

知识点1【qt creator的安装】基本上一路next

---

- 1、双击可执行文件

知识点2【Qt创建c++工程】

---

知识点3【qt creator编译c++工程注意事项】

---

- 1、运行结果 在windows的控制台中 输出。
  - 2、中文输出是乱码 (minGW == mini gnu for widows) 默认编码格式 是linux的UTF8 需要设置系统的编码格式 system
  - 3、qtcreator 假死
- 

知识点4【c++的第一个程序】

---

知识点5【c++的三大特性】

---

知识点6【c++对c的扩展】

---

- 1、::作用域运算符 (表明 数据、方法 的归属性问题)
  - 2、命名空间 namespace 解决命名冲突
    - 2.1: namespace命名空间的定义
    - 2.2:命名空间只能全局范围内定义 (以下错误写法)
    - 2.3:命名空间可嵌套命名空间
    - 2.4:命名空间是开放的, 即可以随时把新的成员加入已有的命名空间中(常用)
    - 2.5: 命名空间 可以存放 变量 和 函数
    - 2.6: 命名空间中的函数 可以在“命名空间”外 定义
    - 2.7: 无名命名空间, 意味着命名空间中的标识符只能在本文件内访问, 相当于给这个标识符加上了static, 使得其可以作为内部连接 (了解)
    - 2.8: 给命名空间 取个别名 (了解)
- 

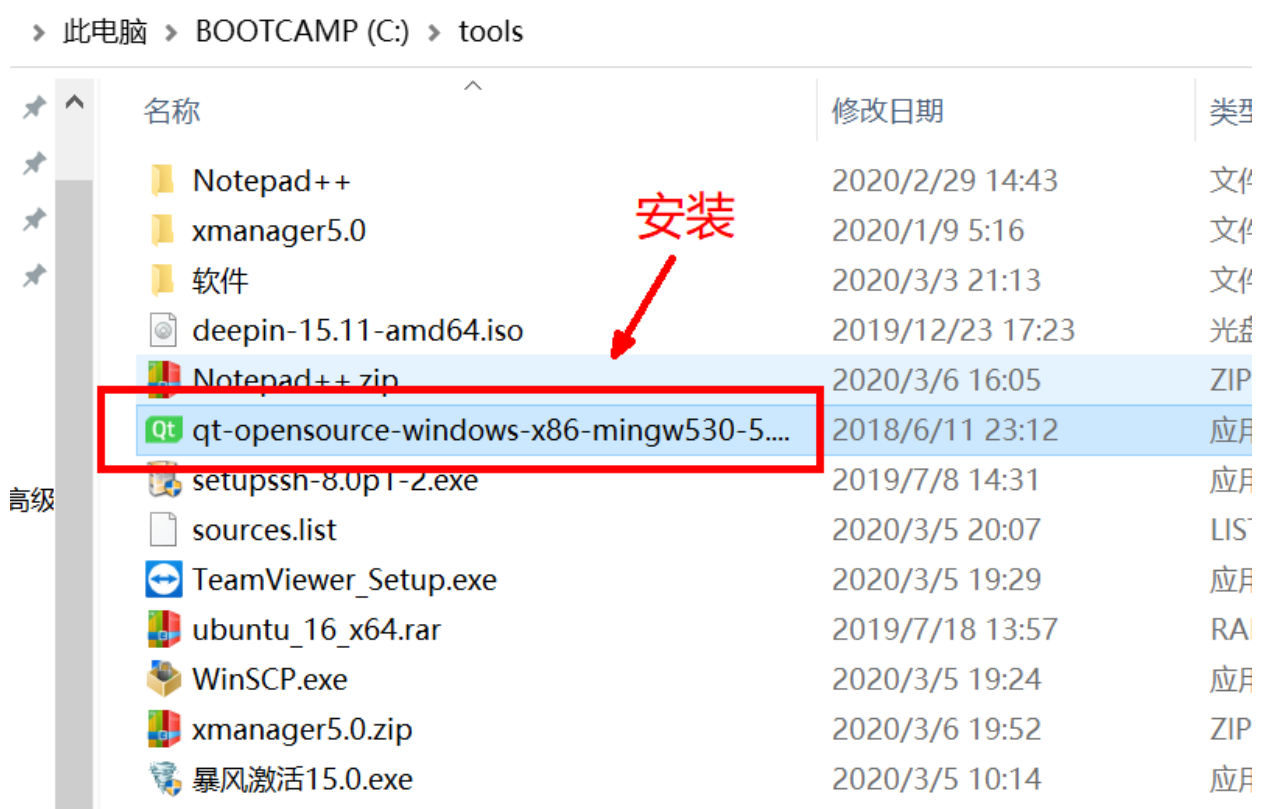
知识点7【using 使用命名空间】

---

- 1、简化了从命名空间的成员访问
- 2、using 使用整个命名空间
- 3、using 指明使用具体的命名空间 成员。（了解）
- 4、using声明碰到函数重载（了解）
- 5、不同命名空间中的 同名成员 使用的时候注意 二义性

## 知识点1 【qt creator的安装】基本上一路next

### 1、双击可执行文件





## Qt 5.8.0 Setup

### Welcome to the Qt 5.8.0 installer



This installer provides you with the open source version of Qt 5.8.0.

You have the option to log in using your Qt Account credentials (e.g. Qt Forum login).

If you do not have a Qt Account yet, you can opt to create one in the next step.

[Qt Account gives you access to everything Qt](#)  
[Packaging and pricing options](#)  
[LGPL compliance & obligations](#)  
[Choosing the right license for your project](#)



Next

Cancel



Qt 5.8.0 Setup

## Qt Account – Your unified login to everything Qt

Please log in to Qt Account

Login

[Forgot password?](#)

Need a Qt Account?

Sign-up



I accept the [service terms](#).

Settings

Skip

Cancel



Qt 5.8.0 Setup

## Setup - Qt 5.8.0

Welcome to open source Qt 5.8.0 setup.

Next

Quit



## Qt 5.8.0 Setup

### Installation Folder

Please specify the folder where Qt 5.8.0 will be installed.

☒ Associate common file types with Qt Creator.



Qt 5.8.0 Setup

## Select Components

Please select the components you want to install.

<input checked="" type="checkbox"/> Qt	Qt 5.8.0
> <input checked="" type="checkbox"/> Qt 5.8	This component will
> <input checked="" type="checkbox"/> Tools	occupy
	approximately 4.38
	GiB on your hard
	disk drive.

Default

Select All

Deselect All

Next

Cancel



## License Agreement

Please read the following license agreement. You must accept the terms contained in this agreement before continuing with the installation.

### GENERAL

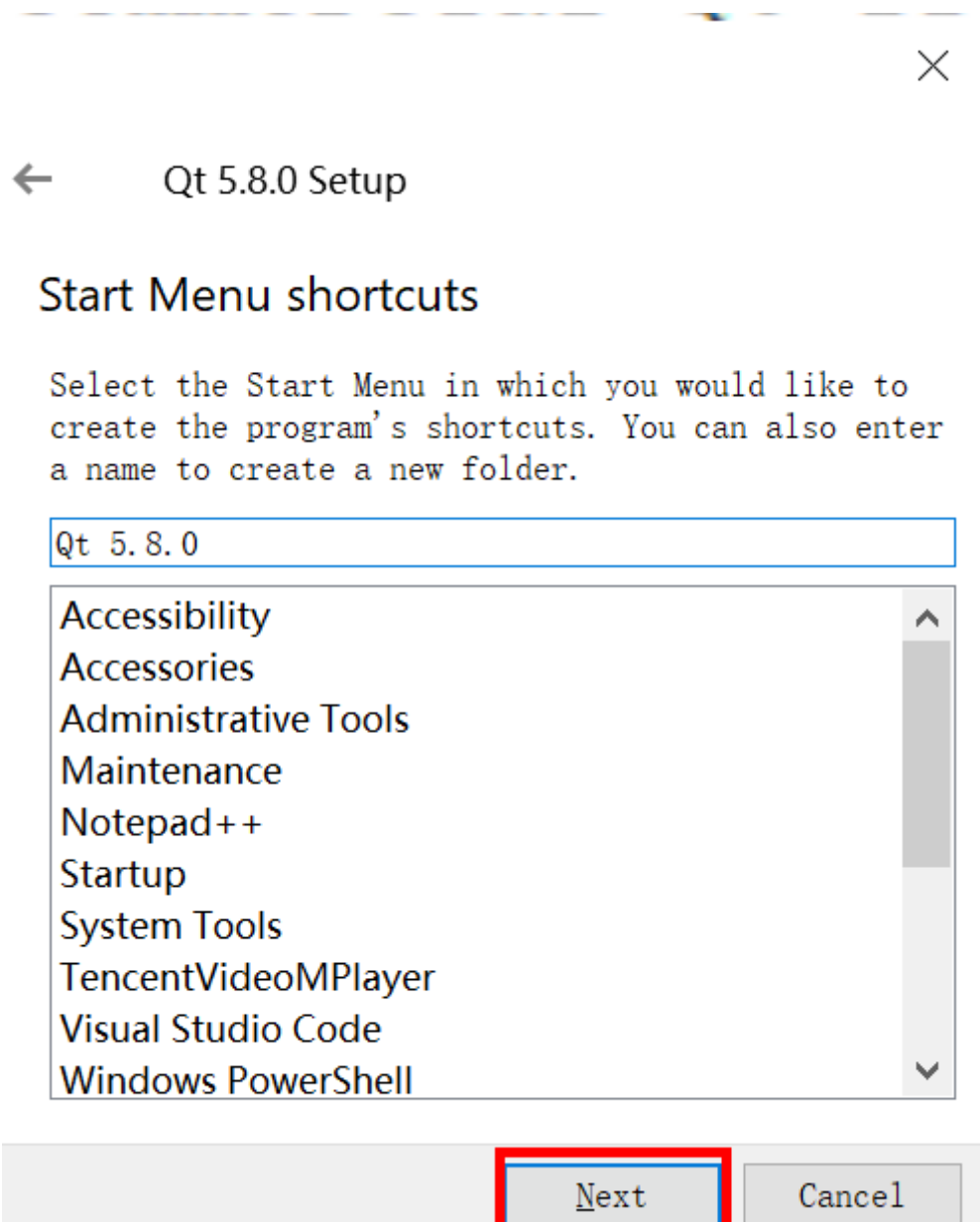
Qt is available under a commercial license with various pricing models and packages that meet a variety of needs. Commercial Qt license keeps your code proprietary where only you can control and monetize on your end product's development, user experience and distribution.

- ☒ I have read and agree to the terms contained in the license agreements.
- ☐ I do not accept the terms and conditions of the above license agreements.

Next

Cancel








Qt 5.8.0 Setup

## Ready to Install

Setup is now ready to begin installing Qt 5.8.0 on your computer. Installation will use 4.38 GiB of disk space.



Install

Cancel



← Qt 5.8.0 Setup

## Installing Qt 5.8.0



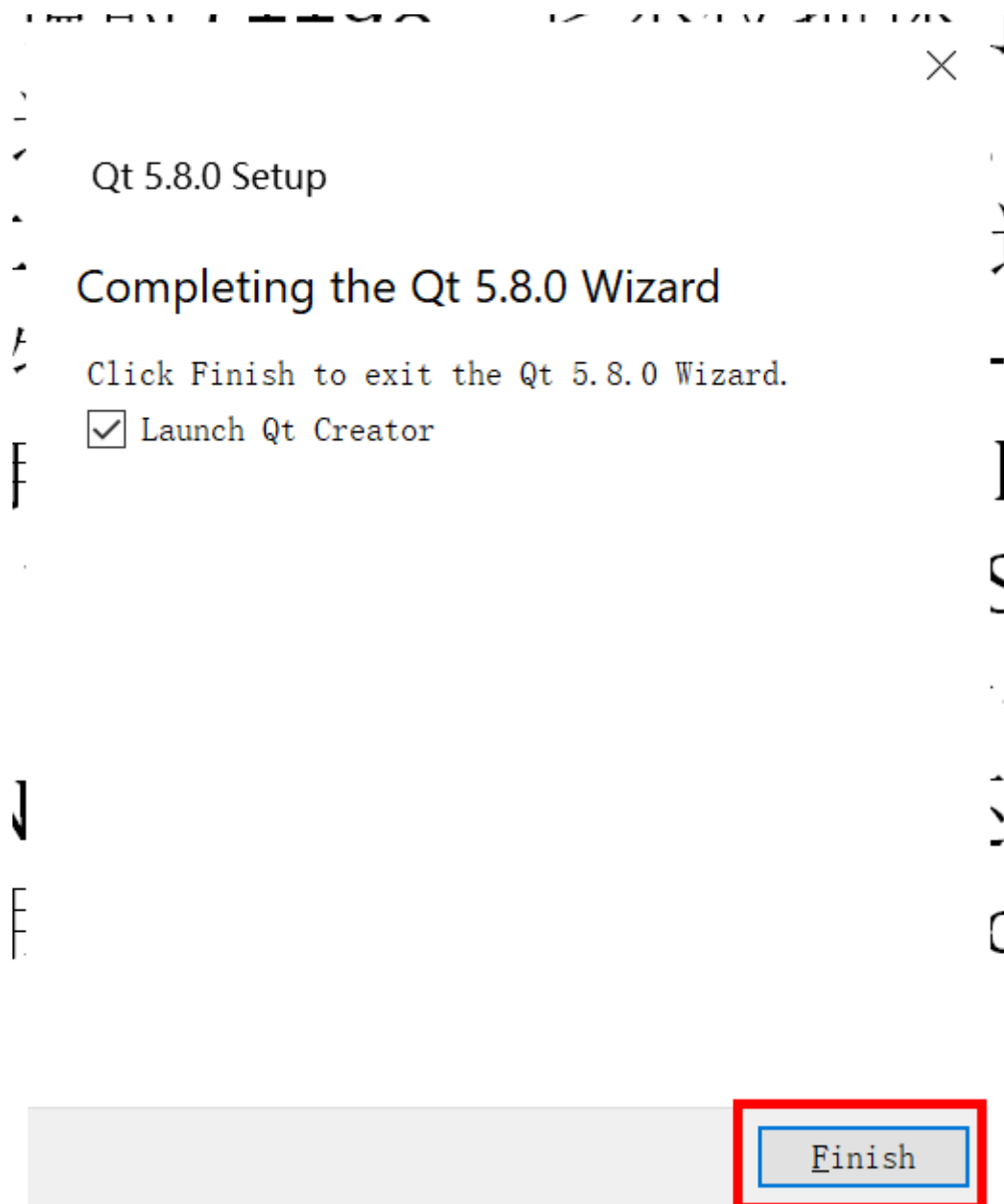
Installing component Qt Creator 4.2.1

Show Details

安静的等待

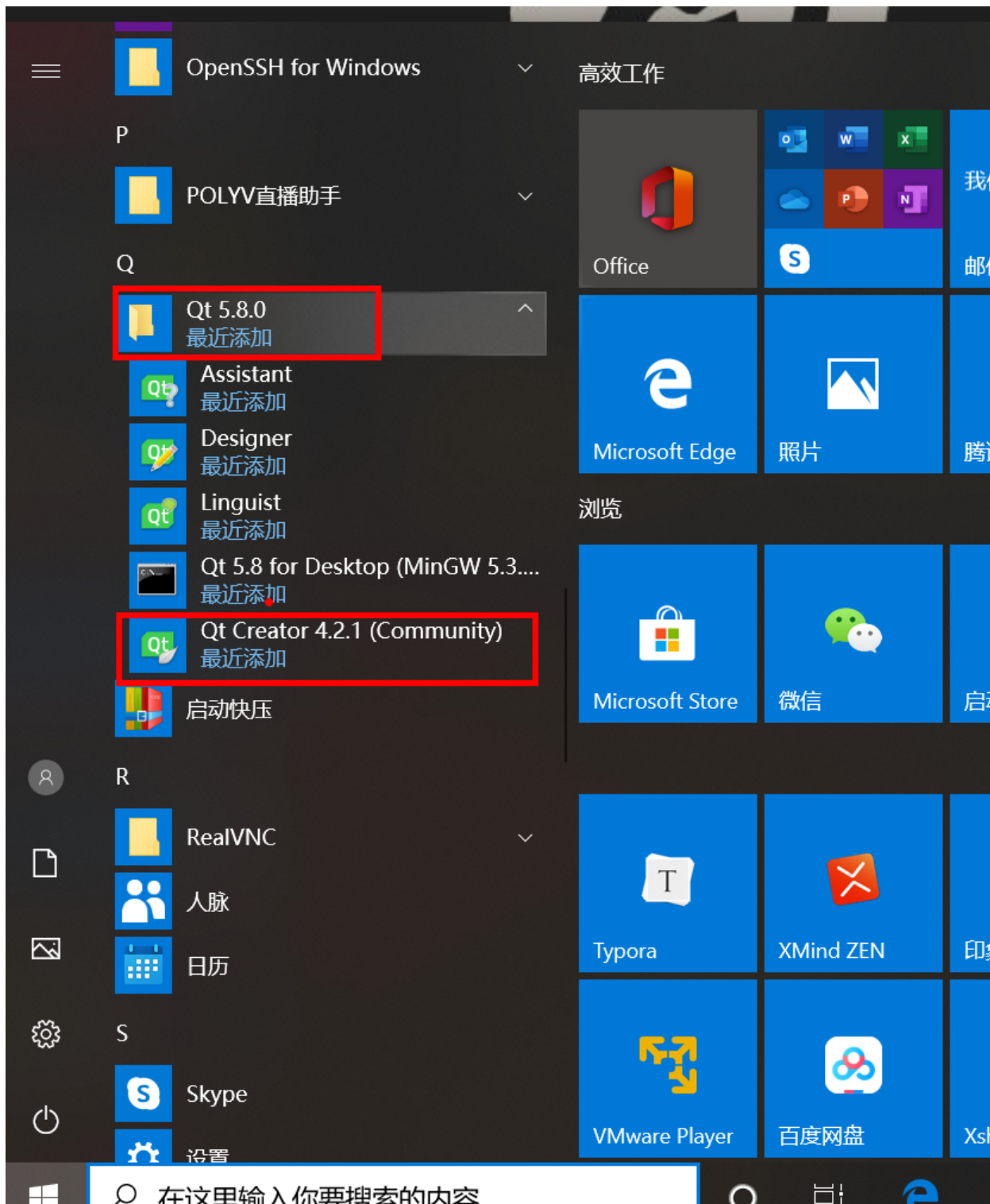
Install

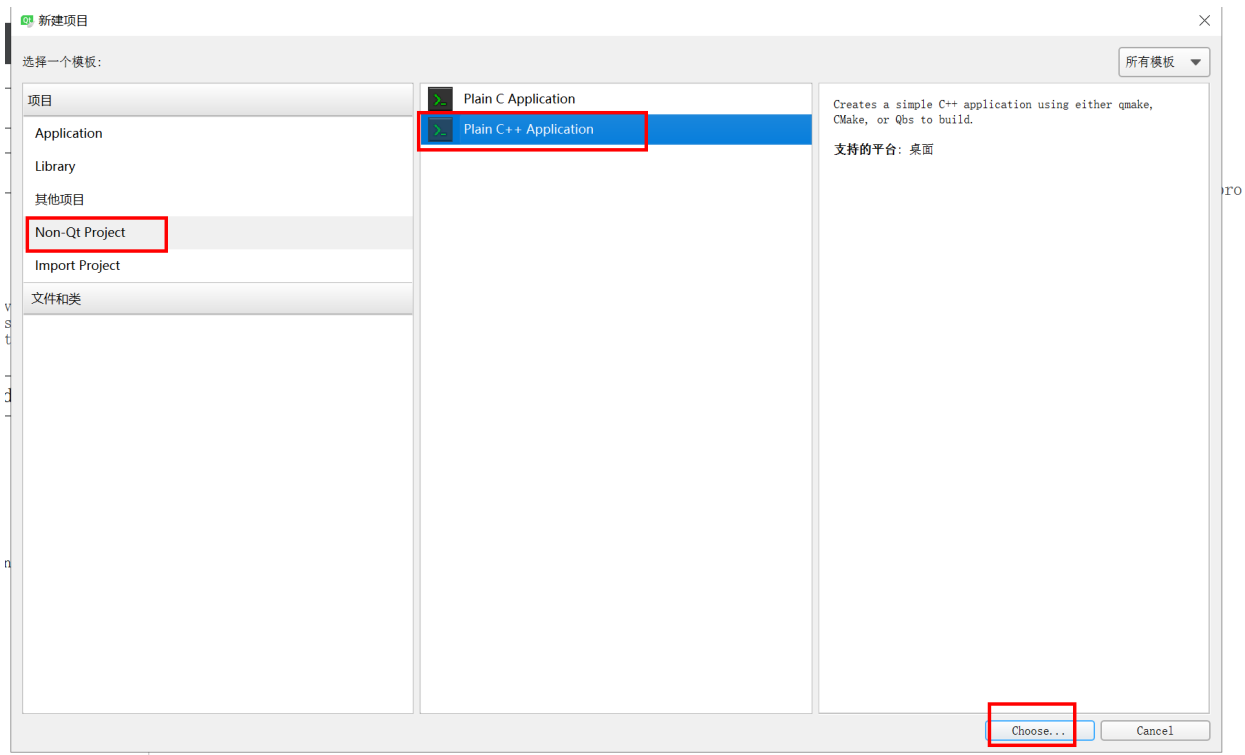
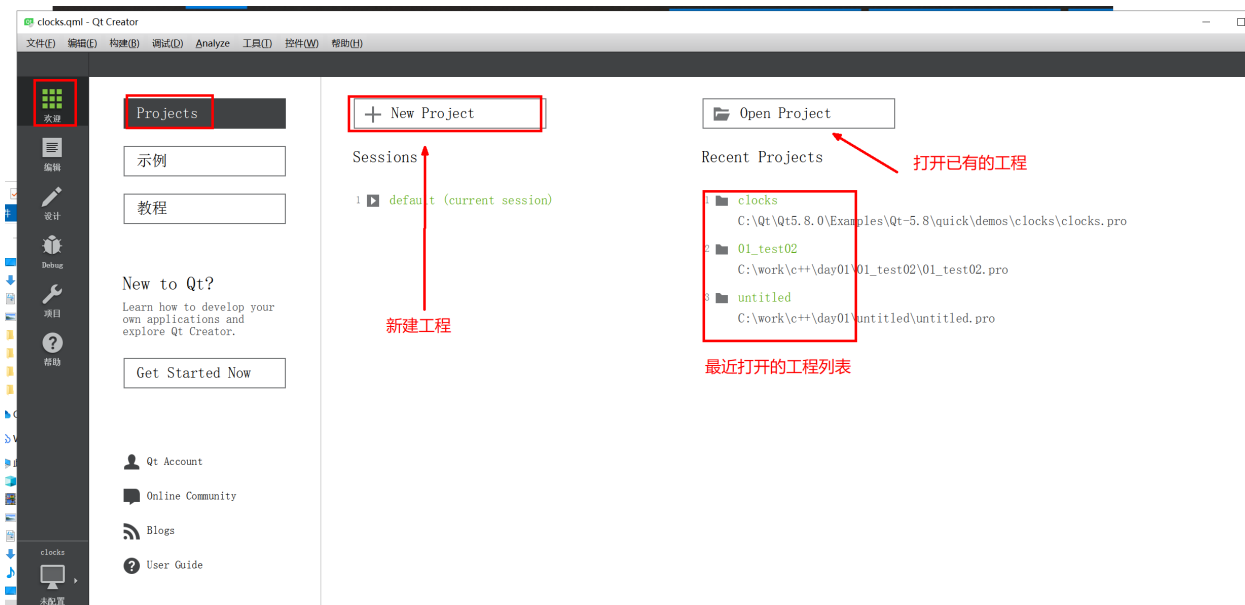
Cancel



---

## 知识点2 【Qt创建c++工程】





Plain C++ Application

Project Location

Creates a simple C++ application using either qmake, CMake, or Qbs to build.

Location

Build System

Kits

Summary

工程名 不能有空格 中文 特殊符号

名称: 00\_hello\_cpp1

创建路径: C:\work\c++\day01 工程存放的路径

☐ 设为默认的项目路径

下一步 (N) 取消

Plain C++ Application

Define Build System

Build system: qmake

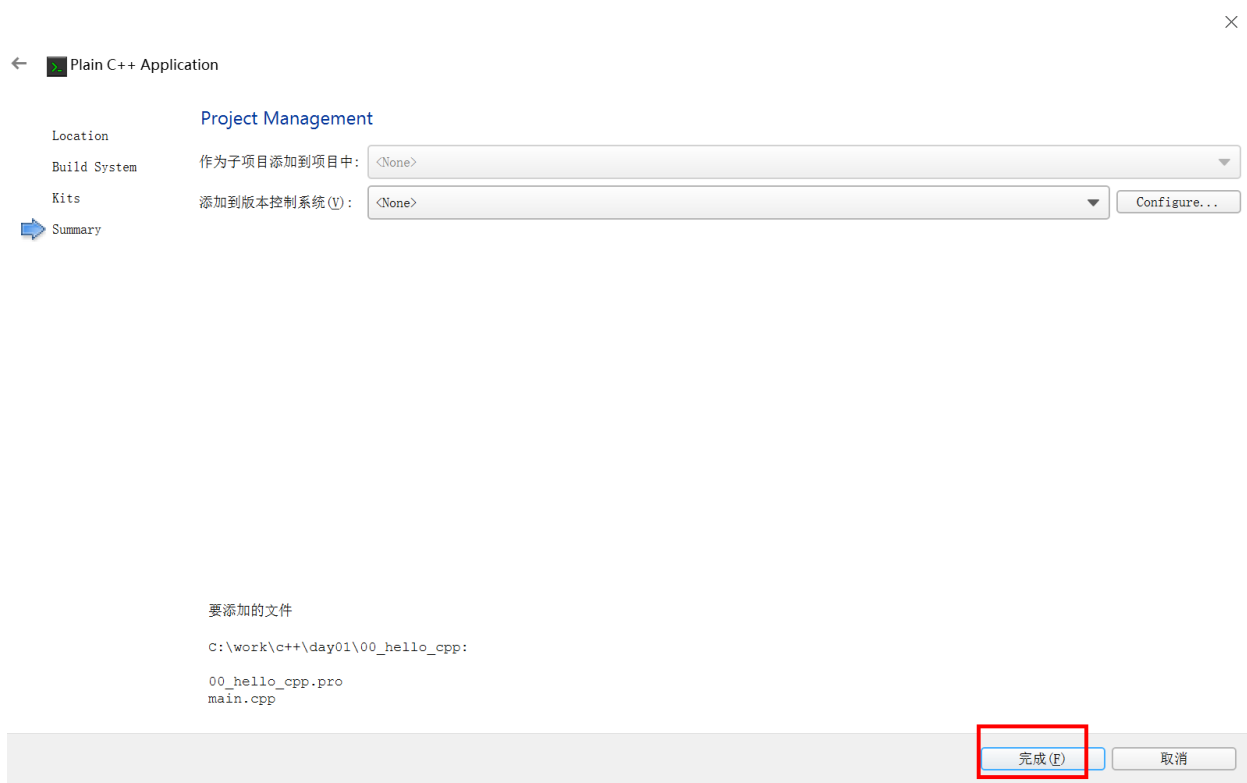
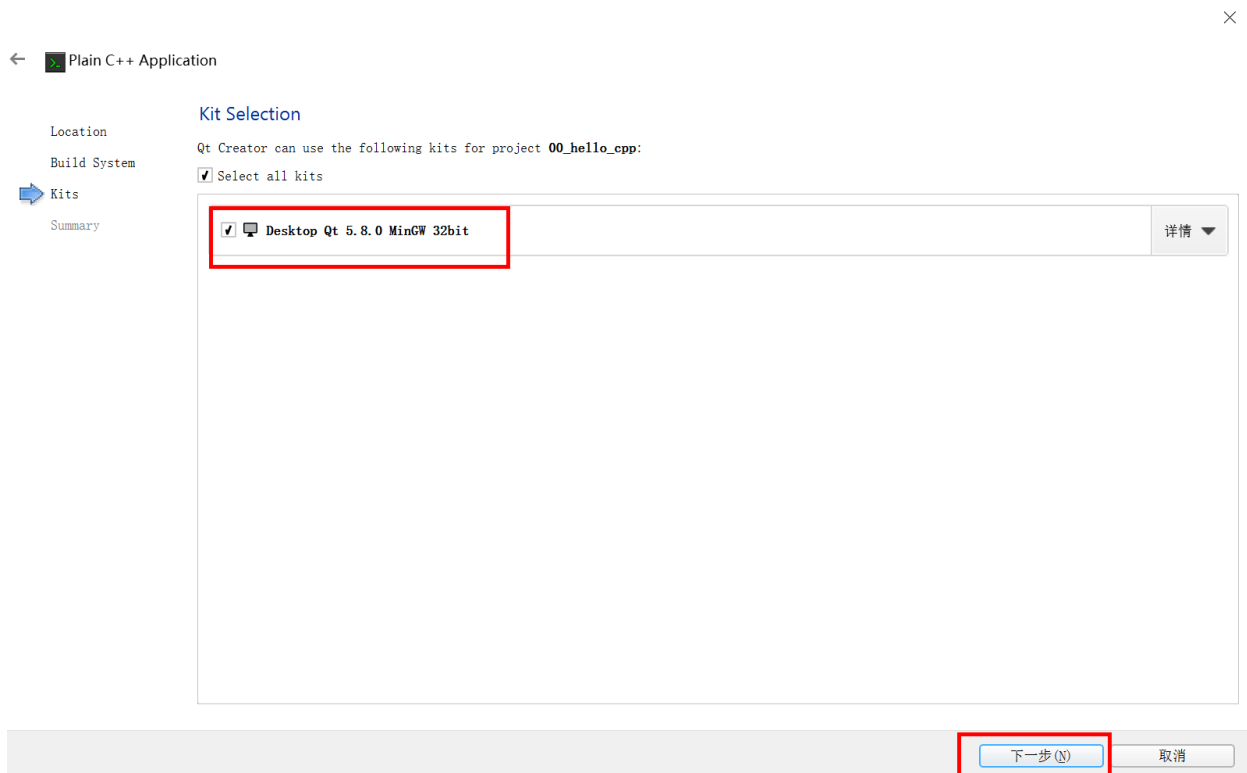
Location

Build System

Kits

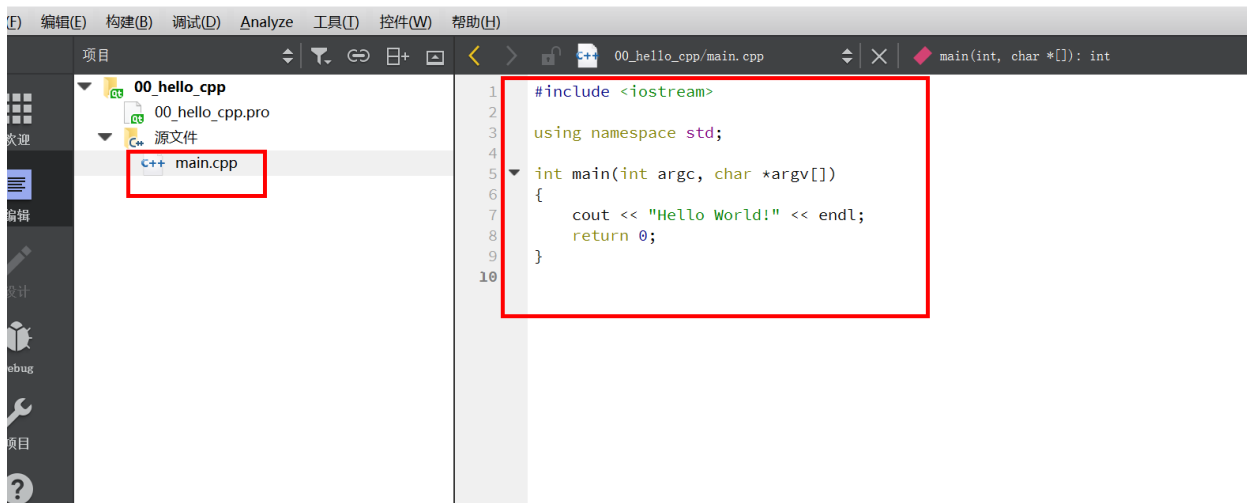
Summary

下一步 (N) 取消

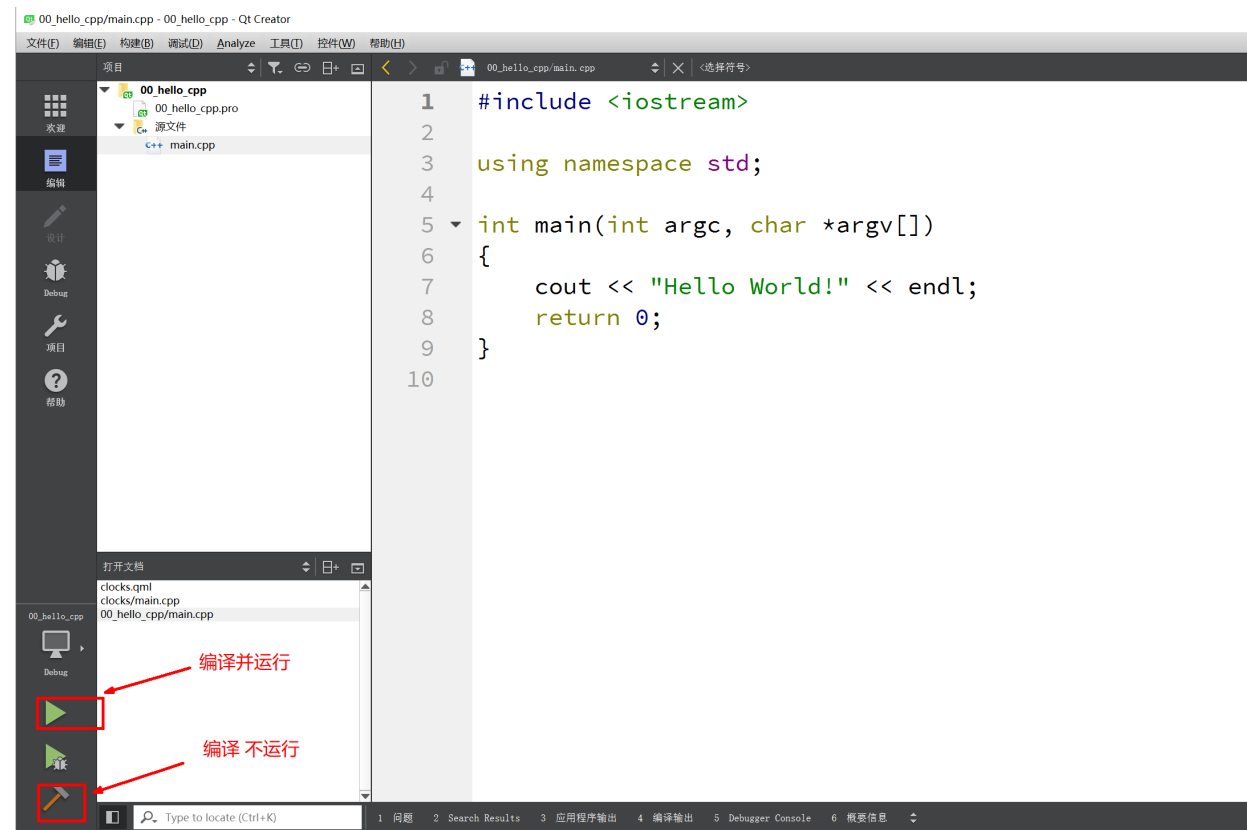


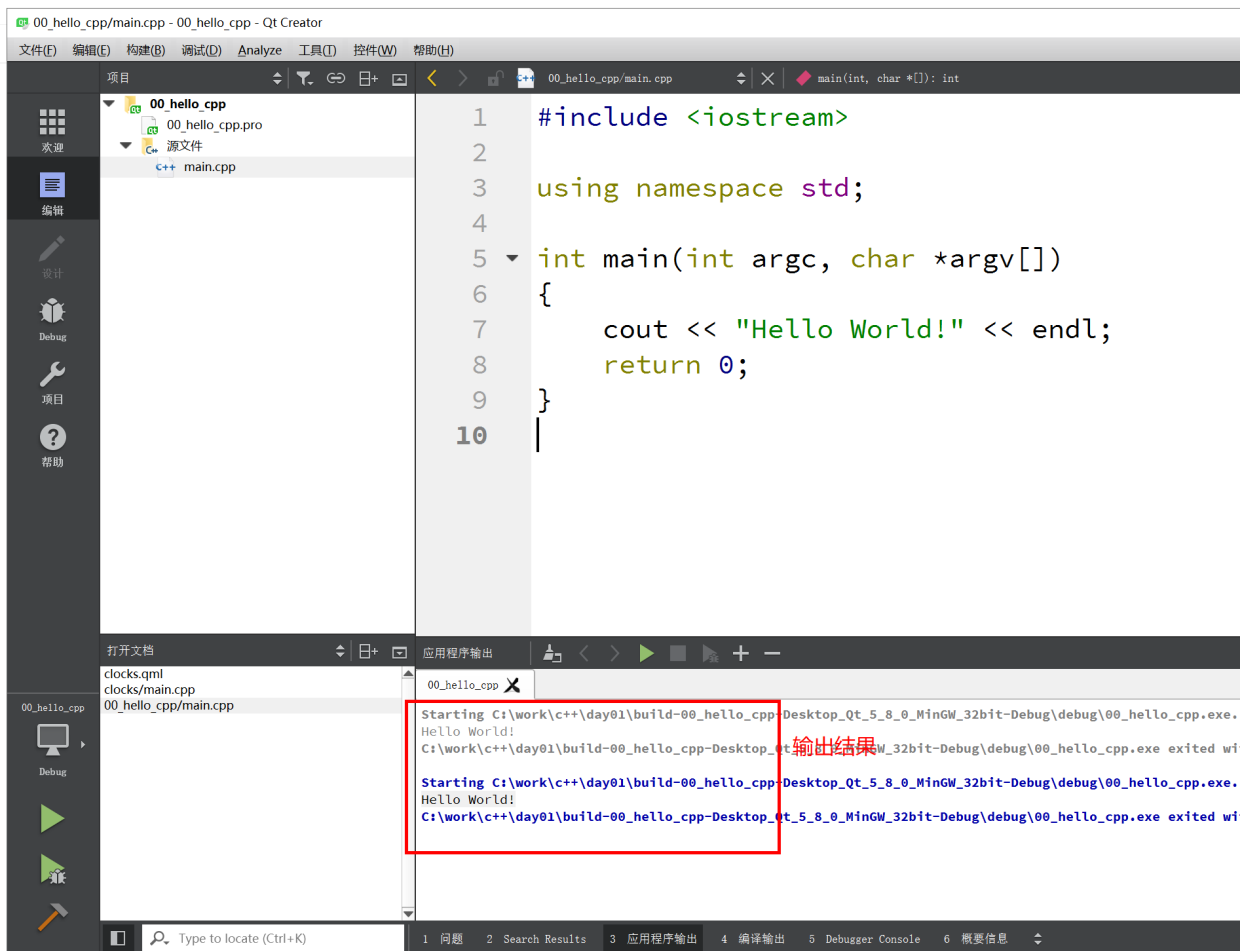
C++工程创建成功



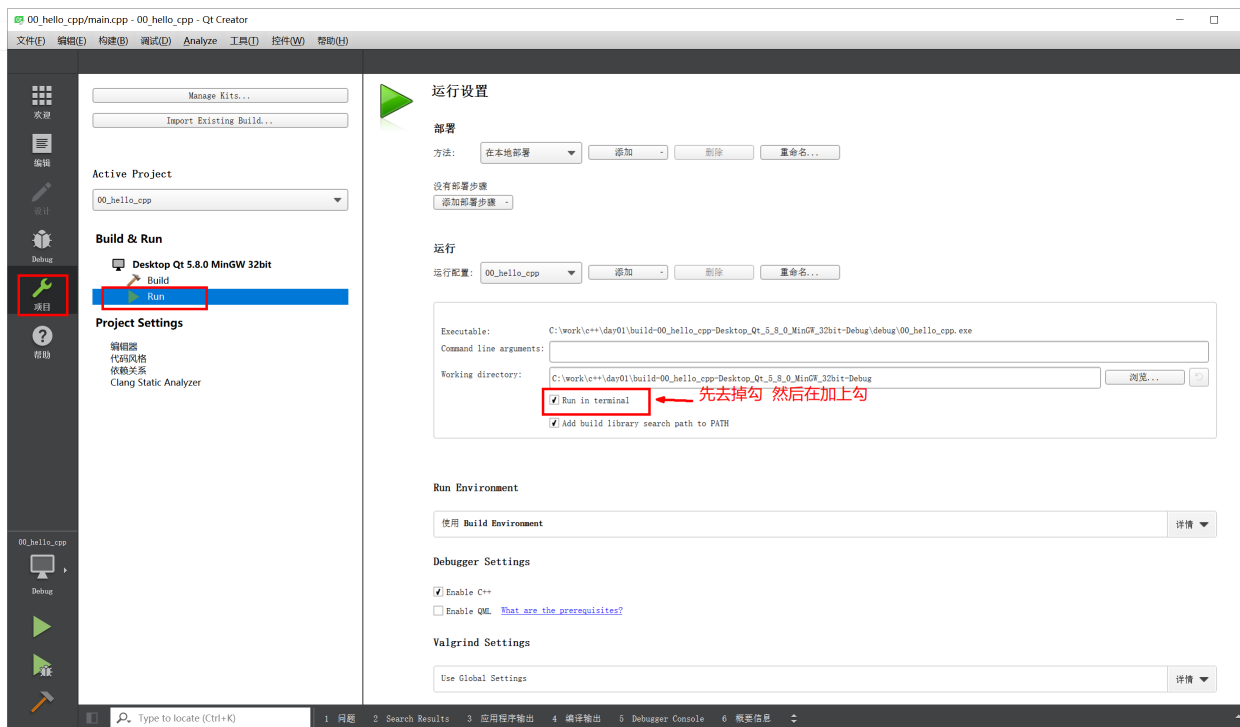


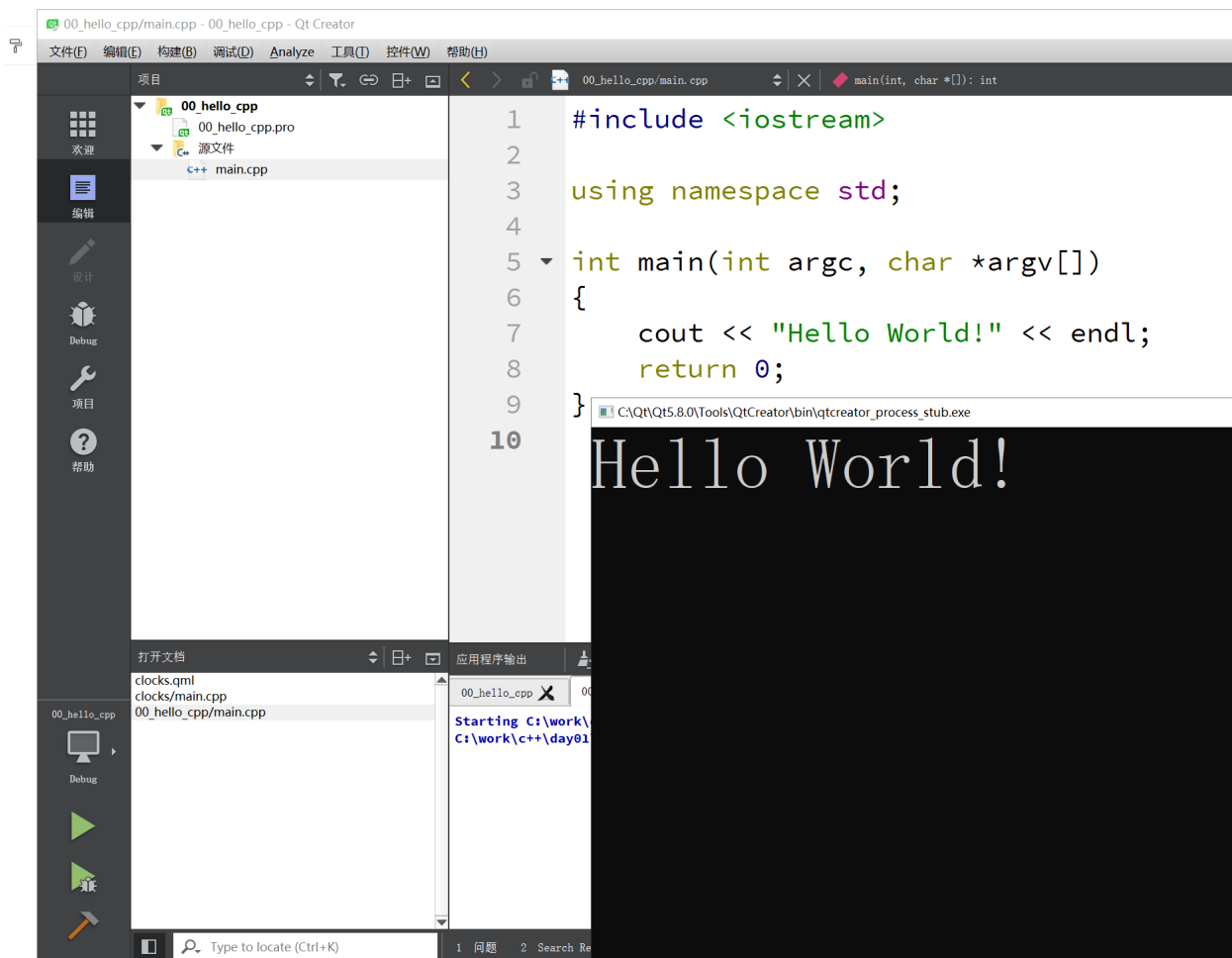
### 知识点3 【qt creator编译c++工程注意项】



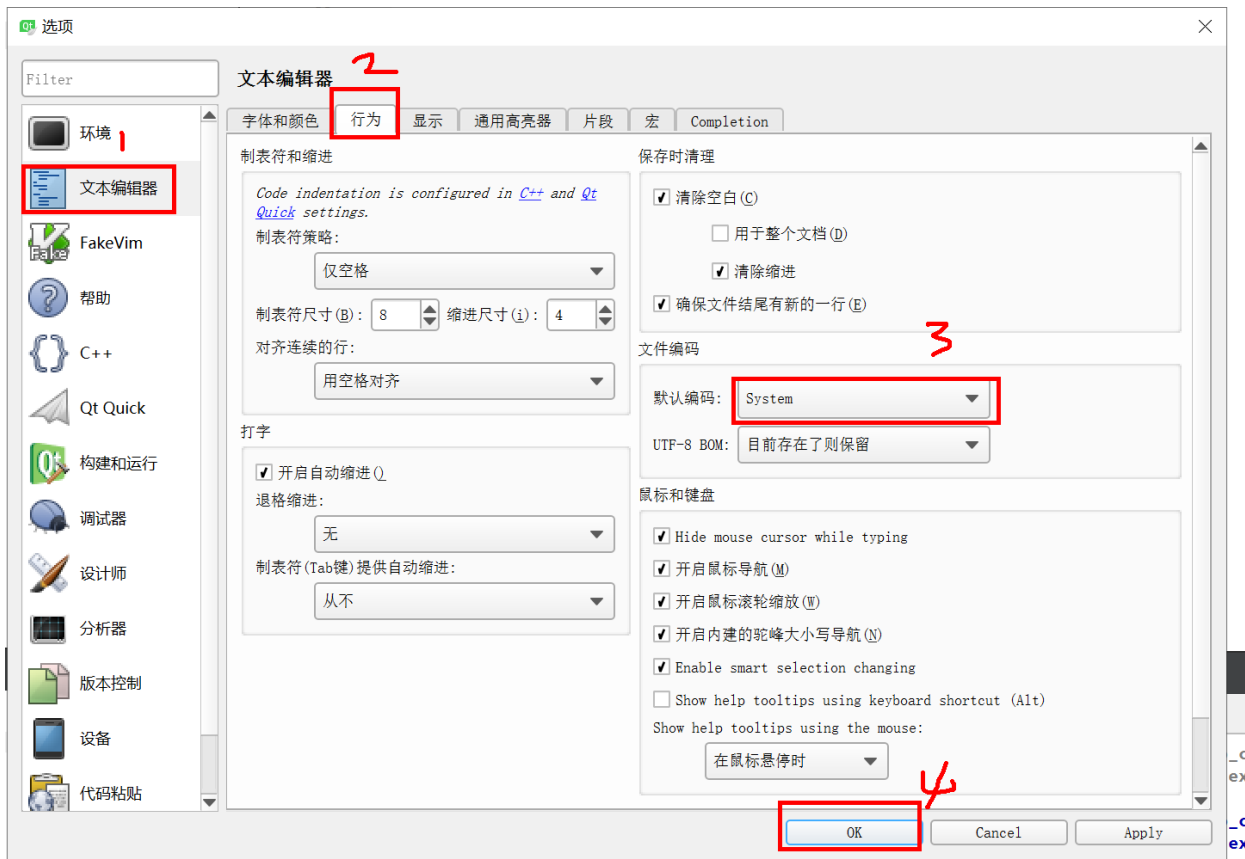
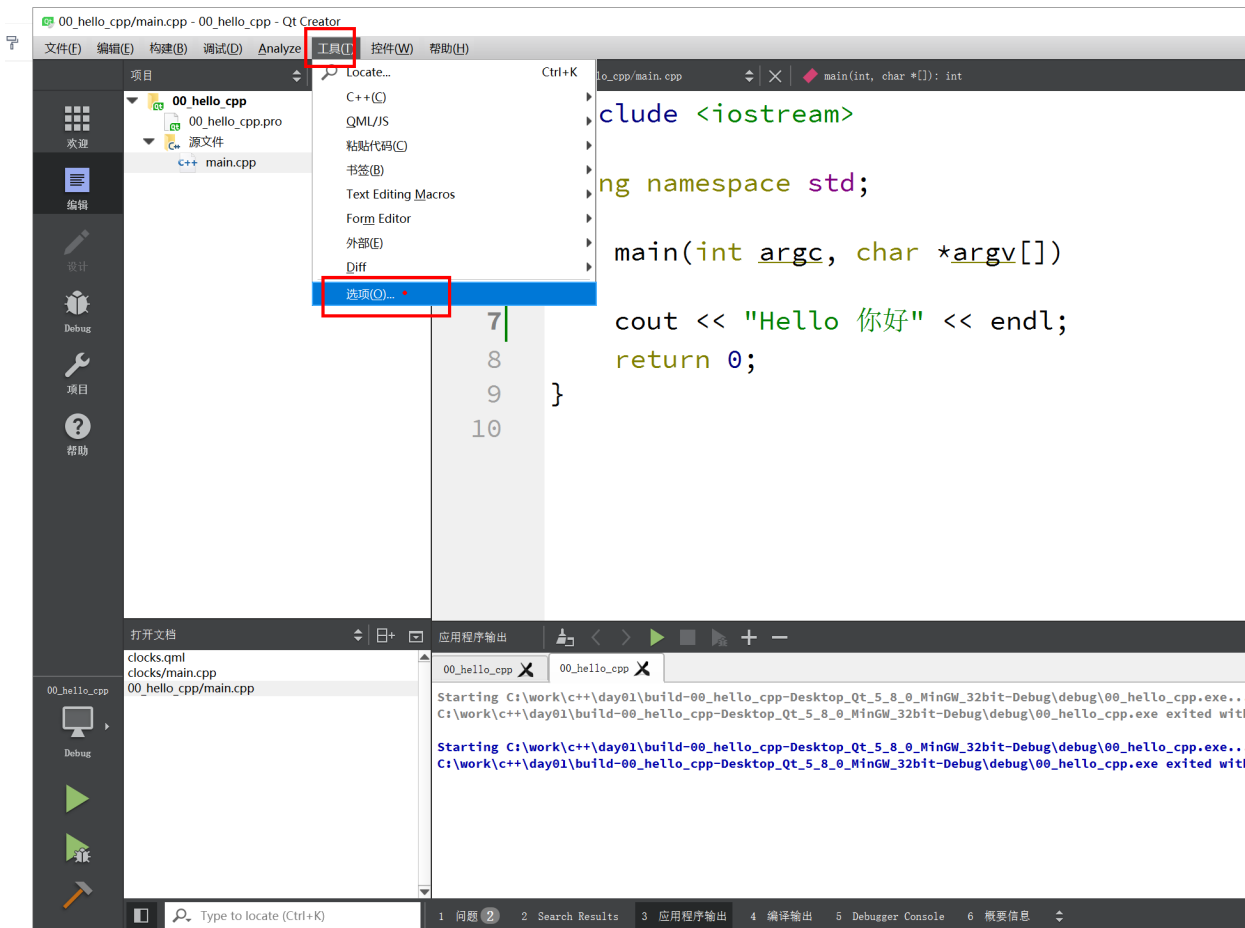


## 1、运行结果 在windows的控制台中 输出。





**2、中文输出是乱码 (minGW == mini gnu for widows) 默认编码格式是linux的UTF8 需要设置系统的编码格式 system**



注意：从新建的工程开始生效 当前工程无效的。

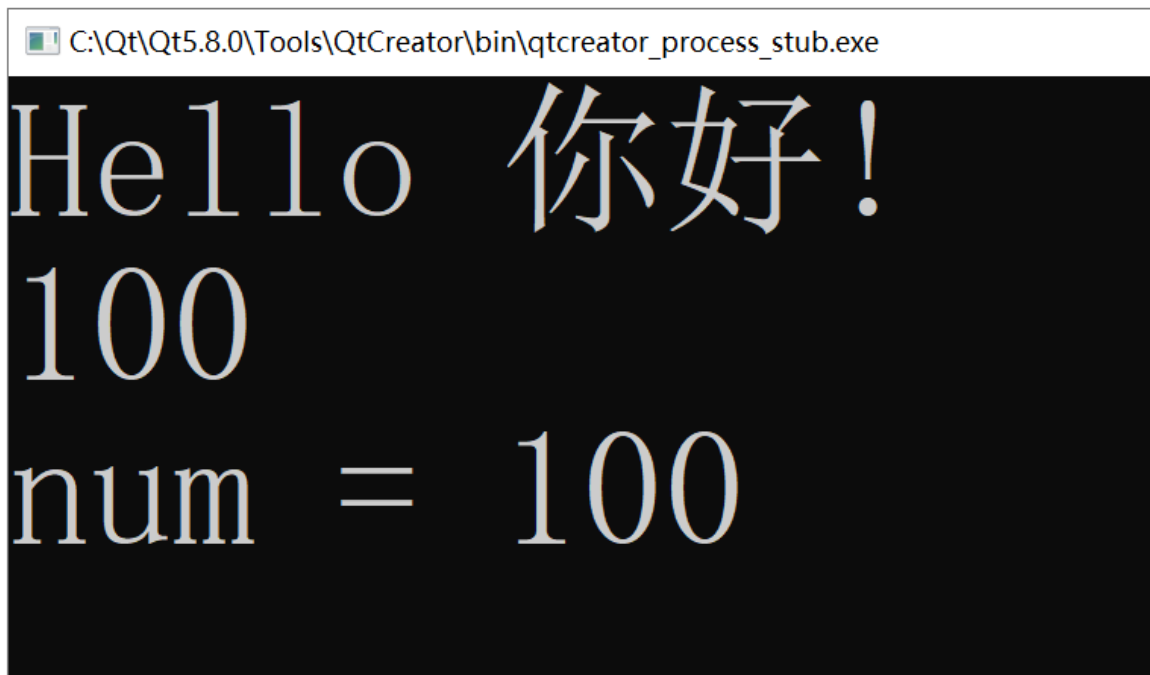
### 3、qtcreator 假死

将电脑的显卡配置成集显

## 知识点4 【c++的第一个程序】

```
1 //i input 输入 o output输出 stream流 输入输出流头文件（类似stdio.h）
2 #include <iostream>
3
4 //std(标准) 使用标准的命名空间
5 using namespace std;
6
7 //有且只有一个主函数 可以有多个其他函数
8 int main(int argc, char *argv[])
9 {
10     //cout 输出 类似 c语言的printf
11     //endl 类似 c语言的 换行符
12     // printf("Hello 你好!\n");
13
14     //cout代表的输出设备
15     cout << "Hello 你好!" << endl;//将字符串输出到 控制台
16
17     //cin代表的是输入设备
18     int num =0;
19     cin >> num;//将建键盘输入的数据 赋值给 num
20     cout<<"num = "<<num<<endl;
21
22     return 0;
23 }
24
```

运行结果：



注意:

- 1、c++系统头文件风格 iostream, c语言的头文件比如string.h 在c++工程 可以写成string.h或cstring
- 2、c语言 是弱语法 语言（很多警告 可以忽略 正常执行），c++强语法语言（很多C语言中的警告 在c++中直接报错）
- 3、c面向过程：面向过程编程思想的核心：功能分解，自顶向下，逐层细化（程序=数据结构+算法）

程序=数据结构+算法

数据结构:就是对数据的存储方式（指的是数据类型：char short int long float struct unions 数组、链表）

算法：就是对存储好的数据 进行分析的**步骤**。（操作数据的步骤 == 功能函数）

- 4、面向对象编程：在面向对象中，算法与数据结构被看做是一个整体，称作对象

对象 = 算法 + 数据结构

程序 = 对象 + 对象 + .....

## 知识点5 【c++的三大特性】（背）

**封装：**把客观的事务抽象成一个类（将数据和方法打包在一起，加以权限的区分，达到保护并安全使用数据的目的）

**继承：**继承所表达的是类之间相关的关系，这种关系使得对象可以继承另外一类对象的特征和能力

目的：避免公用代码的重复开发，减少代码和数据冗余。

多态：多态 多态性可以简单地概括为 “一个接口，多种方法”，字面意思为多种形态。程序在运行时才决定调用的函数，它是面向对象编程领域的核心概念。

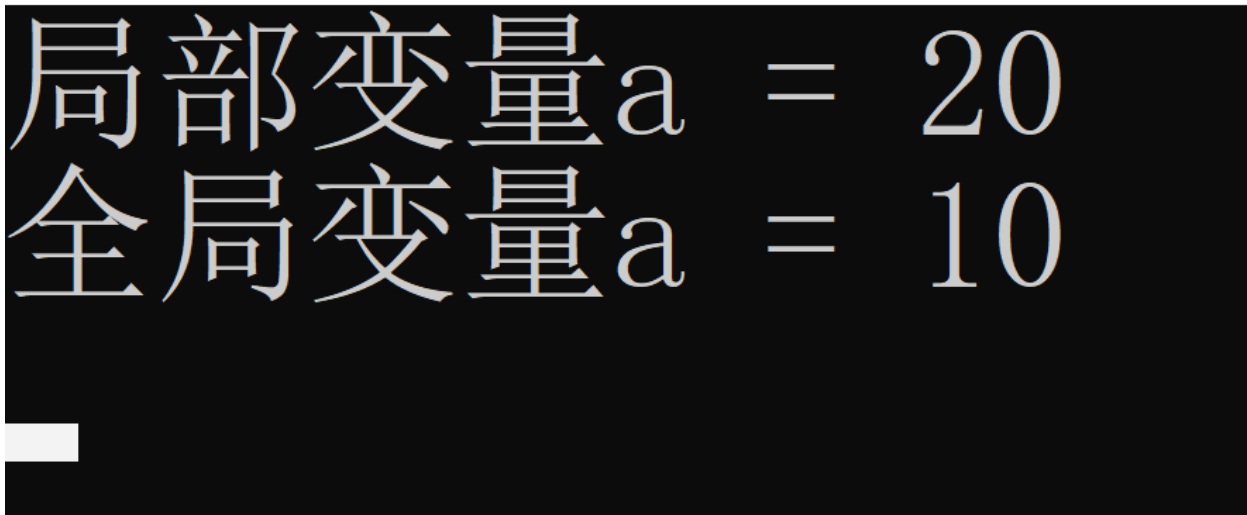
## 知识点6 【c++对c的扩展】

### 1、::作用域运算符（表明 数据、方法 的归属性问题）

```
1 using namespace std;
2 int a = 10; //全局变量
3 void test01()
4 {
5     int a = 20; //局部变量
6     cout<<"局部变量a = "<<a<<endl; //优先选择局部变量
7
8     //::作用域运算符（c++独有）
9     cout<<"全局变量a = "<<::a<<endl; //取全局变量
10 }
```

运行结果：

 C:\Qt\Qt5.8.0\tools\QtCreator\bin\qtcreator\_process\_stub.exe



局部变量a = 20  
全局变量a = 10

## 2、命名空间 namespace 解决命名冲突

### 2.1: namespace命名空间的定义

```
1 //定义一个名字为A的命名空间（变量、函数）
2 namespace A {
3     int a = 100;
4 }
5 namespace B {
6     int a = 200;
```

```

7 }
8 void test02()
9 {
10  //A::a a是属于A中
11  cout<<"A中a = "<<A::a<<endl;//100
12  cout<<"B中a = "<<B::a<<endl;//200
13 }

```

## 2.2:命名空间只能全局范围内定义 (以下错误写法)

局部定义命名空间 是错误的

```

void test(){
    namespace A{
        int a = 10;
    }
    namespace B{
        int a = 20;
    }
    cout << "A::a : " << A::a << endl;
    cout << "B::a : " << B::a << endl;
}

```

## 2.3:命名空间可嵌套命名空间

```

1 namespace A {
2   int a = 1000;
3   namespace B {
4     int a = 2000;
5   }
6 }
7 void test03()
8 {
9   cout<<"A中的a = "<<A::a<<endl; //1000
10  cout<<"B中的a = "<<A::B::a<<endl; //2000
11 }

```

## 2.4:命名空间是开放的, 即可以随时把新的成员加入已有的命名空间中(常用)

```

1 namespace A {
2   int a = 100;
3   int b = 200;
4 }
5 //将c添加到已有的命名空间A中
6 namespace A {

```



```

7   int c = 300;
8   }
9   void test04()
10  {
11      cout<<"A中a = "<<A::a<<endl;//100
12      cout<<"A中c = "<<A::c<<endl;//200
13  }

```

## 2.5: 命名空间 可以存放 变量 和 函数

```

1   namespace A {
2       int a=100;//变量
3
4       void func();//函数
5       {
6           cout<<"func遍历a = "<<a<<endl;
7       }
8   }
9   void test05()
10  {
11      //变量的使用
12      cout<<"A中的a = "<<A::a<<endl;
13
14      //函数的使用
15      A::func();
16  }

```

## 2.6: 命名空间中的函数 可以在“命名空间”外 定义

```

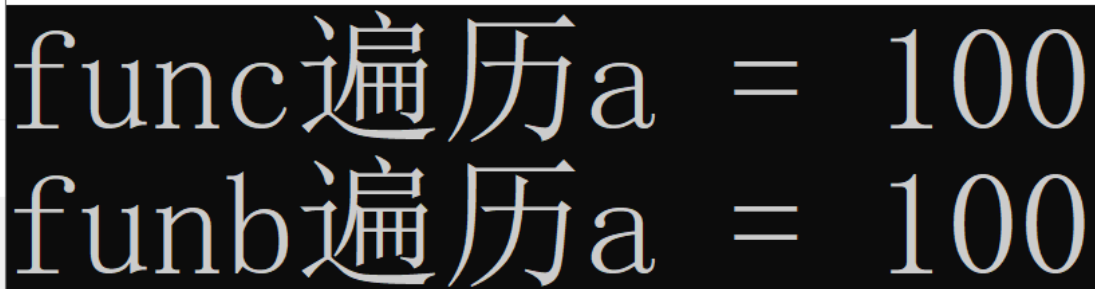
1   namespace A {
2       int a=100;//变量
3
4       void func();
5   }
6
7   void A::func();//成员函数 在外部定义的时候 记得加作用域
8   {
9       //访问命名空间的数据不用加作用域
10      cout<<"func遍历a = "<<a<<endl;
11  }
12
13  void funb();//普通函数
14  {
15      cout<<"funb遍历a = "<<A::a<<endl;

```

```
16 }  
17 void test06()  
18 {  
19     A::func();  
20     funb();  
21 }
```

运行结果：

C:\Qt\Qt5.8.0\Tools\QtCreator\bin\qtcreator\_process\_stub.exe



```
func遍历a = 100  
funb遍历a = 100
```

2.7: **无名命名空间**，意味着命名空间中的标识符**只能在本文文件内访问**，相当于给这个标识符加上了**static**，使得其可以作为内部连接（了解）

```
namespace{

int a = 10;
void func(){ cout << "hello namespace" << endl; }

}
void test(){
```

双击编辑页脚

```
    cout << "a : " << a << endl;
    func();
}
```

## 2.8: 给命名空间 取个别名 (了解)

```
1 namespace veryLongName{
2
3 int a = 10;
4 void func(){ cout << "hello namespace" << endl; }
5
6 }
7
8 void test(){
9     namespace shortName = veryLongName;
10    cout << "veryLongName::a : " << shortName::a << endl;
11    veryLongName::func();
12    shortName::func();
13 }
```

## 知识点7【using 使用命名空间】

### 1、简化了从命名空间的成员访问

```
1 namespace veryLongName {
2     int a=100;
3     void func(){cout<<"hello namespace"<<endl;}
```

```

4 }
5 void test07()
6 {
7
8 //使用veryLongName命名空间
9 using namespace veryLongName;
10
11 //出现的变量 从veryLongName命名空间中找 找不到 从其他地方中
12 cout<<"a = "<<a<<endl;
13 func();
14 }

```

## 2、using 使用整个命名空间

```

1 namespace veryLongName {
2     int a=100;
3     void func(){cout<<"hello namespace"<<endl;}
4 }
5 void test07()
6 {
7     int a=200;
8     //使用veryLongName命名空间
9     using namespace veryLongName;
10
11 //出现的变量 从veryLongName命名空间中找 找不到 从其他地方中
12 cout<<"a = "<<a<<endl;//访问的是局部变量中的a
13 cout<<"a = "<<veryLongName::a<<endl;//访问的是veryLongName的a
14 func();
15 }

```

## 3、using 指明使用**具体的命名空间 成员**。（了解）

using直接使用 命名空间中的成员 会和 局部变量冲突

```

101 namespace veryLongName {
102     int a=100;
103     void func(){cout<<"hello namespace"<<endl;}
104 }
105
106 void test07()
107 {
108     //using直接使用 命名空间中的成员 会和 局部变量冲突
109     int a = 200;
110     //指明 使用命名空间中的具体成员 容易和其他变量冲突
111     using veryLongName::a; //err
112
113     cout<<"a = "<<a<<endl;
114
115     //但是func使用的时候 必须加作用域
116     veryLongName::func();
117 }

```

using直接使用 命名空间中的成员 不会和 全局变量冲突

```

1 namespace veryLongName {
2     int a=100;
3     void func(){cout<<"hello namespace"<<endl;}
4 }
5 int a = 200;
6 void test07()
7 {
8     //using直接使用 命名空间中的成员 不会和 全局变量冲突
9     using veryLongName::a;
10
11     cout<<"命名空间中a = "<<a<<endl; //命名空间中的成员 100
12     cout<<"全局变量中a = "<<::a<<endl; //200
13
14     //但是func使用的时候 必须加作用域
15     veryLongName::func();
16 }

```

#### 4、using声明碰到函数重载（了解）

```

1 namespace A {
2     //函数重载 函数名+参数 组合代表是函数的入口地址
3     void func(){cout<<" 无参的func"<<endl;}
4     void func(int a){cout<<" int的func"<<endl;}
5     void func(int a,int b){cout<<" int int的func"<<endl;}
6 }
7


```

```

8 void test08()
9 {
10 //using指明 使用 A中的func 会对 所有的func起作用
11 using A::func;
12 func();
13 func(10);
14 func(10,20);
15 }

```

运行结果:

 C:\Qt\Qt5.8.0\Tools\QtCreator\bin\qtcreator\_process\_stub.exe

无参的func  
int的func  
int int的func

## 5、不同命名空间中的 同名成员 使用的时候注意 二义性

```

1 namespace A {
2   int a = 10;
3 }
4 namespace B {
5   int a = 20;
6 }
7 void test09()
8 {
9   //此处的a 不知道是A还是B中a
10   //cout<<"a = "<<a<<endl;//err
11
12   //解决方法
13   cout<<"A::a = "<<A::a<<endl;//100
14   cout<<"B::a = "<<B::a<<endl;//200
15 }

```

总结：

```
1 namespace A
2 {
3     int a = 10;
4     void func(){cout<<"func"<<endl;}
5 }
```

- 1、命名空间的定义（不能在函数内定义命名空间）
- 2、使用命名空间的成员 最安全的方式 命名空间名::成员名
- 3、`using namespace 命名空间名`；使用整个命名空间（重要）

```
1 using namespace A;
```

- 4、单独使用命名空间中的具体成员：`using 命名空间名::成员名`；

```
1 using A::a;
```

- 5、说明一下main中的std

```
1 #include <iostream>
2 //使用标准的命名空间std
3 //std中所有成员名 可以直接使用
4 //cout endl cin都是命名空间std的成员
5 using namespace std;
6
7 int main(int argc, char *argv[])
8 {
9     std::cout << "Hello World!" << std::endl;
10    cout << "Hello World!" << endl;
11    return 0;
12 }
```