МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ
ГВУЗ «ДОНЕЦКИЙ НАЦИОНАЛЬНИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Н.А.Маслова
Р.А.Сорокин

МЕТОДИЧЕСКОЕ ПОСОБИЕ

для самостоятельной работы студентов
по дисциплине

«ОСНОВЫ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ»

Донецк,
ДонНТУ,
2013

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ
ГВУЗ «ДОНЕЦКИЙ НАЦИОНАЛЬНИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Н.А.Маслова
Р.А.Сорокин

МЕТОДИЧЕСКОЕ ПОСОБИЕ
для самостоятельной работы студентов
по дисциплине
«ОСНОВЫ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ»

Рассмотрено на заседании
кафедры «Программное
обеспечение интеллектуальных
систем»
протокол №6 от 6.12.2012г.

Утверждено на заседании
Ученого Совета
Донецкого национального
технического университета
протокол № 1 от 21.02.2013г.

Донецк,
ДонНТУ,
2013

УДК 004.4’236

Методическое пособие для самостоятельной работы студентов по дисциплине «Основы визуального программирования»/ Сост.Н.А.Маслова, Р.А.Сорокин. – Донецк: Наука і освіта, 2013. - 90 с.

Учебно-методическое пособие предназначено для студентов, обучающихся по направлению «Программная инженерия», «Компьютерная инженерия», «Компьютерные науки», «Информатика» и «Системная инженерия».

Содержит основные теоретические сведения по технологии визуального программирования, основам построения приложений в визуальной среде, включает разделы работы с графическими объектами, базами данных, компонентами пользователя. Подробно описана технология разработки и создания DLL в визуальной среде.

Материал разбит на отдельные темы, содержит основные определения, которые необходимо усвоить при изучении курса, в конце каждого раздела размещен перечень вопросов.

Пособие построено в презентационно-демонстрационном стиле, что позволило изложить большой объем теоретической информации. Выбранный стиль способствует лучшему усвоению курса студентами за счет наглядности и последовательности изложения материала.

Составители: к.т.н., доц. Маслова Н.А., ст.преп.Сорокин Р.А.

Рецензент: доктор технических наук, профессор, заведующий кафедрой «Вычислительная математика и программирование» Павлыш В.Н.

Содержание

Тема 1. *Технологии программирования. Современные средства создания программного обеспечения. Общие понятия визуального программирования*	
1.1. Технологии программирования	4
1.2. Особенности структурного программирования	5
1.3. Структурно-модульный подход	6
1.4. Технология абстрактных данных Дейкстры	7
1.5. Технология объектно-ориентированного программирования	8
1.6. Структура объекта	9
1.7. Основополагающие принципы	9
1.8. Форма	10
1.9. Компонент	11
1.10. Последовательность разработки программ в технологии визуального программирования	11
1.11. Факторы ускорения разработки в визуальной среде	12
Перечень вопросов по теме 1	13
Тема 2*. Среды и языки визуального программирования. Основные инструменты визуальной разработки программных приложений*	
2.1.Пример работы в среде Powersim	15
2.2. Пример MQL-Studio for HiAsm	15
2.3. Пример диаграмм UML	16
2.4. Пример работы в среде Trace Mode	16
2.5. Классификация языков визуального программирования	17
2.6. Графические и визуальные языки программирования	18
2.7.Визуальные средства разработки	19
Перечень вопросов по теме 2	20
Тема 3. *Библиотеки классов и модель визуальных компонент. Иерархия классов VCL. Управление компонентами. Свойства, события и методы Компонент*	
3.1.Библиотеки классов	21
3.2. Основные категории классов библиотеки OWL	22
3.3. Преимущества и недостатки OWL	22
3.4. Библиотека MFC	23
3.5. Библиотека VCL	23
3.6. Иерархия базовых классов библиотеки VCL	24
3.7. Некоторые классы VCL	25
3.8. Компоненты VCL	26
*3.*9. Общие свойства компонент	27
3.10. Управление компонентами	28

Перечень вопросов по теме 3	30
Тема 4. *Техника визуального программирования. Методика использования компонентов в интегрированной среде визуальной разработки* ***C++Builder***	
4.1. Версии C++ Builder	31
4.2. Особенности IDE C++ Builder	32
4.3. Обеспечение скорости проектирования программ в C++ Builder достигается за счет использования	33
4.4. Создание и открытие файла проекта C++ Builder	34
4.5. Просмотр модуля (формы) C++ Builder	35
4.6. Расшифровка головного файла проекта	35
4.7. Структура файла модуля формы	36
4.8. Описание класса формы	36
4.9. Файлы, автоматически создаваемые средой C++ Builder	37
4.10. Пример экрана с файлами проекта C++ Builder	39
Перечень вопросов по теме 4	40
Тема 5. *Основы проектирования интерфейсов. Понятие интерфейса. Классификация интерфейсов. Компоненты интерфейса. Интерфейс пользователя. Проектирование меню.*	
5.1. Способы организации интерфейсов	41
5.2. Виды интерфейсов	42
5.3. Командный интерфейс	42
5.4. WIMP –интерфейс	43
5.5. SILK-интерфейс	43
5.6. Разработка интерфейса пользователя	44
5.7. Компоненты управления интерфейсом	44
5.8. Возможности редактора меню C++Builder	45
5.9. Свойства Главной формы	45
5.10. Способы запуска внешних программ и создания дочерних процессов	46
Перечень вопросов по теме 5	47
Тема 6. *Основные принципы работы с графикой. Методы и свойства графических компонент. Особенности построения диаграмм и графиков в визуальной среде.*	
6.1.Основные понятия	48
6.2. Представление цвета	49
6.3. Задание цвета 32-битным целым	49
6.4. Задание цвета в формате RGB	50
6.5. Вывод текста	50
6.6.Скрытые изображения	51
6.7.Ограничивающие области	52
6.8.Непрямоугольная форма	52
6.9. Ромбовидная ограничивающая область	53
6.10.Способы построения графиков и диаграмм	53

6.11. Свойства компонента Chart	54
6.12.Порядок создания графика с помощью компонента Chart	54
6.13.Алгоритмы анимации	55
Перечень вопросов по теме 6	56
Тема 7. *Основные принципы разработки приложений баз данных в среде визуального программирования*	
7.1. БД и СУБД	57
7.2. Классификация БД	57
7.3. Архитектура доступа к базам данных	58
7.4. Архитектура доступа к BDE	59
7.5. Доступ к базам данных BDE	60
7.6. Доступ к базам данных ADO	61
7.7. Режимы отображения данных	61
7.8. Порядок настройки работы в режиме таблицы	62
7.9. Порядок настройки работы в режиме конструктора	62
7.10. Настройки ADO-доступа	63
7.11. Выбор информации из БД	65
Перечень вопросов по теме 7	66
Тема 8. *Основные принципы построения диалогов и справочных систем в среде визуального программирования*	
8.1. Основные принципы построения диалогов	67
8.2. Типы диалогов	67
8.3. Способы отображения справочной информации	68
8.4. Порядок создания справочной системы	68
8.5. Экраны справочной системы	69
Перечень вопросов по теме 8	70
Тема 9. *Принципы построения различных компонентов и порождающих их классов в среде визуального программирования. Разработка компонента программиста.*	
9.1. Этапы создания собственных компонент	71
9.2. Создание модуля компонента	72
9.3. Наследование компонента	72
9.4. Порядок регистрации компонент	73
9.5. Последовательность отладки компонент	73
9.6. Инсталляция компонента на Палитру	74
9.7.Удаление компонент из VCL	75
9.8. Перестройка Палитры компонентов	75
9.9. Сохранение файлов нового компонента	75
Перечень вопросов по теме 9	76
Тема 10. *Создание и использование DLL в визуальных средах. Установка приложений на ПК пользователя*	
10.1. Создание и использование DLL	77
10.2. Преимущества DLL	77

10.3. Типы DLL	77
10.4. Категории функций в DLL	78
10.5. Ключевое слово __declspec()	79
10. 6. Порядок создания DLL	79
10.7. В комплект поставки DLL должны входить	80
10.8. Порядок создания DLL с помощью репозитория объектов	80
10.9. Добавление к проекту заголовочного файла для DLL	80
10.10. Создание файла библиотеки импорта	81
10.11. Добавление файла библиотеки импорта к проекту	81
10.12. Создание вызывающего приложения	81
10.13. Статическая и Динамическая з**агрузка DLL**	82
10.14. Установка приложений на ПК пользователя	83
10.15. Статическая компоновка проекта	84
10.16. Возможности генератора дистрибутивов (Install Shield Express)	85
Перечень вопросов по теме 10	86
Перечень рекомендованной литературы	87

Введение

Программирование – сравнительно молодая и быстро развивающаяся отрасль науки и техники, в которой постоянно появляются новые методы и технологии, которые, в свою очередь, служат основой более современных средств разработки программного обеспечения.

Технологией программирования называют совокупность методов и средств, используемых в процессе разработки программного обеспечения.

Различают технологии, используемые на конкретных этапах разработки или для решения отдельных задач этих этапов, и технологии, охватывающие весь процесс разработки. В основе первых, как правило, лежит ограниченно применимый метод, позволяющий решить конкретную задачу. В основе вторых - методология или подход (парадигма), определяющий совокупность методов, используемых на разных этапах разработки.

Визуальное программирование является в настоящее время одной из наиболее популярных парадигм программирования. Оно востребовано, так как обеспечивает приближение формы представления программы и способов ее кодирования к образному способу мышления человека. Программист не создает текст программ, а показывает, что должно получиться в результате. Программа генерируется автоматически с помощью визуального прототипа.

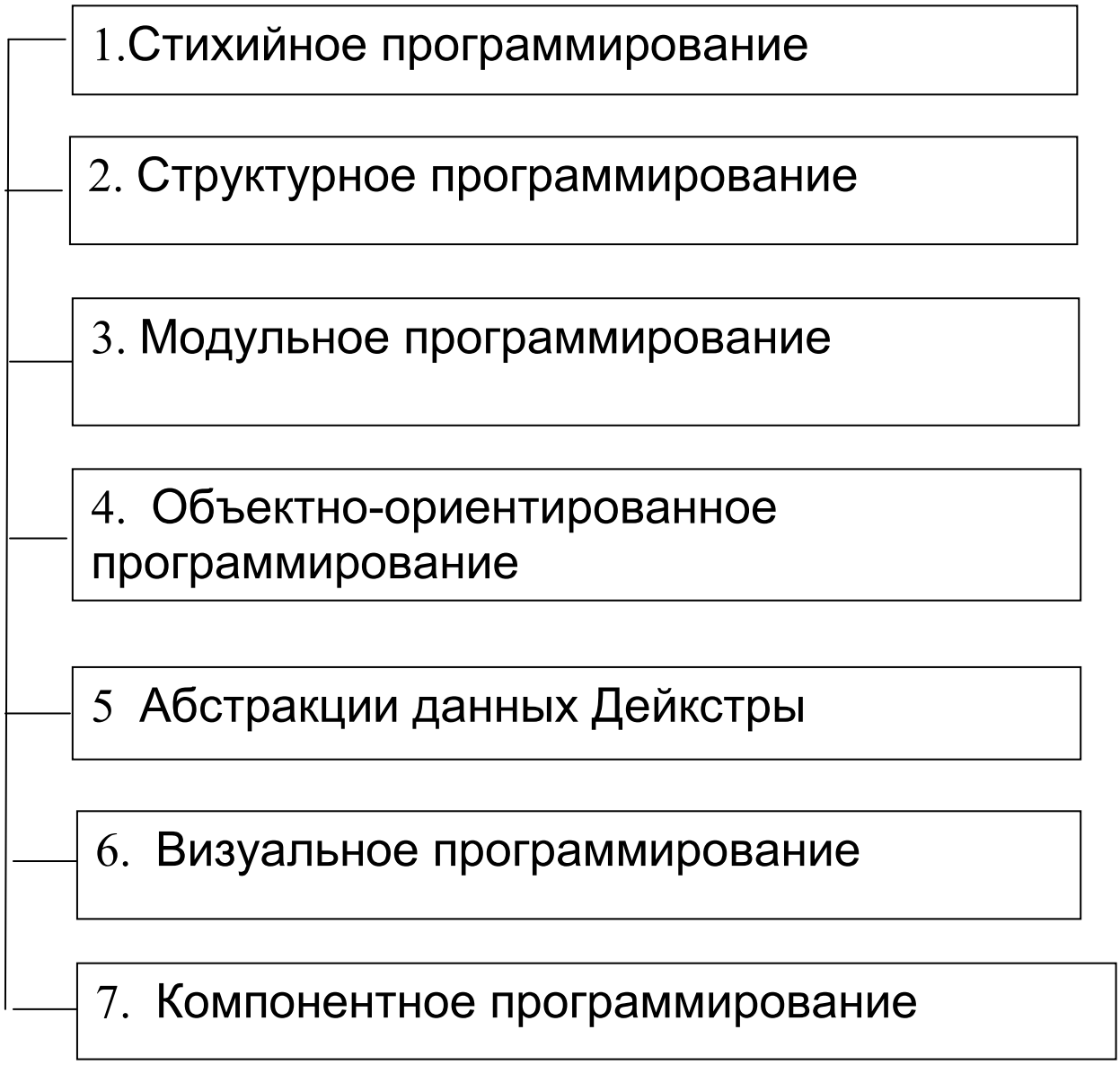
Визуальное программирование основывается на объектно-ориентированном программировании и OLE-технологии, позволяет профессионалам проектировать и разрабатывать большие программные комплексы быстрее. Среды, поддерживающие визуальное программирование, например, Delphi, C++ Builder, Visual C++ и т. д. были созданы еще на этапе становления объектного подхода к программированию.

Для непрофессионалов визуальное программирование позволяет сосредоточиться на сущности задачи предметной области (ее объектах, отношениях между ними, поведении объектов или их состоянии), абстрагируясь от особенностей реализации в каждом конкретном диалекте языка программирования.

Тема 1.

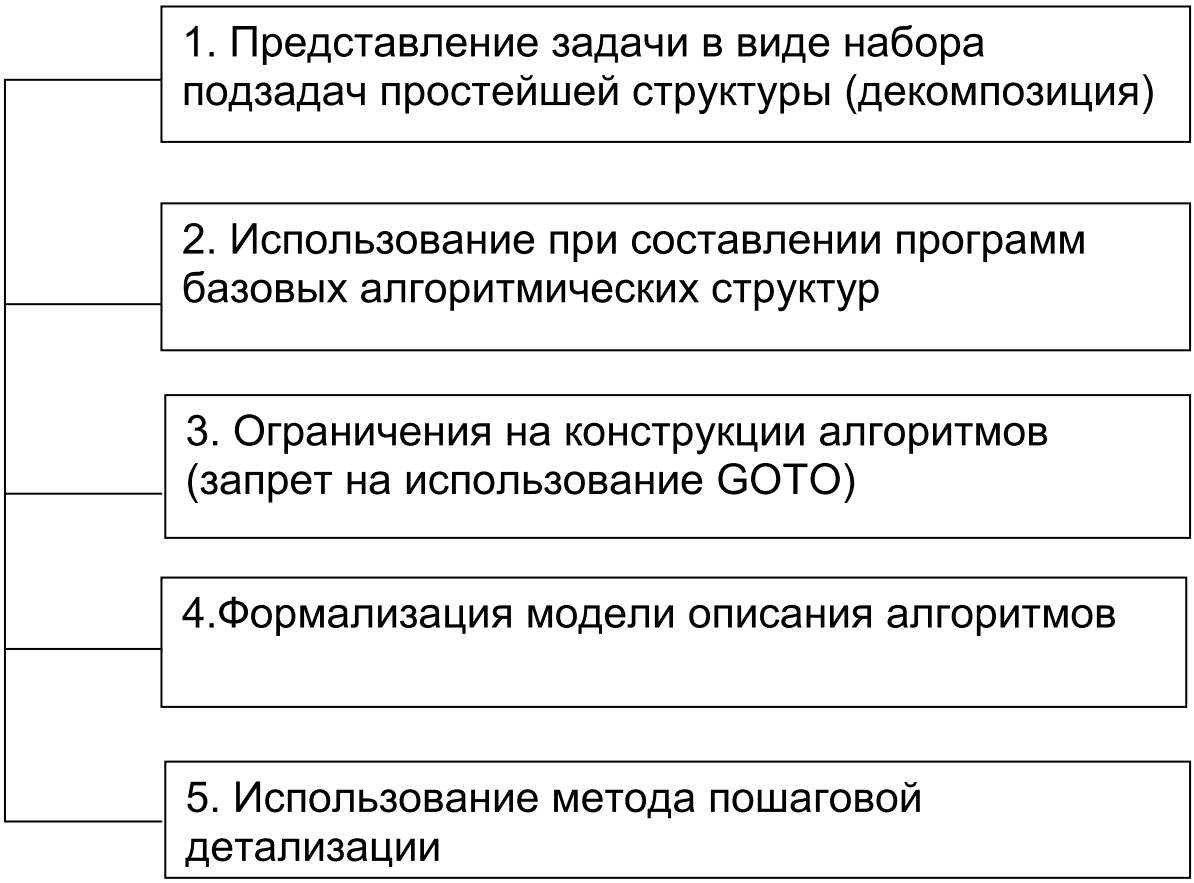
Технологии программирования. Современные средства создания программного обеспечения. Общие понятия визуального программирования

1.1. Технологии программирования



Технология программирования – это совокупность методов и средств, используемых в процессе разработки программного обеспечения. Включает набор операций и технологические инструкции по их использованию, определяет способ описания проектируемой системы или модели, используемой на конкретном этапе.

1.2. Особенности структурного программирования

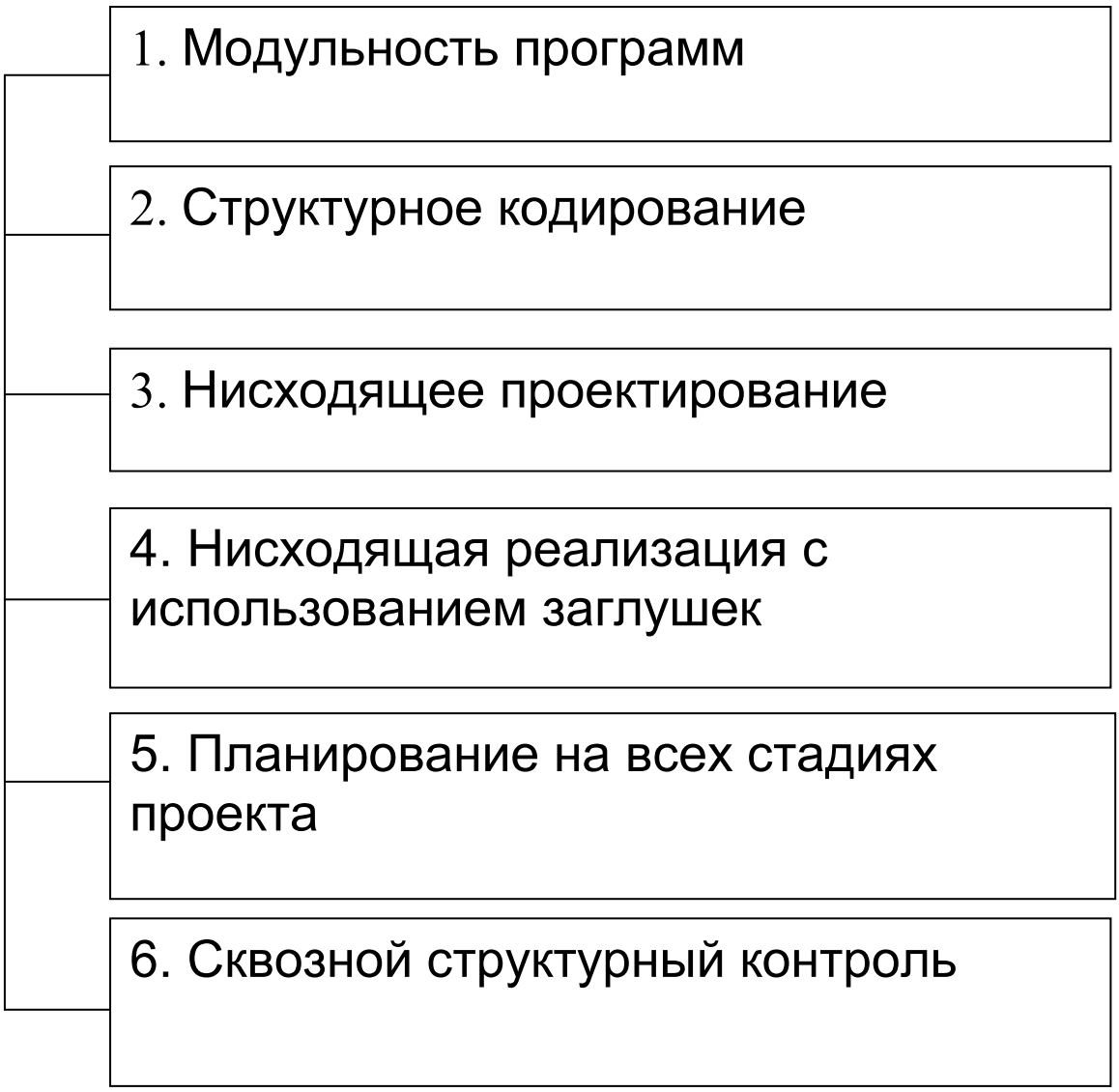


В основе структурного подхода лежат принципы структуризации и декомпозиции. Структура программы представляется набором подчиненных модулей - функций или процедур, оформленных определенным образом и выполняющих строго определенное действие.

Восходящий принцип программирования (снизу вверх) – это переход от примитивных машинных команд к требуемой реализации программы - такой подход к решению задачи, когда программист сначала изучает имеющиеся в его распоряжении вычислительную машину и/или язык программирования, а затем собирает некоторые последовательности инструкций в элементарные процедуры, типичные для решаемой задачи, используя их на следующем уровне иерархии процедур.

Нисходящий принцип программирования (сверху вниз) в Структурном Программировании: при поэтапной декомпозиции и одновременном развитии и детализации программы происходит постепенное продвижение вглубь.

1.3. Структурно-модульный подход

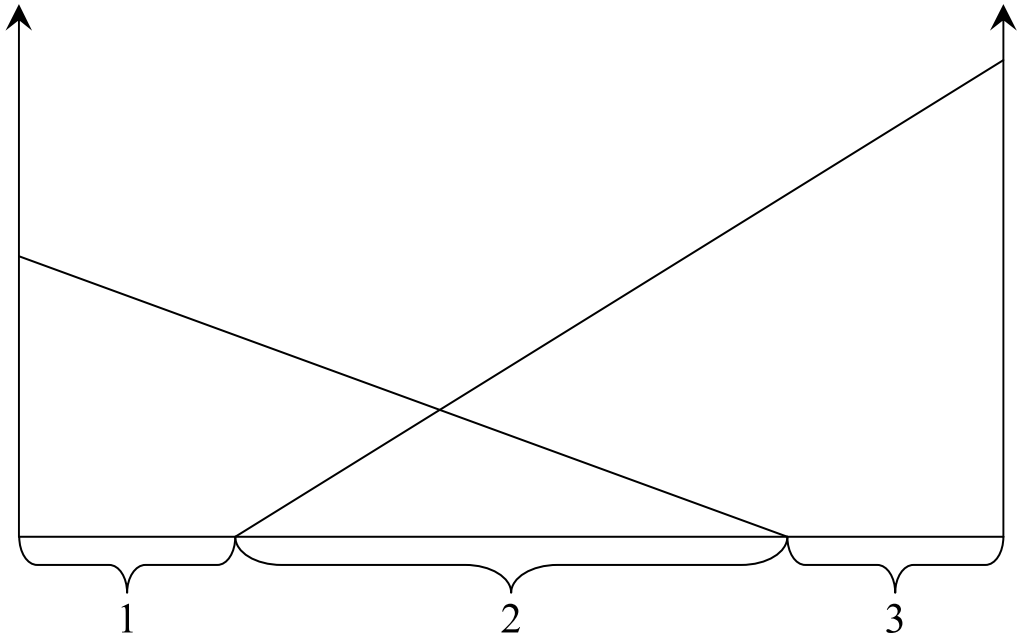


Модульное программирование предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные в отдельно компилируемые модули (библиотеки подпрограмм).

Внедрение Структурно-модульного подхода к программированию обеспечило повышение производительности труда программистов; получение программ, удобных для сопровождения; возможность работы над проектом коллектива разработчиков.

1.4. Технология абстрактных данных Дейкстры

Превращение главного входного потока информации в выходной поток



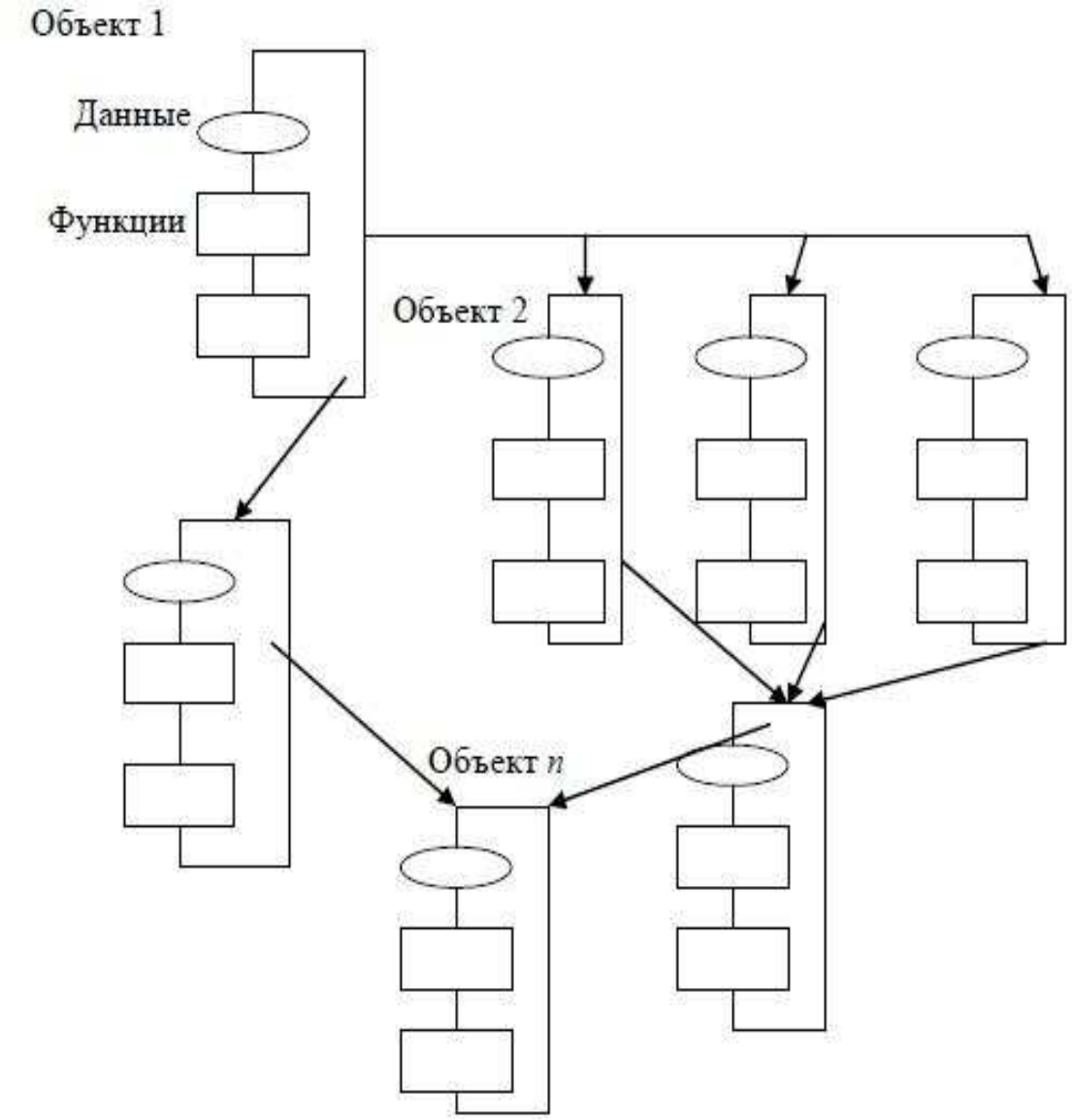
1 – участок соответствует преобразованию информации входного потока в промежуточную информацию

2 – участок соответствует получению выходной и промежуточной информации из входной и промежуточной информации

3 – участок соответствует получению выходной информации из промежуточной информации

Методика ставит во главу данные, при этом тщательно специфицируются выход, вход, промежуточные данные. Основное внимание уделяется типизации данных с использованием структур для объединения близкой по смыслу информации в единые данные.

1.5. Технология объектно-ориентированного программирования



Объектно-ориентированное программирование – это методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса, а классы образуют иерархию на принципах наследуемости.

Объект в языке объектно-ориентированного и визуального программирования — это абстрактный тип данных, содержащий структуры данных и набор функций, обрабатывающих эти данные.

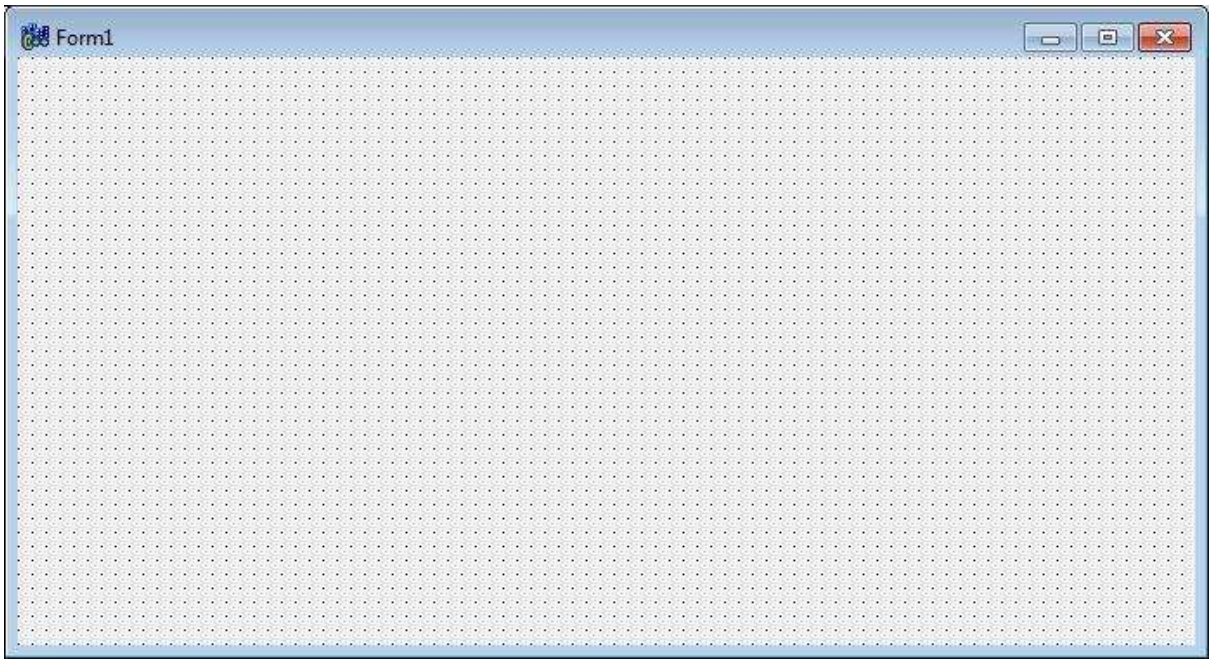
1.6. Структура объекта

1.7. Основополагающие принципы

Компонент (component) — это функционально законченный участок двоичного кода, выполняющий определенную задачу и управляемый через свойства, методы и события.

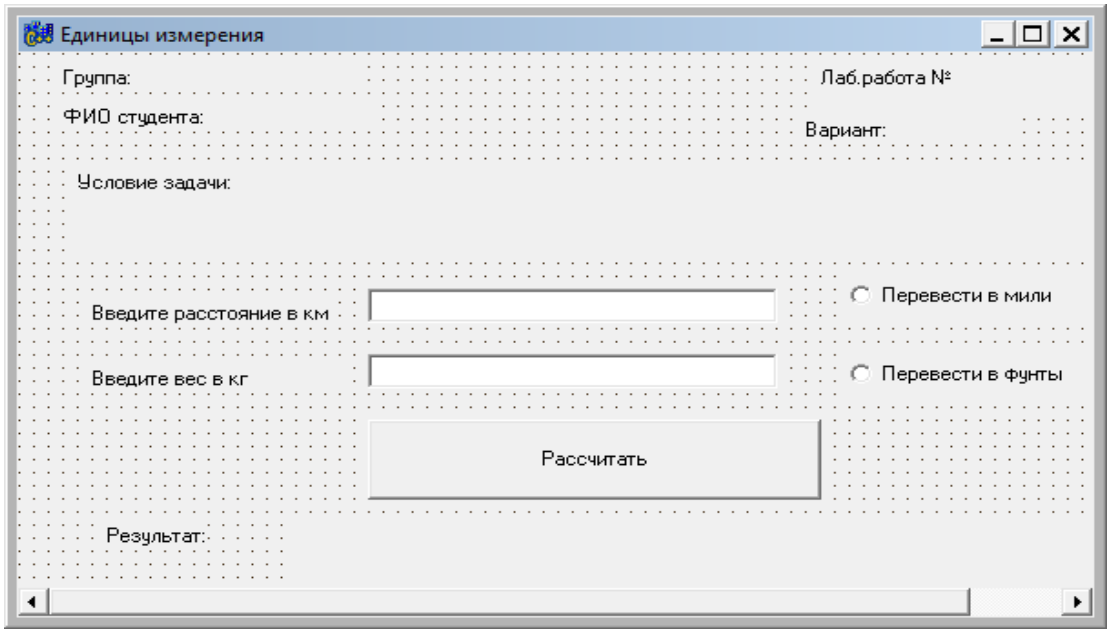
Методы объекта - это процедуры и функции, объявление которых включено в описание объекта и которые выполняют некоторые действия.

1.8. Форма

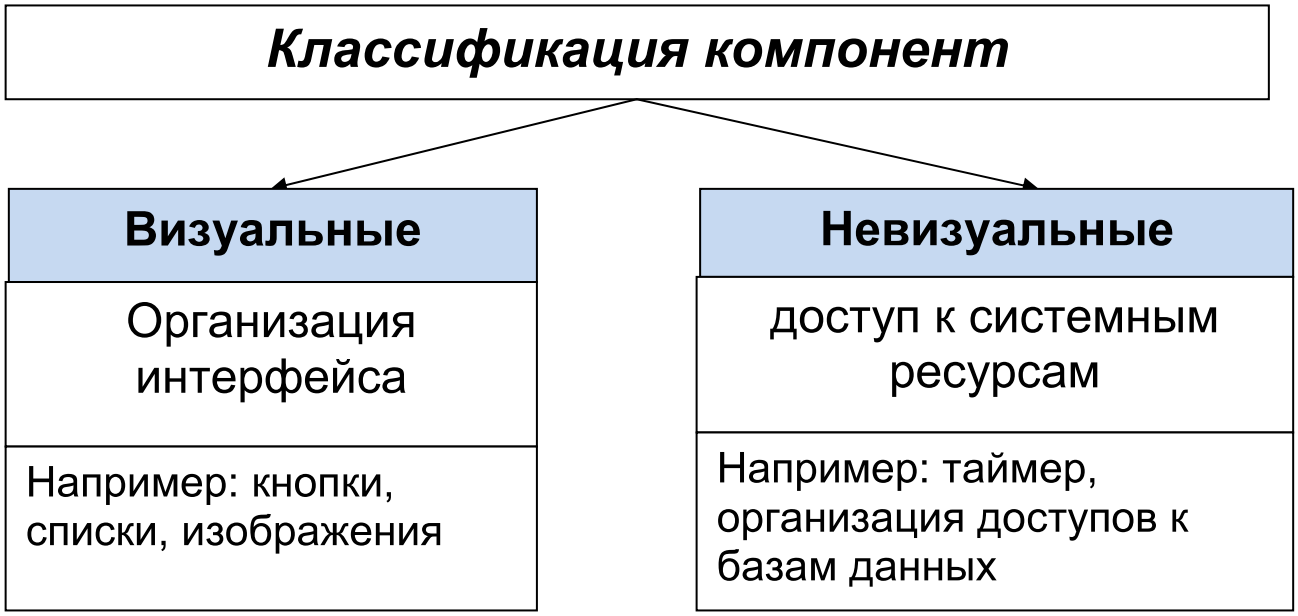


Форма в технологии визуального программирования- это визуальный объект, обладающий свойством окна Windows. При разработке на форме размещаются визуальные и невизуальные компоненты. Форма является контейнером (родителем – **Parent**) размещенных на ней компонентов.

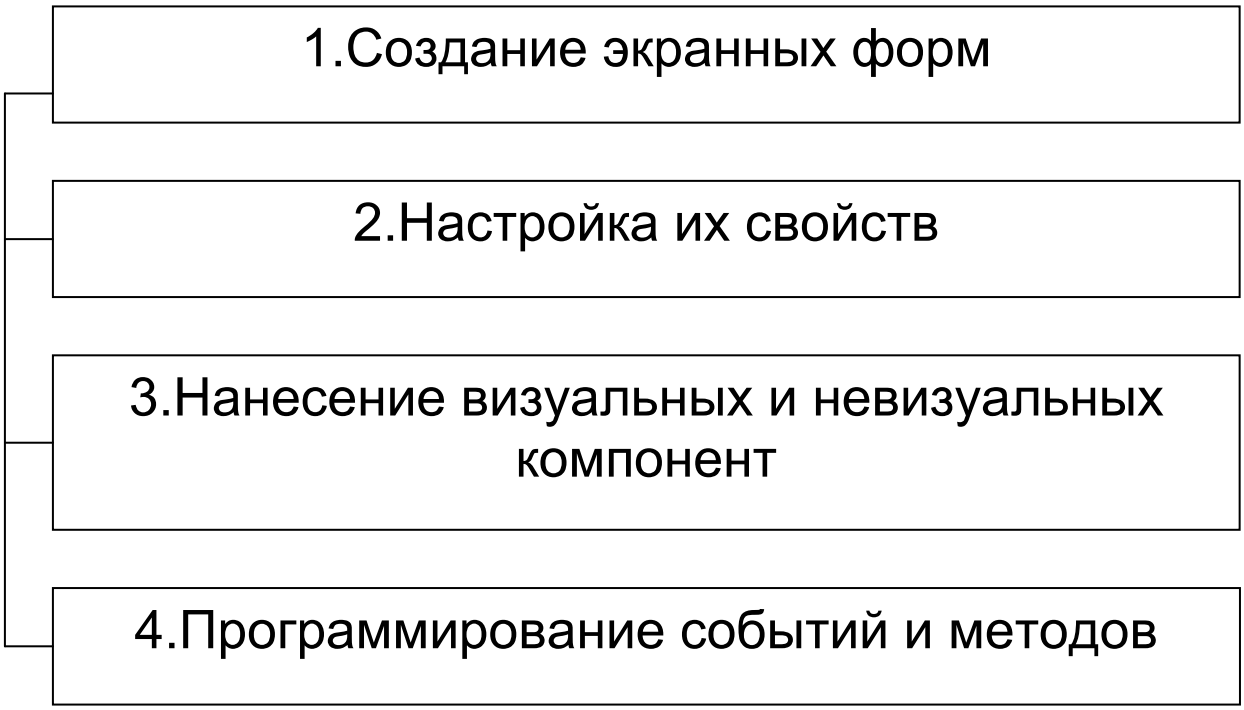
Форма с компонентами



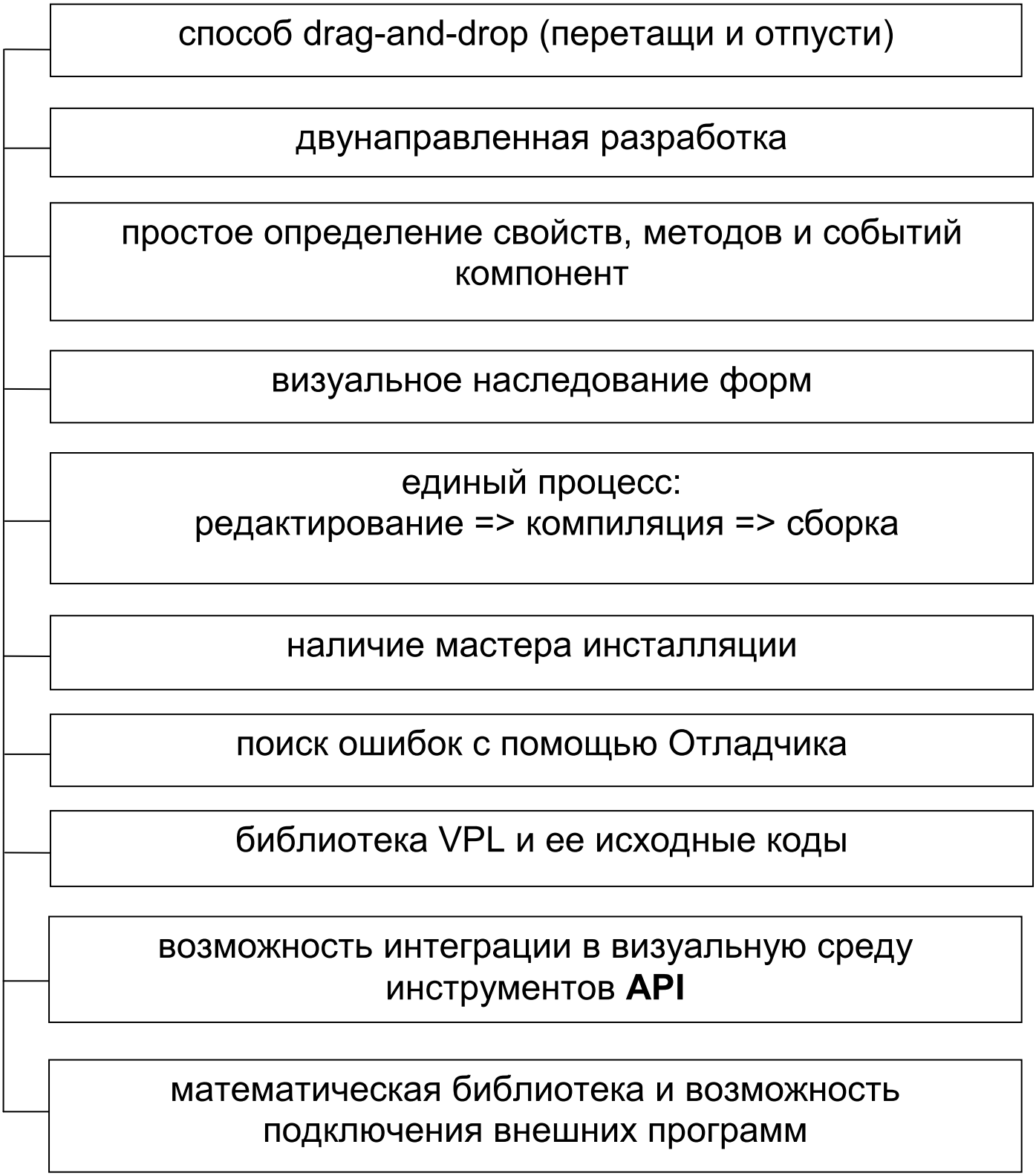
1.9. Компонент



1.10. Последовательность разработки программ в технологии визуального программирования



1.11. Факторы ускорения разработки в визуальной среде



Перечень вопросов по теме 1

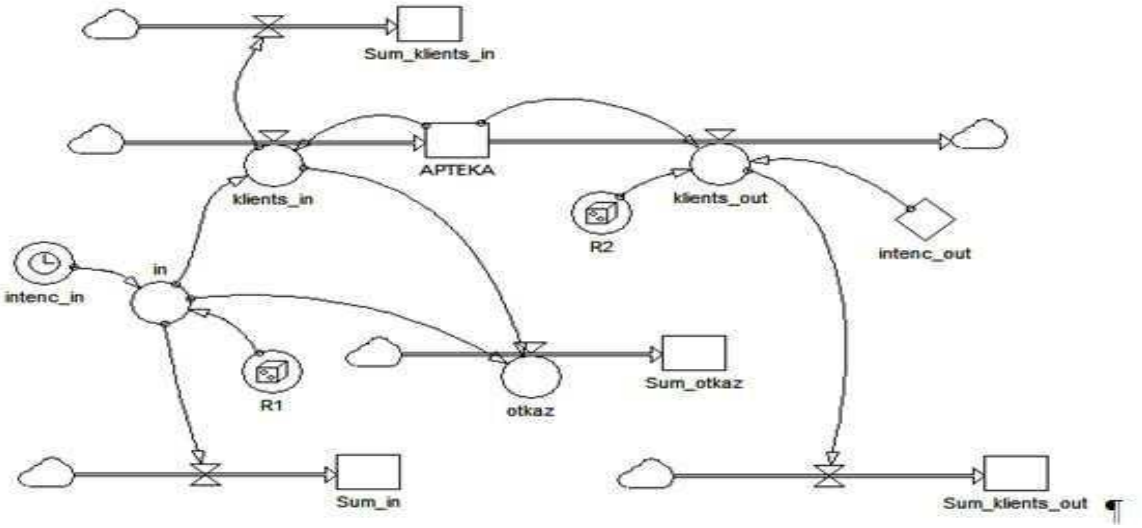
1. Назовите основные принципы структурно-модульного подхода.
2. Какие характеристики являются фундаментальными в ООП.
3. Дайте определение понятию Форма. Для чего она предназначена?
4. Дайте определение понятию Объект с точки зрения визуального программирования.
5. Назовите основные шаги технологии визуального программирования.
6. Укажите особенности технологии, основанной на абстракции данных Дейкстры.
7. Назовите основополагающие принципы объектно-ориентированного программирования.
8. Что относится к понятию методы объекта.
9. Что является основным элементом конструкции в объектно-ориентированных языках
10. Что такое способ drag-and-drop? В какой технологии программирования он применяется.
11. Назовите области применения технологии абстрактных данных Дейкстры. Как преобразуется информация в соответствии с этой технологией.
12. Какие виды компонент вы знаете?
13. Что такое восходящий метод программирования. Когда он используется.
14. Чем привлекает визуальное программирование профессионалов.
15. Что такое декомпозиция. В какой технологии программирования используется это понятие.
16. В чем преимущества определения Свойств, методов и событий в визуальном программировании.
17. Назовите принципы проектирования в структурном программировании.
18. Перечислите функциональные возможности визуального программирования, приводящие к ускорению процесса разработки программ.
19. Что такое системный подход. Дайте определение понятию.
20. Что такое Класс в объектно-ориентированном программировании и в технологии визуального программирования.
21. Дайте определение понятию Компонент в технологии визуального программирования.

22. Охарактеризуйте структурный подход к программированию.
23. Что такое Экземпляр объекта в объектно-ориентированном и визуальном программировании.
24. Дайте определение понятию Невизуальные компоненты. Укажите их назначение. Перечислите известные вам невизуальные компоненты
25. В чем основные цели любой Технологии программирования.
26. В чем заключается важность наследования.
27. Укажите особенности создания программного продукта в технологии визуального программирования.
28. Перечислите особенности структурного программирования.
29. Что такое Объект, каковы его основные особенности.
30. Дайте определение понятию «Визуальное программирование». Какое определение внесено в словарь электричества и электроники.
31. Что такое нисходящий метод программирования. Когда он используется.
32. Найдите в Интернете понятие «two-way-tools» и раскройте его. К какой технологии программирования оно относится?
33. Раскройте понятие «Стихийное программирование».
34. Дайте определение понятию Визуальные компоненты. Когда они применяются. Приведите примеры визуальных компонент.
35. Сформулируйте понятие «Технология программирования». Перечислите наиболее распространенные технологии программирования.
36. Перечислите преимущества визуального программирования.
37. Дайте определение модулю в понятии структурного программирования.
38. Назовите основные этапы разработки программ по технологии визуального программирования.
39. Назовите основной недостаток системного подхода.
40. Для чего предназначены свойства объекта.
41. Что такое Common Object Request Broker Architecture. К какой технологии относится это понятие.
42. Элементы каких технологий получили развитие в технологии визуального программирования.

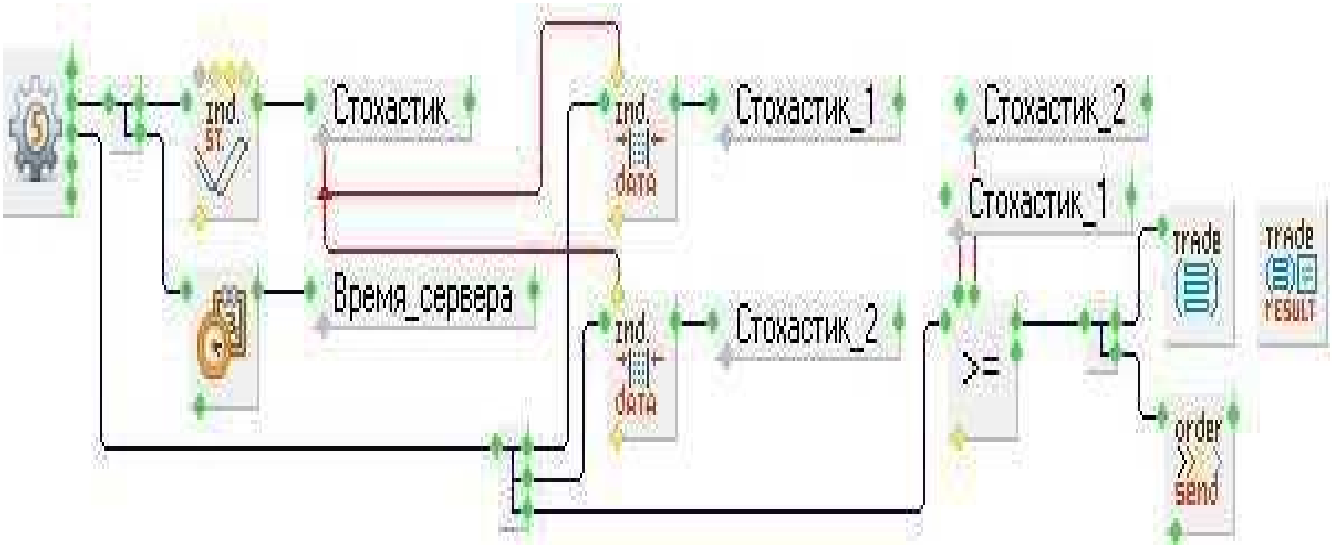
Тема 2.

Среды и языки визуального программирования. Основные инструменты визуальной разработки программных приложений.

2.1.Пример работы в среде Powersim

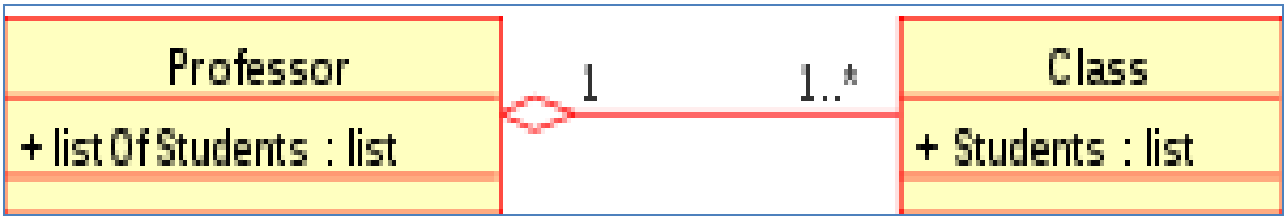


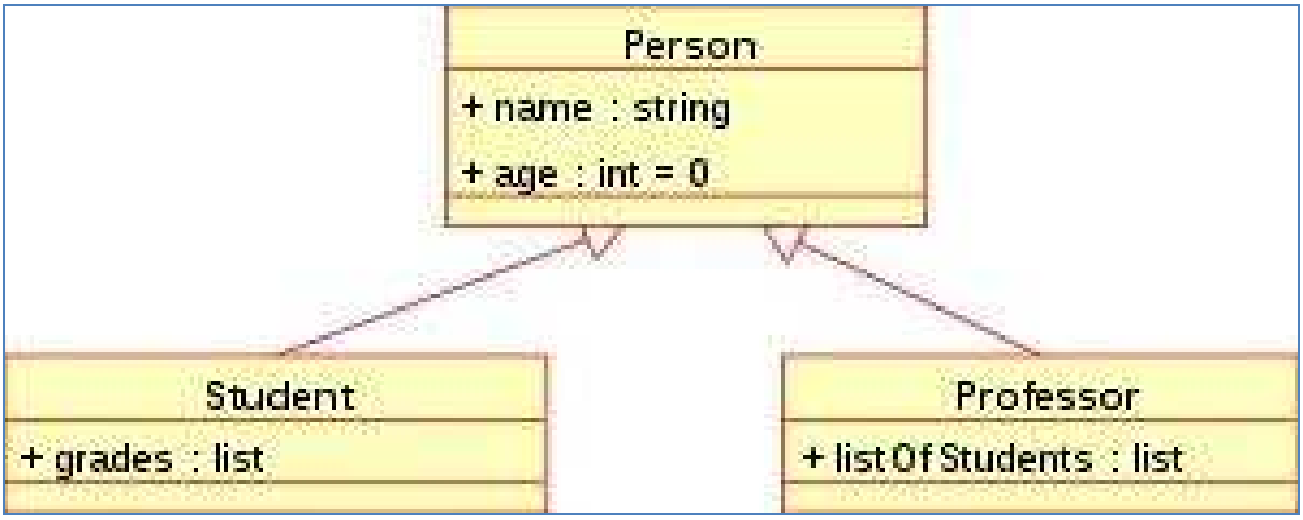
2.2. Пример MQL-Studio for HiAsm



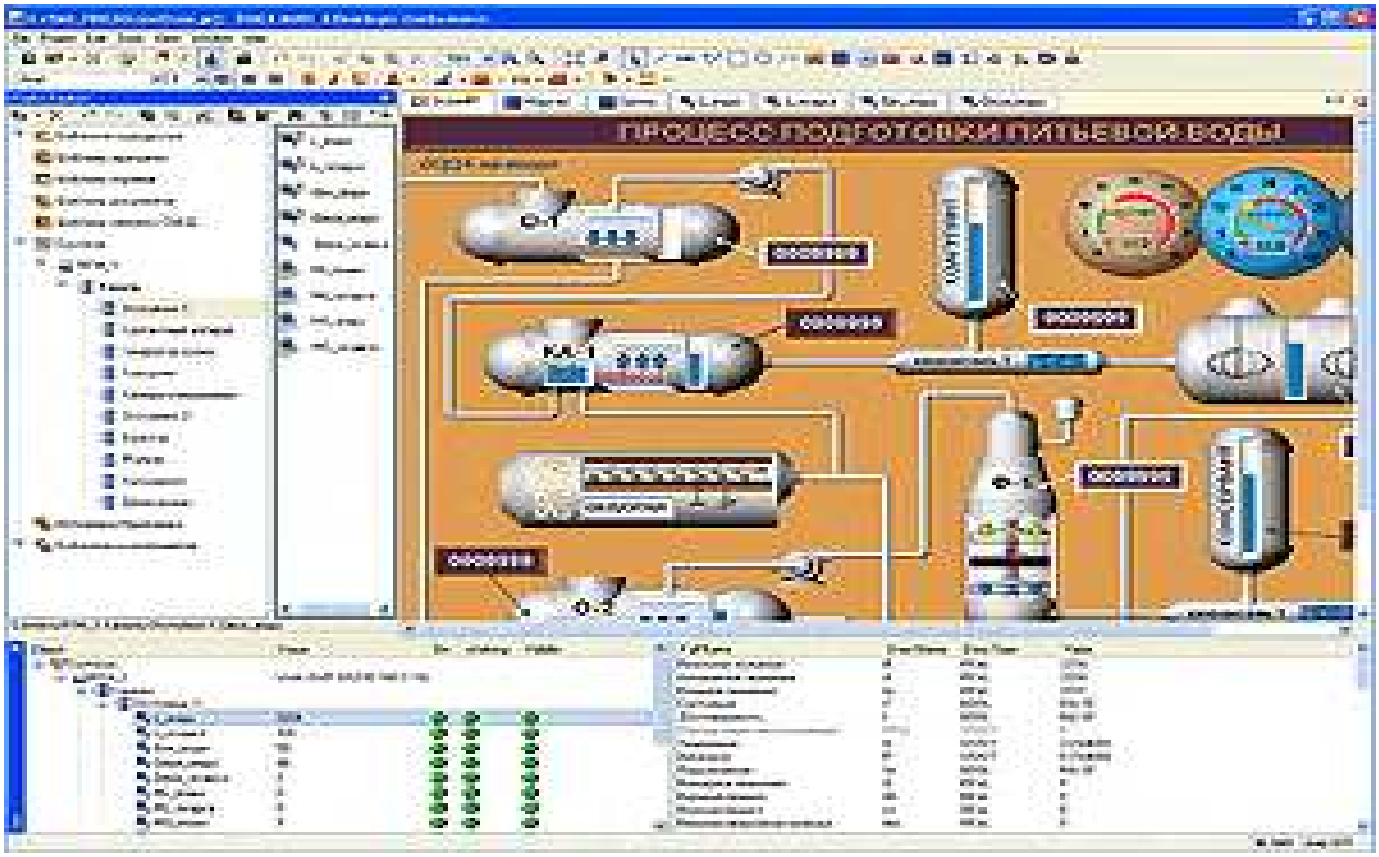
Визуальное программирование (visual programming) — способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста.

2.3. Пример диаграмм UML

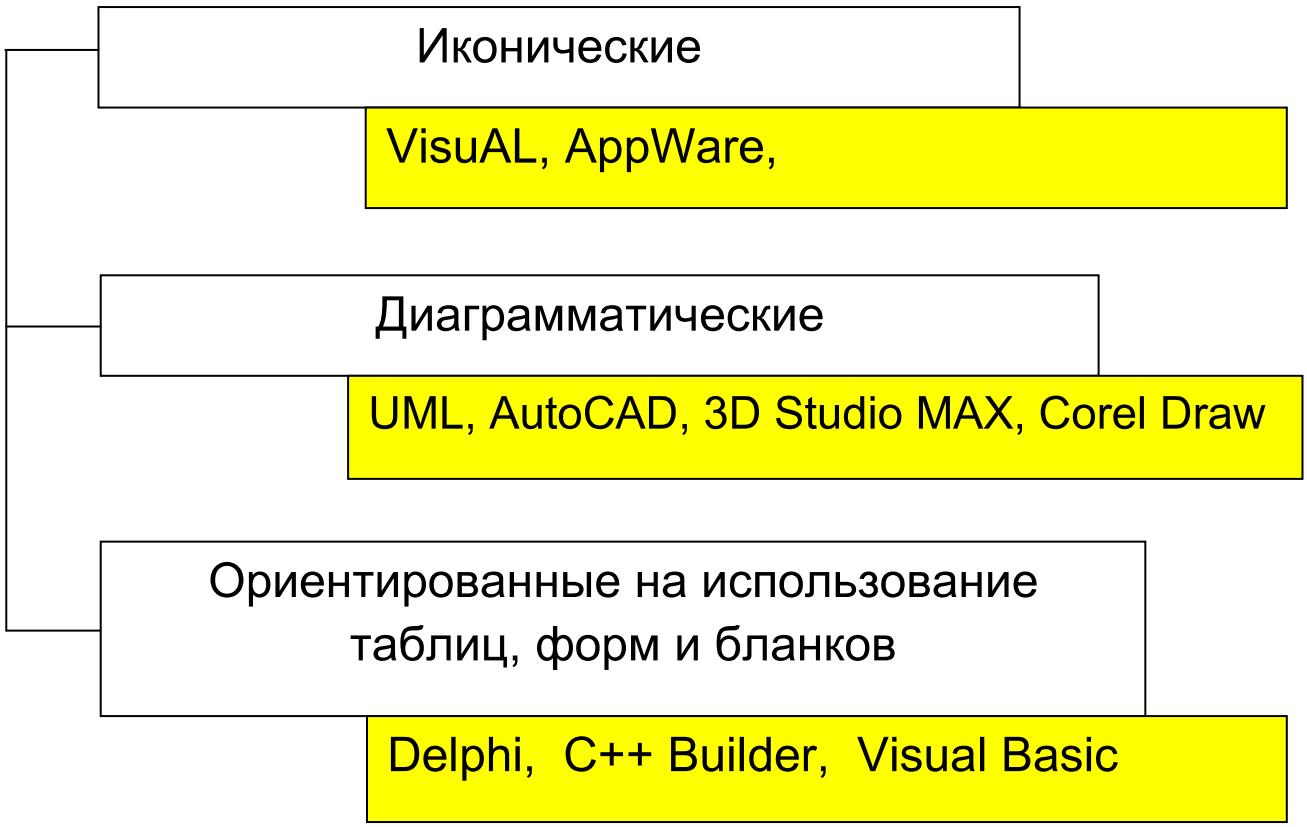




2.4. Пример работы в среде Trace Mode



2.5. Классификация языков визуального программирования



Delphi, C++Builder и **Visual Basic** относятся к профессиональным интегрированным средам разработки (Integrated Development Environment, IDE), построенным по принципам нисходящего визуального стиля программирования и называемым RAD-средами (Rapid Application Development) - инструментами быстрой разработки приложений.

Эти среды используют визуальную технологию проектирования, а код записывается с помощью текстовых языков программирования (создание экранных форм, настройка их свойств, нанесение визуальных и невизуальных компонент, программирование событий и методов оконных форм).

Достоинством языков визуального программирования является наглядное представление данных и структур разрабатываемых программ; предоставляемая программистам возможность работать в терминах своих представлений.

2.6. Графические и визуальные языки программирования

	Наименование языка	Назначение
1	G	язык графического программирования, используемый в системе LabVIEW
2	VisuAL (Visual Algorithmic Language)	язык визуального программирования для обучения алгоритмике и основам программирования
3	FBD	язык Функциональных блоковых диаграмм
4	CFC	высокоуровневый язык графического программирования, создан для проектирования систем управления непрерывными технологическими процессами
5	SFC	графический язык программирования, широко используется для программирования промышленных логических контроллеров PLC и последовательных функциональных схем
6	LD	язык релейно-контактных схем
7	Befunge	язык программирования, в котором команды размещаются графически в текстовом файле
8	eXpecco	основанный на графическом потоке язык программирования и среда разработки для автоматизации тестирования
9	VisSim	визуальный язык программирования для моделирования динамических систем, и микропроцессоров
10	Дракон-схемы	Графический язык программирования для ракетно-космической отрасли

Графический язык программирования — язык, предназначенный для написания программ для компьютера или вычислительного устройства, в котором вместо текстового описания алгоритма работы используется графическое.

2.7. Визуальные средства разработки

	Пакет	Краткое описание
1	AgentSheets	система для игр и научных расчётов
2	AppWare	программирование, на основе иконок для Mac OS и Microsoft Windows
3	Automator	система, созданная Apple для Mac OS X, работает по принципу drag-and-drop. Для использования программы не требуется знаний никаких языков, все действия выполняются полностью в графической среде
4	JMCAD	среда графической разработки систем для промышленности, инженеров и учёных
5	Kwikpoint	пиктограммный визуальный транслятор, созданный Аланом Стилманом
6	LabVIEW	среда разработки и платформа для выполнения программ, созданных на графическом языке «G» фирмы National Instruments (США). Используется в системах сбора и обработки данных, для управления техническими объектами и технологическими процессами.
7	Lazarus	визуальная среда разработки профессионального уровня, по функциональности достигшая уровня Delphi пятой версии
8	Microsoft Developer Studio	интегрированная среда для разработки, позволяющая функционировать различным средам разработки, например, Visual C++ и Visual J++.
9	Microsoft Visual Studio	линейка продуктов компании Microsoft, включающая интегрированную среду разработки программного обеспечения. Позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, веб-сайты, веб-приложения практически для всех платформ, поддерживаемых Microsoft
10	Trace Mode	интегрированная информационная система для управления промышленным производством (CASE-система для быстрой разработки приложений, SCADA-система для программирования микроконтроллеров)

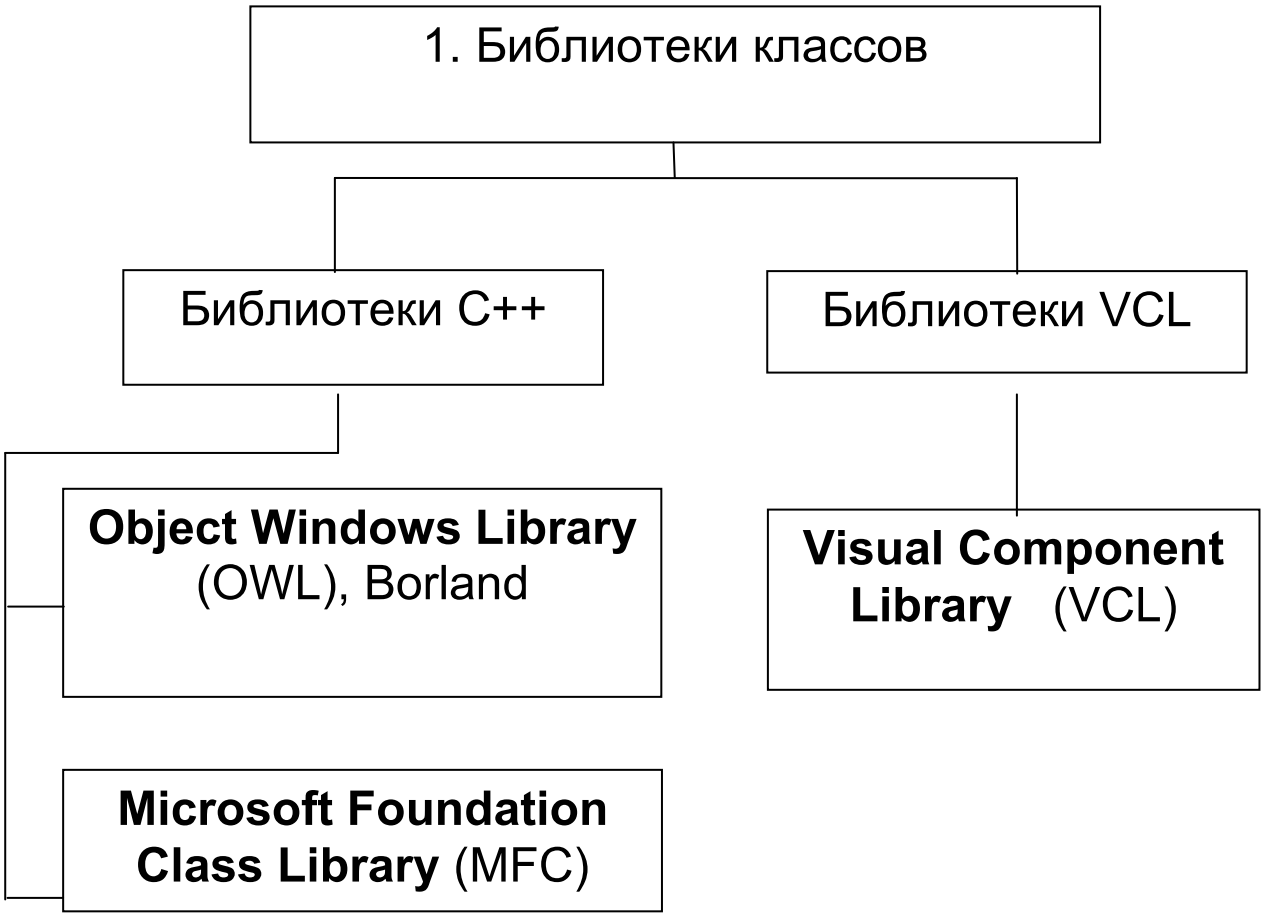
Перечень вопросов по теме 2

1. Укажите различие в понятиях Языки и Среды визуального программирования. Охарактеризуйте наиболее распространенные из них.
2. Дайте определение и раскройте суть понятия «Визуальное программирование».
3. Проведите классификацию и укажите основные особенности языков визуального программирования.
4. Приведите примеры и охарактеризуйте наиболее распространенные визуальные средства разработки. В чем их особенности и отличительные черты.
5. Что такое Диаграммы UML. Виды, область применения, назначение и особенности. По каким признакам они могут быть отнесены к визуальным средствам программирования.
6. Назначение, возможности и особенности среды Trace Mode. Области применения. Почему среда Trace Mode относится к визуальным средам.
7. Укажите основные инструменты визуальной разработки программных приложений, подходы, используемые для их реализации.
8. Чем характеризуются и в чем основные особенности Иконических языков программирования. Приведите примеры.
9. Какие особенности присущи диаграмматическим языкам программирования. Приведите примеры и дайте краткую характеристику диаграмматическим языкам программирования.
10. К какому классу относятся такие языки, как Delphi, C++Builder и Visual Basic. Какими сходными чертами они обладают.
11. Охарактеризуйте области применения и преимущества визуальных средств программирования. Какие именно факторы способствуют развитию визуального программирования.
12. Укажите недостатки визуального способа программирования. Области, в которых этот способ не подтвердил свою эффективность и почему.

Тема 3.

Библиотеки классов и модель визуальных компонент. Иерархия классов VCL. Управление компонентами. Свойства, события и методы компонент.

3.1. Библиотеки классов



Библиотека классов *(class library), frameworks* — это набор классов, инкапсулирующих часто используемые при программировании операции.

Стандартные библиотеки содержат классы для работы с окнами, управления редактированием, графических операций, отображения окон списков, растровых изображений, диалоговых окон и т.д.

3.2. Основные категории классов библиотеки OWL

- Приложения(Applications);
- Окна (Windows);
- Меню (Menus);
- Элементы управления (Controls);
- Графика (Graphics);
- Печать (Printing);
- Контроль ввода (Validators);
- Просмотр документов (Document and views);
- Буфер обмена (Clipboard).

3.3. Преимущества и недостатки OWL

Преимущества OWL:

- включает множество классов;
- имеет хорошо продуманную архитектуру;
- следует всем правилам ООП;
- работает в 16- и в 32-разрядных программах;
- содержит мощную инкапсуляцию окружения Windows.

Слабые стороны OWL:

- сложность.

OWL (Object Windows Library) - это объектно-ориентированная библиотека, базирующаяся на Windows API (Application Programming Interface).

Содержит множество классов, способствующих более быстрому и простому написанию прикладных программ. OWL-классы представляют набор объектов, из которых создается Windows-приложение, содержащее такие объекты, как окна, диалоговые окна, меню, элементы управления и другие, включая некоторые специальные объекты, такие как строки состояния и управления.

3.4. Библиотека MFC

Преимущества:

- относительно легко изучать;
- относительно небольшая надстройка над API Windows;
- принадлежность к Microsoft;
- распространена более широко, чем OWL.

Слабые стороны MFC:

- не ограждает пользователя от той информации, которую ему знать не обязательно;
- не вполне соответствует концепциям ООП;
- ее последние версии полностью 32-разрядные.

MFC менее абстрактна, чем Owl и лежит ближе к API Windows. Это, по сути, подобранная коллекция классов.

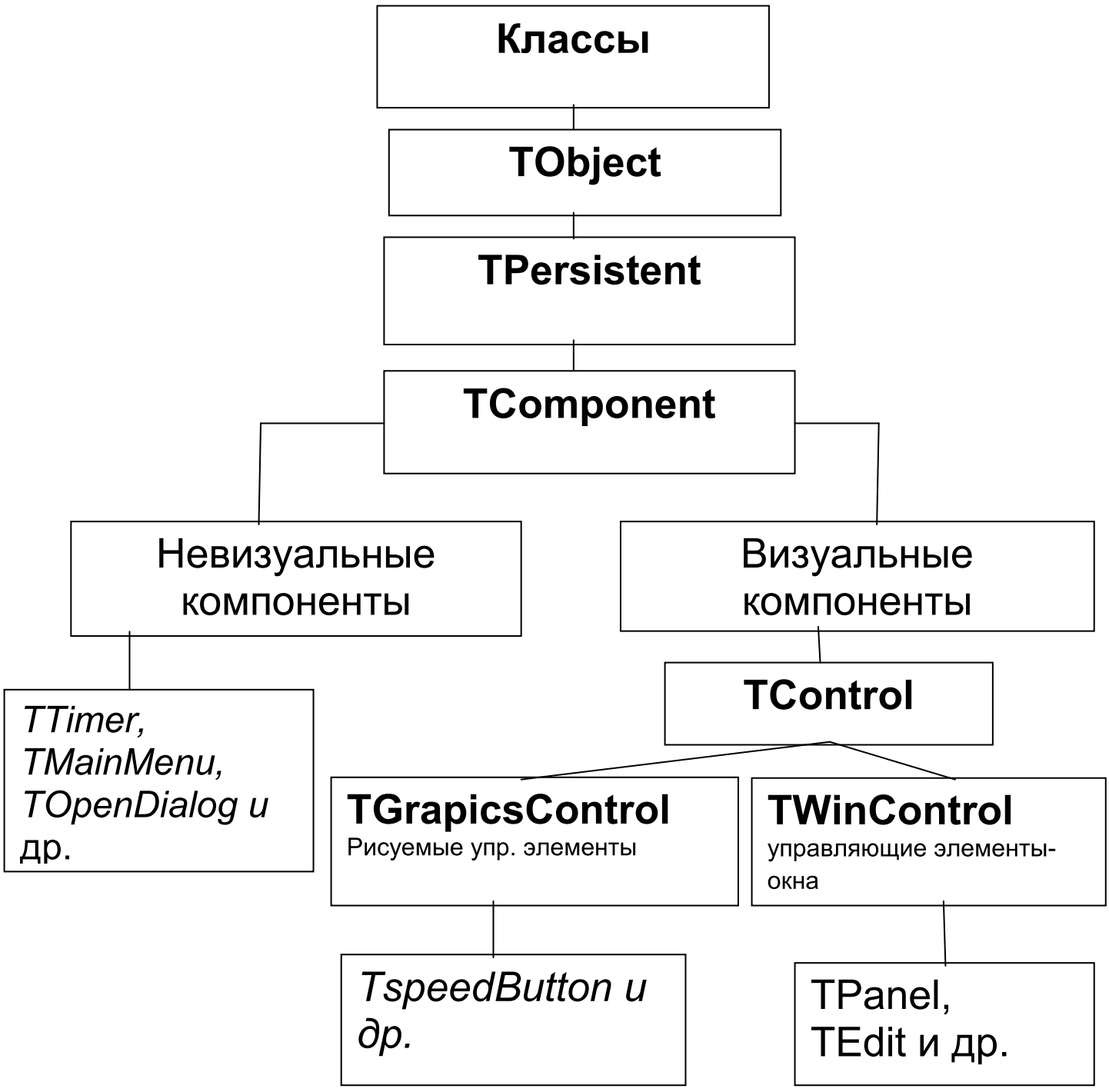
3.5. Библиотека VCL

- построена на основе принципа наследования и образует иерархию классов;
- основу библиотеки составляют классы, представляющие компоненты.

При использовании VCL следует обратить внимание на то, что:

1. Объект должен создаваться с помощью оператора **new**.
2. Все объекты VCL должны размещаться динамически.
3. Классы VCL не имеют перегруженных конструкторов.
4. Функции VCL не имеют аргументов по умолчанию.
5. Классы VCL не поддерживают множественное наследование.

3.6. Иерархия базовых классов библиотеки VCL



Библиотека **VCL** построена на основе принципа наследования и образует иерархию классов. Каждый класс представляет некий объект или компонент, обладающий набором методов, событий и свойств и имеющий специальное назначение.

3.7. Некоторые классы VCL

Формы и приложения *(TApplication, TForm)*

Классы компонентов

- **Стандартные** *(TButton, TEdit, TMemo, TMainMenu, TListBox, TPopupMenu, TCheckBox, TRadioButton, TRadioGroup, TGroupBox, TPanel)*
- **Additional** *(TBitBtn, TSpeedButton, TMaskEdit, TStringGrid, TDrawGrid, TImage, TShape, TBevel и TScrollBox)*
- **доступ к БД** *(TDataSource, TDatabase, TTable и TQuery (невиз.), TDBGrid, TDBNavigator, TDBText, TDBEdit, TDBListBox, TDBImage(виз.)*

Специальные элементы управления Win95
(TListView, TTreeView, TProgressBar, TTabControl, TPageControl, TRichEdit, TlmageList, TStatusBar)

Стандартные диалоги
TOpenDialog, TSaveDialog, TFontDia- log, TColorDialog, TPrintDialog, TPrintSetupDialog, TFindDialog, TReplaceDialog

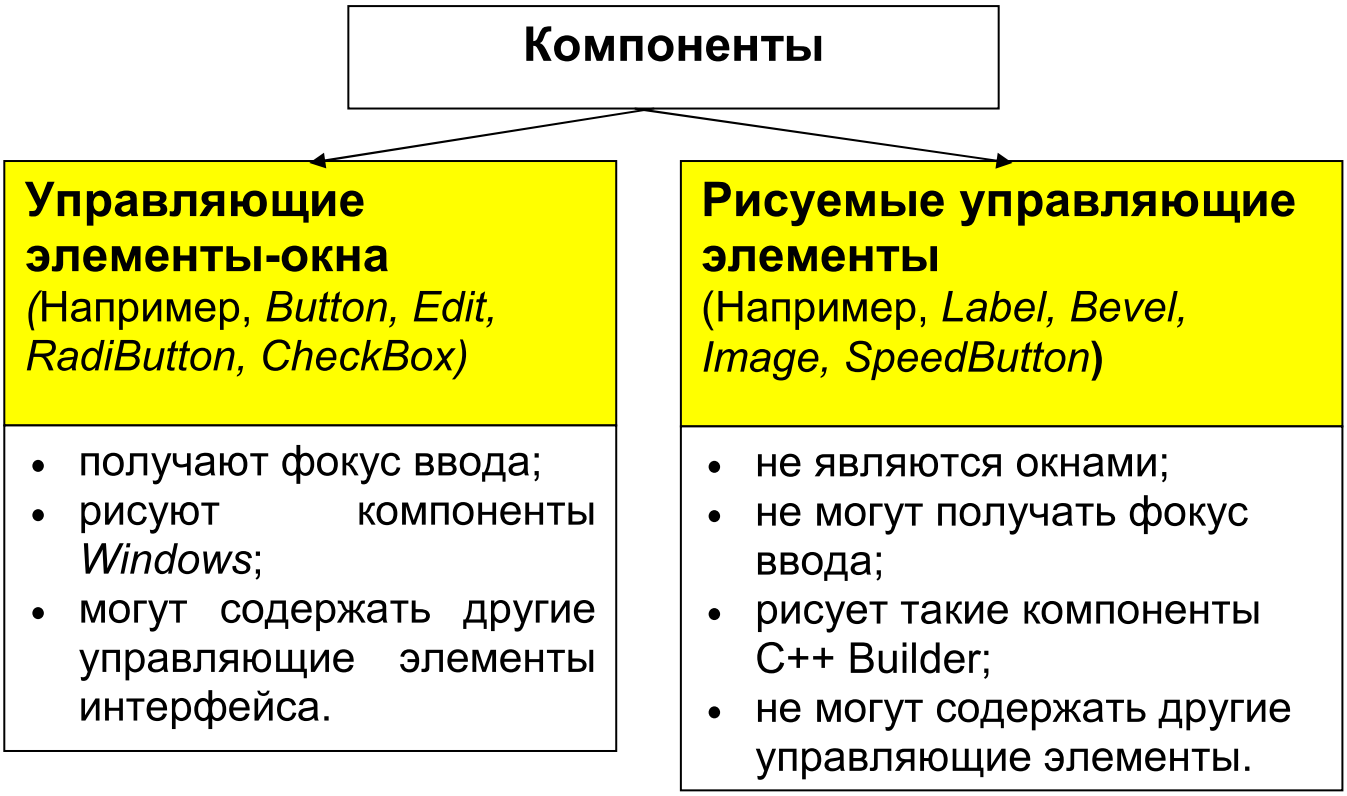
Системные компоненты
TTimer, TFileListBox, TDirectoryListBox, TDriveComboBox, TFilterComboBox, TMediaPlayer

- поддержка OLE
- обмен данными

Классы GDI *(TCanvas., TBrush, TBitmap, TFont)*

Служебные классы *(TIniFile, TRegistry, TRegkeylnfo, TRect, TPoint, TStrings, TStringList, TMemo)*

3.8. Компоненты VCL



Компонент = свойства + методы + события

Свойства компонента - это особым образом оформленные методы, предназначенные как для чтения и контролируемого изменения внутренних данных объекта, так и выполнения действий, связанных с поведением объекта. *Свойства* компонента определяют его внешний вид и поведение.

Событие – это связь между некоторым воздействием на компонент и кодом обработчика события, который реагирует на это воздействие, т.е. то, что происходит в результате взаимодействия компонента с пользователем (например, нажатие клавиш на клавиатуре, кнопок мыши) или **Windows**.

Обработчик события (**event handler**) – это метод, который вызывается операционной системой **Windows** в приложении в ответ на событие. Обработчик события пишется прикладным программистом.

Метод является функцией, которая связана с компонентом и которая объявляется как часть объекта.

3.9. Общие свойства компонент

	Свойства	Примечан
1	***Name***	Определяет имя компонента, с которым компонент будет использоваться в программе. По умолчанию имя состоит из имени класса без буквы ‘Т’ и цифры (*Button1*, *Button2* и т.д
2	***Caption***	Заголовок компонента
3	***Color***	Цвет компоненты
4	***Font***	Параметры шрифта
5	***Height, Width***	Размеры компонента, то есть его высота и ширина
6	***Left и Top***	Положение левой и верхней кромки компонента. Для формы задаются в координатах экрана, а для остальных компонент в координатах клиентской части родительского компонента
7	***ClientHeight и ClientWidth***	Размер клиентской части. Для пересчёта относительных координат точки внутри клиентской области в абсолютные координаты экрана можно использовать методы *TControl*: *ClientToScreen* и *ScreenToClient*
8	***Constraints***	Задаёт ограничения на допустимые изменения размеров
9	***Ctl3d***.	*Объёмность. True* означает объёмный вид, а *false* — плоский
10	**Bevel**	Кромка. (внутренняя (BevelInner), внешняя (*BevelOuter*), внутренняя и внешняя (*BevelStyle*)
11	***Border***	*Стиль и ширина бордюра (обводки)*
12	***Enabled***	Определяет возможность активизации компонента. *False* - запрещён для выбора (серый, не активный)
13	***Visible***	Компонент можно сделать невидимым или показать
14	***Cursor***	Изменение формы курсора
15	***Hint***.	Задать Текст подсказки
16	***ShowHint***	Показать Текст подсказки
17	***Parent***.	Родительский компонент — это компонент, на котором располагается данный (Кнопка, метка и т.п.)
18	***Align***	Выравнивание компонент относительно границ своего родителя
19	***Alignment***	Задаёт взаимное расположение надписей компонента (по центру, с левого края, с правового края)
20	***Anchors***	Привязка к родительскому компоненту при изменении размеров
21	***Parent***	Свойства, связывающие особенность оформления или поведения компонента с соответствующей особенностью родителя
22	***ActiveControl***	Определяет, какой из размещённых на ней компонентов будет в фокусе в начале выполнения приложения
23	***TabOrder***	Определяет последовательность табуляции (или фокусировки) компонентов в зависимости от порядкового номера, указанного в качестве значения этого свойства.

3.10. Управление компонентами

1.Размещение нескольких компонент одного типа:

1) Клавиша *Shift* + щелчок по компоненте в палитре компонент.

2) Отпустить *Shift*.

3) Курсор перемещаем на место первой компоненты и щёлкаем левой клавишей мыши.

4) Курсор перемещаем на место второй компоненты и щёлкаем левой клавишей мыши.

5) И так далее столько раз, сколько надо разместить компонент. Поместив последний компонент, необходимо щёлкнуть по кнопке со стрелкой слева в разделе *Standard*.

2. Для нескольких компонент установить некоторое свойство в одно и то же значение или запрограммировать общий обработчик события:

1) Выделяем несколько компонент традиционным способом: *Shift* и щелчок мышью или протянуть нажатую кнопку мыши над выделяемыми компонентами. Если выделяемые компоненты внутри *Panel* или *GroupBox*, то выделение выполняется с клавишей *Ctrl*. Выделить можно и разные компоненты, например, *Edit* и *Label*.

2) После выделения группы компонент в инспекторе объектов отображаются лишь общие свойства и общие события. Активизируем нужные свойства и устанавливаем их значения или для некоторого события программируем общий обработчик.

3) Щелчком вне компоненты снимаем выделение.

3. Выравнивание компонент.

1) Выделить несколько компонент, которые необходимо выровнять.
2) *Edit/ Align* или по правой кнопке мыши команда *Align* для выравнивания размещения.
3) *Edit / Size* или по правой кнопке мыши команда *Size* для выравнивания размеров.
4) Устанавливаем параметры выравнивания.

4.Копирование или перемещение компонент с одной формы на другую или внутри одной и той же формы:

1) Выделить одну или несколько компонент.
2) *Edit/ Copy (Ctrl + C*) или *Edit/ Cut (Ctrl+X)*.
3) Активизируем другую форму или остаёмся на этой же форме.
4) *Edit/ Paste (Ctrl+V).*

5.Поиск компонент перед размещением, если не знаем, в каком разделе он находится.

1) View / Component List.
2) Вводим имя компоненты полностью или хотя бы несколько начальных букв или выбираем из предложенного списка.
3) Двойным щелчком мыши или с помощью кнопки *Add to form* найденный компонент можно разместить на форму.

Для ***поиска пропавших компонент*** в случае, если некоторая компонента, например, закрыта другой компонентой или не видна по другой причине, необходимо выбрать её имя в списке вверху окна Инспектора Объектов. В любом случае, даже если компонент закрыт, появится рамка с маркерами. При необходимости верхние компоненты можно переместить.

Для ***фиксации компонент*** используется команда *Edit / Lock Controls*. Тогда во время проектирования компоненту нельзя перемещать. Повторное выполнение команды разблокирует фиксацию.

Перечень вопросов по теме 3

1. Библиотеки классов. Особенности, назначение, области применения.
2. Библиотека OWL. Сильные и слабые стороны. Основные категории классов библиотеки OWL
3. Библиотека MFC. Назначение и особенности, преимущества и недостатки. Взаимосвязь с иными библиотеками классов.
4. Структура библиотеки визуальных компонент. Иерархия классов библиотеки VCL.
5. Основные классы библиотеки **VCL** и их назначение. Классификация компонент VCL.
6. Свойства, события и методы компонент. Общие понятия, определения, назначение.
7. Стандартные (базовые) компоненты библиотеки VCL. Основные свойства компонент, особенности и назначение.
8. Классы компонентов. Стандартные компоненты и класс Additional. Компоненты доступа к базам данных.
9. Специальные элементы управления. Возможности, назначение, перечень.
10. Системные компоненты. Свойства, назначение и применение**.**
11. Графические компоненты. Их назначение, особенности и использование.
12. Какие компоненты библиотеки **VCL** предназначены для загрузки стандартных диалогов открытия и сохранения файлов? Методы описания, применения, использования.
13. Служебные классы VCl. Краткая характеристика и назначение.
14. Способы прикрепления «горячих клавиш» при проектировании пунктов меню. Стандартные «горячие клавиши». Важность соблюдения стандартов при разработке Меню пользователя.
15. Какой компонент следует использовать для загрузки в него изображений? Как в приложении с несколькими формами задать главную форму? Как получить информацию о том, сколько форм имеется в приложении?

Тема 4.

Техника визуального программирования. Методика использования компонентов в интегрированной среде визуальной разработки ***C++ Builder****.*

4.1. Версии *C++ Builder*

Год	Версия
1997-2002	v.3, v.4, v.5, v.6
2003	Borland *C++BuilderX* (CBX)
2004	развитие C++ Builder, объединение с Delphi
2005	*Borland Developer Studio 2006, Borland C++ Builder 2006*, Delphi, C++ и C# Builder
2007	CodeGear *C++ Builder 2007*, поддержка Windows Vista
Сент. 2008	CodeGear RAD Studio, Delphi 2009 и C++Builder
Авг.2009	RAD Studio + C++ Builder 2010 PB, FastReport
2010	RAD Studio XE (C++ Builder XE)
2011	CodeGear C++Builder («Com-modore»), поддержка x86-64 и машинный x86-64 код. (XE2)

C++ Builder — интегрированная среда программирования (IDE), система, предназначенная для быстрой разработки приложений (Rapid Application Development, RAD) в среде операционной системы Windows, поддерживающая основные принципы объектно-ориентированного программирования, использующая спецификации и ключевые слова в стандарте языка C++.

C++ Builder содержит библиотеку визуальных компонент (VCL), компилятор, отладчик, редактор кода и прочие инструменты, которые позволяют строить сетевую архитектуру, клиент/серверные приложения, создавать собственные компоненты, работать с базами данных; обеспечивают поддержку систем Internet и телефонии.

4.2. Особенности IDE C++ Builder

Позволяет создавать:

- собственные компоненты;
- консольные приложения Win32;
- использовать интерфейс GUI;
- вносить изменения в процессе отладки;
- сетевую архитектуру;
- клиент/серверные приложения;
- поддержку систем Internet;
- модули для телефонии;
- графические связи и объекты баз данных.

Содержит:

- редактор интерфейса;
- структурированную Палитру компонент;
- комплекс объектных библиотек;
- Библ. визуальных компонент (VCL);
- Инспектор объектов.

Поддерживает:

- основные принципы ООП;
- спецификации и ключевые слова в стандарте языка C++.

Обеспечивает:

- высокое быстродействие;
- полное описание компонентов;
- связь с различными БД

dBASE и Paradox: Sybase, Oracle, InterBase и Informix; Excel, Access, FoxPro и Btrieve.

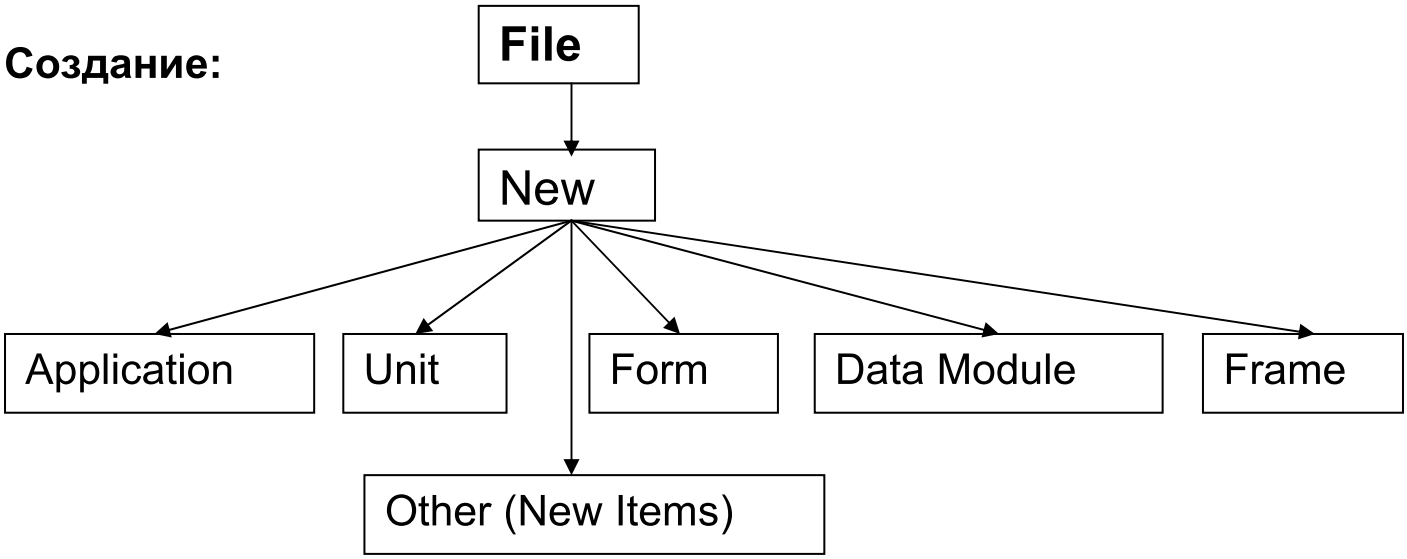
4.3. Обеспечение скорости проектирования программ в *C++ Builder* достигается за счет использования:

Свойства компонента отображаются на странице свойств (**Properties**) инспектора объектов среды ***C++ Builder*** и подразделяются на опубликованные (**published**) и общие (**public**) свойства компонентов, которые доступны только во время выполнения приложения.

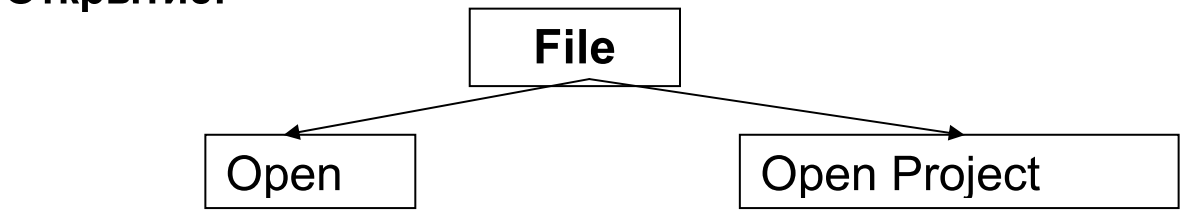
Страница событий (**Events**) инспектора объектов показывает список событий, распознаваемых компонентом.

4.4. Создание и открытие файла проекта C++ Builder

Создание:



Открытие:

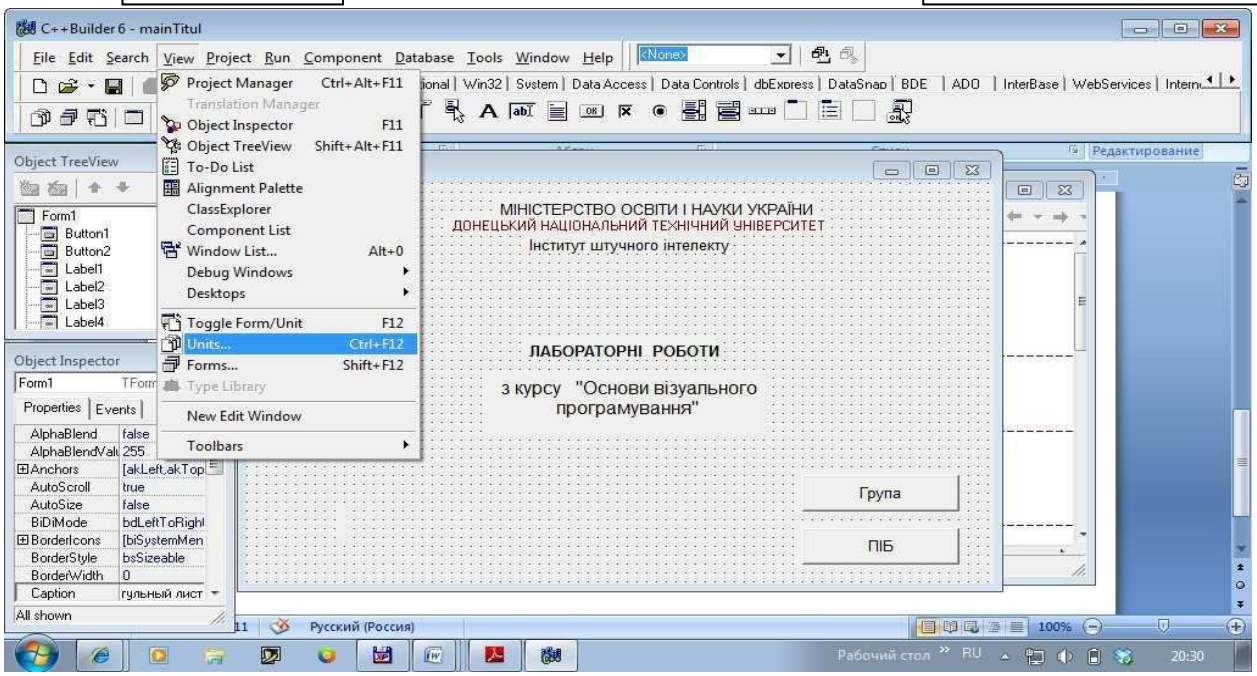


Проект – это набор файлов, которые используются при создании автономного исполняемого файла. Он содержит информацию, необходимую для построения работающего приложения.

Включает формы, модули с их заголовочными файлами и файлами реализации, установки параметров проекта, ресурсов и т. д.

4.5. Просмотр модуля (формы) C++ Builder

View → Units (Forms)



4.6. Расшифровка головного файла проекта

#include <vcl.h>

файл **vcl.h**, содержит объявления, используемые в VCL

USERES и USEFORM,

содержат имена файлов ресурсов проекта и реализации модуля, автоматически создаваемые средой.

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

WinMain – главная функция программы.

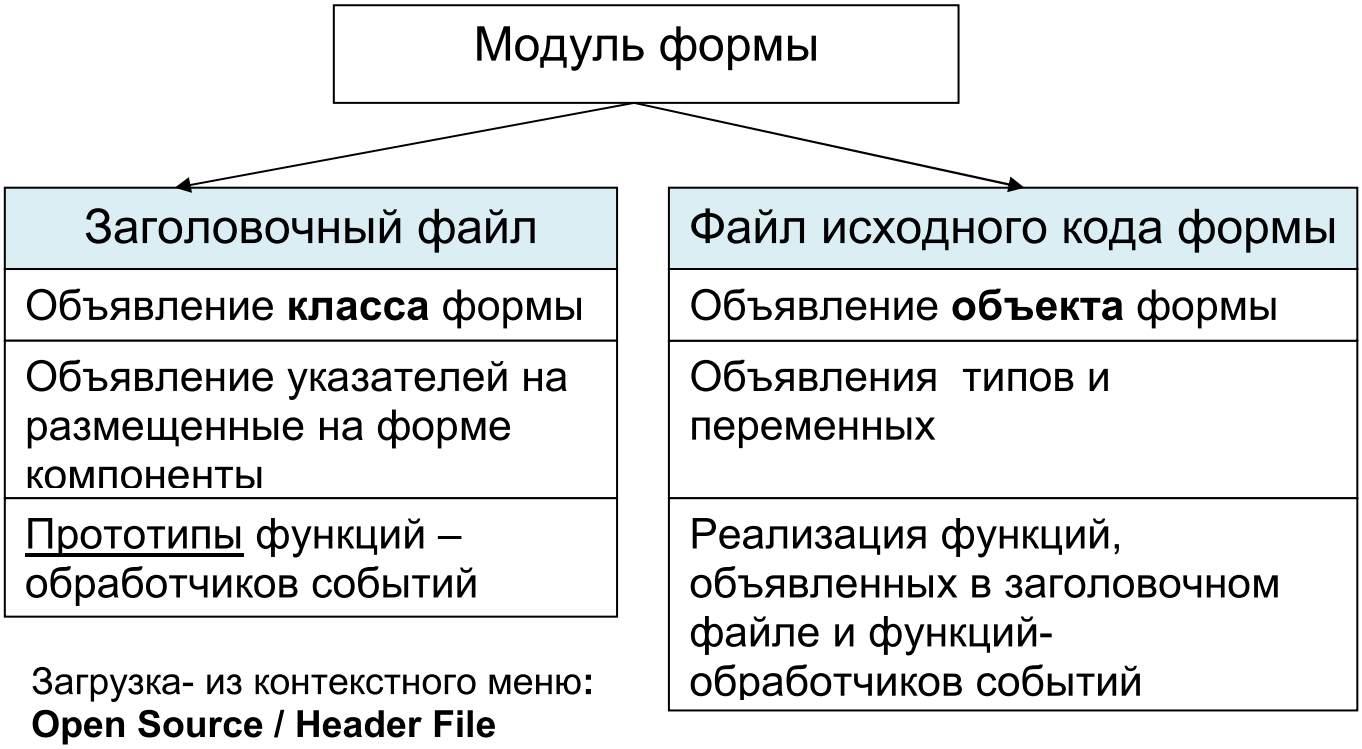
Для анализа в приложении параметров командной строки:

WINAPI WinMain (HINSTANCE, HINSTANCE, LPSTR S, int)

в **S -** текст командной строки.

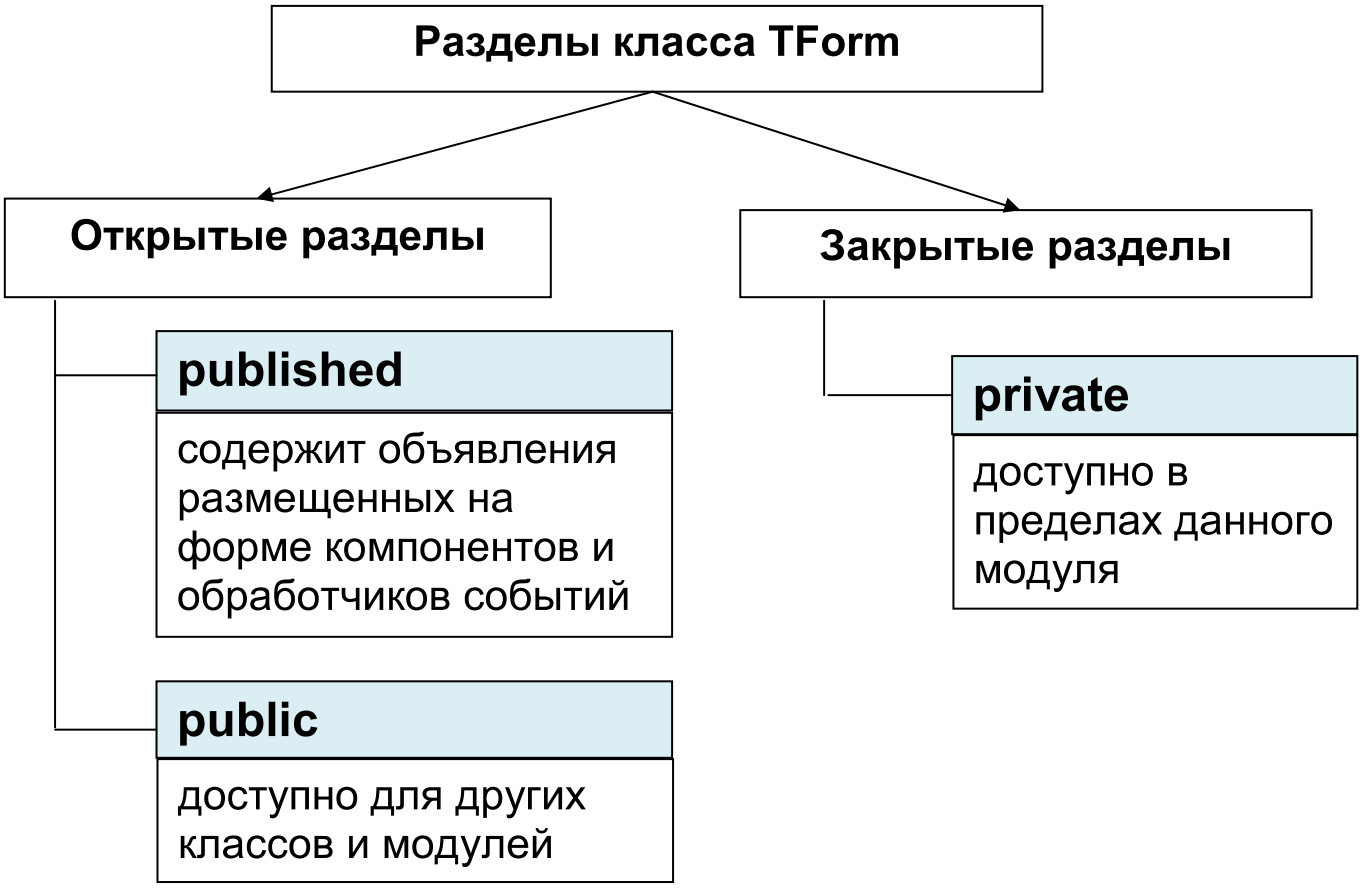
Главный файл проекта – это файл с расширением **.cpp (mainTitul).** Он содержит главную функцию **WinMain**, с которой начинается выполнение программы.

4.7. Структура файла модуля формы



Загрузка- из контекстного меню:
Open Source / Header File

4.8. Описание класса формы



4.9. Файлы, автоматически создаваемые средой C++ Builder

Таблица 1 Основные файлы

Файл	Значение
Головной файл проекта (**.cpp**)	Система **C++Builder** создает файл **.cpp** для головной функции **WinMain**, инициирующей приложение и запускающей его на выполнение.
Файл опций проекта (**.bpr**)	Текстовый файл. Содержит установки опций проекта и указания на то, какие файлы должны компилироваться и компоноваться в проект. Файл сохраняется в формате **XML.**
Файл ресурсов проекта **(.res)**	Двоичный файл, содержащий ресурсы проекта: пиктограммы, курсоры и т. п. По умолчанию содержит только пиктограмму проекта. Может дополняться с помощью редактора изображений (**Image Editor**).
Файл реализации модуля (**.cpp**)	Каждой создаваемой форме соответствует текстовый файл реализации модуля, используемый для хранения кода. Можно создавать модули, не связанные с формами.
Заголовочный файл модуля (**.h**)	Каждой создаваемой форме соответствует его заголовочный файл с описанием класса формы.
Файл формы **(.dfm)**	Двоичный или текстовый файл, который среда **C++Builder** создает для хранения информации о формах. Каждому файлу формы соответствует файл модуля (**.cpp**) .
Заголовочный файл компонента (**.hpp**)	Файл создается при создании нового компонента или подключается к проекту из библиотеки компонентов **C++ Builder.**
Файл группы проектов (**.bpg**)	Текстовый файл для группы проектов
Файлы пакетов (**.bpl** и **.bpk**)	Двоичные файлы, которые используются при работе с пакетами: **.bpl** – файл проекта; **.bpk** – файл, определяющий компиляцию и компоновку пакета.
Файл рабочего стола проекта (**.dsk**)	Текстовый файл, в котором хранится информация о последнем сеансе работы с проектом. Файл создается только в случае, если включена опция **Autosave options / Project desktop.**
Файлы резервных копий (**.~bp**, **.~df**, **.~cp**, **.~h**)	Файлы резервных копий для файлов проекта, формы, реализации модуля и заголовочного файла. Изменив расширения этих файлов можно вернуться к предыдущему варианту проекта.

Таблица 2. **Группа файлов, создаваемых компилятором**

Файл	Значение
Файл (**.exe**)	Исполняемый файл приложения. Является автономным исполняемым файлом, для которого больше ничего не требуется, если только не используются библиотеки, содержащиеся в пакетах **DLL**, **OCX** и т. д.
Объектный файл модуля (**.obj**)	Откомпилированный файл модуля (**.cpp**), который компонуется в окончательный исполняемый файл.
Динамически присоединяемая библиотека (**.dll**)	Файл создается в случае, если проектируется своя собственная **dll.**
Файл таблицы символов (**.tds**)	Двоичный файл, используемый отладчиком в процессе отладки приложения.
Файлы выборочной компоновки (**.il**)	Файлы с расширением, начинающимся с (**.ilc**, **.ild**, **.ilf**, **.ils**), позволяют повторно компоновать только те файлы, которые были изменены после последнего сеанса.

Таблица 3. **Другие файлы, создаваемые средой C++ Builder**

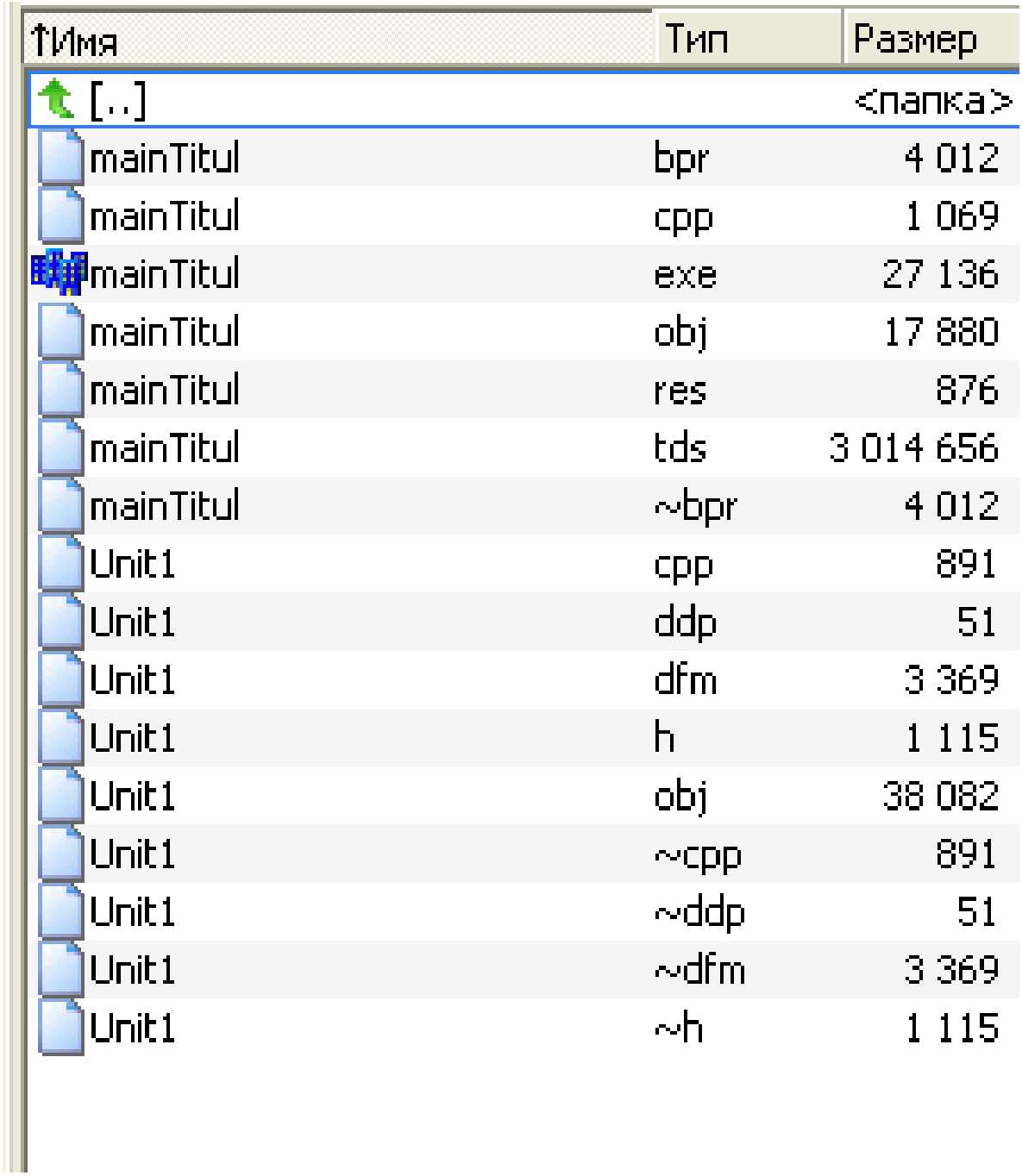
Файл	Значение
Файлы справки (**.hlp**)	Стандартные файлы справки которые могут быть использованы приложением **C++Builder**.
Файлы изображений или графические файлы (**.wmf**, **.bmp**, **.ico**)	Эти файлы обычно используются в приложениях **Windows** для создания привлекательного и дружественного пользовательского интерфейса.

Важнейшими являются файлы с расширением **.cpp**, **.h**, **.dfm**, **.bpr**, **.res**. Это те файлы, которые переносятся на другой компьютер, если хотите продолжить на нем работу над своим проектом.

Информация о формах системы **C++ Builder** хранится в трех файлах: **.dfm**, **.cpp** и **.h**.

Вспомогательные файлы имеют расширения **.obj**, .**res**, **.tds**, **.il?**, **.~***. Если в настоящий момент работа с данным проектом не ведется, то их полезно удалять. Особо обратите внимание на файлы **.tds**, объем которых может быть очень большим.

4.10. Пример экрана с файлами проекта C++ Builder



Имя	Тип	Размер
[..]		<папка>
mainTitul	bpr	4 012
mainTitul	cpp	1 069
mainTitul	exe	27 136
mainTitul	obj	17 880
mainTitul	res	876
mainTitul	tds	3 014 656
mainTitul	~bpr	4 012
Unit1	cpp	891
Unit1	ddp	51
Unit1	dfm	3 369
Unit1	h	1 115
Unit1	obj	38 082
Unit1	~cpp	891
Unit1	~ddp	51
Unit1	~dfm	3 369
Unit1	~h	1 115

Примечание. Если в вашем приложении несколько модулей, сохраняйте их файлы под какими-то осмысленными именами, изменяя тем самым имена модулей, заданные **C++ Builder** по умолчанию, так как работать с осмысленными именами проще.

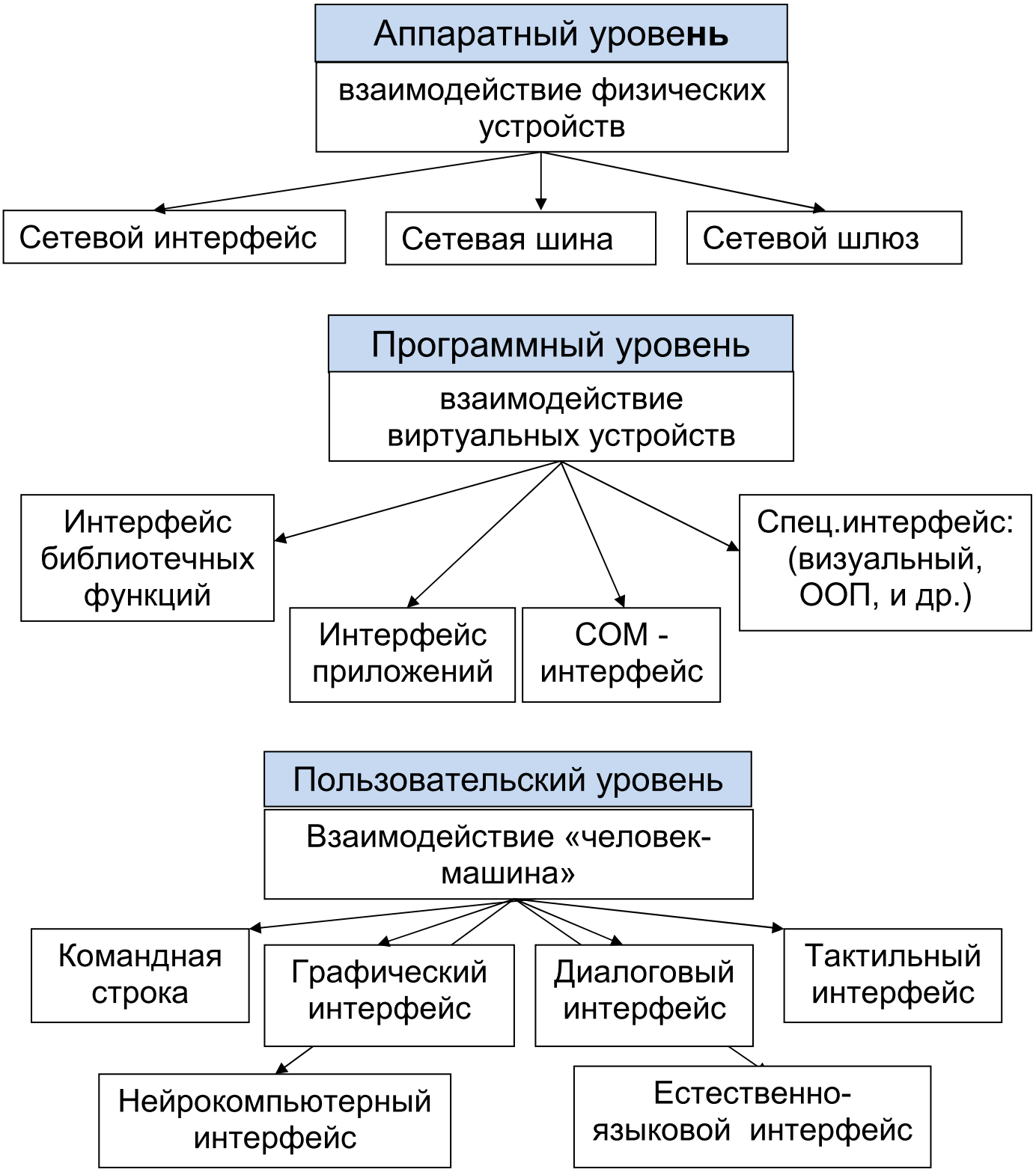
Перечень вопросов по теме 4

1. Охарактеризуйте систему **C++ Builder** – ее особенности, преимущества и недостатки.
2. Расскажите историю развития и особенности IDE C++ Builder. Укажите основные инструменты, и понятия, используемые в этой среде.
3. За счет чего достигается увеличение скорости разработки программ в визуальной среде?
4. Что такое проект системы **C++ Builder**? Перечислите способы, с помощью которых можно создать новый проект.
5. Как выполнить сохранение файлов проекта приложения? Возможно ли самостоятельное назначение имен для файлов проекта?
6. По какому принципу формируются имена файлам проекта. В каких случаях и по какому принципу следует формировать имена файлов проекта. Когда следует оставлять имена, присвоенные файлам средой программирования?
7. Что такое головной файл проекта? Опишите его особенности, назначение, место расположения, обязательно используемые объявления и функции. Какое имя система **C++ Builder** присваивает головному файлу проекта?
8. В каком файле находится определение класса формы? Как можно просмотреть содержимое этого файла? Опишите структуру файла модуля формы.
9. В каком файле находятся функции – обработчики событий?
10. Какие файлы автоматически создаются средой **C++ Builder** при разработке проекта. Укажите их назначение. Какие из них можно удалить после разработки приложения?
11. Опишите способ быстрого перехода от просмотра формы к просмотру файла реализации модуля
12. Как открыть существующий проект? В чем особенности меню Open и Open Project.
13. Укажите назначение и перечислите особенности главной формы приложения.
14. Опишите функциональные возможности инспектора объектов в визуальном программировании.
15. Что такое палитра компонент в визуальном программировании.

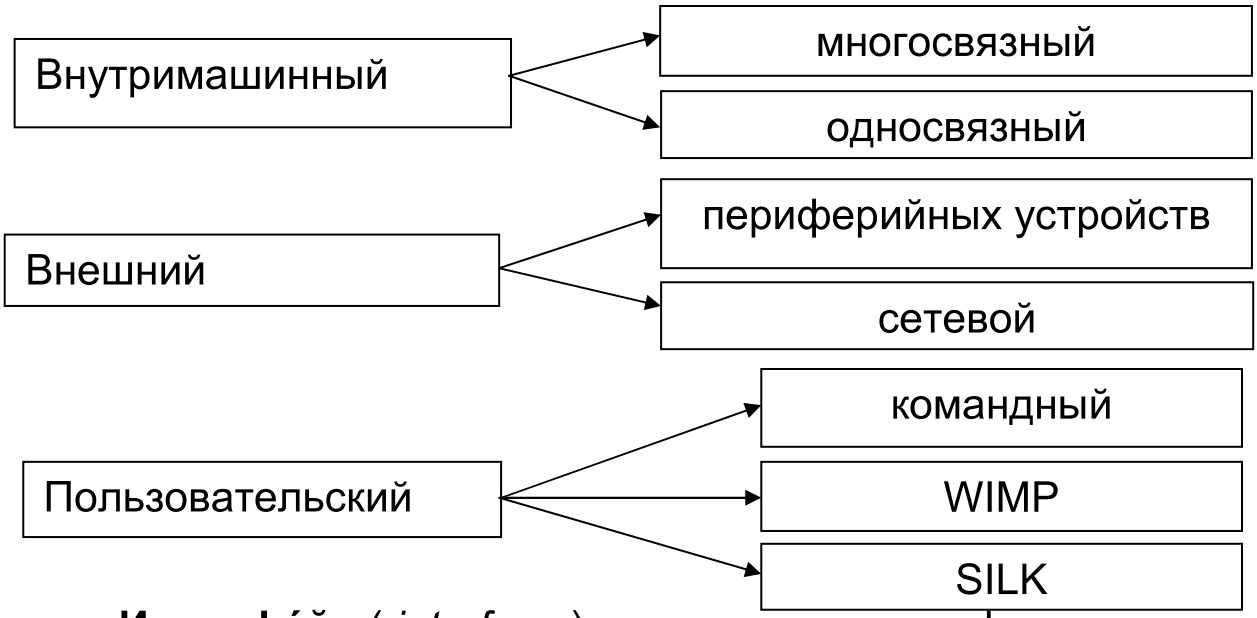
Тема 5

Основы проектирования интерфейсов. Понятие интерфейса. Классификация интерфейсов. Компоненты интерфейса. Интерфейс пользователя.

5.1. Способы организации интерфейсов



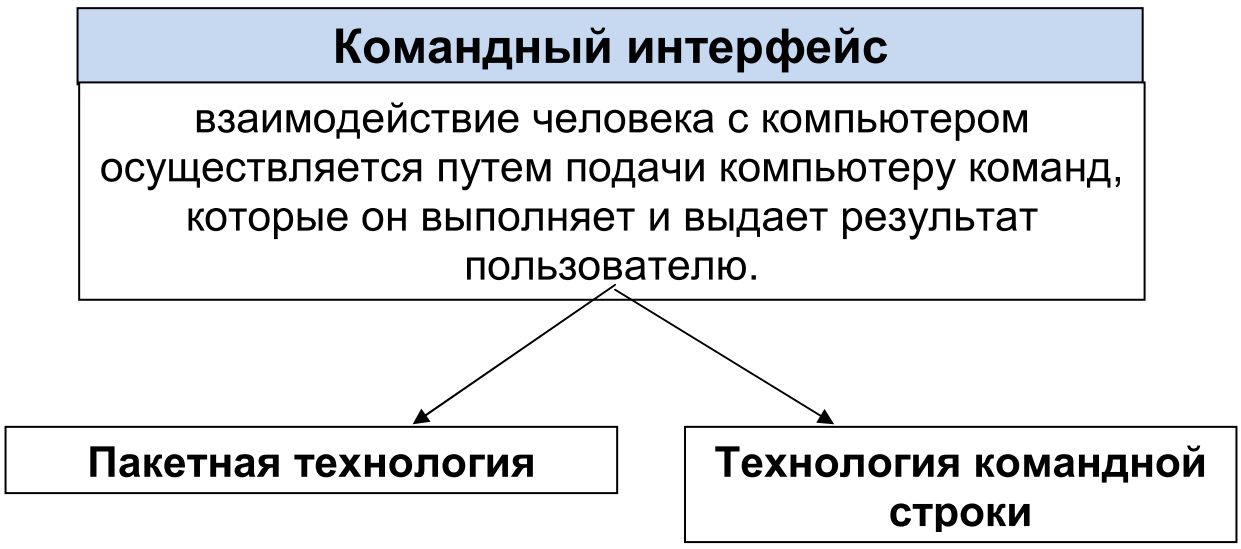
5.2. Виды интерфейсов



Интерфе́йс (*interface*) - совокупность унифицированных технических и программных средств и правил (описаний, соглашений, протоколов, методов), обеспечивающих взаимодействие в вычислительной среде, основа всех современных информационных систем, осуществляемая на пользовательском, программном или аппаратном уровнях.

Интерфейс прикладного программирования или Интерфейс программирования приложений (англ. *application programming interface, API)* — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.

5.3. Командный интерфейс



5.4. WIMP –интерфейс

WIMP (window, image, menu, pointer)
ООМУ (окно, образ, меню, указатель)

Команды машине подаются через графические образы.

Простой графический интерфейс

- цвет, мерцание;
- широкий курсор;
- клавиши- курсор.
- манипуляторы типа мышь;
- NC, MEdit, ChiWriter, MsWord для DOS, Лексикон и др.

Полный WIMP-интерфейс

- окна;
- иконки;
- меню;
- манипулятор типа мышь.

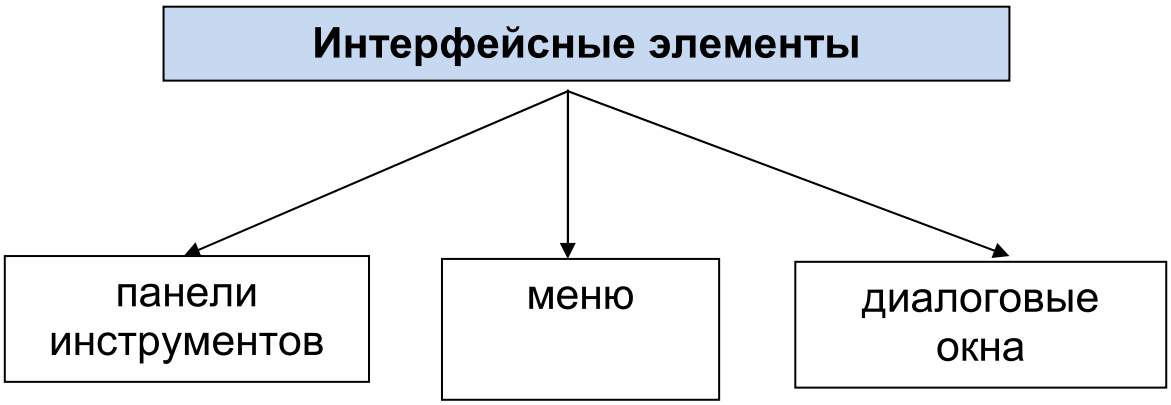
5.5. SILK-интерфейс

SILK (speech,image,language,knowledge)
РОЯЗ (речь, образ, язык, знания)

разговор человека и компьютера

- речевая технология
- биометрическая технология
- семантический (общественный) интерфейс

5.6. Разработка интерфейса пользователя



5.7. Компоненты управления интерфейсом:

Наименование	Назначение
MainMenu	компонент для разработки главного меню;
PopupMenu	компонент для создания всплывающего (контекстного) меню;
ProgressBar	индикатор, бегущая строка;
TrayIcon	значок на системной панели;
OpenDialog, SaveDialog	компоненты для разработки диалогов;
Hint, ShowHint	компоненты для формирования текстов подсказок.

Меню́ (*menu*) – важнейший элемент графического интерфейса пользователя, позволяющий выбрать одну из возможных опций программы.

Главное меню располагается под заголовком окна и содержит перечень основных функций, выполняемых в окне приложения.

Контекстное меню вызывается щелчком правой кнопки мыши. Его содержание, как правило, зависит от того, над каким элементом интерфейса был произведен щелчок.

Всплывающая подсказка (**тултип**) – это компонент интерфейса, появляющийся при наведении курсора мыши на элемент.

Многостраничные панели позволяют экономить пространство окна приложения, размещая на одном и том же месте страницы разного содержания.

5.8. Возможности редактора меню C++Builder

- создание главного и контекстных меню;
- обеспечение оперативного доступа к редактору кода для обработки событий **OnClick**, соответствующих пунктам меню;
- вставка меню из шаблонов или файлов ресурсов;
- сохранение пользовательских меню в виде шаблонов.

5.9. Свойства Главной формы

1. Главной форме передается управление в начале выполнения приложения.

2. Закрытие пользователем главной формы означает завершение выполнения приложения.

3. Главная форма может быть спроектирована невидимой (свойство **Visible**), но если все остальные формы закрыты, то главная форма в любом случае становится видимой.

Форма – основа для расположения всех других элементов – компонентов.

Любое приложение имеет как минимум одну форму, которая называется **главной формой** и появляется на экране в момент запуска программы.

Модальная форма - форма, которая приостанавливает выполнение вызвавшей ее процедуры до тех пор, пока пользователь не закроет эту форму. Модальная форма также не позволяет пользователю переключить фокус курсором мыши на другие формы данного приложения, пока форма не будет закрыта.

MDI – это приложение с множеством документов типа **Word**, **Excel** и т. п. В приложении **MDI** имеется родительская (первичная) форма и ряд дочерних форм (называемых также формами документов).

Запуск одной программы из другой называется **порождением дочернего процесса**.

5.10. Способы запуска внешних программ и создания дочерних процессов

1. Запуск с помощью функции **WinExec():**

Запуск стандартной программы **Блокнот:**
WinExec("notepad.exe",SW_RESTORE);

Запуск собственной программы**:**
WinExec("C:\\bcb6_user1\\ Project2.exe", SW_RESTORE);

2.Запуск с помощью функции **ShellExecute():**

Запуск стандартной программы **Блокнот:**
ShellExecute(Handle, "open", "notepad.exe", NULL, NULL, SW_RESTORE);

Запуск собственной программы**:**
ShellExecute(Handle, "open", "C: \\ bcb6_user1 \\ Project2.exe", NULL, NULL, SW_RESTORE);

2.Запуск с помощью функции **CreateProcess():**

Запуск собственной программы**:**
CreateProcess(NULL, "C: \\ bcb6_user1 \\ Project2.exe", NULL, NULL, false, HIGH_PRIORITY_CLASS, NULL, NULL, &StartInfo, &ProcInfo);

Параметр **SW_RESTORE** означает, что окно запускаемого приложения активизируется и отображается на экране.

Handle – дескриптор родительского окна (32-разрядное целое), **open –** выполняемая операция (открыть запускаемое приложение).

В программе, использующей функцию **CreateProcess():**, следует выполнить объявления:

```
// структура, определяющая основное окно дочернего процесса
STARTUPINFO StartInfo = {sizeof(TStartupInfo)};
PROCESS_INFORMATION ProcInfo;
LPCTSTR s;
StartInfo.cb = sizeof(StartInfo);
StartInfo.dwFlags = STARTF_USESHOWWINDOW;
StartInfo. WShowWindow = SW_SHOWNORMAL;
```

Перечень вопросов по теме 5

1. Перечислите основные способы организации интерфейсов.
2. Что включает в себя понятие «Интерфейс объекта».
3. Укажите особенности и типы взаимодействия устройств в интерфейсах аппаратного уровня.
4. Охарактеризуйте интерфейс программного уровня. Особенности и основные разновидности.
5. Особенности человеко-машинного взаимодействия. Нюансы интерфейса пользовательского уровня.
6. Дайте определение интерфейса как основы взаимодействия всех современных информационных систем.
7. Проведите классификацию интерфейсов по видам.
8. Укажите особенности и назначение командного интерфейса.
9. Укажите особенности и назначение WIMP- интерфейса.
10. Укажите особенности и назначение SILK - интерфейса.
11. Укажите особенности разработки интерфейса пользователя.
12. Перечислите и кратко охарактеризуйте интерфейсные элементы.
13. Перечислите и кратко охарактеризуйте основные компоненты управления интерфейсом.
14. Укажите возможности редактора меню C++Builder.
15. Дайте определение понятию «Главная форма приложения». Укажите ее основные свойства.
16. Что такое «Модальная форма». В чем ее основные особенности.
17. Что означает термин «MDI-приложение».
18. Родительские и дочерние формы. Какой процесс называется дочерним? Как выполнить объединение меню родительской и дочерней форм?
19. Укажите способы запуска внешних программ и создания дочерних процессов.

Тема 6

Основные принципы работы с графикой. Методы и свойства графических компонент. Особенности построения диаграмм и графиков в визуальной среде.

6.1.Основные понятия

1. **GDI** (**Graphics Device Interface**, **Graphical Device Interface**).
2. Свойство **Canvas.**
3. Координаты **X** и **Y.**
4. Пиксель.
5. Свойство **Pen** – перо.
6. Свойство **Brush** (кисть).
7. Методы **TextOut** и **TextRect**, функция **API Windows DrawText()** - вывод текста.
8. Метод **Draw()** изображения в памяти.
9. **HRGN MyReg;** - регионы, ограничивающие области.

GDI (**Graphics Device Interface**, **Graphical Device Interface**) — графический интерфейс устройств - один из трёх основных компонентов или «подсистем», составляющих пользовательский интерфейс Microsoft. Windows. GDI отвечает за отрисовку линий и кривых, отображение шрифтов и обработку палитры.

Windows использует **GDI** для вывода всего, что появляется на экране: линий и кривых, отображение шрифтов и обработку палитры, отображение окон, различных панелей и кнопок управления. Практически любая программа для **Windows** так или иначе использует **GDI**.

Свойство **Canvas** (канва, холст) графических компонентов представляет собой область компонента, на которой можно рисовать или отображать готовые изображения. Канва является битовой картой поверхности для рисования на форме графического компонента.

Канва имеет вложенные свойства и методы, представляющие характеристики пера, кисти и шрифта.

6.2. Представление цвета

6.3. Задание цвета 32-битным целым

Код	Цвет
000000	черный
#FFFFFF	белый
#FF0000	ярко-красный
#00FF00	ярко-зеленый
#0000FF	ярко-синий
#FFFF00	желтый
#00FFFF	ярко-голубой
#FF00FF	ярко-малиновый
#808080	темно-серый
#C0C0C0	светло-серый
#800000	темно-красный, кирпичный
#008000	зеленый
#000080	темно-синий
#808000	голубой, Windows
#008080	оливковый
#800080	темно-малиновый

31 … 24	23 … 16	15 … 8	7 … 0
0	**RED**	**GREEN**	**BLUE**

палитра 2^{24} =16 млн. цветов

6.4. Задание цвета в формате RGB

RGB (0, 0, 0) – черный цвет;

RGB(255, 255, 255) – белый цвет;

RGB(255, 0, 0) – красный цвет;

RGB(0, 0, 255) – синий цвет;

RGB(0, 255, 0) – зеленый цвет;

RGB(255, 255, 0) – желтый цвет;

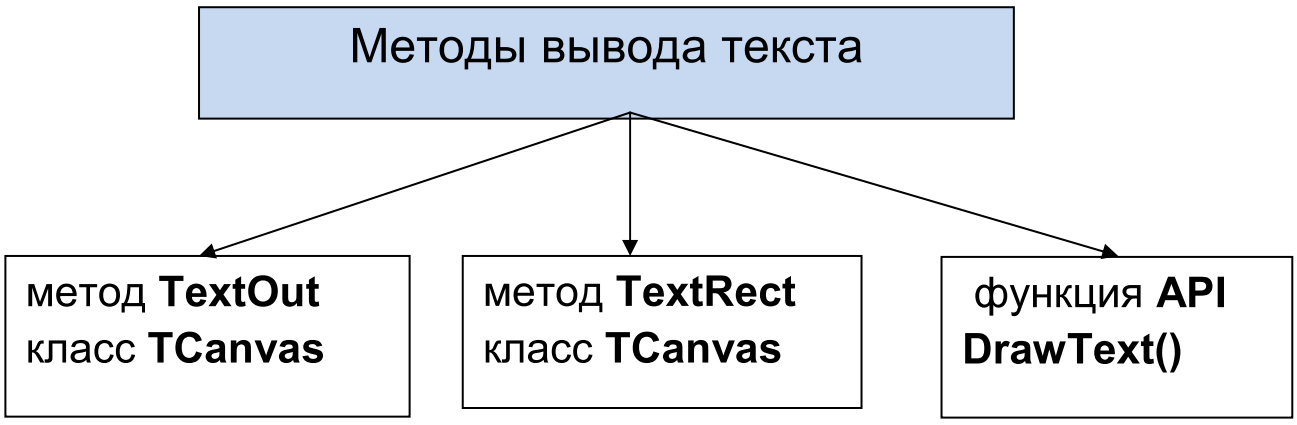
RGB(0, 128, 128) – темно-голубой;

RGB(128, 0, 0) – темно-красный;

RGB(255, 0, 255) – фиолетовый.

Пиксель – это наименьший элемент поверхности рисунка, которым можно манипулировать. Важнейшее свойство пикселя – его цвет.

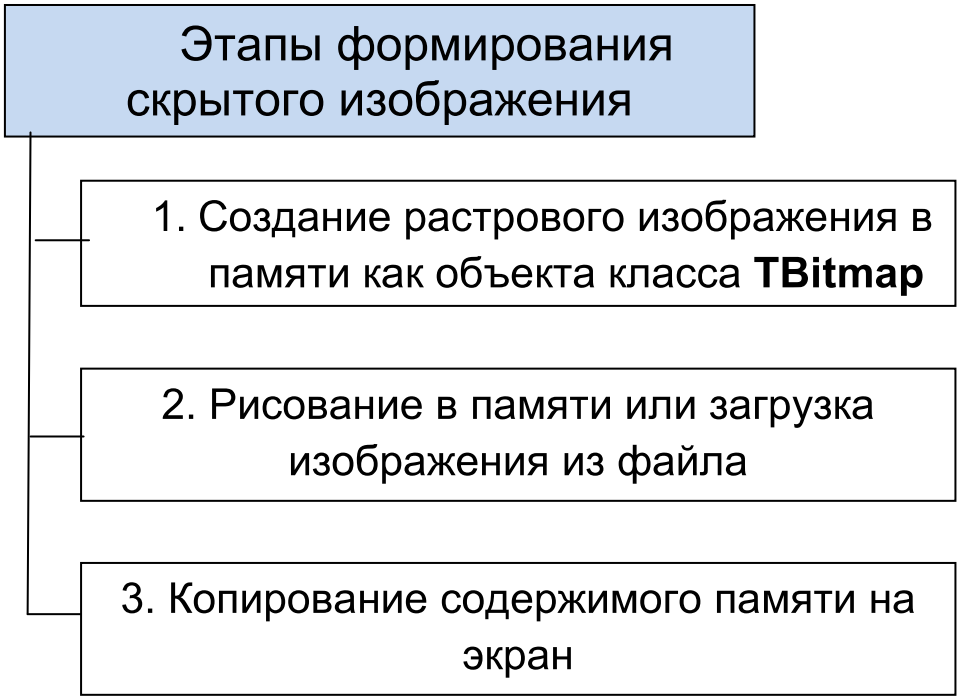
6.5. ВЫВОД ТЕКСТА



Canvas –> TextOut (20, 20, “Вывод текста методом TextOut”);

Canvas –> TextRect(Rect(20, 50, 120, 70),20 ,50, “Этот текст длинный и не будет выведен полностью в ограничивающем прямоугольнике”);

6.6.Скрытые изображения
/offscreen bitmaps, memory bitmaps/



Скрытые изображения (offscreen bitmaps), называемые также изображениями в памяти (memory bitmaps), широко используются в программировании для Windows. Скрытые изображения могут быть нарисованы в памяти или загружены из файла, а затем выведены на экран с помощью метода **Draw()** класса **Tcanvas**.

Скрытые изображения позволяют избежать мерцания, возникающего при выводе на экран большого количества графики за короткий промежуток времени.

Объявление метода **Draw()** класса **TCanvas:**

void _fastcall Draw(int X, int Y, TGraphics *Graphic);

Загрузка изображения из файла в память (метод **LoadFromFile)**:

void __fastcall LoadFromFile (const AnsiString FileName);

6.7.Ограничивающие области.

Области – это участки экрана, которые используются для указания фрагментов канвы, на которых будет выполняться рисование.

Прямоугольная область

Код для обработчика события **OnPaint** для формы:

```
// выделение памяти
Graphics :: TBitmap* bitmap = new Graphics :: TBitmap;
// загрузка изображения из файла
bitmap –> LoadFromFile("handshak.bmp");
// создание прямоугольной области с координатами 50, 50, 250, 250 в памяти
HRGN hRgn = CreateRectRgn (50, 50, 250, 250);
// установка созданного прямоугольника в качестве ограничивающей области
SelectClipRgn (Canvas –> Handle, hRgn);
// рисование в ограничивающей области
Canvas –> Draw(0, 0, bitmap);
delete bitmap;
```

6.8.Непрямоугольная форма

Код для обработчика события **OnPaint** для формы:

```
// выделение памяти
Graphics :: TBitmap* bitmap = new Graphics :: TBitmap;
// загрузка изображения из файла
bitmap –> LoadFromFile("handshak.bmp");
// создание эллиптической области с координатами 50, 50, 250, 250 в памяти
HRGN hRgn = CreateEllipticRgn (30, 30, 170, 170);
// установка созданного эллипса в качестве ограничивающей области
SelectClipRgn (Canvas –> Handle, hRgn);
// рисование в ограничивающей области
Canvas –> Draw(0, 0, bitmap);
delete bitmap;
```

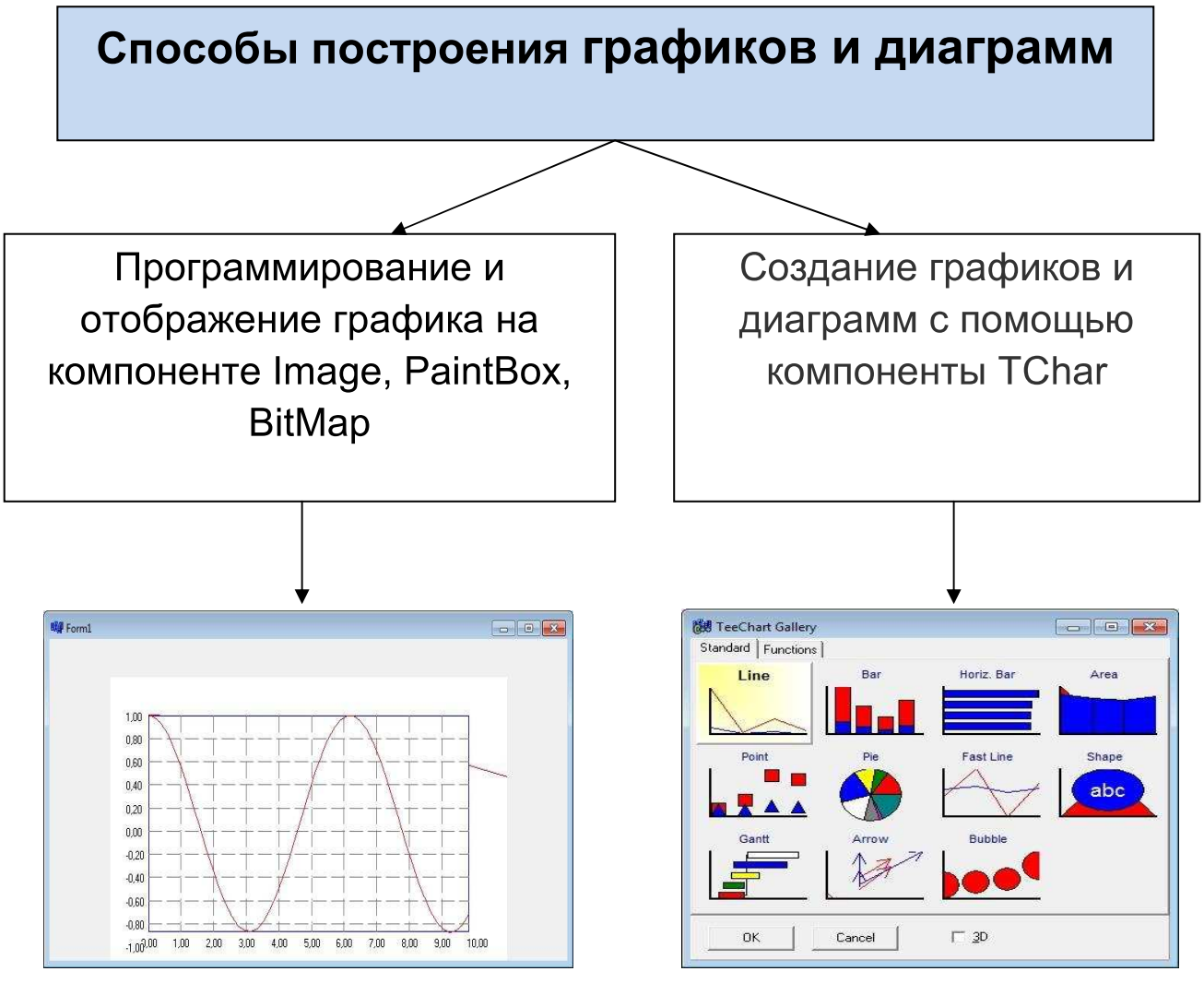
6.9. Ромбовидная ограничивающая область:

```
// 1.Создание массива точек – координат многоугольника
TPoint points [4]= { {80, 0}, {0, 80}, {80, 160}, {160, 80}
      };
// 2. Создание региона
HRGN hRgn = CreatePolygonRgn (points, 4, ALTERNATE);

HRGN MyReg;  - регион,
имя (MyReg) - любое.
```

Регион (HRGN) – это оператор, задающий ограничивающую область непрямоугольной формы.

6.10.Способы построения графиков и диаграмм



6.11. Свойства компонента Chart

Значение	Описание
AllowPanning	Определяет возможность пользователя прокручивать наблюдаемую часть графика с помощью правой кнопки мыши.
AllowZoom	Позволяет пользователю изменять во время выполнения масштаб изображения, вырезая фрагменты диаграммы или графика курсором мыши.
Title	Определяет заголовок диаграммы.
Foot	Определяет подпись под диаграммой. Текст подписи определяется подсвойством Text.
Frame	Определяет рамку вокруг диаграммы.
Legend	Легенда диаграммы - список обозначений.
Margin(~Left,~Top, ~Right,~Bottom)	Значения левого, правого, верхнего и нижнего полей.
BottomAxis, (Left~,Right~)	Определяют характеристики соответственно нижней, левой и правой осей.
LeftWall (Bottom~,Back~)	Определяют характеристики соответственно левой, нижней и задней граней области трехмерного отображения графика.
SeriesList	Список серий данных, отображаемых в компоненте.
View3d	Разрешает или запрещает 3D отображение диаграммы.
View3DOptions	Характеристики трехмерного отображения.
Chart3DPercent	Масштаб трехмерности.

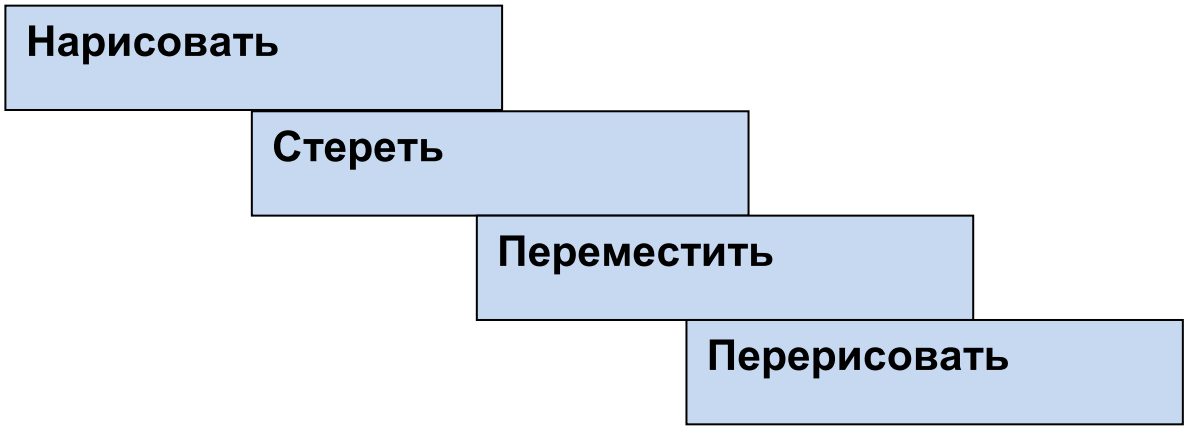
6.12.Порядок создания графика с помощью компонента Chart

1. На форме размещаем компонент TChart (см.Additional).
2. Заносим код в обработчика событий кнопки «Старт».
3. Клик правой кнопкой на компоненте TChart, размещенном на форме, выбор Edit Chart.
4. На вкладке Chart на страничке Series нажимаем Add и выбираем вид графика, который нужен.
5. Закрываем окно компонента Chart, компилируем приложение.
6. Задаем значения точек графика (в диалоге или готовим файл данных).

6.13.Алгоритмы анимации

Если при перемещении изображения сам битовый образ изображения не меняется, то такие битовые образы называют **статичными.**

Простейший алгоритм:



Динамический битовый образ - битовые образы называют *динамическими,* если происходит не только перемещение изображения, но и изменение самого рисунка. При этом способе создания мультипликации необходимо подготовить ряд файлов, хранящих изображения. В каждом файле изображение должно немного отличаться от предыдущего.

Алгоритм отображения динамического образа:

1. В битовую матрицу загрузить рисунок из файла первого файла.

2. По сигналу от таймера рассчитать местоположение изображения и скопировать из битовой матрицы изображение на канву.

3. По сигналу от таймера стереть изображение и загрузить его из следующего файла.

Перечень вопросов по теме 6

1. С помощью каких функций выполняется отображение графики в **Windows**? Какой интерфейс предоставляет среда **C++ Builder** для вывода графики? Назовите основные графические компоненты.
2. Каким образом в **Windows** кодируется цвет изображения? Какое максимальное количество цветов теоретически можно использовать? Приведите примеры констант, содержащих значения конкретного цвета.
3. Что такое канва? Приведите примеры компонент, имеющих это свойство. Наиболее распространенные свойства класса TCanvas. Наиболее используемые методы класса TCanvas. Какие свойства канвы используются для рисования линий, контуров фигур и их заливки? Какие стили пера и кисти можно использовать при рисовании?
4. Что такое ограничивающие области? Когда их следует использовать при программировании графики? Что означает термин Регион. Необходимость и особенности использования.
5. Скрытые изображения. Какие изображения называются скрытыми? Для чего они используются? Как загрузить скрытое изображение из файла? Как сохранить скрытое изображение в файле?
6. Укажите методы, которые используются для вывода текста на графическом изображении.
7. Что такое битовый образ? Какие битовые образы называют статичными, а какие динамическими? Какие алгоритмы могут быть реализованы для создания мультипликации? Какие из них позволяют предотвратить мигание экрана?
8. Что означает аббревиатура GDI в визуальном программировании. Дайте определение и краткую характеристику понятию.
9. Опишите основные принципы создания движущегося изображения в C++Builder. Приведите последовательность действий для создания статического изображения (алгоритм).
10. Опишите основные особенности и способы построения графиков и диаграмм в C++Builder.
11. Компонент **Chart.** Его возможности, назначение, месторасположение в панели VCL, описание свойств компонента. Работа с окном **Editing Chart.**

Тема 7

Основные принципы разработки приложений баз данных в среде визуального программирования.

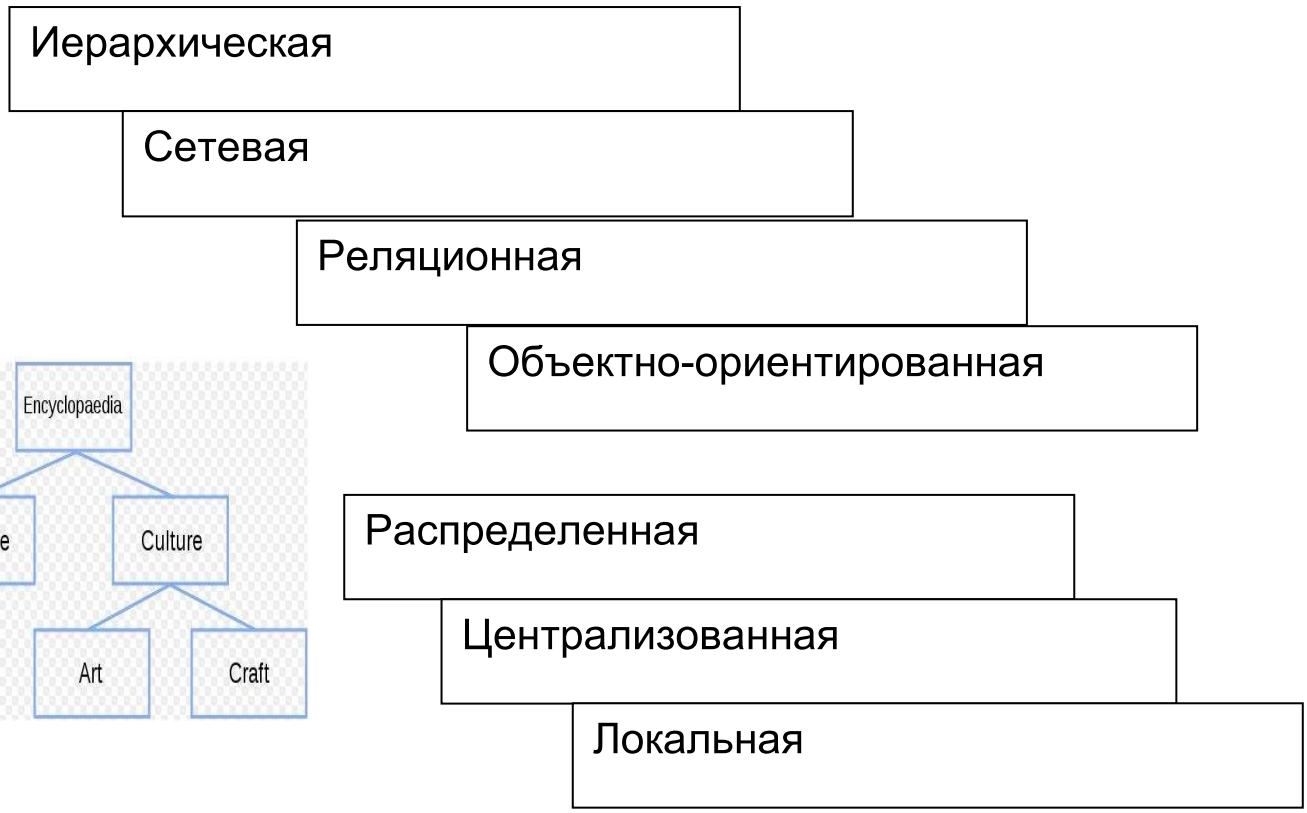
7.1. БД и СУБД

База данных – совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, при этом манипулирование данными выполняют в соответствии с правилами средств моделирования данных.

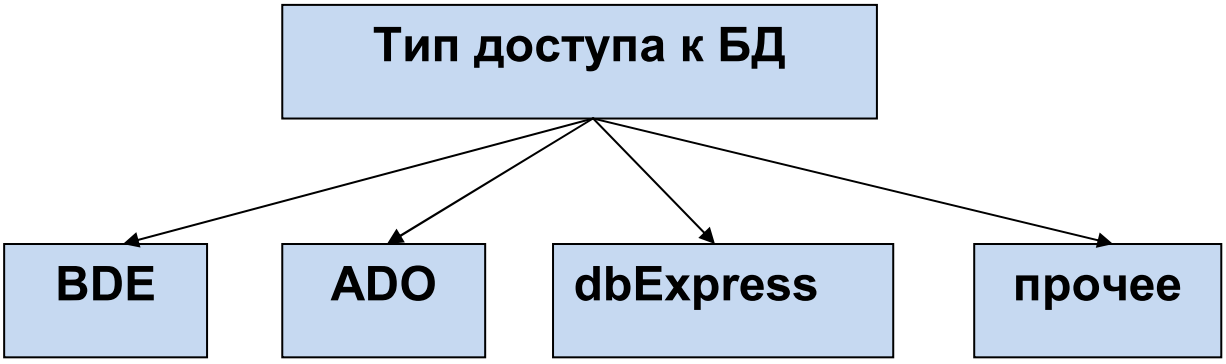
СУБД – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

7.2. Классификация БД

1. По модели данных.
2. По среде постоянного хранения.
3. По содержимому.
4. По степени распределённости.



7.3. Архитектура доступа к базам данных



BDE (**Borland Database Engine**) – набор DLL и вспомогательных файлов, которые обеспечивают доступ к базам данных в C++ Builder и других продуктах фирмы Borland.

Облегчает работу с базами данных, предоставляя высокоуровневый интерфейс прикладного программирования баз данных (database API), одинаковый для всех СУБД, которые он поддерживает.

BDE – механизм доступа к данным, в основе которого лежит ПРОЦЕССОР БД, представляющий собой набор динамических библиотек, драйверов и утилит. BDE обеспечивает работу практически с любой из существующих баз данных.

Недостатком BDE является трудоемкий процесс развертывания приложений, созданных на ее основе, связанный с необходимостью установки и администрирования BDE.

ADO – ActiveX Data Object – разработана Microsoft. Получила наиболее широкое применение.

Преимущество – возможность получения данных из различных источников.

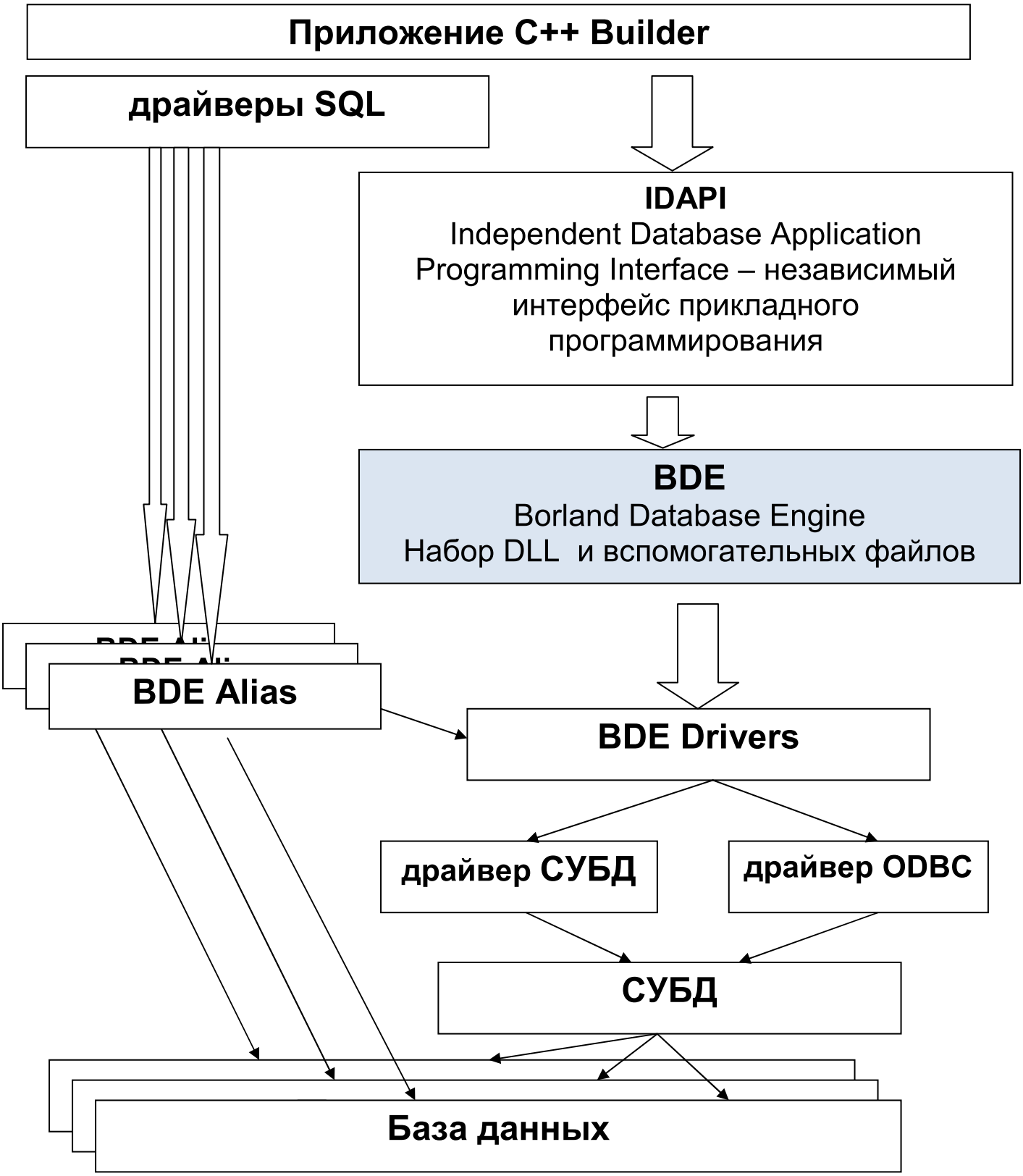
Недостаток ADO – необходимость привязки баз данных для каждого приложения.

dbExpress – разработана Borland. Технология однонаправленного **(unidirect**) доступа к данным.

Преимущество – высокая скорость доступа.

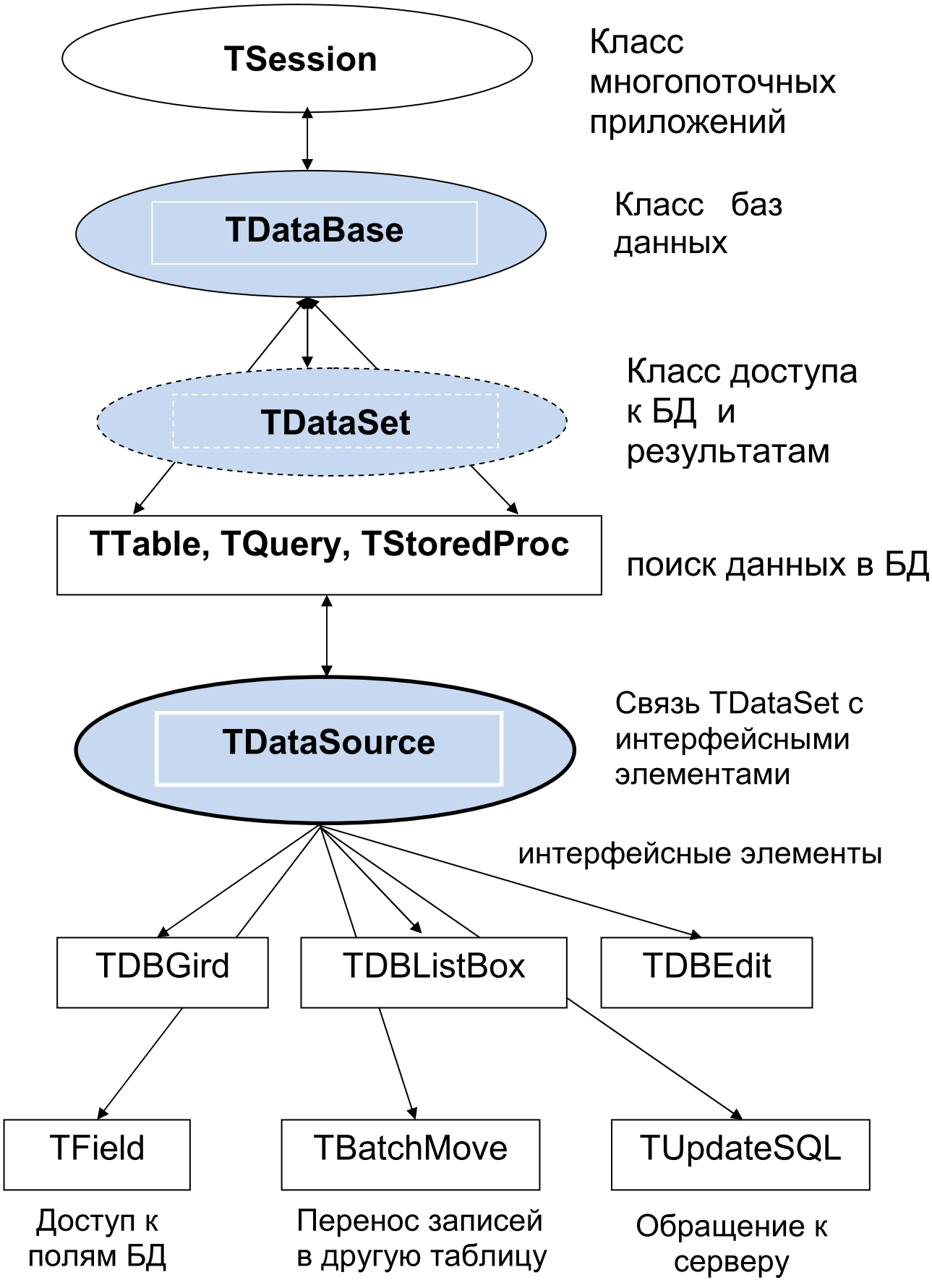
Недостаток – малая распространенность.

7.4. Архитектура доступа к BDE

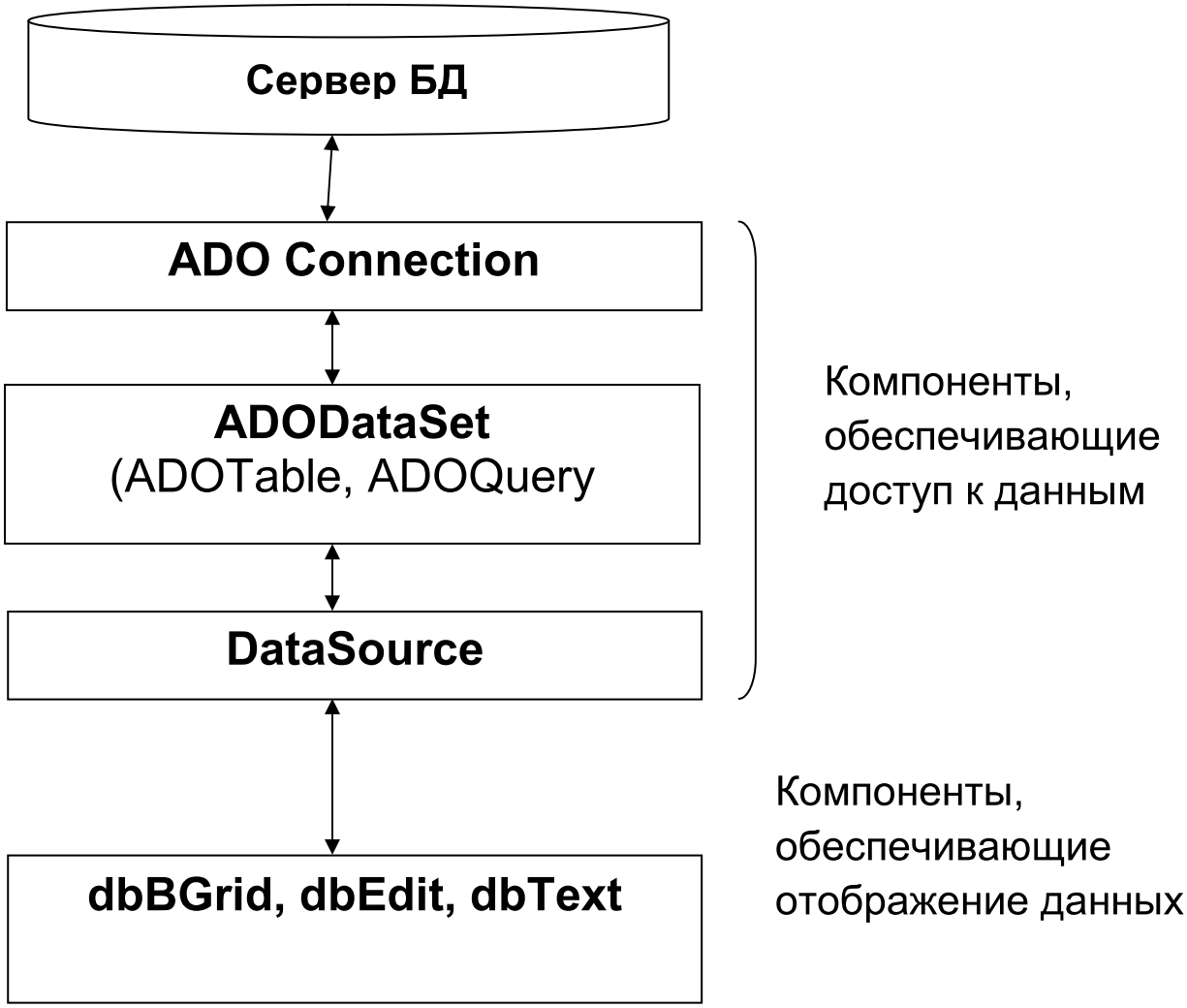


Псевдоним BDE – совокупность конфигурационных параметров, которые сообщают BDE, как обращаться к конкретной базе данных.

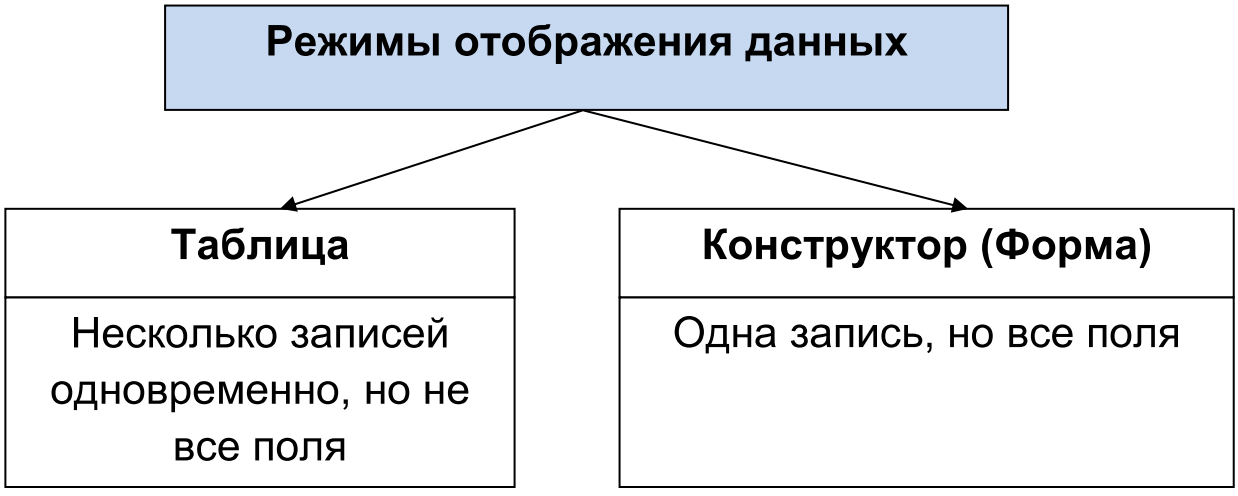
7.5. Доступ к базам данных BDE



7.6. Доступ к базам данных ADO



7.7. Режимы отображения данных



7.8. Порядок настройки работы в режиме таблицы

1.Создать БД (MS Access).

2. Разработать форму, нанести компоненты (ADOConnection, ADODataSet, DataSourse, DBGird).

3. Организовать доступ к данным (настроить компоненты).

4. Обеспечить отображение данных (настроить DBGrid и Columns).

5. Задать класс Oleauto
#include <oleauto.hpp>

6. Создать процедуры обработки событий Activate, Close.

7.9. Порядок настройки работы в режиме конструктора

1.Создать БД (MS Access).

2. Разработать форму, нанести на нее компоненты (ADOConnection,ADODataSet, DataSourse, DBEdit, DBMemo, DBNavigator, Image, OpenDialog)StatusBar,).

3. Организовать доступ к данным (настроить компоненты).

4. Создать процедуры обработки событий Activate, Close,AfterScroll, DBNavigatorClick,ImageClick.

7.10. Настройки ADO-доступа

1. В окне **Object Inspector** выбрать свойство **ConnectionString**.
2. В окне **ConnectionString** нажать кнопку Build.
3. Окно Свойства связи с данными – Поставщик данных – выбрать тип источника данных (для MS Access - **Microsoft Jet 4.0 OLE DB Provider** Microsoft Jet 4.0.
4. Нажать «Далее».
5. На вкладке Подключение задать месторасположение базы данных (файла *.mdb).
6. Проверить подключение к БД.
7. Закрыть окна клавишей «ОК».

Свойства компонента **ADOConnection**

Свойство	Значение
Name	ADOConnection1
ConnectionString	Provider=Microsoft.Jet.OLEDB.4.0; DataSource=C:\.<путь к БД>.\<имя БД>**.mdb;** Persist Security Info=false
LoginPrompt	False (без пароля)
Connected	False (соединено/нет)
Mode	cmReadWrite (cmRead, cmWrite)

Свойства компонента **ADODataSet**

Свойство	Значение	Описание
Name	ADODataSet1	Имя компонента
Connection	ADOConnection1	Ссылка на ADOConnection
CommandText	SELECT Name1,Phone FROM Contacts ORDER BY Name1	Команда серверу
Parameters		Параметры команды
Filter		Установка фильтра
Filtered	true	Признак использования фильтра
Activate	false	Набор данных доступен/нет

Свойства компонента **DataSource**

Свойство	Значение	Описание
DataSet	ADODataSet1	Определяет набор данных, связь с которым обеспечивает комппонент

Настройка DBGrid

1. Разместить DBGrid на форме.
2. Настроить Columns.
3. Настроить DBGrid.

Свойства компонента **DBGrid**

Свойство	Значение	Описание
DataSource	ADODataSet	Ссылка на источники данных
Columns		Отображаемая инф. (столбцы)
BorderStyle		Вид границы вокруг компонента
Options.dgEditing	true	Разрешение на изменение, добавление и удаление данных
Options.dgConfirmDelete	True/ False	Подтверждение удаления записи (true)
Options.dgTitles	True	Разрешение на вывод строки заголовка
Options.dgIndicator	True	Вывод колонки индикатора (*, ∇, $\leftrightarrow$)
Options.dgcolLines	True	Разрешает раздел.линии (колонки)
Options.dgRowLines	True	Разрешает раздел.линии (столбцы)

При этом:
F2- начать редактирование записи
Insert – добавить запись
Ctrl+Del – удалить запись с запросом на удаление
Del – удалить запись (при Options.dgConfirmDelete=false)

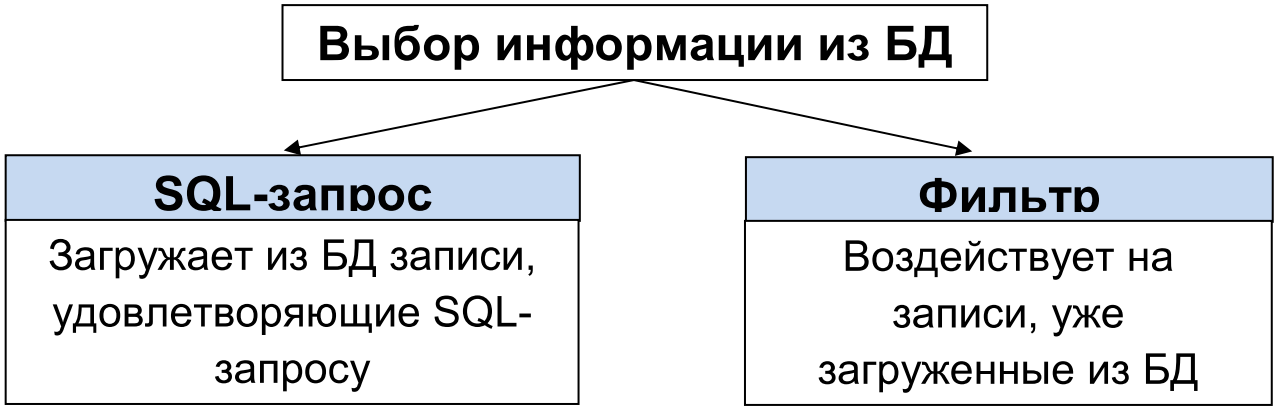
Настройка TColumns

1. Выбрать свойство Columns.
2. Вызвать Columns Editor.
3. Задать элементы (колонки) Columns.
4. Настроить элементы Columns.

Свойства компонента **TColumns**

Свойство	Значение
Font.Name	Arial
Font.Size	9
Columns[0].FieldName	Name1
Columns[0].TitleCaption	ФИО
Columns[0].Title.Width	154
Columns[0].Title.Font.Style	fsBold
Columns[1].FieldName	Phone
Columns[1].TitleCaption	телефон
Columns[1].Title.Width	164
Columns[1.Title.Font.Style	fsBold

7.11. Выбор информации из БД



SELECT <a> **FROM** <b> **WHERE** <c> **ORDER BY** <d>
a – Список полей БД (a=*) - все поля)
b – Имя таблицы из БД
c – Критерий (условие) отбора записей
d – Список полей для упорядочения записей

Примеры:

SELECT Name1, Phone **FROM** Contacts2 **Where** Name1='Петров'
SELECT * **FROM** Contacts2 **Where** Name1 **LIKE** '%ов%'
SELECT * **FROM** Contacts2 **Where** Name1 **CONTAINING** 'Петр'

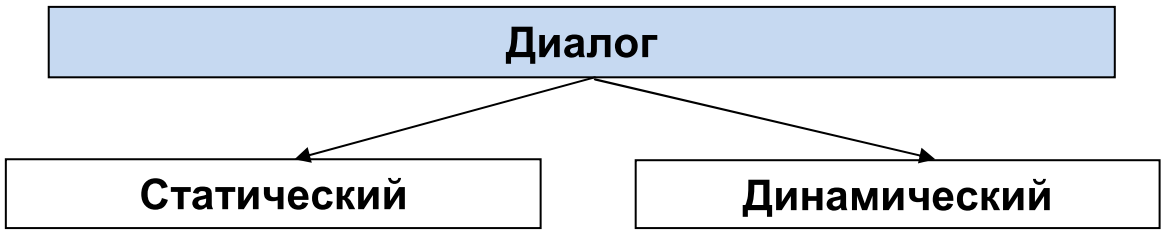
Перечень вопросов по теме 7

1. Дайте определение понятиям «База данных» и «Система управления базами данных». Приведите краткую характеристику и особенности наиболее известных их них.
2. Укажите принципы классификации баз данных. Приведите описание реляционной базы данных, ее основные термины и понятия.
3. Раскройте понятие «Архитектура доступа к базам данных». Какие типы доступа знаете. Охарактеризуйте доступ BDE, его преимущества и недостатки.
4. Раскройте понятие «Архитектура доступа к базам данных». Какие типы доступа знаете. Охарактеризуйте доступ ADO, его преимущества и недостатки.
5. Опишите особенности и назначение компонент ADO Connection, ADODataSet, DataSource.
6. Какие компоненты обеспечивают отображение информации из баз данных. Их особенности, основные свойства, использование.
7. Опишите основные принципы проектирования баз данных. Укажите преимущества и недостатки проектирования в режиме таблицы и в режиме формы (конструктора).
8. Применение SQL-запросов для выборки данных из баз данных в среде C++Builder. Оператор SELECT. Его форматы, параметры, особенности.
9. Опишите принципы и понятия связи двух таблиц при работе с базами данных. Какие компоненты C++Builder используются для связывания. Особенности настройки.
10. Компоненты TSession и TDatabase. Для чего предназначен компонент TSession? Какие преимущества дает использование компонента TDatabase в ваших приложениях?
11. Запрос TQuery. Назначение, особенности, условия применения. Какие условия должны быть выполнены для того, чтобы результаты запроса TQuery стали изменяемыми?
12. Интерфейс работы с базой данных. Какой минимальный набор компонентов должно содержать приложение C++Builder, в которое вы хотите включить интерфейсные элементы для работы с базой данных? Перечислите и докажите необходимость применения.
13. Компонента TTable и ее ключевые свойства. Компоненты, необходимые для обращения интерфейсных компонентов к потомкам TdataSet.
14. Компонента TDataSource, ее ключевые свойства, область применения и назначение.

Тема 8.

Основные принципы построения диалогов и справочных систем в среде визуального программирования.

8.1. Основные принципы построения диалогов



Диалог – это обмен информацией между пользователем и приложением.

Статический диалог – обмен информацией осуществляется в режиме ожидания реакции пользователя (не в реальном времени). Преимущественно 80% диалогов статические, к данному типу диалогов относятся офисные приложения, расчётные, информационные и другие.

Динамический диалог – выполняется в реальном времени. Данный тип диалога характеризируется тем, что требует быстрой реакции пользователя. Обмен информацией осуществляется в краткий период времени, пользователь не должен медлить с ответом, или вводом информации. Динамические диалоги в большей мере работают в режиме онлайн, по управлению в динамическом процессе с оборудованием.

8.2. Типы диалогов

1) вопрос-ответ;
2) выбор из меню;
3) заполнения бланков;
4) на основе команд;
5) работы в окнах;
6) по принципу электронной таблицы;
7) гипертекста;
8) приближения к естественному языку;
9) виртуальной реальности.

8.3. Способы отображения справочной информации

Классический	Современный
Hlp - файлы	Chm - файлы
Microsoft Help Workshop	Microsoft **HTML** Help Workshop
hcw.exe, hcrtf.exe	hhw.exe

Классическая справочная система представляет собой набор файлов, используя которые программа Winhelp, являющаяся составной частью Windows, выводит справочную информацию. Основой такой справочной системы являются hlp-файлы.

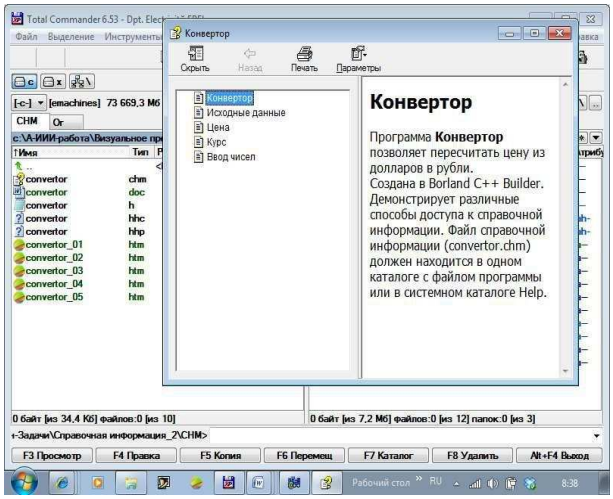
Основой **современной справочной системы** являются chm-файлы. Chm-файл представляет собой компилированный HTML-документ, полученный путем компиляции (объединения) файлов, составляющих HTML-документ, в том числе и файлов иллюстраций.

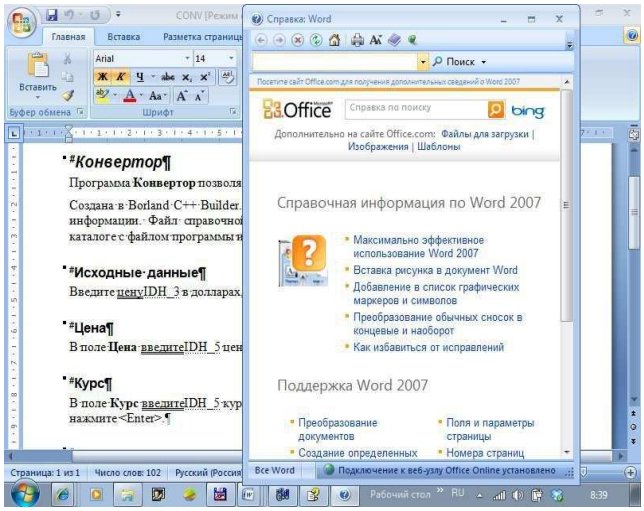
8.4. Порядок создания справочной системы

1.Подготовить справочную информацию

2. Преобразовать справочную информацию в справочную систему

8.5. Экраны справочной системы





Для того чтобы справочная информация появилась на экране в результате выбора в меню «?» команды **Справка,** надо создать функцию обработки события **Onclick** для соответствующей команды меню.

Для того чтобы во время работы программы пользователь, нажав **клавишу <F1>**, мог получить справочную информацию, нужно чтобы свойство **HelpFile** главного окна приложения содержало имя файла справочной системы, а свойство **HeipContext** - числовой идентификатор нужного раздела.

Идентификатор раздела - набор специальных символов, обозначающий отдельный раздел справки, дополненный префиксом и порядковым номером раздела (индекс справочного файла), например, IDH_I, IDH_2. Идентификатор назначается путем вставки перед заголовком раздела справки сноски # (при оформлении в MS Word).

Перечень вопросов по теме 8.

1. Дайте определение понятию «Диалог».
2. Укажите основные принципы построения диалогов.
3. Перечислите известные типы диалогов и дайте им краткую характеристику. Приведите примеры.
4. Укажите способы отображения справочной информации, которые могут быть реализованы в среде визуального программирования.
5. Перечислите типы и виды справок, используемые Windows-приложением. Способы создания справочных файлов.
6. В чем отличие современной и классической справочных систем.
7. Охарактеризуйте особенности справочных файлов и способы их создания в визуальных приложениях. Как реализовать поддержку клавиши F1 для конкретной формы или диалогового окна?
8. Что такое «индекс справочного файла». Когда и каким образом используется это понятие в C++Builder.
9. Что такое «идентификатор раздела» при формировании справочного файла. Когда и каким образом используется это понятие в C++Builder.
10. С помощью каких средств создается справочная система. Этапы разработки справочной системы в C++Builder.

Тема 9.

Принципы построения различных компонентов и порождающих их классов в среде визуального программирования. Разработка компонента программиста.

9.1. Этапы создания собственных компонент

1.Выбор базового класса.

2.Создание модуля для нового компонента.

3.Наследование производным классом для нового компонента уже существующего базового класса библиотеки VCL.

4.Программирование новых свойств, событий и методов.

5.Регистрация компонента в среде C++ Builder.

6. Отладка.

7. Инсталляция компонента на Палитру компонентов.

8. Сохранение файлов компонента.

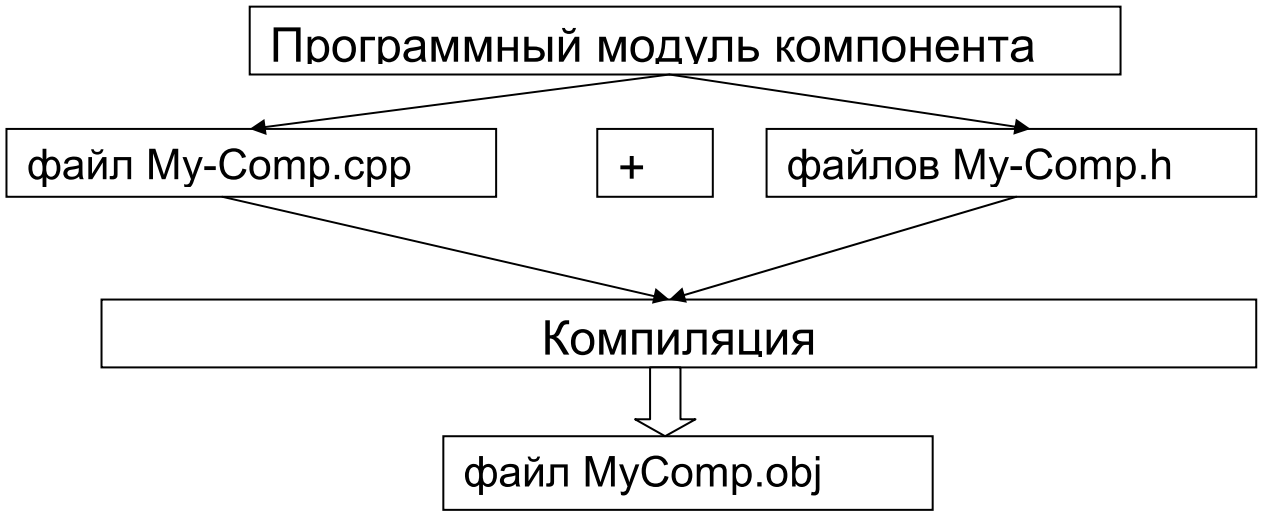
Простейший способ построить новый компонент – это изменить свойства существующего компонента. Используют любой абстрактный класс VCL, в название которого входит слово Custom.

При разработке компонента создают новый модуль, либо модифицируют существующий.

Чтобы создать модуль, выполняют команду File|New и в открывшемся окне New Items выбирают значок Unit.

Чтобы добавить компонент к существующему модулю, выполняют команду File|Open и в открывшемся диалоговом окне выбирают файл имя_модуля.cpp для модуля, в который будет добавлен компонент.

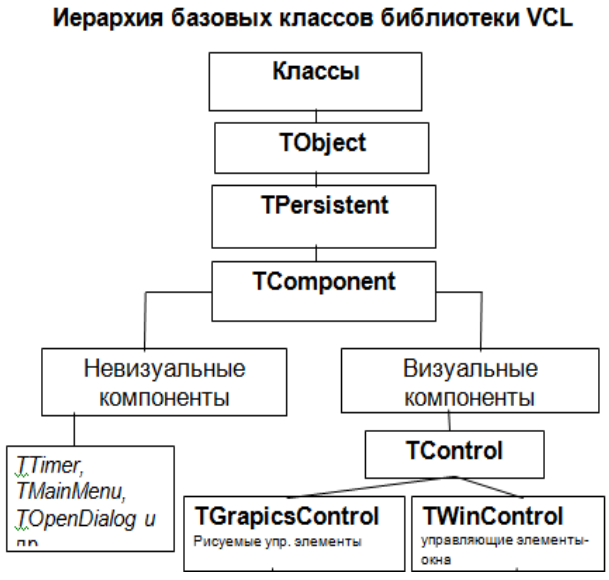
9.2. Создание модуля компонента



Пример заготовки файла MyComp.h:

```
#ifndef MyCompH
#define MyCompH
#include <SysUtils.hpp>
#include <Classes.hpp>
#include <Controls.hpp>
#include <ExtCtrls.hpp>
//class TMyComponent: public <базовый класс компоненты>
// Объявление нового класса компонента должно включать макрос PACKAGE
class PACKAGE TMyComponent : public < базовый класс >
{ public:
__fastcall TMyComponent(TComponent *Owner);
};
```

9.3. Наследование компонента



Любой компонент является производным от базового класса TComponent и его наследников (TControl или TGraphicControl) или от существующего компонентного класса.

Объявление свойств, которые включаются в новый компонент нужно объявить в разделе **_published.**

9.4. Порядок регистрации компонент

	Выполняемые действия
1	В файле MyComp.cpp добавить функцию Register(). При этом функция заключается в пространство имен с именем, совпадающим с именем модуля компонента (первая буква должна быть заглавной, а все остальные – строчными).
2	В теле функции Register объявляется массив типа TComponentClass, в который вводится имя класса регистрируемого компонента.
3	В теле функции Register вызывается функция RegisterComponents с тремя параметрами: названием вкладки Палитры компонентов, массивом компонентных классов и индексом последнего класса в этом массиве.

Регистрация компонента – процесс, который информирует C++ Builder о том, какой компонент добавляется к VCL и на какой вкладке Палитры компонентов он должен появиться:

```
namespace Mycomp {
void __fastcall PACKAGE Register()
{ TComponentClass classes[1] = {__classid(TMyComponent)};
RegisterComponents("Samples", classes, 0);} }
```

9.5. Последовательность отладки компонент

	Выполняемые действия
1	Включить файл модуля MyComp.h компонента в заголовочный файл некоторой формы: #include "MyComp.h".
2	Добавить объектный член данных, представляющий испытываемый компонент, в раздел объявлений **public** класса формы: TMyComponent *MyComponent1.
3	Подсоединить обработчик к событию *OnCreate* формы.
4	Вызвать конструктор компонентного объекта из обработчика этого события, передав ему параметр, указывающий на владельца компонента. Обычно это указатель **this** на объект, который содержит компонент (например, форма).
5	Сразу же за вызовом конструктора установить свойство Parent – родителя компонента. Обычно значением этого свойства является указатель **this.** Если компонент не наследовался от TControl, то этот шаг пропускается.
6	Инициировать значения других свойств компонента.

9.6. Инсталляция компонента на Палитру

	Выполняемые действия
1	С помощью команды Component|Install Component открывается диалоговое окно инсталляции компонент. Можно выбрать вкладку Into New Package, если компонент включается в новый пакет, или вкладку Into existing package, если используется существующий пакет.
2	Вводится имя модуля компонента MyComp.cpp и путь к нему в поле Unit file name. Далее вводится имя нового пакета (совпадающее с названием вкладки Палитры компонентов, на которую устанавливается новый компонент) в поле Package file name, а его краткое описание – в поле Package description. Щелчок на кнопке ОК закрывает диалоговое окно установки. Добавление компонента в существующий пакет выполняется аналогично.
3	С текущим содержимым пакета можно ознакомиться в открывшемся окне Менеджера пакетов. Все файлы, составляющие рассматриваемый пакет, будут созданы (или перестроены), и новый компонент установлен на ту вкладку Палитры компонентов, которая была задана в качестве одного из параметров функции регистрации Register. C++ Builder автоматически включает в проект системные файлы периода выполнения (с расширениями .bpi, .res), необходимые C++ Builder для сборки пакета.
4	Выбирается команда Install в Менеджере пакетов. В результате выполняется компиляция, перестройка VCL и установка нового компонента на Палитру компонентов.
5	Командой Component|Install Packages или Project|Options открывается список установленных пакетов. По щелчку на кнопке Components можно увидеть список всех компонентов, включенных в выбранный пакет. Если нужно, чтобы текущая конфигурация пакетов принималась по умолчанию в каждом новом создаваемом проекте, то необходимо установить флажок Default.

9.7.Удаление компонент из VCL

	Выполняемые действия
1	Выполнить команду Component|Install Component, которая открывает диалоговое окно установки компонентов.
2	Найти удаляемый компонентный класс в списке Component classes выбранной группы Библиотеки и нажать кнопку Remove.
3	Нажать кнопку ОК. Библиотека будет перестроена и компонент удален из Палитры компонентов.

9.8. Перестройка Палитры компонентов

	Выполняемые действия
1	Открыть диалог установки опций Палитры компонентов с помощью команд Component|Configure Palette или Options|Environment|Palette.
2	Нажать кнопку Add и выбрать имя для новой вкладки. Имя добавленной вкладки появится внизу списка Pages названий вкладок.
3	Перетащить мышью выбранный компонент в списке Components на нужную вкладку списка Pages.
4	Нажать кнопку ОК. Библиотека и Палитра компонентов будут перестроены.

9.9. Сохранение файлов нового компонента

После окончания разработки, компонент будет представлен следующими файлами:

1) объектный файл результата компиляции MyComp.obj;
2) файлы модуля компонента (MyComp.h, MyComp.cpp);
3) файлы пакета с именем имя_пакета и расширениями bpl, bpk, lib, bpi,cpp, res;
4) файл пиктограммы компонента для Палитры компонентов My-Comp.dcr;
5) файл формы MyComp.dfm, если компонент использует форму;
6) контекстно-справочный файл My-Comp.hlp.

Перечень вопросов по теме 9

1. Перечислите основные правила разработки собственных компонентов.
2. Укажите и охарактеризуйте основные этапы создания собственных компонент
3. Можно ли создать компонент путем изменения уже существующего? Какой класс при этом может использоваться?
4. Раскройте суть процесса создания программного модуля компонента.
5. Приведите пример построения заголовочного файла модуля.
6. Каким образом происходит объявление нового класса компонента.
7. Что такое наследование компонент. Каким образом оно выполняется?
8. Какой класс является базовым для построения компонент.
9. В каком разделе и каким образом осуществляется объявление свойств новых компонент.
10. Дайте определение понятию Регистрация компонента. Приведите и опишите пример программного кода, используемого при регистрации.
11. Опишите порядок регистрации компонент.
12. Раскройте понятие Отладка компонент.
13. Когда и как выполняется инсталляция компонента на палитру компонентов?
14. Как указать значок кнопки, которую ваш компонент будет использовать в палитре компонентов?
15. Как инициировать событие, определенное пользователем?
16. Как удалить компонент из палитры VCL?
17. Возможно ли провести перестройку палитры компонент и как это делается
18. Перечислите и охарактеризуйте файлы, которые предстаяляют компонент по окончании разработки.

Тема 10

Создание и использование DLL в визуальных средах.
Установка приложений на ПК пользователя

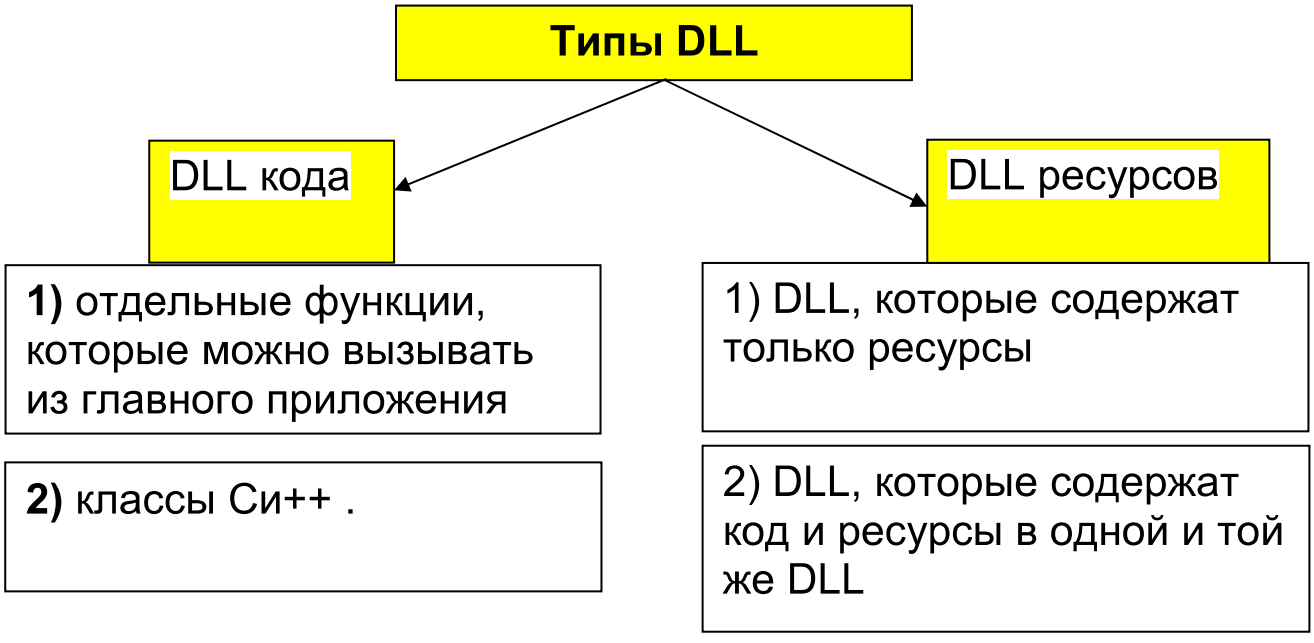
10.1. Создание и использование DLL

DLL (**Dynamic Link Library**) – это один или несколько участков кода, хранимых в файле с расширением .dll. Код, содержащийся в DLL, может быть вызван из исполняемой программы, но сама DLL не является автономной программой.

10.2. Преимущества DLL:

- Эффективное многократное использование кода
- Разделение кода между многими приложениями
- Секционирование кода
- Обеспечение многоязыковой поддержки
- Эффективное использование ресурсов Windows

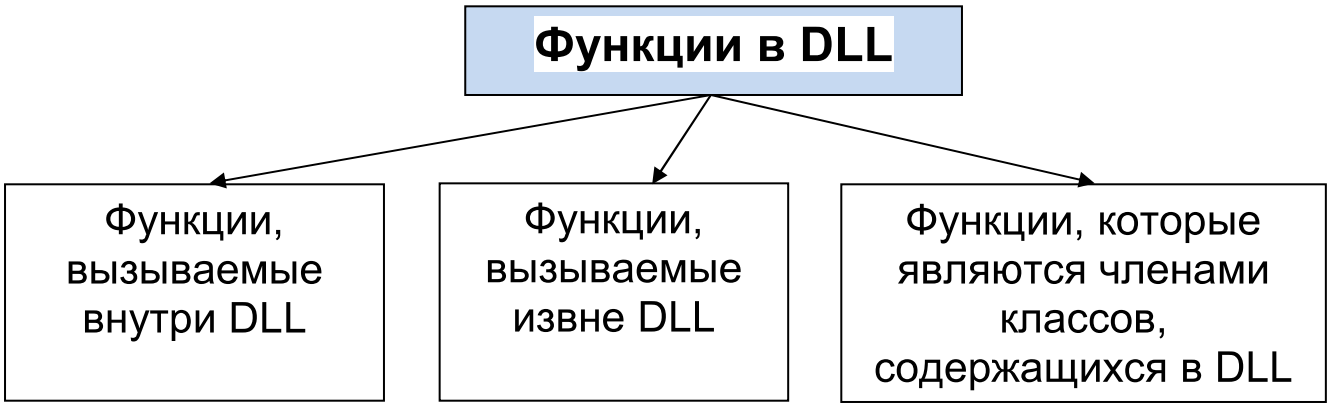
10.3. Типы DLL



Например:
Функции LoadCursor() и DrawText() - размещаются в User32.dll)

Стандартные диалоговые окна (Print, File Open, Printer Setup и т.д.) - размещаются в Comctl32.dll

10.4. Категории функций в DLL



Функции, вызываемые внутри DLL

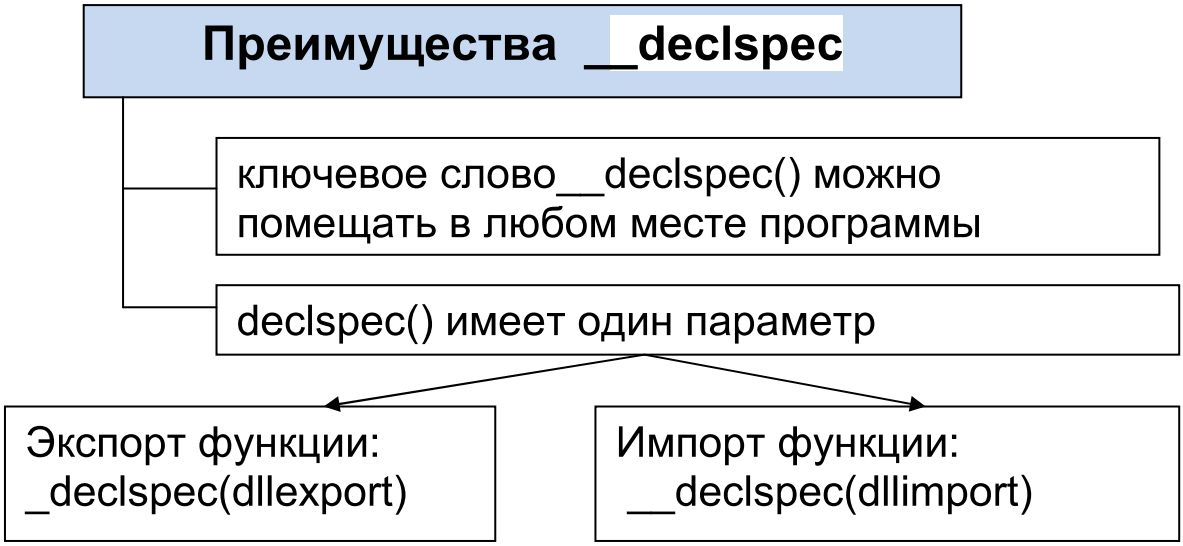
- Объявляются как и обычные функции
- могут быть вызваны другими функциями, но в пределах DLL

Функции, вызываемые извне DLL

- Объявляются как открытые для ***экспорта (__export)***
- могут вызываться как функциями самой DLL, так и из приложения
- функцию, вызываемую из DLL следует ***импортировать*** в приложение *(__**import**)*

Примечание: При использовании функций **__export**, **__import** нужно строго соблюдать правила их размещения

10.5. Ключевое слово __declspec().



Например:

```
__declspec(dllexport) void SaySomething(HWND hWnd);
void declspec(dllexport) SaySomething(HWND hWnd);
```

10.6. Порядок создания DLL

	Выполняемые действия
1	Определить назначение DLL.
2	Определить назначение функций DLL.
3	Задать экспортируемые функции и классы DLL.
4	Создать заголовок DLL (заголовочный файл *.h).
5	Создать библиотеку импорта (файл, содержащий список экспортируемых из DLL функций).

Для статически загружаемых DLL необходимы:

- Один или несколько исходных файлов.
- Один или несколько заголовочных файлов (содержат объявления или классы, которые могут быть вызваны из DLL).
- Файл библиотеки импорта (для компоновщика - связь вызовов в приложении с адресами в DLL).
- Сама DLL.

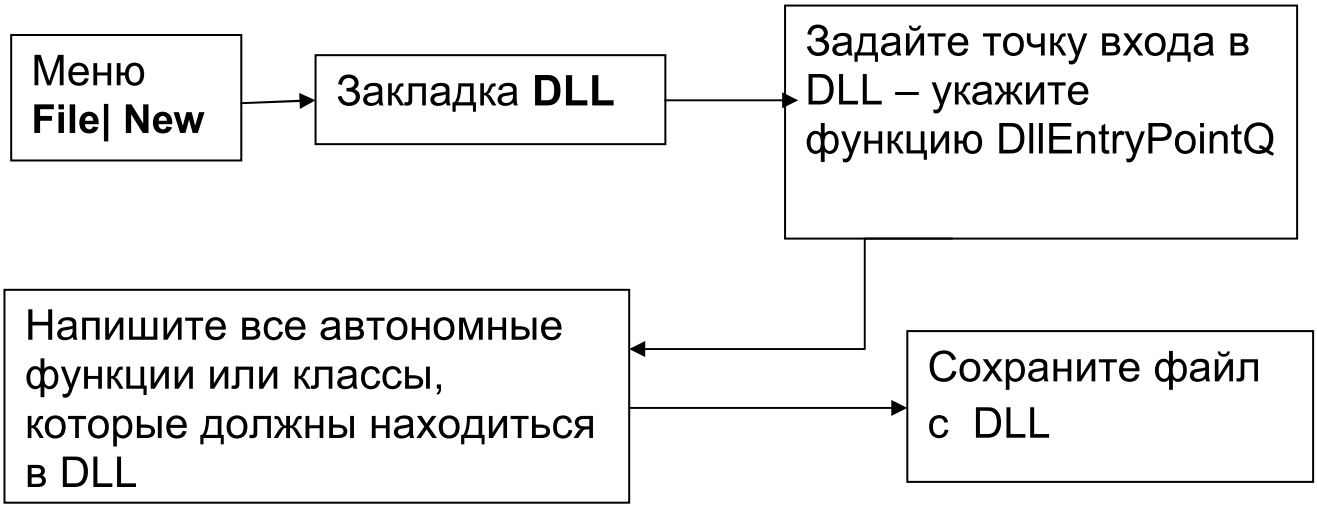
10.7. В комплект поставки DLL должны входить:

1.Заголовок (.h)

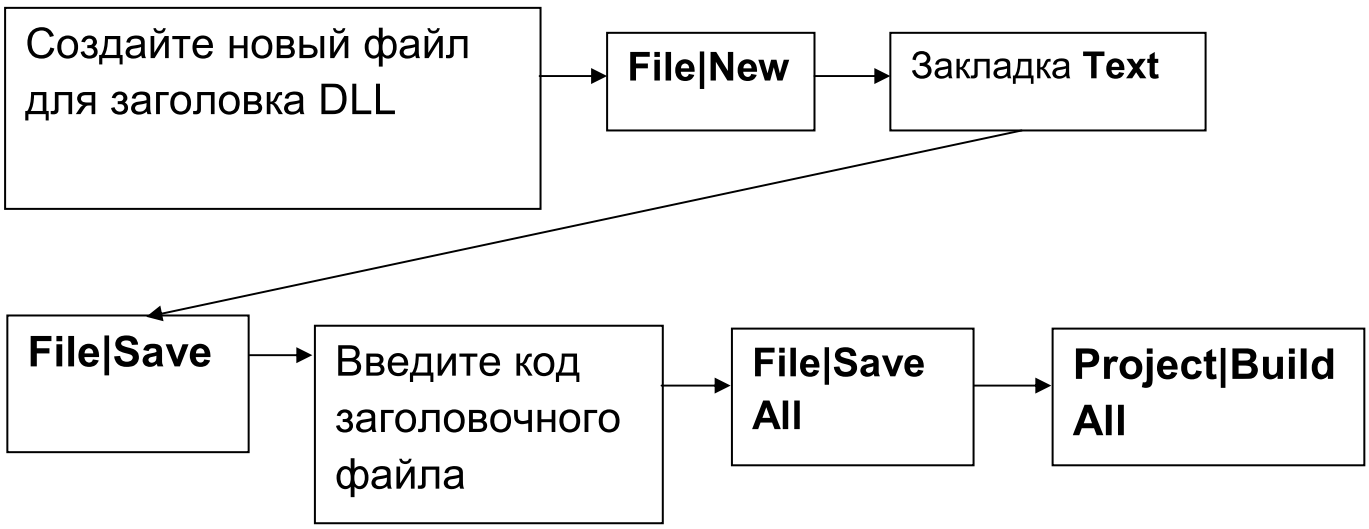
2. Библиотека импорта (.lib)

3. Собственно динамически компонуемая библиотека (.dll).

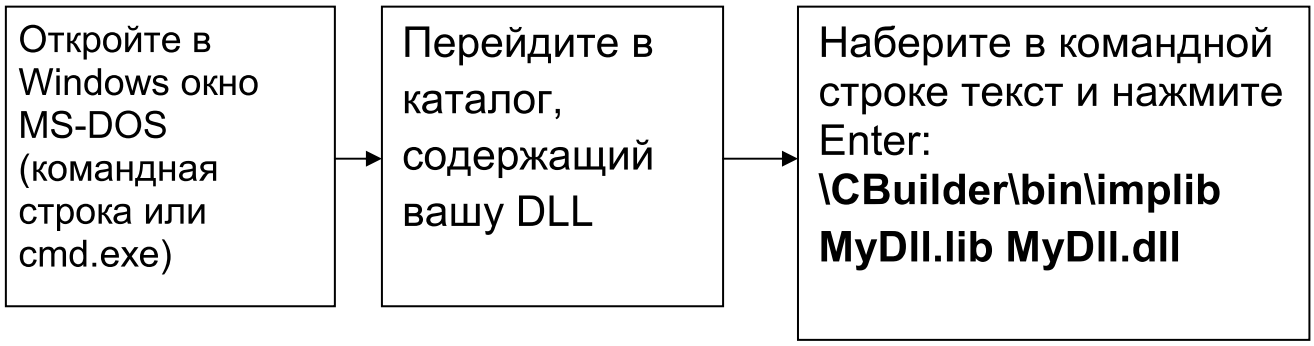
10.8. Порядок создания DLL с помощью репозитория объектов



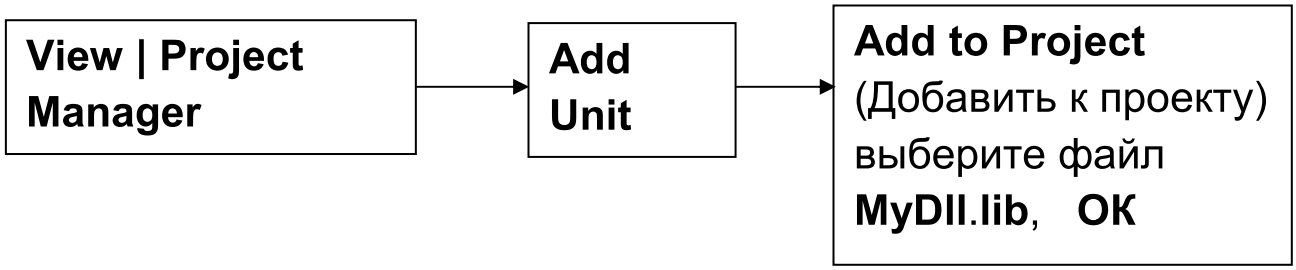
10.9. Добавление к проекту заголовочного файла для DLL



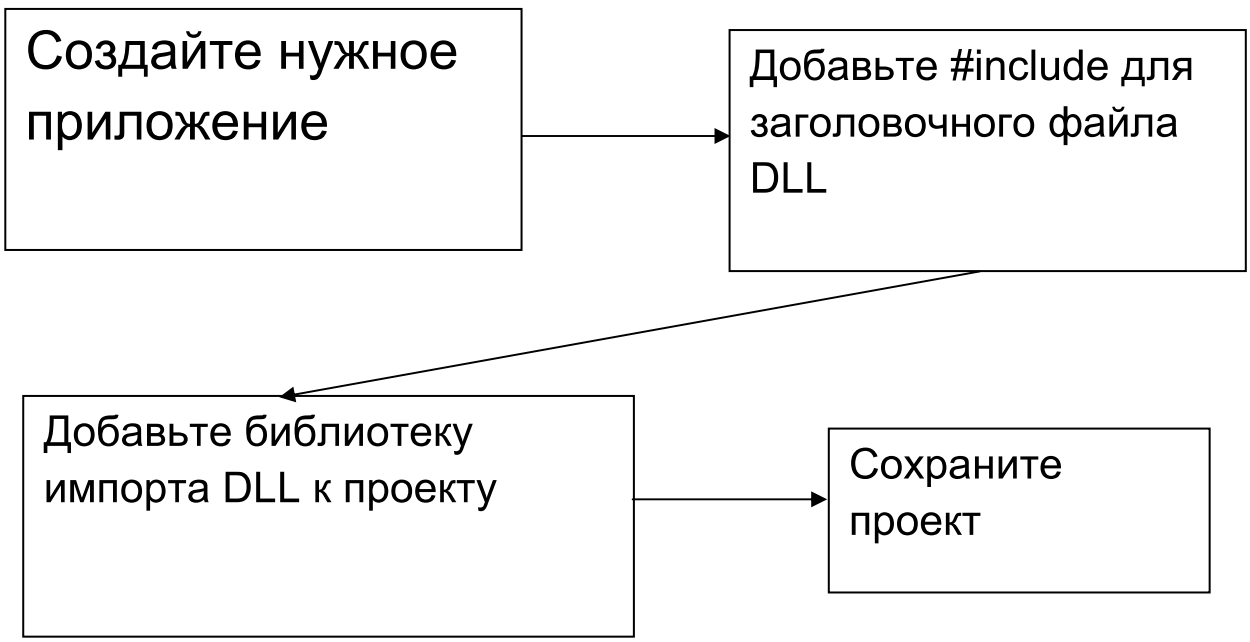
10.10. Создание файла библиотеки импорта



10.11. Добавление файла библиотеки импорта к проекту



10.12. Создание вызывающего приложения



10.13. Статическая и Динамическая загрузка DLL

Статическая загрузка (static loading)
DLL автоматически загружается при запуске
DLL содержит экспортируемые функции
Описание функций находится в специальном файле, называемом ***файлом библиотеки импорта (import library file)***
Файл библиотеки импорта имеет имя, совпадающее с именем DLL, и расширение **.lib**

Преимущества статической загрузки:
- DLL загружается при загрузке приложения;
- это самый легкий путь подключения к проекту и использования кода, содержащегося в DLL.

Недостатки:
при отсутствии необходимой DLL программа откажется загружаться

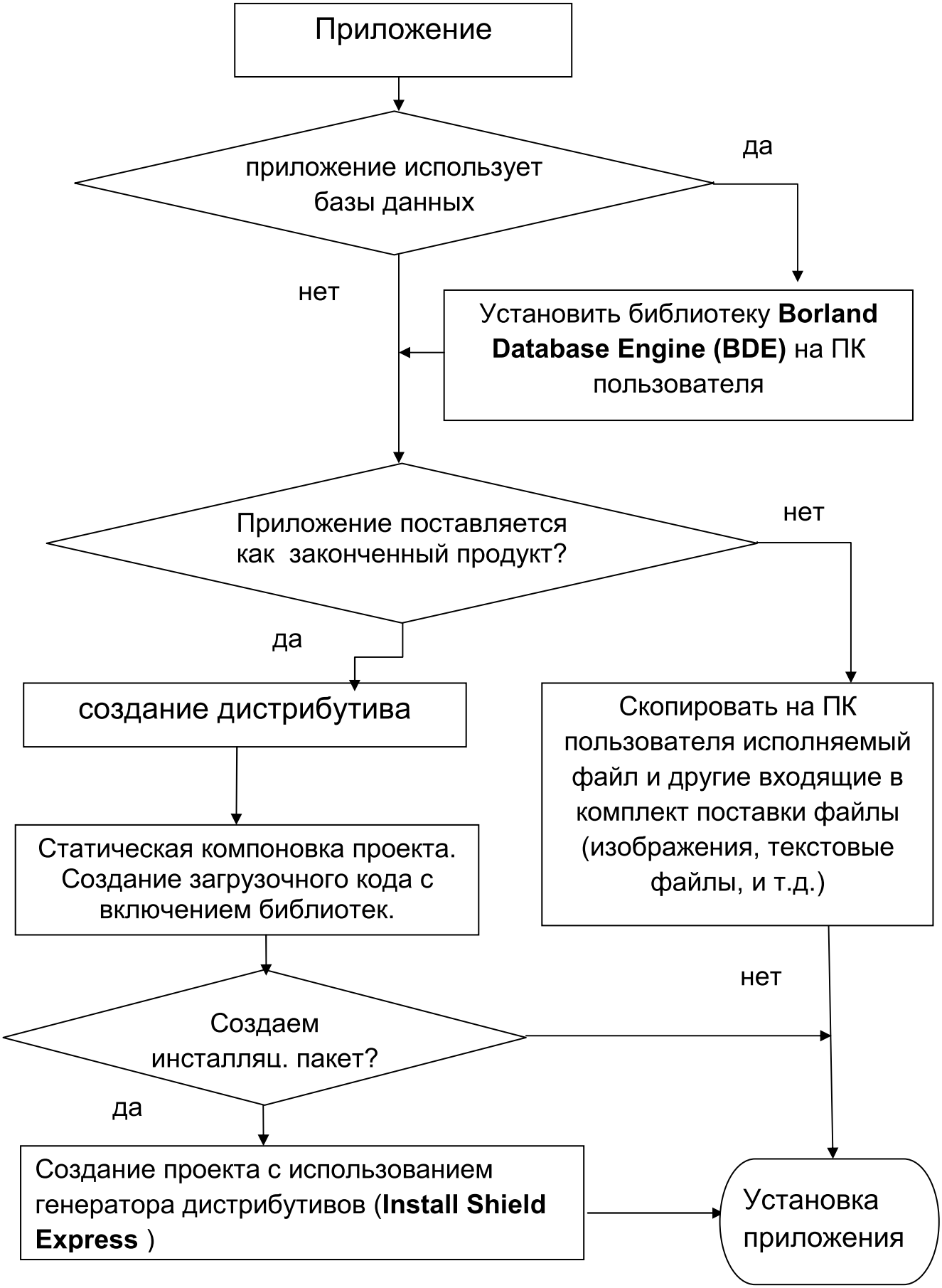
Динамическая загрузка (dynamic loading)
DLL загружается по необходимости, во время выполнения приложения
Необходимо получить указатель на функцию, которую вы хотите вызвать
DLL выгружается по окончании использования

Преимущества динамической загрузки:
- эффективное использование памяти;
- быстрая загрузка приложения.

Недостатки:
Объем работы:
1. Загрузить DLL (используя ф. **LoadLibrary()** из API Windows.
2. В конце выгрузить DLL (используя ф. **FreeLibrary()**)
3. Использовать функцию **GetProcAddress()** для получения указателя на ту функцию, которую вы хотите вызвать.

Если вы создаете одно приложения небольшого или среднего объема, лучше использовать статическую компоновку. Для больших приложений или приложений с большим числом DLL следует выполнить динамическую компоновку.

10.14. Установка приложений на ПК пользователя



10.15. Статическая компоновка проекта

1. Открыть в среде **C++ Builder** готовый проект (.bpr)

2. В меню **Project** выбрать команду **Options,** вкладку **Linker** и снять установленный среде **C++ Builder** по умолчанию флаг **Use dynamic RTL** (использовать библиотеки времени выполнения).

3. На вкладке **Project** -> **Options -> Packages** снять флаг **Build with runtime packages** (строить с пакетами времени выполнения) и щелкнуть **OK**.

4. Создать исполняемое приложение при помощи команды **Project -> Build <имя проекта>**.

5 Скопировать приложение на ПК без **C++ Builder вместе с** файлами, которые будут использоваться в его работе (рисунки, справки, базы данных)

При статической компоновке проекта код, который требуется для запуска приложения, расположен непосредственно в исполняемом файле. Приложение является автономной программой и не требует никаких пакетов поддержки или DLL и будет успешно работать на компьютере, на котором не установлена среда **C++ Builder** (преимущество).

Недостатки статической компоновки:

1. Размер кода приложения увеличивается в несколько раз по сравнению с обычной (локальной) компоновкой пакета, т.к. код библиотечных модулей дублируется в каждом модуле (собственно главном приложении и в каждой DLL). Этот дублированный код является по-сути избыточным.
2. Приложение, использующее BDE-доступ к базам данных, потребует для своей работы установки библиотеки BDE, даже если оно скомпоновано статически.
3. Предполагается, что приложение не использует элементы управления ActiveX.

10.16. Возможности генератора дистрибутивов (Install Shield Express)

Функции Инсталляционного пакета

- копирует файлы (***.exe**, ***.ocx**, ***.dll**, ***.hlp**, шрифты, данные и др.) в соответствующие каталоги;
- вносит необходимые модификации в реестр;
- модифицирует переменную окружения **PATH**;
- создает программную группу и пиктограммы;
- модифицирует меню **Windows**;
- предоставляет пользователю возможность выбора параметров установки (каталог, выбор устанавливаемых частей приложения);
- отображает меню лицензионного соглашения и файл **readme**, с последней информацией о продукте;
- при необходимости перезапускает **Windows**.

Недостатки Install Shield Express

- продукт предполагает, что устанавливаемая (имеющаяся на компьютере, где создается дистрибутив) версия **BDE** – самая новая, что может привести к замене уже имеющейся на компьютере пользователя версии **BDE** на более раннюю;
- некоторые диалоги инсталляционного приложения являются англоязычными.

Перечень вопросов по теме 10

1. Дайте определение понятию **Dynamic Link Library.** Преимущества и недостатки DLL..

2. Категории функций в DLL. Основное назначение и особенности каждой категории.

3. Порядок создания **DLL.** Комплект поставки **DLL.**

4. Ключевые слова **__declspec(), __*export*,** **__import.** Описание особенностей применения.

5. Раскройте последовательность создания DLL с помощью репозитария объектов.

6. Что такое статическая и динамическая загрузка DLL? Как загрузить DLL статически? Как загрузить DLL динамически? Как вызвать функцию из DLL, которая загружена статически?

7. Способы использования DLL. Какие действия необходимо предпринять, чтобы гарантировать вызов функции, содержащейся в DLL, извне?

8. Охарактеризуйте способы поставки приложений пользователю. Укажите основные достоинства и недостатки каждого из способов. Особенности запуска программ на компьютер пользователя.

9. Что такое статическая компоновка проекта и как она выполняется? Как изменяется код приложения при статической компоновке проекта и какими особенностями обладает.

10. Создание инсталляционного пакета. Особенности и этапы создания. Характеристики программного обеспечения, которое следует установить на компьютер программиста для создания инсталляционного пакета.

11. Основные функции и возможности инсталляционного пакета. Обеспечение работоспособности инсталляционного пакета

12. Особенности и нюансы настройки и установки на компьютере пользователя приложения, использующего базы данных.

13. Какие недостатки имеют инсталляционные пакеты, созданные с помощью программы **InstallShield Express**? На каком компьютере следует тестировать инсталляционный пакет?

Перечень рекомендованной литературы

1. Шамис В. М.: Нолидж, Borland C++ Builder 3. Техника визуального программирования. М: ЗАО "Издательство БИНОМ ",1998. - 512 с.
2. Камаев В.А.,Костерин В.В. Технологии программирования. Учебник - М: Высш.шк., 2006.- 454с.
3. Аленский Н.А. Визуальное объектно-ориентированное программирование в задачах. : Учеб.пособие,— Мн.: БГУ, 2008, — 114с.
4. Архангельский А. Я.. Программирование в C++Builder 6 — М: ЗАО "Издательство БИНОМ", 2010. — 1304с.
5. Архангельский А. Я. Справочное пособие по C++Builder 6. Книга 1. Язык C++ — М: ЗАО "Издательство БИНОМ ", 2002.
6. Архангельский А. Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты. — М: ЗАО "Издательство БИНОМ ", 2002. — 528с.
7. Культин Н. C++Builder. – СПб.: БХВ-Петербург, 2008. – 464с.
8. Культин Н. C++ Builder в задачах и примерах. – СПб.: БХВ-Петербург, 2007. – 336с
9. *Кент Рейсдорф,Кен Хендерсон, Borland C++ Builder. Освой самостоятельно.2007-702с.*
10. *Вдовенко В. В. Программирование на языке C++: учебное пособие - Красноярск: СГАУ, 2006.-215с.*
11.Вальпа О.Д. Borland C++ Builder. Экспресс-курс. – СПб.: БХВ-Петербург, 2006. – 224с.
12. Визуальное программирование [Электронный ресурс] /Википедия - свободная энциклопедия, - http://ru.wikipedia.org/wiki/Визуальное_программирование
13. Курс: Технология создания программных и интеллектуальных систем /Центр дистанционного обучения ДонНТУ, - http://temp1.donntu.edu.ua/course/view.php?id=48
14. http://ieg.ucoz.ru/publ/grafika_v_c_builder_6/
15. http://cubook.supernew.org/object/224-chart-grafiki-i-diagrammy