

Qt –視窗影像處理應用程式

Yih-Chuan Lin

CSIE Windows Programming Class

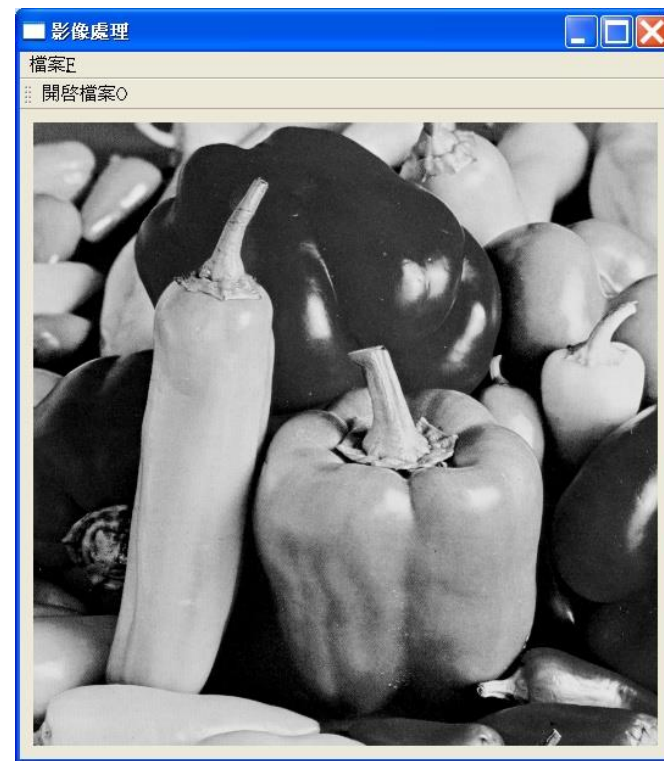
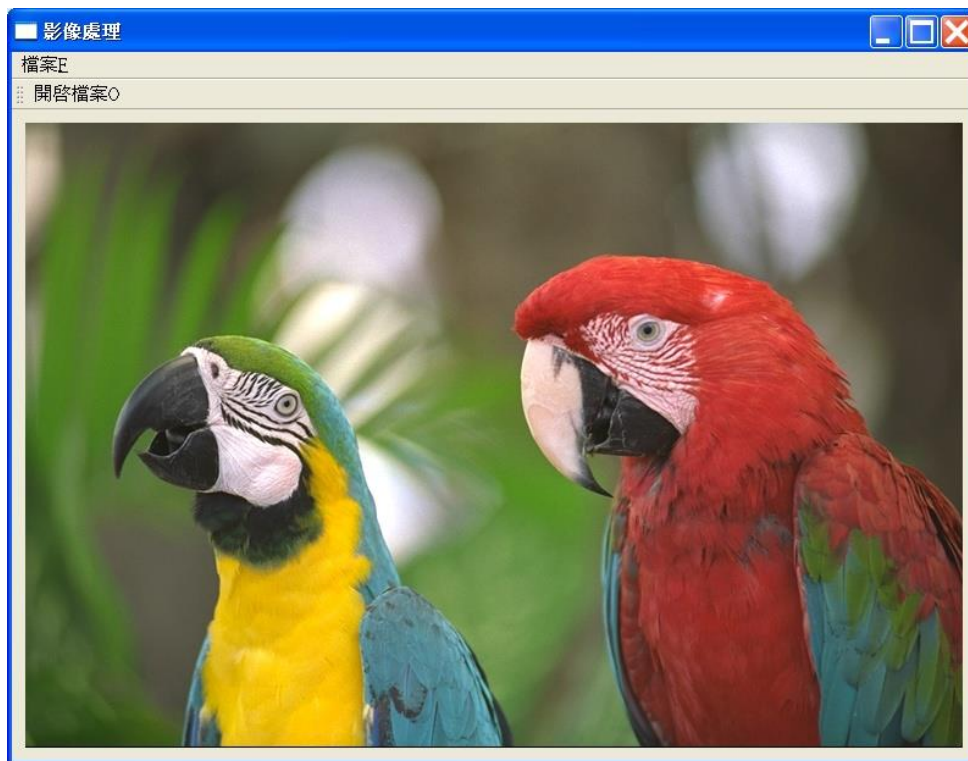
National Formosa University

Image Processor

- 學習目的
 - 熟練Qt MainWindow之視窗程式開發
 - 練習Qt QImage、及QPixmap類別製作影像處理視窗程式
 - 學習如何以程式碼建立視窗人機介面

Image Processor

以Qt 主視窗框架實作 視窗影像處理應用程式



相關類別：

QMainWindow, QMenuBar, QToolBar, QImage, QPixmap.

Image Processor

- Create a mainwindow application:

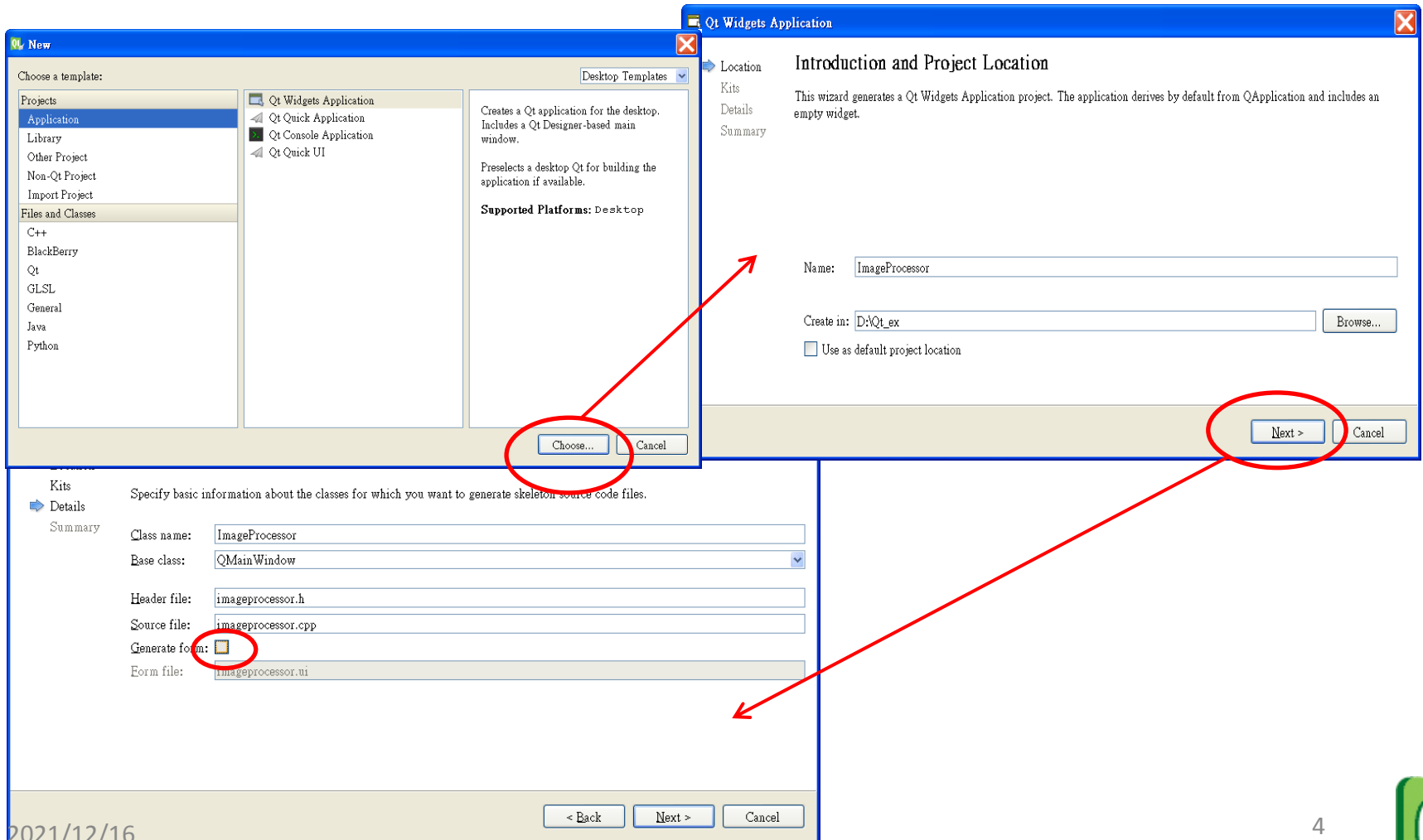
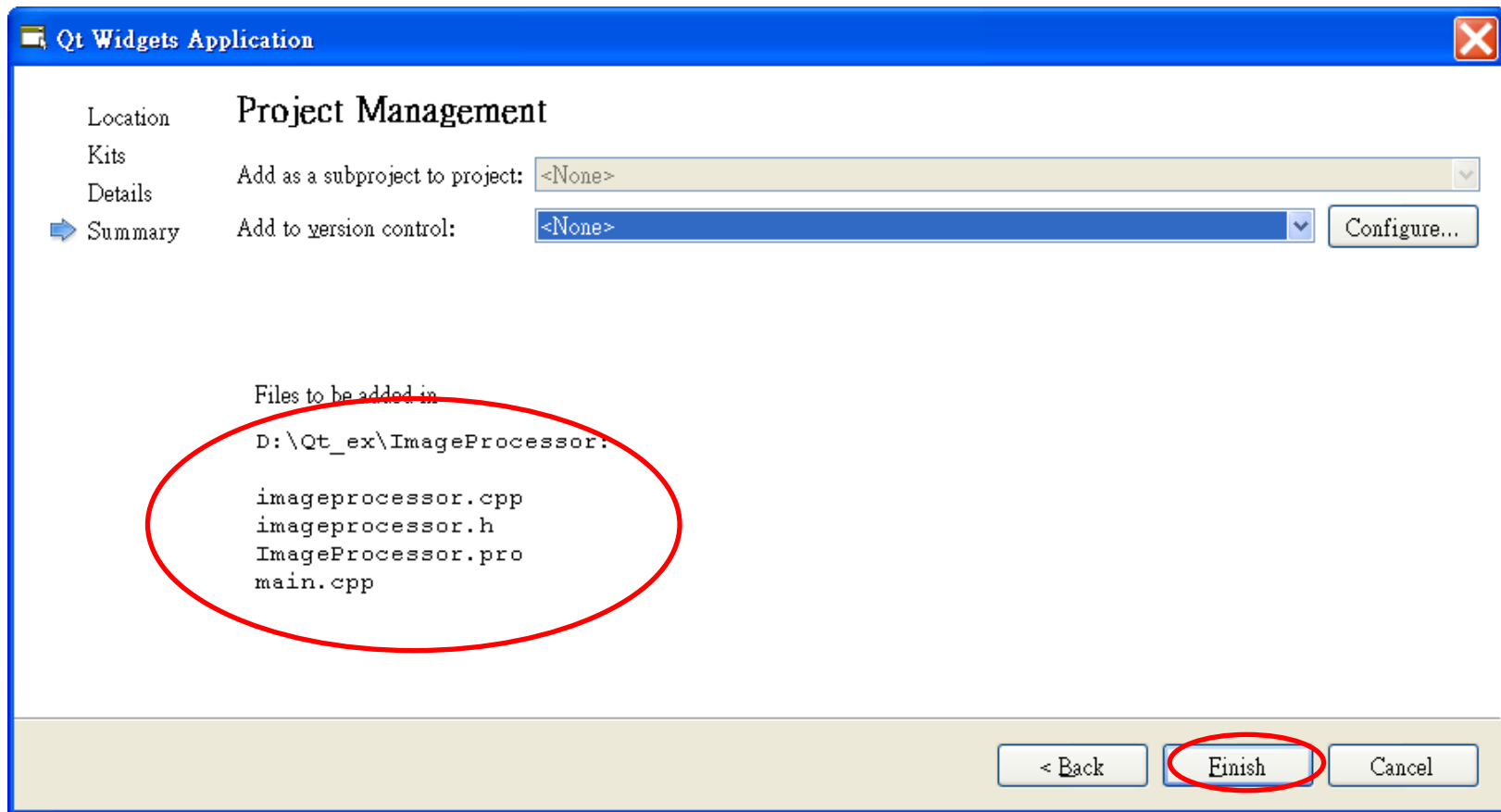




Image Processor

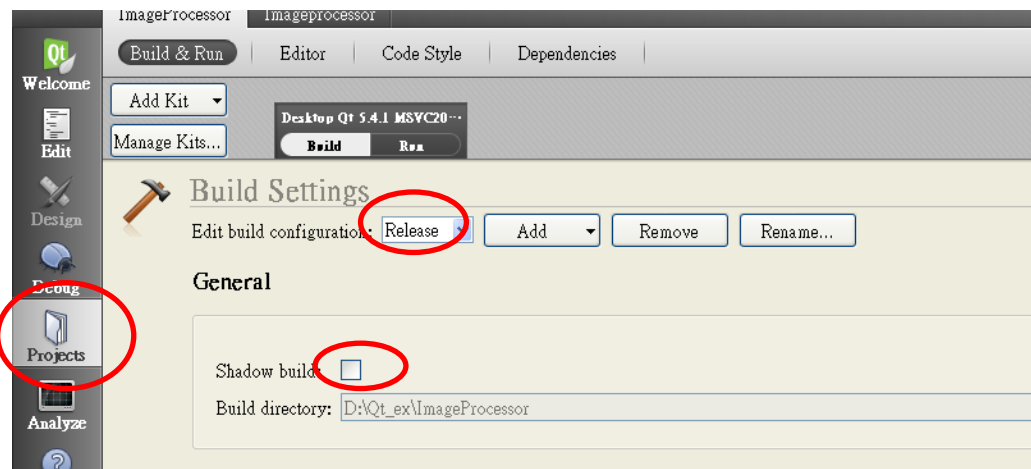
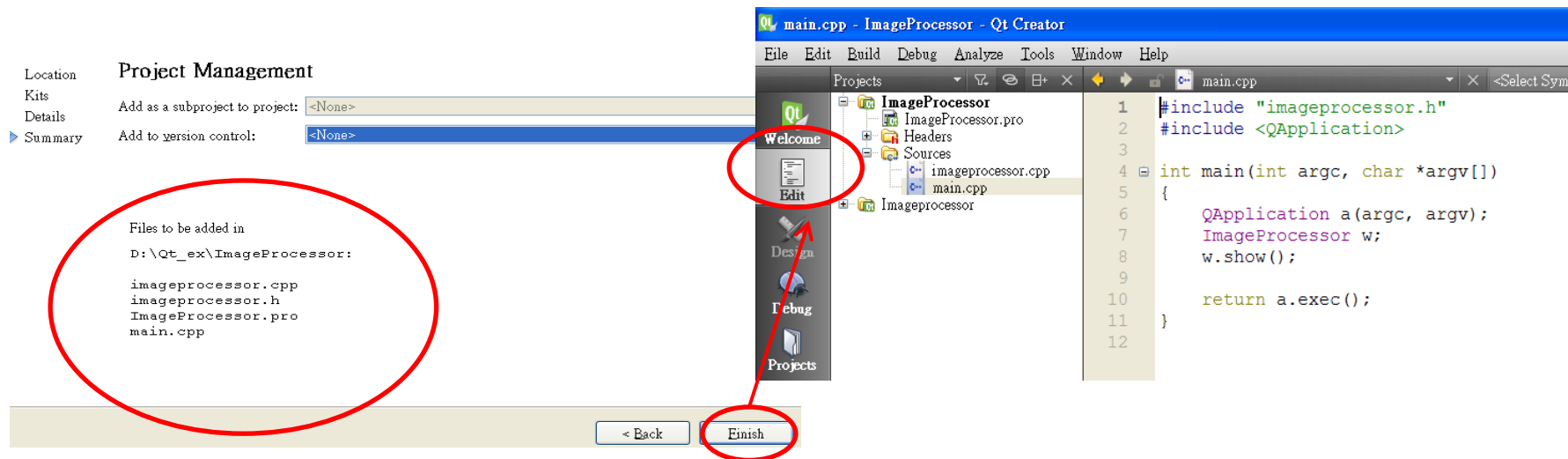
- Create a mainwindow application:



Qt- MainWindow

● Create a mainwindow application:

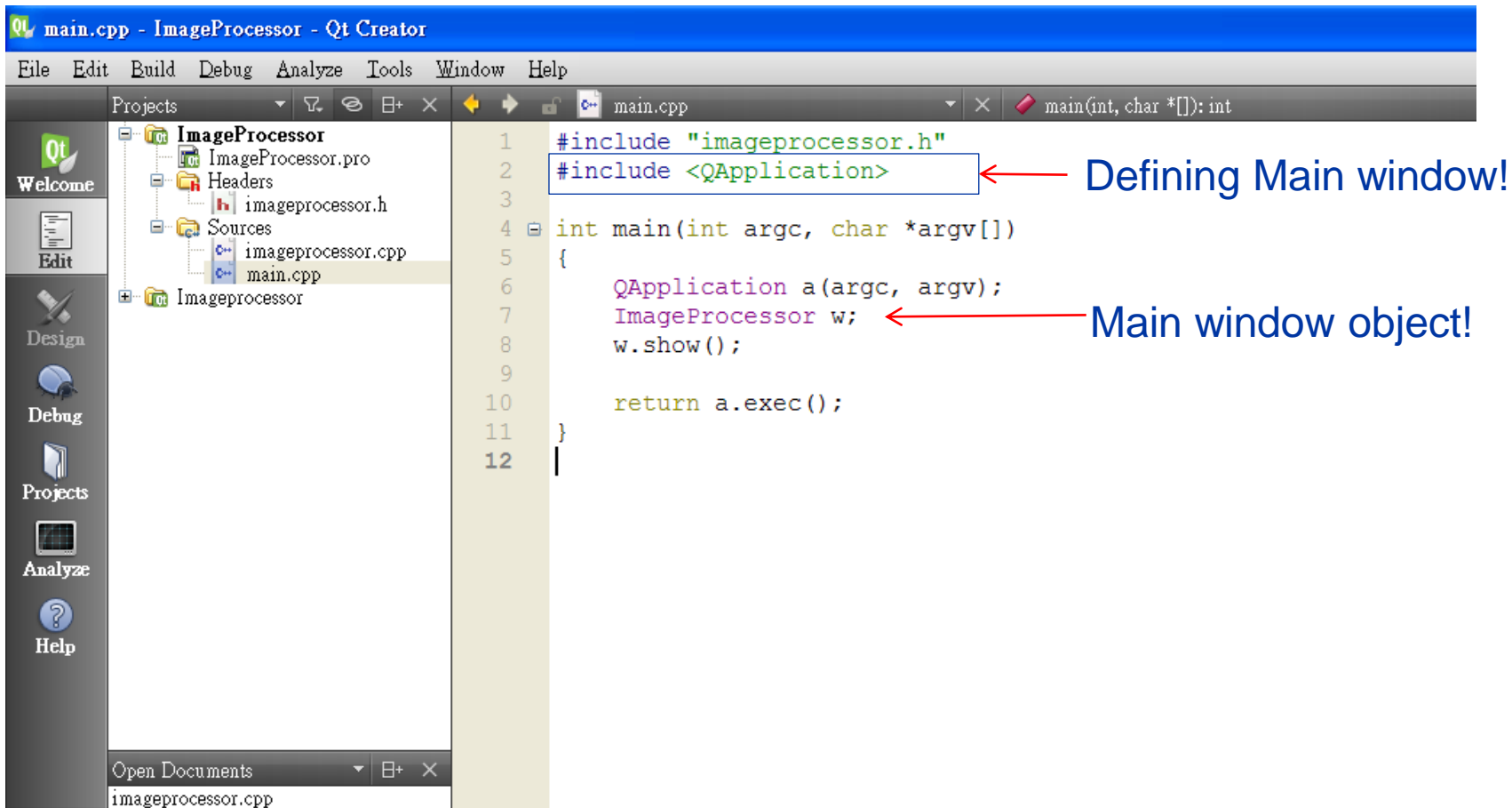
檢視程式進入點



取消分離建置
(Shadow build)

Image Processor

- Check out the main function:



```
main.cpp - ImageProcessor - Qt Creator
File Edit Build Debug Analyze Tools Window Help

Projects
  ImageProcessor
    ImageProcessor.pro
    Headers
      imageprocessor.h
    Sources
      imageprocessor.cpp
      main.cpp
  Imageprocessor

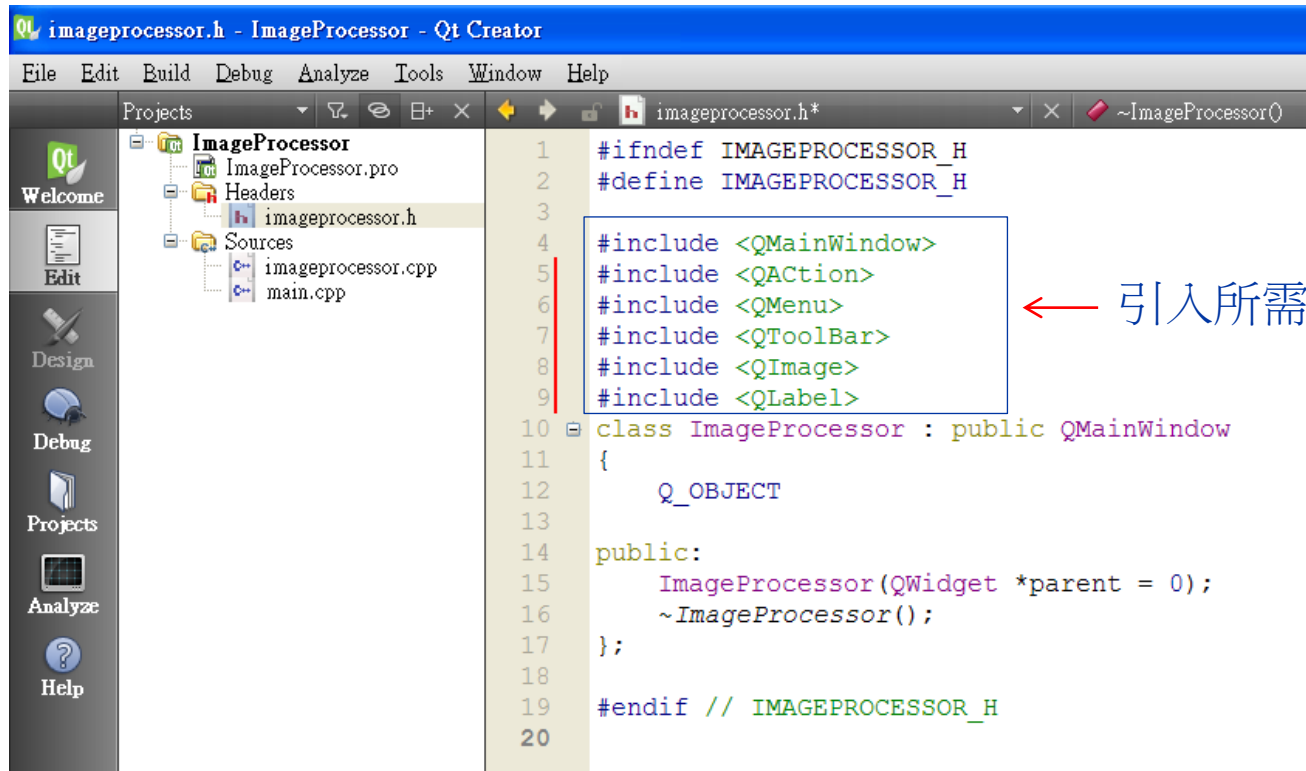
1  #include "imageprocessor.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      ImageProcessor w;
8      w.show();
9
10     return a.exec();
11 }
12
```

Defining Main window!

Main window object!

Image Processor

- Add ImageProcessor's members:



```
1  #ifndef IMAGEPROCESSOR_H
2  #define IMAGEPROCESSOR_H
3
4  #include <QMainWindow>
5  #include <QAction>
6  #include <QMenu>
7  #include <QToolBar>
8  #include <QImage>
9  #include <QLabel>
10 class ImageProcessor : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     ImageProcessor(QWidget *parent = 0);
16     ~ImageProcessor();
17 };
18
19 #endif // IMAGEPROCESSOR_H
20
```

← 引入所需之Qt類別標頭檔

Image Processor

● Add ImageProcessor's members:



The screenshot shows the Qt Creator IDE with the 'ImageProcessor' project open. The left sidebar displays the 'Welcome' screen with options like 'Edit', 'Design', 'Debug', 'Projects', 'Analyze', and 'Help'. The main editor area shows the 'Sources' view with files: 'ImageProcessor.pro', 'Headers', 'imageprocessor.h', 'Sources', 'imageprocessor.cpp', and 'main.cpp'. The 'Open Documents' list at the bottom shows 'imageprocessor.cpp', 'imageprocessor.h*', and 'ImageProcessor.pro'.

The right pane displays the code in 'imageprocessor.cpp'. The code defines the 'ImageProcessor' class, which inherits from 'QMainWindow'. The class is annotated with 'Q_OBJECT' and '宣告Qt類別成員函數' (Declare Qt class member functions). The public methods are: 'ImageProcessor(QWidget *parent = 0);', '~ImageProcessor();', 'void createActions();', 'void createMenus();', 'void createToolBars();', and 'void loadFile(QString filename);'. The private slots section contains 'void showOpenFile();' which is annotated with '宣告Qt類別槽函數' (Declare Qt class slot function). The code is enclosed in a preprocessor guard: '#ifndef IMAGEPROCESSOR_H' and '#define IMAGEPROCESSOR_H'.

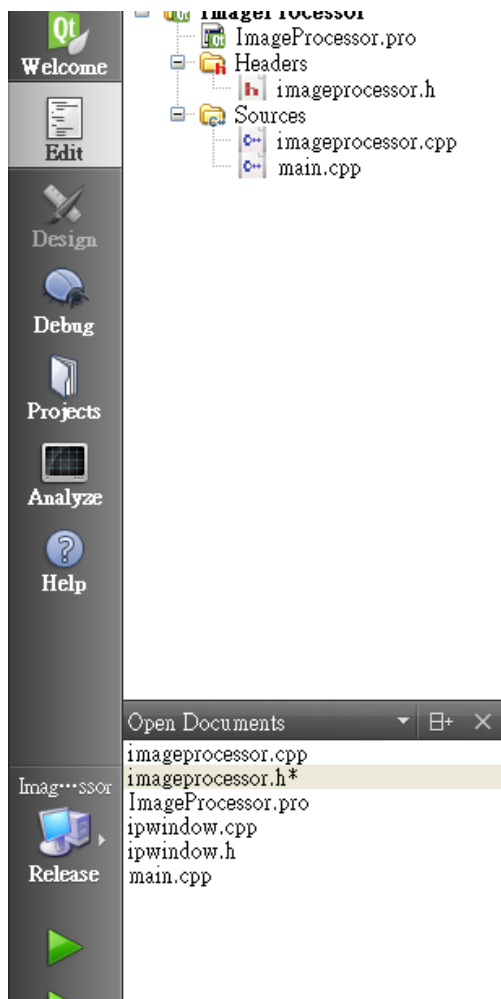
```

1  #ifndef IMAGEPROCESSOR_H
2  #define IMAGEPROCESSOR_H
3
4  #include <QMainWindow>
5  #include <QAction>
6  #include <QMenu>
7  #include <QToolBar>
8  #include <QImage>
9  #include <QLabel>
10 class ImageProcessor : public QMainWindow
11 {
12     Q_OBJECT 宣告Qt類別成員函數
13
14 public:
15     ImageProcessor(QWidget *parent = 0);
16     ~ImageProcessor();
17     void createActions();
18     void createMenus();
19     void createToolBars();
20     void loadFile(QString filename);
21 private slots:
22     void showOpenFile(); 宣告Qt類別槽函數
23 };
24
25 #endif // IMAGEPROCESSOR_H

```

Image Processor

● Add ImageProcessor's members:



```

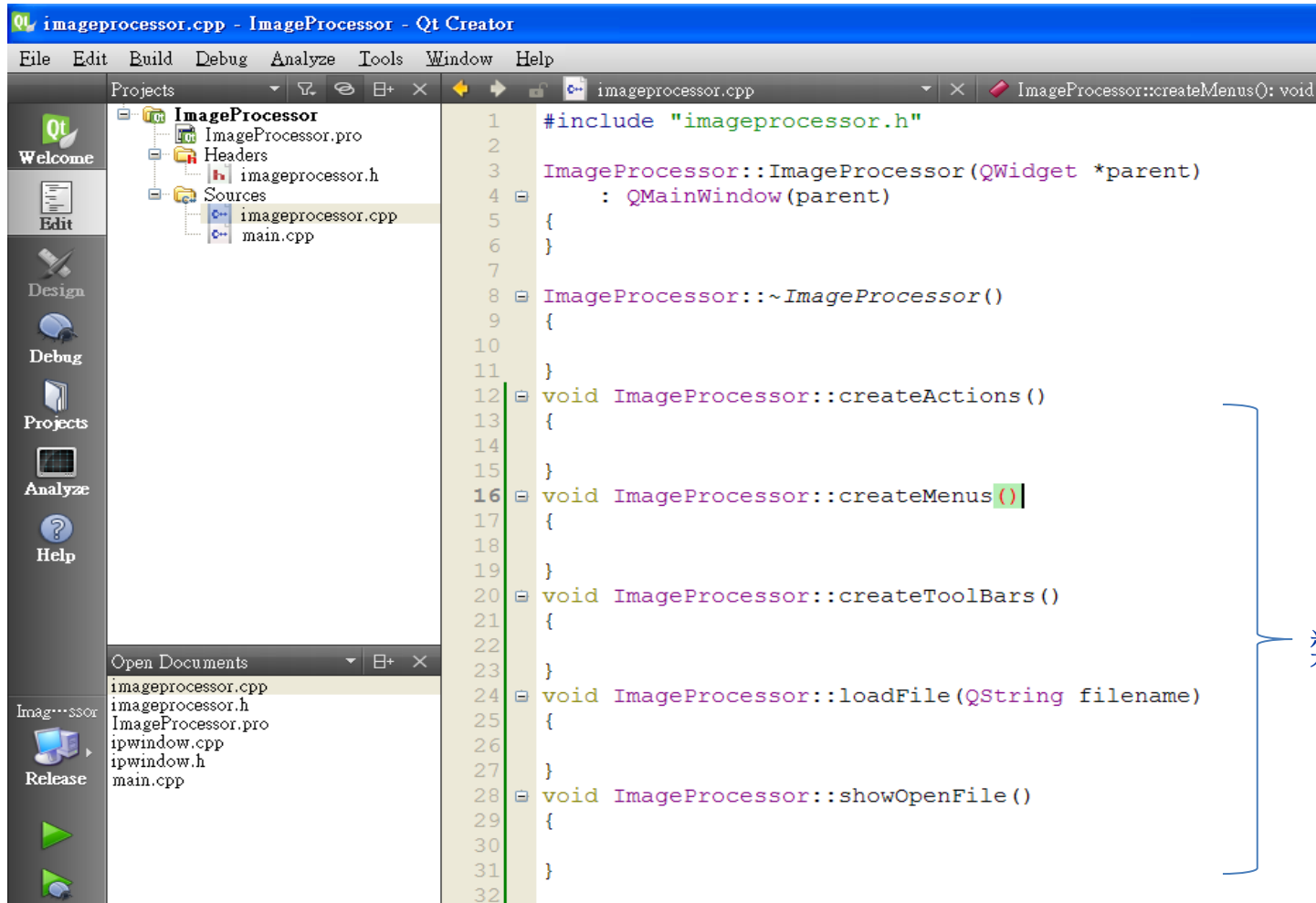
5  #include <QAction>
6  #include <QMenu>
7  #include <QToolBar>
8  #include <QImage>
9  #include <QLabel>
10 class ImageProcessor : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     ImageProcessor(QWidget *parent = 0);
16     ~ImageProcessor();
17     void createActions();
18     void createMenus();
19     void createToolBars();
20     void loadFile(QString filename);
21 private slots:
22     void showOpenFile();
23
24 private:
25     QWidget *central;
26     QMenu *fileMenu;
27     QToolBar *fileTool;
28     QImage img;
29     QString filename;
30     QLabel *imgWin;
31     QAction *openFileAction;
32     QAction *exitAction;
33 };
34
35 #endif // IMAGEPROCESSOR_H

```

宣告Qt類別成員變數

Image Processor

● Implement ImageProcessor's member functions:



The screenshot shows the Qt Creator IDE with the project 'ImageProcessor' open. The file 'imageprocessor.cpp' is selected in the 'Sources' pane. The code editor displays the following C++ code:

```

1  #include "imageprocessor.h"
2
3  ImageProcessor::ImageProcessor(QWidget *parent)
4      : QMainWindow(parent)
5  {
6  }
7
8  ImageProcessor::~ImageProcessor()
9  {
10 }
11
12 void ImageProcessor::createActions()
13 {
14 }
15
16 void ImageProcessor::createMenus()
17 {
18 }
19
20 void ImageProcessor::createToolBars()
21 {
22 }
23
24 void ImageProcessor::loadFile(QString filename)
25 {
26 }
27
28 void ImageProcessor::showOpenFile()
29 {
30 }
31
32

```

A blue bracket on the right side of the code editor groups the functions from line 12 to line 31, labeled with the text '類別成員函數' (Class Member Functions).

類別成員函數

Image Processor

- **Build and execute the program:**

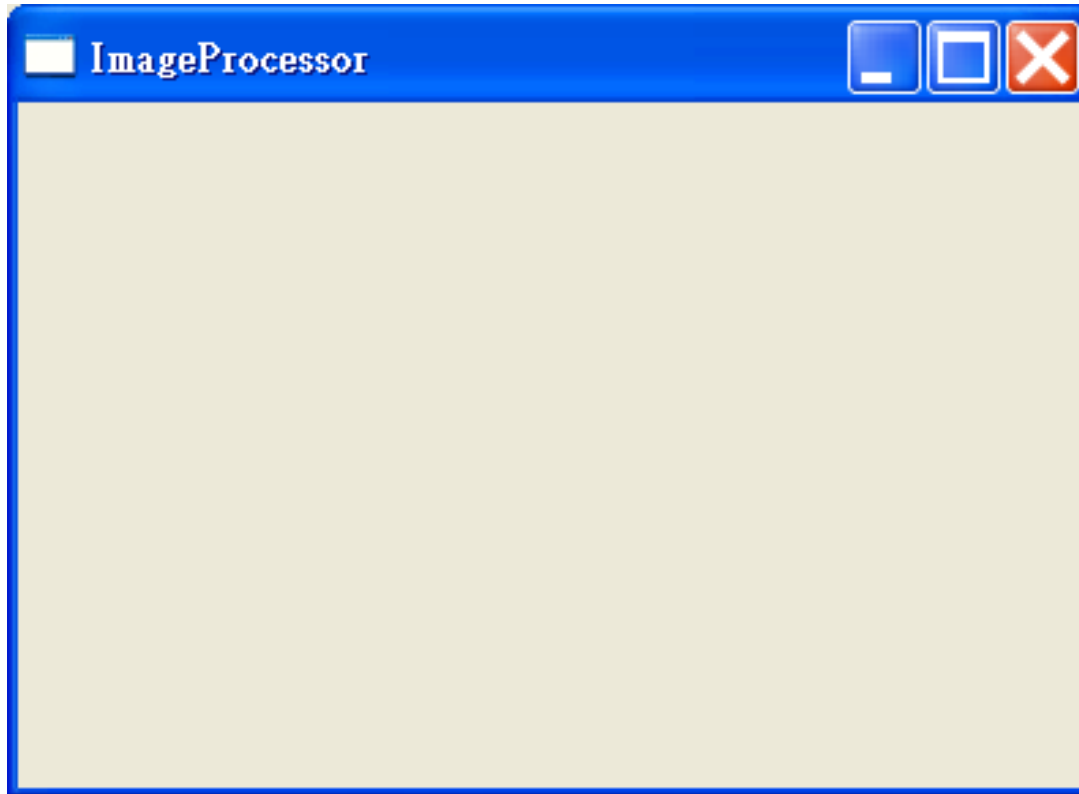
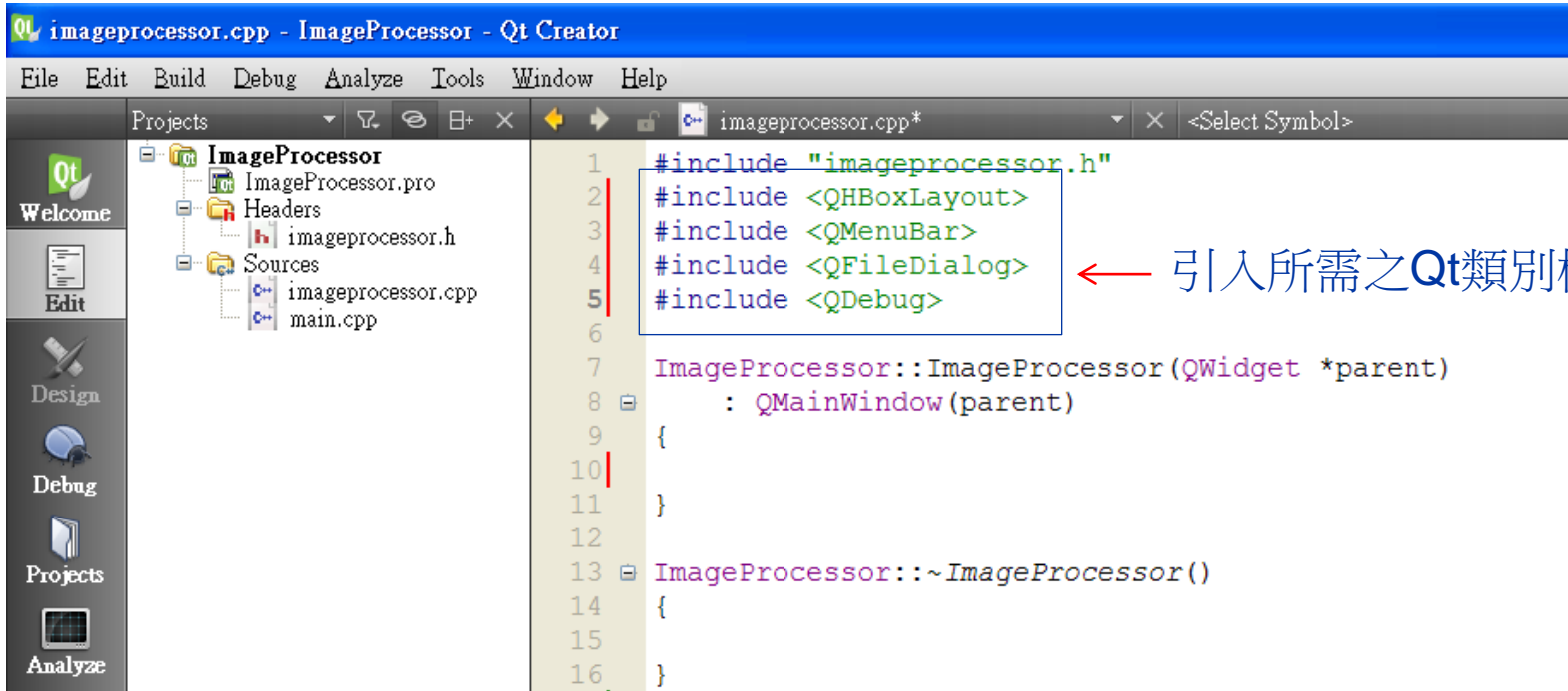


Image Processor

- Implement ImageProcessor's constructor:

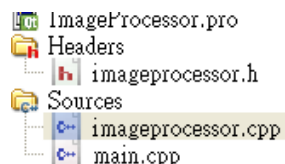


```
1 #include "imageprocessor.h"
2 #include <QBoxLayout>
3 #include <QMenuBar>
4 #include <QFileDialog>
5 #include <QDebug>
6
7 ImageProcessor::ImageProcessor(QWidget *parent)
8     : QMainWindow(parent)
9 {
10
11 }
12
13 ImageProcessor::~ImageProcessor()
14 {
15
16 }
```

← 引入所需之Qt類別標頭檔

Image Processor

● Implement ImageProcessor's member functions:

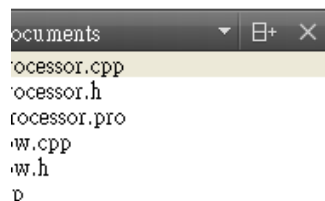


```

2  #include <QHBoxLayout>
3  #include <QMenuBar>
4  #include <QFileDialog>
5  #include <QDebug>
6
7  ImageProcessor::ImageProcessor(QWidget *parent)
8      : QMainWindow(parent)
9  {
10     setWindowTitle(QStringLiteral("影像處理"));
11     central = new QWidget();
12     QHBoxLayout *mainLayout = new QHBoxLayout(central);
13     imgWin = new QLabel();
14     QPixmap *initPixmap = new QPixmap(300,200);
15     initPixmap->fill(QColor(255,255,255));
16     imgWin->resize(300,200);
17     imgWin->setScaledContents(true);
18     imgWin->setPixmap(*initPixmap);
19     mainLayout->addWidget(imgWin);
20     setCentralWidget(central);
21     createActions();
22     createMenus();
23     createToolBars();
24 }
25
26 ImageProcessor::~ImageProcessor()
27 {

```

建構函數程式碼





Code less.
Create more.
Deploy everywhere.

Image Processor

- Build and execute the program:





Code less.
Create more.
Deploy everywhere.

Image Processor

- Implement ImageProcessor's member functions:

```
debug Analyze Tools Window Help
imageprocessor.cpp* ImageProcessor::createActions(): void
# Li

ImageProcessor
ImageProcessor.pro
Headers
imageprocessor.h
Sources
imageprocessor.cpp
main.cpp

21 createActions();
22 createMenus();
23 createToolBars();
24 }
25
26 ImageProcessor::~ImageProcessor()
27 {
28 }
29
30 void ImageProcessor::createActions()
31 {
32     openFileAction = new QAction(QStringLiteral("開啟檔案&O"), this);
33     openFileAction->setShortcut(tr("Ctrl+O"));
34     openFileAction->setStatusTip(QStringLiteral("開啟影像檔案"));
35     connect(openFileAction, SIGNAL(triggered()), this, SLOT(showOpenFile()));
36
37     exitAction = new QAction(QStringLiteral("結束&Q"), this);
38     exitAction->setShortcut(tr("Ctrl+Q"));
39     exitAction->setStatusTip(QStringLiteral("退出程式"));
40     connect(exitAction, SIGNAL(triggered()), this, SLOT(close()));
41 }
42 void ImageProcessor::createMenus()
43 {
```

函數程式碼
建立動作物件



Image Processor

● Implement ImageProcessor's member functions:

```

imageprocessor.cpp*
void ImageProcessor::createActions()
{
    openFileAction = new QAction(QStringLiteral("開啟檔案&O"), this);
    openFileAction->setShortcut(tr("Ctrl+O"));
    openFileAction->setStatusTip(QStringLiteral("開啟影像檔案"));
    connect(openFileAction, SIGNAL(triggered()), this, SLOT(showOpenFile()));

    exitAction = new QAction(QStringLiteral("結束&Q"), this);
    exitAction->setShortcut(tr("Ctrl+Q"));
    exitAction->setStatusTip(QStringLiteral("退出程式"));
    connect(exitAction, SIGNAL(triggered()), this, SLOT(close()));
}

void ImageProcessor::createMenus()
{
    fileMenu = menuBar()->addMenu(QStringLiteral("檔案&F"));
    fileMenu->addAction(openFileAction);
    fileMenu->addAction(exitAction);
}

void ImageProcessor::createToolBars()
{
}

void ImageProcessor::loadFile(QString filename)
{
}

```

函數程式碼
建立功能表



Code less.
Create more.
Deploy everywhere.

Image Processor

- Build and execute the program:

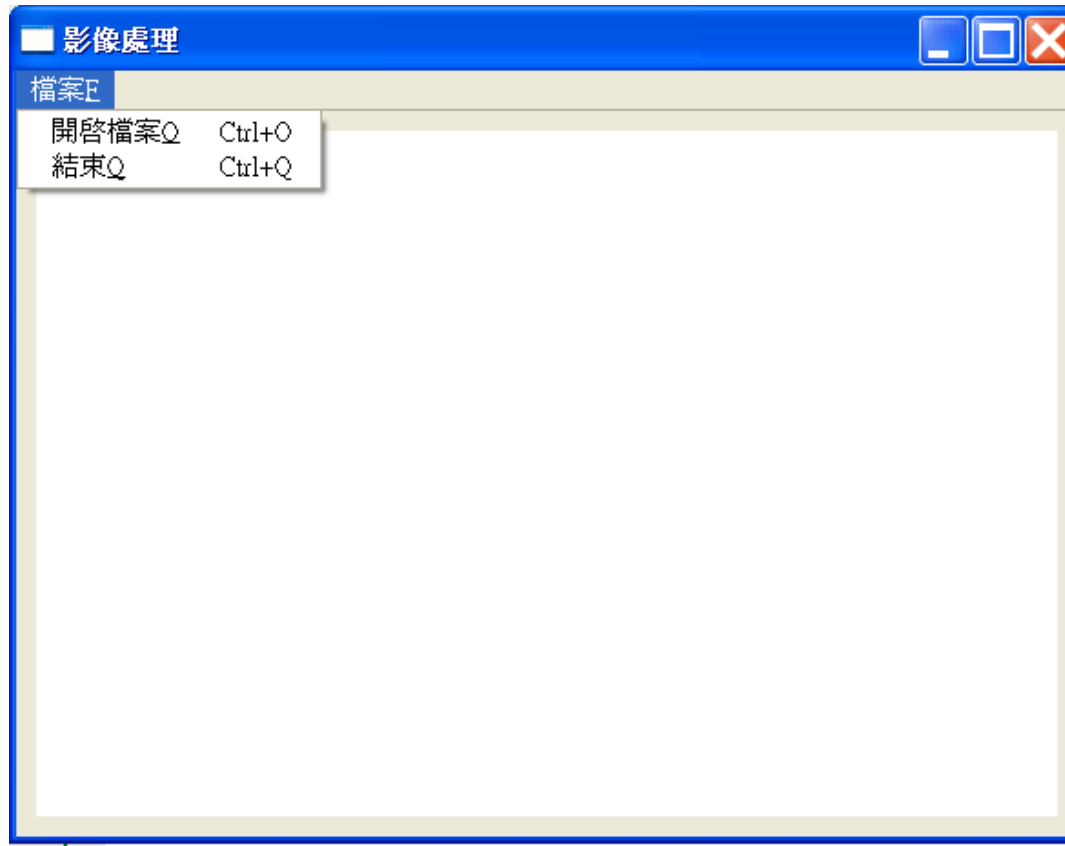


Image Processor

- Implement ImageProcessor's member functions:

```

41 }
42 void ImageProcessor::createMenus()
43 {
44     fileMenu = menuBar()->addMenu(QStringLiteral("檔案&F"));
45     fileMenu->addAction(openFileAction);
46     fileMenu->addAction(exitAction);
47 }
48 void ImageProcessor::createToolBars()
49 {
50 }
51 }
52 void ImageProcessor::loadFile(QString filename)
53 {
54     qDebug() << QString("file name:%1").arg(filename);
55     QByteArray ba=filename.toLatin1();
56     printf("FN:%s\n", (char *) ba.data());
57     img.load(filename);
58     imgWin->setPixmap(QPixmap::fromImage(img));
59 }
60 void ImageProcessor::showOpenFile()
61 {
62 }
63 }
64

```

函數程式碼載入
影像檔並顯示於
視窗中心元件上。



Code less.
Create more.
Deploy everywhere.

Image Processor

- Implement ImageProcessor's member functions:

```
58         img->load(fileName);  
59         imgWin->setPixmap(QPixmap::fromImage(img));  
60     }  
61     void ImageProcessor::showOpenFile()  
62     {  
63         filename = QFileDialog::getOpenFileName(this,  
64             QStringLiteral("開啟影像"),  
65             tr("."),  
66             "bmp (*.bmp);;png (*.png) "  
67             ";;Jpeg (*.jpg)");  
68         if (!filename.isEmpty())  
69         {  
70             if (img.isNull())  
71             {  
72                 loadFile(filename);  
73             }  
74             else  
75             {  
76                 ImageProcessor *newIPWin = new ImageProcessor();  
77                 newIPWin->show();  
78                 newIPWin->loadFile(filename);  
79             }  
80         }  
81     }  
82 }  
83
```

函數程式碼，啟動檔案對話盒，供使用者挑選檔案路徑與檔名後，呼叫載入影像檔並顯示於視窗中心元件上。



Image Processor

- Build and execute the program:

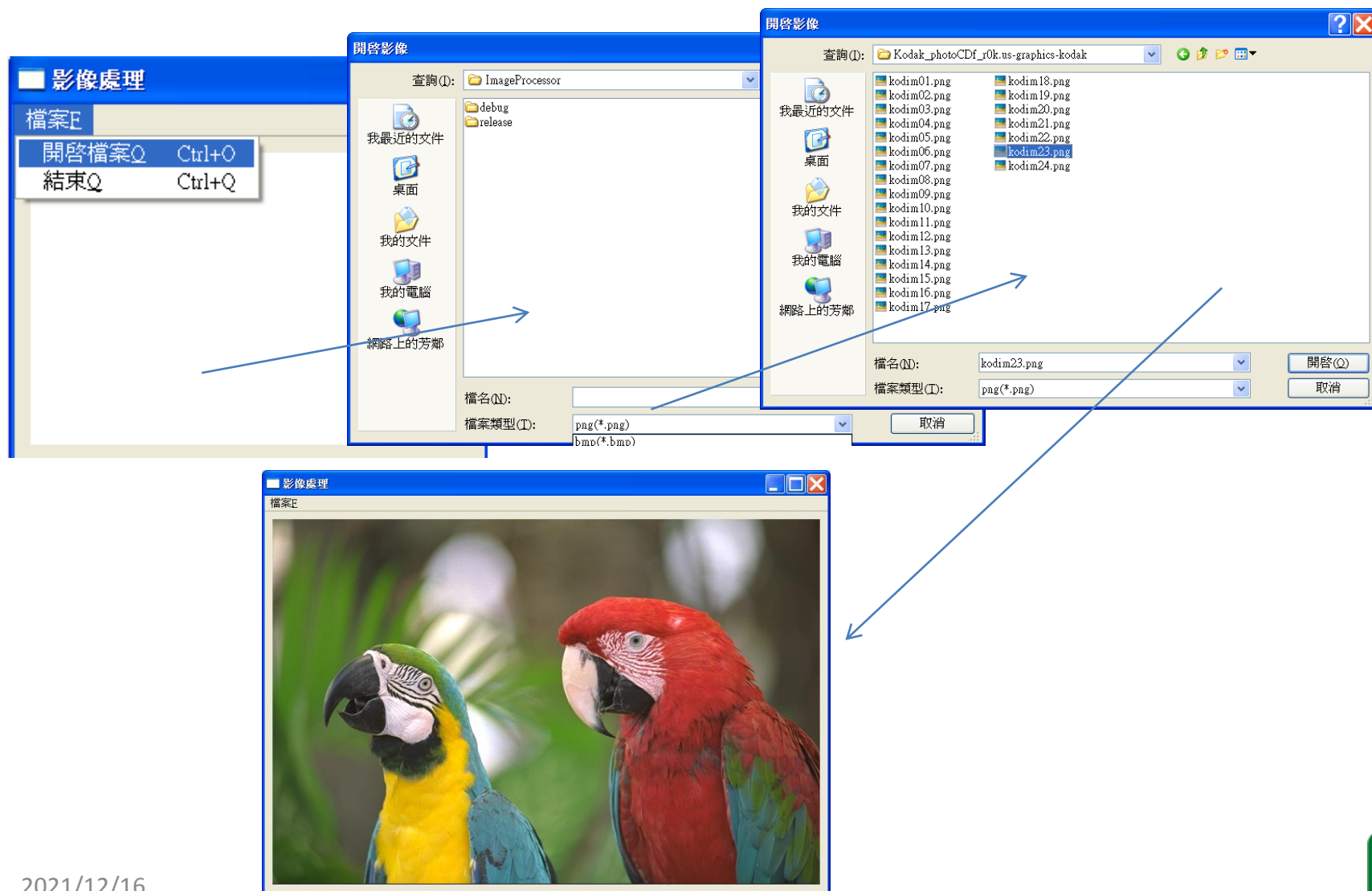


Image Processor

- Implement ImageProcessor's member functions:

```

41 }
42 void ImageProcessor::createMenus()
43 {
44     fileMenu = menuBar()->addMenu(QStringLiteral("檔案&F"));
45     fileMenu->addAction(openFileAction);
46     fileMenu->addAction(exitAction);
47 }
48 void ImageProcessor::createToolBars()
49 {
50     fileTool = addToolBar("file");
51     fileTool->addAction(openFileAction);
52 }
53 void ImageProcessor::loadFile(QString filename)
54 {
55     qDebug()<<QString("file name:%1").arg(filename);
56     QByteArray ba=filename.toLatin1();
57     printf("FN:%s\n", (char *) ba.data());
58     img.load(filename);
59     imgWin->setPixmap(QPixmap::fromImage(img));
60 }
61 void ImageProcessor::showOpenFile()
62 {
63     filename = QFileDialog::getOpenFileName(this,

```

函數程式碼，建立
檔案工具列。

Image Processor

- Build and execute the program:





Code less.
Create more.
Deploy everywhere.

Image Processor

- Exercise: Add more tools into the program; for example,

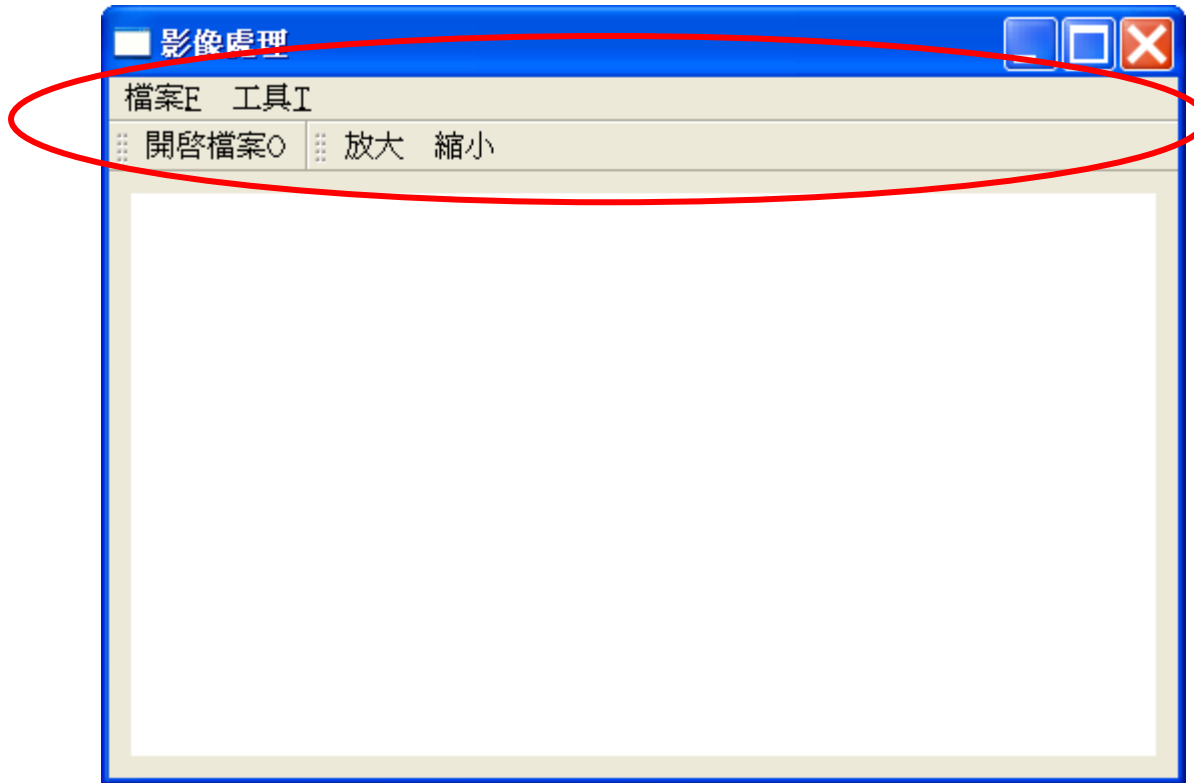


Image Processor

- Exercise: Add more tools into the program; for example,

