МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра автоматизации производственных процессов

ИЗУЧЕНИЕ ОСНОВ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ VC++ В СРЕДЕ WINDOWS

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к комплексу практических работ «Технология программирования»
для студентов направлений подготовки
220700.62 «Автоматизация технологических процессов и производств» и
220400.62 «Управление в технических системах»

Курган 2013

Кафедра автоматизации производственных процессов

Дисциплина: « Технология программирования»

Составил: ст. преподаватель И.В. Скобелев

Утверждены на заседании кафедры «18» сентября 2013 г.

Рекомендованы методическим советом университета «23» сентября 2013 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время программирование трансформировалось в целую индустрию производства программных изделий. Поэтому уже мало знать только язык программирования и операционный подход к составлению алгоритмов. Профессиональный разработчик программных изделий должен владеть теорией проектирования, методами активизации мышления. Ему необходимо умение оперирования моделями, методами генерации решений и выбора их оптимальных
вариантов. Создание программных изделий коллективом разработчиков предопределило необходимость умения планирования работ и их распределения между отдельными участниками проекта.
На основе анализа математических моделей и алгоритмов решения экономических и других задач, программист обязан:
— разрабатывать программы, обеспечивающие возможность выполнения алгоритма и соответственно поставленной задачи средствами вычислительной техники; проводить их тестирование и отладку;
— разрабатывать технологию решения задачи по всем этапам обработки информации;
— осуществлять выбор языка программирования для описания алгоритмов и структур данных;
— определять информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля;
— выполнять работу по подготовке программ к отладке и проводить отладку;
— определять объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению;
— осуществлять запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач;
— проводить корректировку разработанной программы на основе анализа выходных данных;
— разрабатывать инструкции по работе с программами, оформлять необходимую техническую документацию;
— определять возможность использования готовых программных продуктов;
— осуществлять сопровождение внедренных программ и программных средств;
— разрабатывать и внедрять системы автоматической проверки правильности программ, типовые и стандартные программные средства;
— составлять технологию обработки информации;

Данные методические указания призваны помочь студентам получить необходимые практические навыки для самостоятельной работы на языке программирования.

Практическая работа №1
Реализация технологии визуального программирования в среде Microsoft Visual

Продолжительность 2 часа.

Цель работы: Научиться использовать средства визуального программирования Microsoft Visual Studio для разработки диалоговых форм.

Выполнение работы:

Разрабатываем форму вида:

Делаем методы к полям EDIT Для автоматического вычисления значений в вычисляемых полях.

Улучшаем работу диалогового окна, вводим поля со списками для выбора заранее известных параметров.

Контрольные вопросы:

1. Для чего нужна «технология программирования»?.
2. Какие этапы развития прошла дисциплина «технология программирования»?
3. Для чего используется визуальное программирование и с чем связано его появление?
4. Какие системы визуального программирования Вы знаете?
5. Как осуществить выбор из списка.
6. Как привязать программный код к элементам дизайна?
7. Для чего служит ClassWizard?

Практическая работа №2

Исследование возможностей структурного и модульного программирования на базе языка VC++.

Продолжительность 2 часа.

Цель работы: Научиться разбивать программы на отдельные модули и осуществлять связь между модулями в работающей программе.

Выполнение работы:

Создаем методы для сохранения данных в файлах и модули для последующего извлечения этих данных из файлов и их редактированию.

Контрольные вопросы:

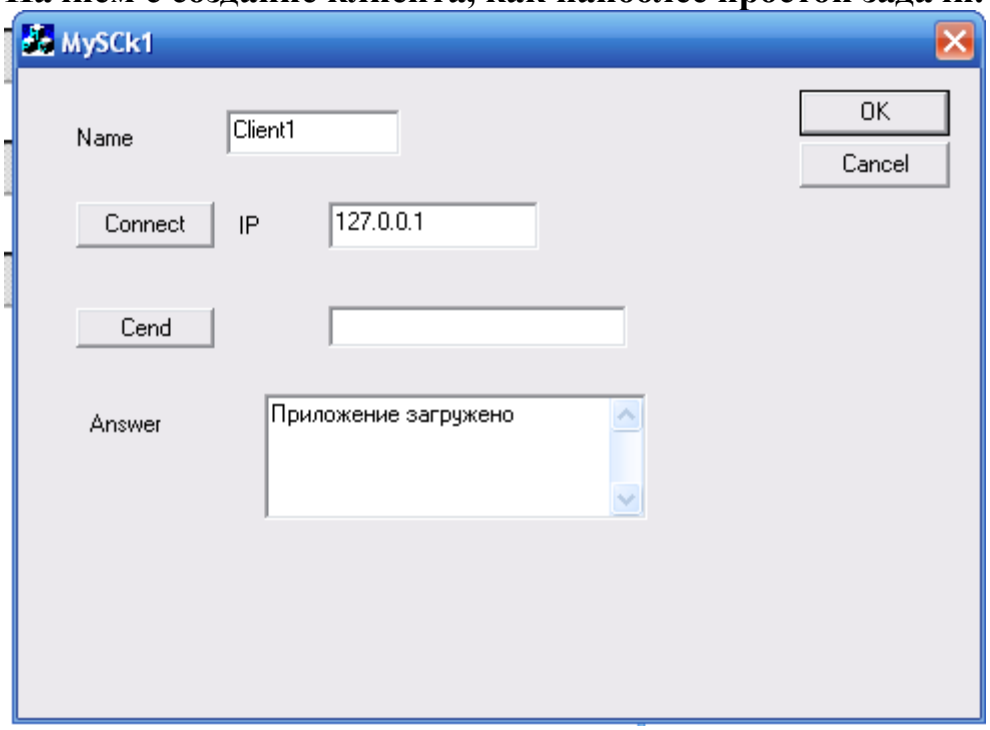
1 Для чего нужно модульное программирование?
2 Какие преимущества и недостатки, по вашему мнению, имеет модульное программирование?
3 Когда появилась необходимость в модульном программировании.?
4 Понятие структурного программирования, в чем особенность структурного подхода?
5 Как осуществляется конструирование графического интерфейса?
6 Как связать переменную с элементом графического интерфейса?
7 Как добавить метод, обрабатывающий реакцию на события, связанные с элементом графического интерфейса?

Практическая работа №3
Решение задач программирования с использованием технологии Объектно-ориентированного метода.

Продолжительность 4 часа.

Цель работы: Изучить основы объектно-ориентированного программирования на примере создания – клиент- серверного приложения.

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.

В результате мы должны создать клиентскую программу способную по сети передавать данные серверной программе.

Контрольные вопросы:

1 В чем смысл объектно-ориентрованного подхода в программирование?
2 Что такое объект и что такое класс?
3 Что значит наследование?
4 Как создаются классы?.
5 Как создаются объекты.
6 Как добавить класс, работающий с Socket?

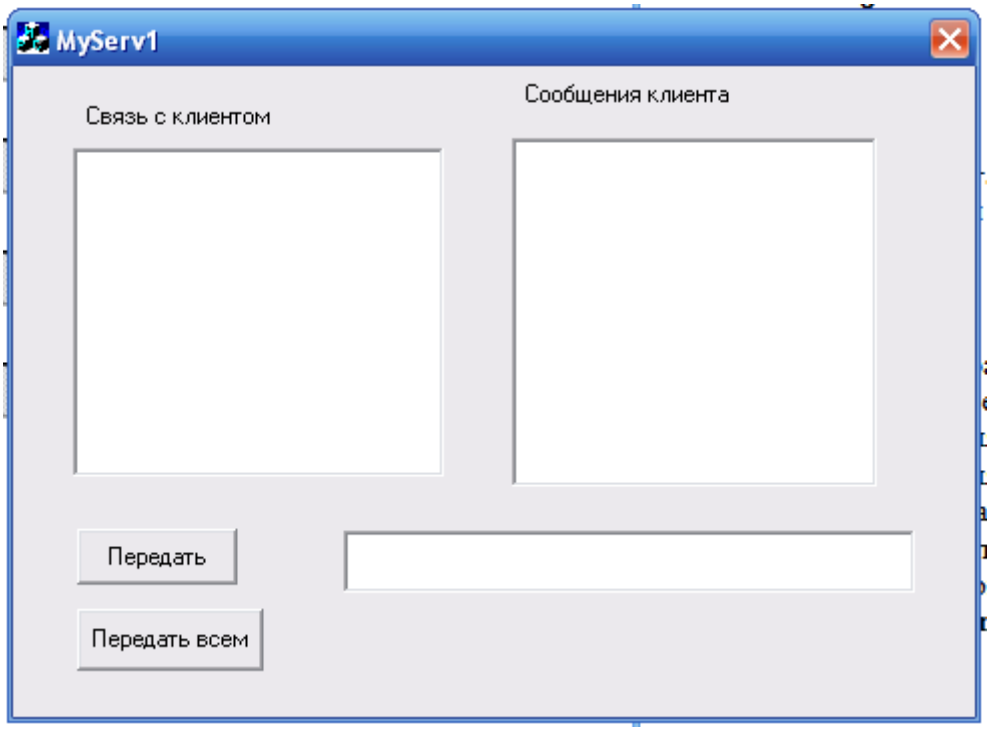
Практическая работа №4
Создание программ в среде Microsoft Visual Studio. Применение средств автоматизации разработки приложений

Продолжительность 4 часа.

Цель работы: Научиться создовать классы и объекты в VC++. Получить практические навыки по разработке приложения на основе классов и объектов в Microsoft Visual Studio VC++ .

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

Контрольные вопросы:

1 Какие средства автоматизации разработки приложения вы знаете?
2 Для чего используются структуры? Приведите примеры использования структур.
3 Как классы вы создавали в лабораторной работе?
4 Как использовали свойства наследования?

Практическая работа №5
Использование классов библиотеки MFC при создании приложений с помощью Microsoft Visual Studio. Использование динамических и статических библиотек

Продолжительность 4 часа.

Цель работы: Изучить приемы работы с классами и объектами библиотеки MFC. Научится создавать и отлаживать приложения работающие в сети

Пояснения к выполнению работы:

Объединяем все разработки по предыдущим практическим работам в одну работающую систему. Программа клиент должна передавать данные от операторов на серверную часть программы, которая ведет базу данных. Возможны различные варианты передачи и приема данных.

Контрольные вопросы:

1 Для чего служит библиотека .MFC?
2 Какие классы в ы использовали в своем приложении из библиотеки .MFC?
3 Чем отличаются классы от объектов?.
4 Какие объекты вы создавали на основе классов из библиотеки .MFC?.

Порядок оформления отчета

Отчет оформляется по результатам выполнения лабораторной работы на стандартных листах бумаги формата А4 с помощью редактора Word. Оформление отчета является важной частью лабораторной работы. При работе над отчетом студент должен осмыслить ход выполнения работы и научиться документировать полученные результаты.

Отчет должен содержать блок-схемы алгоритмов, оформленные по ГОСТу, необходимые структуры данных и распечатки программных кодов, разработанные студентом.

В отчете могут быть приведены тестовые данные, использовавшиеся студентом для проверки правильности работы программы. По результатам проверки программы должны быть сделаны выводы. Если в работе получились не те результаты, которые ожидались, в отчете должны быть отмечены причины, по которым не удалось получить желаемые результаты.

Допускается оформление одного отчета по всему курсу, выполненных в течение семестра лабораторных работ.

Пример оформления титульного листа для лабораторной работы приведен в Приложении.

Список литературы

1 Камаев В.А. Технологии программирования: Учебник/В.А. Камаев, В.В. Костерин. — 2-е изд., перераб. и доп. — М.: Высш. шк.,2006. - 454 с: ил.
2 Жоголев Е.А. Технология программирования/Е.А. Жоголев. — М.: Научный мир, 2004. — 216 с.
3 Одинцов И.О. Профессиональное программирование. Системный подход/И.О. Одинцов. - СПб: BHV-СПб., 2002. - 512 с.
4 Давыдов В.Г Программирование и основы алгоритмизации: Учеб. пособие. - М.: Высш.шк , 2003.
5 Холзнер С. Microsoft Visual C++6 с самого начала – СПб.: Питер 2005.
6 Либерти Джесс. Освой самостоятельно C++ за 21 день.- М.: Издательский Дом "Вильяме", 2001. - 814 с.
7 Павловская Т.А. C/C++: Программирование на языке высокого уровня. Учебник. - СПб.: Питер,2001. - 460 с.
8 http://www.cetiforum.ru
9 http://www.microsoft.ru
10 http:// www.firststeps.ru
11 http://www.realcoding.net/
12 Франка П. С ++ учебный курс**:** Питер – 2005.
13 Павловская Т.А., Щупак Ю.А. **C++.** Объектно-ориентированное программирование: Практикум: Питер – 2005

Приложение А

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра автоматизации производственных процессов

ОТЧЕТ

по лабораторным работам
по дисциплине
«Технология программирования»

Выполнил:
студент гр. 214 Иванов И.И.

Проверил:
преподаватель Скобелев И.В.

Курган 2013

Скобелев Игорь Вадимович

ИЗУЧЕНИЕ ОСНОВ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ VC++ В СРЕДЕ WINDOWS

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к комплексу лабораторных работ по дисциплине
«Технология программирования»
для студентов специальностей 220700. 62 «Автоматизация технологических процессов и производств» и 220400.22 «Управление в технических системах»

Авторская редакция

Подписано к печати 24.10.13	Формат 60x84 1/16	Бумага тип. №1
Печать трафаретная	Усл.печ.л. 1,5	Уч.-изд.л. 1,5
Заказ 167	Тираж 22	Цена свободная

Редакционно-издательский центр КГУ.

640669, г. Курган, ул. Гоголя, 25.

Курганский государственный университет.