第4章　プログラム作成への準備

4.1　ダイアログを作ってみよう

今までコードはすべて main 関数の中に書いてきましたが、このままコードを拡張していったら main 関数が 1000 行くらいになってしまうかもしれません。

ということで、今回からはクラスを使って関数を分割していくことにします。また、ソースファイルも 01.cpp と main.h ファイルに分割します。

まず、今回作成するプログラムの実行結果は、図 4.1, 4.2 のようになります。

図 4.1: 初期状態 (on Windows)

図 4.2: ボタンを押した後 (on Windows)

起動した直後は図 4.1 の状態です。そして、テキストに何か文章を書き set ボタンを押すと、ラベルにテキストの内容がコピーされます（図 4.2)。

このプログラムは次のようなコードとなります。

```
1   //main.h
2
3   #ifndef MAIN_H_
4   #define MAIN_H_
5
6   #include <QDialog>
7
8   class QLabel;
9   class QPushButton;
10  class QLineEdit;
11
12  class MainDialog : public QDialog
13  {
14          Q_OBJECT
15  public:
16          MainDialog(QWidget* parent = 0);
17  private slots:
18          void setLabelText();
19  private:
20          QLabel* label;
21          QPushButton* setButton;
22          QLineEdit* lineEdit;
23  };
24
25  #endif
```

```
1   //01.cpp
2
3   #include <QtGui>
4   #include "main.h"
5
6   MainDialog::MainDialog(QWidget* parent)
7           : QDialog(parent)
8   {
9           label = new QLabel(tr("empty") );
10          setButton = new QPushButton(tr("Set") );
11          lineEdit = new QLineEdit;
```

```
12
13          connect(setButton,SIGNAL(clicked() ),this,SLOT(setLabelText() ) );
14
15          QHBoxLayout* layout = new QHBoxLayout;
16          layout->addWidget(lineEdit);
17          layout->addWidget(label);
18          layout->addWidget(setButton);
19          setLayout(layout);
20  }
21
22  void MainDialog::setLabelText()
23  {
24          QString text = lineEdit->text();
25          label->setText(text);
26  }
27
28  int main(int argc, char** argv)
29  {
30          QApplication app(argc,argv);
31          MainDialog* dialog = new MainDialog;
32          dialog->show();
33          return app.exec();
34  }
```

今回のコードはかなり長いですね。では、main.h ファイルのコードから見ていきましょう。

まず 3,4 行目のコードは C 言語や C++を使ってやや規模の大きなプログラムを作ったことのある人なら知っていると思います。この方法はインクルードガードと呼ばれています。知らない方はインターネットや書籍で勉強してみてください。Visual C++などの`#pragma once` と全く同じです。

6 行目では QDialog ヘッダをインクルードしています。今回はダイアログを作りたいので、このヘッダファイルを使います。

8,9,10 行目では QLabel,QPushButton,QLineEdit のプロトタイプ宣言です。この宣言では QLabel などのクラスの詳細はここでは定義していないけれど、他のヘッダファイルなどにはこのクラスが存在しているよ、ということをコンパイラ側に伝えています。

普通に`#include <QLabel>`などでヘッダファイルをインクルードしてもよいの

ですが、8〜10 行目のように宣言することでいちいちヘッダファイルへクラスを探しにいく手間が省け、コンパイル時間を短縮することができます。

12 行目で MainWindow クラスを作成しています。このクラスがメインのダイアログとなります。ダイアログを作成する場合は QDialog を継承する事により、基本的なダイアログの機能を引き継ぐことができます。よって拡張したい機能だけを自分で付け足せばよいわけです。（図 4.3）

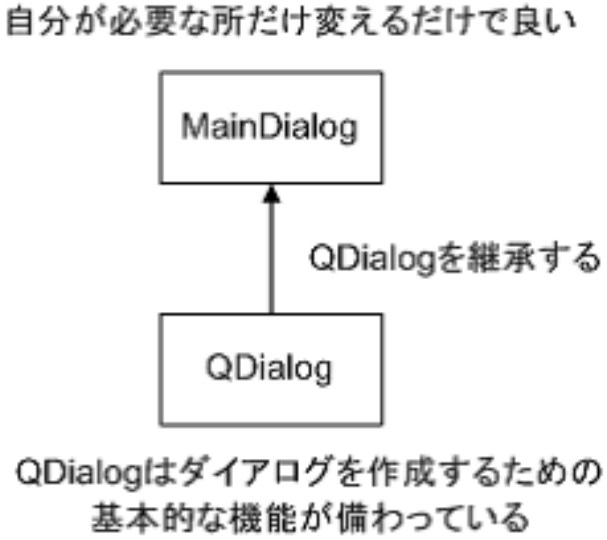

図 4.3: 継承のイメージ図

14 行目の `Q_OBJECT` は 17 行目で slots を使うためのマクロです。最後にセミコロンが付かない事に注意してください。

17 行目の `private slots:` はその先の setLabelText 関数が Slot として呼ばれる事があるという意味です。このようにする事で connect 関数を使って setLabelText 関数をどこかの Signal と結びつけることができます。他にも Signal として呼ばれる可能性のある関数には `signals:` としておきます。`signals:` は `private signals:` や `public signals:` にはならない事に注意してください。

次に 01.cpp ファイルの説明に入ります。

まず 3 行目では`<QtGui>`をインクルードしています。これは QLabel や QPushButton などといった Qt の GUI に関わる部分のヘッダファイルをすべてインクルードしているのと同じです。[1]クラスの規模が大きくなると、どのヘッダファイルをインクルードしたらよいのか分からなくなってしまいます。そういった事をいちいち考えなくてもよくなるので QtGui はたいへん便利です。

[1]正確には QtCore と QtGui の部分をインクルードしている

7 行目では QDialog のコンストラクタに親へのアドレスを渡しています。parent を 0 にした場合は、その部品が新しいウィンドウとして表示されます。つまり親となるわけです。

13 行目では connect 関数を使ってボタンが押されたら setLabelText 関数が呼び出されるようにしています。QDialog を継承すると connect 関数の QObject::というプリフィクスが不要となります。[2]

22 行目の setLabelText 関数では lineEdit の内容を label にセットしています。以上で説明は終了です。他に部品を追加したりしていろいろ試してみてください。

4.2 モーダルダイアログを作ってみよう

複数のウィンドウを表示したい場合があると思います。そういった場合に用いられるのが、モーダルダイアログとモードレスダイアログです。

モーダルダイアログとして新しいダイアログが呼び出された場合、呼び出した側のダイアログはモーダルダイアログが表示されている間は操作をすることができません。モードレスダイアログではそういった制限は特にありません。

今回はモーダルダイアログを作っていきたいと思います。完成したプログラムは図 4.4〜4.6 となります。

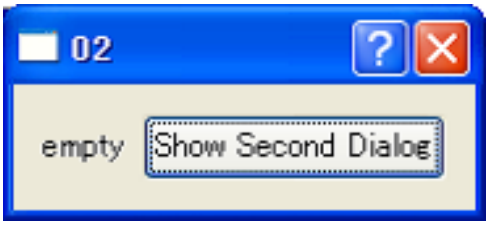

図 4.4: 初期状態 (on Windows)

まず始めに図 4.4 のウィンドウが表示されます。Show Second Dialog ボタンを押すと図 4.5 のモーダルダイアログが表示されます。そして、モーダルダイアログの中にあるテキストに何か文字を書き込み OK ボタンを押すと、始めにあったダイアログのラベルにテキストの文字がセットされます。(図 4.6)

このプログラムは次のようなソースコードとなります。今回はファイルを 5 分割してあり、コードも長くなっています。しかしながら決して難しくは無いと思います。

[2] これは QDialog が QObject から継承されている為です。更にいうと、QDialog は QWidget から継承されており、QWidget が QObject から継承されています。

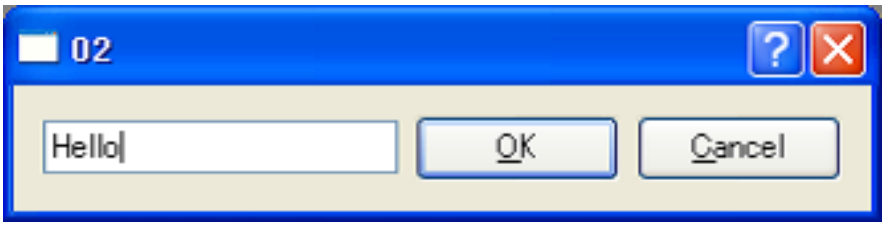

図 4.5: ボタンを押した後に表示されたモーダルダイアログ (on Windows)

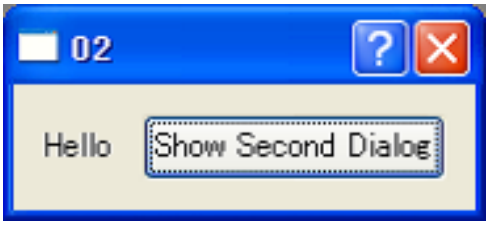

図 4.6: モーダルダイアログの ok ボタンを押した後 (on Windows)

```
1   //main.cpp
2
3   #include <QtGui>
4   #include "MainDialog.h"
5   #include "SecondDialog.h"
6
7   int main(int argc, char** argv)
8   {
9           QApplication app(argc, argv);
10          MainDialog* dialog = new MainDialog;
11          dialog->show();
12          return app.exec();
13  }
```

```
1   //MainDialog.h
2
3   #ifndef MAINDIALOG_H_
4   #define MAINDIALOG_H_
5
6   #include <QDialog>
```

```
 7
 8  class QPushButton;
 9  class QLabel;
10
11  class MainDialog : public QDialog
12  {
13          Q_OBJECT
14  public:
15          MainDialog(QWidget* parent = 0);
16  private slots:
17          void showSecondDialog();
18  private:
19          QPushButton* showDialogButton;
20          QLabel* textLabel;
21  };
22
23  #endif
```

```
 1  //MainDialog.cpp
 2
 3  #include <QtGui>
 4  #include "MainDialog.h"
 5  #include "SecondDialog.h"
 6
 7  MainDialog::MainDialog(QWidget* parent) : QDialog(parent)
 8  {
 9          showDialogButton = new QPushButton("Show Second Dialog");
10          textLabel = new QLabel("empty");
11          connect(showDialogButton,SIGNAL(clicked()),
12                          this,SLOT(showSecondDialog()) );
13
14          QHBoxLayout* layout = new QHBoxLayout;
15          layout->addWidget(textLabel);
16          layout->addWidget(showDialogButton);
17          setLayout(layout);
18  }
19
```

```
20  void MainDialog::showSecondDialog()
21  {
22          SecondDialog secondDialog(this);
23          if(secondDialog.exec()) {
24                  QString str = secondDialog.getLineEditText();
25                  textLabel->setText(str);
26          }
27  }
```

```
1   //SecondDialog.h
2
3   #ifndef SECONDDIALOG_H_
4   #define SECONDDIALOG_H_
5
6   #include <QDialog>
7
8   class QPushButton;
9   class QLineEdit;
10
11  class SecondDialog : public QDialog
12  {
13          Q_OBJECT
14  public:
15          SecondDialog(QWidget* parent = 0);
16          QString getLineEditText();
17  private:
18          QPushButton* okButton;
19          QPushButton* cancelButton;
20          QLineEdit* editor;
21  };
22
23  #endif
```

```
1   //SecondDialog.cpp
2
3   #include <QtGui>
4   #include "SecondDialog.h"
```

```
5
6   SecondDialog::SecondDialog(QWidget* parent) : QDialog(parent)
7   {
8           okButton = new QPushButton(tr("&OK") );
9           cancelButton = new QPushButton(tr("&Cancel") );
10          editor = new QLineEdit;
11
12          QHBoxLayout* layout = new QHBoxLayout;
13          layout->addWidget(editor);
14          layout->addWidget(okButton);
15          layout->addWidget(cancelButton);
16          setLayout(layout);
17
18          connect(okButton, SIGNAL(clicked()), this, SLOT(accept()) );
19          connect(cancelButton,SIGNAL(clicked()), this, SLOT(reject()) );
20  }
21
22  QString SecondDialog::getLineEditText()
23  {
24          return editor->text();
25  }
```

まず、main.cpp ですが、これはもう見てわかると思うので説明は省略します。

MainDialog.h ではメインのダイアログクラスの定義を行っています。

MainDialog.cpp のコンストラクタの中にある connect は、Show Second Dialog ボタンが押された時に showSecondDialog 関数が呼び出されるように設定しています。

今回新しく出てくるところは、showSecondDialog 関数内のコードです。

まず、22 行目では 2 つ目のダイアログである secondDialog 変数を作成しています。引数に this を渡していますが、これは secondDialog.cpp のコンストラクタの定義を見てわかると思います。MainDialog を secondDialog の親にするということです。

次に出てくる secondDialog.exec() ですが、これは secondDialog をモーダルダイアログとして呼び出すということです。モーダルダイアログとして呼び出しているので、secondDialog が表示されている間は MainDialog 側を操作することができません。

また、secondDialog.exec() 関数は QDialog::Accepted または QDialog::Rejected

を返します。この QDialog::Accepted は 1、QDialog::Rejected は 0 と定義されています。つまり、Accepted が返された場合は 24,25 行目が実行され、Rejected が返された場合は 24,25 行目は実行されません。

SecondDialog.h は 2 つ目のダイアログクラスを定義しています。

SecondDialog.cpp の 18,19 行目を見てください。ここで okButton を押した場合、accept 関数を呼び出し、cancelButton を押した場合、reject 関数を呼び出すよう設定しています。accept 関数も reject 関数もウィンドウを隠すという動作を行います。ただ、MainDialog.cpp の 23 行目で呼び出した exec 関数の返却値を QDialog::Accepted にするか QDialog::Rejected にするかが違います。

4.3 モードレスダイアログを作ってみよう

今度はモードレスダイアログを作ってみましょう。モードレスダイアログはモーダルダイアログのように他のウィンドウが操作できないなどの制約は特にありません。

モードレスダイアログのコードは次のようになります。ボタンの配置などは 4.2 節のコードは同じです。

```
1   //main.cpp
2
3   #include <QtGui>
4   #include "MainDialog.h"
5   #include "SecondDialog.h"
6
7   int main(int argc, char** argv)
8   {
9           QApplication app(argc, argv);
10          MainDialog* dialog = new MainDialog;
11          dialog->show();
12          return app.exec();
13  }
```

```
1   //MainDialog.h
2
3   #ifndef MAINDIALOG_H_
4   #define MAINDIALOG_H_
5
```

```
6   #include <QDialog>
7
8   class QPushButton;
9   class QLabel;
10  class SecondDialog;
11
12  class MainDialog : public QDialog
13  {
14          Q_OBJECT
15  public:
16          MainDialog(QWidget* parent = 0);
17  public slots:
18          void showSecondDialog();
19          void setTextLabel();
20  private:
21          QPushButton* showDialogButton;
22          QLabel* textLabel;
23          SecondDialog* secondDialog;
24  };
25
26  #endif
```

```
1   //MainDialog.cpp
2
3   #include <QtGui>
4   #include "MainDialog.h"
5   #include "SecondDialog.h"
6
7   MainDialog::MainDialog(QWidget* parent) : QDialog(parent), secondDialog(NULL)
8   {
9           showDialogButton = new QPushButton("Show Second Dialog");
10          textLabel = new QLabel("empty");
11          connect(showDialogButton, SIGNAL(clicked()),
12                                    this, SLOT(showSecondDialog()) );
13
14          QHBoxLayout* layout = new QHBoxLayout;
15          layout->addWidget(textLabel);
```

```
16          layout->addWidget(showDialogButton);
17          setLayout(layout);
18  }
19
20  void MainDialog::showSecondDialog()
21  {
22          if(!secondDialog){
23                  secondDialog = new SecondDialog;
24                  connect(secondDialog, SIGNAL(okButtonClicked() ),
25                                          this, SLOT(setTextLabel()) );
26          }
27          if(secondDialog->isHidden() ) {
28                  secondDialog->show();
29          }else{
30                  secondDialog->activateWindow();
31          }
32  }
33
34  void MainDialog::setTextLabel()
35  {
36          QString str = secondDialog->getLineEditText();
37          textLabel->setText(str);
38  }
```

```
1   //SecondDialog.h
2
3   #ifndef SECONDDIALOG_H_
4   #define SECONDDIALOG_H_
5
6   #include <QDialog>
7
8   class QPushButton;
9   class QLineEdit;
10  class QString;
11
12  class SecondDialog : public QDialog
13  {
```

```
14          Q_OBJECT
15  public:
16          SecondDialog(QWidget* parent = 0);
17          QString getLineEditText();
18  signals:
19          void okButtonClicked();
20  private:
21          QPushButton* okButton;
22          QPushButton* cancelButton;
23          QLineEdit* editor;
24  };
25
26  #endif
```

```
1   //SecondDialog.cpp
2
3   #include <QtGui>
4   #include "SecondDialog.h"
5
6   SecondDialog::SecondDialog(QWidget* parent) : QDialog(parent)
7   {
8           okButton = new QPushButton(tr("&OK") );
9           cancelButton = new QPushButton(tr("&Cancel") );
10          editor = new QLineEdit;
11
12          QHBoxLayout* layout = new QHBoxLayout;
13          layout->addWidget(editor);
14          layout->addWidget(okButton);
15          layout->addWidget(cancelButton);
16          setLayout(layout);
17
18          connect(okButton,SIGNAL(clicked()),this,SIGNAL(okButtonClicked()) );
19          connect(okButton,SIGNAL(clicked()), this, SLOT(close()) );
20          connect(cancelButton,SIGNAL(clicked()), this, SLOT(close()) );
21  }
22
23  QString SecondDialog::getLineEditText()
```

```
24  {
25          return editor->text();
26  }
```

まず、MainDialog.cpp の showSecondDialog 関数を見てください。この関数は Show Second Dialog ボタンを押すことによって呼び出されます。

22 行目の if 文の中身は secondDialog が NULL であるときに呼び出されます。つまりはじめて secondDialog を呼び出すときに実行されます。

24 行目では secondDialog の okButtonClicked 関数が呼び出されたときに setTextLabel 関数が呼び出されます。okButtonClicked 関数は SecondDialog.h に定義されています。この関数の説明はもう少し先で説明します。

setTextLabel 関数では、secondDialog にあるテキストの中身を MainDialog のラベルにセットしています。getLineEditText 関数も SecondDialog.h および SecondDialog.cpp に定義されています。

SecondDialog.h の 18,19 行目及び SecondDialog.cpp の 18,19 行目を見てください。SecondDialog.h の 18 行目の `signals:` は 19 行目の okButtonClicked 関数をシグナル関数とするためのマクロです。この okButtonClicked 関数の中身は Qt が勝手に作成します。よって定義のみ書けばよいです。

そして SecondDialog.cpp の 18 行目で okButton が押されたときに okButtonClicked 関数が呼び出されるようにセットしています。今までは connect 関数の第 4 引数は SLOT としていましたが、18 行目のように SIGNAL を呼び出すこともできます。

そして 19,20 行目では okButton 及び cancelButton が押されたときにダイアログが閉じるよう close 関数を connect しています。

最後の getLineEditText 関数では secondDialog 内のテキストの中身を取得するための関数です。

参考文献

[1] Jasmin Blanchette & Mark Summerfield, C++ GUI Programming with Qt4.

[2] Trolltech, Qt Assistant Tutorial and Examples Qt Tutorial

[3] Trolltech, Qt Assistant All Classes

[4] Trolltech, Qt Assistant Core Features Layout Management