Проект «Инженерные кадры Зауралья»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра автоматизации производственных процессов

ПРОГРАММИРОВАНИЕ И АЛГОРИТМИЗАЦИЯ

Методические указания
к выполнению курсовой работы
для студентов направления 15.03.04 (220700.62)

Курган 2015

Кафедра: «Автоматизация производственных процессов»

Дисциплина: «Программирование и алгоритмизация»
(направления 15.03.04 и 220700.62).

Составил: ст. преподаватель И.В. Скобелев.

Утверждены на заседании кафедры «27» ноября 2014 г.

Рекомендованы методическим советом университета в рамках проекта «Инженерные кадры Зауралья» «20» декабря 2013 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В соответствии с учебным планом студенты специальности «Автоматизация технологических процессов и производств» всех форм обучения по дисциплине «Программирование и алгоритмизация» выполняют курсовую работу.

Курсовая работа направлена на закрепление знаний и получение навыков программирования. В процессе выполнения работы студенты получают возможность ознакомиться с основными этапами разработки программных объектов, начиная от постановки задачи и заканчивая документированием результатов.

Курсовая работа – самостоятельное научное исследование студента, завершающее изучение конкретной научной дисциплины и посвященная одной из актуальных проблем программирования. Выполнение курсовой работы предполагает отражение уровня общетеоретической специальной подготовки студента, его способности к научному творчеству, умение использовать полученные навыки в научных исследованиях по избранной специальности.

Курсовая работа выполняется либо в среде программирования Visual C++, либо в среде 1С: Предприятии, по согласованию с преподавателем.

Разработка программного продукта в среде 1С: Предприятие подробно рассмотрена в учебном пособии: «Объектно-ориентированное программирование автоматизированных систем управления на платформе 1С: Предприятие 8.2» [11].

Знания и умения, полученные в ходе курсового проектирования по дисциплине «Программирование и алгоритмизация», могут быть использованы в ходе выполнения последующих курсовых работ и проектов, а также при изучении смежных дисциплин.

1. Содержание курсовой работы

В курсовой работе студент должен продемонстрировать полученные во время обучения, знания по предмету, и навыки работы с конкретными программными продуктами, умение использовать интегрированную среду программирования, возможности предоставляемые средствами автоматизации программирования, владение приемами конфигурировании и визуального программирования и умение создавать работоспособный программный код.

Курсовая работа является самостоятельной работой студента. Не допускаются, две одинаковы курсовые работы. Курсовая работа может быть выполнена группой студентов, но в этом случае, должно быть четко определено, кто какую часть работы выполнял. Индивидуальное задание должно быть отражено в задание на проектирование (приложение 2) и в пояснительной записке.

Цель курсовой работы – приобретение практических навыков в области:

1) разработки программных продуктов (структурной и функциональной схем программного обеспечения, структур данных, алгоритмов и реализующих их программ, стратегии тестирования и тестовых данных и т.п.);
2) тестирования и отладки интерактивных систем программного обеспечения;
3) составления пояснительной записки, содержащей обоснование принятых проектных решений;
4) применения нормативных документов, регламентирующих состав, содержание и форму технической документации на разработанный программный продукт.

Курсовая работа содержит работоспособный программный продукт, состоящий из необходимого для решения поставленной задачи программных модулей, файлов данных, визуальных форм и программных объектов.

Разработанные программные продукты должны быть продемонстрированы в работе, на контрольных примерах. Проект записывается на машинный носитель и сдается вместе с пояснительной запиской.

Пояснительная записка должна содержать следующие разделы:

1. Техническое задание, т.е. конкретизация выбранной темы. В данном разделе необходимо отразить входные и выходные данные и формы их представления, предполагаемые к использованию структуры данных и файлов, математические модели решения задач.
2. Общую блок-схему алгоритма. Блок-схема, должна ясно и полно отражать алгоритм решаемой задачи.
3. Инструкцию по инсталляции программного продукта, с указанием требований к оборудованию и установленному программному обеспечению.
4. Инструкцию программисту, содержащею всю необходимую информацию о ограничениях и выявленных при тестировании ошибочных ситуациях.

Также данная инструкция содержит информацию о возможности модернизации программного продукта.

5. Инструкцию оператору, содержащую порядок действий оператора при рутинной работе с программным продуктом и в случае возникновения нештатных ситуаций.
6. Инструкцию по проверке и тестированию программного обеспечения с тестовыми данными.
7. Вывода по проделанной работе.
9. Список использованных источников.

2. Оформление пояснительной записки к курсовой работе

Пояснительная записка должна содержать обоснование основных проектных решений, принятых студентом на каждом этапе разработки. Решения должны приниматься исходя из особенностей проектируемого продукта и специфики области его применения. Не должно быть обоснований типа «удобнее», «целесообразнее» и т. п. Необходимо пояснить, чем удобнее, почему целесообразно. По возможности необходимо четко формулировать основания для принятия того или иного решения.

Пояснительная записка оформляется на листах формата А4. Графический материал можно оформлять на листах формата А3. Поля на листе определяются в соответствии с общими требованиями. При использовании текстовых редакторов для оформления записки параметры страницы заказываются в зависимости от устройства печати. При ручном оформлении выбираются из соображений удобства.

Нумерация страниц – сквозная. Номер проставляется сверху справа арабской цифрой. Страницами являются листы с текстами, рисунками и текстами приложения.

Первая страница – титульный лист расчетно-пояснительной записки. Номер страницы на титульном листе не проставляется. Образец титульного листа представлен в Приложении 1.

Вторая страница – задание на проектирование Приложение 2 .

Третья страница – оглавление, отражающее содержание изложенного материала. Затем следуют разделы записки в порядке, определенном логикой изложения материала.

Записка завершается списком литературы.

Далее могут следовать приложения, содержащие материал, не вошедший в записку по причине ее ограниченного размера, но интересный для более

глубокого понимания назначения и возможностей разработки. Расчетно-пояснительная записка может содержать одно и более приложений.

Наименование разделов и подразделов пишутся строчными буквами, кроме первой прописной. Расстояние между заголовками и текстом, а также между заголовками раздела и подразделов должно быть равно:

- при выполнении документа машинописным способом – двум интервалам;
- при выполнении рукописным способом –10 мм;
- при использовании текстовых редакторов – определяется возможностями редактора.

Расстояние между последней строкой текста предыдущего раздела и последующим заголовком при расположении их на одной странице должно быть равно:

- при выполнении документа машинописным способом – трем интервалам;
- при выполнении рукописным способом – не менее 15 мм;
- при использовании текстовых редакторов – определяется возможностями редактора.

Разделы и подразделы нумеруются арабскими цифрами с точкой. Разделы должны иметь порядковые номера 1, 2, и т. д. Номер подраздела включает номер раздела и порядковый номер подраздела, входящего в данный раздел, разделенные точкой. Например: 2.1., 3.5.

Перечисления надо нумеровать арабскими цифрами со скобкой; Например: 2), 3) и т. д. – с абзацного отступа. Допускается выделять перечисление простановкой дефиса перед пунктом текста или символом, его заменяющим, в текстовых редакторах. Иллюстрации (графики, схемы, диаграммы) могут быть приведены как в основном тексте, так и в приложении. Все иллюстрации именуются рисунками. Все рисунки, таблицы и формулы нумеруются арабскими цифрами последовательно (сквозная нумерация). В приложении – в пределах приложения. Иллюстрации могут быть в компьютерном исполнении, в том числе и цветные. Чертежи, графики, диаграммы, схемы должны соответствовать требованиям ЕСКД.

Рисунки, за исключением рисунков приложений следует нумеровать арабскими цифрами сквозной нумерацией.

Допускается нумеровать рисунки в пределах раздела. В этом случае номер рисунка состоит из номера раздела и порядкового номера рисунка, разделенных точкой.

Если рисунок один, то он обозначается «Рисунок 1». Рисунок может иметь наименование и пояснительные данные (подрисуночный текст). Слово «рисунок» и его наименование располагают посередине строки. Если есть

подрисуночный текст, то слово «рисунок» и его наименование помещают после пояснительных данных.

Например: Рисунок 12 - Форма окна основного меню

На все рисунки, таблицы и формулы в записке должны быть ссылки в виде: «(рисунок 12)» или « форма окна основного меню приведена на рисунке 12».

Слово «рисунок» и его наименование располагается посредине строки.

Рисунки и таблицы должны размещаться сразу после той страницы, на которой, в тексте записки, она упоминается в первый раз. Если позволяет место, рисунок (таблица) может размещаться в тексте на той же странице, где на него дается первая ссылка.

Если рисунок занимает более одной страницы, на всех страницах, кроме первой, проставляется номер рисунка и слово «Продолжение». Например:

Рисунок 12 - Продолжение

Рисунки следует размещать так, чтобы их можно было рассматривать без поворота записки. Если такое размещение невозможно, рисунки следует располагать так, чтобы для рассматривания надо было повернуть записку по часовой стрелке. В этом случае верхним краем является левый край страницы. Расположение и размеры полей сохраняются в соответствии с установленными.

Схемы алгоритмов должны быть выполнены в соответствии со стандартом ЕСПД. Толщина сплошной линии при вычерчивании схем алгоритмов должна быть в пределах от 0,6 до 1,5 мм. Надписи на схемах должны быть выполнены чертежным шрифтом. Высота букв и цифр должна быть менее 3,5 мм.

Название таблицы, при его наличии, должно отражать ее содержание, быть точным, кратким. Название следует помещать над таблицей слева, без абзацного отступа в одну строку с ее номером через тире. Заголовок, кроме первой буквы, выполняется строчными буквами. В аббревиатурах используются только заглавные буквы. Например: ПЭВМ.

При переносе части таблицы название помещается только над первой частью таблицы, нижнюю горизонтальную черту, ограничивающую таблицу, не проводят.

Ссылки на таблицы в тексте пояснительной записки должны быть в виде слова «таблица» с указанием ее номера. Например, «Результаты тестов приведены в таблице 4».

Таблицу с большим количеством строк допускается переносить на другой лист (страницу). При переносе слово «Таблица» и ее номер указывается один раз справа над первой частью таблицы, а над другими частями пишут слово «Продолжение» и указывают номер таблицы, например: «Продолжение таблицы 1». При этом заголовок помещают только над ее первой частью.

Оформление таблиц должно соответствовать ГОСТ 1.5 и ГОСТ 2.105.

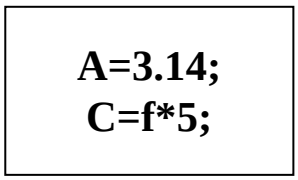
Тексты программ должны оформляться в соответствии с «хорошим стилем» программирования, т.е. должны быть легко читаемы и хорошо документированы. В текстах должны быть комментарии:

- после заголовка программы или подпрограммы приводится общая информация: назначение, входные данные, результаты, метод решения; данные о программисте, дата написания, версия;
- при объявлении данных - назначение переменных;
- в начале и в конце определенной функционально законченной части программы;
- для пояснения логических частей программы (ветвлений, циклов).

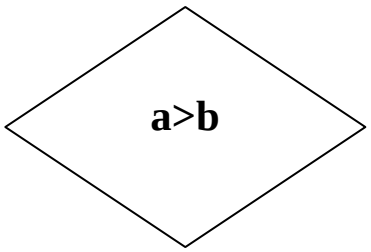
Однако комментарии не должны затенять структуру текста и должны быть ясными и краткими. Наименование программ и подпрограмм должны отражать их назначение. Логическая структура программы должна быть отражена в ее тексте с помощью:

- пустых строк между текстами подпрограмм и отдельных ее функционально законченных частей;
- сдвигами текста в строке при написании:
- заголовков вложенных циклов;
- тела цикла после его заголовка;
- альтернатив разветвлений процесса обработки данных.

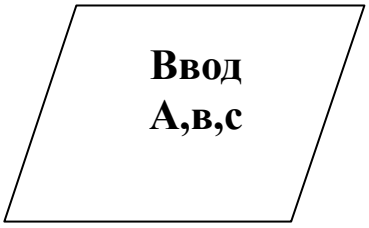
Правила составления блок-схем определяются ГОСТ 19.701-90 (ИСО 5807-85)
Наиболее часто используемые символы:
Процесс – используется для обозначения последовательности вычислительных действий.



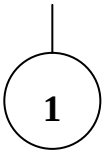
Решение – для проверки условий



Ввод –вывод - обозначает ввод вывод в общем виде:



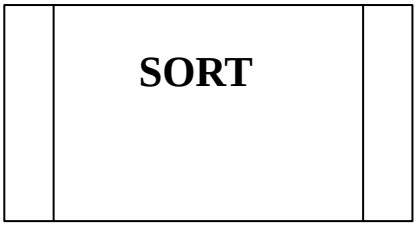
Соединитель :



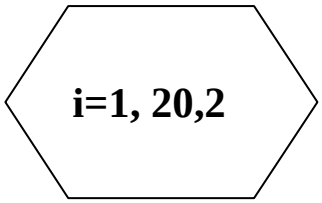
Пуск, останов:

Печать:

Предопределенный процесс или подпрограмма:



Модификация или начало цикла:



Список литературы должен включать все использованные источники. Сведения о книгах (монографиях, учебниках, пособиях, справочниках и т.д.) должны содержать: фамилию и инициалы автора, заглавие книги, место издания, издательство, год издания. При наличии трех и более авторов допускается указывать фамилию и инициалы только первого из них со словами «и др.». Наименование места издания надо приводить полностью в именительном падеже: допускается сокращение названия только двух городов: Москва (М.) и Ленинград (Л.).

Сведения о статье из периодического издания должны включать:
фамилию и инициалы автора, наименование статьи, наименование издания (журнала), наименование серии (если она есть), год выпуска, том (если есть), номер издания (журнала) и номера страниц, на которых помещена статья. При ссылке на источник из списка литературы (особенно при обзоре аналогов) надо указывать порядковый номер по списку литературы, заключенный в квадратные скобки; например: [5].

3.Порядок выполнения курсовой работы

1. Выполнение курсовой работы начинается с выбора темы и утверждения задания. Задание считается утвержденным, если лист приложения 2 подписан руководителем.

2. После утверждения задания студент разрабатывает общую блок-схему программы и техническое задание, в котором конкретизируются входные и выходные формы, структуры данных, файлов и требования к работе программного продукта.

3. Разрабатывается собственно программный продукт, в виде программного кода. Проходят проверку отдельные части программы, корректируется блок-схема и при необходимости отдельные пункты технического задания.

4. Разработанная программа проходит всестороннюю проверку и тестирование, составляются тестовые задания, по результатам проверки вносятся коррективы в программный код.

5. Оформляется расчетно-пояснительная записка со всей необходимой для сопровождения программного продукта документацией.

6. Программный код и записка предоставляется к защите. Студент демонстрирует работоспособность программного кода, с различными сочетаниями исходных данных.

7. Работа либо признается защищенной и выставляется оценка, либо признается нуждающейся в доработке и соответственно перерабатывается.

4. **Защита курсовой работы.**

На защиту студент предоставляет:

Техническое задание.

Программный продукт.

Расчетно-пояснительную записку на 20-30 страницах в рукописном или отпечатанном виде (шрифт 12 через два интервала), содержащую описание разработки и соответствующие иллюстрации.

Программную документацию, указанную в разделе «Требования к программной документации» технического задания.

Программный продукт студент предварительно демонстрирует и сдает преподавателю, который дает отзыв на работу и допускает студента к защите.

В процессе демонстрации программного продукта проверяется:

- соответствие программы техническому заданию;
- работоспособность в различных режимах.

Защита курсовой работы выполняется комиссии, состоящей не менее чем из двух преподавателей.

На защите студент коротко (3–5 мин.) докладывает об основных проектных решениях, принятых в процессе разработки, и отвечает на вопросы членов комиссии.

Оценка за курсовую работу выставляется с учетом:

- качества выполненного программного продукта,
- правильности оформления записки;
- результатов защиты.

5. Примерный перечень тем курсовой работы.

1. Решение задачи сбора первичной информации о деятельности предприятия, на основе счетов-фактур.
2. Создание базы данных контрагентов предприятия.
3. Создание базы данных о сотрудниках предприятия.
4. Исследование задачи транспортно-складской операции в автоматизированном производстве.
5. Решение задачи оптимизации технологических процессов в автоматизированном машиностроительном производстве.
6. Нахождение оптимального решения в реальном производственном процессе.
7. Решение задачи анализа деятельности предприятия.
8. Автоматизация анализа текстовых документов .
9. Задача организации сбора, хранения и поиска информации на предприятии.

6. Пример разработки программы на языке VC++.

Рассмотрим пример разработки простейшего клиент-серверного приложения. Приложение будет состоять из двух программ:

1. Клиента, который устанавливает связь с сервером и передает ему некоторое сообщение.
2. Сервера, с которым могут устанавливать связь несколько клиентов, от которых он принимает информацию и каждому может передать сообщение или передать сообщения всем клиентам.

Для реализации той части программы, которая, собственно, и отвечает за все взаимодействие с сетью, нам понадобится три класса - CClientSocket, CListeningSocket и CPool. Первый создается для каждого нового клиента, подсоединяющегося к серверу. CListeningSocket, как видно из названия, - "слушающий" сокет. Его основная и единственная задача - отвечать на запросы о подключении новых клиентов. CPool хранит в себе все создаваемые сокеты, управляет ими и осуществляет связь с остальной частью программы.

Начнем с создание клиента, как наиболее простой задачи.

Создаем приложение на основе диалогового окна. Это известная операция, только следует включить поддержку sockets. На диалоговом окне предусмотреть поля ввода типа Edit box для имени клиента, IP адреса, и текста передаваемого серверу сообщения. Предусмотреть еще одно поле типа EditBox для отображения ответа от сервера и сделать его много строчным. Предусмотреть на форме две кнопки: одна для установки связи с сервером, другая для передачи ему сообщений. Это тоже все уже давно известно.
Поле для передачи сообщения серверу, я для разнообразия обозвал IDC_MESSAGE.

В свойствах поля IDC_EDIT4 (для приема сообщения) чтобы оно было многострочным я установил галочки напротив пунктов Multiline и Vertical scroll.
Дальше добавил в класс диалогового окна после слова public строчку

CButton *m_cConnect;

Для того, чтобы иметь возможность управлять кнопкой **Connect**
Дальше в метод инициализации диалога добавляю следующие строчки

```
BOOL CMySCk1Dlg::OnInitDialog()
{
        CDialog::OnInitDialog();
   m_cConnect=(CButton*)GetDlgItem(IDC_BUTTON1);
        SetDlgItemText(IDC_EDIT4,"Приложение загружено");
```

Первая строчка привязывает указатель **m_cConnect** к кнопке **Connect,** вторая выводит в многострочное поле сообщение, что программа клиента запущена.
Дольше добавляю обычным образом, функцию в класс диалогового окна.
Называю функцию **AddText(LPCTSTR lpszString),** функция будет добавлять строчку **lpszString** в многострочное поле **IDC_EDIT4.**
Текст функции.

```
void CMySCk1Dlg::AddText(LPCTSTR lpszString)
{
CEdit *pEdit=(CEdit*)GetDlgItem(IDC_EDIT4);
int len=pEdit->GetWindowTextLength();
pEdit->SetSel(len,len);
pEdit->SendMessage(WM_CHAR,WPARAM(13),LPARAM(28));
pEdit->ReplaceSel(lpszString);

}
```

Полям **IDC_EDIT1** и **IDC_EDIT1** с помощью ClassWizard присваиваю переменные

```
CString     Name;
CString     ip;
```

Начальные значения полям придаю в конструкторе

```
Name="Client1";
ip="127.0.0.1";
```

чтобы они в окошках высвечивались при запуске.
Дальше приступаю к созданию нового класса и использую для этого ClassWizard. Нажимаю на кнопку добавить класс, набираю имя например **MySocket** выбираю в качестве базового класса **CSocket.** После создания он появится во вкладке классов.
Теперь добавляем в него указатель на класс диалогового окна **m_pDlg** следующим образом:

```
class MySocket : public CSocket
{
// Attributes
```

```
public:

// Operations
public:
        MySocket();
        MySocket(CMySCk1Dlg *pDlg)
        {m_pDlg=pDlg;};
        virtual ~MySocket();
CMySCk1Dlg *m_pDlg;
// Overrides
```

Добавляем в этот класс еще две **виртуальные** (не забыть установить соответствующий флажок) функции.

virtual void OnReceive(int nErrorCode); Прием сообщения от сервера

virtual void OnClose(int nErrorCode); Сообщение об разрыве связи.

В функции добавляем код.

```
void MySocket::OnClose(int nErrorCode)
{
m_pDlg->m_cConnect->EnableWindow(1);
AfxMessageBox("Connection left");
delete this;
CSocket::OnClose(nErrorCode);

}

void MySocket::OnReceive(int nErrorCode)
{
char st[4096];
int r=Receive(st,4096);
CString s;
st[r]='\0';
m_pDlg->AddText(st);
CSocket::OnReceive(nErrorCode);
}
```

Все с классом **CSocket** покончено, теперь доделываем диалоговое окно

Добавляем в файл **MySCk1Dlg.h** :

```
// CMySCk1Dlg dialog
class MySocket;
class CMySCk1Dlg : public CDialog
{
// Construction
public:
        void AddText(LPCTSTR lpszString);
        CMySCk1Dlg(CWnd* pParent = NULL);   // standard constructor
  // void AddText(LPCTSTR lpszString);
```

```
        CButton *m_cConnect;
        MySocket *pSocket;   - указатель на класс
        CString m_sMessage;  - строка сообщения
// Dialog Data
        //{{AFX_DATA(CMySCk1Dlg)
        enum { IDD = IDD_MYSCK1_DIALOG };
        CString       Name;
        CString       ip;
```

В файл **MySCk1Dlg.cpp** добавляем

```
#include "MySocket.h"
```

Делаем метод для кнопки **Connect**

```
void CMySCk1Dlg::OnButton1()
{
        pSocket=new MySocket(this);
        pSocket->Create();
        GetDlgItemText(IDC_EDIT2,ip);
        if (pSocket->Connect(ip,2049))
        {
        m_cConnect->EnableWindow(0);
        GetDlgItemText(IDC_EDIT1,Name);
   pSocket->Send(Name,Name.GetLength());
        }
        else
                delete pSocket;
}
```

Дальше делаем метод для кнопки **Cend**

```
void CMySCk1Dlg::OnButton2()
{
        GetDlgItemText(IDC_MESSAGE,m_sMessage);
        if(pSocket!=NULL&&m_sMessage.GetLength()!=0)
                pSocket->Send(m_sMessage,80);
}
```

Все можно транслировать и запускать. Хотя транслировать и проверять следует по мере создания кода, чтобы легче локализовать ошибки.

Создаем сервер.

Создаем приложение на основе диалогового окна. Это известная операция, только следует включить поддержку sockets. На диалоговом окне предусмотреть два поля типа List Box: одно для отображения соединений с клиентами, второе для сообщений получаемых от клиентов. Дальше предусмотреть поле типа Edit box для сообщения передаваемого от сервера клиентам, и предусмотреть две кнопки одна отвечает за передачу выбранному клиенту, вторая за передачу всем клиентам. Это тоже все уже давно известно и

трудностей не должно вызывать. С полями List Box связываем переменные m_l1 и m_l2. типа control с полем Edit box переменную m_msg. типа CString.

```
CListBox  m_l2;
CListBox  m_l1;
CString   m_msg;
```

Дальше переходим к созданию классов.
Для этого используем ClassWizard. Нажимаю на кнопку добавить класс, набираю имя например **CClientSocket** выбираю в качестве базового класса **CSocket.** После создания он появится во вкладке классов.
Теперь добавляем в него (файл ClientSocket.h)
Объявление класса

```
class CPool;
/////////////////////////////////////////////////////////////////////////////
// CClientSocket command target

class CClientSocket : public CSocket
```

переменную
CString Name; - имя слиента
И указатель на класс CPool;

```
protected:
    CPool* pPool;
```

для дальнейшей связи с ним.
А заодно изменяю конструктор

```
public:
    CClientSocket(CPool* ppool);
    virtual ~CClientSocket();
```

И вставляю
#include "Pool.h" в файл ClientSocket.cpp
Затем доделываю конструктор в этом файле.

```
CClientSocket::CClientSocket(CPool *ppool)
{
    pPool=ppool;
}
```

Добавляем в этот класс еще две **виртуальные** (не забыть установить соответствующий флажок) функции.
virtual void OnReceive(int nErrorCode); Прием сообщения от сервера
virtual void OnClose(int nErrorCode); Сообщение об разрыве связи.

В функции добавляем код.

```
void MySocket::OnClose(int nErrorCode)
{
pPool->ProcessClose(this);

    CSocket::OnClose(nErrorCode));
```

```
}
void CClientSocket::OnReceive(int nErrorCode)
{
        char buff[4096];
         CString message;
        int nRead;
        nRead=Receive(buff,4096);
        switch (nRead)
        {case 0: Close();
                break;
        case SOCKET_ERROR: AfxMessageBox("Error occurred");
                        Close();
                break;
   default: buff[nRead]='\0';
                 message=buff;
                    if (Name.GetLength()==0)
                    {       Name=buff;
                    Send("Connection Ok!",14);}
                    pPool->ProcessRead(this,message);

        }
        CSocket::OnReceive(nErrorCode);
}
```

Аналогично создаем класс CListeninngSocket.
Теперь добавляем в него (файл ListeninngSocket.h)
Объявление класса

```
class CPool;
//////////////////////////////////////////////////////////////////////////
// CClientSocket command target

class CClientSocket : public CSocket
```

И указатель на класс CPool;
protected:
CPool* pPool; для дальнейшей связи с ним.
А заодно изменяю конструктор
public:

```
        CListeninngSocket (CPool* ppool);
        virtual ~ CListeninngSocket ();
```

И вставляю
#include "Pool.h" в файл ListeninngSocket.cpp
Затем доделываю конструктор в этом файле.

```
CClientSocket::CClientSocket(CPool *ppool)
{
```

pPool=ppool;
}
Добавляем в этот класс одну **виртуальную** (не забыть установить соответствующий флажок) функцию
virtual void OnAccept(int nErrorCode);
и добавляю код
void CListeninngSocket::OnAccept(int nErrorCode)
{
pPool->ProcessAccept();
CSocket::OnAccept(nErrorCode);
}
Функция вызывается при соединении с клиентом и добавляет очередного приконектившегося клиента в список.
Теперь создаем собственно класс class **CPool**
Для этого используем ClassWizard. Нажимаю на кнопку добавить класс, набираю имя например **CPool** , обойдемся без базового класса.
По тому он получился совсем пустой но это не беда, это чисто наш самостийный класс. Все наполнение добавляем в ручную.
class CMyServ1Dlg;
class CListeninngSocket; - с этими классами взаимодействуем
class CClientSocket;

class CPool
{
public:
- все эти функцуии создаем с помощь добавить функцию

BOOL Send(int index,CString data);
void ProcessRead(CClientSocket *pSocket,CString msg);
int ProcessClose(CClientSocket *pSocket);
int ProcessAccept();
BOOL Send2All(CString data);
BOOL Init(void *phost,int port);
CPool();

virtual ~CPool();
CPtrList connectionList; - список приконектившихся слиентов или сокетов

protected:
CListeninngSocket *m_pSocket; - указатель на слушающий класс
CMyServ1Dlg *pHost; - указатель на диалоговое окно
};
В файл Pool.cpp

Добавляем:

```
#include "MyServ1Dlg.h"
#include "ListeninngSocket.h"
#include "ClientSocket.h"
```

Изменяем конструктор и деструктор

```
CPool::CPool()
{m_pSocket=NULL;}

CPool::~CPool()
{
  while(!connectionList.IsEmpty())
  { CClientSocket *pSocket =(CClientSocket *)connectionList.RemoveHead();
    if (pSocket!=NULL)
        {pSocket->Close();
         delete pSocket;
        }
  }
m_pSocket->Close();
delete m_pSocket;
}
```

Т.е. удаляем все созданные сокеты.

Создаем код для обработки различных действий с сокетами

Инициализация соединения:

```
BOOL CPool::Init(void *phost, int port)
{       pHost=(CMyServ1Dlg*)phost;

m_pSocket=new CListeninngSocket(this);
  if (m_pSocket->Create(port))
  { if(m_pSocket->Listen())
      return TRUE;
  }
return FALSE;
}
```

Передача сообщения всем клиентам:

```
BOOL CPool::Send2All(CString data)
{ if (data.GetLength()==0)
    return FALSE;
for (POSITION pos=connectionList.GetHeadPosition(); pos!=NULL;)
{
       CClientSocket *pSock=(CClientSocket *)connectionList.GetNext(pos);
 pSock->Send(data,data.GetLength());
}
return TRUE;
}
```

Добавление нового соединения в список:

```
int CPool::ProcessAccept()
{
CClientSocket *pSocket =new CClientSocket(this);
  if (m_pSocket->Accept(*pSocket))
  { connectionList.AddTail(pSocket);

  }
  else
        delete pSocket;
   pHost->UpdateWiew();
  return 0;
}
```

Удаление соединения из списка:

```
int CPool::ProcessClose(CClientSocket *pSocket)
{
 POSITION pos=connectionList.Find(pSocket);
 connectionList.RemoveAt(pos);
 delete pSocket;
 pHost->UpdateWiew();
 return 0;
}
```

Чтение сообщения от клиента

```
void CPool::ProcessRead(CClientSocket *pSocket, CString msg)
{
pHost->UpdateWiew();
pHost->m_l2.AddString(pSocket->Name+": "+msg);
pHost->UpdateData(0);
}
```

Передача сообщения одному выбранному в списке клиенту:

```
BOOL CPool::Send(int index, CString data)
{
       if (data.GetLength()==0)
    return FALSE;
CClientSocket *pSock;
 POSITION pos=connectionList.FindIndex(index);
 pSock=(CClientSocket *)connectionList.GetAt(pos);
pSock->Send(data,data.GetLength());
return TRUE;
}
```

Все осталось доделать класс диалогового окна

Добавляем в файл. MyServ1Dlg.h

```
#include "Pool.h"
```

class CMyServ1Dlg : public CDialog
{
// Construction
public:
void UpdateWiew(); функция для отображения действий
CMyServ1Dlg(CWnd* pParent = NULL); // standard constructor
CPool pool; - класс.
Добавляем в файл. MyServ1Dlg.ccp
добавляем
#include "Pool.h"
#include "ClientSocket.h"
// CAboutDlg dialog used for App About
class CPool;
class CAboutDlg : public CDialog

в методе инициализации окна:
BOOL CMyServ1Dlg::OnInitDialog()
{
CDialog::OnInitDialog();
int port;
port=2049;
pool.Init(this,port); инициализация при открытии. По порту 2049

Создаем код для функции
```
void CMyServ1Dlg::UpdateWiew()
{
CString ip,name;
 unsigned int port=2049;
  CClientSocket *pSock;
  m_l1.ResetContent();
POSITION pos=pool.connectionList.GetHeadPosition();
for (int i=0; i<pool.connectionList.GetCount();i++)
        {  pSock=(CClientSocket   *)pool.connectionList.GetNext(pos);//pSocket-
>Name
           pSock->GetPeerName(ip,port);
              name=CString(pSock->Name+" ip:"+ip);
             m_l1.AddString(name);
             UpdateData(0);
}
}
```
Создаем метод для кнопки передать клиенту
```
void CMyServ1Dlg::OnButton1()
{ UpdateData(1);
int indx=m_l1.GetCurSel();
pool.Send(indx,m_msg);       }
```

И метод для передачи всем клиентам

```
void CMyServ1Dlg::OnButton2()
{
        UpdateData(1);
pool.Send2All(m_msg);
}
```

Все теперь должно работать, примерно, так как представлено на рисунке 6.1. Здесь мы видим работу сервера с тремя клиентами.

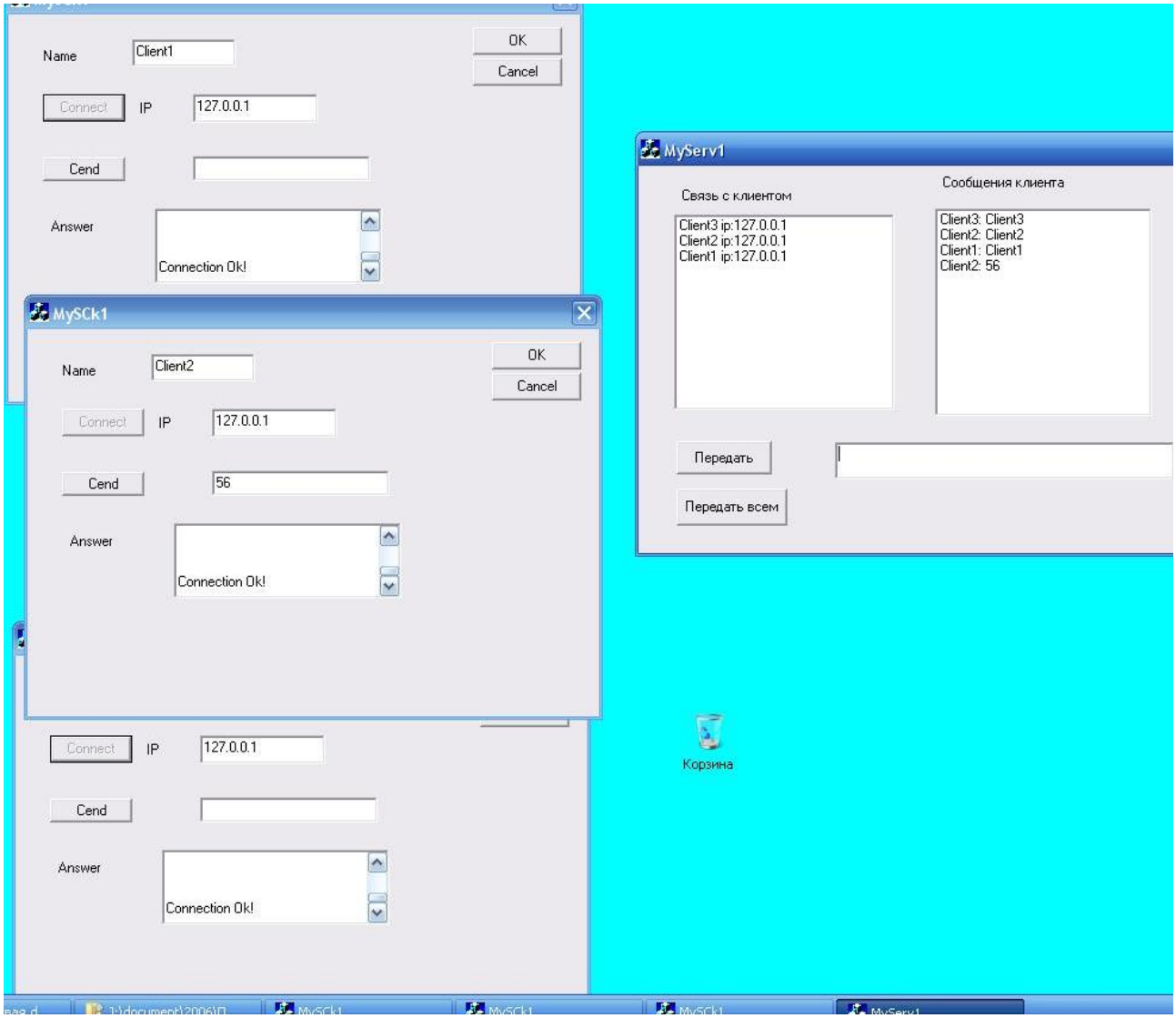


Рис. 6.1. Пример работы сервера с несколькими клиентами на одном компьютере.

Пример графического изображения работы программы – клиента приведен на рисунке 6.2. в виде укрупненной блок-схемы.

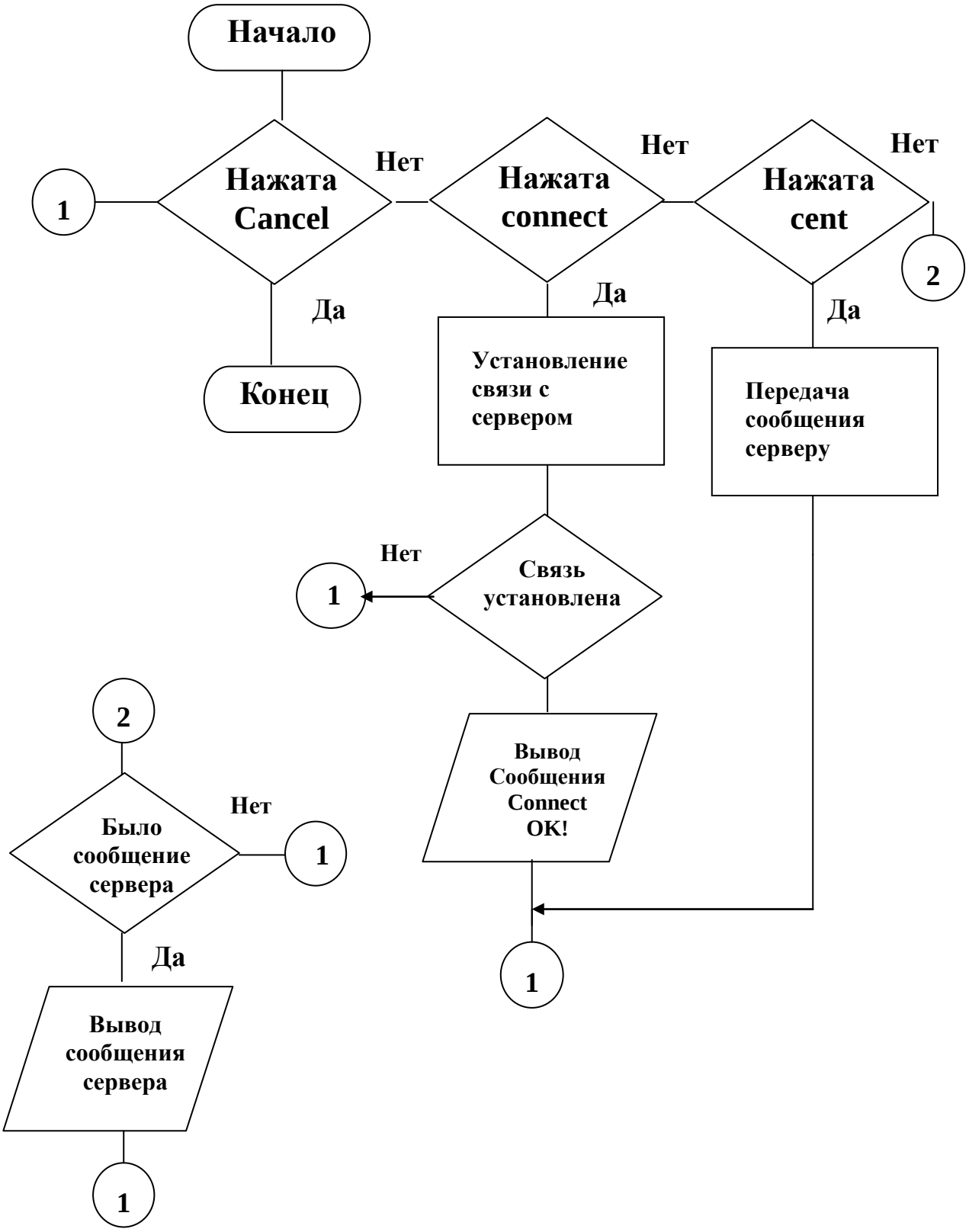


Рис 6.2. Укрупненная блок-схема работы клиента.

Список рекомендуемых источников

1.Давыдов В.Г. Программирование и основы алгоритмизации: Учебное пособие. - М.: Высшая школа , 2003.

2. Холзнер С. Microsoft Visual C++6 с самого начала. – СПб.: Питер, 2005.

3. Либерти Джесс. Освой самостоятельно C++ за 21 день.- М.: Издательский Дом «Вильяме», 2001. - 814 с.

4.Павловская Т.А. C/C++: Программирование на языке высокого уровня: Учебник. - СПб.: Питер,2001. - 460 с.

5. Профессиональная разработка в среде 1С Предприятие 8. / под ред. Радченко М.Г. - « 1С-Паблибшинг», СПб.: Питер, 2006.

6. http://v8.1c.ru/predpriyatie/QuestionsPlatform.htm?printversion=1

7. Рыбалко В.В. HELLO-1C. Пример быстрой разработки приложений на платформе 1С:Предприятие 8.2. М.: ООО « 1С-Паблибшинг», 2009.

8. Радченко М.Г. 1С:Предприятие 8.2 практическое пособие разработчика. Примеры и типовые решения / М.Г. Радченко, Е.Ю. Хрусталева - М.: ООО « 1С-Паблибшинг», 2009.

9. 1С:Предприятие 8.2 Руководство разработчика. Ч.1, Ч.2. Москва. Фирма «1С». 2009.

10. http://1c-uroki.ru/

11. Скобелев И.В., Грибанов К.Н., Фалев В.В. Объектно-ориентированное программирование автоматизированных систем управления на платформе 1С: Предприятие 8.2. Учеб. пособие. - Курган: Издательство Курганского гос. ун-та, 2013.- 107 с.

12. Иванова Г. С., Ничушкина Т. Н., Пугачев Е.К. Методические указания по выполнению курсовой работы по курсу «Технология программирования». Московский государственный технический университет им. Н.Э. Баумана. М 2003.

13. http://www.cetiforum.ru

14. http://www.microsoft.ru

15. http:// www.firststeps.ru

16. http://www.realcoding.net.

Приложение 1 Титульный лист.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра автоматизации производственных процессов

Тема курсовой работы

Курсовая работа

Расчетно-пояснительная записка.
Дисциплина: Программирование и основы алгоритмизации

Студент _____
Группа _____

Руководитель _____

Комиссия_____

Оценка_____

Дата защиты_____

Курган 2014

Приложение 2

Курганский государственный университет **Факультет *Технологический***
Кафедра *АПП*

Задание № _____

Студент группы_____ Специальность_____
Фамилия_____
Имя_____
Отчество_____

Руководитель курсового проектирования_____
Сроки проектирования _____
1.Тема курсового проекта:_____
_____
2. Содержание проекта:_____
_____
_____
_____
3.Особые дополнительные сведения:_____
_____
_____

4. План выполнения курсового проекта

№ п/п	Наименование элементов проектной работы	Сроки	Примечание	Отметка о выполнении

УТВЕРЖДАЮ:

Руководитель_____/_____/

Зав. кафедрой_____/_____/

Скобелев Игорь Вадимович

ПРОГРАММИРОВАНИЕ И АЛГОРИТМИЗАЦИЯ

Методические указания
к выполнению курсовой работы
для студентов направления 15.03.04 (220700.62)

Авторская редакция

Подписано в печать 23.03.15	Формат 60x84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ.л. 1,75	Уч.-изд.л. 1,75
Заказ 65	Тираж 25	Не для продажи

РИЦ Курганского государственного университета.
640000, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.