МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО ЗабГУ)
Кафедра Физики и техники связи

Методические указания к лабораторным и практическим работам по дисциплине

«Информационная безопасность»

Чита, 2014

ЛАБОРАТОРНЫЕ РАБОТЫ ПО КУРСУ
«Методы и средства защиты компьютерной информации»

В ходе выполнения лабораторной работы студент должен выполнить предложенное задание и подготовить отчет о проделанной работе. Форма сдачи лабораторной работы предполагает демонстрацию выполненного задания (программного продукта) и знаний теоретической части вопроса, рассмотренного в лабораторной работе.

Отчет по лабораторной работе должен содержать:

1. Тему и цель лабораторной работы;
2. Вариант задания на лабораторную работу;
3. Краткие теоретические сведения и описание алгоритма работы программы;
4. Листинг разработанной программы с подробными комментариями;
5. Результаты работы программы;
6. Выводы.

Лабораторная работа № 1
Реализация дискреционной модели политики безопасности

***Цель работы*:** ознакомиться с проблемами реализации политик безопасности в компьютерных системах на примере дискреционной модели.

Теоретические сведения

Под политикой безопасности понимают набор норм, правил и практических приемов, регулирующих управление, защиту и распределение ценной информации. Политика безопасности задает механизмы управления доступа к объекту, определяет как разрешенные, так и запрещенные доступы.

Политика безопасности реализуется посредством административно-организационных мер, физических и программно-технических средств и определяет архитектуру системы защиты. Для конкретной организации политика безопасности должна носить индивидуальный характер и зависеть от конкретной технологии обработки информации и используемых программных и технических средств.

Политика безопасности определяется способом управления доступом, который задаёт порядок доступа к объектам системы. Различают два основных вида политики безопасности: избирательную и полномочную.

Избирательная политика безопасности основана на избирательном способе управления доступом. Избирательное (или дискреционное) управление доступом характеризуется заданным администратором множеством разрешенных отношений доступа (например, в виде троек объект – субъект – тип доступа). Обычно для описания свойств избирательного управления доступом применяют математическую модель на основе матрицы доступа.

Матрица доступа представляет собой матрицу, в которой столбец соответствует объекту системы, а строка – субъекту. На пересечении столбца и строки матрицы указывается тип разрешенного доступа субъекта к объекту. Обычно выделяют такие типы доступа субъекта к объекту, как «доступ на чтение», «доступ на запись», «доступ на исполнение» и т.п. Матрица доступа является самым простым подходом к моделированию систем управления доступом. Однако она служит основой для сложных моделей, более адекватно описывающих реальные автоматизированные системы обработки информации (АСОИ).

Избирательная политика безопасности широко применяется в АСОИ коммерческого сектора, так как её реализация соответствует требованиям коммерческих организаций по разграничению доступа и подотчетности, а также имеет приемлемую стоимость.

Полномочная политика безопасности основана на полномочном (мандатном) способе управления доступом. Полномочное (или мандатное) управление доступом характеризуется совокупностью правил предоставления доступа, определенных на множестве атрибутов безопасности субъектов и объектов, например, в зависимости от метки конфиденциальности информации и уровня допуска пользователя. Полномочное управление доступом подразумевает, что:

1) все субъекты и объекты системы однозначно идентифицированы;

2) каждому объекту системы присвоена метка конфиденциальности информации, определяющая ценность содержащейся в нем информации;

3) каждому субъекту системы присвоен определенный уровень допуска, определяющий максимальное значение метки конфиденциальности информации объектов, к которым субъект имеет доступ.

Чем важнее объект, тем выше его метка конфиденциальности. Поэтому наиболее защищенными оказываются объекты с наиболее высокими значениями метки конфиденциальности.

Основное назначение полномочной политики безопасности – регулирование доступа субъектов системы к объектам с различными уровнями конфиденциальности, предотвращение утечки информации с верхних уровней должностной иерархии на нижние, а также блокирование возможных проникновений с нижних уровней на верхние.

При выборе и реализации политики безопасности в компьютерной системе, как правило, работают следующие шаги:

1. В информационную структуру вносится структура ценностей (определяется ценность информации) и проводится анализ угроз и рисков для информации и информационного обмена.

2. Определяются правила использования для любого информационного процесса, права доступа к элементам информации с учетом данной оценки ценностей.

Реализация политики безопасности должна быть четко продумана. Результатом ошибочного или бездумного определения правил политики безопасности, как правило, является разрушение ценности информации без нарушения политики.

Дискреционная политика безопасности

Пусть O – множество объектов, U – множество пользователей, S – множество действий пользователей над объектами. Тогда дискреционная политика определяет отображение $O \rightarrow U$ (объектов на пользователей-субъектов). В соответствии с данным отображением, каждый объект $O_j \in O$ объявляется собственностью соответствующего пользователя $U_k \in U$, который может выполнять над ними определенную совокупность действий $S_i \in S$, в которую могут входить несколько элементарных действий (чтение, запись, модификация и т.д.). Пользователь, являющийся собственником объекта, иногда имеет право передавать часть или все права другим пользователям (обладание администраторскими правами).

Указанные права доступа пользователей-субъектов к объектам компьютерной системы записываются в виде так называемой матрицы доступа. На пересечении i-й строки и j-ого столбца данной матрицы располагается элемент S_{ij} – множество разрешенных действий j-ого пользователя над i-м объектом.

Пример. Пусть имеем множество из трёх пользователей {Администратор, Гость, Пользователь_1} и множество из четырёх объектов {Файл_1, Файл_2, CD-RW, Дисковод}. Множество возможных действий включает следующие: {Чтение, Запись, Передача прав другому пользователю}. Действие «Полные права» разрешает выполнение всех трёх действий, действие «Запрет» запрещает выполнение всех перечисленных действий. В данном случае, матрица доступа, описывающая дискреционную политику безопасности, может выглядеть следующим образом.

Таблица 1. Пример матрицы доступа

Объект / Субъект	Файл_1	Файл_2	CD-RW	Дисковод
1. Администратор	Полные права	Полные права	Полные права	Полные права
2. Гость	Запрет	Чтение	Чтение	Запрет
3. Пользователь_1	Чтение, передача прав	Чтение, запись	Полные права	Запрет

Например, Пользователь_1 имеет права на чтение и запись в Файл_2. Передавать же свои права другому пользователю он не может.

Пользователь, обладающий правами передачи своих прав доступа к объекту другому пользователю, может сделать это. При этом, пользователь, передающий права, может указать непосредственно, какие из своих прав он передает другому.

Например, если Пользователь_1 передает право доступа к Файлу_1 на чтение пользователю Гость, то у пользователя Гость появляется право чтения из Файла_1.

Задание на лабораторную работу

Пусть множество *S* возможных операций над объектами компьютерной системы задано следующим образом: *S* = {«Доступ на чтение», «Доступ на запись», «Передача прав»}.

1. Получить данные о количестве пользователей и объектов компьютерной системы из табл. 2, соответственно варианту.

2. Реализовать программный модуль, создающий матрицу доступа пользователей к объектам компьютерной системы. Реализация данного модуля подразумевает следующее:

2.1. Необходимо выбрать идентификаторы пользователей, которые будут использоваться при их входе в компьютерную систему (по одному идентификатору для каждого пользователя, количество пользователей указано для варианта). Например, множество из трёх идентификаторов пользователей {Ivan, Sergey, Boris}. Один из данных идентификаторов должен соответствовать администратору компьютерной системы (пользователю, обладающему полными правами доступа ко всем объектам).

2.2. Реализовать программное заполнение матрицы доступа, содержащей количество пользователей и объектов, соответственно Вашему варианту.

2.2.1. При заполнении матрицы доступа необходимо учитывать, что один из пользователей должен являться администратором системы (допустим, Ivan). Для него права доступа ко всем объектам должны быть выставлены как полные.

2.2.2. Права остальных пользователей для доступа к объектам компьютерной системы должны заполняться случайным образом с помощью датчика случайных чисел. При заполнении матрицы доступа необходимо учитывать, что пользователь может иметь несколько прав доступа к некоторому объекту компьютерной системы, иметь полные права, либо совсем не иметь прав.

2.2.3. Реализовать программный модуль, демонстрирующий работу в дискреционной модели политики безопасности.

3. Данный модуль должен выполнять следующие функции:

3.1. При запуске модуля должен запрашиваться идентификатор пользователя (проводится идентификация пользователя), при успешной идентификации пользователя должен осуществляться вход в систему, при неуспешной – выводиться соответствующее сообщение.

3.2. При входе в систему после успешной идентификации пользователя на экране должен распечатываться список всех объектов системы с указанием перечня всех доступных прав доступа идентифицированного пользователя к данным объектам. Вывод можно осуществить, например, следующим образом:

```
User: Boris
Идентификация прошла успешно, добро пожаловать в систему
Перечень Ваших прав:
Объект1:     Чтение
Объект2:     Запрет
Объект3:     Чтение, Запись
Объект4:     Полные права
Жду ваших указаний >
```

3.3. После вывода на экран перечня прав доступа пользователя к объектам компьютерной системы, необходимо организовать ожидание указаний пользователя на осуществление действий над объектами в компьютерной системе. После получения команды от пользователя, на экран необходимо вывести сообщение об успешности либо не успешности операции. При выполнении операции передачи прав (grant) должна модифицироваться матрица доступа. Программа должна поддерживать операцию выхода из системы (quit), после которой запрашивается идентификатор пользователя. Диалог можно организовать, например, так:

```
Жду ваших указаний > read
Над каким объектом производится операция? 1
Операция прошла успешно
Жду ваших указаний > write
Над каким объектом производится операция? 2
Отказ в выполнении операции. У Вас нет прав для ее осуществления
Жду ваших указаний > grant
Право на какой объект передается? 3
Отказ в выполнении операции. У Вас нет прав для ее осуществления
Жду ваших указаний > grant
Право на какой объект передается? 4
Какое право передается? read
Какому пользователю передается право? Ivan
Операция прошла успешно
Жду ваших указаний > quit
Работа пользователя Boris завершена. До свидания.
User:
```

4. Выполнить тестирование разработанной программы, продемонстрировав реализованную модель дискреционной политики безопасности.

5. Оформить отчет по лабораторной работе.

Таблица 2. Варианты заданий

Вариант	Количество субъектов доступа (пользователей)	Количество объектов доступа
1	3	3
2	4	4
3	5	4
4	6	5
5	7	6
6	8	3
7	9	4
8	10	4
9	3	5
10	4	6
11	5	3
12	6	4
13	7	4
14	8	5
15	9	6
16	10	3

17	3	4
18	4	4
19	5	5
20	6	6
21	7	3
22	8	4
23	9	4
24	10	5

Окончание табл. 2

Вариант	Количество субъектов доступа (пользователей)	Количество объектов доступа
25	3	6
26	4	3
27	5	4
28	6	4
29	6	5
30	8	6

Контрольные вопросы

1. Что понимается под политикой безопасности в компьютерной системе?
2. В чем заключается модель дискреционной политики безопасности в компьютерной системе?
3. Что понимается под матрицей доступа в дискреционной политике безопасности? Что хранится в данной матрице?
4. Какие действия производятся над матрицей доступа в том случае, когда один субъект передает другому субъекту свои права доступа к объекту компьютерной системы?

Лабораторная работа № 2

Количественная оценка стойкости парольной защиты

Цель работы: реализация простейшего генератора паролей, обладающего требуемой стойкостью к взлому.

Теоретические сведения

Подсистемы идентификации и аутентификации пользователя играют важную роль в системах защиты информации.

Стойкость подсистемы идентификации и аутентификации пользователя в системе защиты информации (СЗИ) во многом определяет устойчивость к взлому самой СЗИ. Данная стойкость определяется гарантией того, что злоумышленник не сможет пройти аутентификацию, присвоив чужой идентификатор или украв его.

Парольные системы идентификации/аутентификации являются одними из основных и наиболее распространенных в СЗИ методами пользовательской аутентификации. В данном случае информацией, аутентифицирующей пользователя, является некоторый секретный пароль, известный только легальному пользователю.

Парольная аутентификация пользователя, как правило, передний край обороны СЗИ. В связи с этим модуль аутентификации по паролю наиболее часто подвергается атакам со стороны злоумышленника. Цель последнего в данном случае – подобрать аутентифицирующую информацию (пароль) легального пользователя.

Методы парольной аутентификации пользователя наиболее просты и при несоблюдении определенных требований к выбору пароля являются достаточно уязвимыми.

Основными минимальными требованиями к выбору пароля и к подсистеме парольной аутентификации пользователя являются следующие.

К паролю:

1) минимальная длина пароля должна быть не менее 6 символов;

2) пароль должен состоять из различных групп символов (малые и большие латинские буквы, цифры, специальные символы ‘(’, ‘)’, ‘#’ и т.д.);

3) в качестве пароля не должны использоваться реальные слова, имена, фамилии и т.д.

К подсистеме парольной аутентификации:

1) администратор СЗИ должен устанавливать максимальный срок действия пароля, после чего, пароль следует сменить;

2) в подсистеме парольной аутентификации необходимо установить ограничение числа попыток ввода пароля (как правило, не более трёх);

3) в подсистеме парольной аутентификации требуется установить временную задержку в случае ввода неправильного пароля.

Как правило, для генерирования паролей в СЗИ, удовлетворяющих перечисленным требованиям к паролям, используются программы – автоматические генераторы паролей пользователей.

При выполнении перечисленных требований к паролям и к подсистеме парольной аутентификации единственно возможным методом взлома данной подсистемы злоумышленником является прямой перебор паролей (brute forcing). В данном случае, оценка стойкости парольной защиты осуществляется следующим образом.

Количественная оценка стойкости парольной защиты

Пусть A – мощность алфавита паролей (количество символов, которые могут быть использованы при составлении пароля: если пароль состоит только из малых английских букв, то $A = 26$), L – длина пароля, $S = A^L$ – число всевозможных паролей длины L, которые можно составить из символов алфавита A, V – скорость перебора паролей злоумышленником, T – максимальный срок действия пароля.

Тогда, вероятность P подбора пароля злоумышленником в течение срока его действия V определяется по следующей формуле:

$P = (V \cdot T) / S = (V \cdot T) / A^L$.

Эту формулу можно использовать в обратную сторону для решения следующей задачи.

Задача. Определить минимальные мощность алфавита паролей A и длину паролей L, обеспечивающих вероятность подбора пароля злоумышленником не более заданной P, при скорости подбора паролей V, максимальном сроке действия пароля T.

Данная задача имеет неоднозначное решение. При исходных данных V, T, P однозначно можно определить лишь нижнюю границу S^* числа всевозможных паролей. Целочисленное значение нижней границы вычисляется по формуле

$$S^* = [V \cdot P / T], \quad (1)$$

где [] – целая часть числа, взятая с округлением вверх.

После определения нижней границы S^* необходимо выбрать такие A и L для формирования $S = A^L$, чтобы выполнялось следующее неравенство:

$$S^* \le S = A^L. \quad (2)$$

При выборе S, удовлетворяющего неравенству (2), вероятность подбора пароля злоумышленника (при заданных V и T) будет меньше, чем заданная P.

Следует отметить, что при осуществлении вычислений по формулам (1) и (2), величины должны быть приведены к одним размерностям.

Пример. Исходные данные: $P = 10^{-6}$, $T = 7$ дней = 1 неделя, $V = 10$ (паролей / минуту) = $10 \cdot 60 \cdot 24 \cdot 7 = 100800$ паролей в неделю. Тогда, $S^* = [(10800 \cdot 1) / 10^{-6}] = 108 \cdot 10^8$.

Условию $S^* \le A^L$ удовлетворяют, например, такие комбинации A и L, как $A = 26$, $L = 8$ (пароль состоит из восьми малых символов английского алфавита),

A = 36, L = 6 (пароль состоит из шести символов, среди которых могут быть малые латинские буквы и произвольные цифры).

Задание на лабораторную работу

1. В табл. 3 найти для указанного варианта значения характеристик P, V, T.
2. Вычислить по формуле (1) нижнюю границу S^* для заданных P, V, T.
3. Выбрать некоторый алфавит с мощностью A и получить минимальную длину пароля L, при котором выполняется условие (2).
4. Реализовать программу для генерации паролей пользователей. Программа должна формировать случайную последовательность символов длины L, при этом должен использоваться алфавит из A символов.
5. Оформить отчет по лабораторной работе.

Коды символов:

1. Коды английских символов : «A» = 65, …, «Z» = 90, «a» = 97,…, «z» = 122.
2. Коды цифр : «0» = 48, «9» = 57.
3. «!» = 33, «“» = 34, «#» = 35, «$» = 36, «%» = 37, «&» = 38, «‘» = 39.
4. Коды русских символов : «А» – 128, … «Я» – 159, «а» – 160,…, «п» – 175, «р» – 224,…, «я» – 239.

Таблица 3. Варианты заданий

Вариант	P	V	T
1	10^{-4}	15 паролей/мин	2 недели
2	10^{-5}	3 паролей/мин	10 дней
3	10^{-6}	10 паролей/мин	5 дней
4	10^{-7}	11 паролей/мин	6 дней
5	10^{-4}	100 паролей/день	12 дней
6	10^{-5}	10 паролей/день	1 месяц
7	10^{-6}	20 паролей/мин	3 недели
8	10^{-7}	15 паролей/мин	20 дней
9	10^{-4}	3 паролей/мин	15 дней
10	10^{-5}	10 паролей/мин	1 неделя
11	10^{-6}	11 паролей/мин	2 недели
12	10^{-7}	100 паролей/день	10 дней
13	10^{-4}	10 паролей/день	5 дней
14	10^{-5}	20 паролей/мин	6 дней
15	10^{-6}	15 паролей/мин	12 дней
16	10^{-7}	3 паролей/мин	1 месяц
17	10^{-4}	10 паролей/мин	3 недели
18	10^{-5}	11 паролей/мин	20 дней
19	10^{-6}	100 паролей/день	15 дней
20	10^{-7}	10 паролей/день	1 неделя
21	10^{-4}	20 паролей/мин	2 недели
22	10^{-5}	15 паролей/мин	10 дней
23	10^{-6}	3 паролей/мин	5 дней

Окончание табл. 3

Вариант	*P*	*V*	*T*
24	10^{-7}	10 паролей/мин	6 дней
25	10^{-4}	11 паролей/мин	12 дней
26	10^{-5}	100 паролей/день	1 месяц
27	10^{-6}	10 паролей/день	3 недели
28	10^{-7}	20 паролей/мин	20 дней
29	10^{-4}	15 паролей/мин	15 дней
30	10^{-5}	3 паролей/мин	1 неделя

Контрольные вопросы

1. Чем определяется стойкость подсистемы идентификации и аутентификации?
2. Перечислить минимальные требования к выбору пароля.
3. Перечислить минимальные требования к подсистеме парольной аутентификации.
4. Как определить вероятность подбора пароля злоумышленником в течение срока его действия?
5. Выбором каких параметров можно повлиять на уменьшение вероятности подбора пароля злоумышленником при заданной скорости подбора пароля злоумышленником и заданном сроке действия пароля?

Лабораторная работа №3

Ассиметричные алгоритмы шифрования данных

Цель работы: освоить методику работы ассиметричных алгоритмов шифрования, где существует два ключа – один для шифрования, другой для дешифрования.

Теоретические сведения

Алгоритм RSA разработан в 1977 г. Роном Ривестом, Ади Шамиром и Леном Адлеманом и опубликован в 1978 г. С тех пор алгоритм Rivest-Shamir-Adleman (RSA) широко применяется практически во всех приложениях, использующих криптографию с открытым ключом.

Алгоритм RSA:

1. Вычисление ключей

Важным моментом в этом криптоалгоритме является создание пары ключей: открытого и закрытого. Для алгоритма RSA этап создания ключей состоит из следующих операций:

1.1. Выбираются два простых различных числа p и q. Вычисляется их произведение $n = p \cdot q$, называемое модулем. Под простым числом будем понимать такое число, которое делится только на 1 и на само себя. Взаимно простыми числами будем называть такие числа, которые не имеют ни одного общего делителя, кроме единицы.

1.2. Вычисляется функция Эйлера $\Phi(n) = (p - 1) \cdot (q - 1)$.

1.3. Выбирается произвольное число e ($e < n$), такое, что $1 < e < \Phi(n)$ и не имеет общих делителей, кроме 1 (взаимно простое) с числом $(p - 1) \cdot (q - 1)$.

1.4. Вычисляется d методом Евклида таким образом, что $(e \cdot d - 1)$ делится на $(p - 1) \cdot (q - 1)$.

1.5. Два числа (e, n) публикуются как открытый ключ.

1.6. Число d хранится в секрете – закрытый ключ есть пара (d, n), который позволит читать все послания, зашифрованные с помощью пары чисел (e, n).

2. Шифрование

Шифрование с помощью пары чисел производится следующим образом:

2.1. Отправитель разбивает своё сообщение M на блоки m_i. Значение $m_i < n$, поэтому длина блока m_i в битах не больше $k = [\log_2(n)]$ бит, где квадратные скобки обозначают, взятие целой части от дробного числа.

Например, если $n = 21$, то максимальная длина блока $k = [\log_2(21)] = [4.39\ldots] = 4$ бита.

2.2. Подобный блок может быть интерпретирован как число из диапазона $(0; 2^k - 1)$. Для каждого такого числа m_i вычисляется выражение (c_i – зашифрованное сообщение): $c_i = ((m_i)^e) \bmod n$.

Необходимо добавлять нулевые биты слева в двоичное представление блока c_i до размера $k = [\log_2(n)]$ бит.

3. Дешифрование

Чтобы получить открытый текст, необходимо каждый блок дешифровать отдельно: $m_i = ((c_i)^d) \bmod n$.

Пример:

Выбрать два простых числа: $p = 7$, $q = 17$.

Вычислить $n = p \cdot q = 7 \cdot 17 = 119$.

Вычислить $\Phi(n) = (p - 1) \cdot (q - 1) = 96$.

Выбрать e так, чтобы e было взаимнопростым с $\Phi(n) = 96$ и меньше, чем $\Phi(n)$: $e = 5$.

Определить d так, чтобы $d \cdot e \equiv 1 \bmod 96$ и $d < 96$, $d = 77$, так как $77 \cdot 5 = 385 = 4 \cdot 96 + 1$.

Результирующие ключи открытый {5, 119} и закрытый ключ {77, 119}.

Например, требуется зашифровать сообщение $M = 19$: $19^5 = 66 \pmod{119}$, $C = = 66$. Для дешифрования вычисляется $66^{77} \pmod{119} = 19$.

Варианты заданий

1. Разработать консольное приложение для шифрования/дешифрования произвольных файлов с помощью алгоритма RSA.
2. Разработать визуальное приложение для шифрования/дешифрования изображений.
3. Разработать визуальное приложение для шифрования/дешифрования произвольных файлов.
4. Разработать клиент-серверное приложение для защищённой передачи файлов по сети.
5. Разработать клиент-серверное приложение для защищённого обмена сообщениями по сети.
6. Разработать визуальное приложение для шифрования/дешифрования чисел.
7. Разработать консольное приложение для генерации ключей.
8. Реализовать программу для шифрования / дешифрования текстов, работающую по алгоритму RSA. Программа должна уметь работать с текстом произвольной длины.

Контрольные вопросы:

1. Дайте определение алгоритма с открытым ключом.
2. Сколько этапов содержит алгоритм RSA?
3. В чем заключается вычисление ключей алгоритма RSA?
4. Как происходит шифрование в алгоритме RSA?
5. Как происходит дешифрование в алгоритме RSA?

Лабораторная работа №4

Защита от копирования. Привязка к аппаратному обеспечению. Использование реестра

***Цель работы*:** ознакомиться с возможностями «привязки» к характеристикам компьютера.

Теоретические сведения

В качестве анализируемых характеристик компьютера могут использоваться:

1. Информация об используемой операционной системе
2. Имя пользователя;
3. Имя компьютера;
4. Наличие звуковой карты;
5. Наличие подключенных принтера, сканера и т.д;
6. Дата создания BIOS;
7. Серийный номер диска;
8. Характеристики процессора.

Для получения подобных характеристик в операционной системе Windows используются API-функции и информация из реестра.

API-функции

API сокращенно Application Programming Interface (интерфейс прикладного программирования). API – набор функций, которые операционная система предоставляет программисту. API обеспечивает относительно простой путь для программистов для использования полных функциональных возможностей аппаратных средств или операционной системы.

32-разрядные версии Windows обычно используют один и тот же набор функций API, хотя имеются некоторые различия между платформами.

Почти все функции, которые составляют Windows API, находятся внутри DLL (Dynamic Link Library). Эти dll-файлы находятся в системной папке Windows. Существует свыше 1000 функций API, которые условно делятся на четыре основные категории:

1) работа с приложениями – запуск и закрытие приложений, обработка команд меню, перемещения и изменения размера окон;

2) графика – создание изображений;

3) системная информация – определение текущего диска, объема памяти, имя текущего пользователя и т.д.

4) работа с реестром – манипуляции с реестром Windows.

Реестр Windows

Реестр – база данных операционной системы, содержащая конфигурационные сведения. По замыслу Microsoft реестр должен был полностью заменить файлы ini, которые были оставлены только для совместимости со старыми программами, ориентированными на более ранние версии операционной системы.

Переход от ini файлов к реестру произошел по той причине, что на эти файлы накладывается ряд серьезных ограничений, и главное из них состоит в том, что предельный размер такого файла составляет 64Кб.

Предупреждение: никогда не удаляйте или не меняйте информацию в реестре, если Вы не уверены что это именно то, что нужно. В противном случае некорректное изменение данных может привести к сбоям в работе Windows и, в лучшем случае, информацию придется восстанавливать из резервной копии.

Реестр имеет следующую структуру:

1) HKEY_CLASSES_ROOT. В этом разделе содержится информация о зарегистрированных в Windows типах файлов, что позволяет открывать их по двойному щелчку мыши, а также информация для OLE и операций drag-and-drop;

2) HKEY_CURRENT_USER. Здесь содержатся настройки оболочки пользователя (например, Рабочего стола, меню "Пуск", ...), вошедшего в Windows. Они дублируют содержимое подраздела HKEY_USER\name, где name – имя пользователя, вошедшего в Windows. Если на компьютере работает один пользователь и используется обычный вход в Windows, то значения раздела берутся из подраздела HKEY_USERS\.DEFAULT;

3) HKEY_LOCAL_MACHINE. Этот раздел содержит информацию, относящуюся к компьютеру: драйверы, установленное программное обеспечение и его настройки;

4) HKEY_USERS. Содержит настройки оболочки Windows для всех пользователей. Как было сказано выше, именно из этого раздела информация копируется в раздел HKEY_CURRENT_USER. Все изменения в HKCU (сокращенное название раздела HKEY_CURRENT_USER) автоматически переносятся в HKU;

5) HKEY_CURRENT_CONFIG. В этом разделе содержится информация о конфигурации устройств Plug&Play и сведения о конфигурации компьютера с переменным составом аппаратных средств;

6) HKEY_DYN_DATA. Здесь хранятся динамические данные о состоянии различных устройств, установленных на компьютере пользователя. Именно сведения этой ветви отображаются в окне "Свойства: Система" на вкладке "Устройства", вызываемого из Панели управления. Данные этого раздела изменяются самой операционной системой, так что редактировать что-либо вручную не рекомендуется.

Примеры процедур и функций, определяющих параметры компьютера

Определение версии операционной системы

```
BOOL DisplaySystemVersion()
{
   OSVERSIONINFOEX osvi;
   BOOL bOsVersionInfoEx;
   ZeroMemory(&osvi, sizeof(OSVERSIONINFOEX));
   osvi.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
   if( !(bOsVersionInfoEx = GetVersionEx ((OSVERSIONINFO *) &osvi)) )
   {
      osvi.dwOSVersionInfoSize = sizeof (OSVERSIONINFO);
      if (! GetVersionEx ( (OSVERSIONINFO *) &osvi) )
         return FALSE;
   }
```

```
switch (osvi.dwPlatformId)
{
   case VER_PLATFORM_WIN32_NT:
      if ( osvi.dwMajorVersion <= 4 )
         printf("Microsoft Windows NT ");
      if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 0 )
         printf ("Microsoft Windows 2000 ");
      if( bOsVersionInfoEx )
      {
         if ( osvi.wProductType == VER_NT_WORKSTATION )
         {
            if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 1 )
               printf ("Microsoft Windows XP ");

            if( osvi.wSuiteMask & VER_SUITE_PERSONAL )
               printf ( "Home Edition " );
            else
               printf ( "Professional " );
         }
         else if ( osvi.wProductType == VER_NT_SERVER )
         {
            if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 2 )
               printf ("Microsoft Windows .NET ");

            if( osvi.wSuiteMask & VER_SUITE_DATACENTER )
               printf ( "DataCenter Server " );
            else if( osvi.wSuiteMask & VER_SUITE_ENTERPRISE )
               if( osvi.dwMajorVersion == 4 )
                  printf ("Advanced Server " );
               else
                  printf ( "Enterprise Server " );
            else if ( osvi.wSuiteMask == VER_SUITE_BLADE )
               printf ( "Web Server " );
            else
               printf ( "Server " );
         }
      }
      else
      {
         HKEY hKey;
         char szProductType[BUFSIZE];
         DWORD dwBufLen=BUFSIZE;
         LONG lRet;
         lRet = RegOpenKeyEx( HKEY_LOCAL_MACHINE,
            "SYSTEM\\CurrentControlSet\\Control\\ProductOptions",
            0, KEY_QUERY_VALUE, &hKey );
         if( lRet != ERROR_SUCCESS )
             return FALSE;
         lRet = RegQueryValueEx( hKey, "ProductType", NULL, NULL,
            (LPBYTE) szProductType, &dwBufLen);
         if( (lRet != ERROR_SUCCESS) || (dwBufLen > BUFSIZE) )
             return FALSE;
         RegCloseKey( hKey );
         if ( lstrcmpi( "WINNT", szProductType) == 0 )
            printf( "Professional " );
         if ( lstrcmpi( "LANMANNT", szProductType) == 0 )
            printf( "Server " );
         if ( lstrcmpi( "SERVERNT", szProductType) == 0 )
            printf( "Advanced Server " );
      }
```

```
            if ( osvi.dwMajorVersion <= 4 )
            {
               printf ("version %d.%d %s (Build %d)\n",
                  osvi.dwMajorVersion,
                  osvi.dwMinorVersion,
                  osvi.szCSDVersion,
                  osvi.dwBuildNumber & 0xFFFF);
            }
            else
            {
               printf ("%s (Build %d)\n",
                  osvi.szCSDVersion,
                  osvi.dwBuildNumber & 0xFFFF);
            }
            break;
         case VER_PLATFORM_WIN32_WINDOWS:
            if (osvi.dwMajorVersion == 4 && osvi.dwMinorVersion == 0)
            {
                printf ("Microsoft Windows 95 ");
                if ( osvi.szCSDVersion[1] == 'C' || osvi.szCSDVersion[1] == 'B' )
                   printf("OSR2 " );
            }
            if (osvi.dwMajorVersion == 4 && osvi.dwMinorVersion == 10)
            {
                printf ("Microsoft Windows 98 ");
                if ( osvi.szCSDVersion[1] == 'A' )
                   printf("SE " );
            }
            if (osvi.dwMajorVersion == 4 && osvi.dwMinorVersion == 90)
            {
                printf ("Microsoft Windows Millennium Edition ");
            }
            break;
      }
      return TRUE;
   }
```

Определение серийного номера раздела диска

```
   TCHAR    szVolName[256];
   DWORD    dwNum;
   DWORD    dwMaxComSize;
   DWORD    dwFlags;
   TCHAR    szFS[256];
   BOOL     bRes;
   bRes = GetVolumeInformation ( "c:\\", szVolName, sizeof(szVolName), &dwNum,
&dwMaxComSize, &dwFlags, szFS, sizeof(szFS));
```

Определение имени компьютера

```
   const int WSVer = 0x101;
   WSADATA wsaData;
   char Buf[128];
   if (WSAStartup(WSVer, &wsaData) == 0)
   {
     gethostname(&Buf[0], 128);
     MessageBox(0, Buf,0,0);
     WSACleanup;
   }
```

Определение имени пользователя

```
   char buffer[UNLEN+1];
   DWORD size;
   size=sizeof(buffer);
   GetUserName(buffer,&size);
```

Определение версии BIOS

```
LPSTR GetSystemBiosVersion()
{
  HKEY hKey;
  LONG Res1, Res2;
  DWORD cData=255;
  TCHAR SystemBiosVersion[255]={'\0'};
  Res1=RegOpenKeyEx(HKEY_LOCAL_MACHINE,"HARDWARE\\DESCRIPTION\\System",NULL,
KEY_QUERY_VALUE, &hKey);
  if(Res1==ERROR_SUCCESS)
    {

 Res2=RegQueryValueEx(hKey,"SystemBiosVersion",NULL,NULL,…
    (LPBYTE)SystemBiosVersion,&cData);
     if(Res2==ERROR_SUCCESS)
       {
         for (const char* p = SystemBiosVersion; *p; p += strlen(p)+1)
           {
             printf("%s\n", p);
           }

       return SystemBiosVersion;
       }
    else
      {
          MessageBox(NULL,"RegQueryValueEx: SystemBiosVesion","ERROR",MB_OK);
          return NULL;
      }
    }
  else
    {
     MessageBox(NULL,"RegOpenKeyEx: SystemBiosVersion","ERROR",MB_OK);
     return NULL;
    }
  RegCloseKey(hKey);
}
```

Определение частоты процессора (способ №1)

```
double CPUSpeed(void)
  {
    DWORD dwTimerHi, dwTimerLo;
    asm
     {
        DW 0x310F
        mov dwTimerLo, EAX
        mov dwTimerHi, EDX
      }
    Sleep (500);
    asm
     {
        DW 0x310F
        sub EAX, dwTimerLo
        sub EDX, dwTimerHi
        mov dwTimerLo, EAX
        mov dwTimerHi, EDX
     }
    return dwTimerLo/(1000.0*500);
  }
```

Задание на лабораторную работу

Разработать программу, реализующую привязку к компьютеру, используя совокупность характеристик согласно варианту задания. Добиться того, чтобы программа не запускалась на другом компьютере.

Таблица 4. Варианты заданий

№ варианта	Характеристики
1	Серийный номер раздела жесткого диска, MAC-адрес сетевой карты
2	Информация из реестра, тактовая частота процессора
3	Версия операционной системы, MAC-адрес сетевой карты
4	Имя пользователя, серийный номер раздела жесткого диска
5	Название компьютера, информация из реестра
6	Версия БИОС, имя пользователя
7	Серийный номер раздела жесткого диска, имя пользователя
8	Имя пользователя, тактовая частота процессора
9	MAC-адрес сетевой карты, тактовая частота процессора

Контрольные вопросы

1. Что понимается под «привязкой» к компьютеру?
2. Какие характеристики обычно используются для идентификации компьютера?
3. Перечислите основные API-функции для определения индивидуальных характеристик компьютера.
4. Что представляет собой реестр Windows?
5. Какую структуру имеет реестр?