Международный консорциум «Электронный университет»

Московский государственный университет экономики, статистики и информатики

Евразийский открытый институт

А.А. Смирнов
Д.В. Хрипков

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Учебно-практическое пособие

Москва, 2009

УДК 004.42
ББК 32.973-018
С 506

Смирнов А.А., Хрипков Д.В.

С 506 **ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ :** учебно-практическое пособие. – М. : Изд. центр ЕАОИ, 2009. – 191 с.

ISBN 978-5-374-00296-6

УДК 004.42
ББК 32.973-018

ISBN 978-5-374-00296-6

Содержание

Тема 1.

Вводная лекция. Важнейшие направления развития технологий программирования

1.1. Программное обеспечение и его классификация

Под программным обеспечением понимается совокупность программ и документации на них, предназначенных для реализации целей и задач. Программное обеспечение в соответствии с выполняемыми функциями делится на системное (общее) и прикладное (специальное) программное обеспечение.

К системному программному обеспечению относится совокупность программ описаний и инструкций, используемых для эффективного функционирования вычислительной системы, а также при разработке новых программ.

По функциональному назначению в системном программном обеспечении выделяют следующие элементы:

Во-первых, операционную систему;

Во-вторых, систему программирования;

В-третьих, средства контроля и диагностики.

Под операционной системой понимается комплекс управляющих программ, обеспечивающих функционирование вычислительной машины.

Под системой программирования понимают комплекс средств для разработки и отладки программ.

Под средствами контроля и диагностики понимается совокупность программ, которые служат для выявления и локализации неисправностей.

Прикладное программное обеспечение предназначено для решения конкретных задач из различных сфер применения.

1.2. Свойства системного программного обеспечения

Можно выделить следующие свойства системного программного обеспечения:

Во-первых, совместимость (COMPATIBLE) программного обеспечения. Совместимость предполагает поддержку различных приложений, разработанных для ранних версий системного программного обеспечения.

Во-вторых, переносимость (PORTABILITY) программного обеспечения. Переносимость означает возможность работы системного программного обеспечения при использовании различных процессоров.

В-третьих, масштабируемость (SCALABILITY), которая означает, что при работе в корпоративной сети имеется возможность добавлять более производительные серверы и рабочие станции.

В-четвертых, безопасность (SECURITY), которая означает, что при использовании корпоративной сети заданным приложениям обеспечивается полностью изолированное окружение.

В-пятых, распределенная обработка (DISTRIBUTED PROCESSING) , которая предполагает обеспечение связи с различными типами XOCT-компьютеров благодаря поддержке разнообразных протоколов.

В-шестых, надежность (RELIABILITY) и отказоустойчивость (RUBUSTNESS) , которые предполагают защиту программ от повреждения друг другом и операционной системой.

1.3. Организация системного программного обеспечения в виде компонент

Развитие технологии объектно-ориентированного программирования привело к возникновению понятия «компонент». Под компонентом (COMPONENT) понимается некий функциональный элемент, содержащий определенный свой-

ства. Таким образом, понятие "компонент", является в достаточной степени абстрактным и может в определенной степени варьироваться в различных системах.

Использование компонент позволяет перейти к технологии разработки приложений на основе компонентной архитектуры. В этом случае, компонент представляет собой особый вид приложения, т.к. он поставляется как двоичный код, скомпилированный и готовый к использованию. При использовании компонентной архитектуры приложение представляет собой совокупность компонент. Отдельные компоненты подключаются, во время выполнения, к другим компонентам, формируя приложение. Приложение не является статичным. Модификация или расширение приложения сводится к замене одного из составляющих компонентов новой версией.

Компонентная архитектура хорошо приспособлена для создания приложений легко адаптируемых к конкретным требованиям пользователя. Готовые продукты, созданные на основе компонентной архитектуры, позволяют при необходимости заменить любой компонент другим, более соответствующим конкретной ситуации.

Используя компонентную архитектуру можно упростить процесс разработки распределенных приложений. В этом случае, приложение состоит из компонент, разбросанных по разным машинам сети. При разработке распределенных приложений могут быть использованы специальные компоненты-переадресовщики, которые вместо выполнения заданной обработки, обеспечивают передачу запроса к требуемой компоненте.

На основе компонентной архитектуры разрабатываются многие приложения, работающие в сети INTERNET/INTRANET.

Для реализации компонентной архитектуры предназначены такие средства как COM(Component Object Model, модель компонентных объектов).

Концепция компонент реализована практически во всех современных языках программирования.

Тема 2.

Особенности использования объектно-ориентированного программирования в различных системах

2.1. Объекты

2.1.1. Общая структура программы на C++. Пример простейшей программы

Традиционно, давайте напишем программу, выводящую на экран строку приветствия:

```
#include <iostream.h>

main()
{
   cout << "Hello, world\n";
}
```

Строка #include сообщает компилятору, чтобы он включил стандартные возможности потока ввода и вывода, находящиеся в файле iostream.h. Без этих описаний выражеине cout << "Hello, world\n" не имело бы смысла. Операция << пишет свой первый аргумент во второй (в данном случае, строку "Hello, world\n" в стандартный поток вывода cout). Строка - это последовательность символов, заключенная в двойные кавычки. В строке символ обратной косой черты «\», за которым следует другой символ, обозначает один специальный символ; в данном случае, \n является символом новой строки. Таким образом выводимые символы состоят из Hello, world и перевода строки.

Остальная часть программы main() { ... } определяет функцию, названную main. Каждая программа должна содержать функцию с именем main, и работа программы начинается с выполнения этой функции. [3]

Любая программа, написанная на языке C++, состоит из одной или более функций, являющихся основными модулями, из которых она собирается. Наша программа состоит из одной функции main(), и круглые скобки указывают именно на то, что main - имя функции (англ. main – главный). Программа, написанная на языке C++, всегда начинает выполняться с функции, названной main(), поэтому мы имеем возможность выбирать имена для всех используемых нами функций кроме той, с которой начинается выполнение программы.

В круглых скобках в общем случае содержится информация, передаваемая этой функции. В нашем примере передача информации отсутствует, и, следовательно, скобки пусты.

{ и } – фигурные скобки отмечают начало и конец тела функции. Фигурные скобки применяются также и для того, чтобы объединить несколько операторов программы в составной оператор или блок (аналогичны операторным скобкам Begin и End языка Pascal). [4]

Программа на C++ обычно состоит из большого числа исходных файлов, каждый из которых содержит описания типов, функций, переменных и константа.

2.1.2. Файлы заголовков. Введение понятия директивы препроцессора #INCLUDE

Самый обычный способ обеспечить согласованность исходных файлов – это поместить такие описания в отдельные файлы, называемые заголовочными (или хэдер) файлами, а затем включить, то есть скопировать, эти заголовочные файлы во все файлы, где нужны эти описания.

Поскольку обычные заголовочные файлы включаются во многие исходные файлы, они не содержат описаний, которые не должны повторяться. [3]

В заголовочном файле могут содержаться:

Определения типов	struct point { int x, y; }
Описания функций	extern int strlen(const char*);
Определения inline-функций	inline char get() { return *p++; }
Описания данных	extern int a;
Определения констант	const float pi = = 3.141593
Перечисления	enum bool { false, true };
Директивы include	#include
Определения макросов	#define Case break; case
Комментарии	/* проверка на конец файла */

но никогда

Определения обычных функций	char get() { return *p++; }
Определения данных	int a;

Определения сложных константных объектов

const tbl[] = { /* ... */ } [3]

Заголовочные файлы оказываются весьма эффективным средством при модульной разработке крупных программ, когда связь между модулями, размещаемыми в разных файлах, реализуется не только с помощью параметров, но и через внешние объекты, глобальные для нескольких или всех модулей. Описания таких внешних объектов (переменных, массивов, структур и т.п.) помещаются в одном файле, который с помощью директив #include включается во все модули, где необходимы внешние объекты. [4]

Командная строка компилятора вида

#include "имя_файла"

вызывает замену этой строки полным содержимым файла имя_файла. Сначала именованный файл ищется в директории первоначального исходного файла, а затем в стандартных или заданных местах.

Альтернативный вариант, командная строка вида

#include <имя_файла>

производит поиск только в стандартном или заданном месте, и не ищет в директории первоначального исходного файла. (То, как эти места задаются, не является частью языка.)

Включения с помощью #include могут быть вложенными. [3]

Замена директивы #include на содержимое файла осуществляется на этапе предпроцессорной обработки исходного кода программы, то есть до компиляции.

По принятому соглашению заголовочные файлы имею расширение «.h», что означает header.

2.2. Классы

Класс – это тип. Этот производный тип вводится в программу с помощью специального оператора объявления класса. В объявлении класса используется ранее описанный инструментальный набор средств для построения и преобразования производных типов.

Очередное множество форм Бэкуса-Наура определяет синтаксис объявления класса.

Объявление ::= [СписокСпецификаторовОбъявления] [СписокОписателей];

СписокСпецификаторовОбъявления
::= [СписокСпецификаторовОбъявления] СпецификаторОбъявления

СпецификаторОбъявления ::= СпецификаторТипа
::= *****

СпецификаторТипа ::= СпецификаторКласса
::= УточнённыйСпецификаторТипа
::= *****

УточнённыйСпецификаторТипа ::= КлючевоеСловоКласса ИмяКласса
::= КлючевоеСловоКласса Идентификатор
::= enum ИмяПеречисления

КлючевоеСловоКласса ::= union
::= struct
::= class

ИмяКласса ::= Идентификатор

СпецификаторКласса ::= ЗаголовокКласса {[СписокЧленов]}

ЗаголовокКласса
::= КлючевоеСловоКласса [Идентификатор] [СпецификацияБазы]
::= КлючевоеСловоКласса ИмяКласса [СпецификацияБазы]

КлючевоеСловоКласса ::= union
::= struct
::= class

ИмяКласса ::= Идентификатор

Спецификатор класса представляет то, что называется объявлением класса. Уточнённый спецификатор типа объявляет расположенный за ним идентификатор именем класса. Уточнённый спецификатор обеспечивает неполное предварительное объявление класса и перечисления.

Назначение и смысл необязательного нетерминального символа СпецификацияБазы будут обсуждаться позже, в разделах, посвящённых наследованию.

Предварительное объявление обеспечивается уточнённым спецификатором типа и является своеобразным прототипом класса или перечисления. Его назначение - сообщение транслятору предварительной информации о том, что существует (должно существовать) объявление класса (или пере-

числения) с таким именем. Идентификатор, используемый в контексте уточнённого спецификатора имени становится именем класса (или именем перечисления).

Класс считается объявленным даже тогда, когда в нём полностью отсутствует информация о членах класса (пустой список членов класса). Неименованный класс с пустым множеством членов - уже класс!

Имя класса можно употреблять как имя (имя типа) уже в списке членов этого самого класса.

Класс может быть безымянным.

Следующая последовательность операторов объявления

```
class {}; /* Объявлен пустой неименованный класс.*/
class {};
class {};
class {};
/* Это всё объявления. Их количество ничем не ограничивается. */
struct {};
/* Структура – это класс, объявленный с ключевым словом struct.
Опять же пустой и неименованный.*/
```

не вызывает у транслятора никаких возражений.

На основе класса, пусть даже неименованного, может быть объявлен (вернее, определён) объект-представитель этого класса. В таком контексте объявление неименованного (пусть даже и пустого!) класса является спецификатором объявления. Имена определяемых объектов (возможно с инициализаторами) составляют список описателей.

```
class {} Obj1, Obj2, Obj3;/* Здесь объявление пустого класса.*/
class {} Obj4, Obj5, Obj6;/* Просто нечего инициализировать.*/
class {} Obj1;
    /* ^ Ошибка. Одноименные объекты в области действия имени.*/
```

Неименованные классы также можно применять в сочетании со спецификатором typedef (здесь может быть объявление класса любой сложности - не обязательно только пустой). Спецификатор typedef вводит новое имя для обозначения безымянного класса. Описанное имя типа становится его единственным именем.

Сочетание спецификатора typedef с объявлением безымянного класса подобно объявлению класса с именем:

```
class MyClass {/*…*/};
typedef class {/*…*/} MyClass;
```

Правда в первом случае класс имеет собственное имя класса, а во втором – описанное имя типа. Использование описанного имени типа в пределах области действия имени делает эквивалентными следующие определения (и им подобные):

```
class {} Obj1;
MyClass  Obj1;
```

Класс считается объявленным лишь после того, как в его объявлении будет закрыта последняя фигурная скобка. До этого торжественного момента информация о структуре класса остаётся неполной.

Если можно ОБЪЯВИТЬ пустой класс, то можно ОПРЕДЕЛИТЬ и объект-представитель пустого класса. Эти объекты размещаются в памяти. Их размещение предполагает выделение объекту участка памяти с уникальным адресом, а это означает, что объекты пустого класса имеют ненулевой размер.

Действительно, значения выражений sizeof(MyClass) и sizeof(MyObj1) (это можно очень просто проверить) отличны от нуля.

А вот пустое объединение (ещё одна разновидность класса – класс, объявленный с ключевым словом union) не объявляется:

```
union {}; /* Некорректное объявление объединения. */
```

При объявлении объединения требуется детальная информация о внутреннем устройстве этого объединения.

Мы продолжаем формальное определение класса. Теперь рассмотрим синтаксис объявления членов класса.

СписокЧленов ::= ОбъявлениеЧленаКласса [СписокЧленов]
::= СпецификаторДоступа : [СписокЧленов]

ОбъявлениеЧленаКласса ::= [СписокСпецификаторовОбъявления]
[СписокОписателейЧленовКласса];
::= ОбъявлениеФункции
::= ОпределениеФункции [;]
::= КвалифицированноеИмя;

СписокОписателейЧленовКласса ::= ОписательЧленаКласса
::= СписокОписателейЧленовКласса,
ОписательЧленаКласса

ОписательЧленаКласса ::= Описатель [ЧистыйСпецификатор]
::= [Идентификатор] : КонстантноеВыражение

ЧистыйСпецификатор ::= = 0

КвалифицированноеИмяКласса ::= ИмяКласса
::= ИмяКласса :: КвалифицированноеИмяКласса

СпецификаторДоступа ::= private
::= protected
::= public

Список членов определяет полный набор членов данного класса. В этом списке объявляются все члены класса. Таковыми могут быть данные, функции-члены, ранее объявленные классы, перечисления, битовые поля, дружественные функции и даже имена типов. Некоторые из перечисленных понятий нам уже знакомы, о других речь ещё впереди. Этот список не подлежит модификации. Он формируется за один раз.

В соответствии с синтаксическими правилами, членами класса могут быть как определения функций, так и их прототипы. Действительно:

ОбъявлениеЧленаКласса ::=
[СписокСпецификаторовОбъявления]
[СписокОписателейЧленовКласса]; ::=
СпецификаторОбъявления ОписательЧленаКласса; ::=
СпецификаторТипа Описатель; ::=
void Описатель (СписокОбъявленийПараметров); ::=
void ff (void);

С другой стороны,

ОбъявлениеЧленаКласса ::=
ОпределениеФункции [;] ::=
Описатель (СписокОбъявленийПараметров) ТелоФункции ::=
ff (void) {int iVal = 100;}

В соответствии с синтаксическими правилами, членами класса могут быть как определения функций, так и их прототипы. Действительно:

ОбъявлениеЧленаКласса ::=
[СписокСпецификаторовОбъявления]
[СписокОписателейЧленовКласса]; ::=
СпецификаторОбъявления ОписательЧленаКласса; ::=
СпецификаторТипа Описатель; ::=
void Описатель (СписокОбъявленийПараметров); ::=
void ff (void);

С другой стороны,

ОбъявлениеЧленаКласса ::=
ОпределениеФункции [;] ::=
Описатель (СписокОбъявленийПараметров) ТелоФункции ::=
ff (void) {int iVal = 100;}

Точка с запятой после определения функции является декоративным элементом. Ни один член класса не может входить в список членов класса дважды. Поэтому определяемая в теле класса функция оказывается без прототипа. Если класс содержит прототип функции в качестве члена класса, функция располагается за пределами класса. Как мы скоро увидим, всё разнообразие объявлений и определений функций-членов транслятор приводит к единому стандартному виду.

Функции-члены могут определяться вне списка членов класса. При определении функции-члена класса за пределами данного класса, в списке членов класса размещается прототип функции-члена. А при определении функции-члена используется квалифицированное имя. Квалифицированное имя состоит из последовательности имён классов, разделённых операциями разрешения области видимости. Эта последователь-

ность имён завершается именем определяемой функции. Последовательность имён классов в квалифицированных именах определяется степенью вложенности объявлений классов.

Наличие функций-членов делает объявление класса подобным определению (как и любые функции, функции-члены определяются). Как сказано в Справочном руководстве по C++, "Если бы не исторические причины, объявление класса следовало называть определением класса".

Данные-члены класса не могут объявляться со спецификаторами auto, extern, register.

Ни при каких обстоятельствах не допускается объявление одноименных членов. Имена данных-членов должны также отличаться от имён функций-членов. Использование одноимённых функций, констант и переменных в выражениях в пределах одной области действия имён приводит к неоднозначности. Как известно, имя функции, как и имя константы и переменной, является выражениями. Если допустить объявление одноимённых переменных, констант и функций, то в ряде случаев просто невозможно будет определить, о чём в программе идёт речь.

Объявляемые в классе данные-члены, которые являются представителями классов, должны представлять ранее объявленные классы. Транслятор должен знать заранее о структуре подобных данных-членов.

Описатель члена класса в объявлении класса не может содержать инициализаторов (это всего лишь объявление).

Структура является классом, объявленным с ключевым словом класса struct. Члены такого класса и базовые классы по умолчанию обладают спецификацией доступа public.

Назначение спецификаторов доступа будет обсуждаться в разделах, посвящённых управлению доступом. Пока будет достаточно в объявлении класса указать спецификатор public. В этом случае члены класса оказываются доступны (к ним можно будет свободно обращаться) из любого оператора программы.

Объединение является классом, объявленным с ключевым словом класса union. Его члены также по умолчанию обладают

спецификацией доступа public. В каждый момент исполнения программы объединение включает единственный член класса. В этом его специфика. Именно поэтому не может быть пустого объединения. Позже мы вернёмся к объединениям.

Если функция-член определяется вне тела класса, в список членов класса включается прототип функции. Определение функции сопровождается квалифицированным именем, которое указывает транслятору на принадлежность определяемой функции-члена классу. Последняя часть квалифицированного имени (собственно имя функции) должна совпадать с именем прототипа функции-члена, объявленного ранее в классе.

Подобно определению данных основных типов, в программе могут быть определены объекты ранее объявленного типа. В ходе определения объекта-представителя класса выделяется память для размещения данных-членов класса. При этом непосредственно в этой области памяти размещаются все данные-члены, за исключением данных, объявленных со спецификатором static (об этом спецификаторе будет сказано ниже).

Разбор структуры класса осуществляется транслятором в несколько этапов.

На первом этапе исследуется список данных-членов класса. Именно этот список и определяет общую структуру класса. До окончания этой стадии разбора класса, а фактически до завершения объявления класса, его имя в объявлении данных-членов может быть использовано лишь в таком контексте, где не используется информация о размерах класса. Это объявления указателей, ссылок и статических членов класса (о них после).

Таким образом, объект-представитель класса не может быть членом собственного класса, поскольку объект-представитель класса может быть объявлен как член класса лишь после того, как завершено объявление этого класса.

Функция-член класса существует в единственном экземпляре для всех объектов-представителей данного класса. Переобъявление и уточнение структуры класса в C++ недопустимо.

Серия простых примеров демонстрирует, что можно, а что нельзя делать при объявлении данных-членов класса.

```
class C1
{
 C1 MyC;
// Это ошибка. В классе не допускается объявления данных-членов
// объявляемого класса.
 C1* pMyC;
// А указатель на класс объявить можно.
};
```

Для объявления таких указателей или ссылок на объекты объявляемого класса достаточно неполного предварительного объявления класса. Указатели и ссылки имеют фиксированные размеры, которые не зависят от типа представляемого объекта.

```
class C2;
class C1
{
 C1* pMyC1;
 C2* pMyC2;
};
C2* PointOnElemOfClassC2;
```

Назначение неполного объявления подобно прототипу функции и используется исключительно в целях предварительного информирования транслятора. Очевидно, что создание объектов на основе предварительного неполного объявления невозможно. Однако это не снижает ценности уточнённого спецификатора.

На втором проходе трансляции объявления класса осуществляется проверка списков параметров в объявлениях функций-членов класса, и определяется размер класса. К этому моменту транслятору становится известна общая структура класса. И потому, как ни странно это выглядит, в классе может быть объявлена функция-член класса, которая возвращает значение объявляемого класса и содержит в списке параметров параметры этого же класса:

```
class C2;
class C1
{
 C1 F1(C1 par1) {return par1;};
```

//Объявить данные-члены класса C1 нельзя, а функцию – можно!

```
 C1* pMyC1;
 C2* pMyC2;
// C1 MyC;
};
C2* PointOnElemOfClassC2;
```

Где бы ни располагалась объявляемая в классе функция-член, транслятор приступает к её разбору лишь после того, как он определяет общую структуру класса.

В соответствии с формальным определением создадим наш первый класс:

```
СпецификаторКласса ::= ЗаголовокКласса { [СписокЧленов] };      ::=
КлючевоеСловоКласса Идентификатор { ОбъявлениеЧленаКласса
                        ОбъявлениеЧленаКласса }; ::=
class FirstClass { СпецификаторОбъявления ОписательЧленаКласса;
                        ОписаниеФункции; }; ::=
class FirstClass { СпецификаторОбъявления ОписательЧленаКласса;
                    int FirstClassFunction(void);}; ::=
class FirstClass {
          long int* PointerToLongIntVal;
          int FirstClassFunction(void);
         };
```

За исключением квалифицируемого имени синтаксис определения функции-члена класса вне класса ничем не отличается от определения обычной функции:

```
int FirstClass::FirstClassFunction(void)
      {
      int IntVal = 100;
      return IntVal;
      };
```

Вот таким получилось построенное в соответствии с грамматикой C++ определение (или объявление) класса.

Заметим, что в C++ существует единственное ограничение, связанное с расположением определения функции-члена класса (конечно, если оно располагается вне тела класса): определение должно располагаться за объявлением класса, со-

держащего эту функцию. Именно «за объявлением»! Без каких-либо дополнительных ограничений типа «непосредственно за» или «сразу за».

Более того, в ряде случаев, например, когда требуется определить функцию-член, изменяющую состояние объекта другого класса, данная функция-член должна располагаться за объявлением класса, состояние объекта которого она изменяет. И это понятно. При разборе такой функции-члена транслятор должен иметь представление о структуре класса.

Допускается и такая схема расположения объявлений, при которой первыми располагаются неполные объявления классов, следом соответствующие объявления классов и лишь затем определения функций-членов. Подобные определения мы будем называть отложенными определениями. Позже мы рассмотрим пример программы, в которой отложенный вариант определения функции-члена является единственно возможным вариантом определения.

Класс – это то, что делает C++ объектно-ориентированным языком. На основе классов создаются новые производные типы и определяются функции, которые задают поведение типа.

Рассмотрим несколько строк программного кода, демонстрирующих свойства производных типов.

```
class Class1 {int iVal;};
class Class2 {int iVal;};
/*
```

Объявление производных типов Class1 и Class2. Эти объявления вводят в программу два новых производных типа. Несмотря на тождество их структуры, это разные типы.

```
*/
void ff(Class1);
```

/* Прототип функции с одним параметром типа Class1.*/

```
void ff(Class2);
/*
```

Прототип функции с одним параметром типа Class2. Это совместно используемые (или перегруженные) функции. Об этих функциях мы уже говорили.

```
*/
Class1 m1; /* Объявление объекта m1 типа Class1. */
Class2 m2; /* Объявление объекта m2 типа Class2. */
int m3;
m1 = m2;
m1 = m3;
m3 = m2;
/*
```

Последние три строчки в данном контексте недопустимы.

Неявное преобразование с участием производных типов в C++ невозможно. Транслятор не имеет никакого понятия о том, каким образом проводить соответствующее преобразование. При объявлении классов необходимо специально определять эти алгоритмы.

```
*/
void ff (Class1 pp)
// Определение первой совместно используемой функции...
{
:::::
}
void ff (Class2 pp)

// Определение второй совместно используемой функции...
{
:::::
}
ff(m1);//Вызов одной из двух совместно используемых функций...
ff(m2);//Вызов второй функции...
```

Ещё один пример объявления класса.

```
class ClassX
{
```

ClassX Mm; //Здесь ошибка. Объявление класса ещё не завершено.

ClassX* pMm; //Объект типа "Указатель на объект". Всё хорошо.

```
ClassX FF(char char,int i = sizeof(ClassX));
/*
```

Прототип функции. Второму параметру присваивается значение по умолчанию. И напрасно! Здесь ошибка. В этот момент ещё неизвестен размер класса ClassX.

```
*/
```

// А вот вполне корректное определение встроенной функции.

```
int RR (int iVal)
{
int i = sizeof(ClassX);
        return i;
}
/*
```

Полный разбор операторов в теле функции производится лишь после полного разбора объявления класса. К этому моменту размер класса уже будет определён.

```
*/
}
```

Тема 3.

Технологии программирования, основанные на динамическом распределении памяти

3.1. Динамическое распределение памяти

Все переменные, встречающиеся в программе, должны быть описаны. Перед началом выполнения программы каждой переменной для размещения ее значений выделяется место в памяти.

Обращение в программе к элементу, размещенному в некотором месте памяти, происходит с помощью идентификатора (имени) элемента. Соответствие между переменной и сопоставленным ей местом в памяти сохраняется для описанных в программе переменных на всем протяжении выполнения программы.

В частности, если обрабатывается массив, то память выделяется в соответствии с максимальным числом элементов в массиве. При обработке экономической информации элементами массива, как правило, являются записи (RECORD). Кроме того, количество элементов в массиве может колебаться в значительных пределах. Поэтому могут возникнуть проблемы при размещении экономической информации в оперативной памяти.

В Delphi имеются средства, позволяющие заниматься отведением и освобождением памяти для размещения объектов непосредственно по ходу выполнения программы. Delphi для этих целей имеет особый, ссылочный, тип данных.

Ссылка (указатель) представляет собой адрес байта оперативной памяти, начиная с которого располагается динамическая переменная. Для резервирования памяти под динамические переменные используется специальная область оперативной памяти, называемая «хипом» (Heap) или «кучей». Вся динамическая память рассматривается как сплошной массив байтов.

Для объявления динамической переменной в программе определяется ссылочный тип с использованием конструкции следующего вида:

TYPE <идентиф. ссылочного типа> = ^ <тип динамич. переменной>

Символ "^" является признаком типа указатель. В качестве типа динамической переменной может быть использован любой тип, кроме файлового.

Например:

```
Type
  T_Pointer = ^ T_Rec;
  T_Rec = record
    Number: Integer;
    Family:  String[30];
  End;
Var Point: T_Pointer;
```

Следует обратить внимание, что конструкция, заданная в "Var", выделяет память не для записи, а для указателя, т.е. адреса этой записи.

Для обращения к динамической переменной используется идентификатор указателя с добавлением к нему справа символа "^". Например, "Point^" обозначает динамическую переменную, которая располагается по адресу, заданному в указателе "Point".

Выделение памяти для динамической переменной выполняется при помощи процедуры "NEW". Процедура NEW создает новую динамическую переменную и устанавливает на нее указатель. Описание процедуры имеет следующий вид:

```
Procedure NEW ( Var P: Pointer );
```

Размер выделяемого блока памяти соответствует размеру того типа, на который ссылается указатель, заданный в качестве параметра.

Для освобождения памяти, выделенной динамической переменной с использованием процедуры NEW, предназначена процедура "DISPOSE". Описание процедуры имеет следующий вид:

```
Procedure DISPOSE ( Var P: Pointer );
```

К средствам управления динамической памятью на физическом уровне относятся следующие стандартные процедуры:

Во-первых, процедура GETMEM (Var P: Pointer; Size: Integer), которая выделяет под переменную, располагаемую в динамической памяти, требуемое количество байт памяти;

Во-вторых, процедура FREEMEM (Var P: Pointer [; Size: Integer]), которая освобождает память, выделенную процедурой GETMEM.

Пример программы, обеспечивающей обработку информации с использованием динамической переменной:

```
Unit Dynam_U;
INTERFACE

Uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls;
Type
  TDYNAM = Class(TForm)
    CmdProcess: TButton;
    CmdExit: TButton;
    LblMesTN: TLabel;
    LblMesFIO: TLabel;
    procedure Process(Sender: TObject);
    procedure Exit(Sender: TObject);
  End;
  T_Pointer = ^T_Rec; {Описание указателя на запись}
  T_Rec = Record {Описание типа записи}
    Number: Integer; {Табельный номер}
    Family: String[30];{ Фамилия сотрудника}
  End;
Var
  DYNAM: TDYNAM;
  Point:T_Pointer;

IMPLEMENTATION
{$R *.DFM}

Procedure TDYNAM.Process(Sender: TObject);
{Обработка информации с использованием
   динамических переменных}
Begin
```

```
 {Выделение динамической памяти}
   New(Point);
 {Занесение информации в динамическую память}
  Point^.Number:= 10;
  Point^.Family:= 'Иванов И.И.';
  {Выдача информации из динамической памяти}
  LblMesTN.Caption:='Табельный номер  '
                + IntToStr(Point^.Number);
  LblMesFIO.Caption:='Фамилия И.О. ' + Point^.Family;
  {Освобождение динамической памяти}
  Dispose(Point);
End;

Procedure TDYNAM.Exit(Sender: TObject);
{закрытие проекта}
Begin
 Close;
End;

End.
```

Динамическое распределение памяти позволяет организовать различные варианты взаимосвязанных структур:

Во-первых, связные списки, использующие одну ссылку на последующий элемент;

Во-вторых, двунаправленные списки, использующие две ссылки. Одна ссылка указывает на последующий элемент, а другая ссылка указывает на предыдущий элемент;

В-третьих, деревья, которые используют три ссылки. Две ссылки указывают на два последующих элемента, и одна ссылка указывает на предыдущий элемент;

В-четвертых, сети, которые используют четыре ссылки. Две ссылки указывают на последующие элементы, и две ссылки на предыдущие элементы.

3.2. Использование связных списков

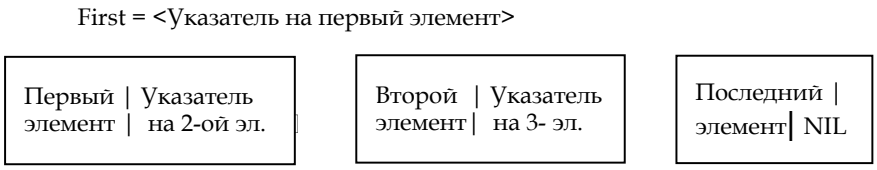
Связные списки представляют собой один из вариантов использования динамического распределения памяти. Связные списки обеспечивают объединение совокупности динамических переменных в единое целое.

Связные списки основаны на использовании типизированных указателей. Типизированные указатели связных списков используют тип RECORD. Записи связного списка содержат указатель на последующий соседний элемент. Последняя запись в качестве указателя содержит специальное значение “NIL”.

Число элементов, входящих в список не определяется заранее. Следовательно, для связного списка не требуется определять максимально возможное число элементов и выделять память в соответствии с указанным числом. Обращение к конкретным элементам списка возможно только путем просмотра всех предшествующих элементов списка, начиная с первого. Следовательно, первый элемент списка имеет особое значение.

Для хранения указателя на первый элемент списка формируется отдельная переменная. Как правило, она называется “FIRST”. Потеря указателя на первый элемент списка делает обращение ко всем элементам списка невозможным.

Схема организации информации в виде связного списка.



При разработке в среде DELPHI программного комплекса для обработки экономической информации с использованием связных списков может быть спроектирована экранная форма следующего вида:

Программный модуль, обеспечивающий использование связных списков может иметь следующий вид:

```
Unit LISTS_U;
Interface
Uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs,
  Menus, Grids, StdCtrls;
Type
```

```
 TListsForm = Class(TForm)
  MainMenuLISTS: TMainMenu;
  ItemCreate: TMenuItem;
  ItemView: TMenuItem;
  ItemAppend: TMenuItem;
  ItemSeek: TMenuItem;
  ItemDelete: TMenuItem;
  ItemInsert: TMenuItem;
  ItemExit: TMenuItem;
  LbInput: TLabel;
  LbNumI: TLabel;
  LbFamilyI: TLabel;
  EdNumI: TEdit;
  EdFamilyI: TEdit;
  LbSeek: TLabel;
  LbNumS: TLabel;
  LbFamilyS: TLabel;
  EdNumS: TEdit;
  EdFamilyS: TEdit;
  LbList: TLabel;
  GridList: TStringGrid;
  LbMes: TLabel;
  Procedure CreateClick(Sender: TObject);
  Procedure Activate(Sender: TObject);
  Procedure ItemExitClick(Sender: TObject);
  Procedure Seek(Sender: TObject);
  Procedure View(Sender: TObject);
  Procedure Append(Sender: TObject);
  procedure Delete(Sender: TObject);
  procedure Insert(Sender: TObject);
 End;
 T_Pointer=^T_Rec;{Указатель на элемент списка}
 T_Rec=Record
  Number:Integer;{ Табельный номер}
  Family:String[30];{Фамилия}
  Next:T_Pointer;{Ссылка на следующий элемент}
  End;
Var
 ListsForm: TListsForm;
 First:T_Pointer; {Ссылка на первый элемент}
Implementation
 {$R *.DFM}
Procedure TListsForm.Activate(Sender: TObject);
{Присвоение первоначальных значений}
```

```
Begin
 Caption:='Обработка связных списков';
 LbInput.Caption:='Ввод информации о работнике';
 LbInput.AutoSize:=True;
 LbNumI.Caption:='Табельный номер';
 LbNumI.Autosize:=True;
 EdNumI.Text:='';
 LbFamilyI.Caption:='Фамилия И.О.';
 LbFamilyI.Autosize:=True;
 EdFamilyI.Text:='';
 LbSeek.Caption:='Поиск работника(Удаление работника)';
 LbSeek.AutoSize:=True;
 LbNumS.Caption:='Табельный номер';
 LbNumS.Autosize:=True;
 EdNumS.Text:='';
 LbFamilyS.Caption:='Фамилия И.О.';
 LbFamilyS.Autosize:=True;
 LbFamilys.Visible:=False;
 EdFamilys.Visible:=False;
 LbList.Caption:='Список работников';
 LbList.AutoSize:=True;
 LbList.Visible:=False;
 GridList.ColCount:=3;
 GridList.Cells[0,0]:='Номер п/п';
 GridList.Cells[1,0]:='Табельный номер';
 GridList.Cells[2,0]:='Фамилия И.О.';
 GridList.Visible:=False;
 LbMes.AutoSize:=True;
 LbMes.Visible:=False;
 EdNumi.SetFocus;
End;

Procedure TListsForm.ItemExitClick(Sender: TObject);
{Закрытие выполняемого проекта}
Begin
Close;
End;

End.
```

3.3. Создание связного списка

Создание связного списка предусматривает следующие действия:

Во-первых, ввод первого элемента в объекты, располагаемые на экранной форме;

Во-вторых, выделение специальной переменной для хранения указателя на первый элемент списка;

В-третьих, выделение динамической памяти для первого элемента;

В-четвертых, перенесение информации из объектов в динамическую память, выделенную для первого элемента списка;

В-пятых, занесение ссылки равной "Nil" на место указателя на следующий элемент.

```
Procedure TListsForm.CreateClick(Sender: TObject);
{Процедура создания первого элемента списка}
Begin
  NEW(First);
  First^.Number:=StrToInt(EdNumI.Text);
  First^.Family:=EdFamilyI.Text;
  First^.Next:=Nil;
End;
```

3.4. Просмотр связного списка

Просмотр связного списка обеспечивается последовательно, начиная с первого элемента и заканчивая последним элементом. Указатель на первый элемент засылается в отдельную переменную, определенную с тем же типом, что и элементы связного списка. Отличительной особенностью последнего элемента является указатель, имеющий специальное значение "NIL".

Для организации просмотра связного списка целесообразно использовать компонент TstringGrid.

```
Procedure TListsForm.View(Sender: TObject);
{Просмотр списка}
```

```
Var
  P:T_Pointer;
  L:Integer;
Begin
 P:=First;
 L:=0;
 {Формирование выходной таблицы}
 While P<>Nil Do
 Begin
   L:=L+1;
   GridList.Cells[0,L]:=IntToStr(L);
   GridList.Cells[1,L]:=IntToStr(P^.Number);
   GridList.Cells[2,L]:=P^.Family;
   P:=P^.Next;
 End;
 {Просмотр таблицы}
  GridList.RowCount:=L+1;
  GridList.Visible:=True;
  LbList.Visible:=True;
End;
```

3.5. Добавление элементов в конец списка

Добавление элемента в конец связного списка предусматривает следующие действия:

Во-первых, ввод добавляемого элемента в объекты, располагаемые на экранной форме;

Во-вторых, нахождение последнего элемента списка;

В-третьих, выделение динамической памяти для добавляемого элемента;

В-четвертых, формирование в последнем элементе списка ссылки на добавляемый элемент;

В-пятых, перенесение информации из объектов в динамическую память, выделенную для нового элемента списка;

В-шестых, занесение в добавляемый элемент ссылки равной “Nil” на место указателя на следующий элемент.

```
Procedure TListsForm.Append(Sender: TObject);
{Добавление нового элемента в конец списка}
Var
```

```
  P:T_Pointer;
Begin
  P:=First;
  {Нахождение последнего элемента}
  While P^.Next<>Nil Do
    P:=P^.Next;
  {Создание нового элемнта}
  New(P^.Next);
  P:=P^.Next;
  P^.Number:=StrToInt(EdNumI.Text);
  P^.Family:=EdFamilyI.Text;
  P^.Next:=Nil;
End;
```

При добавлении нового элемента может быть обеспечена передача информации из предыдущего элемента списка в последующий. Например, при формировании табельных номеров в возрастающем порядке может быть использована следующая совокупность операторов:

```
WN:=P^.Number;
New(P^.Next);
P:= P^.Next;
P^.Number:=Wn+1;
```

3.6. Поиск требуемого элемента в списке

Поиск элемента в списке предусматривает следующие действия:

Во-первых, ввод ключевого значения признака искомого элемента;

Во-вторых, последовательный просмотр элементов списка;

В-третьих, сравнение ключевого поля элемента списка с искомым значением;

В-четвертых, проверку на окончание списка.

```
Procedure TListsForm.Seek(Sender: TObject);
  {Поиск элемента в списке}
Var
  P:T_Pointer;
Begin
```

```
P:=First;
 {Поиск требуемого элемента}
While (P<>Nil) And (P^.Number<>StrToInt(EdNumS.Text))Do
  P:=P^.Next;
If P=Nil Then
  Begin
   LbMes.Caption:='Табельный номер задан неверно';
   LbMes.Visible:=True;
   EdFamilys.Visible:=False;
   LbFamilys.Visible:=False;
  End
 Else
 Begin
   EdFamilyS.Text:=P^.Family;
   EdFamilys.Visible:=True;
   LbFamilys.Visible:=True;
   LbMes.Visible:=False;
 End;
End;
```

3.7. Удаление требуемого элемента из списка

Удаление элемента из списка предусматривает следующие действия:

Во-первых, ввод ключевого значения признака удаляемого элемента;

Во-вторых, поиск элемента с заданным ключевым признаком;

В-третьих, определение элемента, предшествующего удаляемому элементу;

В-четвертых, пересылку указателя из удаляемого элемента в предшествующий элемент списка. При пересылке указателя будет уничтожен указатель на удаляемый элемент. После этого, удаляемый элемент станет недоступным.

При удалении элемента из списка следует отдельно рассматривать ситуацию удаления первого элемента. Удаление первого элемента предполагает замену ссылки FIRST, определяющей начало списка.

Для освобождения занимаемой динимаческой памяти используется процедура Dispose.

```
Procedure TListsForm.Delete(Sender: TObject);
{Удаление элемента с заданным табельным номером}
Var
  P,Q:T_Pointer;
Begin
  P:=First;Q:=First;
  If P^.Number=StrToInt(EdNumS.Text) Then
    Begin
    {Удаляется первый элемент}
     First:=P^.Next;
     Dispose(P);
     LbMes.Visible:=False;
    End
  Else
  Begin
   {Поиск удаляемого элемента}
   While (P<>Nil) And (P^.Number<>StrToInt(EdNumS.Text))Do
    Begin
     Q:=P;
     P:=P^.Next;
    End;
   If P=Nil Then
    Begin
     {Выдача сообщения об ошибке}
     LbMes.Caption:='Табельный номер задан неверно';
     LbMes.Visible:=True;
    End
   Else
    Begin
     {Удаление найденного элемента}
     Q^.Next:=P^.Next;
     Dispose(P);
     LbMes.Visible:=False;
    End;
  End;
End;
```

3.8. Вставка элементов в список, упорядоченный по ключевому признаку

Вставка элементов в список предусматривает следующие действия:

Во-первых, ввод вставляемого элемента;

Во-вторых, выделение динамической памяти для нового элемента;

В-третьих, занесение информации во вновь выделенную память;

В-четвертых, определение элемента списка, предшествующего вставляемому элементу;

В-пятых, засылку указателей, обеспечивающих подключение нового элемента в список.

При засылке указателя отдельно рассматриваются три ситуации:

Во-первых, вставка на место первого элемента;

Во-вторых, вставка в середину списка;

В-третьих, вставка на место последнего элемента.

При вставке перед первым элементом предусматривается выполнение следующих действий:

Во-первых, замена ссылки FIRST, определяющей начало списка;

Во-вторых, во вставляемый элемент заносится указатель на элемент, который раньше был первым.

При вставке в середину списка требуется выполнение следующих действий:

Во-первых, в предшествующий элемент списка заносится указатель на вставляемый элемент;

Во-вторых, во вставляемый элемент заносится указатель на элемент, следующий за вставляемым элементом.

При вставке на место последнего элемента предполагается выполнение следующих действий:

Во-первых, в последний элемент списка заносится указатель на вставляемый элемент;

Во-вторых, во вставляемый элемент на место указателя на следующий элемент заносится специальное значение Nil.

```
Procedure TListsForm.Insert(Sender: TObject);
{Вставка элемента в упорядоченный список,
 в соответствии с заданным табельным номером}
Var
 R,P,Q:T_Pointer;
Begin
 {Формирование нового элемента}
 New(R);
 R^.Number:=StrToInt(EdNumI.Text);
 R^.Family:=EdFamilyI.Text;
 P:=First;
 LbMes.Visible:=False;
 If R^.Number < P^.Number Then
  {Новый элемент вставляется перед
   первым элементом списка}
 Begin
   First:=R;
   R^.Next:=P;
 End
 Else
 Begin
   While (P^.Next<>Nil) And (R^.Number>P^.Number)Do
   {Пропуск записей с меньшими значениями
    табельных номеров. Просмотр завершается
    на последнем элементе списка.}
     Begin
       Q:=P;
       P:=P^.Next;
     End;

   If R^.Number<P^.Number Then
   {Новый элемент вставляется между Q и P }
     Begin
       Q^.Next:=R;
       R^.Next:=P;
     End;
   If R^.Number>P^.Number Then
   {Новый элемент вставляется
    после последнего элемента списка}
     Begin
       P^.Next:=R;
       R^.Next:=Nil;
     End;
   If R^.Number=P^.Number Then
```

```
    {Ошибка: дублирование табельного номера}
   Begin
     LbMes.Caption:='Табельный номер задан неверно';
     LbMes.Visible:=True;
   End;
 End;
End;
```

3.9. Особенности организации двунаправленных списковых структур

Двунаправленный список состоит из элементов, в которых имеются указатели на последующий и предыдущий элемент (previous).

Схема организации информации в виде двунаправленного списка

First = <Указатель на первый элемент>
Last = <Указатель на последний элемент>

Элементы двунаправленного списка могут быть описаны следующим образом:

```
TYPE
 T_Pointer=^T_Rec;{Указатель на элемент списка}
       T_Rec=Record
          Number:Integer;{ Табельный номер}
          Family:String[30];{Фамилия}
          Prev:T_Pointer;{Ссылка на предыдущий элемент}
          Next:T_Pointer;{Ссылка на следующий элемент}
       End;
```

Двунаправленные списки могут быть симметричными и несимметричными.

В симметричном списке ссылки на предыдущий элемент обеспечивают просмотр элементов в обратном порядке.

В несимметричном списке просмотр элементов, выполняемый с последнего элемента, не связан с прямым просмотром списка.

Двунаправленный список может иметь головной элемент. Головной элемент, как правило, включает в себя следующие части:

Во-первых, ссылку на первый элемент (FIRST);

Во-вторых, ссылку на последний элемент (LAST);

В-третьих, число элементов (NUMBER);

В-четвертых, справочная информация (REFER).

Головной элемент двунаправленного списка может быть описан следующим образом:

```
TYPE
 T_Head=Record {Тип головного элемента}
    First: T_Pointer;{Ссылка на первый элемент}
    Last: T_Pointer;{Ссылка на последний элемент}
    Number: Integer;{Количество элементов}
    Refer: String[20];{Справочная информация}
  End;
 Var
   Head:T_Head; {Головной элемент}
```

Процедуры обработки двунаправленных связных списков в значительной степени аналогичны процедурам обработки однонаправленных списков. Важной особенностью обработки является то, что при работе с симметричными списками предшествующие элементы, у рассматриваемого элемента, целесообразно определять обратным ходом.

3.10. Создание двунаправленного списка

Создание двунаправленного списка предусматривает следующие действия:

Во-первых, ввод первого элемента в объекты, располагаемые на экранной форме;

Во-вторых, выделение динамической памяти для первого элемента;

В-третьих, перенесение информации из объектов в динамическую память, выделенную для первого элемента списка;

В-четвертых, формирование специальных переменных для хранения указателей на первый и последний элементы списка;

В-пятых, занесение ссылки равной "Nil" на место указателей на предшествующий и по следующий элементы.

```
Procedure TListsForm.CreateClick(Sender: TObject);
{Процедура создания первого элемента списка}
Var P: T_Pointer;
Begin
  NEW(P);
  P^.Number:=StrToInt(EdNumI.Text);
  P^.Family:=EdFamilyI.Text;
  P^.Next:= Nil;
  P^.Prev:= Nil;
  Head.First:= P;
  Head.Last:= P;
  Head.Number:= 1;
End;
```

3.11. Просмотр двунаправленного списка

Просмотр двунаправленного списка обеспечивается последовательно, начиная с первого элемента и заканчивая последним элементом. Указатель на первый элемент засылается в отдельную переменную, определенную с тем же типом, что и элементы связного списка. Отличительной особенностью последнего элемента является указатель, имеющий специальное значение "NIL".

```
Procedure TListsForm.View(Sender: TObject);
{Просмотр списка}
Var
  P:T_Pointer;
  L:Integer;
Begin
```

```
P:=Head.First;
L:=0;
{Формирование выходной таблицы}
While P<>Nil Do
Begin
  L:=L+1;
  GridList.Cells[0,L]:=IntToStr(L);
  GridList.Cells[1,L]:=IntToStr(P^.Number);
  GridList.Cells[2,L]:=P^.Family;
  P:=P^.Next;
End;
{Просмотр таблицы}
 GridList.RowCount:=L+1;
 GridList.Visible:=True;
 LbList.Visible:=True;
End;
```

3.12. Добавление элементов в конец двунаправленного списка

Добавление элемента в конец связного списка предусматривает следующие действия:

Во-первых, ввод добавляемого элемента в объекты, располагаемые на экранной форме;

Во-вторых, переход к последнему элементу списка;

В-третьих, выделение динамической памяти для добавляемого элемента;

В-четвертых, перенесение информации из объектов в динамическую память, выделенную для нового элемента списка;

В-пятых, занесение в добавляемый элемент ссылки равной “Nil” на место указателя на следующий элемент;

В-шестых, формирование в добавляемом элемент ссылки на предшествующий элемент;

В-седьмых, формирование в последнем элементе списка ссылки на добавляемый элемент;

В-восьмых, формирование нового значения ссылки на последний элемент списка, находящейся в головном элементе.

```
Procedure TListsForm.Append(Sender: TObject);
```

```
{Добавление нового элемента в конец списка}
Var
  Q,P: T_Pointer;
Begin
  P:=Head.Last;
  {Нахождение последнего элемента}
  {Создание нового элемнта}
  New(Q);
  Q^.Number:=StrToInt(EdNumI.Text);
  Q^.Family:=EdFamilyI.Text;
  Q^.Next:=Nil;
  Q^.Prev:=P;
  P^.Next:= Q;
  Head.Last:=Q;
  Head.Number:=Head.Number+1;
End;
```

3.13. Поиск требуемого элемента в списке

Поиск элемента в списке предусматривает следующие действия:

Во-первых, ввод ключевого значения признака искомого элемента;

Во-вторых, последовательный просмотр элементов списка;

В-третьих, сравнение ключевого поля элемента списка с искомым значением;

В-четвертых, проверку на окончание списка.

```
Procedure TListsForm.Seek(Sender: TObject);
   {Поиск элемента в списке}
Var
  P:T_Pointer;
Begin
 P:=Head.First;
  {Поиск требуемого элемента}
 While (P<>Nil) And (P^.Number<>StrToInt(EdNumS.Text))Do
   P:=P^.Next;
 If P=Nil Then
   Begin
    LbMes.Caption:='Табельный номер задан неверно';
    LbMes.Visible:=True;
```

```
    EdFamilys.Visible:=False;
    LbFamilys.Visible:=False;
   End
  Else
  Begin
    EdFamilyS.Text:=P^.Family;
    EdFamilys.Visible:=True;
    LbFamilys.Visible:=True;
    LbMes.Visible:=False;
  End;
End;
```

3.14. Удаление требуемого элемента из двунаправленного списка

Удаление элемента из списка предусматривает следующие действия:

Во-первых, ввод ключевого значения признака удаляемого элемента;

Во-вторых, поиск элемента с заданным ключевым признаком;

В-третьих, определение элементов, предшествующего удаляемому элементу и следующего за удаляемым элементом;

В-четвертых, формирование в предшествующем элементе ссылки на элемент следующий за удаляемым;

В-пятых, формирование в элементе, следующем за удаляемым, ссылки на элемент предшествующим удаляемому.

При удалении элемента из списка следует отдельно рассматривать ситуации удаления первого и последнего элемента.

Удаление первого элемента предполагает замену ссылки FIRST, определяющей начало списка.

Удаление последнего элемента предполагает замену ссылки LAST, определяющей конец списка.

```
Procedure TListsForm.Delete(Sender: TObject);
{Удаление элемента с заданным табельным номером}
Var
  P,Q,R:T_Pointer;
Begin
```

```
P:=Head.First;
If P^.Number=StrToInt(EdNumS.Text) Then
  Begin
  {Удаляется первый элемент}
   R:= P^.Next;
   Head.First:=R;
   R^.Prev:=Nil;
   Dispose(P);
   Head.Number:=Head.Number-1;
   LbMes.Visible:=False;
  End
Else
Begin
 {Поиск удаляемого элемента}
 While (P<>Nil) And (P^.Number<>StrToInt(EdNumS.Text))Do
   P:=P^.Next;
 If P=Nil Then
  Begin
    {Выдача сообщения об ошибке}
     LbMes.Caption:='Табельный номер задан неверно';
     LbMes.Visible:=True;
  End
 Else
 Begin
   {Удаление найденного элемента}
   If P <> Head.Last Then
   Begin
    {Удаление элемента внутри списка}
    R:= P^.Next;
    Q:= P^.Prev;
    R^.Prev:= Q;
    Q^.Next:= R;
    Head.Number:=Head.Number-1;
    Dispose(P);
    LbMes.Visible:=False;
   End;
    If P= Head.Last Then
   Begin
    {Удаление последнего элемента в списке}
    Q:= P^.Prev;
    Q^.Next:= Nil;
    Head.Last:=Q;
    Head.Number:=Head.Number-1;
    Dispose(P);
```

```
        LbMes.Visible:=False;
      End;
    End;
  End;
End;
```

3.15. Вставка элементов в список, упорядоченный по ключевому признаку

Вставка элементов в список предусматривает следующие действия:

Во-первых, ввод вставляемого элемента;

Во-вторых, выделение динамической памяти для нового элемента;

В-третьих, занесение информации во вновь выделенную память;

В-четвертых, определение элемента списка, предшествующего вставляемому элементу;

В-пятых, засылку указателей, обеспечивающих подключение нового элемента в список.

При засылке указателя отдельно рассматриваются три ситуации:

Во-первых, вставка на место первого элемента;

Во-вторых, вставка в середину списка;

В-третьих, вставка на место последнего элемента.

При вставке перед первым элементом предусматривается выполнение следующих действий:

Во-первых, замена ссылки FIRST, определяющей начало списка;

Во-вторых, во вставляемый элемент заносится указатель на элемент, который раньше был первым;

В-третьих, во вставляемом элементе, на месте ссылки на предшествующий элемент, формируется ссылка равная NIL;

В-четвертых, в элементе, который раньше был первых, формируется ссылка на вставляемый элемент;

При вставке в середину списка требуется выполнение следующих действий:

Во-первых, в предшествующий элемент списка заносится указатель на вставляемый элемент;

Во-вторых, во вставляемый элемент заносится указатель на элемент, следующий за вставляемым элементом;

В-третьих, во вставляемый элемент заносится указатель на элемент, предшествующий вставляемому элементу;

В-четвертых, в элемент, следующий за вставляемым элементом, заносится ссылка на вставляемый элемент.

При вставке на место последнего элемента предполагается выполнение следующих действий:

Во-первых, замена ссылки LAST, определяющей конец списка;

Во-вторых, в последний элемент списка заносится указатель на вставляемый элемент;

В-третьих, во вставляемый элемент на место указателя на следующий элемент заносится специальное значение Nil;

В-четвертых, во вставляемый элемент на место указателя на предшествующий элемент заносится на элемент, который раньше был последним.

```
Procedure TListsForm.Insert(Sender: TObject);
{Вставка элемента в упорядоченный список,
 в соответствии с заданным табельным номером}
Var
 R,P,Q:T_Pointer;
Begin
 {Формирование нового элемента}
 New(R);
 R^.Number:=StrToInt(EdNumI.Text);
 R^.Family:=EdFamilyI.Text;
 P:=Head.First;
 LbMes.Visible:=False;
 If R^.Number < P^.Number Then
  {Новый элемент вставляется перед
   первым элементом списка}
 Begin
   Head.First:=R;
   R^.Next:=P;
   R^.Prev:=Nil;
   P^.Prev:=R;
```

```
  Head.Number:=Head.Number+1;
 End
 Else
 Begin
  While (P^.Next<>Nil) And (R^.Number>P^.Number)Do
  {Пропуск записей с меньшими значениями
   табельных номеров. Просмотр завершается
   на последнем элементе списка.}
    P:=P^.Next;
  If R^.Number<P^.Number Then
  {Новый элемент вставляется между Q и P }
   Begin
    Q:= P^.Prev;
    Q^.Next:=R;
    R^.Prev:=Q;
    R^.Next:=P;
    P^.Prev:=R;
    Head.Number:=Head.Number+1;
   End;
  If R^.Number>P^.Number Then
  {Новый элемент вставляется
   после последнего элемента списка}
   Begin
    P^.Next:=R;
    R^.Next:=Nil;
    R^.Prev:=P;
    Head.Last:=R;
    Head.Number:=Head.Number+1;
   End;
  If R^.Number=P^.Number Then
  {Ошибка: дублирование табельного номера}
  Begin
   LbMes.Caption:='Табельный номер задан неверно';
   LbMes.Visible:='True;
  End;
 End;
End;
```

Тема 4.

Технологии программирования, используемые при обработки экономической информации в сети INTERNET/INTRANET

4.1. Основные понятия INTERNET

INTERNET представляет собой общемировую совокупность сетей, связывающую десятки миллионов компьютеров. Транснациональные компьютерные сети, входящие в INTERNET объединяют всевозможные типы компьютеров. Передача информации обеспечивается с применением различных технических средств, в частности: телефонных проводов, спутников, радиомодемов.

Сети, входящие в INTERNET имеют единое адресное пространство. Каждый компьютер (хост, HOST) в INERNET имеет уникальный IP-адрес. Адрес состоит из двух частей: во-первых, адреса сети (идентификатора сети, NETWORK ID) и, во-вторых, адреса хоста (идентификатора хоста, HOST ID). Хосты объединяются в группы, которые называются доменами (DOMAIN). Доменам присваиваются уникальные имена. Для идентификации хостов, входящих в домен используется доменное имя хоста (DOMAIN HOST NAME). Доменное имя хоста, также как и IP-адрес является уникальным, но состоит из символьных обозначений, имеющих смысловую нагрузку.

INTERNET не имеет единого управления, но существуют общественные комитеты, которые вырабатывают стандарты для INTERNET, распределяют электронные адреса и т.д. Организация работы в отдельных узлах доступа обеспечивается специальной фирмой, которая называется провайдером (INTERNET SERVICE PROVIDER, ISP). От провайдера в значительной степени зависит набор предоставляемых сервисных

услуг, время поиска необходимой информации, а также скорость и надежность передачи информации.

Технология INTERNET может быть использована для передачи служебной и деловой информации в рамках одного или нескольких предприятий. Для этой цели используются сети, называемые ИНТРАНЕТ (INTRANET). Сети INTRANET, установленные в различных организациях в значительной степени отличны друг от друга. Как правило, для элементов сети INTRANET не выделяются IP-адреса. Обычно сеть INTRANET подсоединяется к сети INTERNET через интегрированную систему сетевой безопасности.

Применение INTERNET позволяет, по сравнению с другими информационными технологиями, в значительной степени повысить количество доступной информации. Однако, следует учитывать, что 99 процентов документов написаны на английском языке.

В INTERNET широко используются такие понятия как ГИПЕРТЕКСТ (HYPERTEXT) и ГИПЕРМЕДИА (HYPERMEDIA).

Под гипертекстом понимается текст, представленный в виде ассоциативно связанных блоков. Информационные блоки можно представить в виде вершин графа. В этом случае, связи между блоками представляются в виде дуг графа.

Гипертекст значительно отличается от линейного текста. Линейные тексты имеют последовательную структуру и предусматривают их чтение слева направо и сверху вниз. Использование гипертекста позволяет передвигаться в любых направлениях, определяемых ассоциативными связями.

Под гипермедиа понимается документ, в котором кроме текста ассоциативно связаны графика, звуковые клипы и видео клипы. Как правило, в документах гипермедиа для организации ссылок используются не элементы текста, а изображения.

INTERNET включает различные подсистемы информационного обслуживания (сервисы). Сервисы можно подразделить на сервисы интерактивные, прямые (ON-LINE) и отложенного чтения (OFF-LINE). Интерактивными называются сервисы, где требуется немедленная реакция на полученную

информацию. Сервисы прямого обращения характерны тем, что информация по запросу возвращается немедленно. Однако от получателя информации не требуется немедленной реакции. Для сервисов отложенного чтения характерно, что запрос и получение информации могут быть значительно разделены во времени.

Можно выделить следующие сервисы INTERNET.

Во-первых, электронная почта (E-MAIL), которая заключается в возможности посылать и принимать сообщения через компьютер. Скорость доставки сообщений электронной почты в значительной степени зависит от того, каким образом она передается. Первый вариант передачи сообщений предполагает прямую пересылку почты от одного компьютера к другому. В этом случае, путь электронного письма между двумя машинами, непосредственно подключенными в INTERNET, занимает секунды. При данном варианте пересылки вероятность потери или подмены письма минимальна. Другой вариант пересылки предполагает использование системы с промежуточным накоплением. Она не требует связывания компьютеров напрямую. Будучи отправленным, послание пересылается от одного компьютера к другому до тех пор, пока оно не достигнет места назначения. При втором варианте письмо будет идти долго и к тому же может быть потеряно или подменено.

Во-вторых, сетевые новости USENET (телеконференции), которые предполагают передачу одного и того же сообщения одновременно многим получателям. При передачи новостей каждый узел сети передаёт информацию всем узлам, с которыми он обменивается новостями. Таким образом, посланное сообщение распространяется по сети, достигая за короткие сроки всех участников телеконференций USENET во всем мире.

В-третьих, WWW (WORLD WIDE WEB - всемирная паутина), которая представляет собой всемирную распределенную базу гипермедийных документов. В WWW существует множество серверов, которые по запросу клиента возвращают ему гипермедийный документ. В возвращенном гипермедийном документе каждый элемент может являться ссылкой на

другой документ или его часть. Каждая страница WEB (WEB-PAGE) или группа логически связанных страниц (WEB-SITE) имеют уникальный адрес, называемый URL (UNIFORM RESOURCE LOCATOR, UNIVERSAL RESOURCE LOCATOR, USER RESOURCES LOCATOR). URL используется в гипертекстовых ссылках и обеспечивает доступ к распределенным ресурсам сети. При помощи URL можно адресовать как гипертекстовые документы, так и ресурсы других сервисов, например ресурсы E-MAIL. Как правило, URL записывается в соответствии со следующим синтаксисом:

<протокол> **://** <адрес сервера> **:** <номер порта> **/** <имя директории>**/**<имя файла>

URL содержит расширение, которое определяет принадлежность сайта к определенной группе. В частности, расширение 'GOV' свидетельствует о том, что сайт принадлежит правительству Российской федерации; расширение 'COM' свидетельствует о том, что сайт принадлежит коммерческой организации; расширение 'EDU' свидетельствует о том, что сайт принадлежит образовательной организации; расширение 'ORG' свидетельствует о том, что сайт принадлежит некоммерческой организации.

Например, HTTP:// WWW.MICROSOFT.COM.

Для обеспечения соответствия URL и IP-адреса используются такие системы как WINS (Windows Internet Naming Service), DNS (Domain Name System) и другие. WWW представляет собой сервис, требующий быстрых линий связи.

4.2. Протоколы INTERNET

Передача информации в сети INTERNET обеспечивается с использованием определенных правил, которые называются протоколами. В INTERNET имеются десятки различных протоколов. Можно выделить следующие протоколы.

Во-первых, базовый протокол сети INTERNET TCP/IP (TRANSFER CONTROL PROTOCOL / INTERNET PROTOCOL, TRANSPORT CONTROL PROTOCOL / INTERNET PROTOCOL).

Протокол IP представляет собой протокол, описывающий формат пакета данных, передаваемого по сети. Пакет данных представляет собой поток битов. Протокол IP определяет, где в передаваемом потоке располагается адрес и служебная информация, а где сами передаваемые данные.

Протокол TCP предназначен для контроля целостности передаваемой информации. Для контроля на искажение информации при передачи данных используется метод CRC (CYCLIC REDUNDANCY CHECK, код циклического контроля). CRC представляет собой специальную функцию, определяемую по всему содержимому передаваемого пакета данных.

Во-вторых, протокол FTP (FILE TRANSFER PROTOCOL, FILE TRANSPORT PROTOCOL), который предназначен для передачи файлов между компьютерными системами. Протокол FTP представляет собой оптимизацию протокола TCP, ориентированную на передачу файлов между программой-сервером и программой-клиентом. Как правило, протокол FTP используется для распространения публичных файловых архивов, демонстрационных версий программного обеспечения, законов, книг, отчетов и т.д.

В-третьих, протокол пользовательских блоков данных UDP (USER DATAGRAM PROTOCOL), который предназначен для передачи пользовательской информации по частям. Протокол UDP намного проще, чем протокол TCP. Данный протокол полезен в ситуациях, когда мощные механизмы обеспечения надежности протокола TCP не обязательны. Заголовок UDP включает следующие четыре поля: поле порта источника (source port), поле порта пункта назначения (destinator port), поле длины (length) и поле контрольной суммы.

В-четвертых, транспортный протокол сетевых новостей NNTP (NETWORK NEWS TRANSPORT PROTOCOL), который используется для передачи сетевых новостей.

В-пятых, простой протокол передачи почты SMTP (SIMPLE MAIL TRANSFER PROTOCOL), который предназначен для передачи электронной почты.

В-шестых, шлюзовые протоколы, которые предназначены для передачи по сети сообщений о маршрутах и передачи

информации о состоянии сети. Например, протокол GATEWAY PROTOCOL.

В-седьмых, гипертекстовый протокол передачи данных HTTP (HYPER TEXT TRANSFER PROTOCOL), который используется для работы с большими гипертекстами, расположенными в нескольких абонентских системах сети. Протокол HTTP определяет взаимодействие на прикладном уровне. Он предназначен для передачи сообщений, являющихся блоками гипертекста. Этот протокол используется для взаимодействия клиентов с программами шлюзов INTERNET, разрешающих доступ к ресурсам электронной почты, файлам и программам глобальной информационной системы. Протокол HTTP характеризуется следующими особенностями:

во-первых, в протоколе реализуется принцип 'запрос/ответ';

во-вторых, протокол HTTP использует технологию окон и пиктограмм;

в-третьих, протокол позволяет абонентам передвигаться по гиперсреде;

в-четвертых, протокол позволяет отображать текст, графику, анимацию и воспроизводить звук.

В-восьмых, транспортный протокол реального времени RTP (REAL-TIME TRANSPORT PROTOCOL), который гарантирует доставку данных одному или более адресатам с задержкой в заданных пределах, т.е. данные могут быть воспроизведены в реальном времени. В приложениях реального времени отправитель генерирует поток данных с постоянной скоростью, а получатель или получатели должны предоставлять эти данные приложению с той же самой скоростью. Такие приложения включают, например, аудио- и видео конференции, живое видео, удаленную диагностику в медицине, компьютерную телефонию, распределенное интерактивное моделирование, игры, мониторинг в реальном времени и др.

4.3. Программы, обеспечивающие просмотр гипертекстовых документов

Программное обеспечение, используемое для просмотра гипертекстовых документов называется браузером (WEB BROWSER) или WWW-навигатором. Браузер выдает информацию о том, где и какие связи имеются в документе. В браузерах текст, имеющий ссылку, отмечается специальным образом, не нарушающим общий вид документа на экране. Для управления работой с документами используются меню и стандартные элементы управления.

Браузеры являются программами-клиентами.

Возможности браузера подразделяются на два уровня:

во-первых, встроенные (INLINE) возможности;

во-вторых, возможности вспомогательных программ (HELPER APPLICATIONS).

Встроенные возможности позволяют работать с наиболее распространенными форматами графических файлов, файлов анимации и звуковых файлов.

Браузеры имеют модульную структуру и, при необходимости, могут быть дополнены требуемыми программами.

Можно выделить такие браузеры как Netscape Navigator и Internet Explorer. Указанные браузеры разработаны конкурирующими фирмами. Netscape Navigator разработан корпорацией NETSCAPE COMMUNICATIONS. Internet Explorer разработан корпорацией MICROSOFT. Фирма MICROSOFT включает Internet Explorer в состав операционной системы WINDOWS .

Возможности данных программных продуктов в основном одинаковые. Поэтому, при освоении одного из браузеров, работа с другим браузером не создает проблем.

При запуске браузера Internet Explorer на экране появляется основное окно с изображением начальной страницы (START PAGE). В окне имеются следующие основные части.

Во-первых, строка заголовка (TITLE BAR), которая содержит стандартные элементы окна WINDOWS приложения.

Во-вторых, область просмотра документа (DOCUMENT VIEWING AREA), в которой отображается текст документа со встроенными изображениями .

В-третьих, строка меню (MENU BAR), которая содержит пункты меню, обеспечивающие доступ ко всем необходимым функциям.

В-четвертых, строка адреса (LOCATOR BAR), которая предназначена для указания сетевого адреса текущего документа.

В том случае, если необходимо обратиться к конкретной странице, то в строке адреса вводится адрес требуемой страницы в формате URL.

Если необходимо найти конкретную информацию по какой-либо теме, то используется служба поиска. Имеется значительное число систем, предназначенных для поиска информации.

Можно выделить следующие поисковые системы:

Во-первых, поисковая система YAHOO, которая представляет собой иерархический предметно-ориентированный указатель основных ресурсов WWW и INTERNET.

Во-вторых, поисковая система MAGELLAN, которая содержит указатель наиболее популярных ресурсов INTERNET и тематический указатель. На многих популярных WWW-серверах можно увидеть специальный значок, присваиваемый службой Magellan. Этот значок свидетельствует о том, что указатели поисковой системы Magellan содержат ссылки на документы данного сервера.

В-третьих, поисковая система ALIWEB, применяемая для поиска WWW-ресурсов в странах Европы.

Какая из систем предоставляется в распоряжение пользователя, как правило, зависит от провайдера узла.

Обращение к службе поиска обеспечивается через управляющий элемент SEARCH (ПОИСК). Окно службы поиска позволяет определить режим поиска и задавать ключевую информацию. Результатом работы службы поиска является список документов с указанием общего числа страниц и

количества страниц, содержащих ключевую информацию. При необходимости обеспечивается обращение к найденным документа.

Пример ключевого выражения, обеспечивающего поиск электронного учебника по программированию при использовании поисковой системы "RAMBLER": «ПРОГРАММИРОВАНИЕ ЭЛЕКТРОННЫЙ УЧЕБНИК».

Для облегчения поиска ключевой информации на просматриваемой странице можно использовать пункт "Найти (на данной странице)" (FIND (on this page)) в меню "Правка" (EDIT).

Располагаемая на странице информация может быть выделена, скопирована в буфер обмена и вставлена в текстовый документ.

Текущая страница может быть сохранена в виде отдельного файла. Для этого необходимо выполнить команду "Сохранить как файл" (SAVE AS) в меню "Файл"(FILE). Информация может быть сохранена в виде простого текста (*.TXT) или в виде файла в формате HTML.

При просмотре русскоязычных WEB-страниц может возникнуть проблема чтения страницы из-за несоответствия шрифтов. Для преобразования текста можно использовать различные возможности меню "ВИД" (VIEW).

В Internet Explorer сохраняются сведения о пяти страницах, которые были рассмотрены последними. Для возврата на одну из ранее рассмотренных страниц целесообразно использовать меню "Переход" (GO).

Чтобы прекратить загрузку страницы, отображение которой требует слишком много времени можно использовать кнопку "Стоп" (STOP), расположенную на панели управления.

Internet Explorer позволяет при просмотре WEB-страницы отобразить рассматриваемую страницу в виде исходного текста на языке HTML. Для этого достаточно выбрать пункт "Источник" (SOURCE) из меню "Вид" (VIEW).

4.4. Технологии программирования, основанные на использовании специальных языков, предназначенных для работы в сети INTERNET

Имеется совокупность технологий программирования, которые предназначены для создания программных продуктов, ориентированных на обработку информации в сети Internet. Данные технологии быстро развиваются, а также появляются новые технологии.

4.4.1. Язык разметки гипертекстов HTML

Гипертекстовые документы описываются на специальном языке HTML (HYPER TEXT MARKUP LANGUAGE, язык разметки гипертекстов). Команды этого языка указывают вид и расположение рисунков, ссылки на другие ресурсы и т.д.

Язык HTML является инструментальным программным обеспечением, использующим технологию гипертекста при создании разнообразных документов. Главной задачей этого языка является придание документам стандартной для глобального соединения формы.

Использование HTML предоставляет следующие возможности:

во-первых, позволяет описывать документы и их составляющие;

во-вторых, позволяет указывать ассоциативные связи между документами;

в-третьих, позволяет просматривать документы и находить в них необходимые сведения;

в-четвертых, позволяет выполнять обработку документов.

WWW файл, содержащий разметку документа, может формироваться и модифицироваться с помощью таких текстовых редакторов как Notepad (Блокнот) или FrontPage Editor. Использование текстового редактора Word может вызвать проблемы с совместимостью.

Сформированный HTML документ, как правило, должен иметь расширение '**.HTM**' или '**.HTML**'.

Совокупность элементов языка HTML, управляющих отображением текста называется разметкой документа (MARKUP). Элемент (element) разметки состоит из пары кодов, которые называются тегами (TAG). Первый тег называются открывающим тегом. Второй тег называется завершающим тегом. Завершающий тег отличается от открывающего тега тем, что начинается с символа '/'. Каждый элемент имеет имя, которое соответствует тегам.

Между открывающим и завершающим тегами могут присутствовать текст или данные. HTML-документ, как правило, имеет следующую структуру

```
<HTML>
<HEAD>
<TITLE>Заголовок HTML-документа</TITLE>
</HEAD>
 <BODY>
    Тело HTML-документа
 </BODY>
</HTML>
```

При создании HTML-документа, могут быть использованы, в частности, следующие теги:

Во-первых, теги "HTML", которые указывают на то, что заключенные в эту пару тегов данные представлены в формате HTML;

Во-вторых, теги "HEAD", которые определяют часть документа, в которой указаны общие сведения об HTML-странице;

В-третьих, теги "TITLE", которые определяют название HTML-документа;

В-четвертых, теги "BODY", которые определяют часть документа, в которой находится его основное содержание;

В-пятых, теги "FONT … ", которые обеспечивают выбор шрифта, его размера и цвета;

В-шестых, теги "CENTER", которые размещают ограниченный парой тегов текст по центру страницы;

В-седьмых, теги "IMG', которые служат для размещения в документе встроенного изображения;

В-восьмых, теги "A", которые определяют якорь (Anchor). Якорь (в некоторях переводах ссылка), представляет собой либо исходную точку гиперссылки, либо ее точку назначения. Теги якоря могут записываться в соответствии со следующим синтаксисом:

```
<A HREF="Адрес ссылки"> Текст ссылки </A>
```

Адрес сылки может быть представлен различными вариантами.

В частности, в качестве адреса ссылки может быть использован URL. Например, тег содержащий гиперссылку на домашнюю страницу фирмы Microsoft, может иметь следующий вид:

```
<A HREF='HTTP://WWW.MICROSOFT.COM/' >
ДОМАШНЯЯ СТРАНИЦА MICROSOFT </A>
```

Конструкция "Текст ссылки", определяет специально выделенный текст, изображаемый на экране. При наложении курсора на выделенный текст, меняется его форма.

Теги якоря могут быть использованы для вызова из сформированной страницы программных продуктов, разработанных в различных программных системах. Пример. HTML документ, обеспечивающий формирование страницы для вызова программного продукта, разработанного в среде Visual Basic, может быть представлен в следующем виде:

```
<html>
<head>
<title> Страница вызова программ </title>
<body>
 <FONT SIZE=+3>
 <Center> ПРОГРАММЫ, РАЗРАБОТАННЫЕ НА РАЗЛИЧНЫХ ЯЗЫКАХ</Center>
</FONT>
 <A HREF="C:\SMIRNOV\Example_aver\project_aver.exe">
  <FONT COLOR="RED"> Программа на VB </Font></A>
</body>
</html>
```

Для просмотра созданного HTML-документа можно использовать Web-броузер. Для этого предназначена команда «Открыть» в меню «Файл».

Специальные программы, предназначенные для работы с языком HTML, подразделяются на HTML - конверторы и HTML - редакторы.

HTML - конверторы, представляют собой программу, которая преобразует существующий документ в набор HTML страниц. В гипертекстовый вид могут быть преобразованы файлы текстовых процессоров, таблицы базы данных, электронные таблицы и другая информация.

HTML – редакторы представляют собой программы, которые предназначены для создания новых WEB документов.

HTML – редакторы можно подразделить на две категории:

во-первых, редакторы, обеспечивающие строгий контроль синтаксиса,

во-вторых, редакторы, не предусматривающие контроль синтаксиса.

Если используется контроль синтаксиса, то созданный текст будет содержать исключительно известные данному редактору команды разметки страниц. Отсутствие синтаксического контроля позволяет применять нестандартные команды.

Дальнейшим развитием стандарта HTML является язык XML (eXtended Markup Language). Данный язык позволяет реализовать объектный подход к созданию Internet-компонента и структурированную передачу и обработку данных через Internet.

4.4.2. Технология FLASH

Технология FLASH (flash-сверкнуть, блеснуть) предоставляет возможность разработчикам сайтов использовать анимацию, звук, графику на новом технологическом уровне. Данная технология предназначена для разработки эффективного прикладного программного обеспечения в таких на-

правлениях, как интернет-магазины, электронная биржа и т.п.. Flash технология достаточно широко используется во всем мире профессиональными разработчиками сайтов.

Технология FLASH разработана компанией Macromedia. Для реализации FLASH технологии предназначена специальная среда “Macromedia Flash”. Данная программная среда обеспечивает удобный интерфейс и обеспечивает создание программных продуктов, откомпелированных в файле с расширением “*.SWF”. При программировании в среде FLASH используется язык “ActionScript”. Программные конструкции данного языка, в значительной степени аналогичны конструкциям языка JAVA. В языке ActionScript имеется возможность подключения кодов из внешних файлов, написанных на языке JAVA. Программная среда “Macromedia Flash” имеет специальные «конструкторы», которые обеспечивают генерацию программного кода.

4.4.3. Использование языка JAVA

JAVA представляет собой язык, специально разработанный для работы в открытой сетевой среде. Текст программы, написанной на языке JAVA, может компилироваться в бинарный псевдокод и передаваться по сети для исполнения на виртуальной машине в удаленном интерпретаторе. Такие передаваемые по сети программы называются аплетами (APPLETS). С серверов сети INTERNET могут вызываться не только программы, но и описания объектов или форматов данных.

Принципиально новым в JAVA технологии (JAVA TECHNOLOGY) является то, что при обработке информации могут быть использованы различные виды компьютеров и устройств. Примечательно, что диапазон, используемых в JAVA технологии устройств, необычайно широк: от суперкомпьютеров, до простейших вычислительных устройств, встроенных в телевизоры и телефоны. Компоненты JAVA технологии не зависят от видов компьютеров и операционных систем, в которых

они используются. Использование JAVA технологии значительно упрощает связи между различными вычислительными устройствами, объединенными в глобальную сеть.

Для работы с языком JAVA, как правило, используются WWW-браузеры, которые должны уметь вызвать для исполнения аплетов JAVA-интерпретатор. В частности могут быть использованы такие браузеры как Internet Explorer и Netscape Navigator.

JAVA позволяет решать такие проблемы, как отсутствие интерактивности; ограниченный контроль вида документа; ограниченный набор форматов встроенной графики и других объектов мультимедиа. Например, с использованием языка JAVA можно создать документ и включить в него рисунок во вновь разработанном уникальном формате. Для чтения рисунка, в этом случае, делается ссылка на программу, которая умеет читать уникальный формат.

Использование в языке JAVA технологии объектно-ориентированного программирования позволяет объединять в одном приложении как документы, так и методы их обработки. Данная технология дает возможность построения средствами JAVA больших корпоративных информационных систем. В этом случае, данные хранятся на одном сервере, обрабатываются на другом, а отображаются на JAVA-терминалах.

Во всех реализациях платформы JAVA имеется стандартный набор системных программ, выполняющих функции операционной системы. Поэтому, приложения, написанные на языке JAVA, могут выполняться без операционной системы.

4.4.4. Использование сокетов (SOCKET)

Сокет (socket – гнездо, муфта, розетка) представляет собой конечную точку сетевых коммуникаций. Каждый использующиеся сокет имеет тип и ассоциированный с ним процесс. Сокет существует внутри доменов сети Internet.

В Internet домене сокет организован, как комбинация IP адреса и номера порта, которая однозначно определяет отдель-

ный сетевой процесс во всей глобальной сети Internet. Для обмена информацией формируются два сокета: первый сокет предназначен для отправки сообщений; второй сокет предназначен для приема сообщений. Оба сокета должны быть настроены на один и тот же протокол, предназначенный для передачи информации. Например на протокол TCP или протокол UDP.

Сокеты могут быть подразделены на следующие типы:

Во-первых, Stream socket, который обеспечивает последовательный надежный, ориентированный на установление двусторонней связи поток байтов.

Во-вторых, Datagram socket, который поддерживает двусторонний поток данных, с предопределенными границами записи данных. При использовании данного сокета не гарантируется, что этот поток будет последовательным, надежным и что данные не будут дублироваться.

В-третьих, Raw socket, которые обеспечивает возможность пользовательского доступа к низлежащим коммуникационным протоколам.

Сокеты создаются и настраиваются при помощи библиотеки “WinSock”. Данная библиотека находится в файле “WINSOCK.DLL”, который входит в стандартный набор установки операционной системы Windows.

4.4.5. VBScript

VBScript входит в семейство языков Visual Basic, включающее Visual Basic(VB), Visual Basic for Application (VBA) и Visual Basic Script(VBScript). VBScript представляет собой инструмент для написания приложений, функционирующих в программах, работающих в Internet. VbScript позволяет разрабатывать клиентские приложения, автоматически загружаемые вместе с WEB-страницей. Затем скрипты могут выполняться на клиентской ЭВМ как обычные программы. VBScript был разработан для применения в корпоративных Интранет-сетях для создания приложений клиент-сервер. VBScript поддерживается броузером InternetExplorer.

4.4.6. Perl

Оъектно-ориентированный язык PERL (Practical Extractionand Report Language) является переносимым, интерпретируемым языком, хорошо приспособленным для фильтрации и преобразования текста. Интерпретаторы PERL являются бесплатными программными продуктами. PERL находит широкое распространение в Internet в среде UNIX. Язык PERL в значительной степени похож на язык "C".

Язык PERL, в частности, предоставляет программисту следующие возможности:

Во-первых, автоматическое преобразование типов;

Во-вторых, автоматическое изменение размера массивов;

В-третьих, форматированный вывод с генерацией отчетов на основе шаблонов;

В-четвертых, функции обработки списковых структур данных;

В-пятых, сетевые операции по сокетам.

4.5. Программирование в среде Delphi, с использованием сети Internet

В Delphi имеются возможности, которые позволяют обеспечивать связь разрабатываемой программы с глобальной сетью Internet. Delphi позволяет использовать различные сервисы сети Internet, в частности передавать электронную почту, разнообразные файлы, устраивать телеконференции, а также использовать всемирную паутину World Wide Web (WWW). В Delphi имеется возможность использовать для передачи данных пакеты XML (XML data packets).

Delphi может использоваться также и при работе с сетями INTRANET, которые позволяют использовать технологии INTERNET внутри предприятия.

4.5.1. Компоненты DELPHI, предназначенные для работы в WWW

Для работы в World Wide Web предназначены компоненты, расположенные на странице Internet. Можно выделить следующие из данных компонент:

Во-первых, компонент ClientSocket. Компонент ClientSocket (клиентское соединение) создает для экранной формы или модуля обработки данных (data module) соединительный компонент (socket), обеспечивающий получение информации клиентом при помощи протокола TCP/IP. С помощью этого компонента программа устанавливает связь с протоколом TCP/IP сервера. Для каждого клиента необходим отдельный компонент ClientSocket;

Во-вторых, компонент ServerSocket. Компонет ServerSocket (серверное соединение) создает соединительный компонент (socket), обеспечивающий принятие запроса (request) клиента. С помощью этого компонента программа устанавливает связь с протоколом TCP/IP клиента. Для каждого клиента необходим отдельный компонент ServerSocket;

В-третьих, WebDispatch. Компонет WebDispatch (диспетчер Web), обрабатывает клиентские HTTP (HYPER TEXT TRANSFER PROTOCOL) -сообщения с требованиями выполнения тех или иных действий. Данный компонент позволяет обрабатывать сообщения, созданные с помощью языка XML;

В-четвертых, компонент TPageProducer. Компонет TPageProducer (поставщик страниц), передает клиенту строку HTML-команд описания шаблонов страниц. Шаблоны включают HTML-команды и HTML-этикетки (tag, теги), которые заменяются по требованию пользователя заданным содержанием при наступлении события OnHTMLTag;

В-пятых, компонент NMHTTP, который реализует протокол передачи гипертекста. Данный компонент руководит передачей HTTP-трансферов (transfers) через World Wide Web;

В-шестых, компонент TNMURL, который кодирует символьную строку в формат URL (USER RESOURCES LOCATOR,

UNIFORM RESOURCE LOCATOR, UNIVERSAL RESOURCE LOCATOR) для выполнения HTTP трансмиссии (transmission). Данный компонент может использоваться для перекодировки URL в символьную строку;

В-седьмых, компонент TWEBMODULE, который автоматически генерирует WEB-модуль (WEB MODULE) при создании нового WEB-приложения. Web модуль служит в качестве хранилища для невизуальных компонентов, таких как TPageProducer.Web-приложение может иметь только один WEB-модуль.

4.5.2. Компоненты DELPHI, предназначенные для работы в других сервисах INTERNET

Можно выделить следующие компоненты, предназначенные для работы в других сервисах Internet:

Во-первых, компонент NMMsg, предназначенный для передачи простых текстовых сообщений по сети Internet. Данный компонент предоставляет пользователю широкие возможности для решения различных проблем, связанных с конкретными особенностями передачи данных;

Во-вторых, компонент NMFTP, который предназначен для передачи данных с использованием протокола FTP (FILE TRANSFER PROTOCOL, FILE TRANSPORT PROTOCOL). При работе этого компонента требуется специальный модуль, использующий стековую организацию памяти. Данный компонент обеспечивает работу с файлами, созданными в различных операционных системах;

В-третьих, компонент NMNNTP, который используется для чтения и передачи новостей при помощи протокола NNTP (NETWORK NEWS TRANSPORT PROTOCOL);

В-четвертых, компонент NMPOP3, который используется для получения электронной почты (E-Mail) при помощи протокола POP (POST OFFICE PROTOCOL). При работе этого компонента требуется специальный модуль, использующий стековую организацию памяти. Данный компонент обеспечи-

вает работу с файлами, созданными в различных операционных системах;

В-пятых, компонент NMSMTP, который используется для отправления электронной почты (E-Mail) при помощи протокола SMTP (SIMPLE MAIL TRANSFER PROTOCOL). Кроме того, данный компонент является инструментом для других команд, специфицированных в стандартах RFC (REQUEST FOR COMMENT). Стандарты RFC определяют создание и обновление инструментальных средств интернет IETF (INTERNET ENGINEERING TASK FORCE);

В-шестых, компонент NMStrm, предназначенный для отправления потока данных по сети Internet. Данный компонент предоставляет пользователю широкие возможности для решения различных проблем, связанных с конкретными особенностями передачи данных;

В-седьмых, компонент NMUDP, который используется для отправления блоков данных при помощи протокола UDP (USER DATAGRAM PROTOCOL);

В-восьмых, компонент TPowerSock, который используется как родительский класс для разработки новых компонентов, реализующих нестандартные протоколы;

В-девятых, компонент TNMGeneralServer, который обеспечивает создание базовых классов для развития интернетовских серверов. Данный компонент поддерживает стандарт RFC (REQUEST FOR COMMENT).

4.5.3. Компоненты DELPHI, предназначенные для работы в INTRANET

Можно выделить следующие компоненты, предназначенные для работы в Intranet:

Во-первых, компонент TQueryTableProducer. Компонент TQueryTableProducer (поставщик табличных запросов), используется в Intranet для преобразования запроса, разработанного на языке SQL (Structured Query Language), в серию команд языка HTML;

Во-вторых, компонент TDataSetTableProducer. Компонет TDataSetTableProducer (поставщик табличных наборов данных), используется в Intranet для создания серии команд языке HTML, обеспечивающих передачу клиенту информации из табличных наборов данных;

В-третьих, компонент TNMDayTime, которые используется в Intranet для получения от сервера данных типа дата-время. Данные передаются в соответствии со стандартом RFC. При работе этого компонента требуется специальный модуль, использующий стековую организацию памяти. Данный компонент обеспечивает работу с различными операционными системами.

4.5.4. Использование InternetExpress

InternetExpress представляет собой набор компонентов, позволяющих реализовать полный цикл клиент-серверной обработки данных на базе Internet с использованием как уже имевшихся в распоряжении разработчиков на Delphi средств, так и новых средств, например стандарта XML (eXtended Markup Language).

В InternetExpress реализована специальная технология поддержки XML на основе JavaScript.

С точки зрения VCL (Visual Components Library, библиотеки визуальных компонент) InternetExpress представляет собой две компоненты базового набора: TXMLBroker и TMIDASPageProducer.

TXMLBroker отвечает собственно за формирование XML-пакета, реакцию на изменения в данных и оповещение о действиях, выполняемых клиентом.

TMIDASPageProducer отвечает за формирование сборного DHTML-документа, который, собственно, и является клиентским приложением, поскольку содержит все те визуальные элементы, которые соответствуют структуре пакета данных XML(XML data packets).

В этот документ передаются XML-пакеты, формируемые компонентом XMLBroker. В тот момент, когда от клиентского приложения приходит сообщение о необходимости передать изменения в данных на сервер приложений, TMIDAS-PageProducer выполняет следующие действия:

Во-первых, осуществляет опрос каждого из элементов управления HTML;

Во-вторых, формирует пакет с данными, подлежащими обновлению;

В-третьих, передает сформированный пакет серверу приложений.

Таким образом, обработка данных на клиенте происходит с использованием средств HTML, а передача структурированных данных к клиенту и изменений от него осуществляется при помощи пакетов данных XML.

Указанные компоненты помещаются в web-модуль (WebModule) серверного приложения. Серверная часть приложения состоит из следующих элементов:

Во-первых, исполняемого модуля, написанного на Delphi;

Во-вторых, из Web-модуля, включающего компоненты InternetExpress;

В-третьих, из файлов-библиотек JavaScript.

Клиентская часть приложения, созданного на основе InternetExpress представляет собой собственно HTML-документ, порожденный одним или более компонентами типа TMIDASPageProducer.

В целом, схема работы приложения на основе InternetExpress выглядит следующим образом:

Во-первых, браузер (browser) обращается по ссылке (URL) к серверному приложению InternetExpress, которое возвращает HTML-документ, являющийся, как правило, некой отправной точкой в алгоритме обработки данных.

Во-вторых, по запросу пользователя серверное приложение возвращает очередной HTML-документ, содержащий ссылки на библиотеки JavaScript, отвечающие за обработку XML-пакетов.

Затем данный документ посылает запрос серверной части приложения. Серверная часть приложения посылает клиенту данные в виде пакетов XML, интерпретируемых соответствующими библиотеками JavaScript.

В-третьих, после того, как пользователь просмотрел набор данных и, при необходимости, внес в них изменения, он имеет возможность передать изменения серверной части приложения. Это процесс запускается событием, которое, как правило, связано с "кнопкой" "Apply Updates" (Внести изменения) и передается серверной части приложения InternetExpress, а именно - компоненту TMIDASPageProducer. Все изменения в данных передаются серверной части приложения в виде разностных пакетов XML (XML delta packets).

В-четвертых, серверная часть получает информацию об изменениях в данных и использует сервер приложений для внесения этих изменений в базу данных. При возникновении конфликта (reconcile error) имеется возможность сформировать HTML-вариант Reconcile Dialog (reconcile-примирять) из состава Delphi или разрешить конфликтную ситуацию автоматически, включив компонент TReconcilePageProducer в состав серверной части приложения.

4.6. Обеспечение безопасности при работе в INTERNET

При работе в сети INTERNET возможно возникновение следующих проблем, связанных с нарушением безопасности.

Во-первых, конфиденциальные документы, хранящиеся в каталогах WEB-сервера, могут попасть в руки посторонних пользователей.

Во-вторых, конфиденциальная информация, отправленная на сервер удаленным пользователем, может быть перехвачена.

В-третьих, может произойти утечка информации о компьютере, на котором установлен WEB-сервер. При этом воз-

никает угроза проникновения посторонних лиц и получения ими доступа к данным.

В-четвертых, посторонние лица могут выполнять на компьютере WEB-сервера команды, изменяющие и повреждающие программное обеспечение. В частности, может быть выполнена атака на систему запрета доступа. При этом атакующие посылают на компьютер такое большое количество запросов, что система запрета доступа выходит из строя.

Уровень безопасности в значительной степени зависит от используемой операционной системы. Существует правило, в соответствии с которым, чем большей мощностью и гибкостью обладает операционная система, тем больше она страдает от нападений.

Уязвимыми являются, в частности, UNIX-системы с большим количеством серверов, сервисов, языков и интерпретаторов. В перечисленных элементах могут найтись лазейки. Операционные системы MICROSOFT WINDOWS в меньшей степени подвержены взлому. Однако система UNIX, управляемая опытным UNIX-администратором, обеспечит более высокую степень защиты, чем MICROSOFT WINDOWS, установленная недостаточно опытным программистом.

Тема 5.

COM-ТЕХНОЛОГИИ и их использование при обработке экономической информации

5.1. Основные понятия COM технологий

Продукты фирмы Microsoft обладают огромными функциональными возможностями. Для более полного использования имеющихся возможностей разработана COM модель (COMPONENT OBJECT MODEL, компонентная модель объектов, многокомпонентная модель объектов, модель многокомпонентных объектов). Данная модель позволяет при обработке конкретной информации использовать несколько прикладных программных продуктов в максимальной степени раскрывая их функциональные возможности.

Технология COM является стандартом корпорации Microsoft на взаимосвязи между объектами. Данный стандарт регламентирует отправление и получение сообщений. Использование COM позволяет объектам, написанным на различных языках программирования связываться друг с другом так, как если бы они были написаны на одном и том же языке. Например, могут быть связаны объекты, разработанные на таких языках как Visual FoxPro , Visual Basic, Visual C++, Delphi.

Данная технология предназначена для того, чтобы одна программа (клиент) смогла заставить работать объект, являющийся частью другой программы (сервера). На COM базируются такие технологии как OLE, ActiveX.

Необходимость стандартизации определяется, в частности, наличием различных механизмов доступа к различным видам программного обеспечения. Например:

Во-первых, приложения, скомпонованные с библиотекой, могут пользоваться ее сервисами, вызывая функции из этой библиотеки;

Во-вторых, два локальных процесса могут взаимодействовать посредством передачи сообщений, в соответствии с заданным протоколом;

В-третьих, приложение, использующее сервисы операционной системы, обычно выполняет системные вызовы, обрабатываемые операционной системой;

В-четвертых, приложения, использующие различные средства сетевого обмена.

Можно выделить несколько разновидностей использования COM технологий:

Во-первых, использование визуальных объектов. Визуальные объекты могут быть разработаны на любом из языков, поддерживающих COM стандарт. Например, управляющие элементы ActiveX, такие как календарь связываются с Visual FoxPro при помощи COM технологии.

Во-вторых, невизуальные COM объекты. Например, можно использовать COM объекты для разработки специализированного ценового калькулятора брокерской компании. В этом случае, COM объект может быть разработан в группе, отвечающей за секретный ценовой алгоритм компании. Разработанный объект должен включать методы и свойства, которые будут использоваться для обработки информации. Однако, нет необходимости делать данный объект визуальным.

В-третьих, открытые объекты. Данные объекты содержат открытые, незащищенные (EXPOSED) от внешнего мира части. Эти объекты предназначены для использования не только пользователями, но и разработчиками прикладного программного обеспечения. Например, приложения, обеспечивающие работу с факс-документами (faxing application) могут иметь объекты открытые для разработчиков. В этом случае разработчики могут обеспечить автоматическое формирование факс-документов.

Приложения, использующие COM-технологии являются открытыми для разработчика и могут быть использованы для управления на расстоянии (remote control).

Основываясь на технологии объектно-ориентированного программирования, COM стандарт позволяет решить про-

блему монолитности приложений и сделать их открытыми. Разработчик может использовать классы одного приложения для строительства других приложений. Посредством управления на расстоянии обеспечивается прямой доступ к заданному приложению.

Идеи, заложенные в COM технологии многие специалисты считают назвать революционными по своей сути. Данные идеи меняют не только языки программирования, но также прикладное программное обеспечение в сфере экономики. Например, применение COM технологии позволяет использовать одну и ту же банковскую ценовую модель в различных прикладных программах без изменения программного кода. Начав скромно, как способ создания составных документов, COM развилась в фундаментальную основу прикладного и системного программного обеспечения.

При реализации COM-технологии допустимы следующие особенности:

Во-первых, клиент и сервер могут находиться на компьютерах, расположенных в разных странах;

Во-вторых, компьютеры, на которых находятся программы, могут быть разного типа. Например, IBM-совместимый компьютер и рабочая станция SUN;

В-третьих, обе программы могут быть написаны на разных языках;

В-четвертых, данные программы могут исполняться под управлением разных операционных систем.

5.2. Интерфейс COM-объектов

Программы, созданные с использованием COM технологии, предоставляют свои сервисы через один или несколько COM-объектов. Каждый такой объект является экземпляром некоторого класса и поддерживает определенное количество интерфейсов, обычно не менее двух. В состав каждого интерфейса входит один или более методов, т.е. функций которые

могут вызываться клиентом объекта. Чтобы вызывать любой из методов, у клиента объекта должен быть указатель на интерфейс, содержащий соответствующий метод.

Каждый поддерживаемый объектом интерфейс, может рассматриваться аналогично контракту между этим объектом и его клиентами. В соответствии с контрактом COM-объект обязуется поддерживать методы интерфейса, а COM-клиент корректно вызывать методы.

Для определения интерфейсов в технологии COM имеется специальный язык IDL (Interface Definition Language, язык описания интерфейсов). С помощью IDL можно составить полную и точную спецификацию интерфейсов объекта COM.

Интерфейсы COM-объектов являются фиксированными. Это означает, что после того как интерфейс задействован в каком-либо программном обеспечении изменять его нельзя. Добавление новой или изменение существующей функциональности требует определения полностью нового интерфейса.

Каждый объект COM должен поддерживать интерфейс IUnknow. Все интерфейсы объекта являются потомками IUnknow. С помощью интерфейса IUnKnow он можно получить доступ к другим интерфейсам объекта.

У каждого интерфейса COM-объекта два имени.

Первое имя, представляет собой легко воспринимаемое человеком симольное имя. Как правило, читабельные имена COM-интерфейсов начинаются с буквы “I” (от слова Interface). Читабельные имена имеют смысловую нагрузку, описывающую назначение интерфейса. Например, IspellChecker (время задержки). Допускается, чтобы читабельное имя было одинаковым у двух интерфейсов различных COM-объектов.

Второе имя интерфейса используется программным обеспечением. Данное имя является уникальным и называется GUID(GLOBAL UNIQUE IDENTIFIER, глобально уникальный идентификатор).

Получив интерфейс внешнего COM-объекта, клиент может его использовать так же как и свои собственные объекты.

5.3. Идентификаторы, используемые в COM технологии

Важнейшей частью COM технологии является совокупность идентификаторов, используемых при работе с COM объектами. Можно выделить следующие идентификаторы:

Во-первых, идентификатор GUID (GLOBAL UNIQUE IDENTIFIER), который называется глобально уникальным идентификатором. GUID представляет собой программно генерируемую 16 байтовую величину уникальную во времени и пространстве. Уникальность во времени достигается за счет того, что каждый GUID содержит метку времени, указывающую, когда он был создан. Метка времени гарантирует отличие друг от друга всех GUID, сгенерированных на данной машине. Для обеспечения уникальности в пространстве при генерации GUID используется адрес платы сетевого интерфейса, который уникален у каждого компьютера.

Во-вторых, идентификатор интерфейса IID (INTERFACE IDENTIFIER), который является частным случаем идентификатора GUID.

В-третьих, идентификатор CLSID (CLASS IDENTIFIER), который определяет класс предназначенный для описания параметров интерфейса. Всякий COM-объект является экземпляром некоторого класса, и каждому классу может быть присвоен CLSID.

В-четвертых, идентификатор PROGID (PROGRAMMATIC IDENTIFIER), который определяет версию прикладного программного продукта. Например, Visual FoxPro версия 6 имеет PROGID следующего вида:

Visual.FoxPro.Application.6

В то время как Visual FoxPro версия 5 имеет PROGID следующего вида:

Visual.FoxPro.Application.5

Идентификатор PROGID является именем класса и может быть использован для подтверждения (instantiate) COM сервера.

Однако, нет необходимости однозначно задавать необходимую версию. В этом случае, система выбирает последнюю версию из имеющихся. Для этой цели используется специальный идентификатор

VersionIndependentProgID.

Когда приложение определяется как COM сервер оно регистрируется в специальном регистре WINDOWS REGISTRY.

5.4. Инструментарий, обеспечивающий создание COM-объектов в системе Delphi

В системе DELPHI имеется совокупность мастеров (WIZARD), которые позволяют создавать COM-объекты различных видов. С помощью инструментария WIZARD могут быть созданы такие системные компоненты как: AUTOMATION CONTROLLER, AUTOMATION SERVER, ACTIVE X, ACTIVE SERVER PAGE.

Мастер COM-объектов выполняет следующие задачи:

Во-первых, создает новый элемент (UNIT);

Во-вторых, описывает новый класс в соответствии с системным компонентом TCOMOBJECT.

Системный компонент TCOMOBJECT, является базовым классом для работы с COM-объектами. Обрабатываемые COM-объекты должны иметь идентификатор класса (CLSID). TCOMOBJECT может быть использован как для работы с отдельными COM-объектами, так и для обработки COM-объектов, являющихся частью агрегатов(aggregate).

Процесс создания COM-объекта включает следующие шаги:

Во-первых, конструирование (DESIGN) COM-объекта;

Во-вторых, использование мастера для создания COM-объекта;

В-третьих, регистрация созданного COM-объекта;

В-четвертых, тестирование COM-объекта.

Подробное описание работы с мастером COM-объектов дано в MSDN (MICROSOFT DEVELOPER'S NETWORK DOCUMENTATION).

5.5.Особенности использования COM-технологий при программировании в среде Visual FoxPro

5.5.1. Создание COM объектов в Visual FoxPro

COM объект (COM OBJECT) используется подобно остальным объектам. Создается COM объект при помощи функции CreateObject, записываемой в соответствии со следующим синтаксисом

CREATEOBJECT(<имя COM-класса>)

Имя COM-класса состоит из двух слов. Первое слово ссылается на приложение, а второе слово определяет класс, используемый в приложении для организации доступа. Например: EXCEL.APPLICATION или WORD.APPLICATION. Имя COM-класса для конкретного приложения можно определить по технической документации или с помощью WINDOWS REGISTRY. Пример создания COM объекта для работы с табличным процессором EXCEL:

OBJCOMEXCEL = CREATEOBJECT("EXCEL.APPLICATION")

Создаваемый объект содержит описание, которое включает следующие элементы:

Во-первых, определение типов и имен, используемых в нем полей;

Во-вторых, количество и типы параметров обращения к доступным методам и свойствам;

В-третьих, имена методов, свойств и другие элементы.

Обработка созданного объекта обеспечивается при помощи технологии объектно-ориентированного программирования. Например, чтобы объект OBJCOMEXCEL стал видимым требуется задать свойству VISIBLE значение TRUE :

OBJCOMEXCEL.VISIBLE=.T.

Когда в приложении клиента выполняется функция CREATEOBJECT с именем COM класса, то клиент запрашивает операционную систему о затребованном объекте. Операционная система просматривает регистр WINDOWS REGISTRY,

чтобы найти GUID, принадлежащий COM объекту. В регистре имеется ссылка на требуемый файл с расширениями .EXE или .DLL. После определения файла вызывается требуемое приложение и обеспечивается необходимая обработка.

При работе с технологией DCOM для создания объектов используется функция CREATEOBJECTEX. Функция CREATEOBJECTEX реализована в версиях начиная с Visual FoxPro 6. Для своей работы данная функция требует как минимум, Windows 2000 с инсталлированной технологией DCOM.

Одним из основополагающих элементов COM-технологии является TYPE LIBRARY (библиотека типов). В этой библиотеке собраны объекты, методы и свойства, которые являются открытыми (exposed) для использования клиентскими COM приложениями. Если TYPE LIBRARY является частью файла объекта, то она имеет расширение '.OLB'. Если TYPE LIBRARY представляет собой отдельный файл, то она имеет расширение '.TLB'. В Visual FoxPro библиотеки типов выполняют значительно меньше функций, чем в таких системах как Visual Basic.

Для установления связи с уже запущенным (running) COM сервером предназначена функция GETOBJECT(). Функция GETOBJECT записывается в соответствии со следующим синтаксисом:

GETOBJECT (<имя файла> [, <имя COM класса>])

Параметр <имя файла> определяет файл, который необходимо открыть.

Параметр <имя COM класса> аналогичен данному параметру в функции CREATEOBJECT. В том случае, если расширение, указанное в имени файла, однозначно определяет заданное приложение, то имя COM-клиента задавать не обязательно. Например, для создания COM объекта для связи с файлом 'C:\TREND.XLS' достаточно задать следующую команду

ObExcel = GETOBJECT ('C:\TREND.XLS') .

При установлении связи с уже запущенным COM-сервером функция GETOBJECT может использовать OLE REGISTRY. Например, может открыть файл с расширением ".XLS", для которого установлена связь, в соответствии с тех-

нологией OLE. Если затребованный файл отсутствует, то выдается сообщение об ошибке.

По умолчанию, если система не может найти класс, специфицированный в функции CREATEOBJECT, то Visual FoxPro будет просматривать WINDOWS REGISTRY. Просмотр регистра может быть отменен заданием следующей команды

SET OLEOBJECT OFF.

Для получения информации о созданном COM-объекте можно использовать либо команду DISPLAY OBJECT, либо функцию ComClassInfo().

5.5.2. Использование Visual FoxPro в качестве COM-клиента

Использование Visual FoxPro в качестве COM-клиента (COM CLIENT) предусматривает написание программного файла с расширением '.PRG'. Программный файл должен включать в себя обращение к функции CREATEOBJECT и совокупность команд обработки созданного COM-объекта.

При использовании Visual FoxPro в качестве клиента целесообразно использовать коды, формируемые макрогенераторами таких прикладных программных продуктов, как EXCEL и WORD. Генерируемые макрогенераторами программные коды содержат большое число описательных констант. Все используемые константы документированы в приложениях TYPE LIBRARY.

Код, сгенерированный макрогенератором, представляет собой подпрограмму (SUB MACRO), написанную на языке Visual Basic. Данная подпрограмма редактируется и преобразуется в программу на языке Visual FoxPro (файл .PRG). Можно выделить следующие особенности преобразования:

Во-первых, в начало программы включается препроцессорная директива #INCLUDE. Данная директива обеспечивает подключение библиотеки описаний серверного прикладного программного продукта (header file). Например, при ис-

пользовании EXCEL включается препроцессорная директива следующего вида

```
#INCLUDE EXCEL.H
```

Во-вторых, сгенерированные конструкции преобразуются в обращения к свойствам COM объекта. При задании свойств COM объекта целесообразно широкое использование конструкций WITH/ENDWITH. В конце программного файла свойству VISIBLE COM объекта присваивается значение TRUE (.T.).

5.5.3. Создание COM-сервера с помощью Visual FoxPro

COM сервер (COM SERVER) может быть либо файлом с расширением '.EXE', либо файлом с расширением '.DLL'. Основная разница в использовании файлов '.EXE' и файлов '.DLL' в качестве COM-сервера заключается в том, что либо выполняется InProc сервер, либо выполняется OutProc сервер.

При организации в виде файлов '.DLL' выполняется InProc сервер, который использует память, выделенную приложению, обратившемуся к COM-серверу.

При организации в виде файлов '.EXE' выполняется OutProc сервер, который имеет свою собственную память. При использовании OutProc сервера операционная система специальным образом организует передачу информации в приложение обратившееся к COM серверу.

Таким образом, использование файлов '.EXE' обеспечивает большую надежность, а использование файлов '.DLL' большее быстродействие.

Как файлы '.DLL', так и файлы '.EXE' создаются Диспетчером проектов (PROJECT MANAGER) Visual FoxPro на основе любого программного файла.

Для создания COM-сервера могут быть использованы различные варианты программных файлов. В наиболее простом случае, COM-сервер представляет собой минимально возможную программу. Как правило, данную программу на-

зывают STUB (заглушка). Программный файл, выполняющий функции заглушки может иметь вид:

```
* Program STUB (заглушка)
* Используется для создания COM сервера
RETURN
```

В более сложном варианте, для создания COM-сервера может быть использован специально сформированный класс. Формирование класса обеспечивается с помощью команды DEFINE CLASS. В качестве базового класса для создания COM-сервера может быть использован любой базовый класс Visual FoxPro. Некоторые программисты для этой цели используют базовый класс LINE, т.к. он имеет минимальное количество надстроек. Процедуры, описывающие методы пользовательского класса, целесообразно определять с использованием опции PROTECTED. Например,

```
PROTECTED PROCEDURE  CDESCRIPTION
```

Опция PROTECTED означает, что данный метод может быть использован исключительно внутри класса и является невидимым вне класса.

После того, как сервер будет построен, для него генерируется глобальный уникальный идентификатор (GUID), который заносится в регистр (REGISTRY). Зарегистрированный сервер специфицируется либо для работы исключительно с отдельными требованиями (SINGLE INSTANTING), либо для работы с множественными требованиями (MULTIPLE INSTANTING).

После создания в Visual FoxPro COM-сервера к нему возможно обращение как клиентов, созданных в Visual FoxPro, так и клиентов, созданных в других приложениях, таких как Visual Basic, Excel.

При работе с Visual Basic следует учитывать, что в Visual Basic, в отличие от Visual FoxPro существует строгая типизация данных. В частности, должны быть типизированы и COM-объекты. Поэтому создание COM-клиента на языке Visual Basic может выглядеть следующим образом:

```
DIM Client_X AS New TS.TIMETRANS
SET Client_x = CREATEOBJECT ("TS.TIMETRANS")
```

где, TS.TIMETRANS – имя пользовательского (UDC) COM-класса.

В целом различия между конструкциями языка Visual Basic и языка Visual FoxPro незначительны. Поэтому, при создании COM-клиента на языке Visual Basic целесообразно преобразовывать программы, разработанные на языке Visual FoxPro.

При работе с COM-сервером, созданным в Visual FoxPro, могут быть полезны функции WIN32 API. Например, функция GetModuleFileName, которая определяет имя файла '.EXE', выполняемого в данный момент.

Создание COM-сервера позволяет перейти к программированию много-ярусных (много-звенных) клиент/серверных приложений (MULTY-TIERED CLIENT/SERVER APPLICATION). Под многоярусным приложением понимается приложение, созданное из нескольких логических частей. Пользователь видит верхний ярус программы (FRONT END). Информация запоминается в нижнем ярусе (BACK END). Средний ярус (MIDDLE END) обеспечивает взаимодействие между верхним и нижним ярусом.

Для разработки верхнего яруса корпорация Microsoft рекомендует использовать браузеры (BROWSER-BASED FRONT END). Для этой цели применяются такие программные средства как HTML (HYPER TEXT MARKUP LANGUAGE), DHTML (DYNAMIC HYPER TEXT MARKUP LANGUAGE), XML (eXtended Markup Language), ASP(Active Server Pages) и другие.

Нижний ярус многоярусных приложений, как правило, разрабатывается с использованием таких прикладных программных продуктов как Visual Foxpro, SQL-сервер (SQL SERVER).

Средний ярус организуется в виде COM сервера. Для создания среднего яруса целесообразно использоватьVisual FoxPro.

5.5.4. Использование функции ComArray

При программировании на Visual FoxPro обычно используются массивы, у которых нумерация элементов начинается с единицы. Однако, существуют языки программирования, в которых нумерация начинается с нулевого элемента.

Функция ComArray позволяет определить способ, с помощью которого массив, может быть передан в заданный COM объект. Visual FoxPro показывает каким образом отдельные элементы COM сервера должны обрабатываться. Функция записывается в соответствии со следующим синтаксисом:

COMARRAY(<COM-объект>, <порядок нумерации>)

Параметр COM-объект определяет используемый объект.

Параметр <порядок нумерации> может принимать следующие значения:

0 – нумерация элементов начинается с нуля и элементы передаются в COM объект по значения (BY VALUE);

1 – нумерация элементов начинается с единицы и элементы передаются в COM объект по значения (BY VALUE). Данный вариант определен как задаваемый по умолчанию (DEFAULT);

10 – нумерация элементов начинается с нуля и элементы передаются в COM объект по ссылке (BY REFERENCE);

11 – нумерация элементов начинается с единицы и элементы передаются в COM объект по ссылке (BY REFERENCE).

5.6.Технология DCOM

DCOM (DISTRIBUTED COMPONENT OBJECT MODEL, распределенная модель объектов, распределенная модель компонентных объектов) является протоколом, который позволяет компонентам программного обеспечения связываться через сеть используя принципы эффективности (EFFICIENT), надежности (RELIABLE) и безопасности (SECURITY). DCOM используется при выполнении распределенных приложений в

сети. Под распределенными системами понимают программные комплексы, составные части которых функционируют на различных компьютерах в сети. Распределенное приложение выполняется как совокупность нескольких процессов, совместно решающих определенную задачу.

Распределенные системы позволяют обеспечить выполнение следующих требований:

Во-первых, обеспечение масштабируемости систем, т.е. способности эффективно обслуживать как малое, так и очень большое количество клиентов одновременно;

Во-вторых, надежность создаваемых приложений. Программный комплекс должен быть устойчив не только к ошибкам пользователей, но и к сбоям в системе коммуникаций;

В-третьих, возможность непрерывной работы в режиме круглосуточной работы в течение недель и месяцев;

В-четвертых, высокий уровень безопасности системы. Под безопасностью понимается не только контроль доступности тех или иных ресурсов системы и защищенность информации, но и отслеживание выполняемых действий с высокой степенью достоверности;

В-пятых, высокая скорость разработки приложений и простота их сопровождения и модификации.

Большая часть реальных распределенных приложений имеют один или более критический компонент, который участвует в большинстве операций. Это могут быть, например, компоненты базы данных, доступ к которым должен осуществляться постоянно для реализации политики «первым пришел – первым обслужен». Компоненты такого типа не могут дублироваться, поскольку их предназначение заключается в организации единственной точки синхронизации среди всех пользователей приложения. Для увеличения быстродействия распределенного приложения в целом такие компоненты, создающие «узкие места», должны перераспределяться на специально выделенный мощный сервер. DCOM помогает выявить такие критические компоненты на ранних этапах проектирования. Технология DCOM позволяет первоначально

разместить все компоненты на одной машине, а затем перенести критические компоненты на отдельные машины.

Технология программирования DCOM облегчает схемы перераспределения по мере разрастания приложения. Первоначально машина сервера может содержать все компоненты. По мере увеличения потребностей можно добавить другие машины с перераспределением компонентов на эти машины без всякого изменения кода. Таким образом, технология DCOM характеризует такое свойство COM технологий, как масштабируемость (SCALABILITY).

При реализации COM технологии, в зависимости от местоположения клиента и сервера возможны три варианта:

Во-первых, клиент и сервер располагаются на одной машине и запускаются в одном процессе (именно так взаимодействует программа DELPHI с компонентами ActiveX). В этом случае, сервер представляет собой DLL (DYNAMIC LINK LIBRARY);

Во-вторых, клиент и сервер располагаются на одной машине, но запускаются в разных процессах (например, таблицы Excel вставлены в документ Word). В этом случае, сервер представляет собой программу;

В-третьих, клиент и сервер располагаются на разных машинах. Сервером может быть как программа, так и DLL. В данном случае используется распределенный вариант COM, т.е. DCOM

Использование технологии DCOM означает, что COM серверы существуют и выполняются на различных машинах в сети. В связи с тем, что при использовании технологии DCOM, сервер запускается на другой машине, между объектом и клиентом располагаются два посредника:

Во-первых, Proxy (уполномоченный);

Во-вторых, Stub (заглушка).

Клиент помещает параметры вызова в стек и обращается к методу интерфейса объекта. Однако, это обращение перехватывает Proxy, упаковывает параметры вызова в пакет COM и направляет его в Stub.

Чтобы использовать приложения с моделью DCOM, необходимо в программе настройки DCOM задать свойства приложений, определяющие их размещение и параметры безопасности. На компьютере, выполняющем приложение-клиента необходимо указать расположение приложения-сервера, из которого оно будет запускаться или вызываться. Для приложения-сервера необходимо указать учетную запись пользователя, имеющего разрешение на доступ к приложению или его запуск, и учетные записи пользователей, используемые для выполнения приложений.

При переходе от технологии COM к технологии DCOM добавляются три основных элементы:

Во-первых, способ создания удаленного объекта;

Во-вторых, протокол вызова методов этого объекта;

В-третьих, механизмы обеспечения безопасного доступа к объекту.

Для создания удаленного объекта клиент может воспользоваться моникером (Moniker). Моникер представляет собой объект, который знает, как создавать и инициализировать один экземпляр другого объекта. Моникеры могут выполнять создание экземпляров и инициализацию объектов как на локальных, так и на удаленных машинах.

При создании объекта на той же машине, что и клиент, библиотека COM этой машины предоставляет выполнение основной работы по запуску соответствующего сервера диспетчеру управления сервисами (Service Control Manager – SCM). Когда же объект создается на удаленной машине, SCM клиентской машины должен в свою очередь делегировать выполнение этой задачи SCM удаленной машины.

При реализации DCOM используются протоколы сети INTERNET, такие как HTTP (HYPER TEXT TRANSFER PROTOCOL).

В технологии DCOM реализован контроль за тем, кто имеет право создавать объекты на данной машине и обеспечивается безопасный доступ к этим объектам по сети. С этой целью в основу DCOM положен набор сервисов контроля доступа.

5.7. Особенности использования DCOM технологии при программировании в среде Delphi

Системные компоненты, предназначенные для реализации DCOM расположены на странице MIDAS (MULTI-TIER DISTRIBUTED APPLICATION SERVICES SUITE). Можно выделить следующие компоненты страницы MIDAS:

Во-первых, компонент DCOMCONNECTION, который предназначен для установления связи с удаленным сервером, используя средства технологии DCOM.

Во-вторых, компонент CLIENTDATASET, который предназначен для непосредственного доступа к данным на клиентском жестком диске. Компонент может быть использован как для одноярусных приложений, так и для выборочного отбора данных в многоярусных приложениях.

В-третьих, компонент DATASETPROVIDER, который обеспечивает работу с наборами данных при реализации многоярусных приложений.

5.8. Технология COM+

Технология COM+ появилась для унификации и объединения технологий COM, DCOM и MTS (MICROSOFT TRANSACTION SERVER) в согласованную технологию, понятную и удобную для реализации приложений корпоративного уровня. COM+ приложение является основной единицей для защиты и администрирования компонент. COM+ приложение представляет собой множество COM компонент, которые выполняют согласованные функции. Создание логических групп позволяет получить следующие преимущества COM+:

Во-первых, определение пространства развертывания COM компонент;

Во-вторых, определение конфигурирования COM компонент, включая службу безопасности. В COM+ используется

служба безопасности, основанная на концепции ролей (ROLE-BASED SECURITY), применяемая в технологии MTS;

В-третьих, балансировку загрузки компонент;

В-четвертых, диспетчеризацию очередей сообщений компонент;

В-пятых, загрузку компонентных DLL по усмотрения разработчика;

В-шестых, управление выполнением компонент.

Разработка COM+ приложений включает следующие этапы:

Во-первых, определение COM классов и их воплощение;

Во-вторых, группировка классов в компоненты;

В-третьих, определение множества COM+ сервисов, необходимых для компоненты;

В-четвертых, разработка COM+ приложения;

В-пятых, интеграция компонент в COM+ приложение, уже существующее или новое;

В-шестых, определение правильного множества атрибутов для каждого класса. Данные атрибуты представляют зависимости компонент от используемых сервисов;

В-седьмых, установка системы защиты. При установке системы защиты определяются права и роли классов, методов и интерфейсов;

В-восьмых, Конфигурация переменных окружения для классов и приложений;

В-девятых, Подготовка приложения для распространения и развертывания;

В-десятых, Администрирование COM+ приложений;

В-одиннадцатых, Инсталляция приложения на машине администратора;

В-двенадцатых, Установка переменных окружения;

В-тринадцатых. Создание брокеров (PROXY) приложения;

В-четырнадцатых, Тестирование приложения.

5.9. Технология CORBA

Технология CORBA (Common Object Request Broker Architecture) разрабатывается отраслевым комитетом OMG (Object Management Group- группа управления объектами). Данная группа представляет собой объединение многих фирм и пользователей. CORBA создавалась как универсальная технология создания сложных и надежных распределенных систем, т.е. систем с сотнями и тысячами серверами и миллионами клиентов. Технология CORBA обеспечивает реальную масштабируемых создаваемых приложений. Поэтому, данная технология в принципе может быть использована при создании систем любой сложности. Однако, при использовании простых и недорогих систем использование технологии CORBA экономически нецелесообразно. Технология CORBA широко применяется при работе в таких операционных системах как UNIX и LINUX.

Важным отличием CORBA от COM является интегрированный в технологии CORBA, реализующий доступ к удаленным объектам.

В соответствии с этой технологией схема взаимодействия клиента и сервера выглядит следующим образом:

На машине клиента создаются два объекта посредника:

Во-первых, Stub (заглушка);

Во-вторых, ORB (OBJECT REQUIRE BROKER – брокер требуемого объекта).

Stub выступает как полномочный представитель объекта. С помощью интерфейса объекта клиент обращается к Stub так, как если бы это был сам объект.

Получив вызов метода, Stub транслирует этот вызов объекту ORB, который посылает в сеть сообщение. На это сообщение откликается один из объектов Smart Agent (умный агент), установленный в сетевом окружении клиента. Smart Agent моделирует сетевой каталог, в котором зарегистрированы известные ему серверы объектов. Он отыскивает нужный сетевой адрес сервера и передает запрос объекту ORB на машине сервера. Обмен данными между ORB (клиента и сервера) и Smart Agent осуществляется с помощью специального протокола UDP (US-

ER DATAGRAM PROTOCOL), который более бережно использует сетевые ресурсы, чем протокол TCP (TRANSPORT CONTROL PROTOCOL, TRANSFER CONTROL PROTOCOL). Через BOA (Basic Object Adapter – базовый объектный адаптер) данные получает особый объект сервера, который называется Skeleton (скелет). Skeleton помещает параметры вызова в стек адресного пространства объекта и реализует собственно вызов.

Роль объекта BOA заключается в фильтрации обращений к объекту сервера. С помощью его методов сервер через Skeleton может объявить некоторые свои поля и свойства доступными только для чтения или вовсе скрытыми для какого-либо клиента.

Важнейшим элементом CORBA является способ описания спецификации проекта и его составных частей. Для этих целей разработан специальный язык IDL (INTERFACE DEFINITION LANGUAGE –язык описания интерфейса). IDL позволяет определить абсолютно все аспекты поведения объектов, из которых состоит проект. Однако, никакие детали реализации этих объектов на этапе создания IDL-деклараций во внимание не принимаются. Описание интерфейса не зависит от используемого языка программирования, операционной системы или архитектуры процессора.

IDL является достаточно выразительным языком. Он позволяет определять типы данных, которые необходимы для функционирования системы, включая структуры и массивы. Для методов, входящих в удаленные объекты IDL позволяет задать списки аргументов, тип возвращаемого результата, а также исключительные ситуации, которые могут быть возбуждены при вызове этого метода.

После описания интерфейса в терминах этого языка компилятор IDL автоматически создает объекты Stub и Skeleton.

Другим важным элементом технологии CORBA являются шаблоны программных конструкций (DESIGN PATTERNS). Наиболее распространенным шаблоном является «фабрика» (FACTORY).

Для реализации технологии в сетевом окружении клиента должен существовать хотя бы один Smart Agent. Если обмен

данными осуществляется в локальной сети, то Smart Agent устанавливается на головную машину, т.е. на файл-сервер или машину с SQL-сервером. При использовании технологии INTERNET/INTRANET Smart Agent устанавливается на один из узлов сети. При создании сервера осуществляется автоматическая регистрация в одном или нескольких Smart Agent.

Используется технология CORBA, в основном, при программировании на таких языках как JAVA, C++, PERL. Кроме перечисленных могут использоваться другие системы, например DELPHI.

CORBA является основой для других технологий, как EJB (Enterprise JavaBeans (EJB).

5.10. Особенности использования CORBA технологии при программировании в среде Delphi

Можно выделить следующие системные компоненты, предназначенные для работы с технологией CORBA:

Во-первых, компонент TCORBAFACTORY, который является базовым классом для удаленных объектов клиентских приложений, использующих технологию CORBA.

Во-вторых, компонент TCORBASTUB, который является базовым классом для всех STUB-объектов технологии CORBA.

В-третьих, компонент TCORBASKELETON, который является базовым классом для всех SKELETON-объектов технологии CORBA.

В-четвертых, компонент TCORBACOMOBJECTFACTORY, который предназначен для создания COM объекта в соответствии с требованиями CORBA клиента. Используя созданный объект сервер может реагировать на требования CORBA клиента, также как и на COM клиента.

В-пятых, компонент ECORBADISPATCH, который является классом, предназначенным для обработки исключительных ситуаций, возникающих при работе с системными компонентами, входящими в DLL библиотеки.

5.11. Управляющие элементы ActiveX

Управляющие элементы ActiveX представляют собой 32 разрядные объекты, содержащие коды и данные. Данные объекты могут создаваться с помощью различных средств разработки, например Visual C++ или Visual Basic. Основным преимуществом ActiveX компонентов является их огромное количество, т.к. их разработкой занимаются фирмы и отдельные программисты.

Первоначально управляющие элементы ActiveX были известны под названием управляющие элементы OLE или OCX. Корпорация Microsoft изменила название, чтобы отразить некоторые новые возможности, сделавшие эти элементы более подходящими для Internet и WWW. Например, управляющий элемент ActiveX может хранить свои данные на странице WWW, либо может быть выкачен с сервера WWW и затем запущен на машине клиента. Контейнер, в котором работает управляющий элемент, не обязательно является средой программирования, а в частности он может быть средством просмотра WWW.

Спецификации управляющих элементов ActiveX определяют стандарты для построения компонентов сложной структуры. Управляющие элементы ActiveX могут быть весьма сложны. Поэтому, спецификация также определяет правила создания контейнеров управляющих элементов (control container). Обычно контейнеры управляющих элементов позволяют программисту задать действие (в виде кода функции или метода), которое должно быть выполнено в ответ на сообщение, полученное от управляющего элемента.

Функциональность, определяемая спецификацией управляющих элементов ActiveX, распадается на четыре основных аспекта. Для реализации каждого аспекта предназначена особая группа интерфейсов:

Во-первых, обеспечение пользовательского интерфейса;

Во-вторых, обеспечение вызова методов управляющего элемента контейнером;

В-третьих, посылка событий контейнеру;

В-четвертых, получение информации о свойствах среды контейнера и обеспечение доступа к свойствам управляющего элемента и их модификации.

Компоненты ActiveX являются «чужими» для DELPHI, т.к. они создаются другими инструментальными средствами разработки программ. Подключение ActiveX компонент обеспечивается благодаря использованию COM технологии.

При работе в DELPHI, могут быть использованы, в частности, следующие ActiveX компоненты:

Во-первых, компонент CHARTFX, который предназначен для организации интерактивной графики. Данный компонент предоставляет программисту удобное средство включения в программу интерактивных графиков.

Во-вторых, компонент VSSPELL, который называется спеллер (good speller переводится с английского языка, как «грамотно пишущий человек»). Данный компонент осуществляет орфографическую проверку правильности написания английских слов.

В-третьих, компонент FLBOOK, который позволяет создавать и использовать рабочие книги электронных таблиц.

В-четвертых, компонент VTCHART, который обеспечивает мощные средства построения двухмерных и трехмерных диаграмм по результатам табличных вычислений.

В-пятых, компонент GRAPH, который предназначен для включения в программу графических двухмерных средств отображения данных.

Тема 6.

Case-технологии

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС), создаваемых в различных областях экономики. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

- сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;
- наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);
- отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
- необходимость интеграции существующих и вновь разрабатываемых приложений;
- функционирование в неоднородной среде на нескольких аппаратных платформах;
- разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
- существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков, и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

В 70-х и 80-х годах при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако, широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении

серьезных изменений практически неосуществима. Ручная разработка обычно порождала следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

С другой стороны, разработчики ИС исторически всегда стояли последними в ряду тех, кто использовал компьютерные технологии для повышения качества, надежности и производительности в своей собственной работе (феномен "сапожника без сапог").

Перечисленные факторы способствовали появлению программно-технологических средств специального класса - CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с

разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т.д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

- подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
- широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;
- внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Согласно обзору передовых технологий (Survey of Advanced Technology), составленному фирмой Systems Development Inc. в 1996 г. по результатам анкетирования более 1000 американских фирм, CASE-технология в настоящее время попала в разряд наиболее стабильных информационных технологий (ее использовала половина всех опрошенных

пользователей более чем в трети своих проектов, из них 85% завершились успешно). Однако, несмотря на все потенциальные возможности CASE-средств, существует множество примеров их неудачного внедрения, в результате которых CASE-средства становятся «полочным» ПО (shelfware). В связи с этим необходимо отметить следующее:

- CASE-средства не обязательно дают немедленный эффект; он может быть получен только спустя какое-то время;
- реальные затраты на внедрение CASE-средств обычно намного превышают затраты на их приобретение;
- CASE-средства обеспечивают возможности для получения существенной выгоды только после успешного завершения процесса их внедрения.

Ввиду разнообразной природы CASE-средств было бы ошибочно делать какие-либо безоговорочные утверждения относительно реального удовлетворения тех или иных ожиданий от их внедрения. Можно перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

- широкое разнообразие качества и возможностей CASE-средств;
- относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
- широкое разнообразие в практике внедрения различных организаций;
- отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
- широкий диапазон предметных областей проектов;
- различная степень интеграции CASE-средств в различных проектах.

Вследствие этих сложностей доступная информация о реальных внедрениях крайне ограничена и противоречива. Она зависит от типа средств, характеристик проектов, уровня

сопровождения и опыта пользователей. Некоторые аналитики полагают, что реальная выгода от использования некоторых типов CASE-средств может быть получена только после одно- или двухлетнего опыта. Другие полагают, что воздействие может реально проявиться в фазе эксплуатации жизненного цикла ИС, когда технологические улучшения могут привести к снижению эксплуатационных затрат.

Для успешного внедрения CASE-средств организация должна обладать следующими качествами:

- Технология. Понимание ограниченности существующих возможностей и способность принять новую технологию;
- Культура. Готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями;
- Управление. Четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

Если организация не обладает хотя бы одним из перечисленных качеств, то внедрение CASE-средств может закончиться неудачей независимо от степени тщательности следования различным рекомендациям по внедрению.

Для того, чтобы принять взвешенное решение относительно инвестиций в CASE-технологию, пользователи вынуждены производить оценку отдельных CASE-средств, опираясь на неполные и противоречивые данные. Эта проблема зачастую усугубляется недостаточным знанием всех возможных "подводных камней" использования CASE-средств. Среди наиболее важных проблем выделяются следующие:

- достоверная оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки ПО;
- внедрение CASE-средств может представлять собой достаточно длительный процесс и может не принести немедленной отдачи. Возможно даже краткосрочное снижение продуктивности в результате усилий, затрачиваемых на внедрение. Вследствие этого руководство организации-пользователя мо-

жет утратить интерес к CASE-средствам и прекратить поддержку их внедрения;

- отсутствие полного соответствия между теми процессами и методами, которые поддерживаются CASE-средствами, и теми, которые используются в данной организации, может привести к дополнительным трудностям;
- CASE-средства зачастую трудно использовать в комплексе с другими подобными средствами. Это объясняется как различными парадигмами, поддерживаемыми различными средствами, так и проблемами передачи данных и управления от одного средства к другому;
- некоторые CASE-средства требуют слишком много усилий для того, чтобы оправдать их использование в небольшом проекте, при этом, тем не менее, можно извлечь выгоду из той дисциплины, к которой обязывает их применение;
- негативное отношение персонала к внедрению новой CASE-технологии может быть главной причиной провала проекта.

Пользователи CASE-средств должны быть готовы к необходимости долгосрочных затрат на эксплуатацию, частому появлению новых версий и возможному быстрому моральному старению средств, а также постоянным затратам на обучение и повышение квалификации персонала.

Несмотря на все высказанные предостережения и некоторый пессимизм, грамотный и разумный подход к использованию CASE-средств может преодолеть все перечисленные трудности. Успешное внедрение CASE-средств должно обеспечить такие выгоды как:

- высокий уровень технологической поддержки процессов разработки и сопровождения ПО;
- положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;
- приемлемый уровень отдачи от инвестиций в CASE-средства.

Тема 7.

Программное обеспечение и его классификация

Под программным обеспечением понимается совокупность программ и документации на них, предназначенных для реализации целей и задач. Программное обеспечение в соответствии с выполняемыми функциями делится на системное (общее) и прикладное (специальное) программное обеспечение.

К системному программному обеспечению относится совокупность программ описаний и инструкций, используемых для эффективного функционирования вычислительной системы, а также при разработке новых программ.

По функциональному назначению в системном программном обеспечении выделяют следующие элементы:

Во-первых, операционную систему;

Во-вторых, систему программирования;

В-третьих, средства контроля и диагностики.

Под операционной системой понимается комплекс управляющих программ, обеспечивающих функционирование вычислительной машины.

Под системой программирования понимают комплекс средств для разработки и отладки программ.

Под средствами контроля и диагностики понимается совокупность программ, которые служат для выявления и локализации неисправностей.

Прикладное программное обеспечение предназначено для решения конкретных задач из различных сфер применения.

Тема 8.

Свойства системного программного обеспечения

Можно выделить следующие свойства системного программного обеспечения:

Во-первых, совместимость (COMPATIBLE) программного обеспечения. Совместимость предполагает поддержку различных приложений, разработанных для ранних версий системного программного обеспечения.

Во-вторых, переносимость (PORTABILITY) программного обеспечения. Переносимость означает возможность работы системного программного обеспечения при использовании различных процессоров.

В-третьих, масштабируемость (SCALABILITY), которая означает, что при работе в корпоративной сети имеется возможность добавлять более производительные серверы и рабочие станции.

В-четвертых, безопасность (SECURITY), которая означает, что при использовании корпоративной сети заданным приложениям обеспечивается полностью изолированное окружение.

В-пятых, распределенная обработка (DISTRIBUTED PROCESSING) , которая предполагает обеспечение связи с различными типами ХОСТ-компьютеров благодаря поддержке разнообразных протоколов.

В-шестых, надежность (RELIABILITY) и отказоустойчивость (RUBUSTNESS), которые предполагают защиту программ от повреждения друг другом и операционной системой.

8.1. Организация системного программного обеспечения в виде компонент

Развитие технологии объектно-ориентированного программирования привело к возникновению понятия «компо-

нент». Под компонентом (COMPONENT) понимается некий функциональный элемент, содержащий определенный свойства. Таким образом, понятие "компонент", является в достаточной степени абстрактным и может в определенной степени варьироваться в различных системах.

Использование компонент позволяет перейти к технологии разработки приложений на основе компонентной архитектуры. В этом случае, компонент представляет собой особый вид приложения, т.к. он поставляется как двоичный код, скомпилированный и готовый к использованию. При использовании компонентной архитектуры приложение представляет собой совокупность компонент. Отдельные компоненты подключаются, во время выполнения, к другим компонентам, формируя приложение. Приложение не является статичным. Модификация или расширение приложения сводится к замене одного из составляющих компонентов новой версией.

Компонентная архитектура хорошо приспособлена для создания приложений легко адаптируемых к конкретным требованиям пользователя. Готовые продукты, созданные на основе компонентной архитектуры, позволяют при необходимости заменить любой компонент другим, более соответствующим конкретной ситуации.

Используя компонентную архитектуру можно упростить процесс разработки распределенных приложений. В этом случае, приложение состоит из компонент, разбросанных по разным машинам сети. При разработке распределенных приложений могут быть использованы специальные компоненты-переадресовщики, которые вместо выполнения заданной обработки, обеспечивают передачу запроса к требуемой компоненте.

На основе компонентной архитектуры разрабатываются многие приложения, работающие в сети INTERNET/INTRANET.

Для реализации компонентной архитектуры предназначены такие средства как COM(Component Object Model, модель компонентных объектов).

Концепция компонент реализована практически во всех современных языках программирования.

Тема 9.

Интегрированная среда разработки

9.1. Назначение интегрированной среды разработки

Интегрированная среда разработки IDE (INTEGRATED DEVELOPMENT ENVIROMENT) представляет собой совокупность всех инструментов, необходимых для разработки, отладки и выполнения объектно-ориентированных приложений.

Использование интегрированной среды разработки обеспечивает следующие преимущества:

Во-первых, повышение производительности труда разработчика прикладного программного обеспечения;

Во-вторых, снижение стоимостных затрат на разработку программного обеспечения;

В-третьих, сокращение сроков разработки готового приложения;

В-четвертых, обеспечение обработки информации с использованием сети INTERNET.

Среда разработки называется интегрированной потому, что с экрана можно вызвать любой необходимый инструмент. Можно выделить следующие элементы интегрированной среды разработки:

Во-первых, главное меню;

Во-вторых, конструктор форм;

В-третьих, панель элементов;

В-четвертых, окно свойств;

В-пятых, окно программы.

Главным меню называется строка текста, расположенная в верхней части окна. Как правило, главное меню включает пункты FILE, EDIT, VIEW, RUN и другие. В процессе работы могут появляться новые пункты главного меню.

Окно конструктора форм располагается в центре экрана. В данном окне происходит визуальное конструирование макета формы и расположенных на ней элементов.

Панель элементов содержит компоненты, из которых складывается интерфейс прикладного приложения. Все кнопки изображенные на панели элементов соответствуют объектам, которые размещаются необходимо разместить на экранной форме. Для перемещения элемента на экранную форму требуется выполнить следующие действия: во-первых, поместить курсор мыши на требуемый элемент; во-вторых, двойным щелчком активизировать выбранный элемент; в-третьих, расположить элемент на нужном месте в экранной форме.

Окно свойств, предназначено для отображения и модификации свойств (property) выделенных объектов.

Окно программы предназначено для создания процедур (методов, method), обеспечивающих требуемую обработку при возникновении определенного события (Event).

Основные принципы интегрированной среды разработки реализованы во всех современных системах программирования, таких как Visual Basic, Delphi, Visual FoxPro . Однако, в каждой из перечисленных систем имеются свои особенности реализации.

9.2. Особенности реализации интегрированной среды разработки в Visual Basic

Рассмотрим интегрированную среду разработки системы Visual Basic. Интегрированная среда разработки Visual Basic характеризуется большим разнообразием разрабатываемых проектов. В частности, при создании нового проекта допускаются следующие варианты:

Во-первых, Standard EXE (стандартный EXE-файл);

Во-вторых, ActiveX EXE (EXE-файл ActiveX);

В-третьих, ActiveX DLL (DYNAMIC LINK LIBRARY библиотека ActiveX);

В-четвертых, ActiveX Control (управляющий элемент ActiveX);

В-пятых, VB Application Wizard (мастер приложений Visual Basic);

В-шестых, Data Project (проект базы данных);

В-седьмых, IIS Application (приложение Internet Information Server);

В-восьмых, Add-in (надстройка);

В-девятых, ActiveX Document DLL (DLL-библиотека документа ActiveX);

В-десятых, ActiveX Document EXE (EXE-файл документа ActiveX);

В-одинадцатых, DHTML Application (приложение DYNAMIC HYPERTEXT MARKUP LANGUAGE).

9.3. Создание прикладного программного приложения

Рассмотрим процесс создания приложения на примере ввода двух чисел и нахождения их среднего арифметического.

Прикладное приложение целесообразно организовать в виде стандартного файла .EXE. поэтому в PROJECT WIZARD выбирается вкладка NEW и определяется шаблон STANDARD EXE.

Формирование требуемого приложения можно выполнять в следующей последовательности:

Во-первых, для формируемой экранной формы в окне свойств определяются следующие значения: для свойства Name задается значение “frmAvar”, для свойства Caption задается значение «Расчет среднего арифметического».

Во-вторых, формируется текстовое поле для ввода первого вводимого значения. Для этого, двойным щелчком активизируется элемент «текстовое поле (TEXT BOX)» и определяется местоположение данного элемента на экранной форме. В окне свойств определяются следующие значения: для свойства Name задается значение “txtfirst”, для свойства Text задается значение пустой строки.

В-третьих, формируется текстовое поле для ввода второго вводимого значения. В окне свойств определяются следующие значения: для свойства Name задается значение “txtsecond”, для свойства Text задается значение пустой строки.

В-четвертых, формируется текстовое поле для полученного результата. В окне свойств определяются следующие значения: для свойства Name задается значение “txtresult”, для свойства Text задается значение пустой строки.

В-пятых, формируется пояснительная надпись для первого вводимого значения. Для этого, активизируется элемент-надпись (LABEL) и определяется местоположение данного элемента на экранной форме. В окне свойств определяются следующие значения: для свойства Name задается значение “lblfirst”, для свойства Caption задается значение «Первое вводимое число».

В-шестых, формируется пояснительная надпись для второго вводимого значения. В окне свойств определяются следующие значения: для свойства Name задается значение “lblsecond”, для свойства Caption задается значение «Второе вводимое число».

В-седьмых, формируется пояснительная надпись для полученного результата. В окне свойств определяются следующие значения: для свойства Name задается значение “lblresult”, для свойства Caption задается значение «Среднее арифметическое».

В-восьмых, формируется командная кнопка для выполнения расчета. Для этого, активизируется элемент-кнопка (Command Button) и определяется местоположение данного элемента на экранной форме. В окне свойств определяются следующие значения: для свойства Name задается значение “cmdOk”, для свойства Caption задается значение «Расчет».

В-девятых, создается процедура (метод), обеспечивающая необходимую обработку. Для этого производится двойной щелчок на кнопке cmdOK. Двойной щелчок на элементе открывает окно программы со стандартным событием. Для командной кнопки стандартным является событие CLICK. На экране дол-

жен появиться шаблон процедуры cmdOK_Click. В шаблон вставляется следующий набор команд

```
      Dim A
      A =
(Val(txtfirst.text)+Val(txtsecond.text))/2
      TxtResult.text =str(A)
```

При необходимости, вызвать необходимый шаблон для создаваемого метода можно выбрав в меню VIEW пункт CODE.

В-девятых, формируется командная кнопка для завершения работы. В окне свойств определяются следующие значения: для свойства Name задается значение “cmdExit”, для свойства Caption задается значение «Выход».

В-девятых, создается процедура(метод), обеспечивающая необходимую обработку. Для этого производится двойной щелчок на кнопке cmdExit. На экране должен появиться шаблон процедуры cmdExit_Click. В шаблон вставляется команда

```
Unload Me
```

Данная команда обеспечивает завершение работы приложения.

Созданный проект необходимо сохранить под требуемым именем. Для этого в меню FILE выбирается пункт SAVE PROJECT AS.

Выполнение проекта обеспечивается выбором в меню RUN пункта START WITH FULL COMPILE или пункта START.

Для представления созданного проекта в виде файла с расширением ‘.EXE’, необходимо в меню FILE выбрать пункт ‘MAKE PROJECT.EXE’.

Тема 10.

Системные компоненты общего назначения

10.1. Краткая характеристика компонентов

Под компонентом (component) понимается некий функциональный элемент, содержащий определенные свойства. Как правило, компоненты представляют собой элементы, из которых конструируется видимое изображение, создаваемое работающей программой. Однако существует значительное количество компонентов, которые не создают видимого изображения.

Во многих системах, особо выделяются компоненты, обеспечивающие создание экранных форм. Экранная форма образует визуальную основу прикладного приложения. По своей сути экранная форма представляет собой окно, в котором размещаются управляющие элементы в процессе создания приложения.

Отдельные компоненты могут входить в другие компоненты. Компоненты, которые могут включать в себя другие компоненты называют контейнерами (CONTAINER). Каждый компонент, входящий в контейнер может рассматриваться отдельно. Такая зависимость компонентов друг от друга позволяет существенно упростить процесс управления ими. Например, для того, чтобы уничтожить окно с включенными в него компонентами, достаточно вызвать деструктор окна. Деструктор окна поочередно вызовет деструкторы всех других компонентов, владельцем которого является компонент-окно, и таким образом полностью освободит выделенные окну ресурсы.

10.2. Свойства, задаваемые в компонентах

Визуальные объекты, имеют значительное число общих для всех объектов свойств. Можно выделить следующие свойства:

Свойство NAME, которое определяет имя компонента. Как правило компонент автоматически получает создаваемое средой имя. Имя, задаваемое по умолчанию, совпадает с именем порождающего класса дополненным числовым суффиксом. Например, Form1. В дальнейшем программист, обычно, присваивает компоненте новое имя. При разработке собственных имен целесообразно использовать кодировку имени с помощью префиксов. Например: Label_rez или LblRez для компоненты класса метка, используемой при выдаче результата.

Свойство CAPTION, которое определяет текстовое описание, связанное с компонентой. Текстовое описание может располагаться в различных местах. У экранной формы текстовое описание представляет собой заголовок формы. У командных кнопок текстовое описание располагается внутри кнопки. У кнопок, обеспечивающих выбор режима обработки, текстовое описание располагается справа от них.

Свойство TEXT, которое содержит текстовую строку, связанную с управляющим элементом. Как правило, данное свойство используется для ввода информации.

Свойство VISIBLE, которое позволяет сделать компонент видимым. Изменить свойство возможно задав логическое значение True (истинно) или False (ложно).

Свойство TOP, которое определяет расстояние от верхнего края контейнера до верхней границы рассматриваемой компоненты.

Свойство LEFT, которое определяет расстояние от левого края объекта-контейнера.

Свойство HEIGHT, которое определяет высоту объекта.

Свойство WIDTH, которое определяет ширину объекта.

Свойство AUTOSIZE, которое позволяет автоматически настраивать размер формируемой компоненты в зависимости от величины заданного пояснительного текста. Для установки свойства требуется задать логическое значение True.

Свойство FONT, определяющее размер, имя шрифта и другие особенности выдаваемого текста.

Названия свойств в различных системах в значительной степени совпадают. Например, такие свойства как NAME,

CAPTION, TOP, LEFT и другие используются в компонентах Visual Basic, Delphi, Visual FoxPro.

10.3. Методы, используемые в компонентах

Формируемые компоненты могут использовать в частности следующие методы:

Метод SHOW, который используется для того, чтобы сделать видимой ранее не видимую экранную форму.

Метод REFRESH, который используется для того, чтобы обновить изображение экранной формы.

Метод SETFOCUS, который используется для установки фокуса на заданном визуальном элементе.

Метод ADDITEM, который позволяет добавлять информацию в такие элементы, как List Box, Combo Box.

Каждый компонент предусматривает использование значительного числа методов. Причем состав используемых методов в значительной степени различается в зависимости от вида компоненты.

10.4. События, используемые в компонентах

Выполнение действий над объектом обеспечивается при возникновении событий. Можно выделить следующие события:

во-первых, событие CLICK, которое происходит тогда, когда курсор находится на объекте и пользователь щелкает левой клавишей мыши. Все видимые объекты способны обрабатывать событие CLICK, в том числе картинки;

во-вторых, событие ACTIVATE, которое происходит тогда, когда экранная форма становится активной, т.е. на нее переносится фокус.

Количество событий, которое может быть использовано для управления различными компонентами достаточно большое. Список событий, на которые реагирует конкретная компонента всегда можно уточнить по технической документации.

10.5. Компоненты, используемые в задачах обработки экономической информации

Экономическая информация, как правило, организуется в виде базы данных на внешнем носителе. Поэтому, все современные системы программирования содержат системные компоненты для работы с базами данных.

При работе с Visual Basic обычно используются таблицы Microsoft Access. Помимо Access могут быть использованы таблицы других баз данных.

Для работы с базами данных ACCESS, SQL, ORACLE и другими используется универсальная технология доступа к данным ACO (ActiveX Data Objects).

Однако, использование базы данных предполагает соблюдение следующих условий:

Во-первых, необходимо приобрести выбранную базу данных;

Во-вторых, требуется предварительно подготовить обрабатываемые таблицы;

В-третьих, необходимо изучить технологию ADO и освоить системное программное обеспечение, связанное с данной технологией.

Обработка экономической информации в оперативной памяти, как правило, обеспечивается стандартным набором системных компонент. Наиболее часто при создании объектов, обеспечивающих обработку экономической информации используются следующие компоненты:

Во-первых, компонент Form, который предназначен для создания экранных форм. Любая программа имеет как минимум одну связанную с ней форму, которая появляется на экране в момент старта программы. Данная экранная форма называется главной. Другие, экранные формы появляются по мере надобности

Во-вторых, компонент Command Button, который предназначен для создания командных кнопок. Командные кнопки обеспечивают управление работой программы.

В-третьих, компонент Label, который обеспечивает размещение в экранной форме текстовых надписей.

В-четвертых, компонент Text Box, который предназначен для создания однострочного редактора текста. Формируемый редактор текста позволяет вводит и редактировать текстовые строки.

В-пятых, компонент Check Box, который обеспечивает создание независимого переключателя (индикаторной кнопки). Элемент Check Box имеет два состояния "Да" (TRUE) и "Нет" (FALSE). Экранная форма может содержать несколько переключателей. Состояние любого из них не зависит от состояния остальных.

В-шестых, компонент List Box, который обеспечивает представление просматриваемой информации в виде списка.

При использовании Visual Basic обработка экономической информации в оперативной памяти имеет ряд особенностей.

Во-первых, обработка экономической информации, как правило, предусматривает использование специальной программной конструкции, которая называется программным модулем. Программный модуль представляет собой текстовый файл, содержащий подпрограммы, функции, переменные и константы.

Во-вторых, описание обрабатываемой экономической информации осуществляется на модульном уровне (module level). Для описания структурных взаимосвязей различных реквизитов предназначено утверждение TYPE, которое относится к типам данных, определенных пользователем (user defined type). Конструкция TYPE записывается в соответствии со следующим синтаксисом:

```
[PRIVATE | PUBLIC] TYPE <имя структуры>
 <имя элемента> AS <тип>
 [<имя элемента> AS <тип>]
END TYPE
```

В простейшем случае, задание конструкции TYPE осуществляется на участке (General) (Declaration) программного кода (CODE). Конструкция TYPE в этом случае задается с описателем PRIVATE.

Пример: Имеется документ «Классификатор ценник», содержащий следующие поля:

код предмета – 6 цифр;

наименование предмета – 20 символов;

цена – 7 цифр, из них две определяют дробную часть.

Разработать в среде Visual Basic программу определения статистических частот для реквизита «ЦЕНА».

Для решения задачи формируется экранная форма. При создании экранной форме задаются свойства NAME и CAPTION.

В экранную форму включаются следующие элементы:

Во-первых, текстовые поля (Text Box), предназначенные для ввода информации. Для ввода каждого реквизита формируется отдельный компонент. При создании текстового элемента задаются свойства NAME и TEXT.

Во-вторых, надписи (Label), предназначенные для задания пояснительного текста к вводимым реквизитам. При создании надписи задаются свойства NAME и CAPTION.

В-третьих, список (List Box), предназначенный для выдачи полученных значений статистических частот. Для формируемого элемента свойство Name устанавливается равное значению ‘LstRez’.

В-четвертых, надпись, предназначенная для задания пояснительного текста при выдачи результатного значения. Для формируемого элемента свойство Name устанавливается равное значению ‘LblRez’.

В-пятых, командная кнопка (Command Button), предназначенная для задания команды на ввод информации. Для командной кнопки устанавливается свойство Name равное значению ‘CmdInput’. Для события CLICK задается метод, обеспечивающий ввод очередной строки документа.

В-шестых, командная кнопка, предназначенная для задания команды на выполнение расчета статистических частот. Для командной кнопки устанавливается свойство Name равное значению ‘CmdProcess’. Для события CLICK задается метод, обеспечивающий расчет статистических частот и выдачу полученных значений.

В-седьмых, командная кнопка, предназначенная для задания команды на прекращение работы. Для командной кнопки устанавливается свойство Name равное значению ‘CmdCancel’. Для события CLICK задается метод, обеспечивающий закрытие активного окна проекта.

Текст модуля, включающих общее описание и совокупность методов имеет следующий вид:

```
'Раздел общих описаний
' GENERAL DECLARATIONS
'Заданные в данной разделе
'переменные и масивы
'действительны во всех методах
Private Type TClassif
 Num As Integer
 Name As String * 20
 Price As Single
End Type
Dim Classif(1 To 100) As TClassif
Dim Q

' Активизация экранной формы
Private Sub Form_Activate()
 Caption = "Расчет статистических частот"
 LstRez.AddItem ("ЦЕНА ЧАСТОТА")
 LstRez.Visible = False
 LblRez.Caption = "Статистические частоты"
 LblRez.AutoSize = True
 LblRez.Visible = False
 Q = 0
 TxtNum.SetFocus
End Sub

' Ввод очередной записи
Private Sub CmdInput_Click()
 Q = Q + 1
 Classif(Q).Num = Val(TxtNum.Text)
 Classif(Q).Name = TxtName.Text
 Classif(Q).Price = Val(TxtPrice.Text)
 TxtNum.SetFocus
End Sub
```

```
'Расчет статистических частот
Private Sub CmdProcess_Click()
 Dim FL As Boolean
 Dim I, J, K As Integer
 Dim Ws As String
 For I = 1 To Q
 K = 0
 ' Вычисление статистической частоты
 ' для элемента Classif(I).Price
 For J = 1 To Q
 If Classif(I).Price = Classif(J).Price Then
 K = K + 1
 End If
 Next J
 FL = True
 ' Проверка на повторный расчет частоты
 For J = 1 To I - 1
 If Classif(I).Price = Classif(J).Price Then
 FL = False
 End If
 Next J
 If FL Then
 ' Добавление нового значения в список
 Ws = Str(Classif(I).Price) + " " + Str(K)
 LstRez.AddItem (Ws)
 End If
 Next I
 LstRez.Visible = True
 LblRez.Visible = True
 LstRez.SetFocus
End Sub

'Завершение работы проекта
Private Sub CmdExit_Click()
 Unload Me
End Sub
```

10.6. Основные понятия о библиотеках динамической компоновки

Под библиотекой динамической компоновки DLL (Dynamic Link Library) понимается библиотека подпрограмм, которая загружается и связывается с приложениями в момент выполнения.

Необходимость применения данных библиотек можно рассмотреть на простом примере.

Если библиотечные функции статически связаны с программой, то код этих функций включается в исполняемый EXE-модуль приложения при его создании. Таким образом, при наличии, к примеру, 100 программ, каждая из которых выводит строку на экран помощью стандартной функции печати, код этой функции 100 раз повторяется.

В том случае, если библиотеки динамически связаны с программами, то при наличии 100 программ, каждая из которых выводит в окно строку текста, на жестком диске находится только одна копия кода функции печати. Все программы включают лишь небольшой фрагмент для вызова этого общего кода. Такие общие процедуры содержаться в библиотеках динамической компоновки, которые обычно имеют расширение DLL.

Значение DLL возрастает в связи с широким использованием данных библиотек при работе с COM-моделями и программирование в сети INTERNET.

DLL-библиотека является программным модулем. Она находится в памяти в единственном экземпляре и содержит сегмент кода и ресурсы, а также сегмент данных:

DLL-библиотека код
Ресурсы
Данные

DLL-библиотека, в отличие от приложения не имеет ни стека, ни очереди сообщений. Функции, помещенные в DLL, выполняются в контексте вызвавшего приложения, пользуясь

его стеком. Но эти же функции используют сегмент данных, принадлежащий библиотеке, а не копии приложения.

В силу такой организации DLL, экономия памяти достигается за счет того, что все запущенные приложения используют один модуль DLL, не включая те или иные стандартные функции в состав своих модулей.

Часто, в виде DLL создаются отдельные наборы функций, объединенные по тем или иным логическим признакам

Программы, работающие под управлением Windows, вызывают функции, предоставляемые операционной системой. Эти функции обеспечивают возможность создания окон, изменения их размера, считывания и записи данных в реестр, выполнения операций с файлами и т.д. Большинство таких функций хранится в файлах, организованных в виде динамических библиотек.

Для использования DLL необходимо знать, какие процедуры в ней находятся и какие аргументы следует передавать каждой из этих процедур. Функции Windows хорошо описаны в документации. Для вызова функций Windows API достаточно ознакомиться с документацией по соответствующей DLL. Чтобы применять другие DLL также придется разобраться с соответствующей документацией.

10.7. Создание DLL

DLL, как правило, пишутся на языке C++. Однако их можно создавать в различных системах, в частности в Visual Basic, в Delphi, в Visual FoxPro и др.

При работе в Visual Basic создание библиотеки DLL выполняется в следующей последовательности.

Во-первых, при создании нового проекта определяется тип разрабатываемого проекта как ActiveX DLL. Проекты данного типа включают в себя определения модуля класса (Class module).

Во-вторых, разрабатываются процедуры и функции, входящие в библиотечный модуль.

В-третьих, в диалоговом окне Project Properties определяются поля Project Name, Project Description и Application Title.

В-четвертых, созданный класс необходимо откомпилировать, чтобы им можно было пользоваться в приложениях. Для этого в меню File выполняется команда Make <Application>.DLL.

Разберем создание библиотеки .DLL на конкретном примере.

Требуется создать библиотеку, включающую две функции, предусматривающими работу с диалоговыми окнами. Создание проекта будет выполняться в следующей последовательности:

Во-первых, в меню File выбирается команда New Project.

Во-вторых, выбирается тип проекта ActiveX DLL.

В-третьих, в окне программы формируется обязательная процедура-заглушка Sub main().

В-четвертых, определяются следующие свойства формируемого модульного класса: свойству Name задается значение 'ClsDialogs' и свойству Instancing задается значение '5-Multiuse'. Свойство Instancing (Инстанция) устанавливает место вне проекта, где может быть затребована создаваемая библиотека. Значение 'MultiUse' определяет организацию многопользовательского доступа. Модульный класс будет представлен в виде файла 'ClsDialogs.cls'.

В-четвертых, определяются требуемые функции.

Первая функция с именем YNBox формирует элемент Message Box. Элемент Message Box обеспечивает формирование специального диалогового окна, предназначенного для выдачи сообщений и получения ответа. При описании функции YNBox используются специальные константы vbYesNo и vbQuestion, которые определяют особенности работы функции MsgBox.

Вторая функция с именем LoginBox формирует элемент Input Box. Элемент Input Box обеспечивает формирование специального диалогового окна, предназначенного для ввода текстовой информации. Для создания элемента Input Box в функции LoginBox формируется обращение к системной функции InputBox.

Полностью программный код, записанный в окне программы модуля будет иметь следующий вид:

```
      Sub Main()
      ' Заглушка (STUB)
      End Sub
       Public Function YNbox(title As String, msg
As String) As Integer
       Dim rc As Integer
       Dim DlgDef As Long
       DlgDef = vbYesNo + vbQuestion
       rc = MsgBox(msg, DlgDef, title)
       YNbox = rc
       End Function
       Public Function LoginBox(title As String,
msg As String, _
       default As String) As String
       Dim rc As String
       rc = InputBox(msg, title)
       LoginBox = rc
       End Function
```

В-пятых, созданный проект сохраняется под именем 'Dialogs.vbp'.

В-шестых, в меню Project выбирается команда Dialogs Properties и открывается диалоговое окно Project Properties. В поле Project Name вводится строка 'Dialogs'. В поле Project Description вводится строка "Класс для работы с диалоговыми окнами". На вкладке Make в поле Application Title вводится строка 'Dialigs'.

В-седьмых, в меню File выполняется команда 'Make Dialogs.Dll'.

10.8. Использование DLL

При работе в Visual Basic использование библиотеки DLL выполняется в следующей последовательности.

Во-первых, создается новые проект типа Standard Exe.

Во-вторых, устанавливается ссылка (Reference) на используемую библиотеку модульных классов.

В-третьих, формируются методы, обеспечивающие обращение к процедурам и функциям, имеющимся в библиотеке.

Разберем использование библиотеки .DLL на примере библиотеки, созданной в проекте 'Dialogs'. Разработка приложения, использующего библиотеку будет выполняться в следующей последовательности:

Во-первых, в меню File выбирается команда New Project и формируется проект Standard EXE.

Во-вторых, определяются следующие свойства экранной формы: свойству Name задается значение 'frmDialogs' и свойству Caption задается значение 'Создание диалоговых окон'.

В-третьих, формируются текстовые поля для ввода информации, передаваемой при вызове DLL-библиотеки. У первого текстового поля определяются следующие свойства: свойству Name задается значение 'txtTitle' и свойству Text задается значение пустого поля. У второго текстового поля определяются следующие свойства: свойству Name задается значение 'txtMsg' и свойству Text задается значение пустого поля.

В-четвертых, формируются надписи для вводимой информации. У первой надписи определяются следующие свойства: свойству Name задается значение 'lblTitle' и свойству Caption задается значение 'Заголовок'. У второй надписи определяются следующие свойства: свойству Name задается значение 'lblMsg' и свойству Caption задается значение 'Сообщение'.

В-пятых, формируются командные кнопки, обеспечивающие работу с билиотекой-DLL. У первой командной кнопки определяются следующие свойства: свойству Name задается значение 'cmdYN' и свойству Caption задается значение 'Да/Нет'. У второй надписи определяются следующие свойства: свойству Name задается значение 'cmdLogin' и свойству Caption задается значение 'Регистрация'.

В-шестых, выполняется команда REFERENCES из меню Project. В окне 'Available References' (Доступные ссылки) при-

веден список библиотек, которые могут быть использованы в разрабатываемом приложении. Операционная система включила в этот список ранее созданную библиотеку модульных классов с описательным именем (description) 'Класс для работы с диалоговыми окнами'. Для установления связи требуется поставить флажок слева от данной строки.

В-седьмых, разрабатывается программный код, обеспечивающий вызов функций, помещенных в DLL-библиотеку. Полностью программный код, записанный в окне программы модуля будет иметь следующий вид:

```
Option Explicit
' Описание объекта, формируемого на основе
' модульного класса
Dim dlg As ClsDialogs

'Формирование объекта при загрузке
'экранной формы
Private Sub Form_load()
 Set dlg = New ClsDialogs
 End Sub

'Уничтожение объекта при удаление
'экранной формы с экрана
Private Sub Form_Unload(Cancel As Integer)
 Set dlg = Nothing
End Sub

'Организация обращения к функции YNBox,
'расположенной в DLL-библиотеке
Private Sub cmdYn_Click()
 Dim Rc As Integer
 Rc = dlg.YNbox(TxtTitle.Text, TxtMsg.Text)
 If Rc = vbYes Then
 MsgBox "Нажата кнопка Yes"
 Else
 MsgBox "Нажата кнопка NO"
 End If
End Sub
```

```
      'Организация обращения к функции LoginBox,
      'расположенной в DLL-библиотеке
      Private Sub CmdLogin_Click()
      Dim UserId As String
      UserId = dlg.LoginBox(TxtTitle.Text,
TxtMsg.Text, "")
       If UserId = "Смирнов" Then
       MsgBox UserId & " успешно зарегистрирован!"
       Else
       MsgBox "Вам отказано в регистрации"
       End If
      End Sub
```

10.9. Вызов DLL-процедур

Вызов процедур из DLL аналогичен вызову стандартных процедур. Разница заключается в том, что тело процедуры находится не в программном модуле, а в соответствующей DLL.

Рассмотрим реализацию вызова процедур и функций из DLL на примере VBA(Visual Basic Application).

Перед вызовом функции DLL необходимо сообщить VBA, где она находиться. В действительности существует два типа DLL, и вы сообщаете VBA, как их вызывать, двумя способами:

- указав библиотеку типа;
- используя операторы Declare.

В первом случае программист, разрабатывающий DLL, создает файл, который называется библиотекой типа, и описывая процедуры в рамках DLL. Библиотека типа обычно имеет расширение OLB или TLB и регистрируется с помощью OLE-компонента Windows. Программа установки, инсталлирующая DLL, как правило, вносит соответствующие записи в реестр Windows, регистрируя эту библиотеку. Если выбрать команду Ссылки(References) из меню Сервис(Service) при открытом модуле, в диалоговом окне Ссылки отображается список всех доступных зарегистрированных библиотек типа. Установив флажок возле имени вашей библиотеки типа, вы обеспечиваете доступ к ней из VBA.

При использовании библиотеки типа нет необходимости применять в программе операторы Declare. Библиотека типа включает все функциональные возможности оператора Declare, а также позволяет избежать трудностей, связанных с передачей строк функциям DLL. Но, к сожалению, Windows API не имеет библиотеки типа, и для вызова функций Windows API следует использовать операторы Declare.

Второй вариант предусматривает использование оператора Declare. Оператор Declare представляет собой описание, которое вводится в раздел описаний модуля и сообщает VBA о том, где находится функция и как ее вызывать. Важно отметить, что оператор Declare необходимо использовать для вызова функций DLL, не указанных в библиотеке типа. Поскольку компания Microsoft не предоставляет библиотек типа для функций Windows API, для их вызова нужно включать в раздел описаний соответствующие операторы Declare.

К счастью эти операторы уже существуют. Компанией Microsoft предоставлен файл WIN32API.TXT, включающий большинство операторов Declare, которые могут понадобиться для разработки приложений и определения некоторых констант и описания пользовательских типов.

Многие программные продукты Microsoft, предназначенные для разработчиков, снабжены сервисной утилитой API Text Viewer. Эта утилита тоже предоставляет необходимые операторы Declare и другие определения. Она просматривает файл WIN32API.TXT и находит в нем требуемое значение. К сожалению, пользовательский интерфейс данной утилиты затрудняет ее использование. Для поиска оператора Declare в файле WIN32API.TXT проще воспользоваться текстовым редактором. Вполне вероятно, что по мере развития Win32 API появятся бесплатные или условно-бесплатные программные средства, предоставляющие операторы Declare и другие определения.

Как упоминалось, операторы Declare необходимо размещать в разделе описаний модуля. После задания оператора Declare можно использовать процедуру, которая описана этим оператором, как обычную встроенную функцию.

Тема 11.

Операционные системы, как важнейший элемент системного ПО

11.1. Функции операционных систем

Под операционной системой понимается комплекс управляющих программ, который управляет компьютером, запускает программы, обеспечивает защиту данных, выполняет различные сервисные функции по запросам пользователя и программ.

Каждая программа пользуется услугами операционной системы и может работать только под управлением операционной системы. Программа, разработанная под управлением операционной системы, может выполняться либо под управлением данной системы, либо под управлением операционной системы, совместимой с данной системой.

Операционная система определяет производительность работы прикладной программы, степень защиты данных, необходимые аппаратные средства и т.д.

Для расширения возможностей операционной системы существует совокупность системных программ. В частности:

Во-первых, драйверы, которые позволяют работать программам с различными внешними устройствами, использовать различные протоколы обмена данными и т.д.

Во-вторых, программы оболочки, которые обеспечивают более удобный и наглядный способ общения с компьютером. Примером программы оболочки является программа NORTON COMMANDER.

В-третьих, архиваторы, которые позволяют обеспечить эффективное хранение больших объемов информации на дисках.

В-четвертых, утилиты работы с дисками, которые позволяют следить за состоянием дисков и обеспечивать восстановление файлов.

В-пятых, антивирусные, программы, которые предназначены для предотвращения заражения компьютерным вирусом и ликвидаций последствий заражения .

В-шестых, переводчики, которые, в частности, облегчают работу с документацией на английском языке.

11.2. Краткая характеристика различных операционных систем

Среди наиболее популярных операционных систем можно выделить следующие:

Во-первых, операционные системы семейства WINDOWS. В Windows используется графический интерфейс. Все необходимые пользователю действия выполняются со стандартизированными элементами диалога. Операционные системы семейства Windows являются многозадачными операционными системами. Это значит одновременно могут выполняться несколько программ и информация может передаваться из одной операционной системы в другую. Наиболее распространенными операционными системами семейства Windows являются Windows 95, Windows 98, Windows NT, Windows 2000.

Во-вторых, операционная система UNIX. Система UNIX является многопользовательской операционной системой с эффективной технологией авторизации доступа различных пользователей к файлам. Программа пользователя может получить доступ к файлу только в том случае, если хранящиеся при файле ограничения доступа позволяют это сделать. Отличительными свойствами системы UNIX являются надежность и масштабируемость. Операционная система UNIX получила распространение при работе в корпоративных сетях. Имеются версии UNIX, предназначенные для работы с 64 разрядными процессорами.

В-третьих, операционная система LINUX. Система LINUX является свободно распространяемой по сети INTERNET открытой системой. Основные модули системы LINUX рас-

пространяются в виде исходных текстов, снабженных документацией. Открытость системы заключается в возможности добавления самостоятельно разработанных блоков, обеспечивающих наиболее эффективную конфигурацию системы для конкретного применения. Особенностями операционной системы LINUX являются следующие:

во-первых, широкий диапазон поддерживаемых аппаратных платформ;

во-вторых, разнообразие применений, включая рабочие места, серверы, конкретные приложения;

в-третьих, возможность обеспечения совместимости с имеющимся программным обеспечением.

В-четвертых, операционная система OS/2 Warp. В операционной системе OS/2 Warp могут выполняться практически все программы, разработанные в среде Windows. OS/2 Warp достаточно хорошо защищена от некорректного поведения программ и может использоваться в ответственных применениях.

11.3. Понятие файловой системы

Под файлом понимается поименованная область на внешнем носителе информации. Файл может располагаться на жестких дисках, дискетах, компакт-дисках и т.д. В файле может содержаться следующая информация: во-первых, текст программы на исходном языке; во-вторых, готовый к выполнению модуль во внутренних кодах машины; в-третьих, данные для работы программы; в-четвертых, текстовые документы; в-пятых, электронные таблицы; в-шестых, закодированное графическое изображение; в-седьмых, закодированная звуковая информация и т.д.

Физически файл состоит из совокупности участков (блоков, кластеров). Кластер является минимальной единицей размещения информации на диске. Кластеры одного и того же файла могут располагаться в различных местах диска.

Имя файла, как правило, состоит из двух частей, разделенных точкой. Первая часть представляет собой индивидуальное имя, отличающее один файл от другого. Вторая часть (расширение), представляет собой типовое имя, отличающее группу файлов, обрабатываемых одним приложением, от файлов обрабатываемых другими приложениями.

Можно выделить группы файлов, которые представляют собой программы, подготовленные к выполнению. К данным группам относятся следующие типы файлов:

во-первых, файлы с расширением '**.COM**', которые называются программными файлами, являющимися внешними командами;

во-вторых, файлы с расширением '**.EXE**', которые называются исполняемыми файлами;

в-третьих, файлы с расширением '**.BAT**', которые называются командными файлами.

Под файловой системой (FILE SYSTEM) понимается набор соглашений, определяющих организацию данных на внешних носителях информации. Файловая система определяет:

во-первых, способ хранения файлов на диске;

во-вторых, структуру сведений о хранящихся файлах;

в-третьих, представление информации о свободных участках на диске;

в-четвертых, особенности представления другой служебной информации.

Можно выделить следующие файловые системы.

Во-первых, файловая система FAT (FILE ALLOCATION TABLE). Файловая система FAT поддерживает имена файлов из 8 символов плюс три символа в расширении имени. Данная система малопроизводительна для больших дисков и не приспособлена к многозадачной работе. В системе FAT размер кластера зависит от емкости диска. Поэтому при использовании дисков большого объема размер кластера становится недопустимо большим. При записи значительного числа коротких файлов дисковое пространство расходуется непроизводительно.

Во-вторых, файловая система VFAT(FAT32). Система VFAT представляет собой модификацию системы FAT, ориен-

тированную на операционные системы WINDOWS 95, WINDOWS 98. Основным отличием является возможность использования имен длиной до 255 символов.

В-третьих, файловая система HPFS (HIGH PERFORMANCE FILE SYSTEM). Эта система ориентирована на диски большого объема, содержащие множество файлов. В системе HPFS приняты существенные меры по обеспечению эффективности хранения данных. В частности, система HPFS использует кластеры фиксированного размера. Файловая система HPFS используется в операционной системе OS/2.

В-четвертых, файловая системы NTFS, которая используется в операционных системах WINDOWS NT, WINDOWS 2000. Отличительными особенностями этой системы являются обеспечение надежности файловой системы и возможность восстановления данных при сбоях. Для этого файловая система NTFS дублирует всю важную информацию и обеспечивает регистрацию всех изменений на дисках в специальном файле регистрации.

В-пятых, файловая система CDFS, предназначенная для работы с компакт-дисками. Физическое устройство компакт дисков не такое, как у жестких дисков или дискет. В компакт дисках информация записывается не в кольцевых дорожках, а в единственной спиралеобразной дорожке. В операционных системах Windows поддержка CDFS является встроенной.

11.4. Многоуровневая система каталогов

Группа логически взаимосвязанных файлов может быть объединена в единый каталог. В операционных системах семейства Windows каталоги называются папками (FOLDER).

Каталог представляет собой специальный файл, в котором хранятся имена файлов, сведения о размере файлов, времени их последнего обновления, атрибуты (свойства) файлов и т.д.

Допускается вхождение одних каталогов в другие. Каталог, который входит в другой каталог, называется подкатало-

гом. Каталог, содержащий другие каталоги, называется родительским каталогом или над каталогом. Каталог, с которым в настоящий момент работает пользователь называется текущим.

Одноименные файлы могут входить в различные каталоги. Одноименные файлы, расположенные в различных каталогах могут иметь различное содержание.

Для точной идентификации файла необходимо, кроме имени указать местоположение, т.е. цепочку подчиненных каталогов. Цепочка называется маршрутом или путем по файловой системе. Имена каталогов в маршруте разделяются символом ' \ '(обратная наклонная черта). Перед маршрутом может задаваться имя внешнего устройства. Имя внешнего устройства отделяется от маршрута символом ':' (двоеточие). Например: C:\SMIRNOV\WORK.TXT

Для указания группы файлов, выбираемых из одного каталога, предназначены символы '*' (звездочка) и '?'(вопросительный знак).

Символ '*' обозначает любое число любых символов в имени файла или в расширении имени файла. Например, запись C:\SMIRNOV*2*.* означает обращение ко всем файлам, имеющим цифру 2 в любой позиции.

Символ '?' обозначает один произвольный символ или отсутствие символа. Например, запись C:\SMIRNOV\WORK?.TXT означает обращение к текстовым файлам с именами WORK, WORK1, WORK2 и т.д.

11.5. Представление экономической информации в виде файлов

Файлы, используемые для хранения экономической информации могут быть организованы различными способами. Можно выделить следующие типы файлов:

Во-первых, текстовые файлы. Текстовые файлы предусматривают хранение алфавитно-цифровой информации в виде отдельных символов, закодированных в соответствии со

стандартами ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE) или ANSI (THE AMERICAN NATIONAL STANDARD INSTITUTE). Текстовые файлы не содержат управляющую информацию, ориентированную на конкретную систему обработки информации.

Данные файлы могут создаваться и обрабатываться практически всеми прикладными системами. Например: программами, разработанными в системе DELPHI; системой управления базами данных Visual FoxPro; табличным процессором EXEL; текстовым редактором WORD. Текстовые файлы являются универсальным способом передачи данных между различными прикладными программными продуктами. Как правило, текстовые файлы имеют расширение '**.TXT**'.

Во-вторых, файлы базы данных. Файлы базы данных создаются системами управления базами данных для хранения обрабатываемой информации. Обрабатываемая информация в базе данных хранится в виде совокупности взаимосвязанных таблиц. В системе Visual FoxPro база данных организована при помощи файлов с расширениями '**.DBC**' и '**.DBF**'.

В-третьих, файлы электронных таблиц. Данные файлы создаются и обрабатываются табличными процессорами. Электронная таблица представляет собой прямоугольную сетку, в которой хранится информация. В системе EXCEL электронные таблицы имеют расширение '**.XLS**' .

11.6. Особенности операционных систем семейства WINDOWS

Операционные системы семейства Windows являются широко используемыми и эффективными. Разрабатываются операционные системы Windows фирмой Microsoft. Операционные системы семейства Windows постоянно совершенствуются. Каждая новая версия системы включает в себя те черты предыдущей версии, которые показали свою эффективность.

Отличительной особенностью WINDOWS является эффективная реализация принципа WYSIWYG (WHAT YOU SEE

IS WHAT YOU GET; что видите, то имеете). В соответствии с этим принципом экран дисплея рассматривается как поверхность рабочего стола. Отдельные участки экрана называются окнами (window). Разделение экрана на окна позволяет одновременно управлять выполнением нескольких процессов.

Эффективное представление совокупности выполняемых процессов на одном экране достигается благодаря использованию специальных графических символов – «пиктограмм». Пиктограмма (ICON, иконка, значок) представляет обрабатываемый объект в виде закрепленного за объектом графического символа.

Для быстрого перехода от объекта к объекту предназначено специальное устройство – «манипулятор мышь». Мышь представляет собой специальное устройство, которое обеспечивает перемещение указателя (курсора) мыши на экране. Курсор мыши используется для того, чтобы указывать на различные объекты, запускать программы, выбирать пункты меню, выделять текст, перетаскивать объекты и для других действий. Вид указателя меняется в зависимости от операций, которые выполняет пользователь. Имеется свыше десяти различных вариантов указателя мыши в зависимости от места его использования.

Можно выделить следующие характерные черты операционных систем семейства Windows :

Во-первых, Windows является операционной системой, которая безраздельно управляет всеми ресурсами.

Во-вторых, система Windows является многозадачной и многопоточной. Процессы в разных окнах идут одновременно.

В-третьих, система имеет широкие сетевые возможности. Она оснащена множеством драйверов и поддерживает разнообразные сетевые протоколы.

В-четвертых, система имеет удобный пользовательский интерфейс.

При разработке операционной системы WINDOWS 95 были введены некоторые термины и понятия. В частности:

Во-первых, рабочее место получило название «РАБОЧИЙ СТОЛ» (DESKTOP).

Во-вторых, все аппаратные и программные ресурсы компьютера как-то диски, модемы, принтеры, файлы, ярлыки распределены по папкам (FOLDER). Все папки образуют единую иерархическую систему.

В-третьих, для обозначения ссылок на объект предусмотрено использование ярлыков (SHORTCUT). Для одного объекта может быть задано несколько ярлыков, расположенных в самых разных частях файловой системы. Ярлыки могут указывать на любые объекты, включая папки, диски , компьютеры и принтеры.

В-четвертых, под документом в Windows понимаются не только текстовые файлы, а практически любой файл ,содержащий информацию, например: текст, графическое изображение, электронная таблица, звуковая запись, видеофильм. Документ связан с приложением, в котором он создан. При обращении к пиктограмме документа (двойной щелчок мыши) вызывается приложение и в него загружается выбранный документ.

В-пятых, введен новый инструмент для характеристики объектов, который называется «Свойства» (PROPERTIES). При выделении объектов с помощью команды «Свойства» можно вызвать «Окно свойства» (PROPERTY SHEET). В этом окне объединена справочная информация об объекте и некоторый набор инструментов, позволяющий изменить его свойства.

При разработке операционной системы WINDOWS 98 были добавлены следующие понятия и возможности:

Во-первых, введено новое понятие ACTIVE DESKTOP (активный рабочий стол), предназначенное для расширения возможностей работы в сети INTERNET.

Во-вторых, в операционную систему включена программа INTERNET EXPLORER, предназначенная для просмотра WEB-страниц.

В-третьих, для освоения сложных средств операционной системы разработана совокупность мастеров (WIZARD).

В-четвертых, для обнаружения и устранения ошибок, возникших из-за некорректного конфигурирования системы

или аппаратуры разработан набор монтеров (TROUBLESHOOTERS).

При разработке операционной системы WINDOWS 2000 предусмотрены следующие особенности:

Во-первых, реализованы эффективные средства защиты данных;

Во-вторых, используется технология, обеспечивающая работу с большими наборами данных;

В-третьих, в систему включены некоторые элементы искусственного интеллекта;

В-четвертых, вводится новое понятие «персонализация меню». Под персонализацией понимается автоматическое формирование меню в зависимости от накопленной частоты обращений к конкретным пунктам;

В-четвертых, расширены возможности работы в сети INTERNET. Как сервер интернета и интрасетей операционная система лучше защищена.и более полно соответствует стандартам;

В-пятых, разработано большое количество разнообразных сервисных программ. В частности: сервисная программа повышающая эффективность работы с устройствами мультимедиа; сервисная программа голосового сопровождения происходящего на экране и другие.

11.7. Организация рабочего стола

Рабочее место пользователя организовано в виде Рабочего стола (DESK TOP). На рабочем столе располагаются обозначения всех ресурсов, к которым имеется доступ. В нижней части рабочего стола располагается Панель задач (TASKBAR). На панели задач располагается кнопка Пуск (START), кнопки активных приложений, переключатель кодировки клавиатуры, часы и некоторые другие элементы.

Кнопка «Пуск» открывает доступ к Главному меню (START MENU). При помощи Главного меню обеспечивается реализация системы вложенных меню и переход в требуемую

прикладную программу. Кроме того, последняя кнопка Главного меню обозначенная как «Завершение работы» (SHUTDOWN), обеспечивает подготовку компьютера к выключению.

Кнопки активных приложений позволяют видеть какие приложения запущены и обеспечивают переход из одного приложения в другое.

Наиболее важные инструменты Windows, как правило, представлены в левой части рабочего стола в виде столбца значков (пиктограмм). Обычно, там располагаются следующие значки:

Во-первых, значок «Мой компьютер» (MY COMPUTER), который обеспечивает доступ ко всем ресурсам компьютера.

Во-вторых, значок «Сетевое окружение» (NETWORK NEIGHBOURHOOD), который позволяет получить доступ ко всем ресурсам локальной сети.

В-третьих, значок «ИНТЕРНЕТ» (INTERNET), который обеспечивает доступ к сети INTERNET.

В-четвертых, значок «Входящие» (INBOX), который предназначен для работы с электронной почтой, а также для приема и передачи факсимильных сообщений.

В-пятых, значок «Корзина» (RECYCLE BIN), который позволяет временно сохранять уничтоженные файлы.

В WINDOWS 98 наряду с традиционным рабочим столом WINDOWS используется специальный активный рабочий стол ACTIVE DESKTOP. Интерфейс рабочего стола ACTIVE DESKTOP представляет собой вариант страницы WEB для сети INTERNET. Активный рабочий стол может содержать WEB-страницы, динамические страницы языка разметки гипертекстов HTML (HYPER TEXT MARKUP LANGUAGE), а также отдельные компоненты языка JAVA.

11.8. Проводник

Проводник (EXPLORER) представляет собой инструмент, предназначенный для отображения всех ресурсов компьютера и доступа к ним. Проводник вызывается из меню ко-

манды «Программы» (PROGRAMS), которая входит в «Главное меню» (START MENU).

Как правило, окно Проводника включает две панели. Левая панель показывает все ресурсы, представленные в виде иерархического дерева. Эта панель включает все объекты, в том числе папки «Мой компьютер» и «Сетевое окружение».

В правой области (панели содержимого) на экран выводится содержимое выбранной папки. Объекты, входящие в папку могут выдаваться в одном из четырех видов: во-первых, в виде крупных значков; во-вторых, в виде мелких значков; в-третьих, в виде списка; в- четвертых, в виде таблицы. Выбор формы представления осуществляется с помощью меню «Вид».

Если объекты содержат вложенные папки, то в дереве они помечаются маленьким значком «плюс» (+). Для отображения структуры папок нужно щелкнуть мышкой на значке «плюс». Когда папка откроется и отобразится ее структура, знак «плюс» изменится на знак «минус» (–).

Для свертывания папки нужно щелкнуть мышкой на значке «минус». Свертывание папки дает возможность увидеть в дереве большее число объектов.

Для того, чтобы открыть объект, находящийся внутри папки необходимо дважды щелкнуть мышкой на объекте в панели содержимого. Второй вариант открытия объекта заключается в выделении объекта и выборе из меню «Файл» команды «Открыть».

Если объект является программой, то Windows запускает эту программу. Если объект является документом, то Windows запускает программу, в которой этот документ создан, и загружает в нее документ.

Функционированием Проводника можно управлять с помощью параметров, которые находятся в меню «ВИД». В меню «ВИД» для настройки Проводника используются следующие пункты:

Во-первых, пункт «Панель инструментов», который включает и выключает показ панели инструментов. Использование элементов панели инструментов позволяет ускорить работу.

Во-вторых, пункт «Строка состояния», который включает и выключает вывод на экран строки состояния Проводника. Строка состояния содержит информацию о текущем выделенном объекте. Если выбран диск, то строка состояния укажет объем свободного пространства на нем. При выделении файлов строка состояния показывает размер дискового пространства, занятого выделенными файлами.

В-третьих, пункт «Параметры», который позволяет обратиться к листу свойств Проводника . Лист свойств определяет ,будут или нет некоторые типы файлов показываться в панели содержимого окна Проводника и в папке на рабочем столе .

11.9. Работа с файлами и папками

Операционные системы семейства Windows поддерживают длинные имена файлов. Длинные имена файлов могут включать символы, недопустимые в именах файлов формата «8.3». Например, квадратные скобки [ ]. В длинных именах файлов различаются строчные и заглавные буквы.

Для того, чтобы с файлами, созданными в операционной системе Windows, могли работать программы, не поддерживающие длинные имена, для каждого файла автоматически создается короткое имя. Короткое имя для формата «8.3» создается из первых шести знаков длинного имени, тильды (~) и цифры. Символы, не разрешенные к использованию в коротких именах, игнорируются.

Для работы с файлами используется Проводник (EXPLORER). Проводник позволяет копировать, перемещать, переименовывать, удалять и восстанавливать файлы.

Для выполнения требуемой обработки можно выделить один или несколько файлов. Выделение одиночного объекта обеспечивается щелчком мыши. Выделение группы смежных объектов обеспечивается последовательностью следующих действий: выделяется первый объект, нажимается клавиша «SHIFT» и выделяется последний объект. Для выделения совокупности

отдельно расположенных объектов нажимается клавиша «CTRL» и с помощью мыши выделяются отдельные объекты.

Копирование или перемещение файлов или папок обеспечивается последовательностью следующих действий. Дерево на панели Проводника располагается таким образом, чтобы можно было видеть как текущее место расположения объектов, так и их новое место расположения. Затем выделенный объект перетаскивается на новое место.

Перетаскивание выполняется с использованием либо правой клавиши мыши, либо левой клавиши мыши. Отличие заключается в том, что при использовании правой клавиши появляется контекстное меню, которое позволяет выбрать необходимую операцию копирования или перемещения. При использовании левой клавиши контекстное меню не появляется, но сама система в зависимости от анализа ситуации выполнит либо копирование, либо перемещение.

Переименование файла или папки обеспечивается последовательностью следующих действий. Сначала выделяется объект, затем выделяется имя объекта и набирается новое имя.

Для создания папки прежде всего открывается папка, в которой создается новая папка. Затем используя правую клавишу мыши вызывается контекстное меню, в котором выбираются пункты «Создать» и «Папка».

Удаления объекта обеспечивается выделением объекта и вызовом контекстного меню с помощью правой клавиши мыши.

Все перечисленные операции над объектами, кроме того, можно выполнить используя команды меню «ФАЙЛ».

11.10. Ярлыки объектов

Под ярлыком объекта понимается путь к объекту, который хранится отдельно. Ярлык не является объектом, а представляет собой ссылку на объект.

Один объект может иметь несколько ярлыков, расположенных в самых различных частях файловой системы. Ярлык

объекта позволяет сделать один и тот же объект доступным из разным мест и позволяет избежать появления различных версий объекта (документа, программы).

Удаление ярлыка не приводит к уничтожению самого объекта. Ярлыки могут указывать на любые объекты, включая папки, диски и принтеры. С ярлыком можно выполнять все те же операции, что и с обычными значками.

Значок ярлыка повторяет значок того объекта, на который этот ярлык ссылается, но к нему в нижней части добавлен маркер в виде стрелки.

11.11. Настройка операционной системы

Средства настройки применяются для изменения состава аппаратных или программных средств, установленных на компьютере, а также для модификации установок операционной системы Windows. В исходном состоянии операционная система функционирует в соответствии с настройками, сделанными разработчиками или поставщиками WINDOWS по умолчанию (DEFAULT SETTING).

Для настройки WINDOWS на конкретного пользователя имеется совокупность средств, доступ к которым обеспечивается через команду «НАСТРОЙКА» (SETTINGS) главного меню (START MENU).

В меню команды «НАСТРОЙКА» включены следующие инструменты:

во-первых, панель управления (CONTROL PANEL);

во-вторых, панель задач (TASK BAR & START MENU);

в-третьих, принтеры (PRINTERS);

в-четвертых, настройка папок (FOLDER OPTIONS);

в-пятых, настройка активного рабочего стола (ACTIVE DESKTOP);

в-шестых, обновление Windows (WINDOWS UPDATE).

Панель управления представляет собой специальную папку, в которой собраны многочисленные средства для на-

стройки компьютера, подключения нового оборудования, установки нового программного обеспечения. В Windows реализован объектно-ориентированный подход. Поэтому обращение к какому-либо элементу панели управления вызывает появление специального инструмента для работы с объектами, который называется «Окно свойств» (PROPERTY SHEET).

В Панели управления можно выделить следующие элементы:

Во-первых, элемент «Дата/Время» (DATE/TIME), который позволяет устанавливать системную дату и время с учетом часового пояса и перехода на летнее время.

Во-вторых, элемент «Звук» (SOUNDS), который используется для установки различных звуковых сигналов для событий операционной системы WINDOWS. Например для таких событий, как «Запуск Windows», «Выход из Windows», «Восстановление окна», «Приход почты», «Ошибка программы».

В-третьих, элемент «Клавиатура» (KEYBOARD). Данный элемент служит для установки параметров клавиатуры, в частности: частоты мигания курсора, типа клавиатуры, сочетания клавиш для переключения.

В-четвертых, элемент «Мышь» (MOUSE), который предназначен для установки параметров мыши, таких как: назначение кнопок мыши; скорость перемещения указателя, тип мыши.

В-пятых, элемент «Установка и удаление программ» (ADD NEW SOFTWARE), который позволяет установить или удалить прикладную программу, а также добавить или удалить какие-либо компоненты Windows.

В-шестых, элемент «Установка оборудования» (ADD NEW HARDWARE). Обращение к этому элементу запускает программу NEW HADWARE WIZARD (Мастер установки оборудования), которая предназначена для включения в конфигурацию новых устройств.

В-седьмых, элемент «Экран» (DISPLAY), который позволяет устанавливать вид рабочего стола, цветовую гамму и другие параметры экраны.

В-восьмых, элемент «Шрифты» (FONTS), который позволяет просматривать, устанавливать и удалять шрифты.

В-девятых, элемент «Система» (SYSTEM), который представляет основную информацию о компьютере и позволяет изменить наиболее важные установки. При работе с этим элементом во многих случаях выдается предупреждение о том, что изменение параметров должны производить системные администраторы.

Инструмент настройки «Панель задач» предназначен для формирования и модификации «Главного меню» и «Панели задач». Настройка осуществляется с помощью окна свойств «Свойства: панель задач» (TASKBAR PROPERTIES).

Инструмент настройки «Принтеры» (PRINTERS) служит для подключения и отсоединения принтеров, а также для управления их свойствами. Обращение к инструменту настройки «Принтеры», как правило, выполняется с использованием ярлыка «Принтеры», располагаемого в панели управления.

В операционной системе Windows 98 имеется специальный пункт настройки ACTIVE DESKTOP, который предназначен для настройки активного рабочего стола. Для установки активного рабочего стола используются три команды: во-первых, команда VIEW AS WEB PAGE (Показать как WEB-страницу); во-вторых, команды CUSTOMIZE BY DESKTOP (Настроить рабочий стол); в-третьих, команда UPDATE NOW (Обновить сейчас).

11.12. Обмен данными

Операционная система Windows предоставляет возможность различным программам в процессе работы обмениваться данными между собой. Для этой цели предусмотрены:

во-первых, Папка обмена;

во-вторых, технология OLE (OBJECT LINKING AND EMBEDDING).

Папка обмена позволяет организовывать обмен данными между различными компьютерами, соединенными в сеть.

Важнейшей частью Папки обмена является буфер обмена, который обеспечивает обмен данными между различными программами, выполняющимися на данном компьютере.

Буфер обмена (CLIPBOARD) представляет собой специальным образом организованную область памяти, которая используется для переноса и копирования данных.

Операция переноса данных (MOVE) удаляет их в исходной позиции и вставляет их в новом месте. При копировании (COPY) данные остаются в старой позиции и дополнительно вставляются в указанном месте.

При переносе и копировании данные сначала перемещаются в буфер обмена (CLIPBOARD). Они хранятся в нем до тех пор, пока не будет выполнена следующая операция копирования или переноса. При выполнении новой операции копирования предыдущее содержимое CLIPBOARD уничтожается.

Перенос и копирование данных с использованием буфера обмена функционирует для всех прикладных программ среды WINDOWS одинаково. Как правило, прикладные программные продукты имеют в меню EDIT (правка) следующие директивы: CUT (вырезать), COPY (копировать) и PASTE (вставить).

При помощи буфера обмена можно выполнить копирование информации между различными приложениями среды WINDOWS. В частности, при работе в среде Visual FoxPro, копировать участки программы в текстовый отчет, формируемый с помощью текстового редактора WORD. Для этого следует запустить и Visual FoxPro, и WORD. Затем находясь в Visual FoxPro выполнить операцию копирования (COPY) информации в CLIPBOARD. После чего, перейти в WORD и выполнить операцию вставки содержимого буфера в выбранную позицию (PASTE).

Для того, чтобы в процессе выполнения Visual FoxPro распечатать экран, требуется нажать клавишу “PRINT SCREEN”. При нажатии клавиши текущее состояние экрана копируется в CLIPBOARD. Из CLIPBOARD информация переносится с помощью текстового редактора (например, WORD) в текстовый файл. Затем текстовый файл распечатывается.

OLE (OBJECT LINKING AND EMBEDDING) – связывание и встраивание объектов (связь и внедрение), представляет собой один из механизмов обмена данными. Объект OLE представляет собой программно обрабатываемый элемент данных. Например, объектом могут являться следующие элементы: диаграмма, рисунок, текст, таблица.

Объект связан с прикладной программой, при помощи которой он создавался. Это обеспечивает возможность обращения к прикладной программе при помощи объекта. Если необходимо внести изменения в объект, то возможность использования инструмента, обеспечивающего изменения, предоставляется при обращении к объекту. Например, в таблицу Visual FoxPro встроен рисунок, созданный при помощи графического редактора PAINT. Если необходимо, вновь задействовать инструментарий PAINT, то выполняется двойной щелчок мыши по встроенному рисунку.

OLE реализуется в том случае, когда несколько программ WINDOWS одновременно функционируют в мультизадачном режиме.

Прикладные программы, поддерживающие связь и внедрение объектов, делятся на две категории:

Во-первых, программы, объекты которых встраиваются или связываются с другими прикладными программами. Эти программы называются серверами (SERVER).

Во-вторых, программы, позволяющие принимать встраиваемые или связываемые объекты. Эти программы называются клиентами CLIENT.

Например, если в таблицу Visual FoxPro встраивается картинка, нарисованная с помощью средств PAINT, то PAINT будет являться сервером, а Visual Foxpro будет являться клиентом.

Связывание и встраивание объектов различаются следующим образом.

При использовании «связывания» (LINKING) объект сохраняется в отдельном файле. Формат файла определяется родительским для данного объекта приложением (сервером).

При открытии файла, с которым связаны некоторые внешние объекты, выполняется актуализация, и в файл принимается свежая информация из связанных объектов. Один объект можно связать с несколькими файлами. Внесенные в этот объект изменения будут отражены во всех файлах, с которыми связан этот объект.

При «встраивании» (EMBEDDING) объекта Клиент сохраняет объект в своем файле.

11.13. Средства помощи и обучения

В операционной системе Windows разработаны специальные средства, позволяющие постоянно повышать квалификацию пользователя. Можно выделить следующие основные элементы:

во-первых, учебник по Windows;

во-вторых, режим помощи (HELP).

Учебник по Windows представляет собой обучающую программу. Обучающая программа контролирует процесс обучения. При наличии ошибок в усвоении материала учебник автоматически дает пояснения. В необходимых случаях выдаются рекомендации. Учебник использует специальные средства пояснения. Например, курсор необычайно больших размеров.

Режим помощи используется при возникновении затруднений. Можно выделить следующие особенности организации помощи:

Во-первых, выдаваемая справочная информация организована в виде гипертекста. Гипертекстом называется текст, представленный в виде ассоциативно связанных блоков. Использование гипертекста позволяет выделять отдельные понятия, а затем применяя ассоциативные связи, двигаться в любых направлениях.

Во-вторых, в справочной системе Windows доступ к информации реализован с использованием различных подходов: через содержание (оглавление); через предметный указатель; через контекстный поиск.

В-третьих, справочная система обеспечивает обращение к рассмотренным программам.

Пример использования справочной системы. Допустим, возникла необходимость применения стандартных средств операционной системы Windows таких как: Учебник по Windows; Текстовый редактор WORDPAD, Программы, обеспечивающей просмотр буфера обмена, но их на компьютере не оказалось. Так как, данные средства являются стандартными программами и должны быть в любой системе Windows, то проблема сводится к их подключению.

Для подключения стандартных программ целесообразно использовать справочную систему. Через предметный указатель справочной системы ищется раздел «Стандартные программы установка». При обращении к этому разделу появляется справочное окно с разъяснениями по реализации установки стандартных программ. В окне имеется кнопка, которая обеспечивает обращение к программам установки.

Тема 12.

Сервисные программы для системы WINDOWS

12.1. Назначение сервисных программ

Под сервисными программами понимается совокупность обслуживающих программ, предназначенных для организации эффективной работы пользователя. В сервисных программах можно выделить группу программ-утилит, предназначенных для обслуживания жестких дисков и дискет.

Утилиты используются в различных операционных системах. Программы-утилиты позволяют проверять диски, восстанавливать и преобразовывать информацию на дисках.

12.2. Утилита NDD

Утилита NDD (NORTON DISK DOCTOR) предназначена для исправления сбойных дискет и жестких дисков. Исправление достигается перемещением информации со сбойных участков диска на нормальные. Сбойные участки помечаются меткой BAD BLOCK (плохой блок) и исключаются из дальнейшей работы. После лечения файл становится доступным для обработки, хотя возможна потеря информации.

Утилита NDD имеет значительное количество версий, каждая из которых ориентирована на конкретную операционную систему. Использование варианта утилиты, несоответствующего операционной системе может привести к уничтожению информации. При работе с операционной системой Windows, кроме утилиты NDD, может быть использован ее аналог ScanDisk.

Программа NDD может быть использована на компьютерах различных конфигураций, в частности не полностью совместимых с компьютерами фирмы IBM.

Независимо от версии работа утилиты предусматривает организацию диалога, в процессе которого определяются особенности функционирования. Например, NDD может быть настроена таким образом, что будет запускаться автоматически при загрузки Windows.

Кроме того, диалог обеспечивает управление действиями пользователя и выяснение выбора пользователя при определении варианта лечения диска. В частности возможны следующие варианты: во-первых, выдача сообщений об обнаружении ошибки без исправления; во-вторых, при обнаружении ошибок выдается запрос о необходимости исправления; в-третьих, автоматическое исправление ошибок.

Диалог обеспечивается либо на английском, либо на русском языке.

Утилита NDD выполняет либо полную (INTEGRITY) проверку, либо проверку поверхности (SURFACE). Полная проверка означает, что первоначально проверяются служебные области диска (область загрузки, таблица размещения файлов, таблица разбиения диска) и только затем обеспечивается переход к проверке поверхности. Проверка поверхности предполагает считывание подряд всех записей на диске.

По результатам тестирования формируется отчет. Отчет о результатах тестирования может быть представлен следующими вариантами: во-первых, выдан на экран; во-вторых, организован в виде специального файла; в-третьих, отослан по электронной почте (E-MAIL).

12.3. Программа UNERASE WIZARD

Программа UNERASE WIZARD предназначена для восстановления (RECOVER) удаленных файлов. Данная программа позволяет находить и восстанавливать файлы даже, если их нет в Корзине (RECYCLE BIN).

Для своей работы UNERASE WIZARD использует утилиту NORTON PROTECTION. NORTON PROTECTION представляет собой мощную систему защиты (PROTECTION) удаляемых

файлов от затирания другими файлами. Удаленные файлы сохраняются в доступном для полного восстановления виде.

Программой UNERACE WIZARD предусматривается три варианта поиска восстанавливаемых файлов:

во-первых, ищутся недавно удаленные файлы. Определяются последние 25 удаленных файлов;

во-вторых, предлагаются к восстановлению все защищенные файлы на локальном диске;

в-третьих, восстанавливаются все файлы, соответствующие заданному критерию. В этом случае, программа выдает запрос о критерии отбора. В качестве критерия может служить маска имени, расширение и т.д.

12.4. Утилита SPEED DISK

Утилита SPEED DISK предназначена для упорядочивания размещения файлов на диске. После упорядочивания ускоряется работа с диском.

Процесс упорядочивания состоит в следующем. При записи файлов операционная система записывает файл на первые встреченные свободные участки дисковой памяти. Эти участки могут располагаться в различных местах диска. Если файлы часто модифицируются, то разброс участков увеличивается и скорость обращения к диску уменьшается. Утилита упорядочивает различные участки файла и размещает файл в виде непрерывного участка на дисковой поверхности.

Утилита SPEED DISK может быть настроена на индивидуальную оптимизацию. Индивидуальная оптимизация позволяет обеспечивать выполнение следующих действий:

во-первых, размещение часто используемых файлов в начале диска для ускорения доступа;

во-вторых, проверку правильности записи данных, перемещаемых в ходе оптимизации;

в-третьих, заполнение нулями всех неиспользуемых кластеров диска для безопасности данных;

в-четвертых, изменение внешнего вида карты диска.

Следует учитывать, что работает утилита SPEED DISK достаточно долго (может работать полчаса и даже больше) и не должна прерываться в процессе работы.

В процессе работы утилита анализирует состояние диска и выдает рекомендации. Утилита ориентирована на различные версии операционных систем и допустимо использование исключительно соответствующей версии. Для повышения надежности обработки информации рекомендуется перед обращением к утилите создавать резервную копию диска.

12.5. Утилита SPACE WIZARD (SW 32)

Программа SPACE WIZARD дает возможность выявить файлы, которые являются кандидатами на удаление или сжатие.

Используется программа SPACE WIZARD, в том случае, когда необходимо освободить место на диске. При использовании утилиты SPACE WIZARD перед запуском утилиты SPEED DISK достигается максимальная оптимизация дисковой памяти.

Программой SPACE WIZARD отбираются файлы, имена которых говорят о том, что они резервные или временные. Отобранные файлы предлагается удалить. Удаление файлов выполняется исключительно по команды пользователя. Любой файл может быть исключен пользователем из списка на удаление. Кроме того, в процессе работы утилиты SPACE WIZARD обеспечивается возможность немедленного опустошения корзины RECYCLE BIN.

12.6. Назначение программ архивации

При обработке информации на компьютере возможна порча или потеря информации на магнитных дисках. Это может произойти по следующим причинам:

во-первых, из-за физической порчи магнитного диска;

во-вторых, неправильной корректировки файла;

в-третьих, случайного уничтожения файлов;

в-четвертых, разрушения информации компьютерным вирусом.

Для сохранения информации необходимо иметь либо резервные, либо архивные копии используемых файлов (ARCHIVE). Резервное копирование предусматривает сохранение файлов в неизменном виде. Архивирование предусматривает выполнение операции сжатия (COMPRESSION), т.е. преобразование данных в более компактный вид. Как правило, архивирование используется для длительного хранения информации. Кроме того, создание архива является наиболее удобным, а иногда единственным способом перенести большое количество файлов с одной компьютерной системы на другую.

При создании архивных копий необходимо использовать специально разработанные программы для архивации файлов. Как правило, программы архивации позволяют помещать копии файлов в сжатом виде в архивный файл, извлекать файлы из архива, просматривать оглавление архива и т.д.

Разные программы архивации отличаются алгоритмами сжатия, форматом архивных файлов, интерфейсом, инструментарием, скоростью работы, степенью сжатия файлов при помещении в архив и т.д.

Формируемый архивный файл представляет собой набор из одного или нескольких файлов, помещенных в сжатом виде в единый файл. Архивный файл содержит оглавление, а также код циклического контроля для каждого файла. Оглавление позволяет узнать какие файлы содержатся в архиве. Код циклического контроля (CRC, CYCLIC REDUNDANCY CHECK, циклическая проверка на появление излишка) представляет собой специальную функцию, определяемую по всему содержимому файла. Код циклического контроля составляется таким образом, что изменить файл так, чтобы его код циклического контроля остался неизменным, практически невозможно.

Для сжатия информации могут быть использованы различные алгоритмы. Самый простой алгоритм предполагает представление последовательности одинаковых символов в виде

символа и характеристики его повторения. Например, строку ‘AAAAAAAA’ можно представить в виде 8’A’. Большинство архиваторов использует совокупность различных методов.

Экономия памяти при использовании архивации зависит от типа архивируемого файла. При архивации текстовых файлов экономия составляет примерно 60-70%. При архивации выполнимых файлов экономия составляет примерно 20-30%. В наибольшей степени сжимаются файлы, содержащие видео и аудио информацию. Их размер может уменьшится в десятки раз.

Некоторые программы-архиваторы позволяют указывать пароль при помещении файлов в архив. При этом, помещаемые в архив файлы зашифровываются с помощью этого пароля. Поэтому их нельзя будет извлечь из архива, не указав того же пароля.

Некоторые архиваторы позволяют работать со специальными архивами (CAB-файлами), в которые запакованы дистрибутивы, т.е. установочные пакеты.

12.7. Программа ARJ

Программа ARJ отличается от других архиваторов, тем что обеспечивает более высокую степень сжатия. Кроме того, программа ARJ умеет создавать архивы, располагающиеся на нескольких дискетах. Недостатком архиватора ARJ является более длительное время сжатия, чем у некоторых других архиваторов. Архивные файлы, создаваемые архиватором ARJ имеют расширение ‘.ARJ’. Обращение к некоторым версиям программы ARJ может осуществляется с помощью команды, имеющей формат

ARJ <команда> [<режим>] <имя архива> [<имена файлов>]

Параметры, задаваемые в команде имеют следующее значение.

Параметр <команда> определяет действия выполняемые программой. В частности, допустимы следующие команды:

A – добавление файлов в архив;
E – извлечение файлов из архива;
D – удаление файлов из архива;
L – просмотр оглавления .

Параметр <режим> задает или уточняет требуемые от программы архивации действия.

Параметр «имя архива» задает обрабатываемый архивный файл.

Параметр «имена файлов» задает файлы, включаемые в архив. Если имена файлов не заданы, то подразумеваются все файлы, имеющиеся в текущем каталоге.

Пример:
ARJ A ARXIV1

Данная команда добавляет в архивный файл ARXIV1.ARJ все файлы из каталога, в котором находится программа ARJ.

12.8. Архиватор WINZIP

Архиватор WinZip представляет собой широко распространенный архиватор, работающий в среде Windows. Архивы, обрабатываемые программами WinZip, организованы в виде файлов с расширением '**.ZIP**'. Алгоритмы, используемые программой WinZip, при формировании архивов обеспечивают высокую скорость работы и достаточно большую степень сжатия. Основными особенностями архиватора WinZip являются следующие:

Во-первых, архиватор имеет стандартный интерфейс системы Windows.

Во-вторых, он может работать с архивами, созданными другими архиваторами. Например, с архивами типа '.ARJ'.

В-третьих, в архиваторе предусмотрена возможность создания многотомных архивов.

В-четвертых, имеется специальный инструмент «МАСТЕР» (WIZARD), предназначенный для упрощенного архивирования и разархивирования файлов.

В-пятых, в архиваторе предусмотрена возможность управления степенью сжатия. Для этого в раскрывающемся списке «COMPRESSION» предусмотрены следующие пункты: 'NORMAL' (нормальное сжатие); 'FAST' (более быстрое компрессирование, но с меньшей степенью сжатия); 'SUPER FAST' (очень быстрое сжатие).

В-шестых, при создании архива имеется возможность задания пароля.

В-седьмых, архиватор позволяет сортировать файлы по именам, типам, времени последнего изменения, размеру файла и т.д.

В-восьмых, архиватор реализует технологию 'DRAG AND DROP'. Это значит он позволяет выделить нужные файлы в архиве, взять их мышью и оттащить в любое окно Windows.

12.9. Использование программы архивации BACKUP

При работе в операционной системе Windows 98 для работы с архивами можно применять программу архивации BACKUP. Чтобы запустить программу BACKUP требуется выбрать следующую последовательность пунктов:

START->PROGRAMS->ACCESSORIES->SYSTEM TOOLS ->BACKUP

(ПУСК -> ПРОГРАММЫ -> СТАНДАРТНЫЕ -> СЛУЖЕБНЫЕ -> АРХИВАЦИЯ ДАННЫХ)

Программа BACKUP имеет следующие особенности:

Во-первых, программа использует стандартный интерфейс системы Windows.

Во-вторых, для облегчения работы с архивами можно пользоваться специальными инструментами BACKUP WIZARD (мастер создания архивов) и RESTORE WIZARD (мастер восстановления архивов).

В-третьих, пользователь имеет возможность задать различные условия архивации. Условия архивации позволяют

задать, в частности, следующие особенности: а) определить способы архивации; б) задать режимы контроля правильности выполнения операций с архивом; в) обеспечить защиту от несанкционированного доступа с помощью пароля и т.д.

В-четвертых, набор условий для архивации объединяется под термином BACKUP JOB (задание на архивацию). Заданию на архивацию присваивается имя (NAME THE BACKUP JOB). Имя задания на архивацию используется для повторных архиваций.

В-пятых, допустимо формировать архив на сетевом диске.

12.10. Самораскрывающийся архив

Под самораскрывающимся архивом (SELF-EXTRACTOR) понимается специальным образом организованный архив, который не требует для разархивирования программы-архиватора. Использование самораскрывающихся архивов целесообразно, если программа-архиватор может отсутствовать в месте распаковки. Недостатком использования самораскрывающихся архивов является меньшая степень сжатия, из-за того, что в архив включается программное обеспечение, выполняющее извлечение файлов. Самораскрывающиеся архивы формируются в виде исполнимых файлов с расширением '**.EXE**'. Для создания самораскрывающихся архивов можно использовать программу-архиватор ARJ. В этом случае, задается режим архивации '-JE'. Например, конструкция

ARJ A -JE ARXIV2

создает файл ARXIV2.EXE, который будет являться самораскрывающимся архивом.

Архиватор WinZip не создает самораскрывающиеся архивы непосредственно. Если необходимо создать самораскрывающийся архив, то сначала создается обычный архив, а затем архивные файлы преобразуются в самораскрывающиеся архив. Для преобразования требуемой совокупности архивных файлов в самораскрывающийся архив предназначено окно

SELF-EXTRACTOR PERSONAL EDITION (редактирование самораскрывающихся архивов).

Самораскрывающиеся архивные файлы широко используются при поставках программного обеспечения различными фирмами. Например, браузер INTERNET EXPLORER первоначально представляет собой саморазархивирующейся файл, который после разархивации занимает в полтора раза больший объем дисковой памяти.

Тема 13.

Компьютерные вирусы и защита от них

13.1. Понятие и классификация компьютерных вирусов

Компьютерный вирус представляет собой специально разработанную программу, которая внедряется в компьютерные программы, в системные области дисков, в драйверы, в документы и т.д. При внедрении вирус, как правило, формирует свою копию. Процесс внедрения вируса в другую программу называется заражением.

Как правило, вирус начинает работать после обращения к зараженной программе. Когда зараженная программа начинает работать, то сначала управление получает вирус. После получения управления вирус создает свои копии в других программах, а также выполняет другие действия, предусмотренные его алгоритмом. Затем вирус, как правило, передает управление той программе, в которой он находится. Поэтому внешне работа зараженной программы обычно выглядит так же, как и незараженной.

По степени опасности вирусы можно подразделить на неопасные, опасные и особо опасные.

Неопасными называются вирусы, которые не осуществляют порчи информации. Неопасные вирусы могут выдавать на экран сообщения или рисунки, создавать музыкальные или видеоэффекты.

Опасными называются вирусы, которые портят ограниченный тип данных или делают это эпизодически, например в «черную пятницу».

Особо опасными называются вирусы, которые причиняют значительные разрушения и делают работу на компьютере практически невозможной.

По объектам внедрения вирусы подразделяются на файловые и загрузочные.

Файловые вирусы заражают файлы различных видов. Обычно заражаются исполнимые файлы, т.е. файлы с расширением '.COM' и '.EXE'. Кроме того, могут заражаться документы текстового редактора WORD (файлы с расширением '.DOC') и таблицы табличного процессора EXCEL (файлы с расширением '.XLS'). Не подвержены заражению файлы, не содержащие программ, и, не подлежащие преобразованию в программы. Например, текстовые файлы с расширением '.TXT' и графические файлы с расширением '.BMP'.

Загрузочные вирусы внедряются в начальный сектор дисков, в котором находятся загрузчик операционной системы и системные таблицы. Эти вирусы начинают работу при загрузке компьютера с зараженного диска. Распространяются загрузочные вирусы через дискеты.

13.2. Антивирусные программы

Для защиты от вирусов созданы специальные антивирусные программы, позволяющие выявить вирусы и лечить зараженные файлы и диски. Антивирусные программы в соответствии с выполняемыми ими функциями подразделяются на следующие виды:

Во-первых, программы-детекторы, которые позволяют обнаружить файлы, зараженные одним из известных вирусов. Многие программы-детекторы позволяют лечить зараженные файлы или диски, удаляя из них вирусы. Наиболее известными программами-детекторами являются программы AVP (ANTIVIRAL TOOLKIT PRO), DR.WEB и AIDSTEST.

Во-вторых, программы-ревизоры. Программы-ревизоры запоминают сведения о состоянии файлов и системных областей дисков, а при последующих запусках сравнивают их состояние с исходным. При выявлении несоответствий об этом сообщается пользователю. Примером программы ревизора

являются отдельные модули антивирусного комплекса NORTON ANTIVIRUS.

В-третьих, программы-сторожа, которые располагаются резидентно в оперативной памяти компьютера и проверяют на наличие вирусов запускаемые файлы и вставляемые в дисковод дискеты. При наличии вируса об этом сообщается пользователю. Программы-сторожа позволяют обнаружить многие вирусы на самой ранней стадии, когда вирус не успел размножиться и что-либо испортить. Примером программы-сторожа являются отдельные модули антивирусных комплексов NORTON ANTIVIRUS и модуль MONITO системы AVP.

13.3. Методы обнаружения и удаления компьютерных вирусов

Для того, чтобы уменьшить вероятность заражения вирусом и обнаружить заражение рекомендуется выполнять следующие действия.

Во-первых, все принесенные дискеты перед использованием следует проверять на наличие вирусов с помощью программ-детекторов.

Во-вторых, если полученные файлы содержатся в архивах, то следует извлекать их из архивов и проверять программами-детекторами сразу же после этого.

В-третьих, целесообразно обеспечивать периодическую проверку дисков на наличие вирусов.

В-четвертых, не следует выполнять загрузку операционной системы с непроверенных дискет.

В-пятых, рекомендуется периодически обновлять имеющиеся версии программ-детекторов.

Можно выделить следующие признаки заражения компьютера вирусом:

во-первых, программа-детектор сообщает о наличии известного ей вируса;

во-вторых, программа-ревизор сообщает об изменении файлов, которые не должны изменяться;

в-третьих, программа-ревизор сообщает об изменении главной загрузочной записи жесткого диска;

в-четвертых, программа-сторож сообщает о том ,что какая-то программа желает форматировать жесткий диск;

в-пятых, на экран начинают выводится посторонние сообщения, символы и т.д.;

в-шестых, некоторые файлы оказываются испорченными;

в-седьмых, некоторые программы перестают работать.

Лечение вирусов обеспечивается с помощью антивирусных программ и комплексов. В крайнем случае зараженный файл удаляют и восстанавливают на основе резервных копий.

13.4. Защита от вирусов при работе в сети INTERNET

При работе в сети INTERNET защита от вирусов имеет следующие особенности.

Во-первых, в сети INTERNET имеется особый вид вируса, который называют «троянскими конями» или «троянцами».Основная цель вирусов данного типа проникновение в компьютер и чтение конфиденциальной информации. Например, паролей, которые используются для выписки счетов за оказанные услуги.

Вирус «троянский конь» может быть заложен в распространяемую по сети полезную программу. Например, программу ускорения работы интернетовских приложений или игровую программу.

Вирусы типа «троянский конь» не заинтересованы в том, чтобы себя обнаружить. Данный вирус не выдает на печать сообщений и не уничтожает без надобности информацию. Обнаружить троянский вирус очень сложно. Данные вирусы, как правило, обнаруживают по индивидуальным особенностям каждого вируса. Для обнаружения вирусов типа «троянский конь» предназначена программа ‘The Cleaner’ (чистильщик). Данная программа знает индивидуальные особенности нескольких де-

сятков троянцев. Программа 'The Cleaner' сканирует диски и обеспечивает поиск всех известных программе вирусов.

Во-вторых, при получении почтовых сообщений, в том случае когда поступает файл с расширением '.DOC', почтовые программы автоматически вызывают текстовый редактор WORD. В макросах файла '.DOC' может находится вирус и если не предусмотреть защиту от вирусов, то можно заразить компьютер.

В-третьих, для сети INTERNET существует специальный вид резидентных антивирусных программ, которые называются «МОНИТОРЫ». Программы-мониторы проверяют все поступающие по сети файлы. Найдя вирус «МОНИТОР» может:

а) запретить кому бы то ни было доступ к зараженному объекту;

б) автоматически начать лечение зараженного объекта;

в) автоматически удалить зараженный объект.

При обнаружении почтового отправления, содержащего вирус, программа-монитор указывает: отправителя письма, тему письма и имя вложенного файла, содержащего вирус.

Примером антивирусного пакета, предназначенного для работы в сети, является пакет AVP (ANTIVIRAL TOOLKIT PRO, антивирусный набор инструментов).

Практикум

Особенности организации практических занятий при изучении дисциплины «Системное программное обеспечение»

Эффективное проведение практических занятий предусматривает учет ряда следующих особенностей изучения дисциплины:

Во-первых, изучаемый материал ориентирован на международные стандарты. Поэтому, при изучении дисциплины целесообразно широкое использование информации, получаемой из передовых стран.

Во-вторых, дисциплины, связанные с компьютерными технологиями развиваются быстрыми темпами. Поэтому, приобретение устойчивых профессиональных знаний предусматривает использование некоторых аспектов технологии открытого образования.

В-третьих, в соответствии с технологией открытого образования изучение материала предполагает значительную самостоятельную работу.

В-четвертых, в соответствии с технологией открытого образования изучение материала предполагает широкое использование сети INTERNET.

В-пятых, в соответствии с технологией открытого образования контроль изучения материала осуществляется с помощью системы тестов.

В-шестых, успешное освоение дисциплины позволяет достичь высокого профессионального уровня в направлении использования системного программного обеспечения. Рассматриваемые модели получили распространение как в США, так и в других развитых странах. При изучении дисциплины используются самые современные программные продукты. Таким образом, получаемые знания конкурентоспособны как на Российском, так и на международном уровне. Следова-

тельно, у значительного числа студентов имеется стимул к качественному усвоению изучаемого материала.

В-седьмых, практически все студенты имеют в своем распоряжении компьютеры и, следовательно, у них имеются широкие возможности для проведения самостоятельных исследований и творческого выполнения заданных работ.

Исходя из вышеизложенного, значительная часть практических занятий проходит в режиме контролируемой индивидуальной работы студентов.

Самостоятельное изучение материала дисциплины предусматривает следующие этапы работы:

Первый этап. Выдача индивидуальных заданий на выполнение лабораторных работ.

Второй этап. Контроль, за ходом выполнения заданий, и консультации по возникающим проблемам.

Третий этап. Анализ результатов выполненных работ и защита лабораторных работ.

Предмет "СИСТЕМНОЕ ПО" предполагает выполнение большого объема самостоятельных исследований. Требуется: во-первых, изучить учебную литературу; во-вторых, приобрести практические навыки в работе с технической документацией; в-третьих, разработать и отладить значительное число программ

Занятие 1. Программное обеспечение и его классификация

При проведении занятия рекомендуется рассмотреть следующие темы:

1) Виды системного программного обеспечения;
2) Свойство «Совместимость» (COMPATIBLE) и его значение для системного программного обеспечения;
3) Свойство «Переносимость» (PORTABILITY) и его значение для системного программного обеспечения;
4) Свойство «Масштабируемость» (SCALABILITY) и его значение для системного программного обеспечения;

5) Свойство «Безопасность» (SECURITY) и его значение для системного программного обеспечения;

6) Понятие «Компонент» (COMPONENT) в системном программном обеспечении.

Занятие 2. Интегрированная среда разработки

При проведении занятия рекомендуется рассмотреть следующие темы:

– Основные составляющие части интегрированной среды разработки (INTEGRATED DEVELOPMENT ENVIROMENT, IDE);
– Состав прикладного программного приложения;
– Создание прикладного программного приложения.

Занятие 3. Системные компоненты общего назначения

При проведении занятия рекомендуется решить и разобрать решение следующих задач:

Задача 1.

Имеется документ «классификатор-ценник малоценных предметов». В документе имеются следующие реквизиты:

Во-первых, номенклатурный номер –5 символов;

Во-вторых, наименование предмета –20 символов;

В-третьих, цена предмета – 8 цифр, из них две цифры определяют дробную часть.

Разработать модуль, обеспечивающий ввод информации и формирование массива записей.

Задача 2.

Имеется документ «Ведомость норм отходов на детали». В документе имеются следующие реквизиты:

Во-первых, шифр детали –10 символов;

Во-вторых, шифр материала –10 символов;

В-третьих, шифр отхода – 4 цифры;

В-четвертых, норма отхода – 8 цифр, из них четыре цифры определяют дробную часть.

Разработать модуль, выдачу на экран для просмотра строк документа, норма отхода, у которых выше средней арифметической по документу.

Задача 3.

Имеется документ "Ведомость норм отходов на детали". В документе имеются следующие реквизиты:

Во-первых, шифр детали –10 символов;

Во-вторых, шифр материала –10 символов;

В-третьих, шифр отхода – 4 цифры;

В-четвертых, норма отхода – 8 цифр, из них четыре цифры определяют дробную часть.

Разработать модуль, обеспечивающий определение значение реквизита "шифр материала", соответствующее максимальному значению реквизита "норма отхода".

Задача 4.

Имеются документы, представляющее собой товарные чеки. В документе имеются следующие реквизиты:

Во-первых, наименование товара – 20 символов;

Во-вторых, количество товара – 3 цифры;

В-третьих, цена товара – 8 цифр, из них две цифры определяют дробную часть;

В-четвертых, сумма – 8 цифр, из них две цифры определяют дробную часть.

Разработать модуль, обеспечивающий определение общей суммы проданных товаров.

Задача 5.

Имеется документ "Ведомость компенсации потерь". В документе имеются следующие реквизиты:

Во-первых, шифр цеха –5 символов;

Во-вторых, шифр участка –5 символов;

В-третьих, шифр детали – 10 символов;

В-четвертых, процент технологических потерь – 5 цифр, из них три цифры определяют дробную часть.

Разработать модуль, обеспечивающий определение значение реквизита «шифр участка», соответствующее максимальному значению реквизита «процент технологических потерь».

Задача 6.

Имеется документ «Ведомость компенсации потерь». В документе имеются следующие реквизиты:

Во-первых, шифр цеха –5 символов;

Во-вторых, шифр участка –5 символов;

В-третьих, шифр детали – 10 символов;

В-четвертых, процент технологических потерь – 5 цифр, из них три цифры определяют дробную часть.

Разработать модуль, обеспечивающий выдачу на экран для просмотра строки документа, процент технологических потерь, у которых выше средней арифметической по документу.

Задача 7.

Имеется документ «карточка обследования студентов», который имеет следующую структуру:

Факультет – 5 символов;

Курс – 1 цифра;

Пол (мужской, женский) – 1 символ;

Средний доход в семье – 7 цифр.

Разработать модуль, обеспечивающий определение среднего дохода в семье у девушек первого курса.

Задача 8.

Имеется типизированный файл с внешним именем «CHECK», который содержит информацию по товарным чекам. Записи файла определены следующим образом:

Во-первых, название магазина – STRING[20];

Во-вторых, номер чека – STRING[4];

В-третьих, наименование товара – STRING[20];

В-четвертых, количество товара – INTEGER;

В-пятых, цена товара – REAL;

В-шестых, сумма – REAL;

В-седьмых, дата продажи –STRING[6].

Требуется разработать для среды DELPHI модуль, обеспечивающий выдачу на экран записей файла, с ценой в заданном интервале. Для выдачи информации сформировать компоненту TstringGrid. Предусмотреть задание заголовков выдаваемой таблицы на русском языке. Значения цены, определяющие границы интервала вводятся с клавиатуры.

Задача 9.

Имеется типизированный файл с внешним именем «UNPAID», который содержит информацию по неоплаченным материалам. Записи файла определены следующим образом:

Во-первых, дата – STRING[4]; { Строка цифровых символов. Первые два символа определяют месяц.}

Во-вторых, номер склада – STRING[2]; {Строка цифровых символов. Всего десять складов с номерами от 1 до 10}

В-третьих, шифр поставщика – STRING[10];

В-четвертых, номер приходного ордера – INTEGER;

В-пятых, код материала – STRING[10];

В-шестых, сумма – REAL.

Требуется разработать для среды DELPHI модуль, обеспечивающий выдачу на экран информацию, относящуюся к заданному материалу. Информация выдается в виде таблицы следующего вида:

Неоплаченный материал XXX

Код склада	1-ый квартал	2-ой квартал	3-ий квартал	4-ый квартал
1				
2				
…				
10				

Для выдачи информации сформировать компоненту TstringGrid. Предусмотреть задание заголовков выдаваемой таблицы на русском языке. Задаваемое значение кода материала вводится с клавиатуры**.**

Задача 10.

Имеется типизированный файл с внешним именем "INSPECT", который содержит информацию по обследованию студентов. Записи файла определены следующим образом:

Во-первых, факультет – STRING[5];

Во-вторых, курс – INTEGER;

В-третьих, возраст – INTEGER;

В-четвертых, пол (мужской, женский) – STRING[1];

В-пятых, время, на дорогу в минутах – INTEGER;

В-шестых, средний доход в семье – REAL;

В-седьмых, средний балл – REAL.

Требуется разработать для среды DELPHI модуль, обеспечивающий выдачу на экран таблицы, показывающей распределение студентов в зависимости от среднего дохода в семье. Информация выдается в виде таблицы следующего вида:

Распределение студентов в зависимости
от среднего дохода в семье

	Средний доход в семье до 2000 рублей	Средний доход в семье от 2000 рублей до 5000 рублей.	Средний доход в семье от 5000 рублей до 10000 рублей	Средний доход в семье 10000 рублей и выше
Количество студентов				

Для выдачи информации сформировать компоненту TstringGrid. Предусмотреть задание заголовков выдаваемой таблицы на русском языке.

Задача 11.

Имеется типизированный файл с внешним именем "REQUIRE", который содержит информацию по потребности материалах на текущий период. Записи файла определены следующим образом:

Во-первых, дата – STRING[4]; { Строка цифровых символов. Первые два символа определяют месяц.}

Во-вторых, номер цеха – STRING[2]; {Строка цифровых символов. Всего десять цехов с номерами от 1 до 10}

В-третьих, код материала – STRING[10];

В-четвертых, потребность в материалах – REAL.

Требуется разработать для среды DELPHI модуль, обеспечивающий выдачу на экран информацию, относящуюся к заданному материалу. Информация выдается в виде таблицы следующего вида:

Потребность в материале XXX

Код цеха	1-ый квартал	2-ой квартал	3-ий квартал	4-ый квартал
1				
2				
…				
10				

Для выдачи информации сформировать компоненту TstringGrid. Предусмотреть задание заголовков выдаваемой таблицы на русском языке. Задаваемое значение кода материала вводится с клавиатуры.

Задача 12.

Имеется типизированный файл с внешним именем “SUPPLY”, который содержит нормы переходящих запасов материалов. Записи файла определены следующим образом:

Во-первых, номер склада – STRING[2];

Во-вторых, код материала – STRING[10];

В-третьих, норма переходящего запаса в днях – INTEGER.

Требуется разработать для среды DELPHI модуль, обеспечивающий выдачу на экран информацию, относящуюся к заданному складу. Информация выдается в виде таблицы следующего вида:

Распределение материалов по нормам
переходящего запаса для склада XXX

	До 100 дней	От 100 дней до 200 дней	От 200 дней до 300 дней	300 дней и выше
Количество материалов				

Для выдачи информации сформировать компоненту TstringGrid. Предусмотреть задание заголовков выдаваемой таблицы на русском языке. Задаваемое значение кода склада вводится с клавиатуры.

Занятие 4. Системное программное обеспечение, предназначенное для реализации COM-технология

При проведении занятия рекомендуется рассмотреть следующие темы:

1. Основные понятия COM-технологий;
2. Идентификаторы, используемые в COM-технологии;
3. Технология DCOM;
4. Технология CORBA.

Задание на лабораторную работу № 1

Тема лабораторной работы «Использование системных компонент общего назначения». Целью выполнения лабораторной работы является изучение компонент общего назначения. Лабораторная работа выполняется с использованием различных систем программирования. Компоненты, предназначенные для обработки экономической информации в оперативной памяти исследуются на примере компонент системы Visual Basic. Компоненты, используемые для обработки файлов исследуются на примере компонент системы Delphi. Сравнительный анализ использования компонент в различных системах позволяет отработать технологию перехода на вновь появляющиеся системы программирования. Исследование включает следующие части:

Первая часть. Особенности организации и использования компонент общего назначения. (В данной части требуется дать краткую характеристику компонент; рассмотреть назначение интегрированной среды разработки; описать свойства, события и методы, используемые в компонентах. Характеристика дается в сравнительном анализе различных систем, в частности Visual Basic, Delphi).

Вторая часть. Компоненты используемые в задачах обработки экономической информации. В данной части рассматриваются компоненты системы Visual Basic.

Третья часть. Обработка экономической информации с использованием файлов. В данной части рассматриваются компоненты системы Delphi.

Выполненное исследование должно подтверждаться разработанным и отлаженным программным обеспечением. Рекомендуется разработать следующие программы с использованием технологии объектно-ориентированного программирования.

Задача 1.

Имеется документ «классификатор-ценник малоценных предметов». В документе имеются следующие реквизиты:

Во-первых, номенклатурный номер –5 символов;

Во-вторых, наименование предмета –20 символов;

В-третьих, цена предмета – 8 цифр, из них две цифры определяют дробную часть.

Разработать программу ввода информации и формирования массива записей.

Задача 2.

Имеется документ, содержащий информацию о движении малоценных предметов. В документе имеются следующие реквизиты:

Во-первых, дата – 6 символов;

Во-вторых, код операции – 2 цифры;

В-третьих, код цеха получателя – 2 цифры;

В-четвертых, код цеха отправителя – 2 цифры;

В-пятых, код предмета – 6 цифр;

В-шестых, количество – 7 цифр.

Разработать программу ввода информации из документов и формирование типизированного файла.

Задача 3.

Имеется документ «Классификатор ценник», содержащий следующие поля:

код предмета – 6 цифр;

наименование предмета – 40 символов;

единица измерения – 3 символа;

размер – 20 символов;

цена – 7.2 цифр.

Разработать программу определения статистической характеристики «МОДА» для реквизита «Единица измерения». «МОДА» представляет собой наиболее часто встречающийся вариант. Распечатать строки документа, единица измерения, у которых не соответствует статистической характеристике «Мода».

Задача 4.

Имеется типизированный файл, содержащий информацию по документу «наряд на выполненную работу». Записи файла имеют следующую структуру:

Во-первых, шифр наряда – 5 цифр;

Во-вторых, номер цеха – 2 символа;

В-третьих, номер участка – 2 символа;

В-четвертых, количество изготовлено – 4 цифры;

В-пятых, количество принято – 4 цифры.

Разработать программу определения линейного коэффициента корреляции между реквизитами «Количество изготовлено» и «Количество принято». Для расчета линейного коэффициента корреляции рекомендуется использовать следующую зависимость:

$$R = \frac{\sum (X - \bar{X}) * (Y - \bar{Y})}{\sqrt{\sum (X - \bar{X})^2 * \sum (Y - \bar{Y})^2}}$$

где X– текущее значение реквизита «количество изготовлено»;

–

X – среднее арифметическое значение реквизита «количество изготовлено»;

Y – текущее значение реквизита «количество принято»;

$\bar{Y}$ – среднее арифметическое значение реквизита «количество принято».

Задача 5.

Имеется документ «наряд на выполненную работу», который имеет следующую структуру:

Шифр наряда – 5 цифр;

Номер цеха – 2 символа;

Номер участка – 2 символа;

Количество изготовлено – 4 цифры;

Количество принято – 4 цифры.

Разработать программу определения статистической характеристики «Медиана» по реквизиту «Количество принято». Медианой называется значение признака, приходящееся на середину упорядоченной совокупности. Если в совокупности четное число единиц, то медиана равна средней арифметической из двух средних значений вариантов.

Задача 6.

Имеется типизированный файл, который содержит следующую информацию об остатках малоценных предметов:

Дата – 4 символа;

Код цеха – 2 цифры;

Код предмета – 6 цифр;

Количество на конец периода – 7 цифр.

Записи файла упорядочены по возрастанию реквизита «код предмета».

Требуется разработать программу, которая обеспечивает выдачу на экран содержимого файла. Кроме того, требуется сформировать управляющий элемент Check Box и проверить его состояние. При значении равном единице необходимо выполнить нахождение итогов для каждого предмета по реквизиту «количество на конец периода». Итоговые значения

должны выдаваться после текущих записей, относящихся к рассматриваемому предмету.

Примечание 1. Лабораторная работа выполняется группой по два человека.

Примечание 2. Решение перечисленных задач предусматривает следующие действия:

Во-первых, проектирование экрана с указанием имени каждого формируемого объекта;

Во-вторых, определение событий, которые требуются для решения задачи;

В-третьих, разработку методов, обеспечивающих обработку заданных событий.

Оформление лабораторной работы № 1

Отчет состоит из теоретического описания исследования и практической реализации:

Исследовательская часть.

Тексты программ и распечатки экранов.

Исследовательская часть должна включать описание особенностей реализации в системах Visual Basic и Delphi и ссылки на приложения, в которых практически проверены теоретические материалы.

Структура отчета по лабораторной работе.

1. Титульный лист.

2.Оглавление.

Особенности организации и использования компонент в системах Visual Basic и DELPHI.

Компоненты, используемые в задачах обработки экономической информации.

5. Обработка экономической информации с использованием файлов.

6. Литература.

7. Тексты программ.

8. Распечатки экранов, подтверждающих выполнение.

Задание на лабораторную работу № 2

Тема лабораторной работы «Системные компоненты, предназначенные для работы с использованием COM-технологий и программированием в сети INTERNET/INTRANET»

Целью выполнения лабораторной работы является изучение компонент, предназначенных для работы с использованием COM-технологиями и программированием в сети INTERNET/INTRANET.

Исследование включает следующие части:

Первая часть. Краткая характеристика и назначение COM технологии, DCOM технологии, CORBA технологии и их разновидностей.

Вторая часть. Программирование с использованием сети INTERNET/INTRANET.

Отчет по лабораторной работе включает теоретическое рассмотрение указанных вопросов, с примерами практической реализации.

Тест

1. ***Можно выделить следующие технологии программирования:***
   а) автокорреляция;
   б) аналитическое выравнивание;
   в) структурное программирование;
   г) модульное программирование;
   д) объектно-ориентированное программирование;
   е) com-технологии;
   ж) математическое программирование;
   з) программирование с использованием сети internet;
   и) последовательное элиминирование.

2. ***- это программирование, основанное на использовании канонических структур:***
   а) аналитическое выравнивание;
   б) структурное программирование;
   в) модульное программирование;
   г) объектно-ориентированное программирование;
   д) математическое программирование.

3. ***- это технология программирования, обеспечивающая разбиение единой программы на совокупность модулей примерно одного размера***
   а) аналитическое выравнивание;
   б) структурное программирование;
   в) модульное программирование;
   г) объектно-ориентированное программирование;
   д) математическое программирование.

4. ***- это технология программирования, предусматривающую формирование программы на основе заранее подготовленных объектов:***
   а) аналитическое выравнивание;
   б) структурное программирование;

в) модульное программирование;
г) объектно-ориентированное программирование;
д) математическое программирование.

5. ***Под ... понимается способ программирования, программирования, основанный на использовании иерархии абстрактных модульных типов, которые называются объектами:***
   а) математическим программированием;
   б) системой программирования;
   в) архиватором;
   г) объектно-ориентированным программированием;
   д) драйвером.

6. ***Объект включает в себя:***
   а) компьютерные игры;
   б) данные, защищенные от внешних воздействий;
   в) документацию;
   г) локальные процедуры и функции, предназначенные для обработки данных, входящих в объект;
   д) таблицы случайных чисел.

7. ***Под ... понимается действие, которое связано с объектом:***
   а) событием;
   б) системой программирования;
   в) классом;
   г) свойством;
   д) методом.

8. ***Классы могут быть подразделены на следующие группы***
   а) класс тысяч;
   б) базовые классы;
   в) класс единиц;
   г) классы определенные пользователем;
   д) класс миллионов.

9. ***Для резервирования памяти под динамические переменные используется специальная область оперативной памяти, называемая ...***
   а) транслятором;
   б) хипом;
   в) операционной системой.

10. ***... представляет собой адрес байта оперативной памяти, начиная с которого располагается динамическая переменная***
   а) табличный процессор;
   б) метод;
   в) факториал;
   г) ссылка (указатель) .

11. ***Признаком типа указатель является символ***
   а) ::= ;
   б) ^ ;
   в) ? ;
   г) \ ;
   д) !

12. ***Выделение памяти для динамической переменной выполняется при помощи процедуры ...***
   а) Create;
   б) New;
   в) Dispose;
   г) Freemem;
   д) Append.

13. ***Для освобождения памяти, выделенной динамической переменной используется процедура ...***
   а) Free;
   б) Dispose;
   в) Delete;
   г) New;
   д) FreePointer.

14. IP-адрес состоит из следующих частей

а) частного адреса;
б) адреса сети;
в) центрального адреса;
г) адреса хоста;
д) случайный адрес.

15. Под ... понимается текст, представленный в виде ассоциативно связанных блоков

а) текстовым редактором;
б) файлом;
в) массивом;
г) гипертекстом;
д) ассоциацией.

16. Можно выделить следующие протоколы

а) PDU;
б) TCP/IP;
в) PCT;
г) FTP;
д) UNIX.

17. Гипертекстовые документы описываются на специальном языке ...

а) UNIX;
б) LINUX;
в) VBA;
г) HTML;
д) DELPHI.

18. При создании HTML-документа, могут быть использованы следующие теги

а) HTTP;
б) HTML;
в) A;
г) Z;
д) BODY;
е) UNIX.

19. Технология COM является стандартом корпорации ...

а) IBM;
б) NetsCape;
в) General Motors;
г) MicroSoft.

20. При реализации COM-технологии допустимы следующие особенности

а) в качестве сервера можно использовать сотовый телефон;
б) клиент и сервер могут находится на компьютерах, расположенных в разных странах;
в) компьютеры, на которых находятся программы, могут быть разного типа;
г) технология COM, может работать на выключенных компьютерах;
д) обе программы могут быть написаны на разных языках программирования;
е) данные программы могут исполняться под управлением разных операционных систем;
ж) для технологии COM, программное обеспечение не обязательно.

21. Идентификатор ... представляет собой программно генерируемую 16 байтовую величину уникальную во времени и пространстве.

а) NetsCape;
б) MicroSoft;
в) GUID;
г) IBM.

22. Технология ... представляет собой разновидность COM-технологии, которая используется при выполнении распределенных приложений в сети.

а) OOP;
б) Windows Registry;
в) DCOM;
г) ComArray.

23. Технология CORBA разрабатывается отраслевым комитетом ...

а) IBM;
б) NetsCape;
в) MicroSoft;
г) OMG.

24. Под ... понимается программное средство, поддерживающее процессы жизненного цикла ПО, включая анализ требований к системе, проектирование прикладного ПО и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, управление конфигурацией ПО и управление проектом, а также другие процессы.

а) ООР;
б) COM-технологием;
в) CORBA;
г) Case-средством.

25. Case-средствам присущи следующие основные особенности:

а) проектные материалы хранятся в специальных портфелях (кейсах);
б) наличие мощных графических средств для описания и документирования системы;
в) þ интеграция отдельных компонентов Case-средств;
г) Case-технологии, основаны на использовании оператора Case, обеспечивающего множественное ветвление;
д) использование специальным образом организованного хранилища проектных метаданных;
е) для Case-средств, программное обеспечение не обязательно.

26. В соответствии со стандартами IEEE процесс внедрения Case-средств включает следующие этапы:

а) кодирование;
б) определение потребности в Case-средствах;
в) оценка и выбор Case-средств;
г) выполнение пилотного проекта;
д) практическое внедрение Case-средств;
е) сопровождение.

Глоссарий

ActiveX компоненты – 32 разрядные объекты, содержащие коды и данные. Данные объекты могут создаваться с помощью различных средств разработки, например Visual C++ или Visual Basic. Основным преимуществом ActiveX компонентов является их огромное количество, т.к. их разработкой занимаются фирмы и отдельные программисты.

ADO (ACTIVE DATA OBJECTS, ActiveX DATA OBJECTS, активные объекты данных) – технология, позволяющая помещать программные средства работы с источниками данных непосредственно на активные серверные станции. Например, можно поместить специальный элемент DATAGRID и обеспечить просмотр информации, расположенной в базе данных в табличной форме. Технология ADO обеспечивает стандартизированный COM-интерфейс к ODBC-функциям.

ALIAS – альтернативное короткое имя, предназначенное для ссылки на таблицу базы данных.

API (APPLICATION PROGRAMMING INTERFACE) – набор функций, обеспечивающих доступ к сервисам операционной системы.

APPLET (аплет) – небольшая программа, разрабатываемая для передачи по сети INTERNET. Аплеты автоматически устанавливаются и запускаются как часть WEB-документа. Аплеты принципиально отличаются от приложений (application) по структуре и по некоторым ключевым областям.

APPLICATION – программное приложение, обеспечивающее выполнение прикладной задачи. Как правило, прикладное программное приложение представляет собой совокупность файлов различных типов, которые объединяются в единую программу.

CASE-технологии – высокопроизводительные, ресурсосберегающие технологи, предназначенные для создания комплексов программ высокого качества и надежности.

CLASS (класс) – образец, предназначенный для создания работающих объектов определенного типа.

COM (COMPONENT OBJECT MODEL, МОДЕЛЬ КОМПОНЕНТНОГО ОБЪЕКТА) – стандарт, при помощи которого приложения могут использовать объекты, расположенные в других системах. Данный стандарт позволяет представлять объекты в виде исполняемых двоичных файлов. Стандарт COM разработан фирмой MICROSOFT и очень популярен на платформах Windows. Технология COM широко используется при программировании в сети INTERNET.

CUA (COMMON USER ACCESS, общий пользовательский доступ) – международный стандарт на разработку пользовательского интерфейса.

DHTML (DYNAMIC HYPERTEXT MARKUP LANGUAGE) – является расширением языка HTML. Все WEB-браузеры используют HTML для расшифровки информации, которая поступает пользователю через INTERNET. DHTML позволяет выполнять программную обработку получаемой информации.

DLL (DYNAMIC LINK LIBRARY) –внешняя библиотека, обеспечивающая работу в различных объектно-ориентированных системах.

ENCAPSULATION (инкапсуляция) – основополагающий принцип объектно-ориентированного программирования, который предусматривает объединение данных и процедур обработки в единый тип, называемый объектом.

EVENT (событие) –действие, связанное с объектом. Событие может быть инициировано пользователем, вызвано программой или определено системой.

GUI (GRAPHICAL USER UNTERFACE) – набор форм и объектов, которые позволяют пользователю просматривать и обрабатывать информацию. GUI является частью прикладных программных приложений, расположенной между пользователем и процедурами обработки информации.

HTML (HYPER TEXT MARKUP LANGUAGE, язык разметки гипертекстов) – является инструментальным программным обеспечением, использующим технологию гипертекста при создании разнообразных документов. Главной задачей этого языка является придание документам стандартной для глобального соединения формы.

IDE (INTEGRATED DEVELOPMENT ENVIROMENT) – интегрированная среда разработки, которая представляет собой совокупность всех инструментов, необходимых для разработки, отладки и выполнения объектно-ориентированных приложений.

IID (INTERFACE IDENTIFIER, уникальный идентификатор интерфейса) – идентификатор, присваиваемый COM-объекту при использовании COM -технологий.

IIS application (INTERNET INFORMATION SERVER APPLICATION, IIS приложения) – прикладные приложения, предназначенные для выполнения на сервере INTERNET. IIS сервер в первую очередь используется для трансмиссий страниц HTML при использовании HTTP протокола. IIS приложения вызываются клиентскими компьютерами по сети Internet. IIS приложения могут быть разработаны на различных языках программирования, например на C++, VBScript и т.д..

INHERITANCE (наследование) – основополагающий принцип объектно-ориентированного программирования, который определяет способность порожденного класса сохранять характеристики, присущие родительскому классу.

METHOD (метод) – функция или процедура, которая будет управлять работой объекта.

ODBC (OPEN DATABASE CONNECTIVE, открытые средства связи с базами данных) – технология, предназначенная для организации взаимодействия с реляционными СУБД. Позволяет работать с базами данных любого типа. С помощью ODBC можно интегрировать данные, полученные из баз данных Access и Visual FoxPro, таблиц SQL Server и других источников.

OOP (OBJECT ORIENTED PROGRAMMING, объектное ориентированное программирование) – способ программмирования основанный на использовании абстрактных модульных типов, которые называются объектами.

POLYMORPHISM (полиморфизм) – основополагающий принцип объектно-ориентированного программирования, под которым понимается присвоение действию единого имени при допустимости различных вариантов его реализации.

PROPERTY (свойство) – характеристика, с помощью которой описывается внешний вид и функционирование объекта.

TCP/IP (TRANSFER CONTROL PROTOCOL / INTERNET PROTOCOL, TRANSPORT CONTROL PROTOCOL / INTERNET PROTOCOL) – базовый протокол сети INTERNET. Протокол IP представляет собой протокол, описывающий формат пакета данных, передаваемого по сети. Данный протокол определяет, где в передаваемом потоке располагается адрес и служебная информация, а где сами передаваемые данные. Протокол TCP предназначен для контроля целостности передаваемой информации.

PROTOCOL (протокол) – набор правил, который описывает порядок передачи информации и некоторые аспекты ее преобразования при работе в сети INTERNET.

URL (UNIFORM RESOURCE LOCATOR, UNIVERSAL RESOURCE LOCATOR, USER RESOURCES LOCATOR) – уникальный адрес, который используется в гипертекстовых ссылках и обеспечивает доступ к распределенным ресурсам сети.

VBScript – версия языка Visual Basic, предназначенная для работы в сети INTERNET. VBScript может быть использован для разработки прикладных приложений, обрабатывающих информацию, получаемую при работе с Internet Explorer.

Wizard (мастер) – приложение, или один из инструментов приложения, предназначенные для помощи пользователю при решении трудных задач.

XML (eXtended Markup Language) – представляет собой дальнейшее развитие стандарта HTML. Позволяет реализовать объектный подход к созданию Internet-контента и структурированную передачу и обработку данных через Internet.

4GL (FOURTH-GENERATION LANGUAGE) – языки программирования четвертого поколения. Данные языки предназначены для быстрой разработки проектов в определенной предметной области. К языкам четвертого поколения относят такие языки как Visual FoxPro, XML.

Список литературы

Основная литература:

1. Архангельский, А.Я. Программирование в DELPHI 6. – М. : Бином, 2002.
2. Благодатских, В.А. и др. Экономика, разработка и использование программного обеспечения ЭВМ. – М. : Финансы и статистика,1995.
3. Браун, С. Visual Basic 6:учебный курс. – СПб. : Питер, 1999.
4. Боэм, Б. Инженерное проектирование ПО. – М. : Радио, 1985.
5. Браун, С. Visual Basic 6: учебный курс. – СПб. : Питер, 1999.
6. Гетц, К, Джилберт, М. Программирование в MS OFFICE. Полное руководство по VBA., "BHV". – Киев, 1999. – 768 с.
7. Дайсон П. WINDOWS 98. – М., 1999.
8. Использование Visual FoxPro 6. Специальное издание. – М. : Вильямс, 2000. – 928 с.
9. Левин Александр. Самоучитель полезных программ. – М, 1999.
10. Липаев, В.В. Надежность программных средств. М. : СИНТЕГ, 1998.
11. Липаев, В.В. Документирование и управление конфигурацией программных средств. Методы и стандарты. – М. : Синтег,1998.
12. Петруцос, Э., Хау, К. Visual BASIC 6 и VBA для профессионалов. – СПб. : ПИТЕР, 2000.
13. Проектирование пользовательского интерфейса на персональных компьютерах. Стандарты фирмы IBM. – М. : DBS LTD,1993 г.
14. Смирнов, А.А. Прикладное программное обеспечение. Учебное пособие. – М. : МЭСИ 2001.
15. Фаронов, В.В., Шумаков, П.В. Delphi 4. Руководство разработчика баз данных. Учебный курс. – М. : Нолидж, 1999.

16. Фигурнов, В.Э. IBM PC для пользователя. – М., 1998.
17. Фаронов, В.В. Delphi 5. Учебный курс. – М. : Нолидж, 2000.
18. Фигурнов, В.Э. IBM PC для пользователя. – М. :ИНФРА-М, 2000.
19. ISO 12207:1995. Процессы жизненного цикла программных средств. (Международный стандарт).
20. ANSI/IEEE. 983-1986. Руководство по обеспечению качества программных средств. (Международный стандарт).
21. ISO 9126:1991 Оценка программного продукта. Характеристика качества и руководство по их применению. (Международный стандарт).
22. ГОСТ 28806-90. Качество программных средств. Термины и определения.

Дополнительная литература:

1. Левин А. Самоучитель полезных программ. – М. : Нолидж,1999
2. Учебный курс. – М. : Нолидж,2000
3. Фаронов, В.В., Шумаков, П.В. Delphi 4. Руководство разработчика баз данных. – М. : Нолидж, 1999.
4. Хофман, В., Хоменко, А. DELPHI-быстрый старт. – СПб. : БХВ-Петербург, 2002.

Интернет-ресурсы

(Перечень адресов интернет-ресурсов с кратким описанием)

1. www.delphikingdom.com
2. www.cafe.symentec.com
3. www.sun.com
4. www.document.newmail.ru
5. www.microsoft.com