概念题

1. 请简述应用框架的概念和作用。
   概念：通用的、可复用的应用程序结构，封装了某领域的程序处理流程的控制逻辑。
   作用：在一个应用框架中，各个组成成分之间的关系是固定的，
   应用的开发者是通过给各组成成分添加具体的业务代码来实现不同应用的。
   通过复用应用框架，使得开发应用的速度更快、质量更高、成本更低。

2. 请简述基于“文档-视”结构的多文档应用框架中，视对象与文档对象如何进行交互。
   通过 CView 类的成员函数 GetDocument 可获得与视对象对应的文档对象。
   通过 CDocument 类的成员函数 GetFirstViewPosition 和 GetNextView
   可获得与文档对象对应的视对象（可以有多个）。
   修改文档数据后，可以通过 CDocument 类的成员函数 SetModifiedFlag
   设置文档的修改标记为 true，并调用 CDocument 类的成员函数 UpdateAllViews
   通知与文档对象对应的视对象刷新显示：
   这时，CView 类的成员函数 OnUpdate 和 OnDraw 将会被调用。

编程题

1. 有一个 MFC 应用程序，可以通过键盘录入文本，并实时通过视图显示。
   文本可以保存到文件中，打开文件后显示文件中的文本，并可以在文本末尾继续录入。
   文本包括大小写字母，空格，数字和标点符号（代码仅给出自行实现的改动部分）

InputAndShowDoc.h

```
class CInputAndShowDoc : public CDocument
{
……
public:
	CString text;
	int UpdateText(UINT nChar);
	const CString& GetText();
};
```

InputAndShowDoc.cpp

```
CInputAndShowDoc::CInputAndShowDoc() noexcept
{
	text = _T("");
}

void CInputAndShowDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
```

```
        ar << text;
    }
    else
    {
        ar >> text;
    }
}

int CInputAndShowDoc::UpdateText(UINT nChar)
{
    text.AppendChar(wchar_t(nChar));
    UpdateAllViews(NULL);
    SetModifiedFlag(true);
    return 0;
}

const CString& CInputAndShowDoc::GetText()
{
    return text;
}
```

InputAndShowView.h

```
class CInputAndShowView : public CView
{
.....
public:
    afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
};
```

InputAndShowView.cpp

```
void CInputAndShowView::OnDraw(CDC* pDC)
{
    CInputAndShowDoc* pDoc = GetDocument();
    pDC->TextOutW(0, 0, pDoc->GetText());
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;
}

void CInputAndShowView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    Invalidate();
```

```
    CInputAndShowDoc* pDoc = GetDocument();
    pDoc->UpdateText(nChar);
    CView::OnChar(nChar, nRepCnt, nFlags);
}
```

实现效果如图：