Министерство образования Российской Федерации

Санкт-Петербургский Государственный Электротехнический Университет имени В.И. Ульянова (Ленина) «ЛЭТИ»

Кафедра МОЭВМ

**Пояснительная записка к курсовой работе**

**по ООП**

**Разработка объектно-ориентированной программы, использующей обмен сообщениями между проблемными объектами**

|  |
| --- |
| **Выполнил:** Студент группы 7304  Монько А. О.  **Проверил:**  Смольянинов А. В. |
|  |

Санкт-Петербург

2010

Цель работы

Выполнить разработку объектно-ориентированной программы, использующей обмен сообщениями между объектами.

Задание

Способ выполнения работы

Модернизация объектно-ориентированной программы путем включения механизма обмена сообщениями между объектами.

**Исходные программы**

* Объектно-ориентированная программа, моделирующая построение и использование набора геометрических фигур (проблемная программа), разработанная при выполнении индивидуального задания для лабораторной работы № 3.
* Объектно-ориентированная программа, моделирующая процесс обмена сообщениями (почтовая программа), разработанная при выполнении индивидуального задания для лабораторной работы № 4.

**Схема обмена**

Определена индивидуальным заданием для выполнения лабораторной работы №4.

**Содержание модернизации**

1. В системе объектов, разработанных при выполнении лабораторных работ № 2 и   
№ 3, выбираются проблемные объекты, которые могли бы выполнить функции сервера (серверов) и клиентов системы обмена сообщениями.

2. Определяются назначение, структура и представление сообщений, которыми должны обмениваться объекты для реализации проблемных действий.

Должно быть предусмотрено:

- сообщение, отправляемое объекту; при этом содержание сообщения задает действие, которое должен выполнить объект-получатель, а также параметры, необходимые для выполнения заданного действия; получение сообщения означает выполнение объектом-получателем заданного действия;

- подтверждение, направляемое объектом-получателем объекту-отправителю, сообщающее об успешном получении сообщения и выполнении получателем заданного действия; получение подтверждения не требует никаких ответных действий.

3. Выбранные объекты трансформируются с помощью отношения наследования в объекты, способные выполнять роли проблемного почтового сервера (серверов) и проблемных почтовых клиентов.

4. Из новых объектов, способных обмениваться сообщениями, конструируется новая программа, обладающая теми же проблемными возможностями, что и исходная проблемная программа, и позволяющая обрабатывать как объекты, снабженные средствами развитого обмена сообщениями, так и объекты, не наделенные подобными средствами.

**Результат модернизации**

Объектно-ориентированная программа, использующая новые проблемные объекты, способные обмениваться сообщениями и образованные из объектов проблемной программы путем наследования механизма обмена сообщениями.

1. **Формулирование и обоснование основной идеи оснащения системы проблемных объектов средствами обмена сообщениями**

Программа, разработанная в лабораторной работе №3, выполняет перемещение закрытых поршневых механизмов в указанную точку, или на определенное смещение.

Вид на с выполняет роль сервера. Закрытые механизмы, содержатся в нем. Вид дает команду механизмам передвинуться в точку или переместиться на некоторое смещение. Таким образом, можно реализовать схему обмена сообщениями, между видом на стенд и закрытыми механизмами.

1. **Разработка развернутой системы общения объектов, включающей структуру участников обмена сообщениями, виды сообщений, состав информации, входящей в сообщения, структуру сообщений, пути прохождения вариантов сообщений.**

Согласно заданию на лабораторную работу №4, реализуется схема обмена сообщениями «сервер — клиент — сервер».

Можно четко выделить один объект-сервер - вид на стенд с механизмами, и множество объектов-клиентов — закрытые поршневыми механизмами.

Виды передаваемых сообщений:

* Командное сообщение (от сервера клиенту, с указанием действия и параметров)
* Подтверждение получения (от сервера клиенту, сразу в ответ на полученное сообщение.)
* Отчет о выполненном действии (от клиента к серверу.)

В «командном сообщении» информационная часть содержит команду и соответствующие параметры:

2 — Переместиться в указанную точку, передается точка.

3 — Передвинуться на смещение, передаются параметры dx,dy

4 — Передвинуть поршень на смещение dy, передаются параметры dx = 0,dy

Пути прохождения сообщений:

1. «Вид на стенд» отправляет «Механизму» сообщение.

2. «Механизм» получает сообщение.

3. «Механизм» отправляет «Виду на стенд» отчет о выполненном действии

4. «Виду на стенд» отправляет «Механизму» подтверждение о получении сообщения.

1. **Разработка архитектуры системы новых объектов, способных обмениваться сообщениями на основе схемы «почтовый клиент –почтовый сервер»**

Объект «Сервер-вид» (СServerView)

* Расширяет функционал класса «Вид на стенд с закрытыми поршневыми механизмами», добавляя возможность обмена сообщениями.
* Состав атрибутов остается неизменным.
* Способен получать сообщения только от зарегистрированных клиентов и отправлять тоже только зарегистрированным клиентам.
* Отправка подтверждения о получении сообщения обеспечивается системой классов разработанных в лабораторной работе №4.

Объект «Клиент-закрытый механизм» (CClientClosedPistonMechanism)

* По составу атрибутов ничем не отличается от «Закрытый поршневой механизм»
* Способен получать сообщения от Сервера-вида и отправлять сообщения серверу-виду.
* Переопределяет метод Action() базового класса CClient, для осуществления необходимой обработки сообщения и выполнения указанного действия.
* При получении «Командного сообщения» сразу же начинает его обработку, а затем отправку отчета о выполнении серверу.

1. **Формирование с помощью наследования классов для новых проблемных объектов, способных обмениваться сообщениями.**

Класс «Сервер-вид» наследуется от класса «Сервер» и класса «Вид на стенд с поршневыми механизмами», т. к. он является одновременно и сервером и видом.

Класс «Клиент-механизм» наследуется от класса «Клиент» и класса «Закрытый поршневой механизм», т. к. он является одновременно и клиентом и поршневым механизмом.

1. **Проектирование системы описаний классов как системы файлов на языке С++**

CPoint.h Проект класса «Точка»

CPoint.cpp Определение класса «Точка»

CTria.h Проект класса «Треугольник»

CTria.cpp Определение класса «Треугольник»

CSection.h Проект класса «Отрезок»

CSection.cpp Определение класса «Отрезок»

CRect.h Проект класса «Прямоугольник»

CRect.cpp Определение класса «Прямоугольник»

CListND.h Проект класса «Элемент кольцевого списка»

CListN.h Определение класса «Элемент кольцевого списка»

CList.h Определение класса «Кольцевой список»

CListD.h Проект класса «Кольцевой список»

CCup.h Проект класса «Цилиндрический стакан»

CCup.cpp Определение класса «Цилиндрический стакан»

CPiston.h Проект класса «Поршень»

CPiston.cpp Определение класса «Поршень»

CPMec.h Проект класса «Поршневой механизм»

CPMec.cpp Определение класса «Поршневой механизм»

CPCMec.h Проект класса «Закрытый поршневой механизм»

CPCMec.cpp Определение класса «Закрытый поршневой механизм»

CPColl.h Проект класса «Набор поршневых механизмов»

CPColl.cpp Определение класса «Набор поршневых механизмов»

CCover.h Проект класса «Крышка»

CCover.cpp Определение класса «Крышка»

CView.h Проект класса «Вид на стенд с поршневыми механизмами»

CView.cpp Определение класса «Вид на стенд с поршневыми механизмами»

CMsg.h Проект класса «Сообщение»

CMsg.cpp Определение класса «Сообщение»

CCLIENT.H Проект класса «Клиент»

CCLIENT.cpp Определение класса «Клиент»

CServer.h Проект класса «Сервер»

CServer.cpp Определение класса «Сервер»

CSView.h Проект класса «Сервер - вид на поршневые механизмы»

CSView.cpp Определение класса «Сервер - вид на поршневые механизмы»

CCPCMec.h Проект класса «Клиент - закрытый поршневой механизм»

CCPCMec.cpp Определение класса «Клиент - закрытый поршневой механизм»

main.cpp Тестирующая программа и вспомогательные функции.

1. **Количественные характеристики программы на C++**

Общая длина текста программы: 4149

Размер тестирующей программы: 1035

Общее количество классов: 18

Количество классов без изменений: 13

Количество новых классов: 2

Модифицированных классов: 3

Всего файлов: 36

Файлов на классы: 37

1. **Разработанные классы на C++**

**Класс «Сообщение»**

#include "CPoint.h"

class CInfo {

public :

int command; /\*2 - Помещение в точку

3 - Плоскопараллельное перемещение на dx,dy

4 - Перемещение поршня в стакане на dy \*/

double dx;

double dy;

CPoint p;

CInfo (int \_command = 1, double \_dx = 0, double \_dy = 0, const CPoint& \_p = CPoint()) :

command(\_command), dx(\_dx), dy(\_dy), p(\_p) {}

void Print () {

cout << "(dx = " << dx << ", dy = " << dy << ", Point = " << p << ")" <<endl;

}

};

class CMessage {

private:

unsigned int FromID;

unsigned int ToID;

CInfo Info;

int Type;/\* тип сообщения:

0 - подтверждение приема

1 - отчет о выполнение

> 1 - запрос действия \*/

unsigned int direction; /\* направление передачи:

0 - сервер->клиент

1 - клиент->сервер \*/

static unsigned int debug;

static unsigned int total;

const unsigned int id;

static unsigned int current;

public:

CMessage(unsigned int \_FromID = 0, unsigned int \_ToID = 0, int \_Type = 0, const CInfo& \_Info = 0, unsigned int \_dir = 0);

CMessage(const CMessage& msg);

~CMessage();

unsigned int GetFromID() const { return FromID; }

unsigned int GetToID() const { return ToID; }

unsigned int GetType() const { return Type; }

const CInfo& GetInfo() const { return Info; }

unsigned int GetDirection() const { return direction; }

void SetFromID(unsigned int \_FromID);

void SetToID(unsigned int \_ToID);

void SetType(unsigned int \_Type);

void SetInfo(const CInfo& \_Info);

void SetDirection(unsigned int \_dir);

const char\* InterpritateType (unsigned int type);

void Print () const;

static void SetDebug (unsigned int d) { debug = d; }

unsigned int GetID () { return id; }

static unsigned int GetCurrent () { return current; }

static unsigned int GetTotal () { return total; }

static unsigned int Get\_debug() { return debug; }

};

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include "CMsg.h"

#include <iostream.h>

unsigned int CMessage :: current = 0;

unsigned int CMessage :: total = 0;

unsigned int CMessage :: debug = 0;

CMessage :: CMessage(unsigned int \_FromID, unsigned int \_ToID, int \_Type, const CInfo& \_Info, unsigned int \_dir) :

FromID(\_FromID), ToID(\_ToID), Type(\_Type), Info(\_Info), id(++total) {

++current;

if (debug) {

cout << InterpritateType(Type)<< " Message with id = " << id << " created"<<endl;

if (direction == 0) cout << "From server with id = " <<FromID<<"to client with id = "<<ToID<<endl;

else cout << "From client with id = " <<FromID<<"to server with id = "<<ToID<<endl;

cout << "Info = ";Info.Print(); cout<<endl;

}

}

CMessage :: CMessage(const CMessage &msg)

:id(++total), FromID(msg.GetFromID()), ToID(msg.GetToID()), Type(msg.GetType()), Info(msg.GetInfo()),

direction(msg.GetDirection()) {

++current;

if (debug) {

cout << InterpritateType(Type)<< " Message with id = " << id << " copied"<<endl;

if (direction == 0) cout << "From server with id = " <<FromID<<"to client with id = "<<ToID<<endl;

else cout << "From client with id = " <<FromID<<"to server with id = "<<ToID<<endl;

cout << "Info = ";Info.Print(); cout<<endl;

}

}

CMessage :: ~CMessage() {

--current;

if (debug) {

cout << InterpritateType(Type)<< " Message with id = " << id << " deleted"<<endl;

if (direction == 0) cout << "From server with id = " <<FromID<<"to client with id = "<<ToID<<endl;

else cout << "From client with id = " <<FromID<<" to server with id = "<<ToID<<endl;

cout << "Info = ";Info.Print(); cout<<endl;

}

}

void CMessage :: SetFromID(unsigned int \_FromID){

FromID = \_FromID;

}

void CMessage :: SetToID(unsigned int \_ToID){

ToID = \_ToID;

}

void CMessage :: SetInfo(const CInfo& \_Info){

Info = \_Info;

}

void CMessage :: SetType(unsigned int \_Type){

Type = \_Type;

}

void CMessage :: SetDirection(unsigned int \_dir){

direction = \_dir;

}

const char\* CMessage :: InterpritateType (unsigned int type){

switch(type){

case 0 :{

return "Confirm";

}break;

case 1 :{

return "Report";

}break;

default:{

return "Action";

}

}

}

void CMessage :: Print () const{

cout <<endl<< InterpritateType (Type)<<" message ";

if (direction == 0) cout << "from server with id = " <<FromID<<" to client with id = "<<ToID<<endl;

else cout << "from client with id = " <<FromID<<" to server with id = "<<ToID<<endl;

cout << "Info = ";Info.Print(); cout<<endl;

}

**Класс «Клиент»**

#include "CList.h"

#include "CServer.h"

#include "CMsg.h"

#include <iostream.h>

class CServer;

class CClient

{

private:

static unsigned int debug;

static unsigned int total;

const unsigned int id;

static unsigned int current;

//список серверов на которых зарегестрирован клиент

CircleList<CServer\*> ServerList;

virtual void Action(unsigned int \_type, const CInfo& \_info);

public:

CClient();

virtual ~CClient();

CServer\* GetServerByID(unsigned int \_id); //получить сервер по ID

CircleList<CServer\*> GetServerList() const { return ServerList; }

unsigned int ServerCount() const; //кол-во зарегестрированных серверов

void Register(CServer \*server); // Зарегестрироваться на сервере

void Unregister(unsigned int \_id); // Сняться с регистрации

void UnregisterAll(); //Сняться с регистрации со всех серверов

void SendMessage(CMessage \*msg, unsigned int \_id); // отправить сообщение серверу

void SendMessage(unsigned int \_type, const CInfo& \_info, unsigned int \_id); // отправить сообщение серверу

void ReceiveMessage(CMessage \*msg); // Получить сообщение

virtual void Print() const;

static void SetDebug (unsigned int d) { debug = d; }

virtual unsigned int GetID () { return id; }

static unsigned int GetCurrent () { return current; }

static unsigned int GetTotal () { return total; }

static unsigned int Get\_debug() { return debug; }

};

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include "CClient.H"

#include "CList.h"

#include "CListN.h"

#include <iostream.h>

#include <conio.h>

unsigned int CClient :: current = 0;

unsigned int CClient :: total = 0;

unsigned int CClient :: debug = 0;

CClient :: CClient() : id(++total) {

++current;

if (debug)

{

cout << "Client with id = " << id << " is created" <<endl

<< "total = " << total << ", current = " << current<<endl;

}

}

CClient :: ~CClient()

{

UnregisterAll();

--current;

if (debug)

{

cout << "Client with id = " << id << " is deleted" <<endl

<< "total = " << total << ", current = " << current<<endl;

}

}

CServer\* CClient :: GetServerByID(unsigned int \_id)

{

ServerList.GetFirst();

for (int i = 0; i < ServerList.GetSize(); i++,ServerList.NextCur()) {

if (ServerList.GetElemByNum(i)->GetValue()->GetID() == \_id)

return ServerList.GetCur()->GetValue();

}

return 0;

}

unsigned int CClient :: ServerCount() const {

return ServerList.GetSize();

}

void CClient :: Register(CServer \*server) {

if (server == NULL) {

cout << "Client with id = "<<id<<endl

<<"Failed to register on server due to wrong address"<<endl;

}

else {

if (GetServerByID(server->GetID()) != NULL) {

cout << "Client with id = "<<id<<endl

<<"Already registered on server with id = "<<server->GetID()<<endl;

}

else {

ServerList.Append(new CLNode<CServer\*>(server));

if (server->GetClientByID(id) == NULL) {

server->Register(this);

cout << "Client with id = " << id << " was registered on server with id = "

<< server->GetID()<<endl;

}

}

}

}

void CClient::Unregister(unsigned int \_id)

{

CServer \*server = GetServerByID(\_id);

if (server == NULL) {

cout << "Client with id = "<<id << " can not unregister from server with id = "<<server->GetID()<<endl

<<"cause it does not registered on it"<<endl;

}

else {

ServerList.ExcludeCur();

if (server->GetClientByID(id) != NULL) {

server->Unregister(id);

cout << "Client with id = "<<id<<" was unregistered from server with id = "<<server->GetID()<<endl;

}

}

}

void CClient::UnregisterAll() {

if (ServerList.GetSize() == 0) {

cout << "There are already no registered servers exists"<<endl;

}

else {

int tlen = ServerList.GetSize();

for (int i = 0; i < tlen;i++) {

Unregister(ServerList.GetElemByNum(i)->GetValue()->GetID());

}

cout << "Client with id = "<<id<<" was unregistered from all servers"<<endl;

}

}

void CClient::SendMessage(CMessage \*msg, unsigned int \_id)

{

CServer \*server = GetServerByID(\_id);

if (server != NULL) {

msg->SetFromID(id);

msg->SetToID(\_id);

msg->SetDirection(1);

server->ReceiveMessage(msg);

// cout << "Client with id = "<<id<< " sent message to server with id = "<<\_id<<endl;

}

else {

cout << "Client with id = " << id<< "could not send messages to server with id = "<<\_id<<endl

<<"cause it not registered on it"<<endl;

getch();

// delete msg;

}

}

void CClient::SendMessage(unsigned int \_type, const CInfo& \_info, unsigned int \_id)

{

CServer \*server = GetServerByID(\_id);

CMessage \*msg = new CMessage;

if (server != NULL) {

msg->SetFromID(id);

msg->SetToID(\_id);

msg->SetType(\_type);

msg->SetInfo(\_info);

msg->SetDirection(1);

server->ReceiveMessage(msg);

// cout << "Client with id = "<<id<< " sent message to server with id = "<<\_id<<endl;

}

else {

cout << "Client with id = " << id<< "could not send messages to server with id = "<<\_id<<endl

<<"cause it not registered on it"<<endl;

getch();

}

delete msg;

}

void CClient::ReceiveMessage(CMessage \*msg)

{

CServer \*server = GetServerByID(msg->GetFromID());

if (server != NULL) {

cout <<endl<< "Client with id = "<<id<< " recived message"<<endl

<<"from server with id = "<<server->GetID()<<endl;

msg->Print();

getch();

if (msg->GetType() > 1) {

/\*CMessage \*msg1 = new CMessage(\*msg);

msg1->SetType(0); //код подтверждения приёма сообщения

msg1->SetDirection(1);

SendMessage(msg1, server->GetID()); //отправка подтверждения о приёме

//сообщения \*/

Action (msg->GetType(), msg->GetInfo());

//отправка подтверждения о выполнении действия

CMessage \*msg2 = new CMessage();

msg2->SetType(1);

msg2->SetDirection(1);

SendMessage(msg2, server->GetID());

//delete msg2;

}

}

else {

cout << "Client with id = " <<id<< "could not recive message from unknown server"<<endl;

}

//delete msg;

}

void CClient::Action(unsigned int \_code, const CInfo& \_info)

{

switch (\_code)

{

case 2:

cout << "Client is handling action with code = "<<\_code<<endl

<< "And parametrs = ";\_info.Print(); cout <<endl;

break;

case 3:

cout << "Client is handling action with code = "<<\_code<<endl

<< "And parametrs = ";\_info.Print(); cout <<endl;

break;

default:

cout << "Client could not handle action with code = "<<\_code<<endl

<< "case of unknown code "<< endl<<endl;

break;

}

}

void CClient :: Print() const {

cout << "Client : "<<endl

<< "ID = "<<id<<endl

<< "Current number of clients = "<<current<<endl

<< "Number of registered servers = "<<ServerList.GetSize() <<endl

<< "List of servers : "<<endl;

if (ServerList.GetSize() == 0) {

cout << "List is empty"<<endl;

}

else {

ServerList.GetFirst();

for (int i = 0;i < ServerList.GetSize(); i++,ServerList.NextCur()) {

cout << "#"<< i+1<<" Server, ID = "<<ServerList.GetCur()->GetValue()->GetID()<<endl;

}

}

}

**Класс «Сервер»**

#include "CMSG.H"

#include "CCLIENT.H"

#include <iostream.h>

class CClient;

class CServer

{

private:

static unsigned int debug;

static unsigned int total;

const unsigned int id;

static unsigned int current;

CircleList<CClient\*> ClientList;

public:

CServer();

virtual ~CServer();

CClient\* GetClientByID(unsigned int \_id);

unsigned int Client\_Count() const;

void Register(CClient \*client);

void Unregister(unsigned int \_id);

void UnregisterAll();

void SendMessage(CMessage \*msg, unsigned int \_id);

void SendMessage(unsigned int \_type, const CInfo& \_info, unsigned int \_id);

void SendMessageToAll(CMessage \*msg);

void SendMessageToAll(unsigned int \_type, const CInfo& \_info);

virtual void ReceiveMessage(CMessage \*msg);

virtual void Print() const;

static void SetDebug (unsigned int d) { debug = d; }

unsigned int GetID () { return id; }

static unsigned int GetCurrent () { return current; }

static unsigned int GetTotal () { return total; }

static unsigned int Get\_debug() { return debug; }

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "CList.h"

#include "CListN.h"

#include "CServer.H"

#include <iostream.h>

#include <conio.h>

unsigned int CServer :: current = 0;

unsigned int CServer :: total = 0;

unsigned int CServer :: debug = 0;

CServer :: CServer() : id(++total) {

++current;

if (debug) {

cout <<"Server with id = "<<id<<" created"<<endl;

cout << "Total number of servers = "<<total<<endl

<<"Current number of servers = "<<current<<endl;

}

}

CServer :: ~CServer() {

UnregisterAll();

--current;

if (debug) {

cout <<"Server with id = "<<id<<" deleted"<<endl;

cout << "Total number of servers = "<<total<<endl

<<"Current number of servers = "<<current<<endl;

}

}

CClient\* CServer :: GetClientByID(unsigned int \_id) {

ClientList.GetFirst();

for (int i = 0; i < ClientList.GetSize();i++,ClientList.NextCur()) {

if (ClientList.GetCur()->GetValue()->GetID() == \_id)

return ClientList.GetCur()->GetValue();

}

return 0;

}

unsigned int CServer :: Client\_Count() const {

return ClientList.GetSize();

}

void CServer :: Register(CClient \*client)

{

if (client == NULL) {

cout << "Server with id = "<<id<<endl

<<"Failed to register client due to wrong address"<<endl;

}

else {

if (GetClientByID(client->GetID()) != NULL) {

cout << "Client with id = "<<client->GetID()<<endl

<<"Already registered on server with id = "<<id<<endl;

}

else {

ClientList.Append(new CLNode<CClient\*>(client));

if (client->GetServerByID(id) == NULL)

{

client->Register(this);

cout << "Client with id = " << client->GetID() << " was registered on server with id = "

<< id<<endl;

}

}

}

}

void CServer :: Unregister(unsigned int \_id) {

CClient\* client = GetClientByID(\_id);

if (client == NULL) {

cout << "Client with id = "<<client->GetID()<< " can not unregister from server with id = "<<id<<endl

<<"cause it does not registered on it"<<endl;

}

else {

ClientList.ExcludeCur();

if (client->GetServerByID(id) != NULL) {

client->Unregister(id);

cout << "Client with id = "<<client->GetID()<<" was unregistered from server with id = "<<id<<endl;

}

}

}

void CServer ::UnregisterAll() {

if (ClientList.GetSize() == 0) {

cout << "There are already no registered clients exists"<<endl;

}

else {

ClientList.GetFirst();

int tlen = ClientList.GetSize();

for (int i = 0; i < tlen;i++) {

// ClientList.GetFirst();

/\* ClientList.ExcludeCur();\*/Unregister(ClientList.GetFirst()->GetValue()->GetID());

}

cout << "Server with id = "<<id<<" unregistered all clients"<<endl;

}

}

void CServer :: SendMessage(CMessage \*msg, unsigned int \_id)

{

CClient \*client = GetClientByID(\_id);

if (client != NULL) {

msg->SetFromID(id);

msg->SetToID(\_id);

msg->SetDirection(0);

client->ReceiveMessage(msg);

// cout << "Server with id = "<<id<< " sent message to client with id = "<<\_id<<endl;

}

else {

cout << "Server with id = " << id<< " could not send messages to client with id = "<<\_id<<endl

<<"cause it not registered on it"<<endl;

getch();

// delete msg;

}

}

void CServer :: SendMessageToAll(unsigned int \_type, const CInfo& \_info)

{

CMessage\* msg = new CMessage(id,1,\_type,\_info,0);

if (ClientList.GetSize() != 0) {

msg->SetDirection(0);

int tlen = ClientList.GetSize();

for (int i = 0; i < tlen; i ++) {

msg->SetFromID(id);

msg->SetToID(ClientList.GetElemByNum(i)->GetValue()->GetID());

ClientList.GetElemByNum(i)->GetValue()->ReceiveMessage(msg);

}

// cout << "Server with id = "<<id<< " sent message to all clients "<<endl;

}

else {

cout << "Server with id = " << id<< "could not send messages to clients"<<endl

<<"cause they are not registered on it"<<endl;

getch();

}

// delete msg;

}

void CServer :: SendMessageToAll(CMessage \*msg)

{

SendMessageToAll(msg->GetType(),msg->GetInfo());

}

void CServer :: SendMessage(unsigned int \_type, const CInfo& \_info, unsigned int \_id)

{

CClient \*client = GetClientByID(\_id);

CMessage \*msg = new CMessage;

if (client != NULL) {

msg->SetFromID(id);

msg->SetToID(\_id);

msg->SetType(\_type);

msg->SetInfo(\_info);

msg->SetDirection(0);

client->ReceiveMessage(msg);

// cout << "Server with id = "<<id<< " sent message to client with id = "<<\_id<<endl;

}

else {

cout << "Server with id = " << id<< "could not send messages to client with id = "<<\_id<<endl

<<"cause it not registered on it"<<endl;

getch();

}

delete msg;

}

void CServer :: ReceiveMessage(CMessage \*msg) {

CClient\* client = GetClientByID(msg->GetFromID());

if (client != NULL)

{

cout << "Server with id = "<<id<< " recived message"<<endl

<<"from client with id = "<<client->GetID()<<endl;

msg->Print();

getch();

if (msg->GetType() > 0 && msg->GetType() < 2)

{

CMessage \*msg1 = new CMessage(\*msg);

msg1->SetType(0); //код подтверждения приёма сообщения

msg1->SetDirection(0);

SendMessage(msg1, client->GetID());

delete msg1;

}

if (msg->GetType() >= 2)

{

CMessage \*msg1 = new CMessage(\*msg);

msg1->SetType(msg->GetType()); //код подтверждения приёма сообщения

msg1->SetInfo(msg->GetInfo());

msg1->SetDirection(0);

SendMessage(msg1, client->GetID());

delete msg1;

}

}

else

{

cout << "Server with id = " <<id<< "could not recive message from unknown client"<<endl;

}

// delete msg;

}

void CServer::Print() const{

cout << "Server : "<<endl

<< "ID = "<<id<<endl

<< "Current number of servers = "<<current<<endl

<< "Number of registered clients = "<<ClientList.GetSize() <<endl

<< "List of clients : "<<endl;

if (ClientList.GetSize() == 0) {

cout << "List is empty"<<endl;

}

else {

ClientList.GetFirst();

for (int i = 0;i < ClientList.GetSize(); i++,ClientList.NextCur()) {

cout << "#"<< i+1<<" Server, ID = "<<ClientList.GetCur()->GetValue()->GetID()<<endl;

}

}

}

**Класс «Сервер-вид»**

#include "CView.h"

#include "CServer.h"

#include "CPoint.h"

class CServerView : public CView, public CServer {

public:

CServerView(CPoint \_P1,double \_P1P2,CPoint \_P3);

CServerView(const CView& view);

virtual ~CServerView();

void ReciveMessage(CMessage\* msg);

void Print() const;

};

#endif

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "CSView.h"

CServerView :: CServerView(CPoint \_P1,double \_P1P2,CPoint \_P3)

: CView(\_P1,\_P1P2,\_P3){

}

CServerView :: CServerView(const CView& view)

: CView(view){

}

CServerView :: ~CServerView() {

}

void CServerView :: ReciveMessage(CMessage\* msg){

CServer :: ReceiveMessage(msg);

if(msg->GetType() > 2){

CServer :: SendMessage(msg->GetType(),msg->GetInfo(),msg->GetFromID());

}

}

void CServerView :: Print() const {

CView :: Print();

CServer :: Print();

cout << "----------------------";

cout << endl<< endl;

}

**Класс «Клиент-механизм»**

#include <iostream.h>

#include "CPoint.h"

#include "CPCMec.h"

#include "CClient.h"

class CClientClosedPistonMechanism : public CClosedPistonMechanism, public CClient {

public:

CClientClosedPistonMechanism(const CClosedPistonMechanism& mec);

CClientClosedPistonMechanism(CPoint,double,double,double,double,double,double,double,double,double,double);

CClientClosedPistonMechanism(double,double,double,double,double,double,double,double,double,double,double,double);

CClientClosedPistonMechanism();

virtual ~CClientClosedPistonMechanism();

void Move(double dx, double dy);

void SetP1(CPoint P0);

virtual int MovePiston (double);

void Print() const;

unsigned int GetID();

private:

void Action(unsigned int \_type, const CInfo& info);

};

#endif

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "CCPCMec.h"

CClientClosedPistonMechanism :: CClientClosedPistonMechanism(const CClosedPistonMechanism & mec)

: CClosedPistonMechanism(mec), CClient() {

}

CClientClosedPistonMechanism :: CClientClosedPistonMechanism()

: CClosedPistonMechanism(), CClient() {

}

CClientClosedPistonMechanism :: CClientClosedPistonMechanism(double x1, double y1, double leftwidth, double leftheight, double bottomwidth, double bottomheight, double rightwidth, double y2, double pistonheight, double stocklength,double cover\_height,double hole\_width)

: CClosedPistonMechanism(x1, y1, leftwidth, leftheight, bottomwidth, bottomheight, rightwidth, y2, pistonheight, stocklength, cover\_height, hole\_width), CClient() {

}

CClientClosedPistonMechanism :: CClientClosedPistonMechanism(CPoint P1, double leftwidth, double leftheight, double bottomwidth, double bottomheight, double rightwidth, double y2, double pistonheight, double stocklength,double cover\_height,double hole\_width)

: CClosedPistonMechanism(P1, leftwidth, leftheight, bottomwidth, bottomheight, rightwidth, y2, pistonheight, stocklength, cover\_height, hole\_width), CClient() {

}

CClientClosedPistonMechanism :: ~CClientClosedPistonMechanism() {

}

void CClientClosedPistonMechanism :: Move(double dx, double dy){

CServer\* srv = CClient :: GetServerList().GetFirst()->GetValue();

CClient :: SendMessage(3,CInfo(3,dx,dy),srv->GetID());

}

void CClientClosedPistonMechanism :: SetP1(CPoint P0){

CServer\* srv = CClient :: GetServerList().GetFirst()->GetValue();

CClient :: SendMessage(2,CInfo(2,0,0,P0),srv->GetID());

}

int CClientClosedPistonMechanism :: MovePiston(double dy){

CServer\* srv = CClient :: GetServerList().GetFirst()->GetValue();

CClient :: SendMessage(4,CInfo(4,0,dy),srv->GetID());

return 0;

}

void CClientClosedPistonMechanism :: Action(unsigned int \_type, const CInfo& info){

switch (info.command)

{

case 2:

cout << "Client is handling action with code = "<<info.command<<"(Setting P1)"<<endl

<< "And parametrs = ";info.Print(); cout <<endl;

CClosedPistonMechanism :: SetP1(info.p);

break;

case 3:

cout << "Client is handling action with code = "<<info.command<<"(Moving by dx, dy)"<<endl

<< "And parametrs = ";info.Print(); cout <<endl;

CClosedPistonMechanism :: Move(info.dx,info.dy);

break;

case 4:

cout << "Client is handling action with code = "<<info.command<<"(Moving Piston by dy)"<<endl

<< "And parametrs = ";info.Print(); cout <<endl;

CClosedPistonMechanism :: MovePiston(info.dy);

break;

default:

cout << "Client could not handle action with code = "<<info.command<<endl

<< "case of unknown code "<< endl<<endl;

break;

}

}

void CClientClosedPistonMechanism :: Print() const {

CClosedPistonMechanism :: Print();

CClient :: Print();

cout << "--------------------"<<endl<<endl;

}

unsigned int CClientClosedPistonMechanism :: GetID () {

return CClient :: GetID();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**«Тестирующая программа»**

#include <iostream.h>

#include <stdlib.h>

#include <conio.h>

#include <stdio.h>

#include <dos.h>

#include "CSection.h"

#include "CPoint.h"

#include "CRect.h"

#include "CList.h"

#include "CListN.h"

#include "CPiston.h"

#include "CCup.h"

#include "CPMec.h"

#include "CPColl.h"

#include "CCover.h"

#include "CTria.h"

#include "CStand.h"

#include "CPCMec.h"

#include "CView.h"

#include "CMsg.h"

#include "CServer.h"

#include "CClient.h"

#include "CCPCMec.h"

#include "CSView.h"

void main () {

{

cout << endl<<endl<<"-----------------------------------------------"<<endl;

cout << "Programm is creating the view on collection of schematic images of"<<endl

<< "closed piston mechanisms which can be moved or placed in point"<<endl

<< "Server->Client->Server system of message exchange is used."<<endl

<< "Programmed by : Artyom Mon'ko"<<endl<<endl

<< "----------------Press any key to continue or ESC to exit----------------"<<endl;

while(1) {

int key = getch();

if (key == 27) return; // Waiting for some action

else break;

}

int flag = 1;

int ViewCreated = 0;

char\* buffer = new char[80];

int menu = 0;

CPistonCollection c;

CPistonMechanism\* p;

CServerView \*v;

//CServer server;

CircleList<CClient\*> clients;

while (flag) {

cout << "1.Create View"<<endl

<< "2.Add Closed Mechanism to Collection"<<endl

<< "3.Add Mechanism to Collection"<<endl

<< "4.Add Client Closed Mechanism to Collection"<<endl

<< "5.Delete Mechanism from Collection"<<endl

<< "6.Move Collection and Stand"<<endl

<< "7.Place Collection and Stand into Point"<<endl

<< "8.Place Collection into Point"<<endl

<< "9.Place Stand into Point"<<endl

<< "10.Move Collection"<<endl

<< "11.Move Stand"<<endl

<< "12.Move Piston of element"<<endl

<< "13.Move every Piston"<<endl

<< "14.Print View"<<endl

<< "15.Print Collection"<<endl

<< "16.Exit"<<endl;

cin >> buffer;

if ((atoi(buffer) == 0) || (atoi(buffer) < 1 || atoi(buffer) > 16)) {

cout << "Invalid input, try again : "<<endl;

continue;

}

else {

menu = (int)atoi(buffer);

switch (menu) {

case 1: {

double x1 = 0,y1 = 0, \_P1P2,x3,y3;

cout << "Enter x value of point P1 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x1 = atof(buffer);

cout << "Enter y value of point P1 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y1 = atof(buffer);

cout << "Enter lenght of P1P2 side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

\_P1P2 = atof(buffer);

cout << "Enter x value of point P3 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x3 = atof(buffer);

cout << "Enter y value of point P3 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y3 = atof(buffer);

v = new CServerView(CPoint(x1,y1),\_P1P2,CPoint(x3,y3));

if (v->inv()) {

ViewCreated = 1;

if (c.GetLength() > 0) v->Add(c);

cout << "View is created"<<endl;}

else cout << "View do not created due to some input errors"<<endl;

break;

}

case 2: {

double x1 = 0,y1 = 0, lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width;

cout << "Enter x value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x1 = atof(buffer);

cout << "Enter y value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y1 = atof(buffer);

cout << "Enter width of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lwidth = atof(buffer);

cout << "Enter height of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lheight = atof(buffer);

cout << "Enter width of bottom side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bwidth = atof(buffer);

cout << "Enter height of bottom : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bheight = atof(buffer);

cout << "Enter width of right side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

rwidth = atof(buffer);

cout << "Enter y value of piston point P1(I) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y2 = atof(buffer);

cout << "Enter piston height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

pheight = atof(buffer);

cout << "Enter stock length : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

slen = atof(buffer);

cout << "Enter cover height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

cover\_height = atof(buffer);

cout << "Enter hole's width : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

hole\_width = atof(buffer);

p = new CClosedPistonMechanism(x1,y1,lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width);

if (p->inv()){

c.Add(p);

if (ViewCreated) v->Add(p);

cout << "Closed Piston Mechanism Added"<<endl;}

else cout << "Closed Piston Mechanism not added due to some input errors"<<endl;

break;

}

case 3: {

double x1 = 0,y1 = 0, lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width;

cout << "Enter x value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x1 = atof(buffer);

cout << "Enter y value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y1 = atof(buffer);

cout << "Enter width of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lwidth = atof(buffer);

cout << "Enter height of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lheight = atof(buffer);

cout << "Enter width of bottom side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bwidth = atof(buffer);

cout << "Enter height of bottom : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bheight = atof(buffer);

cout << "Enter width of right side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

rwidth = atof(buffer);

cout << "Enter y value of piston point P1(I) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y2 = atof(buffer);

cout << "Enter piston height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

pheight = atof(buffer);

cout << "Enter stock length : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

slen = atof(buffer);

p = new CPistonMechanism(x1,y1,lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen);

if (p->inv()){

c.Add(p);

if (ViewCreated) v->Add(p);

cout << "Piston Mechanism Added"<<endl;}

else cout << "Piston Mechanism not added due to some input errors"<<endl;

break;

}

case 4: {

double x1 = 0,y1 = 0, lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width;

cout << "Enter x value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x1 = atof(buffer);

cout << "Enter y value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y1 = atof(buffer);

cout << "Enter width of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lwidth = atof(buffer);

cout << "Enter height of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lheight = atof(buffer);

cout << "Enter width of bottom side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bwidth = atof(buffer);

cout << "Enter height of bottom : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bheight = atof(buffer);

cout << "Enter width of right side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

rwidth = atof(buffer);

cout << "Enter y value of piston point P1(I) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y2 = atof(buffer);

cout << "Enter piston height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

pheight = atof(buffer);

cout << "Enter stock length : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

slen = atof(buffer);

cout << "Enter cover height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

cover\_height = atof(buffer);

cout << "Enter hole's width : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

hole\_width = atof(buffer);

// p = new CClientClosedPistonMechanism(x1,y1,lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width);

CClientClosedPistonMechanism\* cp = new CClientClosedPistonMechanism(x1,y1,lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width);

if (cp->inv()){

c.Add(cp);

if (ViewCreated) {

v->Add(cp);

clients.Append(new CLNode<CClient\*>(cp));

v->Register(cp);

}

cout << "Client Closed Piston Mechanism Added"<<endl;}

else cout << "Client Closed Piston Mechanism not added due to some input errors"<<endl;

break;

}

case 5: {

cout << "Enter number of element : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

c.Delete(atoi(buffer));

if (ViewCreated) v->Delete(atoi(buffer));

if (c.invColl()) cout << "Element deleted"<<endl;

break;

}

case 6: {

double dx,dy;

cout << "Enter dx : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dx = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MoveCollection(dx,dy);

v->MoveStand(dx,dy);

if (v->inv()) cout << "Collection and stand are moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

else {

c.Move(dx,dy);

if (c.invColl()) cout << "Collection is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

break;

}

case 7: {

double x,y;

cout << "Enter P1 "<<endl;

cout << "Enter x : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x = atof(buffer);

cout << "Enter y : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y = atof(buffer);

if (ViewCreated) {

v->SetP1(CPoint(x,y));

v->SetStandP1(CPoint(x,y));

if (v->inv()) cout << "Collection and stand are placed to Point("<<x<<";"<<y<<")"<<endl;

}

else {

c.SetP1(CPoint(x,y));

if (c.invColl()) cout << "Collection is placed to Point("<<x<<";"<<y<<")"<<endl;

}

break;

}

case 8: {

double x,y;

cout << "Enter P1 "<<endl;

cout << "Enter x : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x = atof(buffer);

cout << "Enter y : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y = atof(buffer);

if (ViewCreated) {

v->SetCollectionP1(CPoint(x,y));

if (v->inv()) cout << "Collection is placed to Point("<<x<<";"<<y<<")"<<endl;

}

else {

c.SetP1(CPoint(x,y));

if (c.invColl()) cout << "Collection is placed to Point("<<x<<";"<<y<<")"<<endl;

}

break;

}

case 9: {

double x,y;

cout << "Enter P1 "<<endl;

cout << "Enter x : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x = atof(buffer);

cout << "Enter y : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y = atof(buffer);

if (ViewCreated) {

v->SetStandP1(CPoint(x,y));

if (v->inv()) cout << "Stand is placed to Point("<<x<<";"<<y<<")"<<endl;

}

else cout << "View is not created"<<endl;

break;

}

case 10: {

double dx,dy;

cout << "Enter dx : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dx = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MoveCollection(dx,dy);

if (v->inv()) cout << "Collection is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

else {

c.Move(dx,dy);

if (c.invColl()) cout << "Collection is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

break;

}

case 11: {

double dx,dy;

cout << "Enter dx : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dx = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MoveStand(dx,dy);

if (v->inv()) cout << "Stand is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

else

cout << "View is not created"<<dy<<endl;

break;

}

case 12: {

double dy, index = 0;

cout << "Index of element : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

index = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->GetCollection().GetElemByNum(index)->GetValue()->MovePiston(dy);

if (v->inv()) cout << "Piston of "<< index<< "'th element is moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

else {

c.GetCollection().GetElemByNum(index)->GetValue()->MovePiston(dy);

if (c.invColl()) cout << "Piston of "<< index<< "'th element is moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

break;

}

case 13: {

double dy;

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MovePiston(dy);

if (v->inv()) cout << "Pistons moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

else {

c.MovePiston(dy);

if (c.invColl()) cout << "Pistons moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

break;

}

case 14: {

if(ViewCreated) v->PrintView();

else cout << "View is not created"<<endl;

break;

}

case 15: {

if(ViewCreated) v->Print();

else c.Print();

break;

}

case 16: {

flag = 0;

break;

}

}

}

}

}

}

**7. Результаты работы программы**

Вид на все тесты : (-10;25), 200, (-12,-100)

-----------------------------------------------

Programm is creating the view on collection of schematic images of

closed piston mechanisms which can be moved or placed in point

Server->Client->Server system of message exchange is used.

Programmed by : Artyom Mon'ko

----------------Press any key to continue or ESC to exit----------------

1.Create View

2.Add Closed Mechanism to Collection

3.Add Mechanism to Collection

4.Add Client Closed Mechanism to Collection

5.Delete Mechanism from Collection

6.Move Collection and Stand

7.Place Collection and Stand into Point

8.Place Collection into Point

9.Place Stand into Point

10.Move Collection

11.Move Stand

12.Move Piston of element

13.Move every Piston

14.Print View

15.Print Collection

16.Exit

10

Enter dx :

1

Enter dy :

-1

Server with id = 1 recived message

from client with id = 1

Action message from client with id = 1 to server with id = 1

Info = (dx = 1, dy = -1, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Action message from server with id = 1 to client with id = 1

Info = (dx = 1, dy = -1, Point = (0;0))

Client is handling action with code = 3(Moving by dx, dy)

And parametrs = (dx = 1, dy = -1, Point = (0;0))

Server with id = 1 recived message

from client with id = 1

Report message from client with id = 1 to server with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Confirm message from server with id = 1 to client with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Collection is moved by dx = 1, dy = -1

1.Create View

2.Add Closed Mechanism to Collection

3.Add Mechanism to Collection

4.Add Client Closed Mechanism to Collection

5.Delete Mechanism from Collection

6.Move Collection and Stand

7.Place Collection and Stand into Point

8.Place Collection into Point

9.Place Stand into Point

10.Move Collection

11.Move Stand

12.Move Piston of element

13.Move every Piston

14.Print View

15.Print Collection

16.Exit

8

Enter P1

Enter x :

3

Enter y :

10

Server with id = 1 recived message

from client with id = 1

Action message from client with id = 1 to server with id = 1

Info = (dx = 0, dy = 0, Point = (3;10))

Client with id = 1 recived message

from server with id = 1

Action message from server with id = 1 to client with id = 1

Info = (dx = 0, dy = 0, Point = (3;10))

Client is handling action with code = 2(Setting P1)

And parametrs = (dx = 0, dy = 0, Point = (3;10))

Server with id = 1 recived message

from client with id = 1

Report message from client with id = 1 to server with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Confirm message from server with id = 1 to client with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Collection is placed to Point(3;10)

1.Create View

2.Add Closed Mechanism to Collection

3.Add Mechanism to Collection

4.Add Client Closed Mechanism to Collection

5.Delete Mechanism from Collection

6.Move Collection and Stand

7.Place Collection and Stand into Point

8.Place Collection into Point

9.Place Stand into Point

10.Move Collection

11.Move Stand

12.Move Piston of element

13.Move every Piston

14.Print View

15.Print Collection

16.Exit

13

Enter dy :

-2

Server with id = 1 recived message

from client with id = 1

Action message from client with id = 1 to server with id = 1

Info = (dx = 0, dy = -2, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Action message from server with id = 1 to client with id = 1

Info = (dx = 0, dy = -2, Point = (0;0))

Client is handling action with code = 4(Moving Piston by dy)

And parametrs = (dx = 0, dy = -2, Point = (0;0))

Server with id = 1 recived message

from client with id = 1

Report message from client with id = 1 to server with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Confirm message from server with id = 1 to client with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Pistons moved by dy = -2

**Добавили еще одного клиента - механизма**

Element 3: ClosedPistonMechanism id = 7

LeftCupSide : Rectangle : id = 20 (6;12),(7;12),(6;5),(7;5) Width = 1, Height =7

BottomCupSide : Rectangle : id = 19 (7;7),(11;7),(7;5),(11;5) Width = 4, Height = 2

RightCupSide : Rectangle : id = 18 (11;12),(12;12),(11;5),(12;5) Width = 1, Height = 7

Piston : Rectangle : id = 17 (7;10),(11;10),(7;8),(11;8) Width = 4, Height = 2

Stock : P1(9;10) -- P2(9;16), L = 6

Cover : Rectangle : id = 22 (6;14),(12;14),(6;12),(12;12) Width = 6, Height = 2

Hole : Rectangle : id = 21 (8.5;14),(9.5;14),(8.5;12),(9.5;12) Width = 1, Height = 2

1.Create View

2.Add Closed Mechanism to Collection

3.Add Mechanism to Collection

4.Add Client Closed Mechanism to Collection

5.Delete Mechanism from Collection

6.Move Collection and Stand

7.Place Collection and Stand into Point

8.Place Collection into Point

9.Place Stand into Point

10.Move Collection

11.Move Stand

12.Move Piston of element

13.Move every Piston

14.Print View

15.Print Collection

16.Exit

10

10

Enter dx

1

Enter dy

1

Server with id = 1 recived message

from client with id = 1

Action message from client with id = 1 to server with id = 1

Info = (dx = 1, dy = 1, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Action message from server with id = 1 to client with id = 1

Info = (dx = 1, dy = 1, Point = (0;0))

Client is handling action with code = 3(Moving by dx, dy)

And parametrs = (dx = 1, dy = 1, Point = (0;0))

Server with id = 1 recived message

from client with id = 1

Report message from client with id = 1 to server with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Client with id = 1 recived message

from server with id = 1

Confirm message from server with id = 1 to client with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Server with id = 1 recived message

from client with id = 2

Action message from client with id = 2 to server with id = 1

Info = (dx = 1, dy = 1, Point = (0;0))

Client with id = 2 recived message

from server with id = 1

Action message from server with id = 1 to client with id = 2

Info = (dx = 1, dy = 1, Point = (0;0))

Client is handling action with code = 3(Moving by dx, dy)

And parametrs = (dx = 1, dy = 1, Point = (0;0))

Server with id = 1 recived message

from client with id = 2

Report message from client with id = 2 to server with id = 1

Info = (dx = 0, dy = 0, Point = (0;0))

Client with id = 2 recived message

from server with id = 1

Confirm message from server with id = 1 to client with id = 2

Info = (dx = 0, dy = 0, Point = (0;0))

Collection is moved by dx = 1, dy = 1