МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕР ИНЖИНИРИНГ

КАФЕДРА «ОСНОВЫ ИНФОРМАТИКИ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ПРЕДМЕТУ «ПРОГРАММИРОВАНИЕ на C/C++»

Ташкент-2015

ВВЕДЕНИЕ

Курсовая работа является одной из форм учебной деятельности, которая выполняется студентом самостоятельно под руководством преподавателя. Курсовая работа представляет собой учебно- исследовательскую деятельность, требующую от студентов освоения элементов научного исследования. Выполнение курсовой работы направлено на формирование у студентов способности самостоятельно мыслить, анализировать и сопоставлять факты, обобщать и логически излагать материал. В результате выполнения курсовой работы у студентов формируется субъективно новое знание по одной из частных проблем.

В ходе работы у студента развивается научная наблюдательность, студент учится не только находить необходимую информацию, но и корректно ее использовать в своем исследовании, грамотно демонстрировать, как и откуда были получены те или иные сведения, и каково их значение для данного исследования.

Курсовая работа способствует формированию у студентов опыта самостоятельного научного творчества, повышению уровня теоретической и профессиональной подготовки, лучшему усвоению учебного материала.

При написании работы студент должен показывать практические навыки работы с персональным компьютером, анализировать литературные данные, делать обоснованные выводы и предложения.

Курсовые работы способствуют закреплению, углублению, обобщению и прикладному применению знаний и умений, формируемых студентами при изучении курса «Программирование на C/C++».

Во время подготовки курсовой работы перед студентом не стоит задача открыть новые научные положения в области программирования. В процессе изложения темы студенту необходимо показать способность научно использовать литературу, понимать методологию изложения материала, уметь систематизировать данные, обрабатывать фактический материал, делать

обобщения и выводы, увязывать теорию с практикой и современной действительностью.

1. РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ КУРСОВОЙ РАБОТЫ

Темы курсовых работ предлагаются преподавателем. По согласованию с руководителем студент может уточнить формулировку предлагаемой темы или предложить собственную тему, обосновав целесообразность исследования.

После утверждения темы курсовой работы и изучения литературы, рекомендованной преподавателем, определяется направление исследования, его цель и задачи. Затем студент самостоятельно подбирает дополнительные источники информации (книги, периодические издания, электронные ресурсы), которые планируется использовать при выполнении исследования, разрабатывает структуру содержания курсовой работы. Составленный список литературы и план курсовой необходимо согласовать с преподавателем.

Выполнение курсовой работы включает в себя изучение теоретического материала, рассмотрение и оценку возможных решений, подбор методов исследования, сбор, анализ и обобщение собственного материала, разработку программного продукта (обеспечения), написание текста, тестирование и отладку программы, анализ результатов, формулировку комментариев и выводов.

2. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ КУРСОВОЙ РАБОТЫ

Результат учебно-исследовательской деятельности во многом зависит от понимания студентом основных характеристик научного исследования и их формулировок. К основным характеристикам исследования относятся: актуальность, проблема, объект, предмет, основная цель, частные задачи и методы исследования.

Большинство тем курсовых работ являются своевременными и актуальными. Если тематика курсовой работы актуальна, то изложение следует начинать с описания *актуальности,* которая определяется необходимостью проведения исследования в современных условиях. В содержании курсовой работы обязательно указывается *проблема исследования*, характеризующая то, что надо изучить из того, что ранее не было изучено.

С проблемой исследования тесно связаны объект и предмет исследования. Их формулировки также обязательно приводятся в содержании курсовой работы.

Под *объектом исследования* понимают часть объективной реальности, которая изучается в процессе теоретической и практической деятельности. *Предметом исследования* считают свойства, отношения объекта, исследуемые в процессе практической деятельности с определенной целью в данных условиях и обстоятельствах. Поэтому объект и предмет исследования как категории научного познания соотносятся между собой как общее и частное. В объекте выделяется та его часть, которая служит предметом исследования. Необходимо четко представлять границы исследования и предполагаемые результаты.

Цель исследования состоит в том, чтобы разрешить поставленную проблему, достичь определенный результат. При формулировке цели исследования обычно используются следующие термины: анализ, выявление, внедрение, изучение, развитие, разработка и т.д.

В зависимости от цели курсовой работы необходимо сформулировать две-три конкретные *задачи исследования*, которые необходимо решить для достижения цели. Это обычно делается в форме перечисления: *изучить ..., описать ..., установить ..., выявить ..., вывести ..., разработать ...* и т.п. Формулировку задач необходимо выполнить тщательно, так как описание хода и результатов их решения составит основное содержание курсовой работы.

3. СТРУКТУРА И СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ

Титульный лист

Аннотация

Задание

Оглавление

Введение

1. Теоретическая часть.
   - 1.1. Анализ и исследование задачи, построение модели.
   - 1.2. Алгоритмы, обеспечивающие функциональность приложения.
2. Основная часть
   - 2.1. Цели и задачи курсовой работы.
   - 2.2. Руководство пользователю.
   - 2.3. Руководство программисту.
3. Тестирование программы
   - 3.1. Оценка результатов проведения тестирования
4. Заключение.
5. Список литературы.
6. Приложения.

4. СОДЕРЖАТЕЛЬНОЕ НАПОЛНЕНИЕ РАЗДЕЛОВ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Аннотация

Содержит очень краткое содержание работы, число страниц пояснительной записки, число рисунков, таблиц, приложений.

Задание

Основным документом, в соответствии с которым выполняется разработка некоторого проекта в любой отрасли, включая проекты по разработке программного обеспечения (ПО), является **задание**.

При разработке ПО задание – документ (спецификация), оговаривающий перечень требований к системе и утверждённый как заказчиком/пользователем, так и исполнителем/производителем системы.

Пример задания приведен в приложении.

Введение

Введение должно содержать общие сведения по теме курсовой работы. Так, если в основе работы лежат информационные системы, то требуется дать информацию о информационных системах (что это, зачем, особенности и т.д.). Если речь идет о компиляторах, то сведения о компиляторах и т.д.

Во введении также необходимо отразить:

- актуальность выбранной темы (например, сказать, что в современном обществе или в деятельности любого современного предприятия информация является одним из важнейших ресурсов, выделяясь в самостоятельный фактор для принятия решений, и от средств ее обработки во многом зависит эффективность принятия решений);
- используемые модели программирования (например, императивное программирование, модульное программирование, структурное программирование, объектно-ориентированное программирование, основанное на классах,...)

- практическую значимость полученных результатов (где можно использовать);
- какого рода ресурсы необходимы для реализации (ПК, программное обеспечение…уточнить какое);
- перспективы совершенствования изготавливаемого программного продукта.

Цели и задачи курсовой работы

Составляющие элементы этого раздела могут быть следующие:

- формулировка условия задачи;
- сбор информации о задаче и выделение физического объекта;
- определение конечных целей решения задачи;
- определение задач, т.е форм выдачи результатов;
- описание данных (их типов, диапазонов величин, структуры и т.п.).

Формулировка условия задачи. В качестве примера для объяснения хода и логической основы некоторых элементов выполнения работы предлагается к рассмотрению следующая постановка:

О товаре, размещенном на складе, имеется информация вида: номер артикула, наименование товара, общее количество, выделенное количество, не поставленное количество, цена единицы. Разработать приложение (информационную систему), которое выдает информацию о товаре по требованию пользователя и подводит итоги по непоставке товаров. Необходимо предусмотреть ввод, корректировку и удаление записей базы данных.

Сбор информации о задаче. Следует определиться, с какими данными разработчик проекта имеет дело. Так как в задаче речь идет о товаре, размещенном на складе, нужно указать, какой товар (перечислить его: кирпич, цемент, гвозди…), что связано с поставкой/непоставкой (документы на заказ и на доставку). Выделить физический объект (множество товаров) и определить его свойства. В том числе, количество элементов—единиц товара. А т.к. в

данном случае физический объект определен как множество, то следует выделить свойства отдельного представителя из этого множества. При этом необходимо выявить самые существенные свойства, необходимые для решения задачи. Выделив наиболее важные факторы, можно пренебречь менее существенными. Параметрами отдельного представителя (единицы товара), например, могут быть: наименование, цена, фирма-изготовитель, поставщик, количество, ...

Определение конечных целей решения задачи.В соответствии с условием данной задачи разрабатываемая программа должна предоставлять пользователю хранить информацию о товарах и формировать из нее нужную информацию по его требованию. Такими требованиями могут быть: получить список товаров заданного наименования с его характеристиками (какими?), список поставщиков, ...

Определение формы выдачи результатов. Например, представить список поставщиков в виде таблицы, в которой можно было бы увидеть наименование поставляемого им товара ... .

Описание данных (их типов, диапазонов величин, структуры и т.п.). Например, наименование товара представляется строкой символов, количество символов не превышает 20. Цена указывается в сумах, может быть представлена вещественным числом, диапазон ценовых колебаний в промежутке от 10000 до 500. ....

Анализ и исследование задачи, построение модели. Составляющими данного параграфа могут быть следующие элементы:

- выделение математического объекта;
- существующих аналогов;
- анализ технических и программных средств;
- разработка математической модели;
- разработка требований к приложению.

Выделение математического объекта. Сказать, например, что для того, чтобы иметь возможность хранить и обрабатывать данные о физическом

объекте, эти данные нужно представить в виде, приспособленном для обработки математическими методами. Для этого нужно перейти от физического объекта к объекту математическому. В нашем случае нужно перейти от физического объекта — множество товаров, где каждый товар имеет свои физические характеристики, к математическому объекту, описывающему это множество. Причем, каждый элемент математического множества должен быть представлен эквивалентными свойствами элемента физического множества (имеет структуру из свойств).

Далее можно сказать о том, что для описания математического множества можно воспользоваться таким математическим объектом, как «массив». В данном случае это будет массив структур.

Анализ существующих аналогов. Сказать, что хранить и обрабатывать массив структур описанных данных можно разными способами, например, средствами базы данных. Привести примеры известных баз, например, Access (особенности…), в Excel (особенности…).

Анализ технических и программных средств. ПК – техническое средство. Объяснить, почему выбрана среда C++ Builder, а не Excel или др. (.. возможности ООП и визуального….).

Разработка математической модели. Под математической моделью понимают систему математических соотношений — формул, уравнений, неравенств и т.д., отражающих существенные свойства объекта. Метод математического моделирования сводит исследование поведения объекта или его свойств к математическим задачам. Следует выписать формулы, например с использованием знаков суммирования…

При этом можно пользоваться и такой схемой действий:

1. выделить предположения, на которых будет основываться математическая модель. Например, предположить, что информация о товаре будет сосредоточена в структурах данных (или содержатся в файле, или в таблице, или…);
2. определить, что считать исходными данными и результатами;

3. записать математические соотношения, связывающие результаты с исходными данными. Какие методы обработки (с описанием подхода к решению, например, «.... решение сводится к задаче накопления суммы элементов...формула....») данных потребуются для решения задачи?

Выведенные зависимости и формулы в общем случае могут быть представлены уравнениями, системами уравнений – линейных, интегральных, матричных, дифференциальных и др.. На этом этапе для нахождения решения также строится неформальный алгоритм (производная от длины и решение уравнения: производная равна 0, ...).

При построении математических моделей далеко не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких случаях используются математические методы, позволяющие дать ответы той или иной степени точности.

Разработка требований к приложению. Например, сказать о том, что пользователь должен иметь возможность для задания исходных данных вручную, возможность для сохранения данных в файле, возможность формировать файл данных с целью повторного использования; извлекать данные из файла данных; выбирать вид отображаемой информации (например, общее количество товара, суммарная стоимость всего товара, номенклатура товара и т.д. в соответствии с требованиями); отображать нужную информацию в соответствии с выбранным видом..., а также обеспечение соединений этих данных в соответствии с пользовательской логикой (все исходя из поставленной задачи).

В первой главе– обычно теоретической – дается анализ научной и методической литературы. Здесь предлагается провести подробное исследование теоретической части курсовой работы.

Необходимо последовательно и логично рассмотреть сущность и основное содержание проблемы, изучаемых вопросов и понятий; изложить мнения различных авторов и свои умозаключения. Не следует забывать о

необходимости делать ссылки на литературные источники, материал которых использовался при написании работы.

Первая глава демонстрирует общий научно-методический уровень подготовки студента, его умение подбирать и изучать литературу, систематизировать знания, делать обобщения и выявлять возможные направления решения проблемы. Содержание параграфа должно быть посвящено отдельному аспекту исследования.

Во второй главе– обычно практической – следует описать и обосновать конкретный подход к решению поставленной проблемы. В качестве проблемы в задании на курсовую студенту предлагается выполнить практическое задание – составить программу.

В разделе **«Руководство пользователю»** второй главы приводится полное описание работы программы с использованием соответствующих скриншотов.

Описание раздела **«Руководство программисту»** второй главы включает в себя описание модулей и информационных объектов.

Тестирование программы

Тестирование созданного программного продукта следует начинать с разработки плана тестирования. При этом следует помнить, что весь процесс тестирования можно разделить на три этапа:

- Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы.
- Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий — это граничные объемы данных, когда массивы состоят из слишком малого или слишком большого числа элементов.

- Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области изменений.

Известно, что все программы разрабатываются в расчете на обработку какого-то ограниченного набора данных. Поэтому важно получить ответ на следующие вопросы:

- Что произойдет, если в программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки придется иметь дело как раз с такими данными?
- Как будет вести себя программа, работающая с массивами, если количество их элементов превысит величину, указанную в объявлении массива?
- Что произойдет, если числа будут слишком малыми или слишком большими?

Наихудшая ситуация складывается тогда, когда *программа воспринимает неверные данные как правильные и выдает неверный, но правдоподобный результат*. Программа должна сама отвергать любые данные, которые она не в состоянии обрабатывать правильно.

Следует попытаться представить возможные типы ошибок, которые пользователь способен допустить при работе с Вашей программой и которые могут иметь неприятные для нее последствия. Нужно не забывать, что способ мышления пользователя отличается от способа мышления программиста. Если помнить об этом, то нужно предусмотреть обработку ошибок типа: отсутствие нужного файла, неправильные форматы данных и т.д. Список действий, которые могут привести к неправильному функционированию программы, довольно длинный и зависит от того, что делает приложение в данный момент времени.

Основной смысл этого этапа состоит в проверке того, насколько программный продукт в том виде, в котором он получился, соответствует требованиям, установленным в процессе согласования спецификации. Каждая

функция или метод класса должны соответствовать требованиям, определенным для них на этапе спецификации.

Разработка плана тестирования - состоит из следующих шагов:

1) определение последовательности действий, которые позволяют проверить как работу отдельных методов, так и их совокупности;

2) подготовка входных данных, для которых известны результаты тестирования;

3) определение места расположения тестовых данных.

Разработка алгоритма процедуры тестирования состоит в разработке процедуры в соответствии с составленным выше планом тестирования. Эту процедуру можно представить блок-схемой с соответствующими комментариями.

Оценка результатов тестирования – содержит сравнение выходных данных работы отдельных процедур с контрольными значениями, которые получены в результате выполнения процедуры тестирования.

В заключении подводятся итоги проделанной работы, на основе теоретических выводов и данных практической главы делаются общие выводы по теме исследования. Необходимо показать, как решены задачи, привести основные результаты работы, сделать свои умозаключения о целесообразности и эффективности использования результатов исследования на практике. Выводы должны соответствовать содержанию работы, быть краткими, ясно, четко и логично сформулированными. В заключении также намечаются дальнейшие перспективы и пути исследования, возможность использования результатов проведенной учебно- исследовательской работы.

Библиография содержит перечень названий книг, статей, документов и электронных ресурсов, которые были использованы при подготовке курсовой работы и включает в себя всю литературу, на которую имеются ссылки и сноски в тексте.

В приложения помещают вспомогательные или дополнительные материалы, изложение которых необходимо для полноценного описания,

проведенного исследования, но которые могут затруднить восприятие основного текста курсовой работы, сделать его трудночитаемым. В приложения следует вынести нормативные акты, требования к программным средствам, коды разработанных компьютерных программ, рисунки, фотографии, демонстрационные материалы и т.п.

5. ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ТЕМ КУРСОВОЙ РАБОТЫ

Данный перечень тем носит рекомендательный характер. Студент может предложить собственную тему для исследования, но обязательно согласовать ее с преподавателем.

1. Program integrated to application related to business (Интегрированное приложение ведения бизнеса).
2. Documentation and status inquiry system, which uses character and image recognition (Программа демонстрирующая распознавание символов и изображений).
3. 3D fighting game of planes, with multilateral representation of the terrain (3D боевая игра самолетов).
4. Real-time video data transmission system (Система передачи видео данных).
5. Production and shipping system of intelligent e-mail marketing (Система интеллектуального маркетинга e-mail).
6. Solutions for Educational multimedia retrieval (Создание образовательной мультимедийной обработки).
7. Information Management System (Система управления информацией).
8. Creating 3D avatar face recognition (Создание 3D фэйс распознавателя).
9. Solution for Multimedia e-book development (Создание электронных мультимедийныхе-book).

10. Online 3D baseball game (Онлайн 3D игра в бейсбол).

11. Real-time MP3 music broadcasting and multi-party voice chat (MP3 воспроизведение и вещание музыки и голосового чата).

12. Standardized project management system (Система управления проектами).

13. Online 3D robot simulation game (3Dонлайн игра-робот).

14. Automatic timetable generator of university (Автоматизация разработки расписания университета).

15. Unattended parking management system (Автоматическая система управления парковкой).

16. Teacher Support System (Информационная система «учитель»).

17. Integrated enterprise resource planning system (Интегрированная система планирования ресурсов предприятия).

18. One-to-one marketing automation system (Система автоматизации маркетинга из рук в руки).

19. Interactive cyber education system (Интерактивная система кибер образования).

20. School office management system (Система управления школьного офиса).

21. Разработка консольного приложения на языке C++, реализующее работу с универсальной очередью ограниченного размера используя шаблон класса.

22. Автоматизированная информационная система: деканат.

23. Автоматизированная информационная система ари: университет

24. Автоматизированная информационная система: кафедра

25. Автоматизированная информационная система: отдел кадров

26. Автоматизированная информационная система: почта

27. Автоматизированная информационная система: продажа авиа-билетов

28. Автоматизированная информационная система: аптека

29. Автоматизированная информационная система: приёмный покой больницы

30. Автоматизированная информационная система: медицинский диагноз

31. Автоматизированная информационная система: основы духовности

32. Автоматизированная информационная система: развлекательные места г.Ташкента

33. Автоматизированная информационная система: кинотеатр

34. Автоматизированная информационная система: метро станции г.Ташкента

35. Автоматизированная информационная система: общежитие студентов

36. Автоматизированная информационная система: автосалон

37. Автоматизированная информационная система: национальные блюда

38. Автоматизированная информационная система: достопримечательности Узбекистана

39. Автоматизированная информационная система: библиотека

40. Автоматизированная информационная система: супермаркеты г.Ташкента.

41. Автоматизированная информационная система: школы г.Ташкента.

42. Автоматизированная информационная система: детские сады г.Ташкента.

43. Автоматизированная информационная система: музеи в г.Ташкента.

44. Автоматизированная информационная система: заводы в г.Ташкента.

45. Автоматизированная информационная система:фабрики г.Ташкента.

46. Автоматизированная информационная система: страховые компании Узбекистана.

47. Автоматизированная информационная система: банки Узбекистана.

48. Автоматизированная информационная система: разные обучающие курсы г.Ташкента.

49. Автоматизированная информационная система: ресторан.

50. Автоматизированная информационная система: вузы Узбекистана.

51. WEB сайт о выдающихся ученых Узбекистана

52. Построение мини компиляторов. Основные конструкции языка.

53. Построение мини компиляторов. Оператор присваивания.Выражения.

54. Построение мини компиляторов. Условный оператор.

55. Построение мини компиляторов. Повторения.

56. Построение мини компиляторов . Строки.

57. Создание автоматизированной обучающей системы.

58. Программа. Много функциональные электронные часы..

59. Программа.. Создание протоколов соревнований.

60. Программа. Создание таблицы чемпионата.

61. Программа. Определение сбойных дорожек диска. .

62. Использование списков.

63. Использование "Панели инструментов".

64. Возможности изображения программной среды.

65. Возможности работы с файлами программной среды.

66. Возможности программной среды на примере использования компонент.

67. Интернет обеспечение программной среды.

68. Интернет обучающие ресурсы курса "Основы программирования" .

Тема теоретической части работы должна соответствовать методам и инструментам, используемым при разработке программы в практической части.

6. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСОВОЙ РАБОТЫ

Цель курсовой работы по дисциплине «Программирование на С/ С++» состоит в закреплении и углублении знаний и навыков, полученных при изучении дисциплины. Курсовая работа предполагает выполнение задания повышенной сложности по проектированию, разработке и тестированию программного обеспечения, а также оформлению сопутствующей документации.

В качестве среды разработки следует использовать MS Visual C++ 2010/BorlandC++ Builder, что позволит получить первичные навыки работы в современной и широко используемой на практике среде разработки приложений. Следует создать Win32 приложение с использованием средств быстрой разработки (RAD-средств) пользовательского интерфейса, обработчиков и т.п., что, в свою очередь, способствует более глубокому пониманию основ создания современного ПО с графическим интерфейсом.

7. ПРИМЕРНОЕ ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Необходимо разработать небольшое Windows-приложение в среде визуального программирования MS Visual C++ с использованием элементов WindowsForms - программу ввода заказа в вашем любимом кафе. Для примера вам предлагается взять кафе «Паркуша» или «Пиноккио», информация по меню этих кафе может быть взята с их официальных сайтов. Но вы можете выбрать для задания любое ваше любимое кафе.

Программа должна позволять просматривать меню кафе, с картинками блюд и подробной информацией о каждом блюде – цене, составе, весу и пищевой и энергетической ценности.

В таблице меню должна быть реализована возможность выбрать блюда и оформить заказ. После оформления на экран выводится форма заказа с общей суммой и информацией по общей пищевой и энергетичеокой ценности заказа.

Также реализовать вывод отчета с информацией о сделанных заказах за произвольный период времени (с возможностью выбора дат). Реализовать вывод полученного заказа и отчета в файл и на печать.

8. СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ

Кодирование

При составлении исходного текста программ стоит придерживаться определенных стандартов и правил.

Стандарт оформления кода обычно принимается и используется некоторой группой разработчиков программного обеспечения с целью единообразного оформления совместно используемого кода. Такой стандарт сильно зависит от используемого языка программирования. Например, стандарт оформления кода для языка C/C++ будет серьёзно отличаться от стандарта для языка BASIC.

Стиль именования переменных, констант и функций. Соглашение об именах делает программы более понятными и упрощает их чтение. Также соглашение может дать информацию о функции, выполняемой тем или иным идентификатором. Например, является ли запись константой, пакетом или классом, что может быть полезным для понимания кода.

Переменные. Имена переменных, по возможности, должны быть короткими, но при этом отражать возложенные на переменную функции. Выбор имени переменной должен быть мнемонический, т.е. указывающий случайному читателю назначение той или иной переменной. Односимвольные имена являются исключением для временных «одноразовых» переменных.

Обычно таким переменные дают следующие имена:

- i, j, k, m, n для целых типов;
- c, d, s для символьных типов.

Константы. Имена переменных, объявленных константами внутри класса, и ANSI констант должны содержать символы только в верхнем регистре.

со словами разделяемыми символом подчеркивания («_»). ANSI констант следует избегать для упрощения отладки.

Пример:

static final int MIN_WIDTH = 4; static final int MAX_WIDTH = 999; static final int GET_THE_CPU = 1;

Количество операторов в строке. Для улучшения читаемости исходного текста программы рекомендуется писать не более одного оператора в строке, что вызвано особенностями человеческого восприятия текста. Кроме того, это облегчает пошаговую отладку в символьных отладчиках.

Пример:

Строки

int *ptr; ptr = new int [100];

ptr[0] = 0;

Следует оформить следующим образом:

int *ptr;

ptr = new int [100];

ptr[0] = 0;

Стиль отступов – правила форматирования исходного кода, в соответствии с которыми отступы проставляются в удобочитаемой манере. Редакторы текста, входящие в состав большинства популярных сред разработки, часто предоставляют средства для поддержки используемого стиля отступов, например, автоматическую вставку пробелов/табуляции при вводе скобок, обозначающих начало/конец логического блока. Желательно при программировании придерживаться стиля отступов, так как это во многом упрощает дальнейшую работу с исходным кодом и помогает избежать ошибок в структуре кода при его составлении.

Пустые строки. Использование пустых строк является важным средством для выделения участков программы. При этом имеет смысл отделять:

1) определения переменных:

char str[80];

```
int counter = 0;
fgets(str, 79, infile);
counter++;
```

2) последовательности однотипных инструкций или директив:

```
#include
#include
#include
#define NAME_SIZE 256
#define MAX_LEN 3000
```

3) функции:

```
int main()
{
...
}
char *get_name(FILE *f)
{
...
}
```

4) любые логически завершенные блоки кода:

```
printf( "Enter size and delta: " ); // Блок ввода данных
scanf( "%d", &size );
scanf( "%f", &delta );
for( i=0; i <size; i++ ) //Блок использования данных
{
a[i] -= delta;
b[i] += delta;
}
```

Комментарии. Большинство специалистов сходятся во мнении, что комментарии должны объяснять намерения программиста, а не код; то, что можно выразить на языке программирования, не должно выноситься в

комментарии – в частности, надо использовать говорящие названия переменных, функций, классов, методов и пр., разбивать программу на лёгкие для понимания части, стремиться к тому, чтобы структура классов и структура баз данных были максимально понятными и прозрачными и т. д.

Концепция грамотного программирования настаивает на включение в текст программы настолько подробных и продуманных комментариев, чтобы она стала исходным текстом не только для исполняемого кода, но и для сопроводительной документации.

Время, потраченное на написание комментариев, многократно окупится при любых модификациях программы. Однако комментировать все подряд включая самоочевидные действия тоже не стоит.

Комментировать следует:

- заголовок файла, описывая содержимое данного файла;
- заголовок функции, поясняя назначение ее аргументов и смысл самой функции;
- вводимые переменные и структуры данных;
- основные этапы и особенности реализуемых алгоритмов;
- любые места, которые трудны для быстрого понимания, в особенности использование различных программных "трюков" и нестандартных приемов.

Некоторые комментарии программисты используют в ходе своей работы. Подобные комментарии особенно полезны, когда над одним кодом работает несколько разработчиков. Так, комментарием TODO обычно помечают участок кода, который программист оставляет незавершённым, чтобы вернуться к нему позже. Комментарий FIXME помечает обнаруженную ошибку, которую решают исправить позже. Комментарий XXX или ZZZ обозначает найденную критическую ошибку, без исправления которой нельзя продолжать дальнейшую работу.

9. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

При оформлении пояснительной записки стоит обращать внимание на количество используемых стилей – в списке стилей должны быть только необходимые (остальные следует скрыть или удалить), это поможет привести части документа к единообразию.

1. Используются поля следующих размеров: слева – 30 мм; справа – 10 мм; сверху – 20 мм; снизу – 20 мм.

2. Нумерация страниц – сквозная, номер проставляется в правом нижнем углу страницы. На титульном листе номер не ставится.

3. Заголовки разделов и подразделов должны быть сформулированы кратко, на первом месте должно стоять имя существительное.

4. Все заголовки иерархически нумеруются. В конце заголовка точка не ставится. Такие разделы как «СОДЕРЖАНИЕ», «ВВЕДЕНИЕ», «ЗАКЛЮЧЕНИЕ», «СПИСОК ИСТОЧНИКОВ», «ПРИЛОЖЕНИЕ» не нумеруются.

5. Каждый раздел (заголовок 1-го уровня) следует начинать с новой страницы. Заголовок 1-го уровня следует располагать в середине строки и набирать прописными буквами. Заголовки 2-го уровня и ниже следует начинать с абзацного отступа и печатать с прописной буквы.

6. Заголовки следует отделять от окружающего текста промежутком размером не менее чем в 15 мм снизу и 30 мм сверху. После любого заголовка должен следовать текст, а не рисунок, формула или таблица.

7. Все рисунки, таблицы и формулы нумеруются. Нумерация может быть либо сквозной по всему тексту, например «таблица 7», либо по разделам, например «Рисунок – 2.5», что означает рисунок 5 в разделе 2. Ссылаться на рисунок следует как «рис. 2.5», на таблицу – «табл. 7». Номер формулы располагается справа от нее в круглых скобках. Нумеруются только те формулы, на которые есть ссылки в тексте.

8. Каждый рисунок должен иметь название. Название для таблиц не обязательно. Точка после названия рисунка и таблицы не ставится.

9. Название рисунка располагается под рисунком по центру. Название таблицы располагается над таблицей справа.

10. На каждый рисунок, таблицу и приложение в тексте должна быть ссылка в основной части пояснительной записки.

11. Для каждого вида текста (обычный, заголовки различных уровней, подписи к рисункам и таблицам) рекомендуется создать соответствующий стиль в текстовом редакторе и использовать его.

12. В разделе «СПИСОК ИСТОЧНИКОВ» помещаются только те источники, которые использовались при написании текста.

13. Ссылки на литературные источники в тексте приводятся в квадратных скобках, например, [1, 2] или [1, 3–7].

14. Приложения идентифицируются номерами или буквами, например «ПРИЛОЖЕНИЕ 1» или «ПРИЛОЖЕНИЕ А». На следующей строке, при необходимости, помещается название приложения, например «ЛИСТИНГ ПРОГРАММЫ», которое оформляется как заголовок 1- го уровня без нумерации.

10. ТРЕБОВАНИЕ К РУКОВОДСТВУ ПОЛЬЗОВАТЕЛЯ

Одной из важнейших составляющих любой законченной программы является руководство пользователя. От того, насколько понятно и доступно написано руководство, зависит успех программы, ее распространенность и популярность.

Существует 3 основных типа руководств:

1. Описание с пошаговой инструкцией. Это наиболее распространённый тип, в котором даётся руководство по выполнению различных действий в программе с точки зрения целей пользователя: каждое действие разбивается на несколько простых операций и пользователю предъявляется информация о том, каким образом и в какой последовательности он может их выполнять (как правило, описание сопровождается большим количеством изображений кнопок, панелей, форм и т.д.).

2. Другим типом является описание, ориентированное на перечисление и описание операций, допустимых в программе. В этом случае структура руководства будет совпадать с иерархией программных операций. В этом типе руководства не описывается, каким образом можно решить какую-либо довольно крупную задачу, предоставляя пользователю самому размышлять над этим вопросом.

3. И, наконец, последним типом является руководство для начинающих, которое помогает новичкам ознакомиться с программным продуктом, узнать, как работает программа и как управлять интерфейсом.

Несмотря на довольно существенные различия в подходах к написанию руководства пользователя, можно рекомендовать следующие обязательные разделы:

1. «О программе» – описание программы, назначение и основные возможности.

2. «Системные требования» – список аппаратных и программных средств и их характеристик, необходимых для запуска и успешного функционирования программы.

3. «Интерфейс» – описание интерфейса программы, основных элементов управления и горячих клавиш. При наличии графического интерфейса необходимо привести скриншоты, иллюстрирующие интерфейс пользователя.

4. «Запуск программы» – описание действий, необходимых для запуска программы.

5. «Работа с программой» – пошаговое описание основных действий (в соответствии с выбранным подходом к написанию руководства), которые доступны в программе, с пояснениями и скриншотами-примерами.

6. «Приложение». Необязательный раздел, добавляется при необходимости и может включать любые сведения, не вошедшие в вышеперечисленные разделы, например, глоссарий.

Необходимо учитывать, что руководство пользователя пишется для людей, которые, вполне возможно, плохо умеют работать с компьютером и не знакомы со многими понятиями информационных технологий. Поэтому руководство пользователя следует писать понятным языком, по возможности употребляя как можно меньше специфической терминологии и аббревиатур.

Также не допускается использование жаргонной лексики. Структура предложений должна быть как можно более простой, не перегруженной сложными речевыми оборотами.

11. ТРЕБОВАНИЕ К РУКОВОДСТВУ ПРОГРАММИСТА

Описание можно начать словами: "Функционирование программ : приложения формируется на базе следующих модулей:

Project.bpr – главный модуль, процедурами которого являются следующие модули: Unit1.cpp, MyUnit. cpp, ...

Unit1.dfm, Unit1cpp – модули, соответствующий главной форме Form1, с помощью которой осуществляется презентация данного программного продукта, а также вызов модулей: MyUnit.cpp, ...

MyUnit.cpp – модуль, содержащий классы для решения задачи (реализующий класс (имя класса) и его подкласс (наследник) (имя)).

Между модульной структурой и структурами данных существует связь, которая может быть представлена в виде следующей схемы (рис.1).

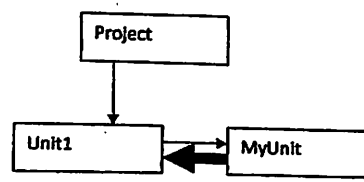

Рис.1. Схема связности модулей

В интерфейсе приложения были определены следующие информационные объекты:

1. Файл, содержащий исходные данные, который имеет следующую структуру: (описание структуры файла).
2. Компонент Edit1, позволяющий создать однострочное текстовое поле, предназначенное для задания числа строк в матрице.
3. Компонент Edit2, позволяющий создать однострочное текстовое поле, предназначенное для задания числа столбцов в матрице.
4. Компонент StringGrid1, позволяющий задать элементы матрицы.
5. Компонент Edit3, позволяющий создать однострочное текстовое поле, предназначенное для отображения на форме результата

ЛИТЕРАТУРА

1.1.Основная литература

1. Win32. Основы программирования. Финогенов К. Г. 2-е изд., испр. и дополн. -Москва: Диалог-МИФИ, 2006. - 411 с.

2. Языки программирования. Концепции и принципы. Кауфман В. Ш. Москва:ДМК Пресс, 2011. - 464 с.

3. Алгоритмы и программы на C++ Builder. Федоренко Ю. П. Москва: ДМКПресс, 2010. - 544 с.

4. Алгоритмы и структуры данных. Вирт Н. Москва: ДМК Пресс, 2010. - 272 с.

1.2.Дополнительная литература

1. Как программировать на C++. Х.М. Дейтел, П.Дж. Дейтел. Москва: Бином-Пресс, 2008 г., 1456 с.

2. Скользкие места C++. Как избежать проблем при проектировании и компиляции ваших программ. Дьюхэрст С. Москва: ДМК Пресс, 2006 г. - 264 с.

3. Введение в C++ Builder 4.0. Елманова Н. З., Кошель С. П. Москва: Диалог-МИФИ, 2000. - 304 с.

4. Эффективное использование C++. 55 верных способов улучшить структуру икод ваших программ. Дьюхэрст С. Москва: ДМК Пресс, 2006 г. - 264 с.

1.3.Интернет-ресурсы

1. Библиотека MSDN (по-русски), раздел - Средства разработки.

http://msdn.microsoft.com/ru-ru/library

2. Cppstudio.com,интернет-ресурс, посвященный программированию на C++

http://cppstudio.com/

Приложение 1

График выполнения курсовой работы

№№	Содержание блока	Число %	Число баллов/(накопленное)	Срок выполнения (номер недели/дата)	Реальные сроки
1	Получение задания. Анализ задачи. Описание объекта и анализ требований.	10%	10/(10)	2-3 (21 сентября)	
2	Разработка интерфейса и алгоритма реализации.	10%	10/(20)	4 (28 сентября)	
3	Проектирование интерфейса программного приложения.	5%	5/(25)	5 (5 октября)	
	Формализация расчетов.	15%	15/(40)	6-7 (30 октября)	
4	Разработка кода программы	15%	15/(55)	8-9 (20 ноября)	
5	Отладка программы	10%	10/(65)	10 (30 ноября)	
6	Тестирование	10%	10/(75)	11 (7декабря)	
7	Оформление отчета	10%	10/(85)	12-13 (21 декабря)	
8	Защита работы	15%	15/(100)	14-15 (10 января)	
	Итого:	100%	100		

Приложение 2

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕР ИНЖИНИРИНГ

КАФЕДРА «ОСНОВЫ ИНФОРМАТИКИ»

Разработка приложений средствами C++Builder

Пояснительная записка к курсовой работе

по курсу «Программирование на C/C++»

Выполнил: Студент гр.823-15
И.О.Фамилия
Проверил: доцент кафедры
И.О.Фамилия
1-член комиссии
И.О.Фамилия
2-член комиссии
И.О.Фамилия

Ташкент 2015

Приложение 3

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕР ИНЖИНИРИНГ

КАФЕДРА «ОСНОВЫ ИНФОРМАТИКИ»

УТВЕРЖДАЮ
Зав. Кафедрой «ОИ»
И.О.Фамилия
"____" _____

ЗАДАНИЕ

на выполнение курсовой работы

Студенту _____

Тема курсовой работы _____

1. Срок сдачи студентом готовой работы _____
2. Исходные данные к работе _____
3. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

_____

Перечень графического материала (с точным указанием обязательных чертежей) _____

Дата выдачи задания _____

Руководитель____(подпись)_____

Задание принял к исполнению_____(подпись)_____

Приложение 4

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕР ИНЖИНИРИНГ

КАФЕДРА «ОСНОВЫ ИНФОРМАТИКИ»

КЛИЕНТ -СЕРВЕР

Пояснительная записка к курсовой работе

по курсу «Программирование на C/C++»

Выполнил: Студент гр.823-15
И.О.Фамилия
Проверил: доцент кафедры
И.О.Фамилия
1-член комиссии
И.О.Фамилия
2-член комиссии
И.О.Фамилия

Ташкент 2015

В данной курсовой работе был рассмотрен процесс создания программы обмена сообщениями по сети (с рекомендацией использования компонент Internet). Приложение обеспечивает следующую функциональность: обмен сообщениями и файлами по сети с использованием протокола TCP, поиск компьютеров в сети и добавление их в список, возможность рассылки файлов и сообщений на несколько компьютеров.

Мазкур курс ишида тармоқда маълумот айирбошлаш дастури тузилиш жараёни кўриб чиқилди(Internet компоненталарини қўллаган ҳолда). Илова қуйидаги имкониятларга эга бўлди: TCP протоколини қўллаган ҳолда тармоқда маълумот ва файллар айирбошлаш, тармоқдаги компьютерлар қидирувини амалга ошириш ва уни рўйхатга қўшиш, файл ва маълумотларни бир нечта компьютерга узатиш имконияти.

Course work was devoted to creation of the messaging program over the network (by the component of the Internet). The application provides the following functionality: the exchange of messages and files over a network using the protocol TCP, search for computers on the network and add them to the list, the possibility to send files and messages on multiple computers.

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕР ИНЖИНИРИНГ

КАФЕДРА «ОСНОВЫ ИНФОРМАТИКИ»

УТВЕРЖДАЮ

Зав. Кафедрой "ОИ"

_____ И.О.Фамилия

"____" _____

ЗАДАНИЕ

на выполнение курсовой работы

Студенту_И.О.Иванову_____

Тема курсовой работы*Разработка программы "Клиент-сервер"* _____

_____

4. Срок сдачи студентом готовой работы __15.12.2015г._____
5. Исходные данные к работе _*научно-техническая литература в соответствии темы, сведения с Internet,Microsoft Visual Studio 2010*

_____

6. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Введение_____ с.

1. Теоретическаячасть_____

2. Основная часть_____

Заключение_____

_____

Перечень графического материала (с точным указанием обязательных чертежей) _*Презентация MicrosoftOfficePowerPoint 2007*_____

Дата выдачи задания _7.09.2015_____

Руководитель_____(подпись)_____

Задание принял к исполнению_____(подпись)_____

Содержание

ВВЕДЕНИЕ

Протокол передачи данных — набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Эти соглашения задают единообразный способ передачи сообщений и обработки ошибок при взаимодействии программного обеспечения разнесённой в пространстве аппаратуры, соединённой тем или иным интерфейсом.

Стандартизированный протокол передачи данных также позволяет разрабатывать интерфейсы (уже на физическом уровне), не привязанные к конкретной аппаратной платформе и производителю (например, USB, Bluetooth).

Сигнальный протокол используется для управления соединением — например, установки, переадресации, разрыва связи. Примеры протоколов: RTSP, SIP. Для передачи данных используются такие протоколы как RTP.

Сетевой протокол— набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами.

Разные протоколы зачастую описывают лишь разные стороны одного типа связи. Названия «протокол» и «стек протоколов» также указывают на программное обеспечение, которым реализуется протокол.

Новые протоколы для Интернета определяются IETF, а прочие протоколы — IEEE или ISO. ITU-T занимается телекоммуникационными протоколами и форматами.

Наиболее распространённой системой классификации сетевых протоколов является так называемая модель OSI, в соответствии с которой протоколы делятся на 7 уровней по своему назначению — от физического (формирование и распознавание электрических или других сигналов) до прикладного (интерфейс программирования приложений для передачи информации приложениями).

Сетевые протоколы предписывают правила работы компьютерам, которые подключены к сети. Они строятся по многоуровневому принципу. Протокол некоторого уровня определяет одно из технических правил связи. В настоящее время для сетевых протоколов используется модель OSI (Open System Interconnection — взаимодействие открытых систем, ВОС).

Модель OSI — это 7-уровневая логическая модель работы сети. Модель OSI реализуется группой протоколов и правил связи, организованных в несколько уровней:

- на физическом уровне определяются физические (механические, электрические, оптические) характеристики линий связи;
- на канальном уровне определяются правила использования физического уровня узлами сети;
- сетевой уровень отвечает за адресацию и доставку сообщений;
- транспортный уровень контролирует очередность прохождения компонентов сообщения;
- задача сеансового уровня — координация связи между двумя прикладными программами, работающими на разных рабочих станциях;
- уровень представления служит для преобразования данных из внутреннего формата компьютера в формат передачи;
- прикладной уровень является пограничным между прикладной программой и другими уровнями — обеспечивает удобный интерфейс связи сетевых программ пользователя.

Примеры сетевых протоколов

TCP/IP— набор протоколов передачи данных, получивший название от двух принадлежащих ему протоколов: TCP (англ. Transmission Control Protocol) и IP (англ. Internet Protocol)

Наиболее известные протоколы, используемые в сети Интернет:

HTTP (Hyper Text Transfer Protocol) — это протокол передачи гипертекста. Протокол HTTP используется при пересылке Web-страниц с одного компьютера на другой.

FTP (File Transfer Protocol) — это протокол передачи файлов со специального файлового сервера на компьютер пользователя. FTP дает возможность абоненту обмениваться двоичными и текстовыми файлами с любым компьютером сети. Установив связь с удаленным компьютером, пользователь может скопировать файл с удаленного компьютера на свой или скопировать файл со своего компьютера на удаленный.

POP (Post Office Protocol) — это стандартный протокол почтового соединения. Серверы POP обрабатывают входящую почту, а протокол POP предназначен для обработки запросов на получение почты от клиентских почтовых программ.

SMTP (Simple Mail Transfer Protocol) — протокол, который задает набор правил для передачи почты. Сервер SMTP возвращает либо подтверждение о приеме, либо сообщение об ошибке, либо запрашивает дополнительную информацию.

telnet — это протокол удаленного доступа. TELNET дает возможность абоненту работать на любой ЭВМ сети Интернет, как на своей собственной, то есть запускать программы, менять режим работы и так далее. На практике возможности лимитируются тем уровнем доступа, который задан администратором удаленной машины.

Другие протоколы:

DTN — протокол, предназначенный для обеспечения сверхдальней космической связи.

2.1 ТЕХНОЛОГИЯ «КЛИЕНТ - СЕРВЕР»

Каждый человек, работавший с компьютером, слышал о технологии клиент-сервер. В основе этой технологии лежат два основных понятия: клиент и сервер. Клиент - компьютер, осуществляющий запрос к серверу на выполнение каких-либо действий или предоставление какой-либо информации. Сервер - компьютер, обычно более мощный, чем компьютер-клиент. Модель функционирования такой системы заключается в следующем: клиент делает запрос серверу, сервер (серверная часть) получает запрос, выполняет его и отсылает результат клиенту (клиентская часть).

Сервер может обслуживать нескольких клиентов одновременно. В этом случае говорят о многопользовательском режиме. Только не стоит понимать слово "одновременно" в буквальном смысле. Запросы выполняются сервером последовательно. Если одновременно приходит более одного запроса, то запросы устанавливаются в очередь. В данном случае очередь - это список невыполненных клиентских запросов. Иногда запросы могут иметь приоритеты. Приоритет - это уровень "важности" выполнения запроса. Запросы с более высокими приоритетами должны выполняться раньше.

Существует маленькое исключение по поводу последовательной обработки. Операционная система Windows является многозадачной и многопоточной. Многозадачность - это свойство операционной системы исполнять несколько пользовательских приложений (программ).

Это означает, что на сервере может быть запущено несколько задач, каждая из которых может исполнять свой отдельный запрос. Много поточность дает возможность исполнять несколько запросов в пределах одной задачи. Что это дает? - Происходит увеличение производительности сервера.

Цикл выполнения запроса состоит из пересылки запроса и ответа между клиентом и сервером и непосредственным выполнением этого запроса на сервере.

Как правило, выполнение запроса (сложного запроса) происходит дольше, чем время, затрачиваемое на пересылку запроса и результата. Поэтому в серверах применяются системы распределения транзакций (запросов) между доменами

(частями) сервера. Транзакция - это набор команд, которые либо выполняются все или не выполняется ни одна. Домен - это элемент сервера. Нередко сервер на физическом уровне представляет собой не один, а группу компьютеров, объединенных в одну систему для обеспечения каких-либо операций. В этом случае домен - это компьютер, входящий в архитектуру сервера.

Существует концепции построения системы клиент-сервер:

1) Слабый клиент - мощный сервер - вся обработка информации осуществляется целиком сервером. Сервер посылает готовый результат, не требующий дополнительной обработки. Клиент только ведет диалог с пользователем: составляет запрос, отсылает запрос, принимает запрос и выводит информацию на экран (на принтер, в файл).

2) Сильный клиент - часть обработки информации перепоручается клиенту. Простой пример: пользователю требуется список сотрудников фирмы, отсортированный по фамилиям. В первом случае, сервер по клиентскому запросу извлекает фамилии (например, из базы данных) и сортирует их сам. Готовый отсортированный список отсылается клиенту. Клиенту необходимо только осуществить вывод на экран. Во втором случае, сервер только извлекает фамилии и отсылает данные клиенту. Клиент сортирует список и выводит его на экран.

Но это далеко не все. Допустим, пользователь захотел отсортировать полученный список по уровню зарплаты. В первом случае клиент формирует новый запрос к серверу, сервер его выполняет и посылает готовый ответ клиенту, который только выводит его на экран. Во втором случае клиент сортирует список сам без обращения к серверу. Естественно, во втором случае сервер будет менее загружен.

Существует еще одно важное понятие "время ожидания". "Время ожидания" - это время, через которое пользователь, послав запрос серверу, получит от него ответ. Время ожидания наиболее важный показатель производительности системы, реализующей концепцию клиент-сервер.

При частичной обработке данных на клиенте "время ожидания" меньше. Меньше оно за счет упрощения запроса и времени его выполнения. Отсюда меньше ожидание в очереди для исполнения запроса.

Безусловно, что время конечной обработки на клиенте может быть немного выше за счет разности в производительности сервера. Отсюда может несколько возрастать время ожидания. В конечном счете, это все равно выгоднее, т.к. время ожидания в очереди запросов на сервере меньше.

Многие серверы не выдерживают нагрузки ("напора" запросов) и просто "подвисают". В таком случае есть два альтернативных способа: увеличение производительности и перенос части операций над данными на клиента. Как правило, увеличение производительности значительно более дорогостоящая операция, и тоже, в своем смысле, конечная. Остается только "разгрузка" сервера и перенос части обработки данных на клиента.

P.S. Нередко в ряде организаций в качестве сервера сознательно используют устаревшие компьютеры. На них хорошо размещать файловые хранилища, принт-серверы (к нему подключается офисный принтер), WEB-серверы (Интернет-серверы), небольшие базы данных (серверную часть). Это оправдано с экономической точки зрения.Применение самых мощных компьютеров в качестве серверов целесообразно в банковской сфере, т.к. объем платежей неизменно возрастает. Соответственно, возрастает необходимый объем вычислительных ресурсов. Здесь оправдана "борьба" за единицы процентов увеличения производительности. Технология "клиент-сервер" описывает взаимодействие между двумя компьютерами, согласно которому клиент запрашивает у сервера некоторые услуги, а сервер обслуживает запрос.

Наверняка, вы когда-нибудь, покупали железнодорожные билеты в кассе. Вспомните, как это происходит. Вы заказываете нужный вам билет, а кассирша сообщает вам о наличии мест на данный поезд. Откуда она знает об этом? С помощью своего компьютера (клиента) она обращается к главному компьютеру (серверу), к которому подключены все компьютеры находящиеся в кассах.

Таким образом, она имеет самую достоверную информацию о проданных билетах и вы никогда не купите два билета на одно и то же место.

Технология клиент-сервер широко используется, например, в банковской системе. Для того чтобы проверить счет пользователя в банке, его компьютер направляет запрос серверной программе, выполняющейся на банковской машине. Требуемая информация возвращается клиентской программе, которая отображает данные, предназначенные для пользователя.

В случае с WWW клиентами выступают хорошо вам знакомые программы Web-браузеры (например, Microsoft Internet Explorer). Серверами же являются так называемые Web-серверы, обрабатывающие запросы Web-браузеров и высылающие им требуемые Web-страницы.

С термином «Web-сервер» (как и термином «сервер») существует определенная путаница. Во-первых, так называют компьютер, подключенный к сети Интернет и хранящий на своих жестких дисках файлы Web-страниц. Во-вторых, так называется программа, работающая на этом самом компьютере, принимающая от Web-браузеров запросы и выдающая им соответствующие файлы. Но чаще и компьютер, и программу объединяют в единое целое и называют одним словом – Web-сервер.

Когда вы набираете в поле адреса Web-браузера какой-либо интернет-адрес, Web-браузер обращается к соответствующему Web-серверу. Сервер же извлекает со своих дисков нужные файлы (сама Web-страница, изображения, внедренные объекты, архивы, исполняемые файлы) и отправляет их Web-браузеру. В результате вы и получите соответствующую Web-страницу.

Понимая работу программ, соответствующих архитектуре клиент-сервер, вы сможете более успешно разрешать проблемы, связанные с передачей и отображением Web-страниц.

2.2. ЦЕЛИ И ЗАДАЧИ КУРСОВОЙ РАБОТЫ

Цель курсовой работы: «Создание программы обмена сообщениями по сети (с рекомендацией использования компоненты Internet). Приложение должно обеспечивать следующую функциональность: обмен сообщениями и файлами по сети с использованием протокола TCP, поиск компьютеров в сети и добавление их в список, возможность рассылки файлов и сообщений на несколько компьютеров.»

Помимо базовых служб и протоколов Интернет существует широкий набор дополнительных сервисов, возможности которых часто используются Интернет-разработчиками. К тому же далеко не всегда возможность отображения информации с помощью браузера является приемлемым решением для Интернет-приложений. В этом случае разумно использовать Интернет-инфраструктуру для обмена данными, а отображение информации обеспечить за счет более сложных клиентских приложений, разработанных, предположим, на C++.

Допустим, требуется реализовать специализированную серверную логику, которая не заложена в стандартные Web-серверы. Для решения такого класса задач в состав C++ включена библиотека Internet Direct (Indy) компании Nevrona Designs .Данная библиотека, разработанная специально для Borland Delphi, насчитывает уже восемь версий, последняя из которых вошла в состав новой версии Delphi. Набор компонентов разделен на три группы: клиентские (Indy Client), серверные (Indy Servers) и вспомогательные (Indy Misc).

Indy Clients и Indy Servers

Большинство компонентов Indy Client и Indy Servers представляют собой пары, соответствующие клиентским и серверным частям протоколов и служб (за исключением отдельных, в основном серверных, компонентов типа TunnelMaster и TunnelSlave), и позволяют использовать такие протоколы, как TCP/IP, UDP, NNTP, SMTP, FTP, HTTP, а также службы ECHO, FINGER, WHOIS и т.д.

Клиентские компоненты Indy написаны с использованием сокетов. Сокет со стороны клиента требует соединения с сервером. Если связь установлена, клиент и сервер могут начинать обмен сообщениями. Эти сообщения носят различный характер, но обычно обмен происходит по определенному протоколу (например, HTTP) TIdTCPClient и TIdTCPServer

Эти компоненты используются для поддержки одного из основных сетевых протоколов — TCP (Transmission Control Protocol), а также являются базовыми классами для компонентов TIdSMTP и TIdFTP. Класс TIdTCPServer обладает свойством ThreadMgr, по умолчанию равным nil. Если ThreadMgr равно nil, когда TIdTCPServer активизирован, класс TIdThreadMgrDeafault будет создан неявно. В противном случае используется установленный менеджер процессов.

В приложении будем использовать 2 формы:

1) Первую (Главную) - на неё поместим компоненту для хранения истории сообщений (например TlistBox, TMemo или TRichEdit), поле для вводе текста (TEdit или TMemo) кнопки «отправить файл» и «отправить сообщение», TListBox или TcomboBox для выбора IP получателя.

2) Вторую (Для поиска компьютеров) — на ней необходимо реализовать структуру для ввода IP адресов, задающих диапазон сканирования (например TMaskEdit или доисать ограничения к TEdit на ввод только цифр).

2.3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ

Открываем программу.

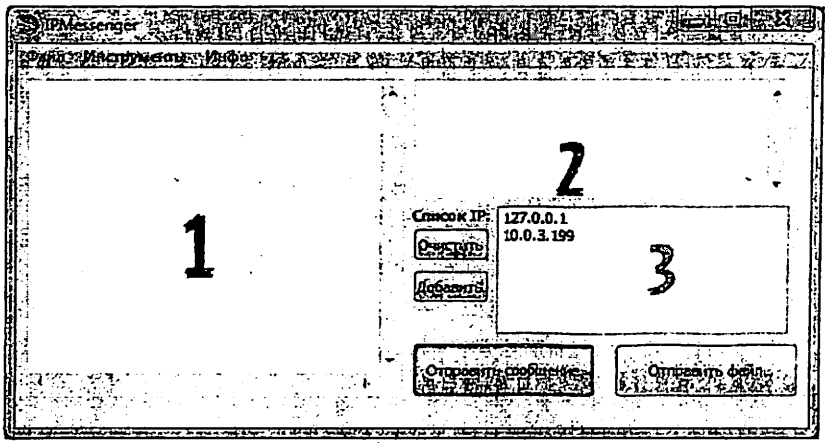

Рис. 2.1

В этой программе можно увидеть 3 поля.

1. Первое поле для чтение отправленных сообщений
2. Второе поле для ввода сообщений
3. В третьем поле отображается список IP-адресов компьютеров которая запушена эта программа

Чтобы найти собеседника нажмите на кнопку «добавить» и в новом окне открывается поиск компьютеров по IP-адресу

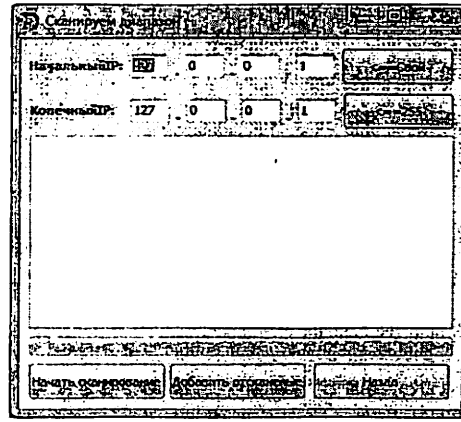

Рис. 2.2

В этом окне вы можете ввести диапазон адресов, которые подвергаются поиску.

В «Начальный IP» вы должны ввести начальный диапазон IP-адреса. Чтобы узнать свой IP-адрес нажмите на кнопку «Свой»

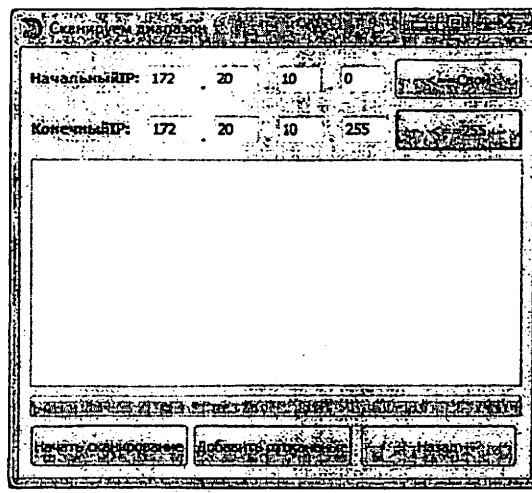

Рис. 2.3

Для выбора конечный IP нажмите на кнопку «255». Чтобы сканировать нажмите на кнопку «сканировать» и программа отобразит все адреса, запущенные этой программой.

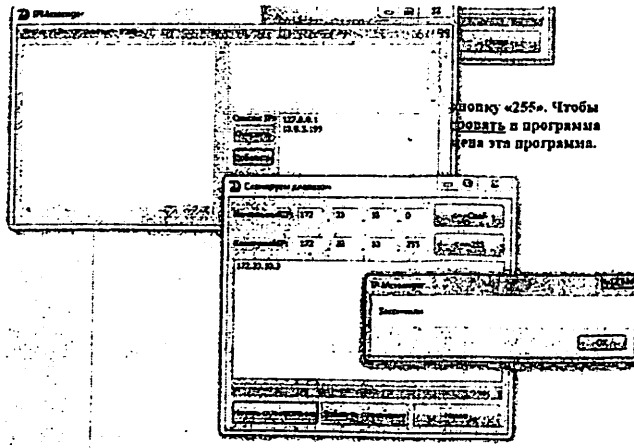

Рис. 2.4

После этого нажмите на кнопку «Добавить отсканированные» чтобы добавить адреса:

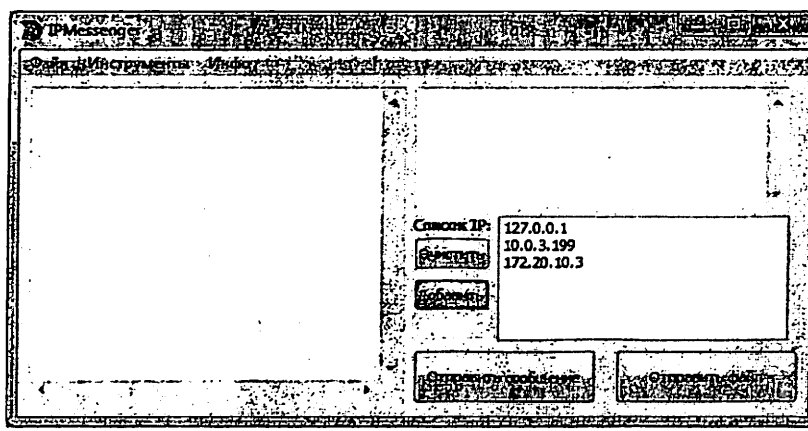

Рис. 2.5

И в нашей программе будет отображена отсканированные IP адреса.

Для отправки сообщений выберите адрес который вы хотите отправить и нажмите на «отправить сообщение».

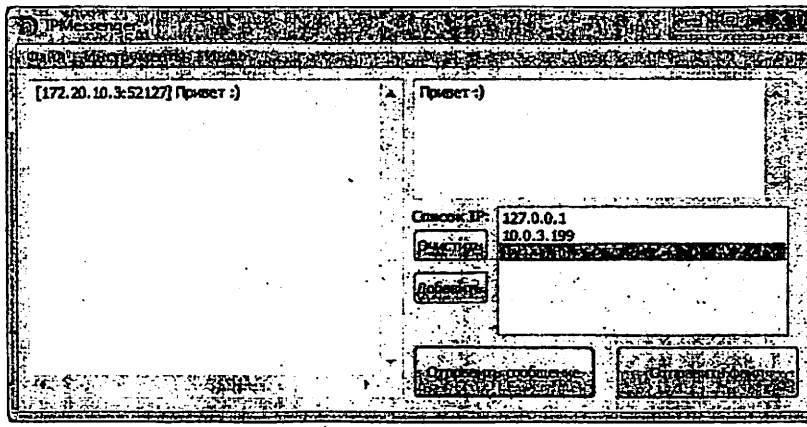

Рис. 2.6

Отправка файлов.

Чтобы отправить файл выберите адрес получателя и нажмите «отправить файл». Затем вам будет отображено окно выбора файла. После отправки файла в программе получателя будет представлено окно информации полученного файла и этот файл сохраняется в папке Incoming:

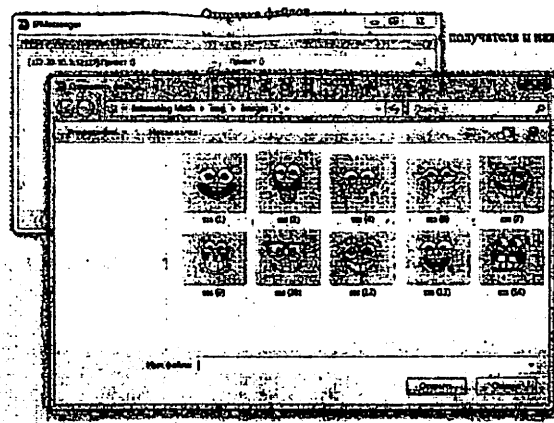

Рис. 2.7

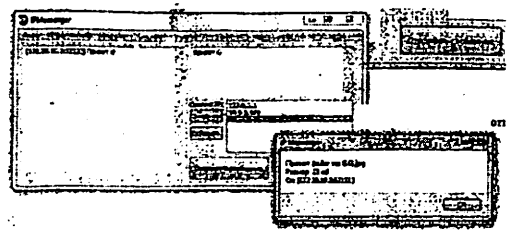

Рис. 2.8

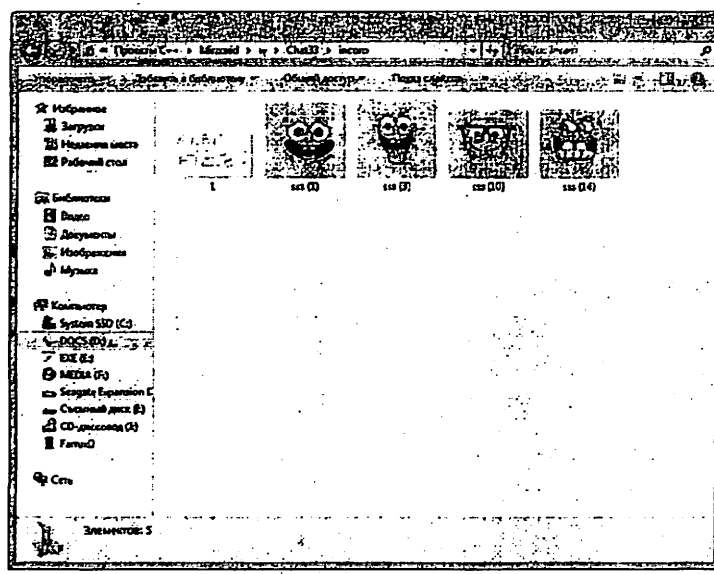

Рис. 2.9

2.4. РУКОВОДСТВО ПРОГРАММИСТУ

Программа разработана на языке C++. Среда разработки C++ Builder 2007.

В программе имеется единая форма и для сервера, и для клиента.

Компоненты, которые использованные в программе

Memo1 – Текстовое поле для написания сообщений

Memo2 - Текстовое поле для чтения сообщений

Listbox – Список добавленных IP – адресов и Компоненты IndyClients

IdTcpClient, IdTCPServer – Протокол для клиента и сервера

IdAntiFreeze - Чтобы при операциях клиентских компонентов (напр., IdTcpClient и т.п.), которые могут заблокировать работу главного потока приложения, периодически вызывать Application.ProcessMessages. Период этот и будет равен тому самому свойству IdleTimeOut. Получается, что пользовательский интерфейс "как бы не замерзает" на время выполнения этой операции.

ЗАКЛЮЧЕНИЕ

В ходе работы научился использовать встроеные функции WinAPI, освоил работу с компонентами Indy, улучшил умение работать в визуальной среде разработки, закрепил навыки работы с компонентами Builder C++.

Часто выбор протокола передачи файлов ограничен возможностями используемой коммуникационной программы. Тем не менее, существует возможность подключения некоторых протоколов передачи файлов к отдельным терминальным программам, которые штатно не предусматривают их использование. Это возможно в основном для большинства программ для DOS. Подключаемые внешние протоколы в таком случае должны быть в виде исполняемых (*.exe) файлов.

Одним из самых быстрых является протокол HyperProtocol. Как иZmodem, он является потоковым протоколом, но помимо этого может сжимать передаваемые данные, как и протокол Kermit. Согласно протокола HyperProtocol приемник посылает подтверждение не после каждого файла, а в конце всего сеанса передачи. Он может быть с успехом использован для передачи информации по высокоскоростным каналам.

СПИСОК ЛИТЕРАТУРЫ

1. Как программировать на C++. Х.М. Дейтел, П.Дж. Дейтел. Москва: Бином-Пресс, 2008 г., 1456 с.
2. codeforces.ru
3. cyberforum.ru
4. Освой самостоятельно C++ за 21 день
5. alfaskins.com

Листинг кода программы

Листинг исходных кодов программы с комментариями.

Main.cpp

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "main.h"
#include "skan.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "IdCmdTCPClient"
#pragma link "IdCmdTCPServer"
#pragma link "IdCustomTCPServer"
#pragma resource "*.dfm"
TForm1 *Form1;
String TestIP() //Узнать свой IP
{
AnsiString out = "WinSock ERR";
WSADATA wsaData;
if (!WSAStartup(WINSOCK_VERSION, &wsaData)){char chInfo[64];
if (!gethostname(chInfo, sizeof(chInfo)))
{
hostent *sh;
sh=gethostbyname((char*)&chInfo);
if (sh!=NULL)
{
int nAdapter = 0;
while (sh->h_addr_list[nAdapter])
{
struct sockaddr_in adr;
memcpy(&adr.sin_addr, sh->h_addr_list[nAdapter], sh->h_length);
out = inet_ntoa(adr.sin_addr);
nAdapter++;
}
}
}
}
WSACleanup();
return out;
}
```

```
bool ChkEdit(int Length,char Key) //Проверяем Edit на нажатие цифры
{
bool error=true;
if ((Key >= '0') && (Key <= '9')) error=false;
if (Key == 8) error=false;
if (Length>=3 && Key!=8) error=true;
return error;
}
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
for(int x=0;x<ListBox1->Items->Count;x+=1)
{
if(ListBox1->Selected[x])
{
IdTCPClient1->Host=ListBox1->Items->Strings[x];//Выставляем IP
IdTCPClient1->Port=2459;//Выставляем порт для текстовых сообщений
try{
IdTCPClient1->Connect(); //Подключаемся.
}
catch(...){ShowMessage("Ошибка подключения.");}//Если не удалось
if(IdTCPClient1->Connected())//Если подключились
{
IdTCPClient1->Socket->WriteLn(Memo1->Lines->Text); //Отсылаем команду.
IdTCPClient1->Disconnect(); //Отключаемся
}
}
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
ListBox2->Items->Clear();//Там мы храним данные
for(int x=0;x<ListBox1->Items->Count;x+=1)
{
if(ListBox1->Selected[x])
{
IdTCPClient1->Host=ListBox1->Items->Strings[x];
IdTCPClient1->Port=2458;//Выставляем порт для файлов
if(OpenDialog1->Execute()){
```

```
try{
IdTCPClient1->Connect(); //Подключаемся
}
catch(...){ShowMessage("Ошибка подключения.");}//Если не подключиличь
if(IdTCPClient1->Connected()){//Если подключились
Screen->Cursor = crHourGlass;//Курсор ожидания
ListBox2->Items->Add(ExtractFileName(OpenDialog1->FileName));//Имя файла
int fh=FileOpen(OpenDialog1->FileName,Sysutils::fmOpenRead);
int fl=FileSeek(fh,0,2);
FileClose(fh);
ListBox2->Items->Add(IntToStr(fl));//Размер файла
//посылаем файл
IdTCPClient1->Socket->WriteRFCStrings(ListBox2->Items,true);
IdTCPClient1->Socket->WriteBufferOpen();
IdTCPClient1->Socket->WriteBufferClear();
IdTCPClient1->Socket->WriteBufferFlush(1024);
IdTCPClient1->Socket->WriteFile(OpenDialog1->FileName,true);
IdTCPClient1->Socket->WriteBufferClose();
Screen->Cursor = crDefault;//Нормальный курсор
IdTCPClient1->Disconnect();//Отключаемся
ListBox2->Items->Clear();//Очищаем
}
}
}
}
}
//---------------------------------------------------------------------------


void __fastcall TForm1::IdTCPServer1Execute(TIdContext *AContext)
{
String msg=AContext->Connection->Socket->ReadLn();//Принимаем сообщение
Memo2->Lines->Add("["+AContext->Connection->Socket->Binding-
>PeerIP+":"+StrToInt(AContext->Connection->Socket->Binding->PeerPort)+"]
"+msg);//Добавляем сообщение в список
AContext->Connection->Disconnect();//Разъединяемся
if (Application->MainForm->WindowState!=wsNormal){//Если форма
свёрнута(или на весь экран) показываем сообщение из трэя
TrayIcon1->BalloonTitle="Новое сообщение";
TrayIcon1->BalloonHint="От: ["+AContext->Connection->Socket->Binding-
>PeerIP+":"+StrToInt(AContext->Connection->Socket->Binding->PeerPort)+"]
\n\r"+msg;
TrayIcon1->BalloonFlags=bfInfo;
TrayIcon1->BalloonTimeout=10;
TrayIcon1->ShowBalloonHint();
```

```
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::IdTCPServer2Execute(TIdContext *AContext)
{
ListBox2->Items->Clear();//Очищаем временную переменную
AContext->Connection->Socket->ReadStrings(ListBox2->Items,2);//Считываем
необходимое инфо
AContext->Connection->Socket->ReadLn();
AnsiString C_Path = ExtractFileDir(Application->ExeName)+"\\incom\\"+
ListBox2->Items->Strings[0];//Путь к принятому файлу


TFileStream *f;//Создаём поток
if(!FileExists(C_Path)){
f = new TFileStream(C_Path,Classes::fmCreate);
}
else
{
f = new TFileStream(C_Path,Sysutils::fmOpenWrite);
}
AContext->Connection->Socket->ReadStream(f,StrToInt(ListBox2->Items-
>Strings[1]),false);//Принимаем файл
AContext->Connection->Disconnect();
if (Application->MainForm->WindowState!=wsNormal){//Если форма
свёрнута(или на весь экран) показываем сообщение из трэя
TrayIcon1->BalloonTitle="Принят файл";
TrayIcon1->BalloonHint="От: ["+AContext->Connection->Socket->Binding-
>PeerIP+":"+StrToInt(AContext->Connection->Socket->Binding->PeerPort)+"]
\n\rИмя:"+ListBox2->Items->Strings[0]+" размер:
"+FloatToStr(StrToInt(ListBox2->Items->Strings[1])/1024)+" кб";
TrayIcon1->BalloonFlags=bfInfo;
TrayIcon1->BalloonTimeout=10;
TrayIcon1->ShowBalloonHint();
}
else
ShowMessage("Принят файл: "+ListBox2->Items->Strings[0]+"\n\rРазмер:
"+FloatToStr(StrToInt(ListBox2->Items->Strings[1])/1024)+" кб\n\rОт:
["+AContext->Connection->Socket->Binding->PeerIP+":"+StrToInt(AContext-
>Connection->Socket->Binding->PeerPort)+"]");
ListBox2->Items->Clear();//Очищаем временную переменную
f->Free();//Удаляем поток
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N3Click(TObject *Sender)
{
Application->Terminate();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N2Click(TObject *Sender)
{
if(SaveDialog1->Execute())
Memo2->Lines->SaveToFile(SaveDialog1->FileName);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N6Click(TObject *Sender)
{
Button2->Click();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N8Click(TObject *Sender)
{
ShowMessage("Курсоваяработа 3-йкурс\n\r\t2008(c)");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::CGCBuilder1Click(TObject *Sender)
{
ShellExecute(0,NULL,"http://www.codegear.com/",NULL,NULL,SW_SHOW);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::TrayIcon1DblClick(TObject *Sender)
{
if (!hided)//Сворачиваем/разворачиваем форму в/из трея
    {
    ShowWindow(Application->MainFormHandle,SW_HIDE);
    hided=true;
    }
else
    {
    ShowWindow(Application->MainFormHandle,SW_SHOW);
    hided=false;
    }
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
hided=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N5Click(TObject *Sender)
{
Form2->ShowModal();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button3Click(TObject *Sender)
{
ListBox1->Items->Clear();
ListBox1->Items->Add("127.0.0.1");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
Form2->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::IdTCPServer3Execute(TIdContext *AContext)
{
AContext->Connection->Socket->ReadLn();
AContext->Connection->Disconnect();
}
//---------------------------------------------------------------------------
Main.dfm
object Form1: TForm1
 Left = 0
 Top = 0
 Caption = 'IPMessenger'
 ClientHeight = 259
 ClientWidth = 593
 Color = clBtnFace
 Font.Charset = DEFAULT_CHARSET
 Font.Color = clWindowText
 Font.Height = -11
 Font.Name = 'Tahoma'
 Font.Style = []
```

```
Menu = MainMenu1
OldCreateOrder = False
OnCreate = FormCreate
PixelsPerInch = 96
TextHeight = 13
object Label1: TLabel
Left = 295
Top = 104
Width = 58
Height = 13
Caption = #1057#1087#1080#1089#1086#1082' IP:'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Button1: TButton
Left = 296
Top = 208
Width = 137
Height = 41
Caption=#1054#1090#1087#1088#1072#1074#1080#1090#1100'
'#1089#1086#1086#1073#1097#1077#1085#1080#1077
TabOrder = 0
OnClick = Button1Click
end
object Button2: TButton
Left = 448
Top = 208
Width = 137
Height = 41
Caption = #1054#1090#1087#1088#1072#1074#1080#1090#1100'
'#1092#1072#1081#1083
TabOrder = 1
OnClick = Button2Click
end
object ListBox2: TListBox
Left = 440
Top = 16
Width = 121
Height = 97
ItemHeight = 13
TabOrder = 2
```

```
 Visible = False
 end
 object Memo1: TMemo
 Left = 296
 Top = 8
 Width = 281
 Height = 90
 ScrollBars = ssVertical
 TabOrder = 3
 end
 object Memo2: TMemo
 Left = 8
 Top = 8
 Width = 281
 Height = 241
 ReadOnly = True
 ScrollBars = ssBoth
 TabOrder = 4
 end
 object ListBox1: TListBox
 Left = 359
 Top = 104
 Width = 218
 Height = 97
 Hint = #1057#1087#1080#1089#1086#1082' IP
'#1072#1076#1088#1077#1089#1086#1074
 ItemHeight = 13
 Items.Strings = (
 '127.0.0.1'
 '10.0.3.199')
 MultiSelect = True
 ParentShowHint = False
 ShowHint = True
 TabOrder = 5
 end
 object Button3: TButton
 Left = 296
 Top = 120
 Width = 57
 Height = 25
 Caption = #1054#1095#1080#1089#1090#1080#1090#1100
 TabOrder = 6
 OnClick = Button3Click
 end
 object Button4: TButton
```

```
Left = 296
Top = 152
Width = 57
Height = 25
Caption = #1044#1086#1073#1072#1074#1080#1090#1100
TabOrder = 7
OnClick = Button4Click
end
object IdTCPClient1: TIdTCPClient
BoundIP = '10.0.3.199'
ConnectTimeout = 10
Host = '10.0.3.199'
IPVersion = Id_IPv4
Port = 2459
ReadTimeout = -1
Left = 368
Top = 16
end
object OpenDialog1: TOpenDialog
OptionsEx = [ofExNoPlacesBar]
Title = #1054#1090#1087#1088#1072#1074#1080#1090#1100'
'#1092#1072#1081#1083
Left = 368
Top = 104
end
object IdTCPServer1: TIdTCPServer
Active = True
Bindings = <
item
Port = 2459
end>
DefaultPort = 2459
OnExecute = IdTCPServer1Execute
Left = 368
Top = 56
end
object IdTCPServer2: TIdTCPServer
Active = True
Bindings = <
item
Port = 2458
end>
DefaultPort = 2458
OnExecute = IdTCPServer2Execute
Left = 400
```

```
Top = 56
end
object IdAntiFreeze1: TIdAntiFreeze
IdleTimeOut = 10
OnlyWhenIdle = False
Left = 400
Top = 16
end
object TrayIcon1: TTrayIcon
Hint = 'IP Messanger v1.0'
Visible = True
OnDblClick = TrayIcon1DblClick
Left = 400
Top = 136
end
object MainMenu1: TMainMenu
Left = 368
Top = 136
object N1: TMenuItem
Caption = #1060#1072#1081#1083
object N2: TMenuItem
Caption = #1057#1086#1093#1088#1072#1085#1080#1090#1100'
'#1080#1089#1090#1086#1088#1080#1102
OnClick = N2Click
end
object N3: TMenuItem
Caption = #1042#1099#1093#1086#1076
OnClick = N3Click
end
end
object N4: TMenuItem
Caption = #1048#1085#1089#1090#1088#1091#1084#1077#1085#1090#1099
object N5: TMenuItem
Caption = #1057#1082#1072#1085#1080#1088#1086#1074#1072#1090#1100'
'#1076#1080#1072#1087#1072#1079#1086#1085
OnClick = N5Click
end
object N6: TMenuItem
Caption = #1054#1090#1087#1088#1072#1074#1080#1090#1100'
'#1092#1072#1081#1083
OnClick = N6Click
end
end
object N7: TMenuItem
Caption = #1048#1085#1092#1086
```

```
object N8: TMenuItem
Caption = #1054' '#1087#1088#1086#1075#1088#1072#1084#1084#1077
OnClick = N8Click
end
object CGCBuilder1: TMenuItem
Caption = 'CG C++ Builder'
OnClick = CGCBuilder1Click
end
end
end
object SaveDialog1: TSaveDialog
DefaultExt = '.txt'
Filter = #1058#1077#1082#1089#1090#1086#1074#1099#1081'
'#1076#1086#1082#1091#1084#1077#1085#1090'|*.txt|'#1051#1102#1073#1086#1
081' '#1090#1080#1087'|*.*'
OptionsEx = [ofExNoPlacesBar]
Title = #1057#1086#1093#1088#1072#1085#1080#1090#1100'
'#1080#1089#1090#1086#1088#1080#1102'
'#1089#1086#1086#1073#1097#1077#1085#1080#1081
Left = 400
Top = 104
end
object IdTCPServer3: TIdTCPServer
Active = True
Bindings = <
item
Port = 2457
end>
DefaultPort = 2457
OnExecute = IdTCPServer3Execute
Left = 432
Top = 56
end
end
```

Main.h

```
//---------------------------------------------------------------------------

#ifndef mainH
#define mainH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "IdCmdTCPClient.hpp"
```

```
#include "IdCmdTCPServer.hpp"
#include "IdCustomTCPServer.hpp"
#include <IdBaseComponent.hpp>
#include <IdComponent.hpp>
#include <IdTCPClient.hpp>
#include <IdTCPConnection.hpp>
#include <IdTCPServer.hpp>
#include <Dialogs.hpp>
#include <IdAntiFreeze.hpp>
#include <IdAntiFreezeBase.hpp>
#include <ExtCtrls.hpp>
#include <Mask.hpp>
#include <Menus.hpp>
bool ChkEdit(int Length,char Key);
String TestIP();
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:    // IDE-managed Components
    TIdTCPClient *IdTCPClient1;
    TButton *Button1;
    TButton *Button2;
    TOpenDialog *OpenDialog1;
    TIdTCPServer *IdTCPServer1;
    TIdTCPServer *IdTCPServer2;
    TIdAntiFreeze *IdAntiFreeze1;
    TListBox *ListBox2;
    TTrayIcon *TrayIcon1;
    TMemo *Memo1;
    TMainMenu *MainMenu1;
    TMenuItem *N1;
    TMenuItem *N2;
    TMenuItem *N3;
    TMenuItem *N4;
    TMenuItem *N5;
    TMenuItem *N6;
    TMenuItem *N7;
    TMenuItem *N8;
    TMenuItem *CGCBuilder1;
    TSaveDialog *SaveDialog1;
    TMemo *Memo2;
    TListBox *ListBox1;
    TLabel *Label1;
    TButton *Button3;
    TButton *Button4;
```

```
    TIdTCPServer *IdTCPServer3;
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall IdTCPServer1Execute(TIdContext *AContext);
    void __fastcall IdTCPServer2Execute(TIdContext *AContext);
    void __fastcall N3Click(TObject *Sender);
    void __fastcall N2Click(TObject *Sender);
    void __fastcall N6Click(TObject *Sender);
    void __fastcall N8Click(TObject *Sender);
    void __fastcall CGCBuilder1Click(TObject *Sender);
    void __fastcall TrayIcon1DblClick(TObject *Sender);
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall N5Click(TObject *Sender);
    void __fastcall Button3Click(TObject *Sender);
    void __fastcall Button4Click(TObject *Sender);
    void __fastcall IdTCPServer3Execute(TIdContext *AContext);

private:    // User declarations
public:         bool hided;// User declarations
    __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
Scan.cpp
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "skan.h"
#include "main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Edit1KeyPress(TObject *Sender, char &Key)
{
```

```
if(ChkEdit(Edit1->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Edit2KeyPress(TObject *Sender, char &Key)
{
if(ChkEdit(Edit2->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Edit3KeyPress(TObject *Sender, char &Key)
{
if(ChkEdit(Edit3->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Edit4KeyPress(TObject *Sender, char &Key)
{
if(ChkEdit(Edit4->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Edit5KeyPress(TObject *Sender, char &Key)
{
if(ChkEdit(Edit5->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Edit6KeyPress(TObject *Sender, char &Key)
{
```

```
if(ChkEdit(Edit6->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------
void __fastcall TForm2::Edit7KeyPress(TObject *Sender, char &Key)
{
if(ChkEdit(Edit7->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------
void __fastcall TForm2::Edit8KeyPress(TObject *Sender, char &Key)
{
if(ChkEdit(Edit8->Text.Length(),Key))
{
Key=0;
MessageBeep(0);
}
}
//---------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
String ip=TestIP();
String tmp[4];
int count=0;
if(ip!="WinSock ERR")
{
    for(int x=0;x<ip.Length();x+=1)
    {
                if(ip.c_str()[x]!='.')
                {
                    tmp[count]=tmp[count]+ip.c_str()[x];
                }
                else
                count+=1;

    }
Edit1->Text=tmp[0];
Edit2->Text=tmp[1];
Edit3->Text=tmp[2];
```

```
Edit4->Text=tmp[3];
Edit5->Text=tmp[0];
Edit6->Text=tmp[1];
Edit7->Text=tmp[2];
Edit8->Text=tmp[3];
}
else ShowMessage("WinSock error!\n\rПроверьте настройки сети!");
}
//---------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
Edit4->Text="0";
Edit8->Text="255";
}
//---------------------------------------------------------------------

void __fastcall TForm2::Button4Click(TObject *Sender)
{
Form2->Close();
}
//---------------------------------------------------------------------


void __fastcall TForm2::Button3Click(TObject *Sender)
{
for(int x=StrToInt(Edit1->Text);x<=StrToInt(Edit5->Text);x+=1)
for(int y=StrToInt(Edit2->Text);y<=StrToInt(Edit6->Text);y+=1)
for(int z=StrToInt(Edit3->Text);z<=StrToInt(Edit7->Text);z+=1)
for(int n=StrToInt(Edit4->Text);n<=StrToInt(Edit8->Text);n+=1)
{
if(x>255 || y>255 || z>255 || n>255)
goto end;
Form1->IdTCPClient1-
>Host=IntToStr(x)+"."+IntToStr(y)+"."+IntToStr(z)+"."+IntToStr(n
Form1->IdTCPClient1->Port=2457;//Выставляем порт для файлое
try{
Form1->IdTCPClient1->Connect(); //Подключаемся
}
catch(...){}//Если не подключиличь
if(Form1->IdTCPClient1->Connected()){//Если подключились
ListBox1->Items->Add(Form1->IdTCPClient1->Host);
Form1->IdTCPClient1->Socket->WriteLn("");
Form1->IdTCPClient1->Disconnect();//Отключаемся
}
end:
```

```
}
ShowMessage("Закончили");
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button5Click(TObject *Sender)
{
Form1->ListBox1->Items->AddStrings(ListBox1->Items);
}
//---------------------------------------------------------------------------
Skan.dfm
object Form2: TForm2
 Left = 0
 Top = 0
 Caption = #1057#1082#1072#1085#1080#1088#1091#1077#1084'
'#1076#1080#1072#1087#1072#1079#1086#1085
 ClientHeight = 328
 ClientWidth = 385
 Color = clBtnFace
 Font.Charset = DEFAULT_CHARSET
 Font.Color = clWindowText
 Font.Height = -11
 Font.Name = 'Tahoma'
 Font.Style = []
 OldCreateOrder = False
 PixelsPerInch = 96
 TextHeight = 13
 object Label1: TLabel
 Left = 7
 Top = 17
 Width = 81
 Height = 13
 Caption = #1053#1072#1095#1072#1083#1100#1085#1099#1081'IP:'
 Font.Charset = DEFAULT_CHARSET
 Font.Color = clWindowText
 Font.Height = -11
 Font.Name = 'Tahoma'
 Font.Style = [fsBold]
 ParentFont = False
 end
 object Label2: TLabel
 Left = 134
 Top = 22
 Width = 3
 Height = 13
```

```
Caption = '.'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Label3: TLabel
Left = 182
Top = 22
Width = 3
Height = 13
Caption = '.'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Label4: TLabel
Left = 230
Top = 22
Width = 3
Height = 13
Caption = '.'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Label5: TLabel
Left = 8
Top = 57
Width = 73
Height = 13
Caption = #1050#1086#1085#1077#1095#1085#1099#1081'IP:'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
```

```
ParentFont = False
end
object Label6: TLabel
Left = 135
Top = 62
Width = 3
Height = 13
Caption = '.'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Label7: TLabel
Left = 183
Top = 62
Width = 3
Height = 13
Caption = '.'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Label8: TLabel
Left = 231
Top = 62
Width = 3
Height = 13
Caption = '.'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Tahoma'
Font.Style = [fsBold]
ParentFont = False
end
object Edit1: TEdit
Left = 94
Top = 14
Width = 34
```

```
Height = 21
TabOrder = 0
Text = '127'
OnKeyPress = Edit1KeyPress
end
object Edit2: TEdit
Left = 143
Top = 14
Width = 34
Height = 21
TabOrder = 1
Text = '0'
OnKeyPress = Edit2KeyPress
end
object Edit3: TEdit
Left = 191
Top = 14
Width = 34
Height = 21
TabOrder = 2
Text = '0'
OnKeyPress = Edit3KeyPress
end
object Edit4: TEdit
Left = 239
Top = 14
Width = 34
Height = 21
TabOrder = 3
Text = '1'
OnKeyPress = Edit4KeyPress
end
object Edit5: TEdit
Left = 95
Top = 54
Width = 34
Height = 21
TabOrder = 4
Text = '127'
OnKeyPress = Edit5KeyPress
end
object Edit6: TEdit
Left = 144
Top = 54
Width = 34
```

```
Height = 21
TabOrder = 5
Text = '0'
OnKeyPress = Edit6KeyPress
end
object Edit7: TEdit
Left = 192
Top = 54
Width = 34
Height = 21
TabOrder = 6
Text = '0'
OnKeyPress = Edit7KeyPress
end
object Edit8: TEdit
Left = 240
Top = 54
Width = 34
Height = 21
TabOrder = 7
Text = '1'
OnKeyPress = Edit8KeyPress
end
object Button1: TButton
Left = 279
Top = 8
Width = 97
Height = 33
Caption = '<==#1057#1074#1086#1081
TabOrder = 8
OnClick = Button1Click
end
object Button2: TButton
Left = 280
Top = 48
Width = 97
Height = 33
Caption = '<==255'
TabOrder = 9
OnClick = Button2Click
end
object Button3: TButton
Left = 8
Top = 287
Width = 119
```

```
 Height = 33
 Caption = #1053#1072#1095#1072#1090#1100'
'#1089#1082#1072#1085#1080#1088#1086#1074#1072#1085#1080#1077
 TabOrder = 10
 OnClick = Button3Click
 end
 object Button4: TButton
 Left = 256
 Top = 287
 Width = 119
 Height = 33
 Caption = #1053#1072#1079#1072#1076
 TabOrder = 11
 OnClick = Button4Click
 end
 object ProgressBar1: TProgressBar
 Left = 8
 Top = 264
 Width = 369
 Height = 17
 TabOrder = 12
 end
 object ListBox1: TListBox
 Left = 8
 Top = 87
 Width = 369
 Height = 170
 ItemHeight = 13
 TabOrder = 13
 end
 object Button5: TButton
 Left = 130
 Top = 287
 Width = 120
 Height = 33
 Caption = #1044#1086#1073#1072#1074#1080#1090#1100'
'#1086#1090#1089#1082#1072#1085#1077#1085#1099#1077
 TabOrder = 14
 OnClick = Button5Click
 end
end
Scan.h
//---------------------------------------------------------------------------

#ifndef skanH
```

```
#define skanH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:    // IDE-managed Components
    TLabel *Label1;
    TLabel *Label2;
    TLabel *Label3;
    TLabel *Label4;
    TEdit *Edit1;
    TEdit *Edit2;
    TEdit *Edit3;
    TEdit *Edit4;
    TLabel *Label5;
    TLabel *Label6;
    TLabel *Label7;
    TLabel *Label8;
    TEdit *Edit5;
    TEdit *Edit6;
    TEdit *Edit7;
    TEdit *Edit8;
    TButton *Button1;
    TButton *Button2;
    TButton *Button3;
    TButton *Button4;
    TProgressBar *ProgressBar1;
    TListBox *ListBox1;
    TButton *Button5;
    void __fastcall Edit1KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit2KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit3KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit4KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit5KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit6KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit7KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit8KeyPress(TObject *Sender, char &Key);
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall Button4Click(TObject *Sender);
```

```
    void __fastcall Button3Click(TObject *Sender);
    void __fastcall Button5Click(TObject *Sender);
private:    // User declarations
public:             // User declarations
    __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ПРЕДМЕТУ «ПРОГРАММИРОВАНИЕ на C/C++»

Рассмотрено и рекомендовано
к изданию на заседании
научно-методического
Совета ТУИТ № 9(80)
от «20» мая 2015г.

Составители: Касымова Ш.Т.
Ганиходжаева Д.З.

Ответственный редактор:
Корректор: Махкамова М.

Формат 60x84 1/16

Заказ № 96 Тираж 30