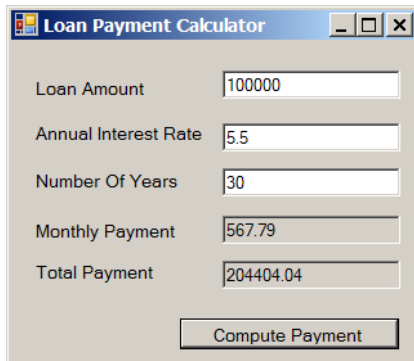


Supplement: Visual C++ 2012 GUI Applications
For Introduction to C++ Programming
By Y. Daniel Liang

1 Introduction

This supplement introduces GUI programming using Visual C++ 2012.

We will create a loan calculator application with the graphical user interface, as shown in Figure 1.



Field	Value
Loan Amount	100000
Annual Interest Rate	5.5
Number Of Years	30
Monthly Payment	567.79
Total Payment	204404.04

Figure 1

The application lets the user enter loan information and compute the monthly payment and total payment.

2 Creating GUI Projects

To create a GUI application using Visual C++ 2012, you have to first create a CLR project. Here are the steps to create a CLR (Common Language Runtime for Microsoft .NET) project named GUIDemo.

1. Choose *File, New Project* to display the New Project dialog box, as shown in Figure 2.
2. Select CRL under C++ and select CLR Empty Project.
3. Enter **GUIDemo** in the Name field and click *OK* to create the project, as shown in Figure 3.

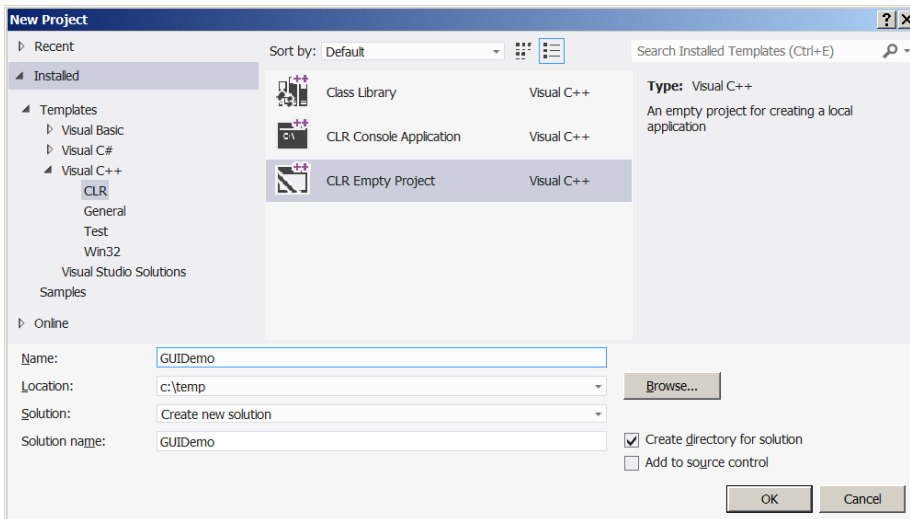


Figure 2
You need to create a CLR project to develop GUI programs in Visual C++ 2012.

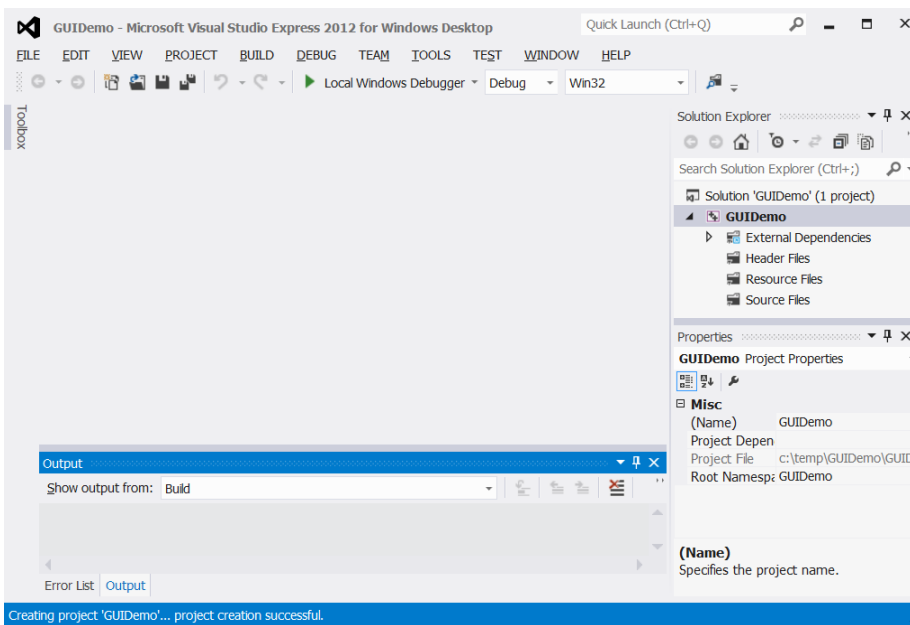


Figure 3
The new project wizard creates an empty CLR project.

3 Creating Forms

Forms are standalone windows for displaying the user interface. Here are the steps to create a form:

1. Choose *Project, Add New Items* to display the Add New Items window, as shown in Figure 4.

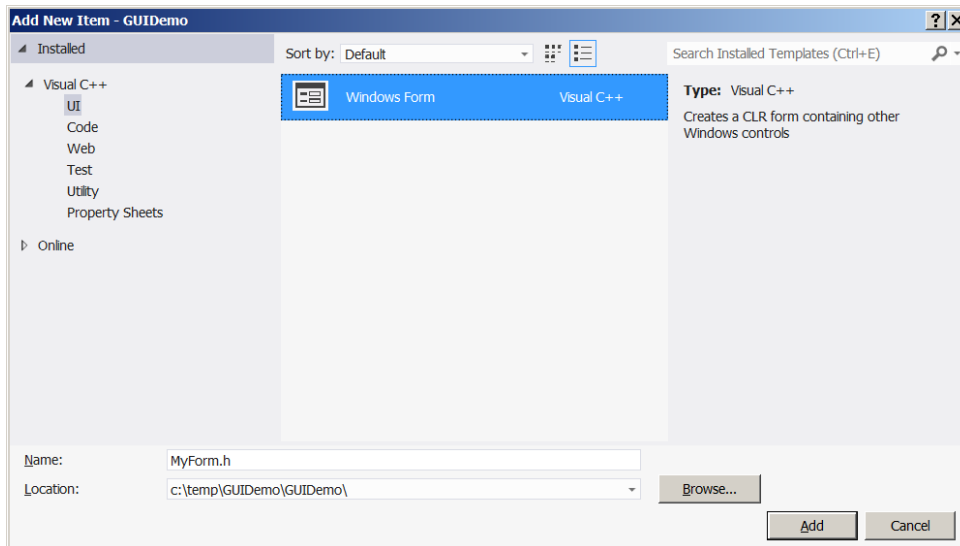


Figure 4

Use the Add New Items wizard to create a new form.

2. Select UI under Visual C++ and select Windows Forms and click *Add* to create a new form as shown in Figure 5.

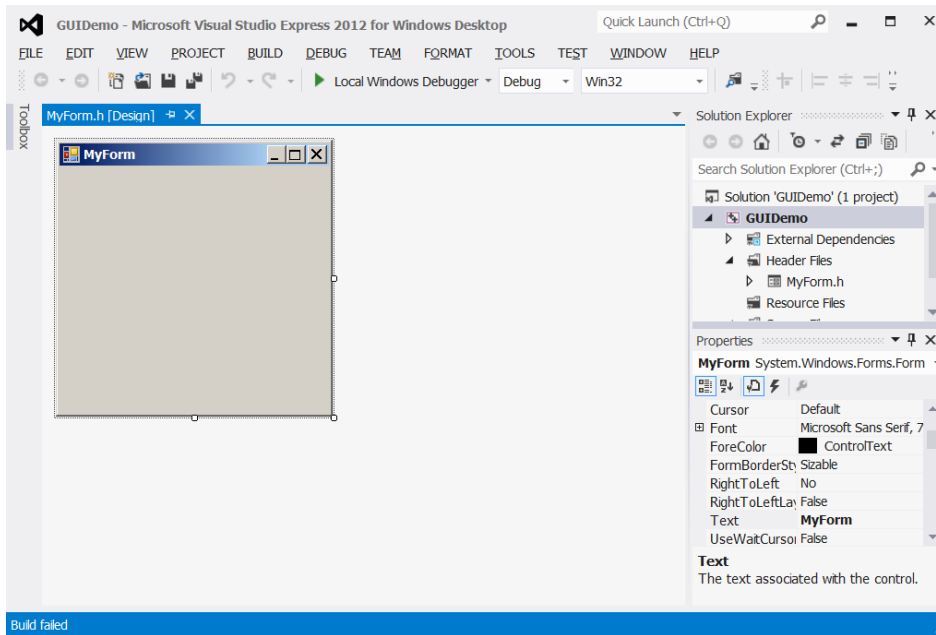


Figure 5

Use the Add New Items wizard to create a new form.

3. Replace MyForm.cpp (see Figure 6) with the following code: (Note GUIDemo is the name of the project you created.)

```
#include "MyForm.h"
```

```

using namespace System;
using namespace System::Windows::Forms;

[STAThread]
int main(array<String^>^ args) {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);

    GUIDemo::MyForm form;
    Application::Run(%form);
}

```

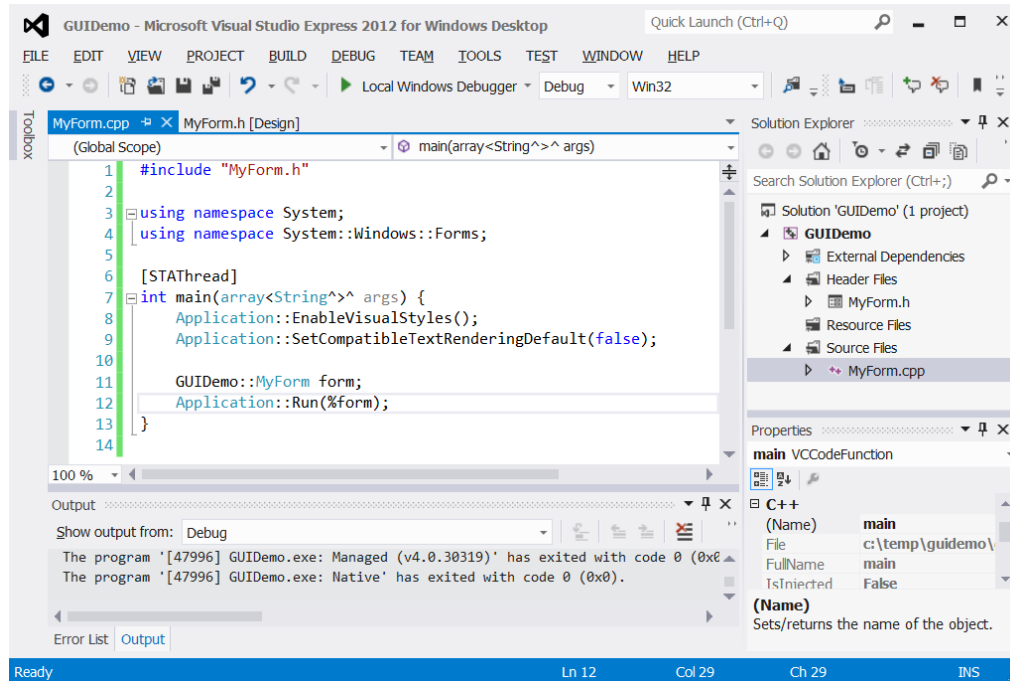


Figure 6
You need to add the code in MyForm.cpp.

4 Designing GUI

The project is created with a graphical form and several supporting files. The form is for creating the graphical user interface. You can add GUI components such as labels, buttons, and text boxes into the form.

Visual C++ provides an intuitive drag and drop feature for creating GUI user interface. To add a GUI component to the form, simply drag the component from the Toolbox to the form. Here are the steps to create the GUI for the loan calculator project:

1. Click MyForm.h [Design] tab and choose *View, Toolbox* to display the Toolbox. (You may need to move Toolbox to a separate window as shown in Figure 7.)

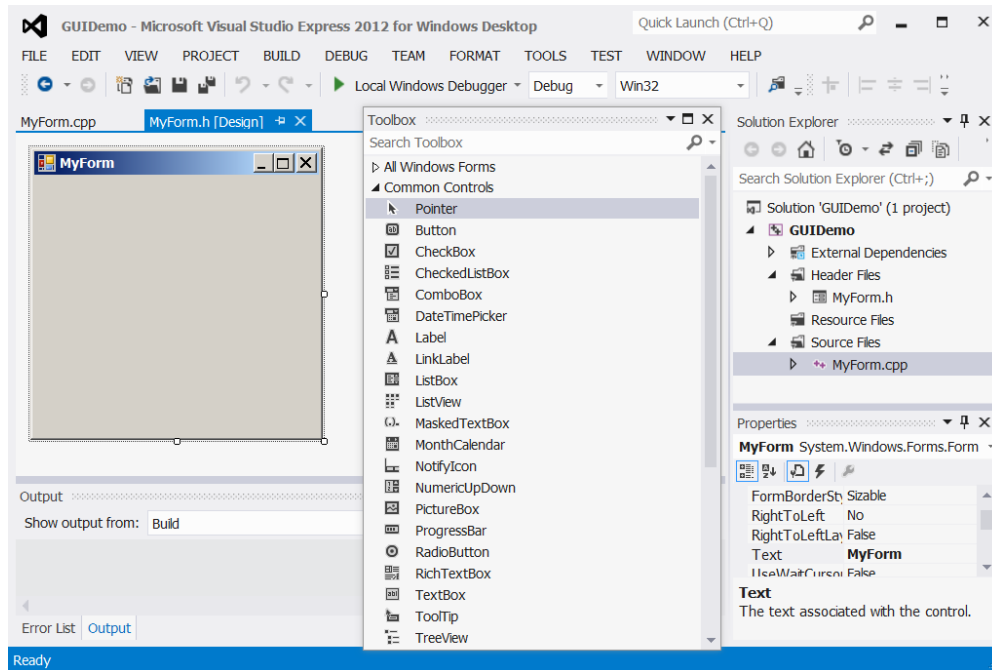


Figure 7

Toolbox is displayed for visually designing the form.

2. Click on Label in the Toolbox and drop it in the form, as shown in Figure 8. In the Property window for the label, change the **Text** property to **Loan Amount**. Similarly drop four more labels to the form and change their **Text** properties to **Annual Interest Rate**, **Number Of Years**, **Monthly Payment**, and **Total Payment**, as shown in Figure 9.

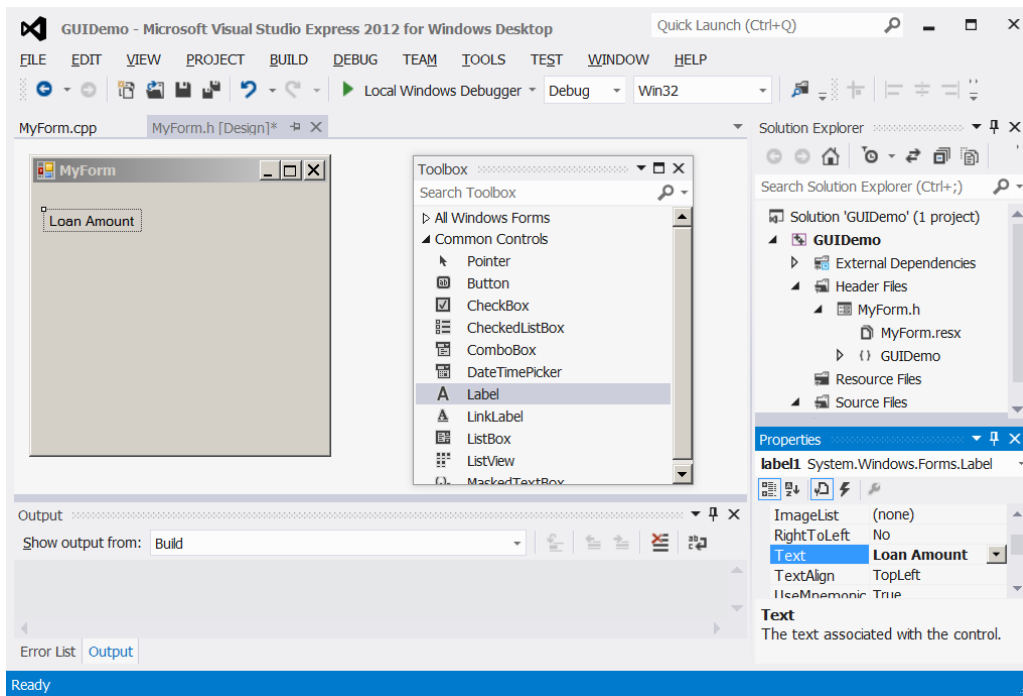


Figure 8
You can add the components from the Toolbox to the form.

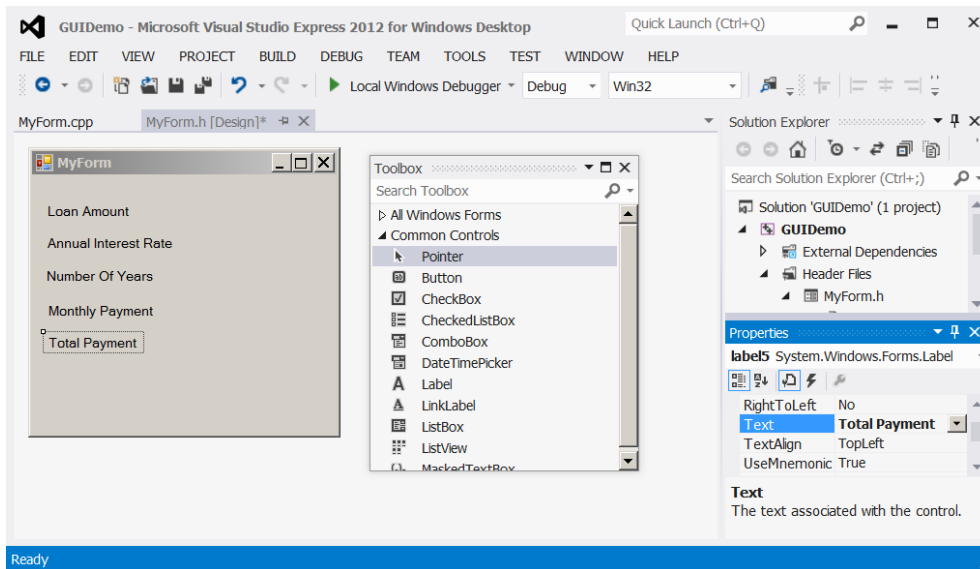


Figure 9
Five labels are added in the form.

3. Click on TextBox in the Toolbox and drop it in the form, as shown in Figure 10. In the Property window for the text box, change the **Name** property to **tbLoanAmount**. Similarly drop four more text boxes to the form and change their **Name** properties to **tbAnnualInterestRate**, **tbNumberOfYears**,

`tbMonthlyPayment`, and `tbTotalPayment`, as shown in Figure 11.

4. Click on Button in the Toolbox and drop it in the form, as shown in Figure 12. In the Property window for the button, change the **Name** property to `btComputePayment` and the **Text** property to **Compute Payment**.

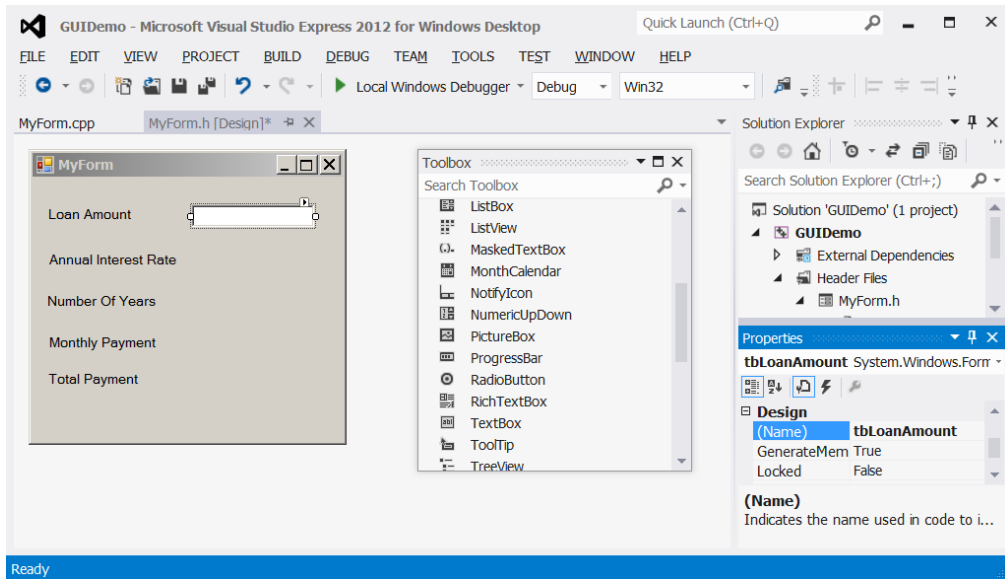


Figure 10
A TextBox is added to the form.

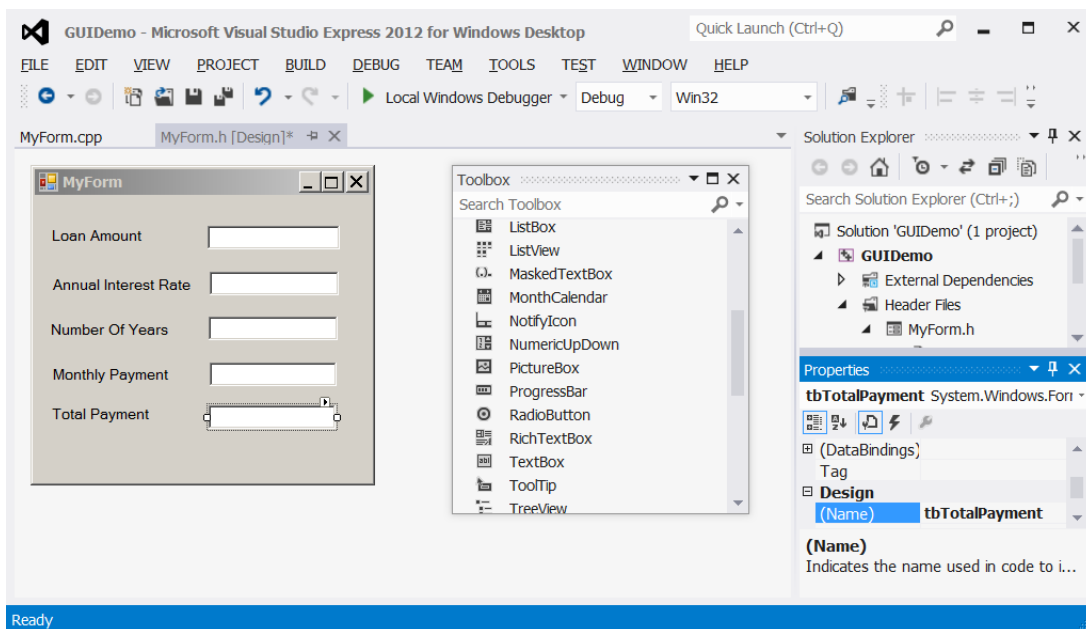


Figure 11

Five text boxes are added to the form.

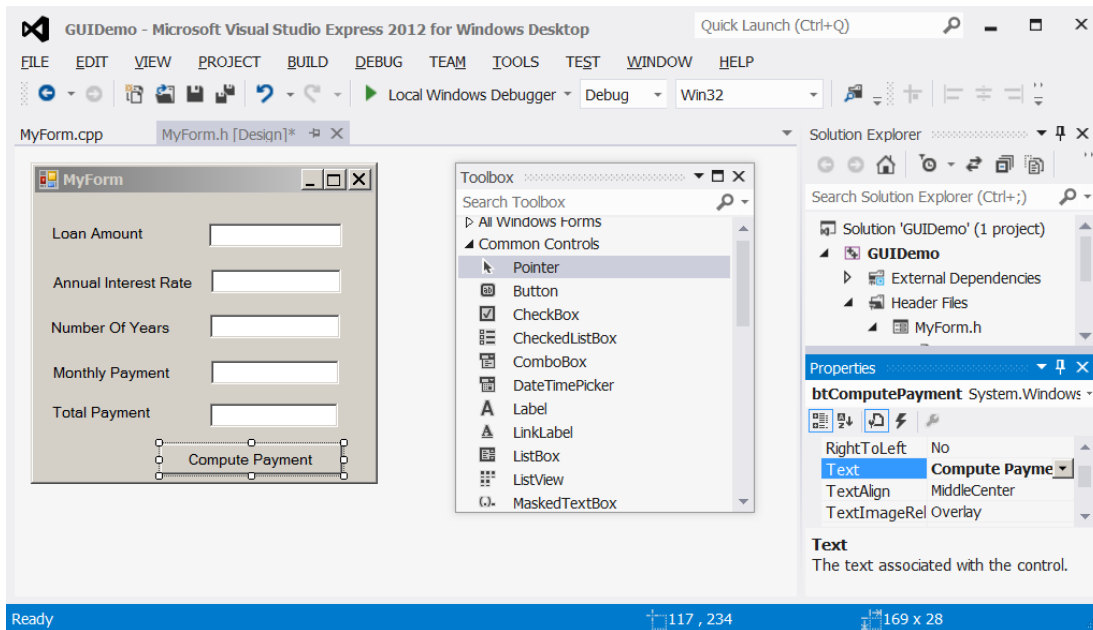


Figure 12

A button is added to the form.

The information about form is stored in MyForm.h. Whenever you change the form in the visual designer, the corresponding code in MyForm.h is also changed, as shown in Figure 13. For example, the information on the Loan Amount label is shown in lines 78–83. (To view Form MyForm.h, right-click MyForm.h in the Solution Explorer, choose View Code.)

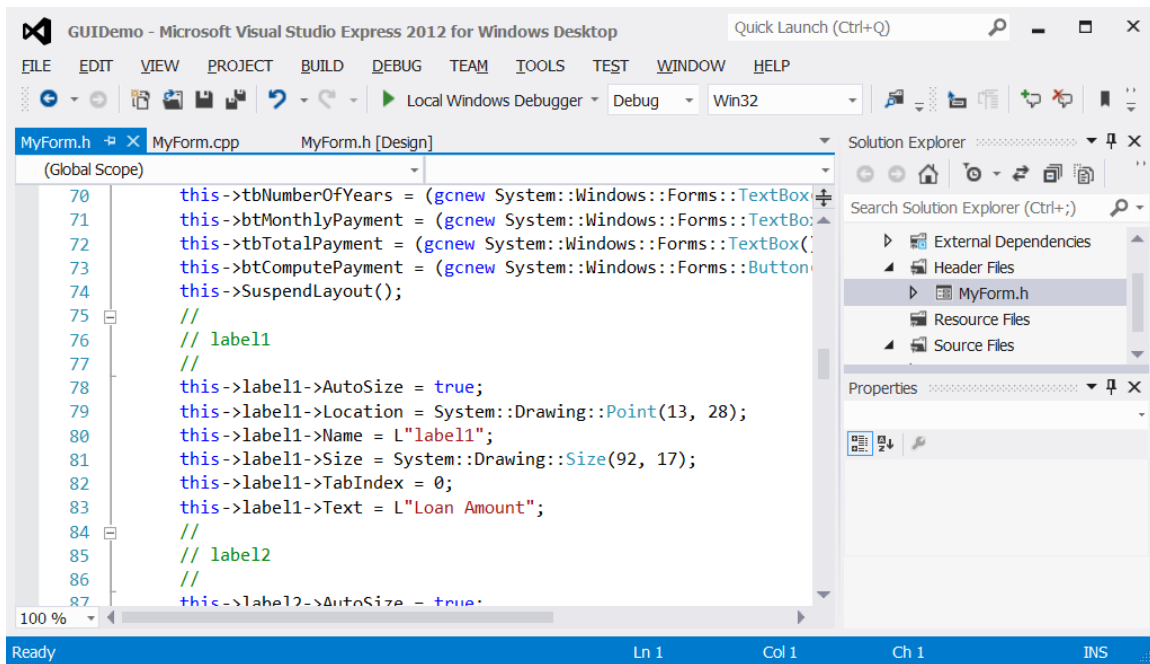


Figure 13
MyForm.h contains the code that describes for the form.

5 Handling Events

When you run a GUI program, the program interacts with the user, and the events drive its execution. An *event* can be defined as a signal to the program that something has happened. Events are triggered either by external user actions, such as mouse movements, button clicks, and keystrokes, or by internal program activities, such as a timer. The program can choose to respond to or ignore an event.

The component for which an event is originated is called the *source component*. For example, a button is the source object for a button-clicking action event. A function that performs a task to respond to the event is called an *event handler*.

For our loan calculator GUI, when the user clicks the *Compute Payment* button, the handler gathers the loan information from the text boxes, computes the monthly payment and total payment, and displays them in the respective text boxes.

Here are the steps to create a handler for the button-click event on the *Compute Payment* button:

1. Choose the Event tab (⚡) in the Property window for `btComputePayment`, as shown in Figure 14.

2. A button source component can handle many events. The one we need for this application is the click event. In the empty text box on the right of Click, press the Enter key. You will see the following handler is automatically generated in MyForm.h, as shown in Figure 15.

```
private: System::Void btComputePayment_Click(System::Object^ sender,
      System::EventArgs^ e)
{
}
}
```

3. Write the code to compute and display monthly and total payment as follows:

```
private: System::Void btComputePayment_Click(System::Object^ sender,
      System::EventArgs^ e)
{
    double loanAmount = System::Convert::ToDouble(tbLoanAmount->Text);
    double annualInterestRate
        = System::Convert::ToDouble(tbAnnualInterestRate->Text);
    double numberOfYears
        = System::Convert::ToDouble(tbNumberOfYears->Text);

    double monthlyInterestRate = annualInterestRate / 1200;

    double monthlyPayment = loanAmount * monthlyInterestRate / (1
        - 1 / pow(1 + monthlyInterestRate, numberOfYears * 12));
    double totalPayment = monthlyPayment * numberOfYears * 12;

    monthlyPayment = floor(monthlyPayment * 100 + 0.5) / 100;
    totalPayment = floor(totalPayment * 100 + 0.5) / 100;

    tbMonthlyPayment->Text = System::Convert::ToString(monthlyPayment);
    tbTotalPayment->Text = System::Convert::ToString(totalPayment);
}
}
```

4. You can also set a title for the form by setting the **Text** property of the form to **Loan Payment Calculator**. You should set the **ReadOnly** property for the monthly payment and total payment text boxes to **True** to prevent the user from editing these text boxes.
5. Add **#include <cmath>** in the beginning of MyForm.h.
6. You can now run the program. The output is shown in Figure 1.

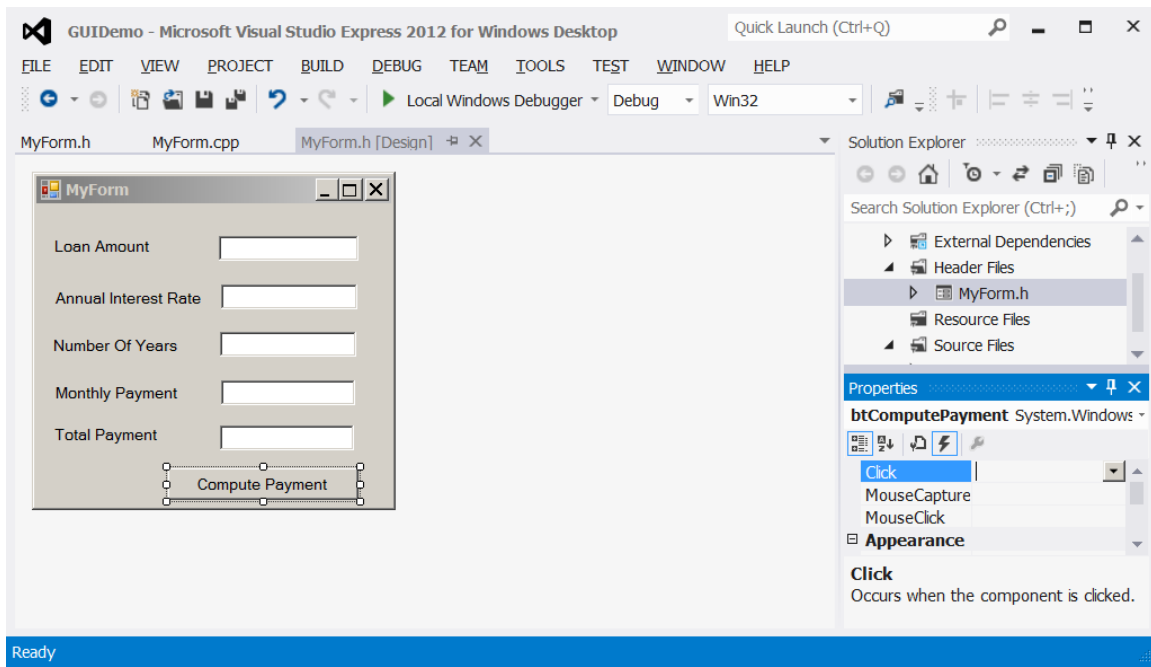


Figure 14

You can generate event handlers from the Event tab of the Properties window.

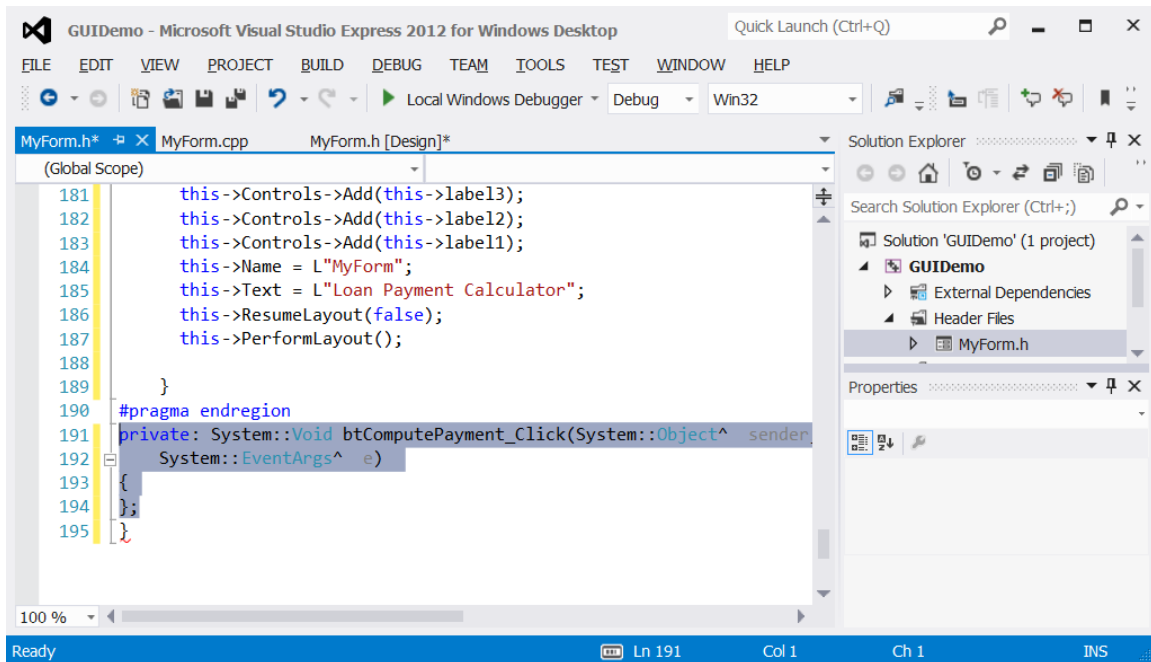


Figure 15

The function for handling the event is automatically generated in MyForm.h.