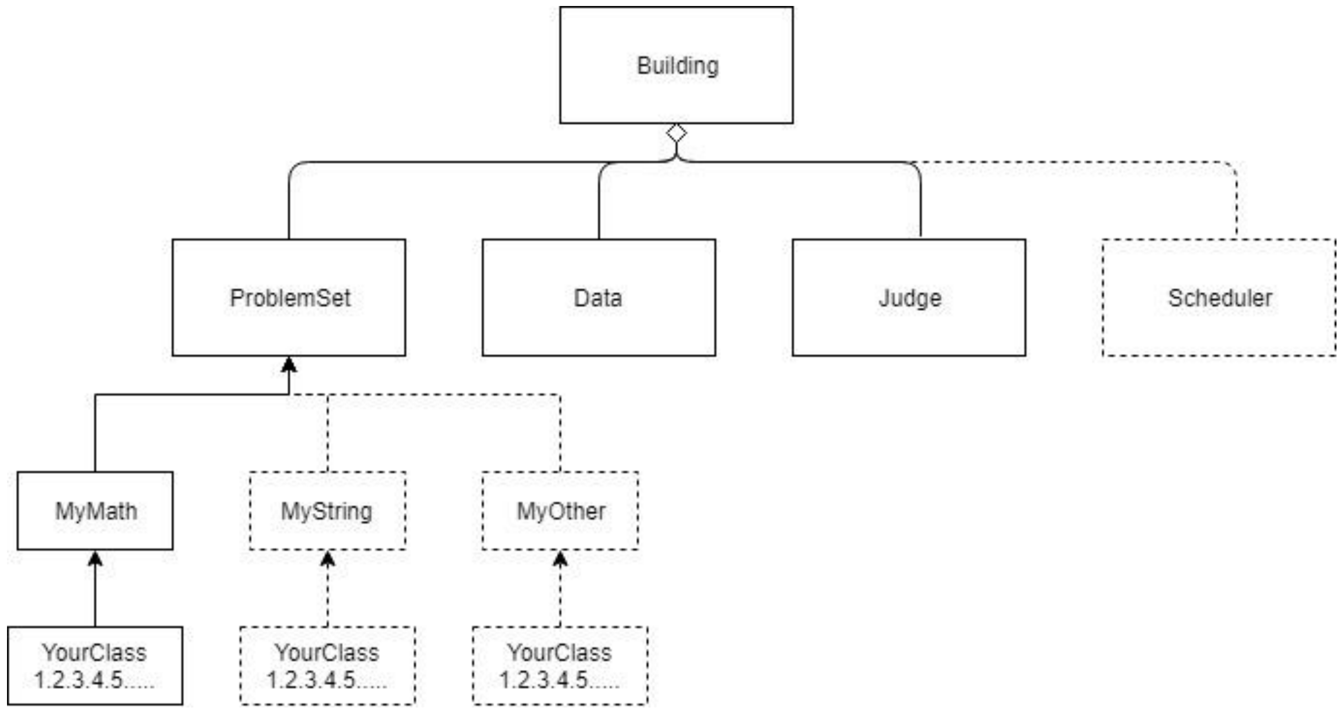
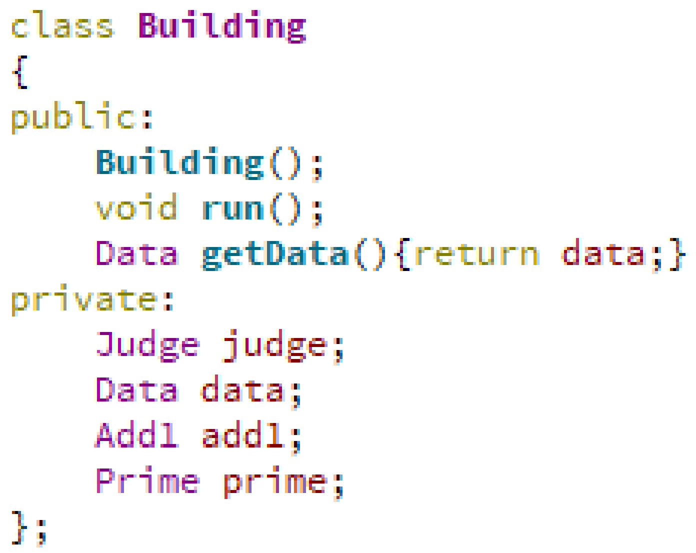
# Course 3

## Structure of Project



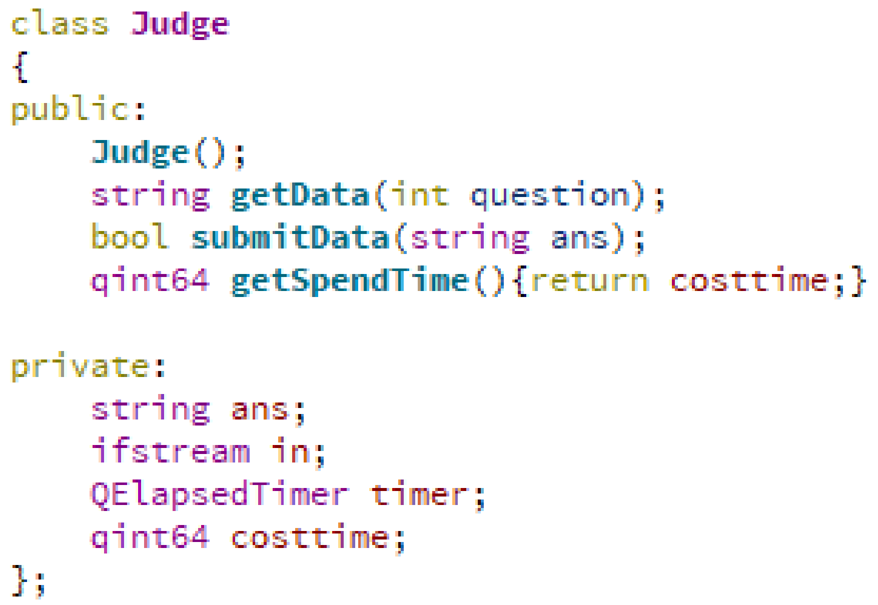
## Building



```
class Building
{
public:
    Building();
    void run();
    Data getData(){return data;}
private:
    Judge judge;
    Data data;
    Add1 add1;
    Prime prime;
};
```

- The basic class of the project.
- Contain Judge , Data ,and two problem sets(Add1 &Prime).

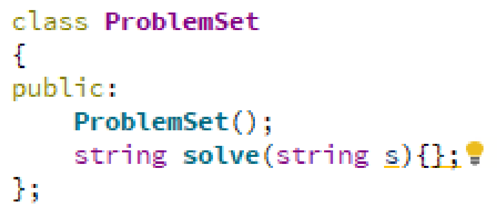
# Judge



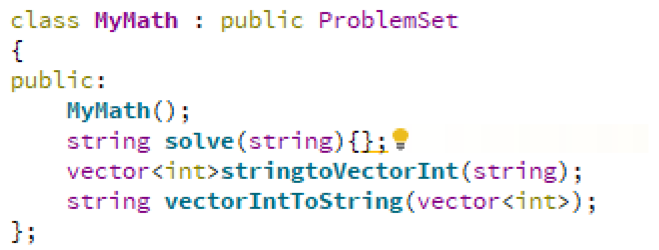
```
class Judge
{
public:
    Judge();
    string getData(int question);
    bool submitData(string ans);
    qint64 getSpendTime(){return costtime;}

private:
    string ans;
    ifstream in;
    QElapsedTimer timer;
    qint64 costtime;
};
```

- getData-read a set of question and answer from file, read all set and randomly select one. And let the timer start.
- submitData-read a submit string. Calculate the time spent, and check the submit answer is correct or not.

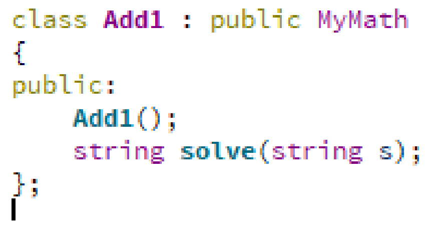
# Problem set, Mymath and MyClass(Add1)…



```
class ProblemSet
{
public:
    ProblemSet();
    string solve(string s){};
};
```



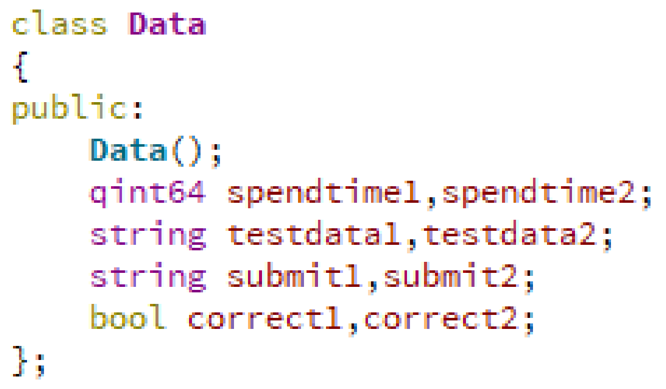
```
class MyMath : public ProblemSet
{
public:
    MyMath();
    string solve(string){};
    vector<int>stringtoVectorInt(string);
    string vectorIntToString(vector<int>);
};
```



```
class Add1 : public MyMath
{
public:
    Add1();
    string solve(string s);
};
```

- ProblemSet-the base class.
- MyMath – inheritance from ProblemSet, has two function to convert between string and vector
- Add1-the class to solve the problem(add 1 to all elements)

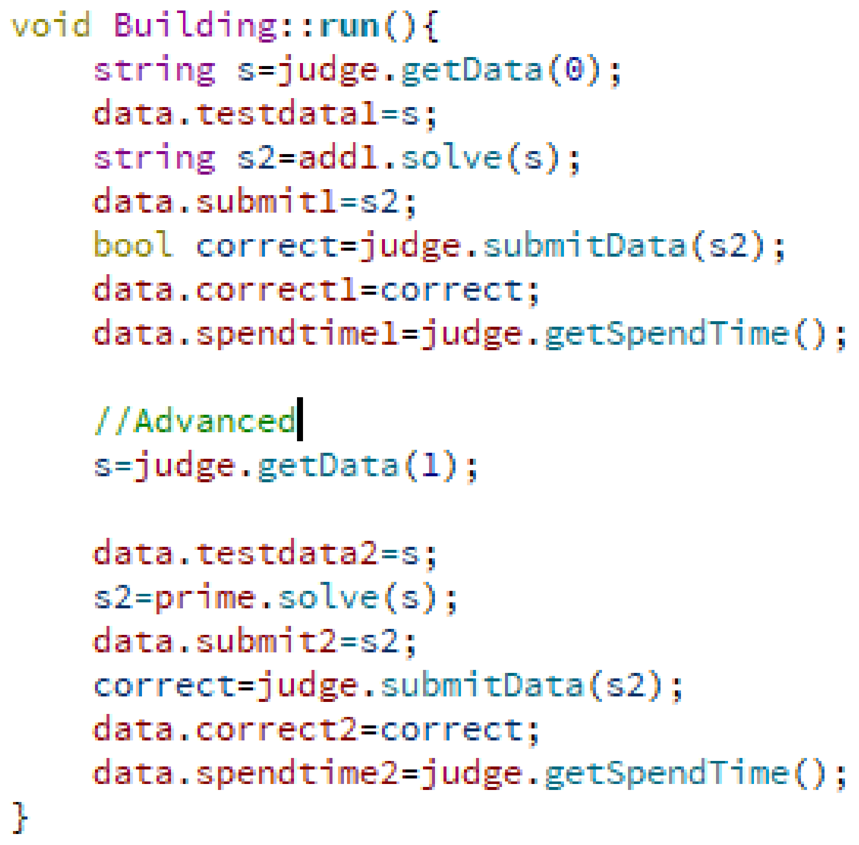
## Data



```
class Data
{
public:
    Data();
    qint64 spendtime1,spendtime2;
    string testdata1,testdata2;
    string submit1,submit2;
    bool correct1,correct2;
};
```

- Just like a structure, store all data(test data, submit data, spend time and correct or not ) that want to display on mainwindow.
- All data members are public.

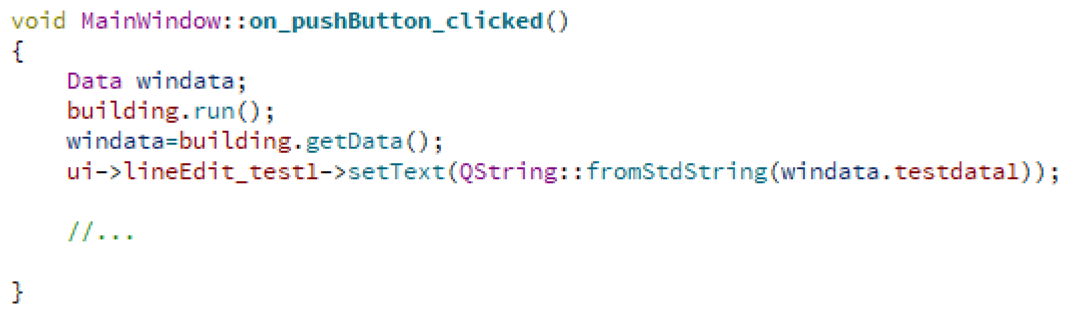
# In Building run function



```
void Building::run(){
    string s=judge.getData(0);
    data.testdata1=s;
    string s2=add1.solve(s);
    data.submit1=s2;
    bool correct=judge.submitData(s2);
    data.correct1=correct;
    data.spendtime1=judge.getSpendTime();

    //Advanced
    s=judge.getData(1);

    data.testdata2=s;
    s2=prime.solve(s);
    data.submit2=s2;
    correct=judge.submitData(s2);
    data.correct2=correct;
    data.spendtime2=judge.getSpendTime();
}
```

1. Get test data from judge
2. Call solve function to get the answer.
3. Submit string to judge and check answer is correct or not
4. Get spend time from judge object.

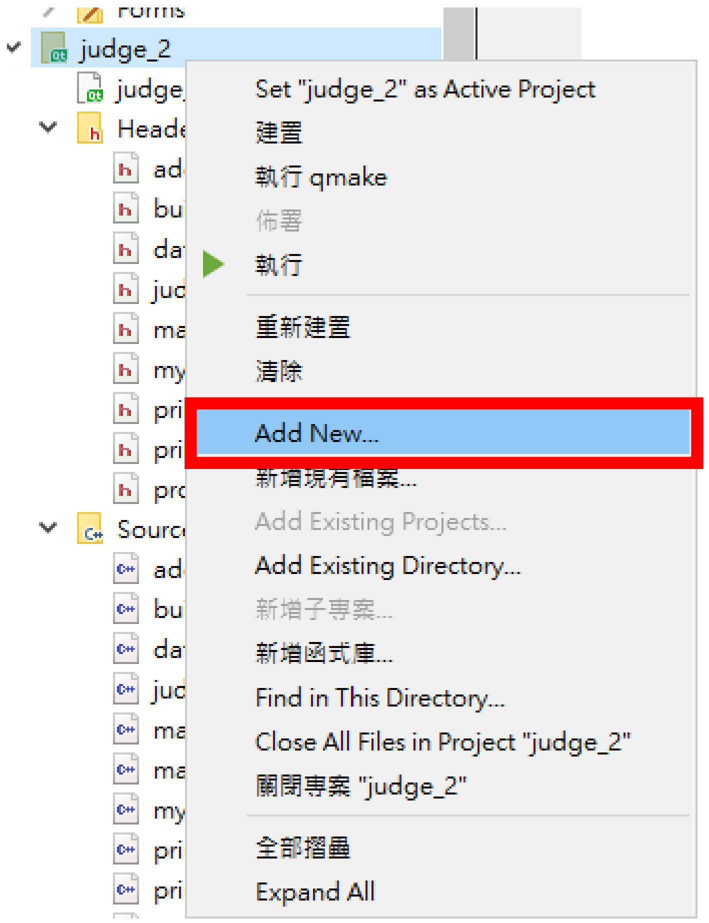
# In Mainwindow

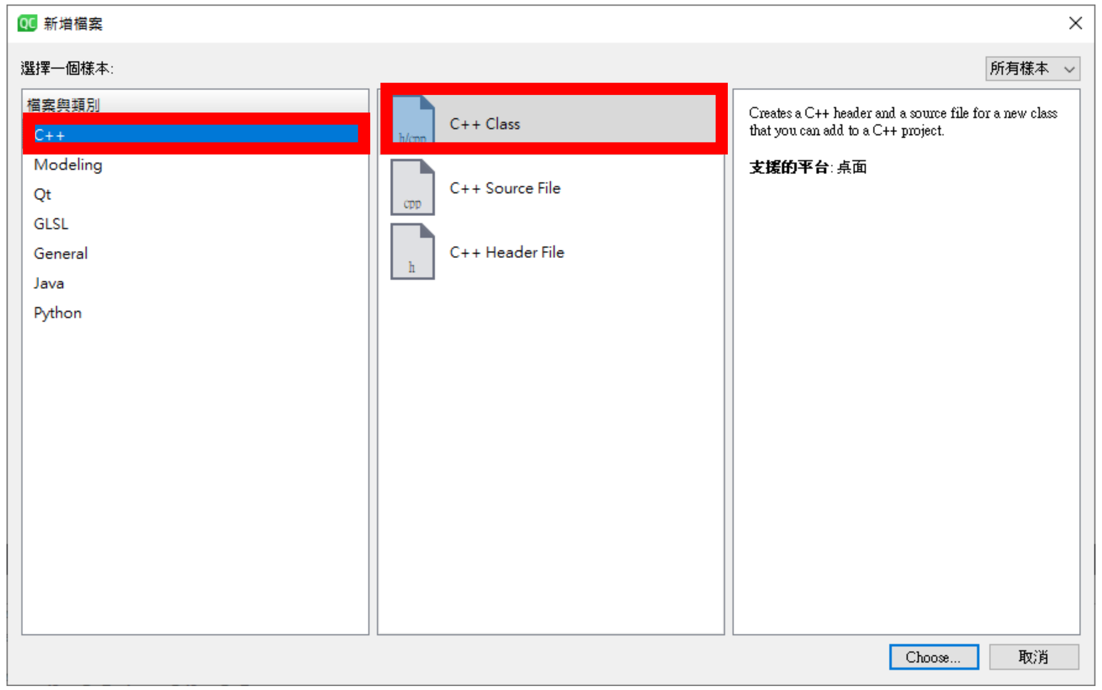


```
void MainWindow::on_pushButton_clicked()
{
    Data windata;
    building.run();
    windata=building.getData();
    ui->lineEdit_test1->setText(QString::fromStdString(windata.testdata1));

    //...

}
```

- Let building start
- Get Data object
- Display the value from data on gui.

# How to add class

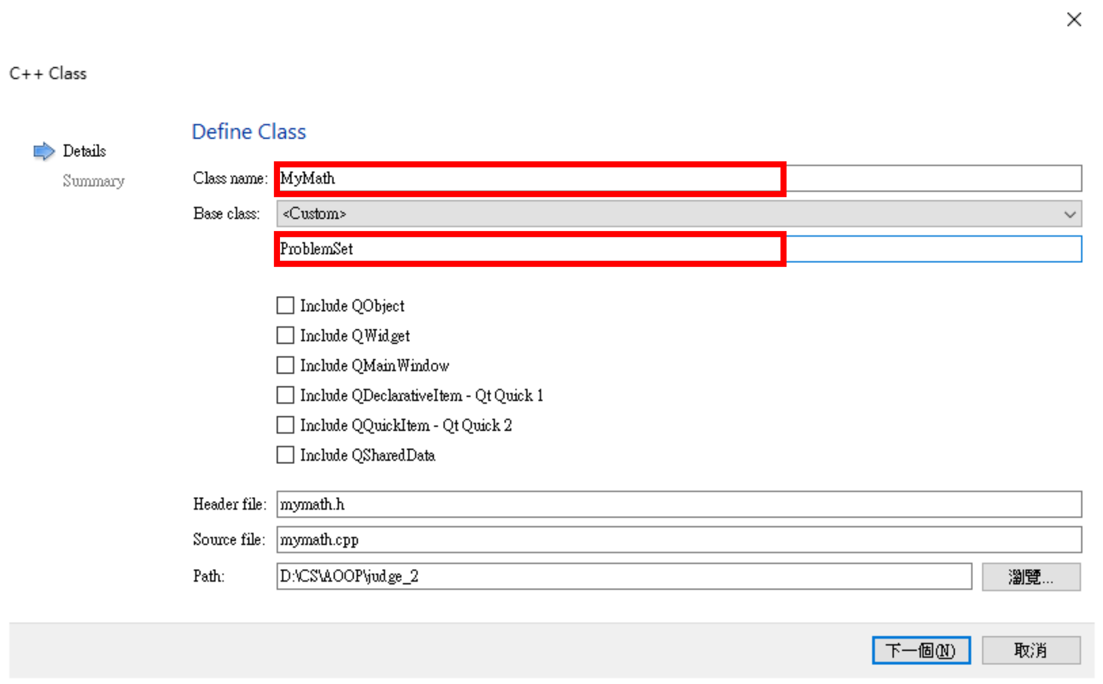




Creates a C++ header and a source file for a new class that you can add to a C++ project.

支援的平台: 桌面

# How to add class



C++ Class

Details
Summary

Define Class

Class name: MyMath

Base class: <Custom>

ProblemSet

- [ ] Include QObject
- [ ] Include QWidget
- [ ] Include QMainWindow
- [ ] Include QDeclarativeItem - Qt Quick 1
- [ ] Include QQuickItem - Qt Quick 2
- [ ] Include QSharedData

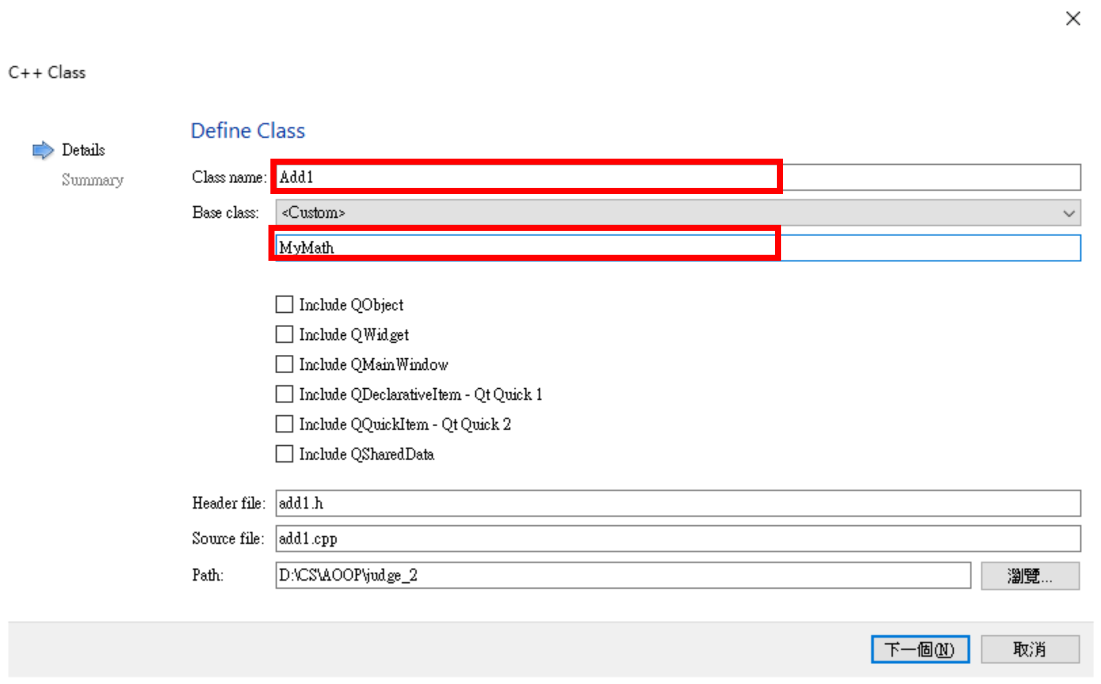
Header file: mymath.h

Source file: mymath.cpp

Path: D:\CS\AOOP\judge_2

瀏覽...

下一個(N) 取消



C++ Class

Details
Summary

Define Class

Class name: Add1

Base class: <Custom>

MyMath

- [ ] Include QObject
- [ ] Include QWidget
- [ ] Include QMainWindow
- [ ] Include QDeclarativeItem - Qt Quick 1
- [ ] Include QQuickItem - Qt Quick 2
- [ ] Include QSharedData

Header file: add1.h

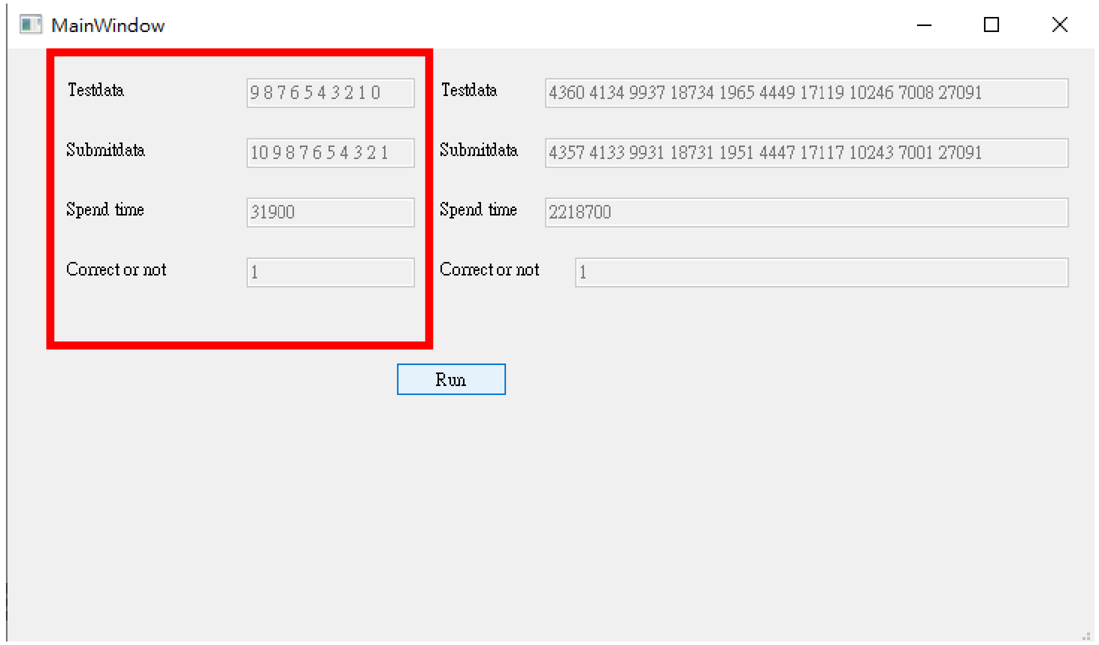
Source file: add1.cpp

Path: D:\CS\AOOP\judge_2

瀏覽...

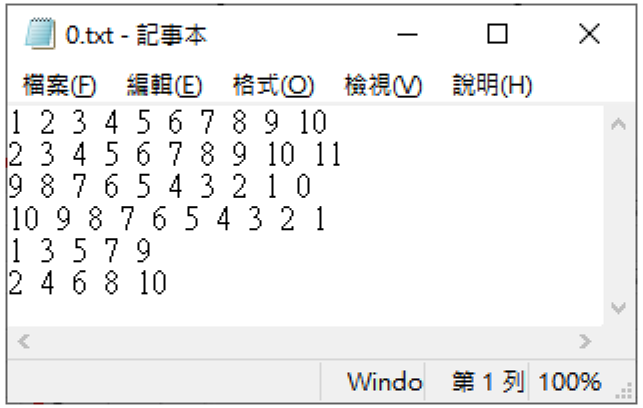
下一個(N) 取消

# Today goal (basic)



MainWindow

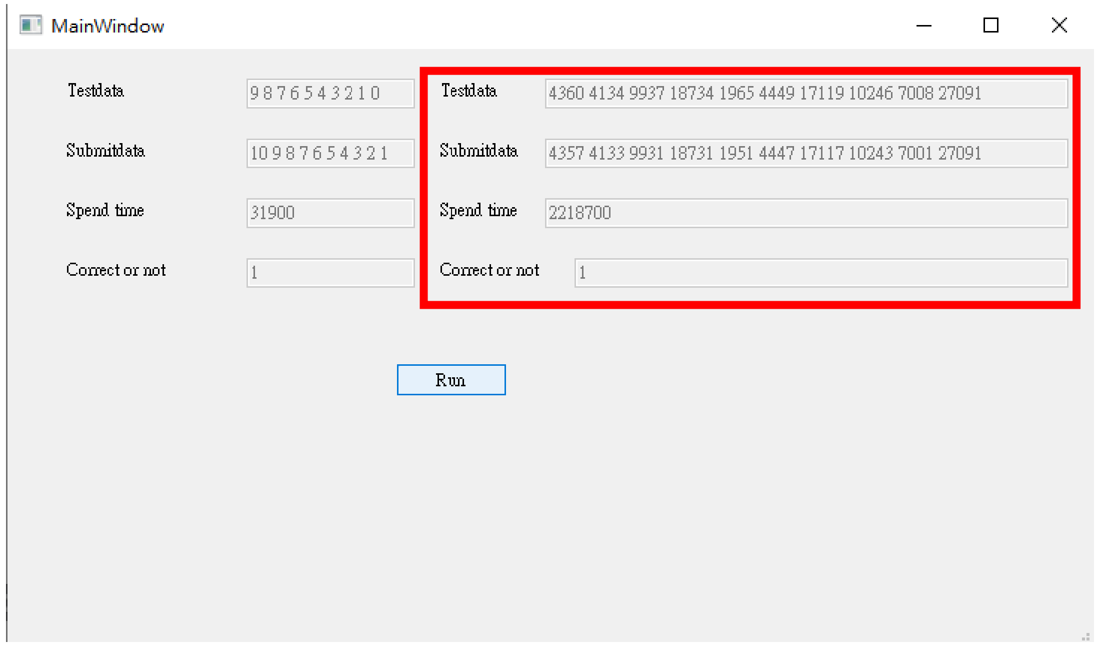
| | | | |
|---|---|---|---|
| Testdata | 9 8 7 6 5 4 3 2 1 0 | Testdata | 4360 4134 9937 18734 1965 4449 17119 10246 7008 27091 |
| Submitdata | 10 9 8 7 6 5 4 3 2 1 | Submitdata | 4357 4133 9931 18731 1951 4447 17117 10243 7001 27091 |
| Spend time | 31900 | Spend time | 2218700 |
| Correct or not | 1 | Correct or not | 1 |



```
1 2 3 4 5 6 7 8 9 10
2 3 4 5 6 7 8 9 10 11
9 8 7 6 5 4 3 2 1 0
10 9 8 7 6 5 4 3 2 1
1 3 5 7 9
2 4 6 8 10
```

- Finish the structure of the project
- Judge read from 0.txt and randomly select one set(each two lines are one set first line is the test data and second is the answer).
- Add 1 to all elements.
- Display the result.

# Today goal(advanced)



| | | | |
|---|---|---|---|
| Testdata | 9 8 7 6 5 4 3 2 1 0 | Testdata | 4360 4134 9937 18734 1965 4449 17119 10246 7008 27091 |
| Submitdata | 10 9 8 7 6 5 4 3 2 1 | Submitdata | 4357 4133 9931 18731 1951 4447 17117 10243 7001 27091 |
| Spend time | 31900 | Spend time | 2218700 |
| Correct or not | 1 | Correct or not | 1 |

- Judge read from 1.txt
- Get the biggest prime that is smaller or equal to the element.
- Display the result

# Useful function

- <string>

int stoi(string)

string to_string(int,double,…)

- <QString>

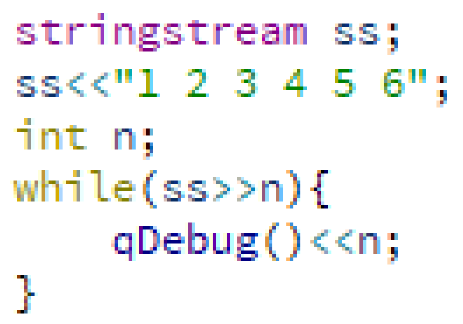
QString QString::fromStdstring(string)

Qstring

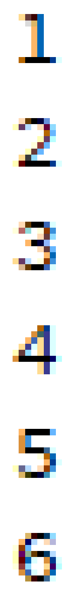
Qstring::number(int,float……)

## Optional

- #include<sstream>
- http://www.cplusplus.com/reference/sstream/stringstream/



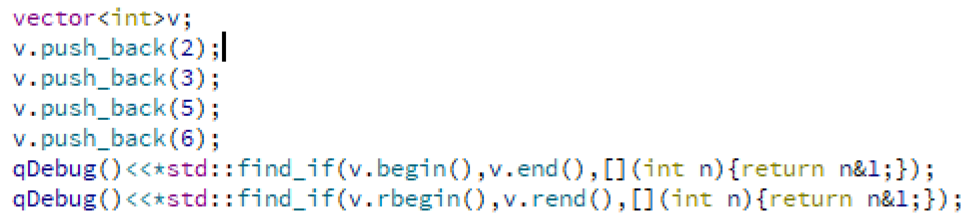
```
stringstream ss;
ss<<"1 2 3 4 5 6";
int n;
while(ss>>n){
    qDebug()<<n;
}
```



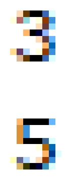
```
1
2
3
4
5
6
```

- #include<algorithm>
- find_if

http://www.cplusplus.com/reference/algorithm/find_if/



```
vector<int>v;
v.push_back(2);
v.push_back(3);
v.push_back(5);
v.push_back(6);
qDebug()<<*std::find_if(v.begin(),v.end(),[](int n){return n&1;});
qDebug()<<*std::find_if(v.rbegin(),v.rend(),[](int n){return n&1;});
```



```
3
5
```