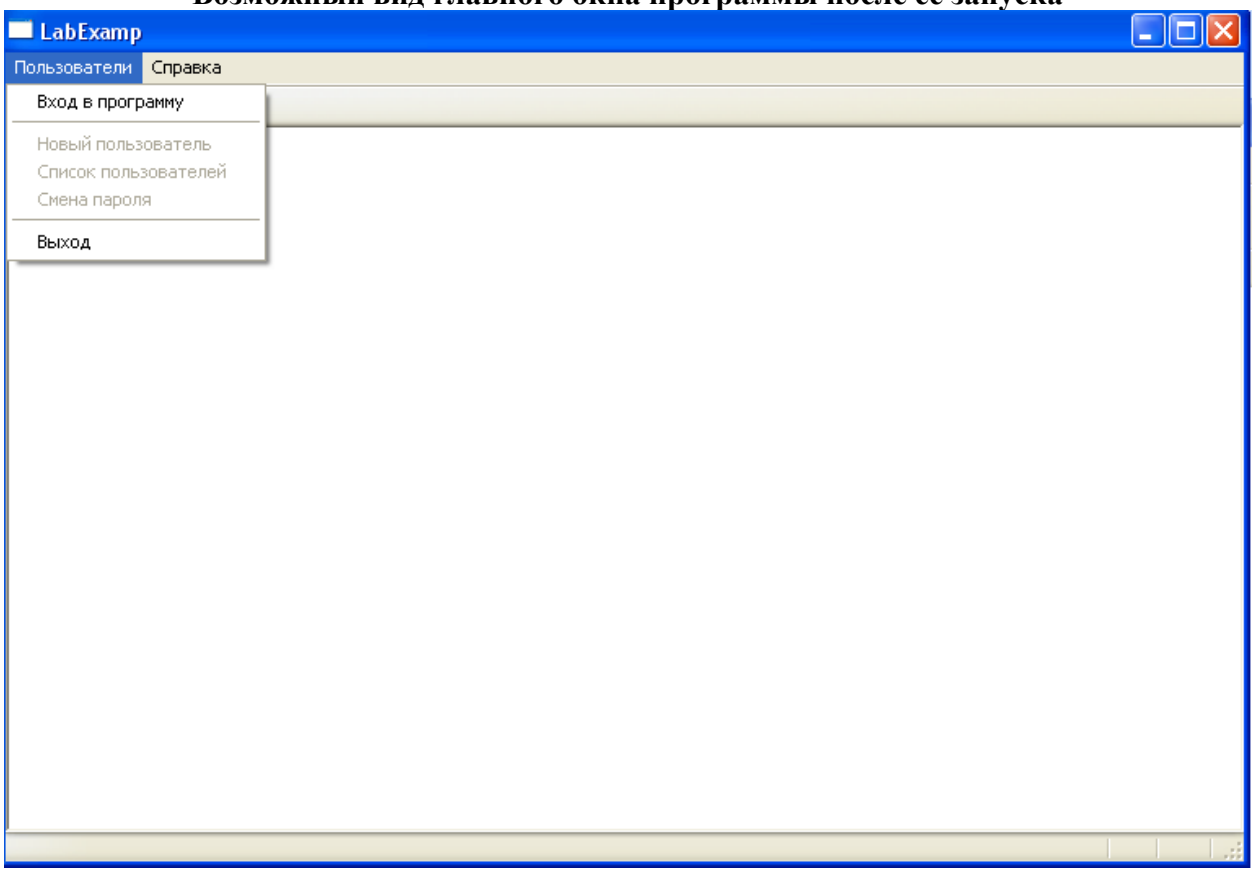
Указания по выполнению лабораторных работ по дисциплине «Защита информации»

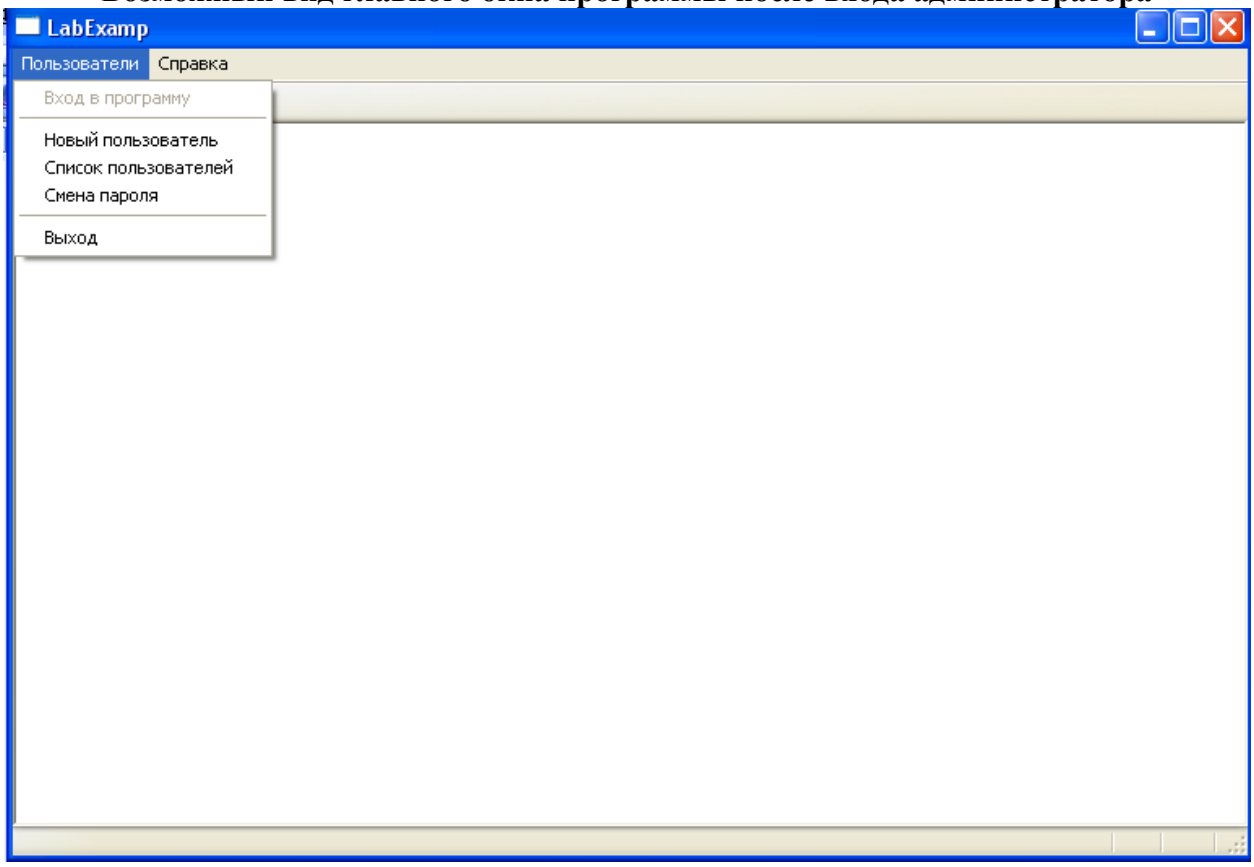
Лабораторная работа №1

1. После запуска системы программирования Microsoft Visual Studio создать новый проект Visual C++ MFC Application (с помощью мастера MFC Application Wizard). Выбрать на странице Application Type следующие опции создаваемого проекта (значения остальных опций на этой и остальных страницах не изменять) – Single Document и Use MFC in a static library, снять выключатель Document/View architecture support.
2. В окне созданного проекта открыть файл его ресурсов (с расширением rc), а в нем – созданное по умолчанию главное меню. Изменить его язык (свойство Language) на Русский. В подменю File перед командой Exit добавить команды Вход в программу (свойству ID задать значение ID_ENTER, свойству Prompt – Вход нового пользователя, свойству Enabled − False), разделитель (separator), Новый пользователь (ID − ID_NEW, Prompt – Создание новой учетной записи, Enabled − False), Список пользователей (ID − ID_LIST, Prompt – Редактирование списка учетных записей, Enabled − False), Смена пароля (ID − ID_CHANGE, Prompt – Задание нового пароля пользователя, Enabled − False) и разделитель. Команду Exit переименовать в Выход, а подменю File – в Пользователи.
   Подменю Edit и View удалить (с помощью клавиши Delete). Подменю Help переименовать в Справка, а его команду About – в О программе.

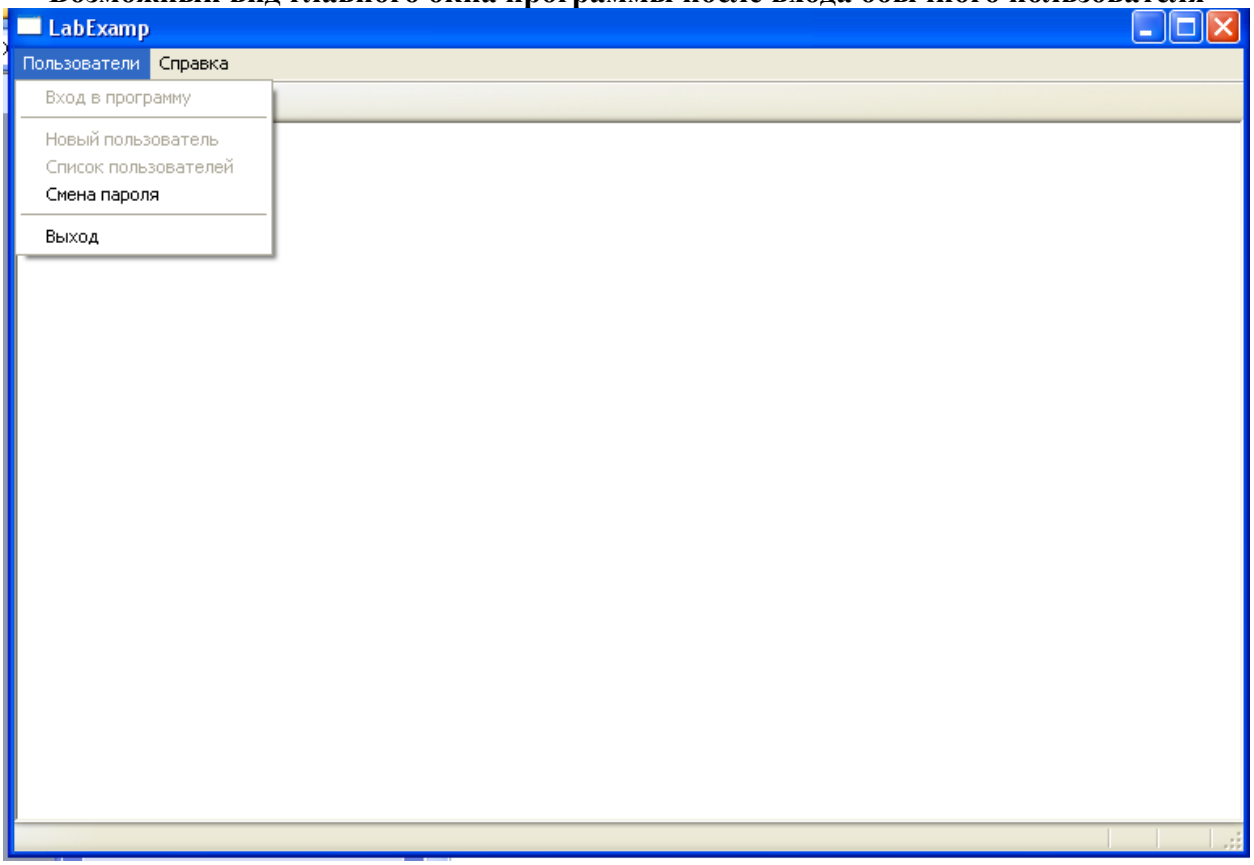
Возможный вид главного окна программы после ее запуска



Возможный вид главного окна программы после входа администратора



Возможный вид главного окна программы после входа обычного пользователя



3. С помощью окна ресурсов проекта добавить в него ресурс диалога (Dialog). Свойству Caption задать значение Вход в программу. С помощью окна элементов управления (View | Toolbox) добавить на созданное окно необходимые элементы

(см. приведенную ниже таблицу). Редактору для ввода пароля задать значение True свойству Password. Кнопку Cancel переименовать в Отмена.

Для созданного диалогового окна добавить класс CDlg1 в проект (команда контекстного меню Add class). С помощью окна просмотра классов проекта добавить в созданный класс поля (Variable), связанные с элементами управления окна (должен быть включен выключатель Control variable): Login (Control ID – IDC_EDIT1, Category – Value, Variable type – CString, Max chars - 20) и Password (Control ID – IDC_EDIT2, Category – Value, Variable type – CString, Max chars - 10).

С помощью окна просмотра классов проекта добавить в созданный автоматически класс приложения (его имя по умолчанию оканчивается на App) поля (Variable) типа bool: pAdmin, pEnter и pError.

Добавить в файл с исходным кодом класса приложения (по умолчанию именуется *имя проекта*.cpp) оператор

```
#include "Dlg1.h"
```

С помощью окна просмотра ресурсов проекта и редактора диалога Вход в программу добавить в класс CDlg1 метод для обработки нажатия на кнопку OK (Message type – BN_CLICKED) и ввести (с помощью буфера обмена) в этот метод код

```
UpdateData();
if(Login.GetLength()) OnOK();
else ::MessageBox(0,(LPCWSTR)_T("Имя не может быть пустым!"),
                  (LPCWSTR)_T("Вход в программу"),0);
```

4. Добавить в проект ресурс диалогового окна Смена пароля. Добавить в него необходимые элементы управления (см. таблицу). Редакторам для ввода пароля задать значение True свойству Password. Кнопку Cancel переименовать в Отмена.

   Для созданного диалогового окна добавить класс CDlg2 в проект. С помощью окна просмотра классов проекта добавить в созданный класс поля, связанные с элементами управления окна: Pass1 (Control ID – IDC_EDIT1, Category – Value, Variable type – CString, Max chars - 10) и Pass2 (Control ID – IDC_EDIT2, Category – Value, Variable type – CString, Max chars - 10).

   Добавить в файл с исходным кодом класса приложения оператор

   ```
   #include "Dlg2.h"
   ```

5. Добавить в проект ресурс диалогового окна Добавление пользователя. Добавить в него необходимые элементы управления (см. таблицу). Кнопку Cancel переименовать в Отмена.

   Для созданного диалогового окна добавить класс CDlg3 в проект. С помощью окна просмотра классов проекта добавить в созданный класс поле Login, связанное с элементом управления окна (Control ID – IDC_EDIT1, Category – Value, Variable type – CString, Max chars - 20).

   Добавить в файл с исходным кодом класса приложения оператор

   ```
   #include "Dlg3.h"
   ```

6. Добавить в проект ресурс диалогового окна Список пользователей. Добавить в него необходимые элементы управления (см. таблицу). Для редактора с именем пользователя установить в True свойство Read Only. Кнопку Cancel переименовать в Отмена.

   Для созданного диалогового окна добавить класс CDlg4 в проект. С помощью окна просмотра классов проекта добавить в созданный класс поля, связанные с элементами управления окна: UserName (Control ID – IDC_EDIT1, Category – Value, Variable type – CString), Block (Control ID – IDC_CHECK1, Category –Control, Variable type – CButton), Restrict (Control ID – IDC_CHECK2, Category –Control, Variable type – CButton), а также поле Next типа bool.

   Добавить в файл с исходным кодом класса приложения оператор

#include "Dlg4.h"

Добавляемые элементы создаваемых диалоговых окон

Тип и текст (свойство Caption) элемента	Свойство ID элемента
Окно входа в программу (IDD_DIALOG1)	
Надпись (Static Text) Имя	
Редактор (Edit Control) для ввода имени	IDC_EDIT1
Надпись (Static Text) Пароль	
Редактор (Edit Control) для ввода пароля	IDC_EDIT2
Окно смены пароля (IDD_DIALOG2)	
Надпись (Static Text) Введите пароль	
Редактор (Edit Control) для ввода пароля	IDC_EDIT1
Надпись (Static Text) Подтверждение пароля	
Редактор (Edit Control) для подтверждения ввода пароля	IDC_EDIT2
Окно добавления нового пользователя (IDD_DIALOG3)	
Надпись (Static Text) Имя пользователя	
Редактор (Edit Control) для ввода имени пользователя	IDC_EDIT1
Окно просмотра (редактирования) списка пользователей (IDD_DIALOG4)	
Редактор (Edit Control) для отображения имени пользователя	IDC_EDIT1
Выключатель (Check Box) Блокировка	IDC_CHECK1
Выключатель (Check Box) Ограничение на пароль	IDC_CHECK2
Кнопка (Button) Следующий	IDC_BUTTON1
Кнопка (Button) Сохранить	IDC_BUTTON2

7. С помощью команды File | Save All сохранить созданный проект и все его файлы.
8. С помощью буфера обмена добавить определения в заголовочный (.h) файл приложения перед определением класса приложения (оператором class):

```
#include <fstream>
// максимальная длина имени учетной записи
#define MAXNAME 20
// максимальная длина пароля
#define MAXPASS 10
// имя файла с учетными записями пользователей
#define SECFILE _T("security.db")
// структурный тип для хранения учетной записи
struct AccountType
{
wchar_t  UserName[MAXNAME]; // имя
int PassLen; // длина пароля
wchar_t  UserPass[MAXPASS]; // пароль
bool Block; // признак блокировки учетной записи администратором
// признак включения администратором ограничений на выбираемые пользователями
//пароли
bool Restrict;
};
extern AccountType UserAcc; // структура для хранения одной учетной записи
// файловая переменная для чтения и записи в файл с учетными записями
using namespace std;
extern fstream AccFile;
extern unsigned RecCount; // номер текущей учетной записи
```

9. Добавить определения глобальных констант и переменных в файл приложения (.cpp) перед операторами с реализацией методов класса приложения:

```
// структура для хранения одной учетной записи
AccountType UserAcc;
```

```
// файловая переменная для чтения и записи в файл с учетными записями
fstream AccFile;
// номер текущей учетной записи
unsigned RecCount;
```

10. С помощью окна просмотра классов проекта добавить программный код в метод InitInstance класса приложения (перед заключительным оператором return):

```
// если файл с учетными записями пользователей не существует (первый запуск
//программы)
CFile tmpf;
if(!tmpf.Open(SECFILE,CFile::modeRead))
  { AccFile.open(SECFILE,ios::out|ios::binary); // создание нового файла
    // подготовка учетной записи администратора
   lstrcpy(UserAcc.UserName,_T("ADMIN"));
   lstrcpy(UserAcc.UserPass,_T(""));
   UserAcc.PassLen=0;
   UserAcc.Block=false;
   UserAcc.Restrict=true;
   // запись в файл
   AccFile.write((const char*)&UserAcc,sizeof(UserAcc));
   // закрытие файла
   AccFile.close(); }
```

11. С помощью окна просмотра ресурсов проекта и редактора меню добавить два обработчика (Event Handler) для команды Вход в программу, поместив соответствующие методы (с помощью списка Class list) в класс приложения (свойству Message type одного обработчика задать UPDATE COMMAND UI, другого - COMMAND).
    После открытия окна редактирования метода с типом UPDATE COMMAND UI ввести код
    pCmdUI->Enable(!pError && !pEnter);
    После открытия окна редактирования метода с типом COMMAND ввести код

```
while(true)
try {
// запрос и проверка имени учетной записи и пароля
  static unsigned EnterCount=0; // счетчик попыток входа в программу
  CDlg1 enterDlg; // диалог входа
  CDlg2 passDlg; // диалог смены пароля
// отображение формы для ввода имени и пароля
  if(enterDlg.DoModal()==IDOK)
{       // если повторная попытка входа, то сброс признака конца файла и его закрытие
    if(AccFile.is_open())
    { AccFile.clear();
      AccFile.close(); }
      // открытие файла для чтения и записи
      AccFile.open(SECFILE,ios::in|ios::out|ios::binary);
      // сброс номера текущей учетной записи
      RecCount=0;
// чтение учетных записей и сравнение имен из них с введенным пользователем именем
      while(!AccFile.eof())
      { AccFile.read((char*)&UserAcc,sizeof(UserAcc));
        RecCount++;
        // прекращение чтения из файла, если обнаружено совпадение имен
        if(enterDlg.Login==UserAcc.UserName) break; }
```

```
        // если совпадения не найдено (достигнут конец файла)
        if(AccFile.eof())
       // генерация исключительной ситуации (пользователь не зарегистрирован)
          throw(_T("Вы не зарегистрированы!"));
       // если пароль отсутствует (первый вход пользователя в программу)
       else if(!UserAcc.PassLen)
         // отображение формы для ввода (смены) пользователем пароля
         if(passDlg.DoModal()==IDOK)
         {   // смещение к началу текущей учетной записи в файле
            AccFile.seekp((RecCount-1)*sizeof(UserAcc),ios::beg);
            // добавление введенного пароля и его длины в учетную запись
            lstrcpy(UserAcc.UserPass,(LPCWSTR)passDlg.Pass1);
            UserAcc.PassLen=passDlg.Pass1.GetLength();
             // запись в файл
            AccFile.write((const char*)&UserAcc,sizeof(UserAcc)); }
          // если пользователь не ввел пароль, то выход из функции
          else return;
         // если пользователь уже имел пароль
         else
         { // сравнение пароля из учетной записи и введенного пароля
          if(lstrcmp(UserAcc.UserPass,(LPCWSTR)enterDlg.Password))
           // если пароли не совпадают и число попыток превысило 2
           if(++EnterCount>2)
           { // блокировка команды «Вход»
             pError=true;
        // генерация исключительной ситуации (вход в программу невозможен)
            throw(_T("Вход в программу невозможен!")); }
           // если пароли не совпадают и число попыток не превысило 2
           else
    // генерация исключительной ситуации (пользователь ввел неверный пароль)
             throw(_T("Неверный пароль!"));
          // если пароли совпадают, то продолжение работы
          else; }
         // если учетная запись заблокирована администратором
         if(UserAcc.Block)
           // генерация исключительной ситуации (пользователь заблокирован)
           throw(_T("Вы заблокированы!"));
         // проверка полномочий пользователя
         // если пользователь является администратором
         if(!lstrcmp(UserAcc.UserName,_T("ADMIN")))
{// снятие блокировки с команд меню «Все пользователи» и «Новый пользователь»
          pAdmin=true;
          // закрытие файла с учетными записями
          AccFile.close(); }
  // снятие блокировки с команды меню «Смена пароля» (для всех пользователей)
         // блокировка команды «Вход»
                pEnter=true;
          // выход из функции
                return;}
          // выход из функции, если пользователь отказался от входа
          else return;
          }
```

```
        catch(LPCWSTR s)  { AfxMessageBox(s);
// выход из функции, если превышено пороговое число попыток входа
        if(!strcmp((const char*)s,  (const char*)_T( "Вход в программу невозможен!")))
              return;}
```

12. Добавить в класс CDlg2 (диалог Смена пароля) метод CheckPassword (Return type – bool, Parameter type − const CString&, Parameter name − Pass). Затем добавить в этот метод код в зависимости от номера варианта (ограничений на выбираемый пользователем пароль).

```
// пример функции для проверки ограничений на введенный пользователем пароль (Pass)
// (проверяется наличие в пароле строчных и прописных букв, цифр и знаков препинания)
 // признаки наличия в пароле требуемых групп символов
 bool High=false,Low=false,Digit=false,Punct=false;
 for(int i=0;i<Pass.GetLength();i++)
 {
```

/* проверка принадлежности очередного символа пароля одной из требуемых групп и изменение соответствующего признака; если в проверке участвуют константы символов кириллицы, то перед ними надо размещать макрос _T, например _T('Я') */

```
   High|=IsCharUpper(Pass[i]);
   Low|=IsCharLower(Pass[i]);
   Digit|=iswdigit(Pass[i]);
   Punct|=iswpunct(Pass[i]);
 }
 // формирование результата выполнения функции проверки
 return High && Low && Digit && Punct;
```

13. Добавить в класс CDlg2 метод для обработки нажатия на кнопку OK (Message type – BN_CLICKED) и ввести в этот метод код

```
UpdateData();
if(Pass1==Pass2)
{       if(UserAcc.Restrict && !CheckPassword(Pass1))
    {// вывод сообщения об ошибке нарушения ограничений на пароль
      AfxMessageBox((LPCWSTR)_T("Пароль не соответствует ограничениям!"));
      // установка фокуса ввода на редактор для ввода пароля
        GetDlgItem(IDC_EDIT1)->SetFocus(); }
    else OnOK();      }
// вывод сообщения об ошибке несовпадения пароля с его подтверждением
else AfxMessageBox((LPCWSTR)_T("Пароли должны совпадать!"));
```

14. Добавить два обработчика для команды Смена пароля, поместив соответствующие методы в класс приложения (свойству Message type одного обработчика задать UPDATE COMMAND UI, другого - COMMAND).

После открытия окна редактирования метода с типом UPDATE COMMAND UI ввести код

```
pCmdUI->Enable(pEnter);
```

После открытия окна редактирования метода с типом COMMAND ввести код

```
// если программа в режиме администратора
  if(pAdmin)
  {// открытие файла с учетными записями для чтения и записи
    AccFile.open(SECFILE,ios::in|ios::out|ios::binary);
    // сброс номера текущей учетной записи
    RecCount=0;
    // чтение учетной записи администратора (первой учетной записи)
    AccFile.read((char*)&UserAcc,sizeof(UserAcc));
    RecCount++; }
```

```
  // отображение формы для смены пароля
  CDlg2 passDlg;
  if(passDlg.DoModal()==IDOK)
  {// смещение к началу текущей учетной записи в файле
    AccFile.seekp((RecCount-1)*sizeof(UserAcc),ios::beg);
    // помещение в учетную запись нового пароля и его длины
    lstrcpy(UserAcc.UserPass,(LPCWSTR)passDlg.Pass1);
    UserAcc.PassLen=passDlg.Pass1.GetLength();
    // запись в файл
    AccFile.write((const char*)&UserAcc,sizeof(UserAcc)); }
  // если программа в режиме администратора, то закрытие файла
       if(pAdmin) AccFile.close();
```

15. Добавить в класс CDlg3 метод для обработки нажатия на кнопку OK (Message type – BN_CLICKED) и ввести в этот метод код

```
UpdateData();
if(Login!="")
          {         // смещение к началу файла с учетными записями
                    AccFile.seekg(0,ios::beg);
// чтение учетных записей из файла для проверки уникальности введенного имени
                    while(!AccFile.eof())
                    { AccFile.read((char*)&UserAcc,sizeof(UserAcc));
// если учетная запись с введенным именем уже существует, то прекращение чтения
                    if(Login==UserAcc.UserName) break; }
// если учетная запись с введенным именем уже существует (не достигнут конец файла)
                    if(!AccFile.eof())
                    {// вывод сообщения об ошибке
                    AfxMessageBox(CString("Пользователь ")+Login+
          _T("\nуже зарегистрирован!")); }
                    // если ошибок нет, то сброс состояния «конец файла»
                    else
                    {         AccFile.clear();
                              OnOK();         }         }
// если не введено имя добавляемого пользователя
          else AfxMessageBox((LPCWSTR)_T("Имя не может быть пустым!"));
```

16. Добавить два обработчика для команды Новый пользователь, поместив соответствующие методы в класс приложения (свойству Message type одного обработчика задать UPDATE COMMAND UI, другого - COMMAND).

После открытия окна редактирования метода с типом UPDATE COMMAND UI ввести код

```
pCmdUI->Enable(pAdmin && pEnter);
```

После открытия окна редактирования метода с типом COMMAND ввести код

```
// открытие файла с учетными записями для чтения и записи
  AccFile.open(SECFILE,ios::in|ios::out|ios::binary);
  // отображение окна добавления нового пользователя
  CDlg3 newDlg;
  if(newDlg.DoModal()==IDOK)
  {// смещение к концу файла
    AccFile.seekp(0,ios::end);
    // сохранение в учетной записи введенного имени пользователя
    lstrcpy(UserAcc.UserName,(LPCWSTR)newDlg.Login);
    // сохранение в новой учетной записи пустого пароля
    lstrcpy(UserAcc.UserPass,_T(""));
```

```
    UserAcc.PassLen=0;
    // установка признака отсутствия блокирования новой учетной записи
    UserAcc.Block=false;
    // установка признака ограничений на выбираемые пароли
    UserAcc.Restrict=true;
    // запись в файл
    AccFile.write((const char*)&UserAcc,sizeof(UserAcc)); }
  // закрытие файла
       AccFile.close();
```

17. Добавить в класс CDlg4 метод для обработки нажатия на кнопку Следующий (Message type – BN_CLICKED) и ввести в этот метод код

```
char ch; // вспомогательная символьная переменная
 // чтение учетной записи из файла
 AccFile.read((char*)&UserAcc,sizeof(UserAcc));
 // увеличение номера текущей учетной записи
 RecCount++;
 // отображение имени учетной записи
 UserName=UserAcc.UserName;
 // отображение признака блокировки учетной записи
 // отображение признака установленных ограничений на выбираемые пароли
 UpdateData(FALSE);
 // попытка чтения следующей учетной записи
 AccFile.read(&ch,1);
 // если текущая учетная запись является последней
 if(AccFile.eof())
 {   // очистка состояния файла от признака конца файла
     AccFile.clear();
     // блокировка кнопки «Следующий»
     Next=false;
     UpdateData(); }
 // если конец файла не достигнут, то смещение к началу следующей учетной записи
 else AccFile.seekg(RecCount*sizeof(UserAcc),ios::beg);
```

18. Добавить в класс CDlg4 метод для обработки нажатия на кнопку Сохранить (Message type – BN_CLICKED) и ввести в этот метод код

```
// сохранение в учетной записи сделанных администратором изменений
 UpdateData();
  // смещение к началу редактируемой учетной записи в файле
 AccFile.seekp((RecCount-1)*sizeof(UserAcc),ios::beg);
  // запись в файл
 AccFile.write((const char*)&UserAcc,sizeof(UserAcc));
  // смещение для чтения следующей учетной записи
 AccFile.seekg(RecCount*sizeof(UserAcc),ios::beg);
```

19. Открыть для редактирования метод DoDataExchange в классе CDlg4 и ввести в него код, заменив им код, созданный автоматически:

```
CDialog::DoDataExchange(pDX);
DDX_Text(pDX, IDC_EDIT1, UserName);
// отображение признака блокировки учетной записи
DDX_Control(pDX, IDC_CHECK1, Block);
if(pDX->m_bSaveAndValidate)
        UserAcc.Block=(Block.GetCheck()==BST_CHECKED);
else if(UserAcc.Block) Block.SetCheck(BST_CHECKED);
else Block.SetCheck(BST_UNCHECKED);
```

```
// отображение признака установленных ограничений на выбираемые пароли
DDX_Control(pDX, IDC_CHECK2, Restrict);
if(pDX->m_bSaveAndValidate)
        UserAcc.Restrict=(Restrict.GetCheck()==BST_CHECKED);
else if(UserAcc.Restrict) Restrict.SetCheck(BST_CHECKED);
else Restrict.SetCheck(BST_UNCHECKED);
GetDlgItem(IDC_BUTTON1)->EnableWindow(Next);
```

20. Добавить два обработчика для команды Список пользователей, поместив соответствующие методы в класс приложения (свойству Message type одного обработчика задать UPDATE COMMAND UI, другого - COMMAND).

После открытия окна редактирования метода с типом UPDATE COMMAND UI ввести код

```
pCmdUI->Enable(pAdmin && pEnter);
```

После открытия окна редактирования метода с типом COMMAND ввести код

```
CDlg4 dlg;
char ch; // вспомогательная символьная переменная
// открытие файла с учетными записями для чтения и записи
AccFile.open(SECFILE,ios::in|ios::out|ios::binary);
// сброс номера текущей учетной записи
RecCount=0;
// чтение первой учетной записи
AccFile.read((char*)&UserAcc,sizeof(UserAcc));
RecCount++;
// отображение имени учетной записи
dlg.UserName=UserAcc.UserName;
// попытка чтения следующей учетной записи
AccFile.read(&ch,1);
// если следующей учетной записи нет, то блокирование кнопки «Следующий» в окне
//просмотра (редактирования) учетных записей
dlg.Next=!AccFile.eof();
// если достигнут конец файла с учетными записями
if(AccFile.eof())
{ // сброс состояния «конец файла»
  AccFile.clear();
  // смещение к началу первой учетной записи
  AccFile.seekg(0,ios::beg); }
// если конец файла не достигнут, то смещение к началу следующей учетной записи
else AccFile.seekg(sizeof(UserAcc),ios::beg);
// отображение окна просмотра (редактирования) учетных записей
dlg.DoModal();
// закрытие файла
AccFile.close();
```

21. Отредактировать автоматически созданный ресурс диалога IDD_ABOUTBOX, задав его свойству Language значение Русский, а для нижней надписи установить свойство Caption в значение © *Фамилия И.О., Год выполнения работы*.

Лабораторная работа №2

1. Добавить в проект ресурс диалогового окна Расшифрование базы учетных записей. Добавить в него необходимые элементы управления (см. таблицу). Редактору для ввода пароля задать значение True свойству Password. Кнопку Cancel переименовать в Отмена.

   Для созданного диалогового окна добавить класс CDlg5 в проект. С помощью окна просмотра классов проекта добавить в созданный класс поле PassFrase, связанное с

редактором для ввода пароля (Control ID – IDC_EDIT1, Category – Value, Variable type – CString, Max chars - 20).

Добавить в файл с исходным кодом класса приложения оператор

```
#include "Dlg5.h"
```

Добавляемые элементы управления диалогового окна

Тип и текст (свойство Caption) элемента	Свойство ID элемента
Окно ввода пароля для расшифрования базы учетных записей (IDD_DIALOG5)	
Надпись (Static Text) Пароль для расшифрования	
Редактор (Edit Control) для ввода пароля	IDC_EDIT1

2. Дополнительные определения глобальных констант и переменных в заголовочном файле приложения (по умолчанию его имя совпадает с именем проекта):

```
// имя временного (расшифрованного) файла с учетными записями
#define TMPFILE "temp.db"
// вспомогательные файловые переменные
extern fstream TmpFile1,TmpFile2;
// буфер для чтения и записи в файл
extern BYTE Buf[128];
// дескриптор криптопровайдера
extern HCRYPTPROV hProv;
// дескриптор ключа шифрования (расшифрования)
extern HCRYPTKEY hKey;
// дескриптор хеш-значения
extern HCRYPTHASH hHash;
```

3. Дополнительные определения глобальных переменных в cpp-файле приложения:

```
// вспомогательные файловые переменные
fstream TmpFile1,TmpFile2;
// буфер для чтения и записи в файл
BYTE Buf[128];
// дескриптор криптопровайдера
HCRYPTPROV hProv=0;
// дескриптор ключа шифрования (расшифрования)
HCRYPTKEY hKey;
// дескриптор хеш-значения
HCRYPTHASH hHash;
```

4. Добавить в файл stdafx.h (после подключения файла afxext.h) оператор

```
#include <wincrypt.h>
```

5. Дополнительный код в методе InitInstance класса приложения:

```
 DWORD Param=CRYPT_MODE_CBC, // режим шифрования (расшифрования)
                 Len; // длина блока данных
 // создание формы для ввода парольной фразы
 CDlg5 dlg;
 // блок с возможной генерацией исключительных ситуаций
 try
 {// получение дескриптора криптопровайдера
       if(!CryptAcquireContext(&hProv,NULL,NULL,PROV_RSA_FULL,0))
  // если пользователь еще не зарегистрирован в криптопровайдере, то создание для него
//контейнера ключей
       if((unsigned)GetLastError()==NTE_BAD_KEYSET)
    CryptAcquireContext(&hProv,NULL,NULL,PROV_RSA_FULL,CRYPT_NEWKEYSET);
   // если доступ к криптопровайдеру невозможен, то генерация исключительной ситуации
    else throw _T("Ошибка при доступе к CryptoAPI!");
   // отображение формы для ввода парольной фразы для расшифрования файла Form6
```

```
  if(dlg.DoModal()!=IDOK)
   // если парольная фраза не введена, то генерация исключительной ситуации
   throw _T("Работа программы невозможна!");
   // создание пустого хеш-значения
  CryptCreateHash(hProv,CALG_SHA,0,0,&hHash);
   // хеширование введенной пользователем парольной фразы
   CryptHashData(hHash,(const BYTE*)(LPCWSTR) dlg.PassFrase, dlg.PassFrase.
         GetLength(),0);
   // генерация ключа расшифрования из хеш-значения парольной фразы
  CryptDeriveKey(hProv,CALG_RC2,hHash,CRYPT_EXPORTABLE,&hKey);
   // установка режима расшифрования
  CryptSetKeyParam(hKey,KP_MODE,(BYTE*)&Param,0);
   // разрушение хеш-значения
  CryptDestroyHash(hHash); }
 // обработка исключительных ситуаций
 catch(LPCWSTR  e)
 {  // освобождение дескриптора криптопровайдера
  if(hProv) CryptReleaseContext(hProv,0);
   // вывод сообщения об ошибке
  AfxMessageBox(e);
   // завершение работы программы
  exit(-1);
  return FALSE; }
// далее в программе при указании имени расшифрованного файла с учетными записями
//пользователей следует использовать константу TMPFILE вместо константы SECFILE
. . .
// если файл с учетными записями существует (второй и последующие запуски
//программы), то он должен быть расшифрован
 else
 { tmpf.Close();
       // открытие зашифрованного файла для чтения
  TmpFile1.open(SECFILE,ios::in|ios::binary);
  // создание временного файла
  TmpFile2.open(TMPFILE,ios::out|ios::binary);
  // цикл расшифрования данных и записи их во временный файл
  do
  { // чтение порции зашифрованных данных из файла
   TmpFile1.read((char*)Buf,sizeof(Buf));
   // получение фактической длины прочитанных данных
   Len=TmpFile1.gcount();
   // расшифрование прочитанных данных
   CryptDecrypt(hKey,0,TmpFile1.eof(),0,Buf,&Len);
   // запись во временный файл
   TmpFile2.write((const char*)Buf,Len); }
  while(!TmpFile1.eof());
  // сброс признака конца файла
  TmpFile1.clear();
  // закрытие файлов
  TmpFile1.close();
  TmpFile2.close();
  // проверка правильности расшифрования файла с учетными записями
  // открытие временного файла для чтения
```

```
  TmpFile2.open(TMPFILE,ios::in|ios::binary);
  // чтение первой учетной записи (администратора)
  TmpFile2.read((char*)&UserAcc,sizeof(UserAcc));
  // закрытие временного файла
  TmpFile2.close();
  // если имя первой учетной записи не совпадает с ADMIN
  if(lstrcmp(UserAcc.UserName,_T("ADMIN")))
  { // разрушение ключа расшифрования
    CryptDestroyKey(hKey);
     // освобождение дескриптора криптопровайдера
    CryptReleaseContext(hProv,0);
    // вывод сообщения об ошибке
    AfxMessageBox(_T("Неверный ключ расшифрования!"));
    // удаление временного файла
    remove(TMPFILE);
    // завершение работы программы
    exit(-1);
    return FALSE; } }
```

6. Добавить код в деструктор класса главного окна программы (по умолчанию этот класс получает имя CMainFrame):

```
DWORD Len; // длина блока данных
// открытие временного файла для чтения
if(AccFile.is_open()) AccFile.close();
TmpFile2.open(TMPFILE,ios::in|ios::binary);
// создание нового зашифрованного файла с учетными записями
TmpFile1.open(SECFILE,ios::out|ios::binary);
// цикл шифрования данных и записи их в файл
do
{// чтение порции данных из временного файла
  TmpFile2.read((char*)Buf,sizeof(Buf));
  // получение фактической длины прочитанных данных
  Len=TmpFile2.gcount();
  // шифрование данных
  CryptEncrypt(hKey,0,TmpFile2.eof(),0,Buf,&Len,sizeof(Buf));
  // запись зашифрованных данных во вновь созданный файл
  TmpFile1.write((const char*)Buf,Len); }
while(!TmpFile2.eof());
// закрытие файлов
TmpFile1.close();
TmpFile2.close();
// удаление временного файла
remove(TMPFILE);
// разрушение ключа шифрования
CryptDestroyKey(hKey);
// освобождение дескриптора криптопровайдера
CryptReleaseContext(hProv,0);
```