Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 1

з дисципліни

“ООП”

Тема: «Класи С++»

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

Київ – 2016

Зміст:

[1 Мета роботи 3](#_Toc461290255)

[2 Постановка задачі (варіант 7, рівень В) 4](#_Toc461290256)

[3 Покроковий алгоритм 5](#_Toc461290257)

[4 Діаграма класів 7](#_Toc461290258)

[5 Код програми 8](#_Toc461290259)

[6 Приклади виконання програми 18](#_Toc461290260)

[7 Висновок 19](#_Toc461290261)

# Мета роботи

Мета роботи – вивчити основні концепції обєктно-орієнтованого програмування. Вивчити особливості використання класів та об’єктів, а також особливості використання конструкторів та деструкторів.

# Постановка задачі (варіант 7, рівень В)

Определить класс «Блэкджек», с закрытым элементом: массив игроков размерности n, не более 8 (элементами массива должны выступать игроки с набором карт и фишек, для этого стоит создать еще один класс «Игрок» с соответствующими атрибутами и методами). Определить такие конструкторы:

– конструктор инициализации, задающий начальную раздачу карт для игроков и дилера.

Определить методы для игры в Блэкджек между пользователем и другими игроками-компьютерами. За дилера играет компьютер.

Дополнительно предусмотреть все необходимые атрибуты.

Реализовать запись процесса игры в текстовый файл.

Определить деструктор класса.

# Покроковий алгоритм

* Обчислення дій учасника-компютера

1. ПОЧАТОК
2. Підрахувати кількість карт, що залишилося в колоді (**total**).
3. Знайти кількість очок, що не вистачає гравцю до «21» (**left**).
4. Підрахувати кількість карт, що залишилися в колоді та їхня вага не перебільшує **left** (**luck**).
5. ЯКЩО ймовірність витягнути хорошу (**luck**) карту більша за 36,36% (тобто luck \* 2,75 > total), ТО гравцю варто витягнути наступну карту та перейти до пункту 2. ІНАКШЕ гравцю не потрібно більше тягнути карту.
6. КІНЕЦЬ

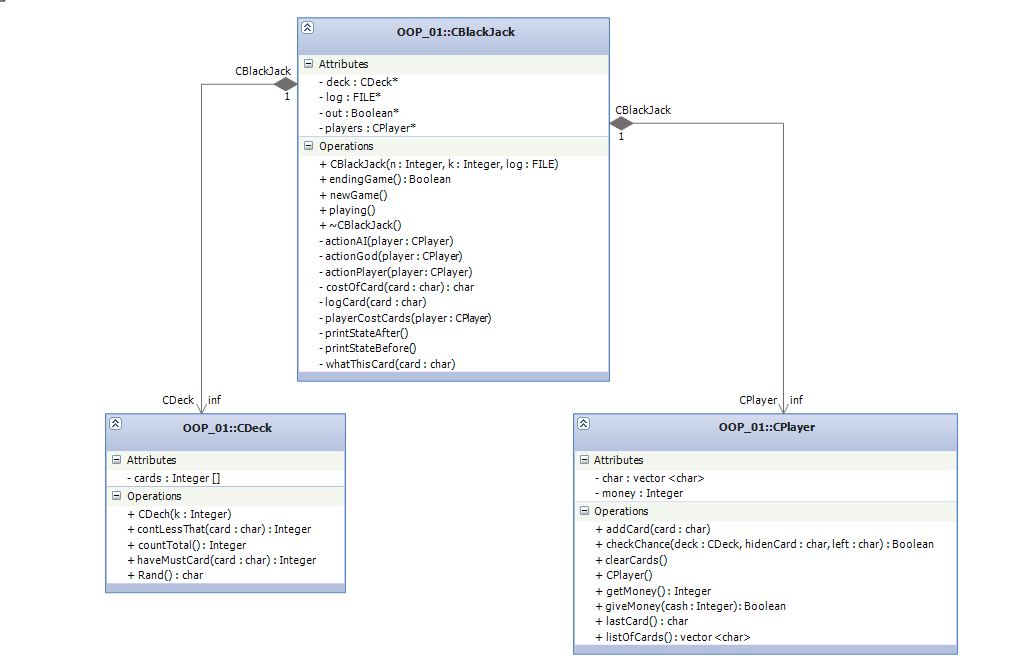
* Обчислення дій ділера-компютера

1. ПОЧАТОК
2. Підрахувати кількість карт, що залишилося в колоді (**total**).
3. Знайти кількість очок, що не вистачає гравцю до «21» (**left**).
4. Підрахувати кількість карт, що залишилися в колоді та їхня вага не перебільшує **left** (**luck**).
5. ЯКЩО ймовірність витягнути хорошу (**luck**) карту більша за 36,36% (тобто luck \* 2,75 > total) ТА поточна сума карт в ділера менша за 17, ТО гравцю варто витягнути наступну карту та перейти до пункту 2. ІНАКШЕ гравцю не потрібно більше тягнути карту.
6. КІНЕЦЬ

* Обчислення (генерація) наступної карти в колоді

1. ПОЧАТОК
2. Підрахувати кількість карт, що залишилося в колоді (**total**).
3. Згенерувати випадкове число від 0 до total виключно (**number**).
4. ЦИКЛ по всім картам (від 2 до Туза) та мастям (від А до D) (тобто цикл йде по елементам 2A, 2B, 2C, 2D, 3A, …., TD):
   1. ЯКЩО кількість карт поточного типу менша, ніж **number** ТО позначити карту поточного типу як наступну карту в колоді та вилучити її з колоди (віддати гравцю). ІНАКШЕ відняти від **number** кількість карт поточного типу.
5. КІНЕЦЬ

# Діаграма класів



# Код програми

“main.cpp”

#include "Interface.h"

#include <cstdlib>

#include <ctime>

int main(void) {

srand(time(NULL));

while (outputMenu());

}

“Interface.h”

#pragma once

#include "CBlackJack.h"

#define PLAYERS\_LIMIT 8

void input(int&, int, char); // Функція введення

bool outputMenu(void); // Меню

bool GameMenu(CBlackJack\* (&)); // Ігрового меню

“Interface.cpp”

#include "Interface.h"

#include <cstdio>

#include <time.h>

// Функція введення

void input(int& x, int limit, char ch) {

printf("Input %c: ", ch);

scanf("%d", &x);

while (x <= 0 || x > limit) {

printf("Wrong number! Try again. ");

scanf("%d", &x);

}

}

// Виведення меню

bool outputMenu(void) {

system("cls");

printf("OOP lab work #1\nVariant - 7, Level - C\nTask: Write BlackJack game\nWriter: Alexander ALEXKIRNAS Zarichkovyi\n\n");

printf("Menu:\n1. Start game\n2. Exit\n\nMake your choice ");

int t;

scanf("%d", &t);

switch (t) {

case 1: {

int n, k;

input(n, PLAYERS\_LIMIT, 'n');

input(k, PLAYERS\_LIMIT, 'k');

time\_t rawtime;

struct tm \* timeinfo;

char buffer[200];

time(&rawtime);

timeinfo = localtime(&rawtime);

strftime(buffer, 200, "%d.%m.%Y-%H.%M.%S.log", timeinfo);

FILE\* file = fopen(buffer, "wt");

fprintf(file, "Beging loging:");

CBlackJack\* game = new CBlackJack(n, k, file);

while (GameMenu(game));

fprintf(file, "End of log");

fclose(file);

delete game;

break;

}

case 2: return 0;

}

return true;

}

// Ігрового меню

bool GameMenu(CBlackJack\* (&game)) {

system("cls");

printf("Menu:\n 1. Next game\n 2. Out\nMake your choice: ");

int t;

scanf("%d", &t);

switch (t) {

case 1: {

game->newGame();

game->playing();

if (game->endingGame()) {

printf("\n\nGame over! =(\n");

system("pause");

return 0;

}

break;

}

case 2: return 0;

}

return 1;

}

“CDeck.h”

#pragma once

#define NUM\_OF\_CARDS 52 // Кількість різних карт в колоді

// Клас колоди карт

class CDeck {

private:

int cards[NUM\_OF\_CARDS + 1]; // Кількість карт кожного типу в колоді

public:

char Rand(void); //Видача випадкової карти

CDeck(int k); // Конструктор класу

int haveMushCards(char) const; // Визначення кількості карт в колоді заданого типу

int countTotal(void) const; // Визначенння кількості карт, що залишилися в колоді

int countLessThat(char) const; // Визначення кількості карт, що не більші за задану

};

“CDeck.cpp”

#include "СDeck.h"

#include <cstdlib>

//Видача випадкової карти

char CDeck::Rand(void) {

int countOfCards = 0;

for (int i = 0; i < NUM\_OF\_CARDS; i++) countOfCards += cards[i];

int numberOfCard = (rand() \* 1.0f / static\_cast <double> (RAND\_MAX)) \* countOfCards;

for(int i = 0; i < NUM\_OF\_CARDS; i++, numberOfCard -= cards[i])

if (numberOfCard <= cards[i]) {

cards[i]--;

return i;

}

}

// Конструктор класу

CDeck::CDeck(int k) {

for (int i = 0; i < NUM\_OF\_CARDS; i++) cards[i] = k;

}

// Визначення кількості карт в колоді заданого типу

int CDeck::haveMushCards(char index) const {

if (index < NUM\_OF\_CARDS) return cards[index];

else return 0;

}

// Визначенння кількості карт, що залишилися в колоді

int CDeck::countTotal(void) const {

int count = 0;

for (int i = 0; i < NUM\_OF\_CARDS; i++) count += cards[i];

return count;

}

// Визначення кількості карт, що не більші за задану

int CDeck::countLessThat(char k) const {

k -= 2;

int size = 4 \* (k + 1);

if (k == 10) k = NUM\_OF\_CARDS - 4;

else if (k >= 11) k = NUM\_OF\_CARDS;

int count = 0;

for (int i = 0; i < size; i++) count += cards[i];

return count;

}

“CPlayer.h”

#pragma once

#include <vector>

#include "СDeck.h"

using std::vector;

#define START\_MONEY 1000

// Клас гравця

class CPlayer {

private:

vector <char> cards; // Карти, що є на руках у гравця

int money; // Кількість грошей (фішок), що є у гравця

public:

void addcard(char); // Функція добавлення нової карти

bool giveMoney(int); // Функція знімання наступної ставки

CPlayer(void); // Конструктор за замовчуванням

bool checkChance(const CDeck\*, char, char); // Підрахувати ймовірність вибору "хорошої карти"

char lastCard(void) const; // Показує останню зданю карту

vector <char> listOfCards(void) const; // Повертає список карт

void clearCards(void); // Забирає всі карти у гравця

int getMoney(void) const; // Повертає кількість грошей в гравця

};

“CPlayer.cpp”

#include "CPlayer.h"

// Функція добавлення нової карти

void CPlayer::addcard(char newCard) {

cards.push\_back(newCard);

}

// Функція знімання наступної ставки

bool CPlayer::giveMoney(int cash) {

if (cash <= this->money) this->money -= cash;

else return false;

return true;

}

// Конструктор за замовчуванням

CPlayer::CPlayer(void) : money(START\_MONEY) {}

// Підрахувати ймовірність вибору "хорошої карти"

bool CPlayer::checkChance(const CDeck\* deck, char hidenCard, char left) {

int total = deck->countTotal() + 1; // Підрахувати кількість карт, що залишилмся в колоді

int luck = deck->countLessThat(left); // Підрахувати кількількіть карт, що можна безпечно взяти

if (hidenCard <= left) luck++; // Якщо серед безпечних карт є прихована карта ділера, то її потрібно забрати

// Обрати дію виходячи з ймовірності

if (luck \* 2.75f > total) return true;

else return false;

}

// Показує останню зданю карту

char CPlayer::lastCard(void) const {

return cards.back();

}

// Повертає список карт

vector <char> CPlayer::listOfCards(void) const {

return cards;

}

// Забирає всі карти у гравця

void CPlayer::clearCards(void) {

cards.clear();

}

// Повертає кількість грошей в гравця

int CPlayer::getMoney(void) const {

return money;

}

“CBlackJack.h”

#pragma once

#include "CPlayer.h"

#include "СDeck.h"

#include <cstdio>

#include <algorithm>

#define COST\_OF\_GAME 500

class CBlackJack {

private:

CPlayer\* players; // Стан гравців

CDeck\* deck; // Колода карт

bool\* out; // Гравці що покинули гру

FILE\* log; // Файл логу гри

int n; // Кількість гравців

int k; // Кількість колод

void actionPlayer(CPlayer&); // Вибір гравцем дій

void actionAI(CPlayer&); // Виконання дій компютером

void actionGod(CPlayer&); // Виконання дій ділера

void whatThisCard(char) const; // Виводить карту на екран

void logCard(char) const; // Записує задану карту в лог

char costOfCard(char) const; // Визначає ціну карти

char playerCostCards(const CPlayer& player) const; // Визначення ціни всіх карт у гравця

void printStateBefore(void); // Виведення стану гри

void printStateAfter(void); // Виведення стану гри

public:

CBlackJack(int, int, FILE\*); // Ініціалізація гри

~CBlackJack(void); // Очищення гри

void newGame(void); // Нова гра

void playing(void); // Процес гри

bool endingGame(void); // Завершення гри

};

“CBlackJack.cpp”

#include "CBlackJack.h"

// Виводить карту на екран

void CBlackJack::whatThisCard(char card) const {

char number = card / 4 + 2;

if (number <= 10) printf("%d", number);

else switch (number) {

case 11: printf("V"); break;

case 12: printf("D"); break;

case 13: printf("K"); break;

case 14: printf("T"); break;

};

printf("-%c ", char(card % 4 + 'a'));

}

// Записує задану карту в лог

void CBlackJack::logCard(char card) const {

char number = card / 4 + 2;

if (number <= 10) fprintf(log, "%d", number);

else switch (number) {

case 11: fprintf(log, "V"); break;

case 12: fprintf(log, "D"); break;

case 13: fprintf(log, "K"); break;

case 14: fprintf(log, "T"); break;

};

fprintf(log, "-%c ", char(card % 4 + 'a'));

}

// Визначає ціну карти

char CBlackJack::costOfCard(char card) const {

char number = card / 4 + 2;

if (number <= 10) return number;

else if (number != 14) return 10;

else return 1;

}

// Визначення ціни всіх карт у гравця

char CBlackJack::playerCostCards(const CPlayer& player) const {

int cost = 0;

vector <char> cards = player.listOfCards();

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

cost += costOfCard(cards[i]);

return cost;

}

// Ініціалізація гри

CBlackJack::CBlackJack(int n, int k, FILE\* log) {

this->n = n; this->k = k;

players = new CPlayer[(n + 1)];

this->log = log;

out = new bool[n + 1];

for (int i = 0; i <= n; i++) out[i] = false;

}

// Очищення гри

CBlackJack::~CBlackJack(void) {

delete[] out;

delete[] players;

}

// Нова гра

void CBlackJack::newGame(void) {

fprintf(log, "\n\nStarting new game...\n\n");

fprintf(log, "You make new ante %d$ and begin new game\n", COST\_OF\_GAME);

players[0].giveMoney(COST\_OF\_GAME);

for (int i = 1; i < n; i++) {

if (!players[i].giveMoney(COST\_OF\_GAME)) out[i] = true;

else fprintf(log, "Player %d make ante %d$ and begin new game\n", i, COST\_OF\_GAME);

}

fprintf(log, "Swaping cards in deck...\n");

deck = new CDeck(k);

for (int i = 0; i < 2; i++)

for (int z = 0; z <= n; z++) if (!out[z]) {

char card = deck->Rand();

if (z != n) fprintf(log, "Adding to player %d card ", i);

else fprintf(log, "Adding to dealer сard ");

logCard(card);

fprintf(log, "\n");

players[z].addcard(card);

}

}

// Процес гри

void CBlackJack::playing(void) {

printStateBefore();

actionPlayer(players[0]);

printf("\n\nActions of other players...\n");

fprintf(log, "\n\nActions of other players...\n");

for (int i = 1; i < n; i++) if(!out[i]) {

fprintf(log, "\n\nActions of Player %d:\n", i);

actionAI(players[i]);

}

fprintf(log, "\n\nActions of dealer:\n");

actionGod(players[n]);

printStateAfter();

system("pause");

}

// Завершення гри

bool CBlackJack::endingGame(void) {

int cost = playerCostCards(players[n]);

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

int playercost = playerCostCards(players[i]);

if ((playercost == cost && playercost == 21 && players[i].listOfCards().size() == 2)

|| (playercost > cost && playercost <= 21) || (playercost <= 21 && cost > 21)) {

players[i].giveMoney(-2 \* COST\_OF\_GAME);

fprintf(log, "\nPlayer %d win %d$", i, COST\_OF\_GAME \* 2);

}

players[i].clearCards();

}

players[n].clearCards();

delete deck;

fprintf(log, "\n\nEnding of game...");

if (players[0].getMoney() < COST\_OF\_GAME) return 1;

return 0;

}

// Вибір гравцем дій

void CBlackJack::actionPlayer(CPlayer& player) {

fprintf(log, "\nAction of Player 0:\n");

while (true) {

int cost = playerCostCards(player);

if (cost > 21) break;

printf("\nYou current cost: %d", cost);

printf("\nDo you want new card? (Y\\N) ");

char str[10];

scanf("%s", str);

if (\*str == 'Y' || \*str == 'y') {

char card = deck->Rand();

player.addcard(card);

fprintf(log, "Added to Player 0 card ");

logCard(card); fprintf(log, "\n");

printf("Added card "); whatThisCard(card);

printf("\n");

system("pause");

}

else if (\*str == 'n' || \*str == 'N') break;

else printf("Wrong letter! Try again.\n");

}

}

// Виконання дій компютером

void CBlackJack::actionAI(CPlayer& player) {

int cost = playerCostCards(player);

while (player.checkChance(deck, players[n].lastCard(), std::max(0, 21 - cost))) {

char card = deck->Rand();

player.addcard(card);

fprintf(log, "Added card ");

logCard(card); fprintf(log, "\n");

cost = playerCostCards(player);

}

}

// Виконання дій ділера

void CBlackJack::actionGod(CPlayer& player) {

int cost = playerCostCards(player);

// Ділер тягне нову карту якщо це вигідно і сума його карт не перебільшує 17

while (cost < 17 && player.checkChance(deck, NUM\_OF\_CARDS, std::max(0, 21 - cost))) {

char card = deck->Rand();

player.addcard(card);

fprintf(log, "Added card ");

logCard(card); fprintf(log, "\n");

cost = playerCostCards(player);

}

}

// Виведення стану гри

void CBlackJack::printStateBefore(void) {

printf("Begin of game status:\n\n");

for (int i = 0; i < n; i++) if(!out[i]){

if(i) printf("Player %d:\n", i);

else printf("You:\n");

vector <char> cards = players[i].listOfCards();

printf("Money: %d$\nCards: ", players[i].getMoney());

for (int z = 0, size = cards.size(); z < size; z++) whatThisCard(cards[z]);

printf("\nTotal cost: %d", playerCostCards(players[i]));

printf("\n\n");

}

printf("Dealer: \n");

vector <char> cards = players[n].listOfCards();

for (int z = 0, size = cards.size(); z < (size - 1); z++) whatThisCard(cards[z]);

printf("\*\n\n");

//printf("\nTotal cost: %d\n\n", playerCostCards(players[n]));

}

// Виведення стану гри

void CBlackJack::printStateAfter(void) {

printf("End of game status:\n\n");

int cost = playerCostCards(players[n]);

for (int i = 0; i < n; i++) if (!out[i]) {

if (i) printf("Player %d:\n", i);

else printf("You:\n");

vector <char> cards = players[i].listOfCards();

int playercost = playerCostCards(players[i]);

for (int z = 0, size = cards.size(); z < size; z++) whatThisCard(cards[z]);

printf("\nTotal cost: %d", playerCostCards(players[i]));

printf("\nStatus: %s\n\n", (((playercost == cost && players[i].listOfCards().size() == 2 && playercost == 21)

|| (playercost > cost && playercost <= 21) || (playercost <= 21 && cost > 21)) ? "win" : "lose") );

}

printf("Dealer: \n");

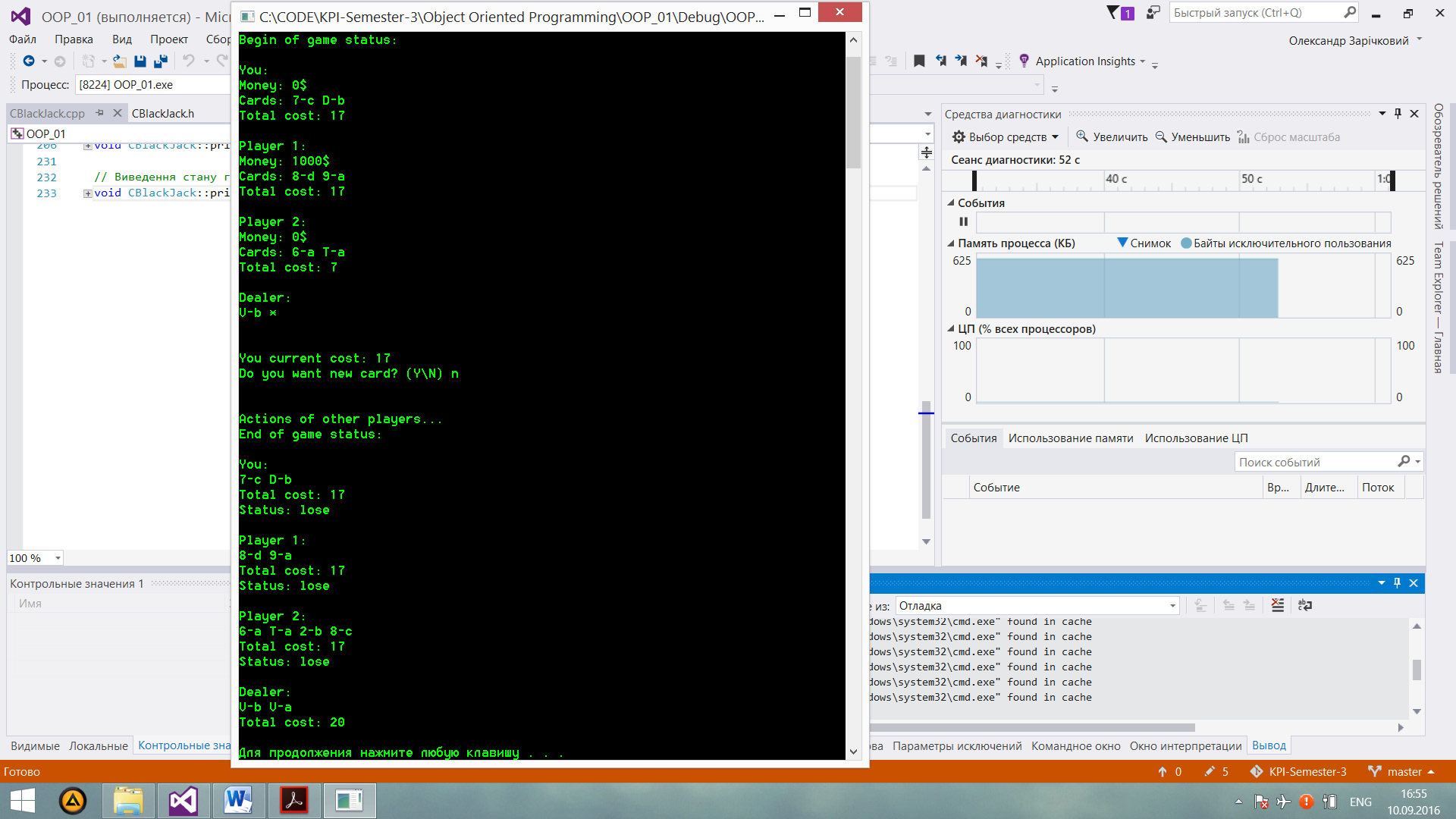
vector <char> cards = players[n].listOfCards();

for (int z = 0, size = cards.size(); z < size; z++) whatThisCard(cards[z]);

printf("\nTotal cost: %d\n\n", playerCostCards(players[n]));

}

# Приклади виконання програми



# Висновок

Отже, дана програма створює об’єкт класа CBlackJack, яка реалізує відому гру Блек Джек, а також об’єкти класів CDeck та СPlayer, які являють собою реалізації карткової колоди та гравця в карти відповідно. 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.