МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ФГБО УВПО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)»

ИНСТИТУТ МЕНЕДЖМЕНТА И ЭКОНОМИКИ В ЭНЕРГЕТИКЕ И ПРОМЫШЛЕННОСТИ

Кафедра «Прикладная и бизнес-информатика»

**ОТЧЕТ ПО Расчетному заданию**

по дисциплине «Информационная безопасность»

Студент группы \_\_Иэ–62–13 \_\_\_\_\_ Смыслина А.И.\_

(Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_Хорев.П.Б.

(Ф.И.О.)

Москва

2015

Оглавление

[1. Расчетное задание. Часть 1 3](#_Toc436871780)

[1.1 Содержание задания 3](#_Toc436871781)

[1.2 Индивидуальное задание. 11 вариант 4](#_Toc436871782)

[Исходный код функции для проверки ограничений на выбранный пароль 4](#_Toc436871783)

[1.3 Экранные формы 5](#_Toc436871784)

[2. Расчетное задание. Часть 2 9](#_Toc436871785)

[2.1 Содержание задания 10](#_Toc436871786)

[2.2 Исходный код фрагмента программы 11](#_Toc436871787)

[2.3 Содержимое зашифрованного файла с учетными записями 14](#_Toc436871788)

# Расчетное задание. Часть 1

Разработка программы разграничения полномочий пользователей на основе парольной аутентификации

## Содержание задания

1. Программа должна обеспечивать работу в двух режимах: администратора (пользователя с фиксированным именем ADMIN) и обычного пользователя.
2. В режиме администратора программа должна поддерживать следующие функции (при правильном вводе пароля):
3. смена пароля администратора (при правильном вводе старого пароля);
4. просмотр списка имен зарегистрированных пользователей и установленных для них параметров (блокировка учетной записи, включение ограничений на выбираемые пароли) – всего списка целиком в одном окне или по одному элементу списка с возможностью перемещения к его началу или концу;
5. добавление уникального имени нового пользователя к списку с пустым паролем (строкой нулевой длины);
6. блокирование возможности работы пользователя с заданным именем;
7. включение или отключение ограничений на выбираемые пользователем пароли (в соответствии с индивидуальным заданием, определяемым номером варианта);
8. завершение работы с программой.
9. В режиме обычного пользователя программа должна поддерживать только функции смены пароля пользователя (при правильном вводе старого пароля) и завершения работы, а все остальные функции должны быть заблокированы.
10. После своего запуска программа должна запрашивать у пользователя в специальном окне входа ввод его имени и пароля. При вводе пароля его символы всегда должны на экране заменяться символом ‘\*’.
11. При отсутствии введенного в окне входа имени пользователя в списке зарегистрированных администратором пользователей программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность повторного ввода имени или завершения работы с программой.
12. При неправильном вводе пароля программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность повторного ввода. При трехкратном вводе неверного пароля работа программы должна завершаться.
13. При первоначальном вводе пароля (обязательном при первом входе администратора или пользователя с зарегистрированным ранее администратором именем) и при дальнейшей замене пароля программа должна просить пользователя подтвердить введенный пароль путем его повторного ввода.
14. Если выбранный пользователем пароль не соответствует требуемым ограничениям (при установке соответствующего параметра учетной записи пользователя), то программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность ввода другого пароля, завершения работы с программой (при первом входе данного пользователя) или отказа от смены пароля.
15. Информация о зарегистрированных пользователях, их паролях, отсутствии блокировки их работы с программой, а также включении или отключении ограничений на выбираемые пароли должна сохраняться в специальном файле. При первом запуске программы этот файл должен создаваться автоматически и содержать информацию только об администраторе, имеющем пустой пароль.
16. Интерфейс с программой должен быть организован на основе меню, обязательной частью которого должно являться подменю «Справка» с командой «О программе». При выборе этой команды должна выдаваться информация об авторе программы и выданном индивидуальном задании. Интерфейс пользователя программы может также включать панель управления с дублирующими команды меню графическими кнопками и строку состояния.
17. Для реализации указанных в пунктах 2-3 функций в программе должны использоваться специальные диалоговые формы, позволяющие пользователю (администратору) вводить необходимую информацию.

## Индивидуальное задание. 11 вариант

Ограничение на выбираемый пароль:

Наличие латинских букв, символов кириллицы и знаков препинания.

## Исходный код функции для проверки ограничений на выбранный пароль

bool CheckPassword(const AnsiString& Pass)

{

// признаки наличия в пароле требуемых групп символов

bool Isalpha=false,Punct=false,Cyr=false;

for(int i=1;i<=Pass.Length();i++)

{ /\* проверка принадлежности очередного символа пароля \*/

Isalpha|=isalpha(Pass[i]); /\*требования к наличию латинских букв

Punct|=ispunct(Pass[i]); /\*требования к наличию знаков пунктуации

if /\* проверка наличия в пароле букв отличных от латинских (в данном случае кириллица)

(IsCharAlpha(Pass[i])&&

!isalpha(Pass[i]) )

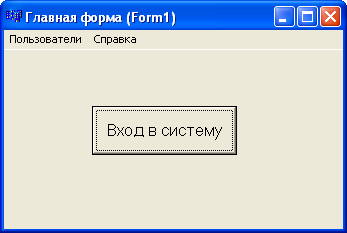
Cyr=true; }

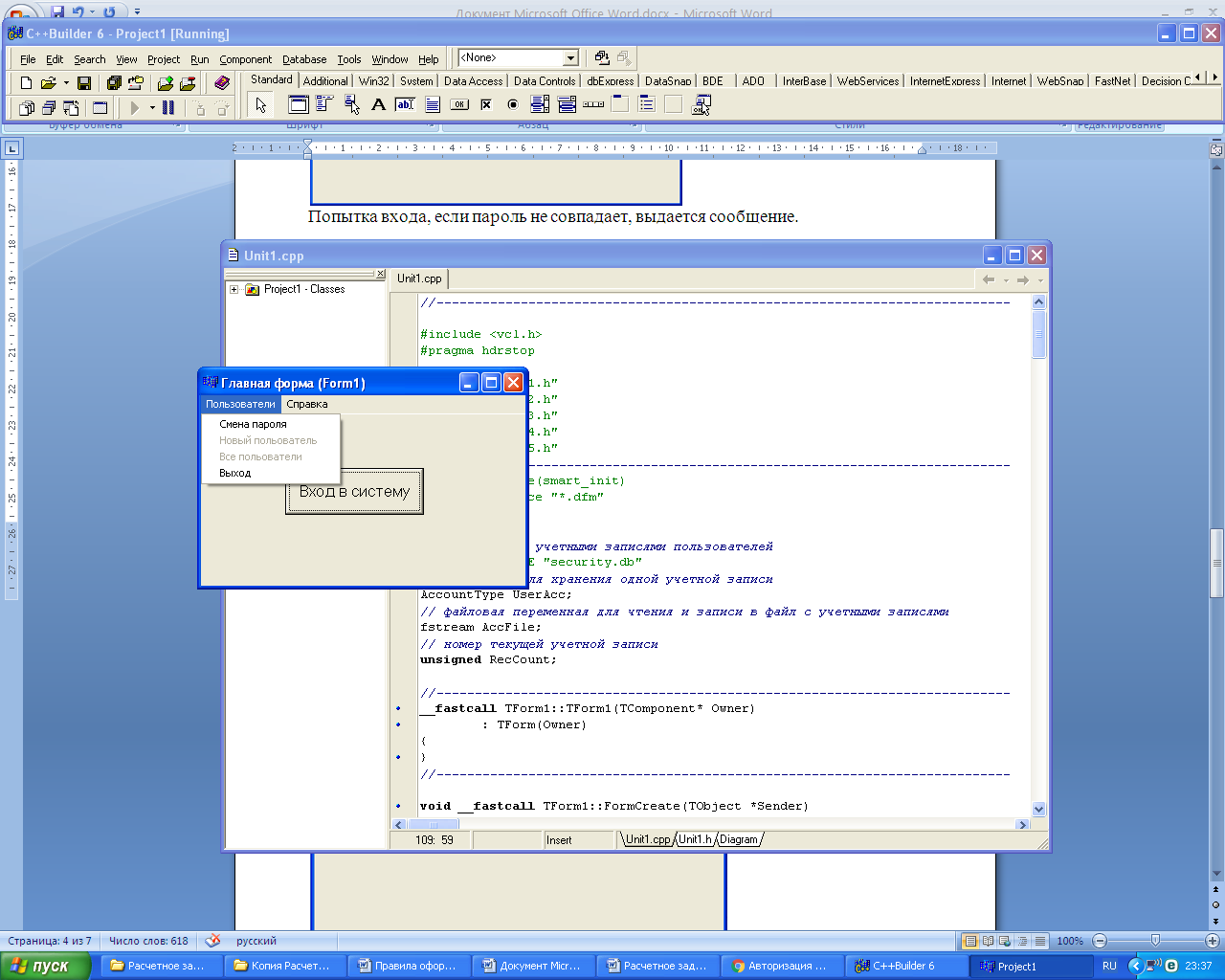
// формирование результата выполнения функции проверки

return Isalpha && Punct && Cyr;}

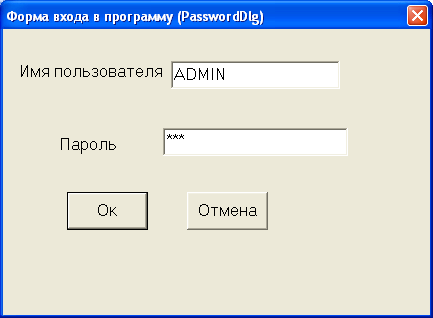
## Экранные формы

Главная форма «Вход в систему»

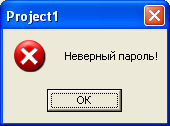




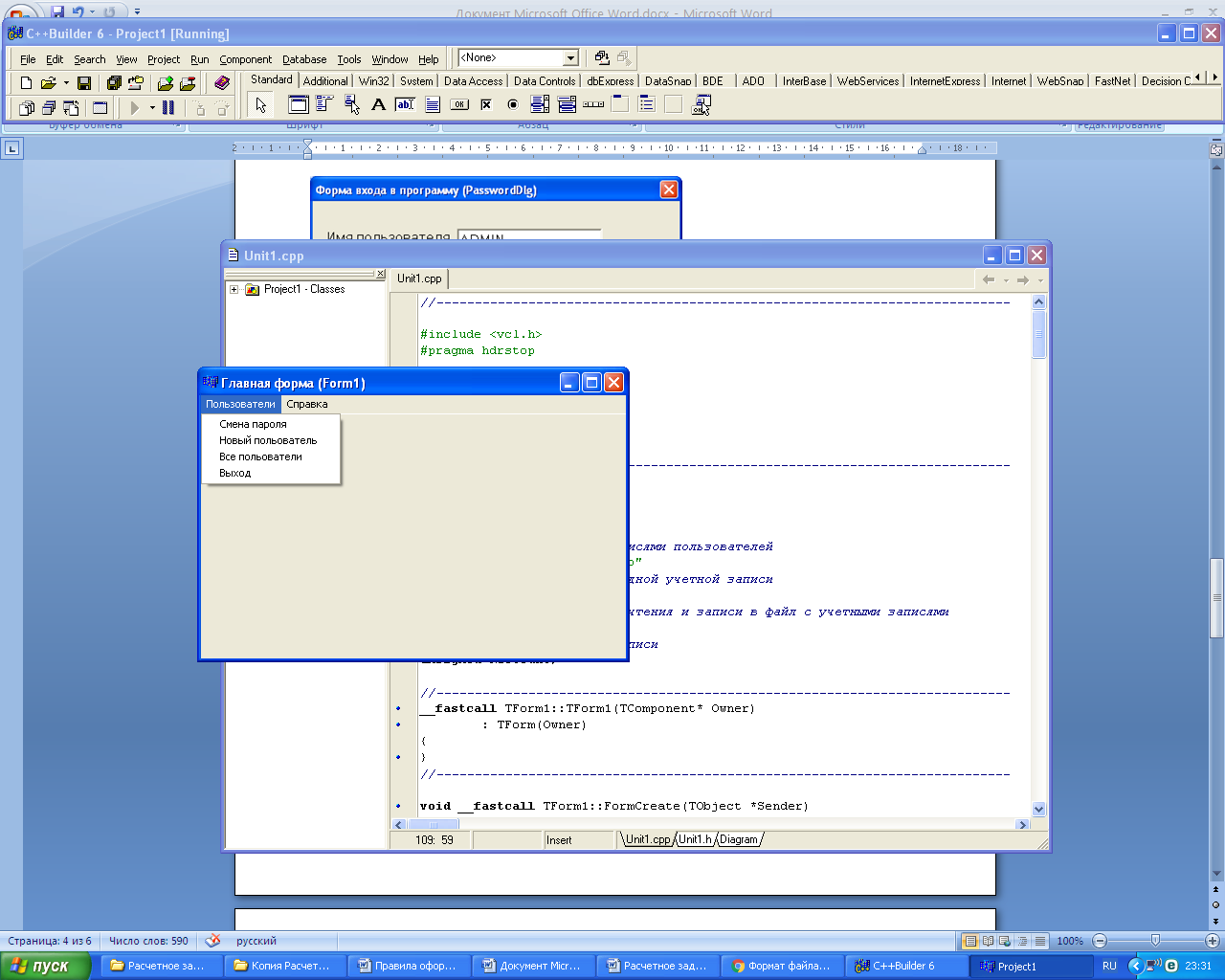
При нажатии на кнопку:



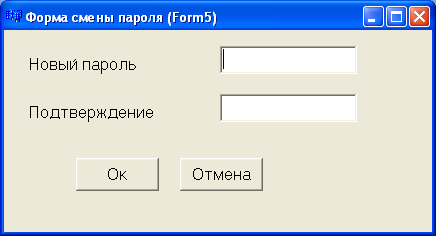
Попытка входа, если пароль не совпадает, выдается сообщение.



При верном введении пароля, кнопка пропадает, доступными становятся раннее скрытые функции.



Форма смены пароля



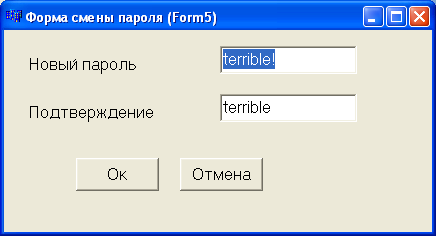
11 вариант задания

Наличие латинских букв, символов кириллицы и знаков препинания.

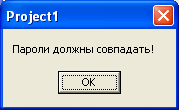
Программа примет пароль, только когда все условия будут выполнены.

Протестируем данную программу, и введем сочетания, в которых условия задания выполняются частично.

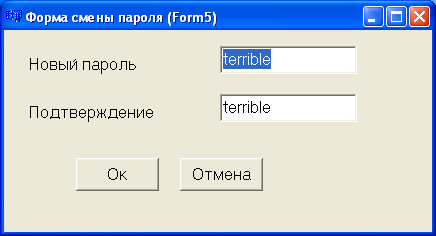
Для начала попробуем ввести отличные пароли в полях

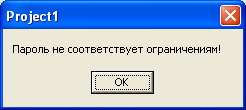


Программа выдала соответствующую ошибку

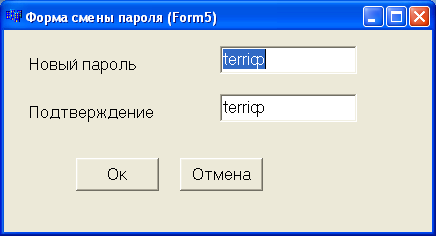


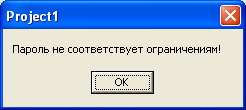
Далее введем пароль, состоящий только из латинских букв



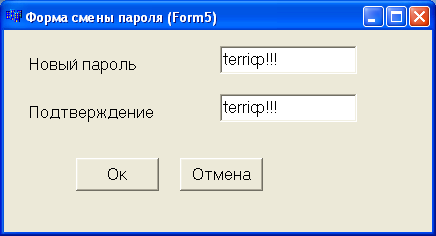


Затем попробуем ввести пароль содержащие латинские буквы и символы кириллицы



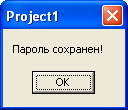


При введении пароля содержащие все типы символов, описанные в функции



Программа принимает данный пароль.

Соответствующее сообщение выводится на экран.



# Расчетное задание. Часть 2

***Использование функций криптографического интерфейса Windows для защиты информации***

## Содержание задания

1. В программу, разработанную при выполнении части 1 расчетного задания, добавить средства защиты от несанкционированного доступа к файлу с учетными данными зарегистрированных пользователей.
2. Файл с учетными данными должен быть зашифрован при помощи функций криптографического интерфейса операционной системы Windows (CryptoAPI) с использованием сеансового ключа, генерируемого на основе вводимой администратором (пользователем) парольной фразы.
3. При запуске программы файл с учетными данными должен расшифровываться во временный файл (или в файл в оперативной памяти), который после завершения работы программы должен быть снова зашифрован для отражения возможных изменений в учетных записях пользователей. «Старое» содержимое файла учетных записей при этом стирается.
4. После ввода парольной фразы при запуске программы, генерации ключа расшифрования и расшифрования файла с учетными данными зарегистрированных пользователей правильность введенной парольной фразы определяется по наличию в расшифрованном файле учетной записи администратора программы.
5. При вводе неправильной парольной фразы или отказе от ее ввода работа программы должна завершаться с выдачей соответствующего сообщения.
6. Временный файл на диске с расшифрованными учетными данными после завершения работы программы удаляется.
7. Варианты использования алгоритмов шифрования и хеширования выбираются в соответствии с выданным преподавателем заданием.
8. Для доступа к функциям CryptoAPI из программ на Паскале следует использовать интерфейсный модуль wincrypt.pas с указанного преподавателем сетевого диска*.*

## Индивидуальное задание. 11 вариант

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип симметричного шифрования | Используемый режим шифрования | Добавление к ключу случайного значения | Используемый алгоритм хеширования |
| 11 | Блочный | Обратная связь по шифротексту | Да | MD4 |

## Исходный код фрагмента программы

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

DWORD Param=CRYPT\_MODE\_CFB, // режим шифрования (расшифрования) -

// **обратная связь по шифротексту**

Len; // длина блока данных

// создание формы для ввода парольной фразы

TForm6\* Form6=new TForm6(Application);

// 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 Exception("Ошибка при доступе к CryptoAPI!");

// отображение формы для ввода парольной фразы для расшифрования файла Form6

if(Form6->ShowModal()!=mrOk)

// если парольная фраза не введена, то генерация исключительной ситуации

throw Exception("Работа программы невозможна!");

// создание пустого хеш-значения, **используемый алгоритм хеширования MD4**

CryptCreateHash(hProv,CALG\_MD4,0,0,&hHash);

// хеширование введенной пользователем парольной фразы

CryptHashData(hHash,Form6->Edit1->Text.c\_str(),

Form6->Edit1->Text.Length(),0);

// уничтожение формы для ввода парольной фразы

delete Form6;

// генерация ключа расшифрования из хеш-значения парольной фразы

//**с добавлением случайного ключа к числу**

CryptDeriveKey(hProv,CALG\_RC2,hHash,CRYPT\_EXPORTABLE|CRYPT\_CREATE\_SALT,&hKey);

// установка режима расшифрования

CryptSetKeyParam(hKey,KP\_MODE,(BYTE\*)&Param,0);

// разрушение хеш-значения

CryptDestroyHash(hHash);

}

// обработка исключительных ситуаций

catch(Exception& E)

{ // уничтожение формы для ввода парольной фразы

delete Form6;

// освобождение дескриптора криптопровайдера

if(hProv) CryptReleaseContext(hProv,0);

// вывод сообщения об ошибке

Application->ShowException(&E);

// завершение работы программы

Application->Terminate();

return;

}

/\* если файл с учетными записями пользователей не существует

(первый запуск программы) \*/

if(!FileExists(SECFILE))

{ AccFile.open(TMPFILE,ios::out|ios::binary); /\* сразу создаем расшифрованный

файл \*/

// подготовка учетной записи администратора

strcpy(UserAcc.UserName,"ADMIN");

strcpy(UserAcc.UserPass,"");

UserAcc.PassLen=0;

UserAcc.Block=false;

UserAcc.Restrict=true;

// запись в файл

AccFile.write((const char\*)&UserAcc,sizeof(UserAcc));

// закрытие файла

AccFile.close(); }

/\* если файл с учетными записями существует

(второй и последующие запуски программы), то он должен быть расшифрован \*/

else

{ // 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(strcmp(UserAcc.UserName,"ADMIN"))

{ // разрушение ключа расшифрования

CryptDestroyKey(hKey);

// освобождение дескриптора криптопровайдера

CryptReleaseContext(hProv,0);

// вывод сообщения об ошибке

ShowMessage("Неверный ключ расшифрования!");

// удаление временного файла

remove(TMPFILE);

// завершение работы программы

Application->Terminate();

return; } }

}

## Содержимое зашифрованного файла с учетными записями

# 