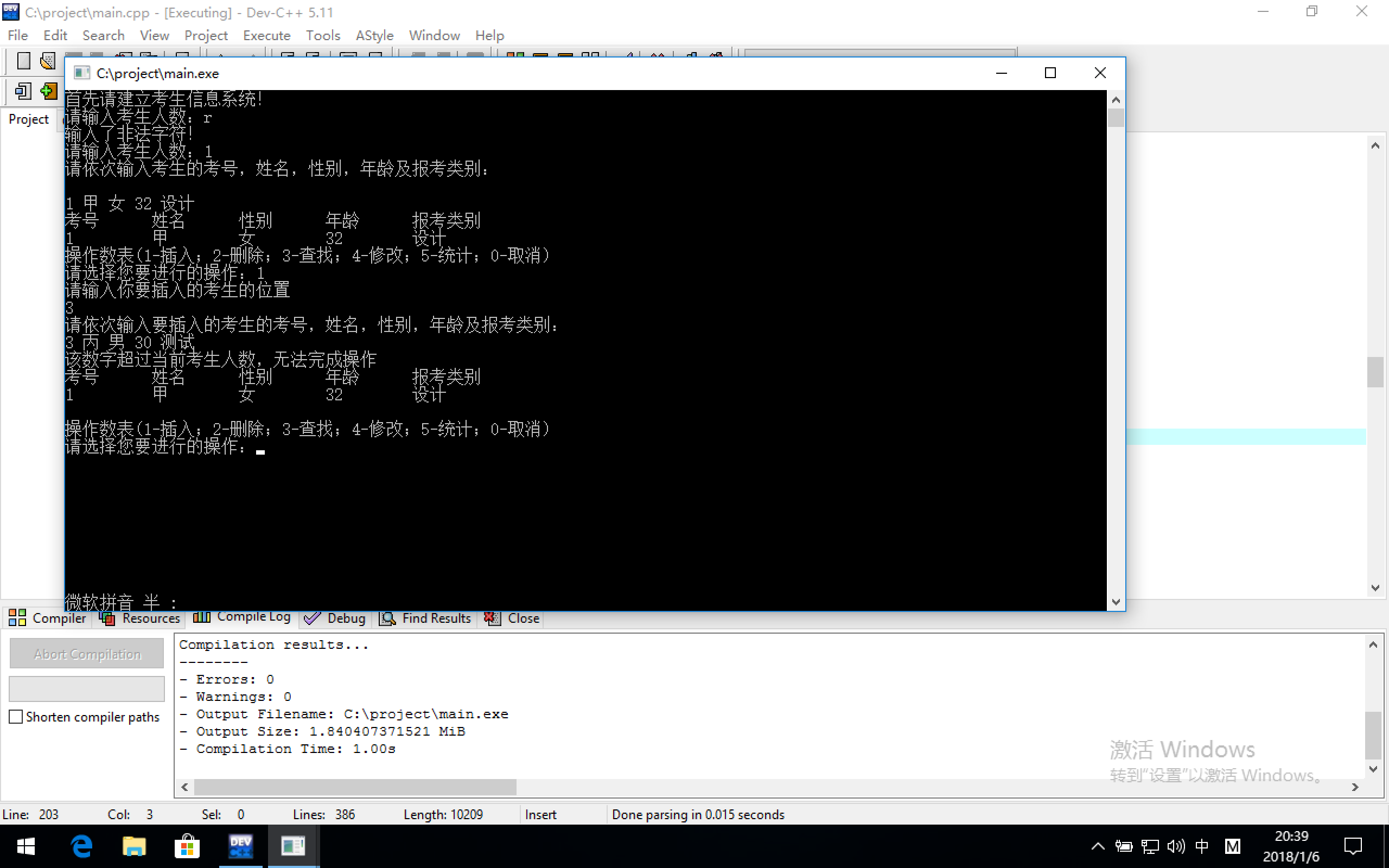


4.2出错测试

4.2.1输入非法字符



4.2.2插入位置不合理



4.2.3根据考号找不到考生

