Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 6

з дисципліни

“ООП”

Тема: «Шаблоні класи С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко  Максим  Миколайович |  | студент 2-го курсу  гр. ІП-51 ФІОТ  Субцельний Олександр Володимирович |

Київ – 2016

**ЗМІСТ:**

[1 Мета роботи 3](#_Toc463896091)

[2 Постановка задачі (варіант 20, рівень В) 4](#_Toc463896092)

[3 Діаграма класів 5](#_Toc463896093)

[4 Код програми 6](#_Toc463896094)

[5 Приклади виконання програми 10](#_Toc463896095)

[6 Висновок 11](#_Toc463896096)

# Мета роботи

Цель работы – изучить особенности шаблонов функций и шаблонов классов С++. Освоить принципы работы шаблонов классов для работы стбазовыми типами и пользовательскими типами.

# Постановка задачі (варіант 20, рівень В)

Работа спорткомплекса. Спорткомплекс предлагает следующий набор услуг: бассейн, фитнесс центр, тренажерный зал, массаж и сауна. Каждый клиент спортзала получает клубную карточку, при оформлении которой учитывается набор услуг, которыми пользуется клиент. Также учитывается режим посещения спортзала:

неограниченный – клиент посещает спорткомплекс на протяжении определенного срока, без ограничения в количестве посещений;

месячный абонент – посещение спорткомплекса в установленное время на протяжении месяца (деньги за пропущенные занятия не возмещаются);

разовый – оплата по каждому занятию отдельно.

При выборе бассейна клиент должен дополнительно пройти медицинское обследование для получения справки о состоянии здоровья. Обследование производится в медицинской клинике, являющейся партнёром спорткомплекса, результаты обследования автоматически направляются в спорткомплекс. На основании услуг, заказанных клиентами спортзала, администрация спорткомплекса формирует детальный отчёт о наиболее популярных услугах. На основании отчёта принимается решение о возможном расширении перечня услуг, об улучшении отдельных видов услуг и т.д. Сформировать коллекцию данных о клиентах спорткомплекса и услугах которыми они пользуются. Задание. Отобразить виды услуг и их количество.

# Діаграма класів

Діаграма класів наведена на рисунку 3.1:

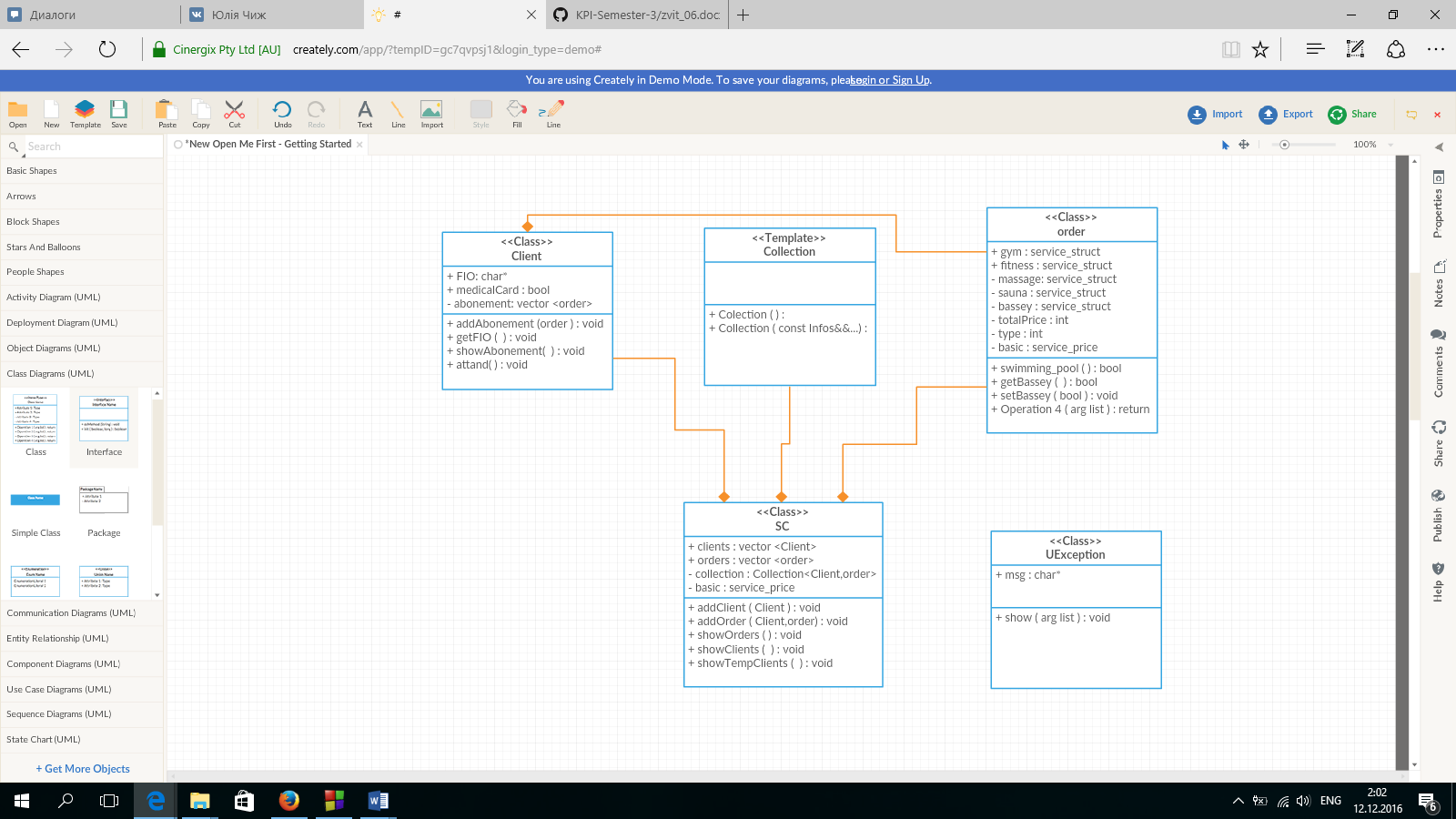


Рисунок 3.1 – Діаграма класів

# Код програми

“service\_price.h”

#ifndef SERVICE\_PRICE\_H\_INCLUDED

#define SERVICE\_PRICE\_H\_INCLUDED

struct service\_price

{

int gym=50,

fitness =40,

masssage=100,

sauna =30,

bassey=35;

int unlimited = 2000;

} ;

#endif // SERVICE\_PRICE\_H\_INCLUDED

“service\_struct.h”

#ifndef SERVICE\_STRUCT\_H\_INCLUDED

#define SERVICE\_STRUCT\_H\_INCLUDED

#define unlimit 1000

struct service\_struct

{

int price,

time,

attand=0;

bool need=false;

int totalPrice=0;

};

#endif // SERVICE\_STRUCT\_H\_INCLUDED

“main.cpp”

#include <iostream>

#include <cstdlib>

# include <algorithm>

#include <stdio.h>

#include "SC.h"

#include "Client.h"

#include "service\_struct.h"

#include "service\_pack.h"

#include "Order.h"

#include "Abonement.h"

#include "UException.h"

#include <string.h>

using namespace std;

int main()

{

cout<<"Subtselnyi Oleksandr LAB:6B. Varian:20"<<endl;

SC Complex;

int i=0;

bool finish=true;

while (finish){

char choise;

cout<<"MENU"<<endl<<endl;

cout<<"1. Create new client"<<endl;

cout<<"2. Make new abonement for client"<<endl;

cout<<"3. Take User and make attand"<<endl;

cout<<"4. Take User and show attands"<<endl;

cout<<"5. Show all used services"<<endl;

cout<<"6. Show all clients"<<endl;

cout<<"7. Show all clients(temporary)"<<endl;

cout<<"e. EXIT"<<endl;

cin>>choise;

switch(choise){

case ('1'):

{

int type;

cout<<"what abonement do you need?"<<endl<<"1.Unlimit"<<endl<<"2.Month(8 attands)"<<endl<<"3.Once"<<endl;

cin>>type;

order ordered(type);

cout<<"Enter the name of Client"<<endl;

char name[20];

scanf("%s",name);

Client cl(name,ordered);

Complex.addClient(cl);

Complex.addOrder(cl,ordered);

break;

}

case ('2'):

{

cout<<"Choose client"<<endl;

for (int i=0;i<Complex.clients.size();i++){

cout<<i+1; Complex.clients[i].getFIO();

}

int num;

cin>>num;

Complex.clients[num-1].showAbonement();

int type;

cout<<"what abonement do you need?"<<endl<<"1.Unlimit"<<endl<<"2.Month"<<endl<<"3.Once"<<endl;

cin>>type;

order ordered(type);

Complex.addOrder(Complex.clients[num-1],ordered);

Complex.clients[num-1].addAbonement(ordered);

break;

}

case ('3'):

{

cout<<"Choose client"<<endl;

for (int i=0;i<Complex.clients.size();i++){

cout<<i+1; Complex.clients[i].getFIO();

}

int num;

cin>>num;Complex.clients[num-1].showAbonement();

cout<<"What does client attand?"<<endl;

cout<<"1.GYM"<<endl<<"2.Fitness"<<endl<<"3.Massage"<<endl<<"4.Sauna"<<endl<<"5.Bassey"<<endl;

int type;

cin>>type;

Complex.clients[num-1].attand(type);

break;

}

case ('4'):

{

cout<<"Choose client"<<endl;

for (int i=0;i<Complex.clients.size();i++){

cout<<i+1; Complex.clients[i].getFIO();

}

int num;

cin>>num;Complex.clients[num-1].showAbonement();

break;

}

case ('5'):

{

Complex.showOrders();

break;

}

case ('6'):

{

Complex.showClients();

break;

}

case ('7'):

{

Complex.showTempClients();

break;

}

case ('e'):

{

cout<<"SportComplex is closing"<<endl;

finish=false;

break;

}

default:

{

cout<<"wrong input"<<endl;

}

}

}

}

“UException.h”

#ifndef UEXCEPTION\_H

#define UEXCEPTION\_H

class UException

{

private:

char\* msg;

public:

UException(char\* message);

~UException();

void show(void);

};

#endif // UEXCEPTION\_H

“SC.h”

#ifndef SC\_H

#define SC\_H

#include "Client.h"

#include "order.h"

#include "Collection.h"

#include <vector>

#include "service\_price.h"

class SC

{

public:

vector <Client> clients;

vector <order> orders;

Collection<Client, order>\* collection; // templates - collection

service\_price basic;

public:

int counted=0;

SC();

virtual ~SC();

void addClient(Client);

void addOrder(Client,order);

void showOrders();

void showClients();

void showTempClients();

};

#endif // SC\_H

“order.h”

#ifndef ORDER\_H

#define ORDER\_H

#include "service\_struct.h"

#include "service\_pack.h"

#include <stdlib.h>

#include <time.h>

#define number 20

#include <iostream>

#include <stdio.h>

#include <string.h>

#include "UException.h"

#include "service\_price.h"

using namespace std;

class order

{

public:

//static const in size = sizeof ... (Values);

service\_struct gym,fitness,massage,sauna,bassey;

int totalPrice=0;

int type;

service\_price basic;

public:

order();

order(const int &type,bool n1, bool n2, bool n3, bool n4, bool n5);

order(const int &type) ;

~order();

bool swimming\_pool();

bool getBassey();

void setBassey(bool);

};

#endif // ORDER\_H

“Collectionn.h”

#ifndef COLLECTION\_H

#define COLLECTION\_H

template<class... Infos>

class Collection : public Infos... // наследуем базовые классы, они же

{

public:

Collection(){};

Collection(const Infos&&... infos) : Infos(infos)... {}; //

};

#endif // COLLECTION\_H

“Client.h”

#ifndef CLIENT\_H

#define CLIENT\_H

#include <vector>

#include "Abonement.h"

#include "order.h"

using std::vector;

class Client

{

public:

char FIO[20];

bool medicalCard;

vector <order> abonement;

public:

Client();

Client(char[20],order ordered);

Client(char name[20], order ordered, bool check);

virtual ~Client();

void addAbonement(order ordered);

void getFIO();

void showAbonement();

void attand(int);

};

#endif // CLIENT\_H

“UException.cpp”

#include "UException.h"

#include <cstdio>

#include <string.h>

UException::UException(char\* message)

{

msg = new char[strlen(message) + 1];

strcpy(msg, message);

}

UException::~UException()

{

delete[] msg;

msg = NULL;

}

void UException::show(void) {

printf("%s\n", msg);

}

“SC.cpp”

#include "SC.h"

SC::SC()

{

cout << "Welcome to our Sport Complex!" << endl;

collection = new Collection<Client, order>[100];

}

SC::~SC()

{

//dtor

}

void SC::addClient(Client cl)

{

clients.push\_back(cl);

counted++;

bool check = cl.medicalCard;

collection[clients.size()] = Collection<Client, order>(

{cl.FIO,cl.abonement[cl.abonement.size()-1],check},

{cl.abonement[cl.abonement.size()-1].type,cl.abonement[cl.abonement.size()-1].gym.need,

cl.abonement[cl.abonement.size()-1].fitness.need,cl.abonement[cl.abonement.size()-1].massage.need,

cl.abonement[cl.abonement.size()-1].sauna.need,cl.abonement[cl.abonement.size()-1].bassey.need}

);

}

void SC::addOrder(Client cl,order rd)

{

orders.push\_back(rd);

bool check = cl.medicalCard;

collection[clients.size()] = Collection<Client, order>(

{cl.FIO,cl.abonement[cl.abonement.size()-1],check},

{rd.type,cl.abonement[cl.abonement.size()-1].gym.need,

cl.abonement[cl.abonement.size()-1].fitness.need,cl.abonement[cl.abonement.size()-1].massage.need,

cl.abonement[cl.abonement.size()-1].sauna.need,cl.abonement[cl.abonement.size()-1].bassey.need}

);

counted++;

}

void SC::showOrders()

{

int gyma=0,fita=0,masa=0,saua=0,basa=0;

int unlimited=0;

for (int i=0;i<counted;i++)

{

if (collection[i].gym.attand!=unlimit)

gyma+=collection[i].gym.attand;

else

unlimited++;

if (collection[i].fitness.attand!=unlimit)

fita+=collection[i].fitness.attand;

if (collection[i].massage.attand!=unlimit)

masa+=collection[i].massage.attand;

if (collection[i].sauna.attand!=unlimit)

saua+=collection[i].sauna.attand;

if (collection[i].bassey.attand!=unlimit)

basa+=collection[i].bassey.attand;

}

cout<<"Totaly was bought "<<gyma<<" GYM attands"<<endl;

cout<<"Totaly was bought "<<fita<<" Fitness attands"<<endl;

cout<<"Totaly was bought "<<masa<<" Massage attands"<<endl;

cout<<"Totaly was bought "<<saua<<" Sauna attands"<<endl;

cout<<"Totaly was bought "<<basa<<" Bassey attands"<<endl;

cout<<"Totaly was bought "<<unlimited<<"Unlimited abonements"<<endl;

}

void SC::showClients()

{

for (int i=0;i<clients.size();i++)

{

int gyma=0,fita=0,masa=0,saua=0,basa=0;

int unlimited=0;

cout<<i+1<<"client: "; clients[i].getFIO(); cout<<endl;

for (int j=0;j<clients[i].abonement.size();j++)

{

if ( clients[i].abonement[j].gym.attand!=unlimit)

gyma+= clients[i].abonement[j].gym.attand;

else

unlimited++;

if ( clients[i].abonement[j].fitness.attand!=unlimit)

fita+= clients[i].abonement[j].fitness.attand;

if ( clients[i].abonement[j].massage.attand!=unlimit)

masa+= clients[i].abonement[j].massage.attand;

if ( clients[i].abonement[j].sauna.attand!=unlimit)

saua+= clients[i].abonement[j].sauna.attand;

if ( clients[i].abonement[j].bassey.attand!=unlimit)

basa+= clients[i].abonement[j].bassey.attand;

}

cout<<"This client bought "<<gyma<<" GYM attands, ";

cout<<fita<<" Fitness attands, ";

cout<<masa<<" Massage attands, ";

cout<<saua<<" Sauna attands, ";

cout<<basa<<" Bassey attands, "<<endl;

cout<<unlimited<<"Unlimited abonements"<<endl;

}

}

void SC::showTempClients()

{

for (int i=0;i<clients.size();i++)

{

int gyma=0,fita=0,masa=0,saua=0,basa=0;

int unlimited=0;

cout<<i+1<<"client: "; clients[i].getFIO(); cout<<endl;

int j=clients[i].abonement.size()-1;

if ( clients[i].abonement[j].gym.attand==unlimit)

unlimited++;

else{

gyma+= clients[i].abonement[j].gym.attand;

fita+= clients[i].abonement[j].fitness.attand;

masa+= clients[i].abonement[j].massage.attand;

saua+= clients[i].abonement[j].sauna.attand;

basa+= clients[i].abonement[j].bassey.attand;

}

if (!unlimited){

cout<<"This client has "<<gyma<<" GYM attands, ";

cout<<fita<<" Fitness attands, ";

cout<<masa<<" Massage attands, ";

cout<<saua<<" Sauna attands, ";

cout<<basa<<" Bassey attands, "<<endl;

}

else

cout<<"This client has Unlimited abonement"<<endl;

}

}

“order.cpp”

#include "Order.h"

order::order()

{

}

order::order(const int &type)

{

if (type==1){

gym.need=true;

fitness.need=true;

massage.need=true;

sauna.need=true;

bassey.need=true;

fitness.attand=massage.attand=sauna.attand=bassey.attand=gym.attand=unlimit;

totalPrice = basic.unlimited;

}

else{

char str[number];

bool elements[number];

int capacity=0;

cout<<"which services do you need?[Enter through ',']"<<endl;

cout<<"1.GYM"<<endl<<"2.Fitness"<<endl<<"3.Massage"<<endl<<"4.Sauna"<<endl<<"5.Bassey"<<endl;

bool ready=true;

do{

str[0]='/0';

scanf("%s",str);

int len = strlen(str);

try{

bool flag=true,

prevflag=true;

for (int i=1;i<len;i+=2){

if (str[i]!=',') flag=false;

if (!flag) throw UException("substring in your Multi\_Set");

if ((int)str[i-1]<33||int(str[i-1])>127||(int)str[0]<33||(int)str[0]>127) throw UException("not a symbol of ASCII");

prevflag=flag;

}

ready=false;

}

catch (UException &e){

e.show();

ready=true;

str[0]='/0';

}

}while(ready);

for (int i = 0; i <= 5; i++) elements[i] = false;

int i=0;

char \* pch = strtok (str,",");

int n = atoi(pch);

elements[n]=true;

while (pch != NULL)

{

pch = strtok (NULL, ",");

n = atoi(pch);

elements[n]=true;

}

gym.need=elements[1];

fitness.need=elements[2];

massage.need=elements[3];

sauna.need=elements[4];

bassey.need=elements[5];

}

this->type=type;

if (type==3){

if (gym.need) {

totalPrice+=basic.gym;

gym.attand=1;

}

if (fitness.need)

{totalPrice+=basic.fitness;

fitness.attand=1;

}

if (massage.need){

totalPrice+=basic.masssage;

massage.attand=1;

}

if (sauna.need) {

totalPrice+=basic.sauna;

sauna.attand=1;

}

if (bassey.need){

totalPrice+=basic.bassey;

bassey.attand=1;

}

}

if (type==2){

if (gym.need) {

totalPrice+=8\*basic.gym;

gym.attand=8;

}

if (fitness.need)

{

totalPrice+=8\*basic.fitness;

fitness.attand=8;

}

if (massage.need){

totalPrice+=8\*basic.masssage;

massage.attand=8;

}

if (sauna.need) {

totalPrice+=8\*basic.sauna;

sauna.attand=8;

}

if (bassey.need){

totalPrice+=8\*basic.bassey;

bassey.attand=8;

}

}

cout<<endl<<"Succesfully created order"<<endl<<"Total price is: "<<totalPrice<<endl;;

}

order::order(const int &type,bool n1, bool n2, bool n3, bool n4, bool n5)

{

if (type==1){

gym.need=true;

fitness.need=true;

massage.need=true;

sauna.need=true;

bassey.need=true;

fitness.attand=massage.attand=sauna.attand=bassey.attand=gym.attand=unlimit;

totalPrice = basic.unlimited;

}

else{

gym.need=n1;

fitness.need=n2;

massage.need=n3;

sauna.need=n4;

bassey.need=n5;

}

this->type=type;

if (type==3){

if (gym.need) {

totalPrice+=basic.gym;

gym.attand=1;

}

if (fitness.need)

{totalPrice+=basic.fitness;

fitness.attand=1;

}

if (massage.need){

totalPrice+=basic.masssage;

massage.attand=1;

}

if (sauna.need) {

totalPrice+=basic.sauna;

sauna.attand=1;

}

if (bassey.need){

totalPrice+=basic.bassey;

bassey.attand=1;

}

}

if (type==2){

if (gym.need) {

totalPrice+=8\*basic.gym;

gym.attand=8;

}

if (fitness.need)

{

totalPrice+=8\*basic.fitness;

fitness.attand=8;

}

if (massage.need){

totalPrice+=8\*basic.masssage;

massage.attand=8;

}

if (sauna.need) {

totalPrice+=8\*basic.sauna;

sauna.attand=8;

}

if (bassey.need){

totalPrice+=8\*basic.bassey;

bassey.attand=8;

}

}

}

order::~order()

{

//dtor

}

bool order::swimming\_pool()

{

char answer;

cout<<"You've chosen a bassey! You need to pass the medical inspection"<<endl<<endl;

cout<<"Do you have gerpes?[Y/n]"<<endl;

cin>>answer;

if (answer=='y'||answer=='Y')

return false;

srand (time(NULL));

int AIDS= rand() % 100 + 1;

if (AIDS<5){

cout<<"We have found problems with our health! NO BASSEY"<<endl;

return false;

}

return true;

}

bool order::getBassey()

{

return bassey.need;

}

void order::setBassey(bool checked)

{

bassey.need=checked;

}

“Client.cpp”

#include "Client.h"

Client::Client()

{

}

Client::Client(char name[20], order ordered, bool check)

{

strcpy(FIO,name);

if (ordered.getBassey())

{

ordered.setBassey(check);

}

abonement.push\_back(ordered);

}

Client::Client(char name[20], order ordered)

{

strcpy(FIO,name);

if (ordered.getBassey())

{

bool check = ordered.swimming\_pool();

ordered.setBassey(check);

}

abonement.push\_back(ordered);

cout<<"created new Client and abonement"<<endl;

}

Client::~Client()

{

//dtor

}

void Client::getFIO()

{

cout<<FIO<<endl;

}

void Client::addAbonement(order ordered)

{

abonement.push\_back(ordered);

cout<<"added new abonement"<<endl;

}

void Client::showAbonement()

{

switch (abonement[abonement.size()-1].type){

case (1):{

cout<<"Your type is UNLIMITED"<<endl;

if(abonement[abonement.size()-1].bassey.need)

cout<<"You can attand bassey"<<endl;

break;

}

default:

if(abonement[abonement.size()-1].gym.need){

cout<<"You can attend GYM"<<endl;

cout<<"You have "<<abonement[abonement.size()-1].gym.attand<<"attands"<<endl;

}

if(abonement[abonement.size()-1].fitness.need){

cout<<"You can attend Fitness"<<endl;

cout<<"You have "<<abonement[abonement.size()-1].fitness.attand<<"attands"<<endl;

}

if(abonement[abonement.size()-1].sauna.need){

cout<<"You can attend sauna"<<endl;

cout<<"You have "<<abonement[abonement.size()-1].sauna.attand<<"attands"<<endl;

}

if(abonement[abonement.size()-1].massage.need){

cout<<"You can attend massage"<<endl;

cout<<"You have "<<abonement[abonement.size()-1].massage.attand<<"attands"<<endl;

}

if(abonement[abonement.size()-1].bassey.need){

cout<<"You can attend bassey"<<endl;

cout<<"You have "<<abonement[abonement.size()-1].bassey.attand<<"attands"<<endl;

}

}

}

void Client::attand(int s)

{

if (abonement[abonement.size()-1].type==1)

{

cout<<"you are unlimited client"<<endl;

}

else

switch (s){

case (1):{

if (abonement[abonement.size()-1].gym.attand)

abonement[abonement.size()-1].gym.attand--;

else

cout<<"no more attands"<<endl;

break;

}

case (2):{

if (abonement[abonement.size()-1].fitness.attand)

abonement[abonement.size()-1].fitness.attand--;

else

cout<<"no more attands"<<endl;

break;

}

case (3):{

if (abonement[abonement.size()-1].massage.attand)

abonement[abonement.size()-1].massage.attand--;

else

cout<<"no more attands"<<endl;

break;

}

case (4):{

if (abonement[abonement.size()-1].sauna.attand)

abonement[abonement.size()-1].sauna.attand--;

else

cout<<"no more attands"<<endl;

break;

}

case (5):{

if (abonement[abonement.size()-1].bassey.attand)

abonement[abonement.size()-1].bassey.attand--;

else

cout<<"no more attands"<<endl;

break;

}

}

cout<<"atanded succesfuly"<<endl;

}

# Приклади виконання програми

Приклад виконання програми наведений на рисунку 5.1:

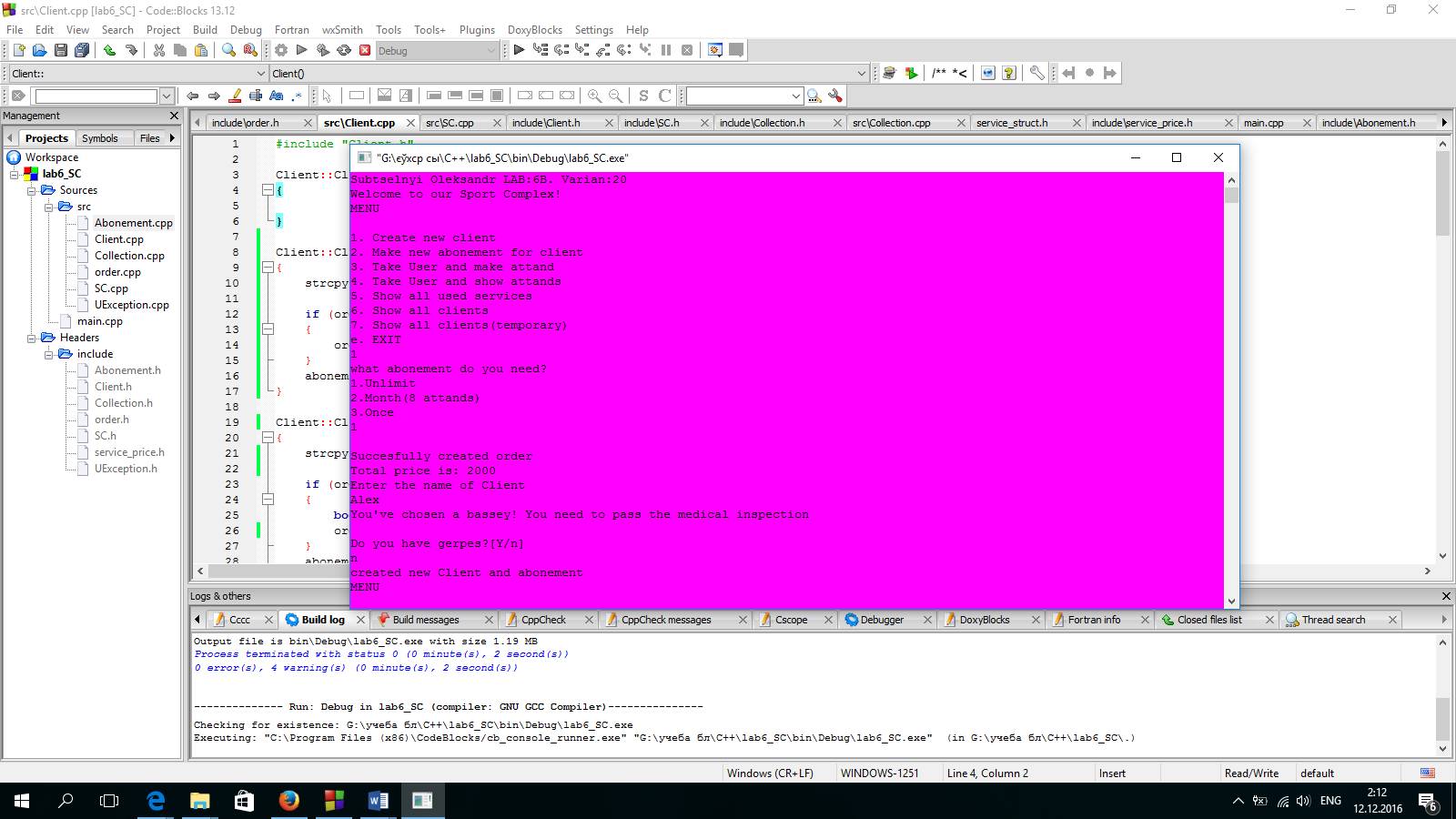


Рисунок 5.1 – Приклад виконання програми

# Висновок

Отже, дана програма ємулює роботу Спортивного комплексу та свторює картотеку клієнтів та їх абонементів. Є можливість створити новий абонемент клієнту, використати можливе відвідування. Контролюється кількість відвідувань, а також інформується про ціну. Є 3 варіанта абонемента: разовий, місячний з вибором послуг та безлімітний. Оскільки дана програма реалізована за модульним принципом і кожен модуль програми створювався й налагоджувався автономно, то дана програма буде видавати очікуваний результат на всіх наборах вхідних даних.