**М і н і с т е р с т в о о с в і т и і н а у к и У к р а ї н и**

**Національний університет “Львівська політехніка”**

Кафедра ЕОМ



**Курсова робота**

з предмету: «Програмування, частина 2 (Об’єктно-

орієнтоване програмування)»

на тему:

**«Базові принципи об’єктно-орієнтованого програмування»**

Індивідуальне завдання: «Казино»

Виконав:

ст. гр. КІ-15

Височанський Сергій

Прийняв:

Львів-2020

АНОТАЦІЯ

Для розв’язання багатьох практичних задач з різних галузей, необхідно використовувати базові принципи об’єктно-орієнтованого програмування, оскільки вони можуть адаптувати вирішення поставлених задач до їх швидкої зміни в реальному часі. Одним з шляхів забезпечення таких вимог є організація об’єктно-орієнтованих процесів і відповідних технічних засобів їх реалізації. Причому, ефективність застосування об’єктно-орієнтованого програмування у вирішенні поставленого завдання залежить, як від продуктивності комп’ютерів, так і від розмірів і структури пам’яті, пропускної здатності каналів зв’язку, використаних об’єктно-орієнтованих мов програмування, компіляторів, операційних систем, чисельних методів та інших математичних досліджень. Такий широкий обсяг параметрів вимагає проведення досліджень на різних рівнях: на рівні опису алгоритмів, створення спеціальних мов програмування, компіляторів, багатопроцесорних систем, неоднорідних систем, кластерів. Для скорочення термінів розробки об’єктно-орієнтованих програм та дослідження їх роботи використовується моделювання. Метою виконання курсової роботи є засвоєння основних методів та алгоритмів об’єктно-орієнтованих структур, принципів побудови відповідних структур, набуття початкових практичних навиків проектування таких засобів. В результаті вивчення курсу студент повинен: знати: основні методи, алгоритми і засоби опрацювання інформації, засоби програмування об’єктно-орієнтованих структур, склад апаратних засобів та програмного забезпечення обчислювальних систем; вміти: виконувати елементарні вправи з об’єктно-орієнтованого програмування, описувати алгоритми поставленого перед ним завдання, використовувати класи та об'єкти, застосовувати модульне програмування.

**Зміст**

[**Анотація** 2](#_Toc515438689)

[**Завдання на курсову роботу** 4](#_Toc515438690)

[**1. Огляд та обґрунтовування вибору технологій об'єктно-орієнтованого програмування** 5](#_Toc515438691)

[**2. Аналіз та розробка алгоритму згідно індивідуального завдання** 6](#_Toc515438692)

[2.1. Опис алгоритму та оцінка його складності 6](#_Toc515438693)

[2.2. Аналіз особливостей алгоритму 8](#_Toc515438694)

[2.3. Опис алгоритму у псевдокоді 9](#_Toc515438695)

[2.4. Розробка граф-схеми алгоритму 13](#_Toc515438696)

[**3. Програмна реалізація алгоритму згідно індивідуального завдання** 14](#_Toc515438697)

[3.1. Опис послідовної програмної реалізації алгоритму 14](#_Toc515438698)

[3.1.1. Опис структур даних програми 14](#_Toc515438699)

[3.1.2. Діаграма класів 15](#_Toc515438700)

[3.1.3. Опис реалізації основних функцій/методів 16](#_Toc515438701)

[3.2. Опис програмної реалізації алгоритму 18](#_Toc515438702)

[**4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання** 33](#_Toc515438703)

[**Висновок** 36](#_Toc515438704)

[**Список використаної літератури** 37](#_Toc515438705)

ЗАВДАННЯ НА КУРСОВУ РОБОТУ

Індивідуальне завдання «Казино».

Вхідні дані взято з сайту одного з казино Лос-Анджелеса. А саме: принцип роботи банку казино, принцип розподілу роботи між працівниками, обслуговування клієнтів, найпопулярніші ігри казино та їхні правила. На базі цієї інформації було створено алгоритм програми, яка буде реалізовувати ці дані. Казино було поділено на банк, персонал та ігри. Усе це реалізовано за допомогою класів та їх методів.

ВСТУП

У сучасному світі розваги відіграють велику роль у житті суспільства. Люди готові витрачати гроші задля того щоб відпочити. Тому тема моєї курсової є досить актуальною.

Звичайно не кожна людина може собі дозволити відвідування казино. Це переважно розвага для багатих людей які просто хочуть забути про роботу та відійти від буденності. Я створив програму, у якій постарався максимально передати відчуття перебування у реальному казино. При цьому робив усе щоб програма була зручною для користувача.

Метою моєї курсової роботи є: оволодіти базовими елементами мови програмування С++, та за допомогою них написати програму.

1.ОГЛЯД ТА ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ОБЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення.

Успадкування (наслідування). Клас може мати «підкласи», спеціалізовані, розширені версії надкласу. Можуть навіть утворюватися цілі дерева успадкування. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть вводити свої власні. Успадкування може бути одиничне (один безпосередній батьківський клас) та множинне (кілька батьківських класів).

Інкапсуляція. Приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення. Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт представляє кожному іншому класу певний інтерфейс — члени, доступні іншим класам. Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дозволить полегшити внесення змін, тобто, без потреби змінювати і користувачів інтерфейсу. Часто, члени класу позначаються як публічні (англ. public), захищені (англ. protected) та приватні (англ.private), визначаючи, чи доступні вони всім класам, підкласам, або лише до класу в якому їх визначено.

Поліморфізм. Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. На практиці - це реалізовується шляхом реалізації ряду підпрограм (функцій, процедур, методів тощо) з однаковими іменами, але з різними параметрами. В залежності від того, що передається і вибирається відповідна підпрограма.

Нижче наведено основні принципи ООП. Вони є дуже важливими, адже дозволяють захищати дані в класі, дозволяють брати необхідну інформацію одним класам у інших. Це робить написання коду набагато зручнішим і ефективнішим. Тому я вважаю було доцільним використовувати ці принци під час написання програми.

2.АНАЛІЗ ТА РОЗРОБКА АЛГОРИТМУ ЗГІДНО ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

2.1.Опис алгоритму та його складності.

Алгоритм розпочинається із основної конструкції switch, що утворює меню консольної програми. Це розгалуження відповідає за головне меню програми, яке має такі можливості:

А) Подивитися інформацію про казино.

Б) Подивитися список персоналу казино.

В) Обміняти гроші на ігрові фішки або навпаки.

Г) Грати ігри.

Д) Вийти з казино.

Якщо користувач вибере подивитися інформацію про казино то далі буде ще одна конструкція switch, яка відповідає за подальші дії програми. Це розгалуження має такі можливості:

А) Подивитися повну інформацію про казино

Б) Подивитися часткову інформацію про казино

В) Повернутися в головне меню

Обравши повну інформацію користувачу виводиться на екран вся доступна інформація про казино. Проте, якщо користувачу не потрібна вся інформація, а він хоче подивитися лише деяку її частину(наприклад йому потрібно знати місце знаходження казино), то він може обрати інший варіант. При виборі часткової інформації спрацьовує ще одна конструкція switch. Вона має такі можливості:

А) Подивитися інформацію про місце знаходження казино

Б) Подивитися інформацію про заснування казино

В) Подивитися інформацію про власника казино

Г) Повернутися назад

Після перегляду усієї необхідної інформації користувач може повернутися до головного меню, звідки може перейти до наступного пункту або вийти з казино.

Якщо ж користувач у головному меню обирає «Подивитися список персоналу казино», то спрацьовує наступна конструкція switch, за допомогою якої можна обрати наступні дії:

А) Подивитися весь список персоналу

Б) Подивитися список персоналу якоїсь певної професії

В) Повернутися у головне меню

Обравши варіант А користувачу виводиться на екран список усього персоналу та їхні професії. Якщо ж буде обраний варіант Б то спрацює ще одна конструкція switch, за допомогою ми можемо обрати людей якої професії він хоче переглянути, а саме:

А) Список прибиральників

Б) Список охоронців

В) Список круп’є

Г) Список консультантів

Д) Список працівників банку казино

Е) Адміністратор казино

Подивившись усю необхідну інформацію користувач може повернутися у головне меню звідки може вийти з казино або обрати інший пункт.

Якщо користувач у головному меню обирає «Обміняти гроші на ігрові фішки або навпаки», то спрацьовує конструкція switch, за допомогою якої ми можемо обрати наступне:

А) Обміняти гроші на ігрові фішки

Б) Обміняти ігрові фішки на гроші

В) Повернутися у головне меню

Користувач проводить певні дії із своїми фішками та грошима і після цього може повернутися у головне меню.

Якщо у головному меню користувач обирає «Грати ігри», то спрацьовує конструкція switch, за допомогою якої ми можемо вибрати наступне:

А) Грати в рулетку

Б) Грати в карточну гру «Двадцять одне»

В) Грати в ігровий автомат

Д) Повернутися в головне меню

Відповідно до того яку гру вибере користувач, у ту він і буде грати. Він може програти певну суму, а може і виграти. Також він може у будь-який момент перейти з однієї гри на іншу.

Отже, алгоритм є достатньо складним. Використовується велика кількість конструкцій switch, які забезпечують зручність використання програми.

2.2. Аналіз особливостей алгоритму.

У кожній конструкції switch викликаються методи відповідних класів.

А) У першій конструкції використовуються функції: Full\_information() та Particle\_information().

Б) У другому випадку використовуються функції: cleaners(), security(), crupye(), consultants(), bankers(), administrator().

В) У третьому випадку використовуються функції: refill() та withdrow().

Г) У четвертому випадку відкриваються ті ігри, які обрав користувач.

2. 2. Блок-схема алгоритму

Відкрити ігровий автомат

Відкрити «Двадцять одне»

Вивід часткових списків

Вивід повного списку

Particle\_information()

Full\_information()

switch

switch

switch

Ввід choice4

Ввід choice3

Ввід choice2

Ввід choice11

Switch

(choice))

Кінець

Відкрити рулетку

Withdrow()

Refill()

switch

Ввід choice

Початок

3.ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ЗГІДНО ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

3.1. Опис програмної реалізації алгоритму.

3.1.1. Опис структур даних програми.

В програмі наявні наступні типи даних:

А)Змінні цілого типу для виконання конструкції switch, та виконання операцій пов’язаних з іграми.

Б)Змінні символьного типу які використовувалися у грі «Двадцять одне», за допомогою яких, я зробив гральні карти.

В) Статична змінна tockens, яка використовується для підрахунку ігрових фішок користувача і яка є доступна у всіх іграх та у банку казино.

Г) Об’єкти класів для викликання методів класів.

3.1.2 Порядок викликів функцій

А) Отримання першого вибору

Б) Перехід до наступного вибору в залежності від першого вибору

Можливі виконання функції:

А) Вивід на екран певної інформації

Б) Ввід необхідних даних

В) Зменшення або збільшення кількості ігрових фішок

Г) Поповнення кількості фішок або їх обмін на гроші

Д) Функції пов’язані з іграми

Усі функції знаходяться у класах які побудовані наступним чином: існує незалежний клас CCassino, який має функції привітання та показу інформації про казино. Клас CStaff також не залежить від жодного іншого класу. Він містить в собі функції показу списку працівників казино. Також є створений клас CBank, який має функції переведення грошей у ігрові фішки та обмін ігрових фішок на гроші. Від цього класу походить три класи які у свою чергу є іграми. Тобто від класу CBank походять класи CRulet, CTwentyOne, CAutomat.

Діаграма класів наведена нижче

CStaff

CCassino CTwentyOne

CAutomat

CRulet

CBank

3.1. Спрощений вигляд діаграми класів

3.1.3. Опис реалізаціх основних функцій/методів

Кожен клас реалізований за допомого розбиття на два файли: header та .cpp, де header зберігає в собі оголошення змінних класу та методів, а .cpp вони реалізуються.

А) Функція виводу повної інформації

void CCasino :: Full\_information()

{

cout << "Casino " << name << " is located in " << location << " on the " << streat << ". Casino was foundet on " << date << ". The owner is " << owner << "." << endl;

}

Б) Функції поповнення та зняття коштів

void CBank::refill()

{

int sum\_user;

start:

cout << "Enter how much money do you have:";

cin >> sum\_user;

if (sum\_user <= 0)

{

cout << "You enter uncorrect sum";

goto start;

}

else

{

user\_sum = sum\_user;

int sum;

start\_1:

cout << "Enter the sum in which you want refill your account: ";

cin >> sum;

if (sum <= 0 || sum > 10000)

{

cout << "You enter uncorrect sum";

goto start\_1;

}

else

{

user\_sum -= sum;

tockens += sum;

bank\_casino += sum;

}

}

}

void CBank :: withdrow()

{

int sum;

start:

cout << "Enter the sum of tockens which you want to exchange in to the money";

cin >> sum;

if (sum <= 0 || sum > tockens )

{

cout << "You enter uncorrect sum";

goto start;

}

else

{

tockens -= sum;

bank\_casino -= sum;

user\_sum += sum;

}

}

В) Гра «Рулетка»

void CRulet :: random()

{

srand(time(NULL));

num[0] = rand() % 37;

if (num[0] > 0)

{

if (num[0] % 2 != 0)

{

colour = "red";

}

else

{

colour = "black";

}

cout << num[0] << " " << colour << endl;

}

if (num[0] == 0)

{

cout << "ZERO" << endl;

}

}

void CRulet :: bet\_1()

{

if (tockens > 0)

{

cout << "Enter 1 if you want to bet on colour" << endl;

cout << "Enter 2 if you want to bet on pair or unpair number" << endl;

cout << "Enter 3 if you want to bet on numbers from 1 to 18" << endl;

cout << "Enter 4 if you want to bet on numbers from 19 to 36" << endl;

cout << "Enter 5 if you want to bet on 1 number" << endl;

cout << "Enter 6 if you want to bet on 2 numbers" << endl;

cout << "Enter 7 if you want to bet on 3 numbers" << endl;

cout << "Enter 8 if you want to bet on 4 numbers" << endl;

cout << "Enter 9 if you want to bet on 6 numbers" << endl;

cin >> choice;

switch (choice)

{

case 1:

{

cout << "Enter in which colour you want no bet: " << endl;

cout << "Enter 1 if you want to bet on red" << endl;

cout << "Enter 2 if you want to bet on black" << endl;

cin >> choice\_1;

if (choice\_1 != 1 && choice\_1 != 2)

{

cout << "Invalid input" << endl;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

}

break;

}

case 2:

{

cout << "Enter in which numbers your want to bet" << endl;

cout << "Enter 1 if you want to bet on pair number" << endl;

cout << "Enter 2 if you want to bet on unpair number" << endl;

cin >> choice\_3;

if (choice\_3 != 1 && choice\_3 != 2)

{

cout << "Invalid input" << endl;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

}

break;

}

case 3:

{

cout << "Enter sum of your bet: " << endl;

cin >> bet;

break;

}

case 4:

{

cout << "Enter sum of your bet: " << endl;

cin >> bet;

break;

}

case 5:

{

start:

cout << "Enter number in which you want to bet: ";

cin >> num[1];

if (num[1] > 36)

{

cout << "Invalid input" << endl;

goto start;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

break;

}

}

case 6:

{

start\_1:

cout << "Enter first number in which you want to bet: ";

cin >> num[1];

cout << "Enter second number in which you want to bet: ";

cin >> num[2];

if (num[1] > 36 || num[2] > 36)

{

cout << "Invalid input" << endl;

goto start\_1;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

break;

}

}

case 7:

{

start\_2:

cout << "Enter first number in which you want to bet: ";

cin >> num[1];

cout << "Enter second number in which you want to bet: ";

cin >> num[2];

cout << "Enter third number in which you want to bet: ";

cin >> num[3];

if (num[1] > 36 || num[2] > 36 || num[3] > 36)

{

cout << "Invalid input" << endl;

goto start\_2;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

break;

}

}

case 8:

{

start\_3:

cout << "Enter first number in which you want to bet: ";

cin >> num[1];

cout << "Enter second number in which you want to bet: ";

cin >> num[2];

cout << "Enter third number in which you want to bet: ";

cin >> num[3];

cout << "Enter forth number in which you want to bet: ";

cin >> num[4];

if (num[1] > 36 || num[2] > 36 || num[3] > 36 || num[4] > 36)

{

cout << "Invalid input" << endl;

goto start\_3;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

break;

}

}

case 9:

{

start\_4:

cout << "Enter first number in which you want to bet: ";

cin >> num[1];

cout << "Enter second number in which you want to bet: ";

cin >> num[2];

cout << "Enter third number in which you want to bet: ";

cin >> num[3];

cout << "Enter forth number in which you want to bet: ";

cin >> num[4];

cout << "Enter fifth number in which you want to bet: ";

cin >> num[5];

cout << "Entre sixth number in which you want to bet: ";

cin >> num[6];

if (num[1] > 36 || num[2] > 36 || num[3] > 36 || num[4] > 36 || num[5] > 36 || num[6] > 36)

{

cout << "Invalid input" << endl;

goto start\_4;

}

else

{

cout << "Enter sum of your bet: ";

cin >> bet;

break;

}

}

default:

{

cout << "Invalid input" << endl;

break;

}

}

}

}

void CRulet :: colours()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

if (choice\_1 == 1 || choice\_1 == 2)

{

int coef = 2;

tockens -= bet;

bank\_casino += bet;

if (choice\_1 == 1)

{

if (num[0] % 2 != 0)

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens <<" tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

else

{

if (num[0] % 2 == 0)

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your account = " << tockens << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

else

cout << "Invalid input" << endl;

}

}

void CRulet :: numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

if (choice\_3 == 1 || choice\_3 == 2)

{

int coef = 2;

tockens -= bet;

bank\_casino += bet;

if (choice\_3 == 1)

{

if (num[0] % 2 == 0)

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

else

{

if (num[0] % 2 != 0)

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

else

cout << "Invalid input" << endl;

}

}

void CRulet ::lower\_numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 2;

tockens -= bet;

bank\_casino += bet;

if (num[0] > 0 && num[0] < 19)

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

void CRulet :: biger\_numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 2;

tockens -= bet;

bank\_casino += bet;

if (num[0] > 18 && num[0] < 37)

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

void CRulet :: one\_number()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 36;

tockens -= bet;

bank\_casino += bet;

if (num[0] == num[1])

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

void CRulet :: two\_numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 18;

tockens -= bet;

bank\_casino += bet;

if (num[0] == num[1] || num[0] == num[2])

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

void CRulet :: threa\_numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 12;

tockens -= bet;

bank\_casino += bet;

if (num[0] == num[1] || num[0] == num[2] || num[0] == num[3])

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

void CRulet :: four\_numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 9;

tockens -= bet;

bank\_casino += bet;

if (num[0] == num[1] || num[0] == num[2] || num[0] == num[3] || num[0] == num[4])

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

void CRulet :: six\_numbers()

{

if (bet > tockens)

{

cout << "Invalid input" << endl;

}

else

{

int coef = 6;

tockens -= bet;

bank\_casino += bet;

if (num[0] == num[1] || num[0] == num[2] || num[0] == num[3] || num[0] == num[4] || num[0] == num[5] || num[0] == num[6])

{

cout << "You win!!!" << endl;

bet \*= coef;

tockens += bet;

bank\_casino -= bet;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

}

}

Г) Гра «Двадцять одне»

void CTwentyOne :: twocards()

{

srand(time(NULL));

card\_1 = 1 + rand() % 13;

card\_2 = 1 + rand() % 13;

cout << "You have this cards: ";

if (card\_1 == 1)

{

card\_mass = 2;

cout << cards[0] << " ";

}

else if (card\_1 == 2)

{

card\_mass = 3;

cout << cards[1] << " ";

}

else if (card\_1 == 3)

{

card\_mass = 4;

cout << cards[2] << " ";

}

else if (card\_1 == 4)

{

card\_mass = 5;

cout << cards[3] << " ";

}

else if (card\_1 == 5)

{

card\_mass = 6;

cout << cards[4] << " ";

}

else if (card\_1 == 6)

{

card\_mass = 7;

cout << cards[5] << " ";

}

else if (card\_1 == 7)

{

card\_mass = 8;

cout << cards[6] << " ";

}

else if (card\_1 == 8)

{

card\_mass = 9;

cout << cards[7] << " ";

}

else if (card\_1 == 9)

{

card\_mass = 10;

cout << cards[8] << " ";

}

else if (card\_1 == 10)

{

card\_mass = 11;

cout << cards\_1[0] << " ";

}

else if (card\_1 == 11)

{

card\_mass = 12;

cout << cards\_1[1] << " ";

}

else if (card\_1 == 12)

{

card\_mass = 13;

cout << cards\_1[2] << " ";

}

else if (card\_1 == 13)

{

card\_mass = 14;

cout << cards\_1[3] << " ";

}

if (card\_2 == 1)

{

card\_mass\_2 = 2;

cout << cards[0] << endl;

}

else if (card\_2 == 2)

{

card\_mass\_2 = 3;

cout << cards[1] << endl;

}

else if (card\_2 == 3)

{

card\_mass\_2 = 4;

cout << cards[2] << endl;

}

else if (card\_2 == 4)

{

card\_mass\_2 = 5;

cout << cards[3] << endl;

}

else if (card\_2 == 5)

{

card\_mass\_2 = 6;

cout << cards[4] << endl;

}

else if (card\_2 == 6)

{

card\_mass\_2 = 7;

cout << cards[5] << endl;

}

else if (card\_2 == 7)

{

card\_mass\_2 = 8;

cout << cards[6] << endl;

}

else if (card\_2 == 8)

{

card\_mass\_2 = 9;

cout << cards[7] << endl;

}

else if (card\_2 == 9)

{

card\_mass\_2 = 10;

cout << cards[8] << endl;

}

else if (card\_2 == 10)

{

card\_mass\_2 = 11;

cout << cards\_1[0] << endl;

}

else if (card\_2 == 11)

{

card\_mass\_2 = 12;

cout << cards\_1[1] << endl;

}

else if (card\_2 == 12)

{

card\_mass\_2 = 13;

cout << cards\_1[2] << endl;

}

else if (card\_2 == 13)

{

card\_mass\_2 = 14;

cout << cards\_1[3] << endl;

}

cout << "You have " << card\_mass + card\_mass\_2 << " points" << endl;

}

int CTwentyOne::gaming()

{

srand(time(NULL));

int sum = card\_mass + card\_mass\_2;

int choice;

Start:

cout << "Do you want to take one more card?" << endl;

cout << "Enter 1 if you want" << endl;

cout << "Enter 2 if you dont want" << endl;

cin >> choice;

if (choice != 1 && choice != 2)

{

cout << "Invalid input" << endl;

goto Start;

}

while (choice == 1)

{

card\_1 = 1 + rand() % 13;

if (card\_1 == 1)

{

cout << "You earn card " << cards[0] << endl;

card\_mass = 2;

}

else if (card\_1 == 2)

{

card\_mass = 3;

cout << "You earn card " << cards[1] << endl;

}

else if (card\_1 == 3)

{

card\_mass = 4;

cout << "You earn card " << cards[2] << endl;

}

else if (card\_1 == 4)

{

card\_mass = 5;

cout << "You earn card " << cards[3] << endl;

}

else if (card\_1 == 5)

{

card\_mass = 6;

cout << "You earn card " << cards[4] << endl;

}

else if (card\_1 == 6)

{

card\_mass = 7;

cout << "You earn card " << cards[5] << endl;

}

else if (card\_1 == 7)

{

card\_mass = 8;

cout << "You earn card " << cards[6] << endl;

}

else if (card\_1 == 8)

{

card\_mass = 9;

cout << "You earn card " << cards[7] << endl;

}

else if (card\_1 == 9)

{

card\_mass = 10;

cout << "You earn card " << cards[8] << endl;

}

else if (card\_1 == 10)

{

card\_mass = 11;

cout << "You earn card " << cards\_1[0] << endl;

}

else if (card\_1 == 11)

{

card\_mass = 12;

cout << "You earn card " << cards\_1[1] << endl;

}

else if (card\_1 == 12)

{

card\_mass = 13;

cout << "You earn card " << cards\_1[2] << endl;

}

else if (card\_1 == 13)

{

card\_mass = 14;

cout << "You earn card " << cards\_1[3] << endl;

}

sum += card\_mass;

cout << "You have " << sum << " points" << endl;

Start\_1:

cout << "Do you want to take one more card?" << endl;

cout << "Enter 1 if you want" << endl;

cout << "Enter 2 if you dont want" << endl;

cin >> choice;

if (choice != 1 && choice != 2)

{

cout << "Invalid input" << endl;

goto Start\_1;

}

}

return sum;

}

void CTwentyOne ::bot\_twocards()

{

srand(time(NULL));

card\_1 = 1 + rand() % 13;

card\_2 = 1 + rand() % 13;

cout << "Bot have: ";

if (card\_1 == 1)

{

card\_mass = 2;

cout << cards[0] << " ";

}

else if (card\_1 == 2)

{

card\_mass = 3;

cout << cards[1] << " ";

}

else if (card\_1 == 3)

{

card\_mass = 4;

cout << cards[2] << " ";

}

else if (card\_1 == 4)

{

card\_mass = 5;

cout << cards[3] << " ";

}

else if (card\_1 == 5)

{

card\_mass = 6;

cout << cards[4] << " ";

}

else if (card\_1 == 6)

{

card\_mass = 7;

cout << cards[5] << " ";

}

else if (card\_1 == 7)

{

card\_mass = 8;

cout << cards[6] << " ";

}

else if (card\_1 == 8)

{

card\_mass = 9;

cout << cards[7] << " ";

}

else if (card\_1 == 9)

{

card\_mass = 10;

cout << cards[8] << " ";

}

else if (card\_1 == 10)

{

card\_mass = 11;

cout << cards\_1[0] << " ";

}

else if (card\_1 == 11)

{

card\_mass = 12;

cout << cards\_1[1] << " ";

}

else if (card\_1 == 12)

{

card\_mass = 13;

cout << cards\_1[2] << " ";

}

else if (card\_1 == 13)

{

card\_mass = 14;

cout << cards\_1[3] << " ";

}

if (card\_2 == 1)

{

card\_mass\_2 = 2;

cout << cards[0] << " ";

}

else if (card\_2 == 2)

{

card\_mass\_2 = 3;

cout << cards[1] << " ";

}

else if (card\_2 == 3)

{

card\_mass\_2 = 4;

cout << cards[2] << " ";

}

else if (card\_2 == 4)

{

card\_mass\_2 = 5;

cout << cards[3] << " ";

}

else if (card\_2 == 5)

{

card\_mass\_2 = 6;

cout << cards[4] << " ";

}

else if (card\_2 == 6)

{

card\_mass\_2 = 7;

cout << cards[5] << " ";

}

else if (card\_2 == 7)

{

card\_mass\_2 = 8;

cout << cards[6] << " ";

}

else if (card\_2 == 8)

{

card\_mass\_2 = 9;

cout << cards[7] << " ";

}

else if (card\_2 == 9)

{

card\_mass\_2 = 10;

cout << cards[8] << " ";

}

else if (card\_2 == 10)

{

card\_mass\_2 = 11;

cout << cards\_1[0] << " ";

}

else if (card\_2 == 11)

{

card\_mass\_2 = 12;

cout << cards\_1[1] << " ";

}

else if (card\_2 == 12)

{

card\_mass\_2 = 13;

cout << cards\_1[2] << " ";

}

else if (card\_2 == 13)

{

card\_mass\_2 = 14;

cout << cards\_1[3] << " ";

}

}

int CTwentyOne :: bot\_gaming()

{

srand(time(NULL));

int sum = card\_mass + card\_mass\_2;

while(sum < 17)

{

card\_1 = 1 + rand() % 13;

if (card\_1 == 1)

{

card\_mass = 2;

cout << cards[0] << " ";

}

else if (card\_1 == 2)

{

card\_mass = 3;

cout << cards[1] << " ";

}

else if (card\_1 == 3)

{

card\_mass = 4;

cout << cards[2] << " ";

}

else if (card\_1 == 4)

{

card\_mass = 5;

cout << cards[3] << " ";

}

else if (card\_1 == 5)

{

card\_mass = 6;

cout << cards[4] << " ";

}

else if (card\_1 == 6)

{

card\_mass = 7;

cout << cards[5] << " ";

}

else if (card\_1 == 7)

{

card\_mass = 8;

cout << cards[6] << " ";

}

else if (card\_1 == 8)

{

card\_mass = 9;

cout << cards[7] << " ";

}

else if (card\_1 == 9)

{

card\_mass = 10;

cout << cards[8] << " ";

}

else if (card\_1 == 10)

{

card\_mass = 11;

cout << cards\_1[0] << " ";

}

else if (card\_1 == 11)

{

card\_mass = 12;

cout << cards\_1[1] << " ";

}

else if (card\_1 == 12)

{

card\_mass = 13;

cout << cards\_1[2] << " ";

}

else if (card\_1 == 13)

{

card\_mass = 14;

cout << cards\_1[3] << " ";

}

sum += card\_mass;

}

cout << endl;

cout << "Bot have " << sum << " points" << endl;

return sum;

}

void CTwentyOne :: resault\_of\_game()

{

int bet;

Start:

cout << "Enter sum of your bet: ";

cin >> bet;

int sum\_1, sum\_2;

if (bet > tockens || bet <= 0 )

{

cout << "Invalid input" << endl;

goto Start;

}

else

{

int coef = 2;

tockens -= bet;

twocards();

sum\_1 = gaming();

bot\_twocards();

sum\_2 = bot\_gaming();

int resault\_1, resault\_2;

resault\_1 = 21 - sum\_1;

resault\_2 = 21 - sum\_2;

if(abs(resault\_1)<abs(resault\_2))

{

tockens += coef \* bet;

cout << "You win. Now you have " << tockens << " tockens" << endl;

}

else if(abs(resault\_1) > abs(resault\_2))

{

cout << "You lose. Now you have " << tockens << " tockens" << endl;

}

else if (abs(resault\_1) == abs(resault\_2))

{

if (sum\_1 < sum\_2)

{

tockens += coef \* bet;

cout << "You win. Now you have " << tockens << " tockens" << endl;

}

else if(sum\_1 < sum\_2)

{

cout << "You lose. Now you have " << tockens << " tockens" << endl;

}

else

{

tockens += bet;

cout << "Drow. Now you have " << tockens << " tockens" << endl;

}

}

}

start:

cout << "Enter 1 if you want to continue" << endl;

cout << "Enter 2 if you want to exit" << endl;

cin >> choice;

if (choice != 1 && choice != 2)

{

cout << "Invalid input";

goto start;

}

else if (choice == 1)

goto Start;

}

int CTwentyOne :: get\_choice()

{

if (choice == 2)

return 1;

if (choice != 2)

return 2;

}

Д) Гра «Ігровий автомат»

int CAutomat::generator()

{

srand(time(NULL));

int num;

for (int i = 0; i < 3; i++)

{

num = rand() % 101;

if (num >= 0 && num < 50)

{

resault[i] = slots[0];

}

else if (num >= 50 && num < 71)

{

resault[i] = slots[1];

}

else if (num > 70 && num < 81)

{

resault[i] = slots[2];

}

else if (num > 80 && num < 91)

{

resault[i] = slots[3];

}

else if (num > 90 && num < 98)

{

resault[i] = slots[4];

}

else

{

resault[i] = slots[5];

}

cout << resault[i] << " ";

}

int coeficientIndex = 0;

if (resault[0] == resault[1] && resault[1] == resault[2])

{

for (int i = 0; i < 6; i++)

{

if (resault[0] == slots[i])

{

coeficientIndex = i;

}

}

switch(coeficientIndex)

{

case 0:

{

return 8;

break;

}

case 1:

{

return 125;

break;

}

case 2:

{

return 1000;

break;

}

case 3:

{

return 1000;

break;

}

case 4:

{

return 3000;

break;

}

case 5:

{

return 40000;

break;

}

default:

{

break;

}

}

}

else

{

return 0;

}

cout << endl;

}

void CAutomat::beting()

{

int bet;

start:

cout << "Enter 1 if you want to continue" << endl;

cout << "Enter 2 if you want to exit" << endl;

cin >> choice;

switch (choice)

{

case 1:

{

Start:

cout << "Enter sum of your bet: ";

cin >> bet;

if (bet > tockens)

{

cout << "Invalid input";

goto Start;

}

else

{

tockens -= bet;

bet \*= generator();

if (bet == 0)

{

cout << "You lose" << endl;

cout << "Your have " << tockens << " tockens" << endl;

}

else

{

cout << "You win!!! " << endl;

tockens += bet;

cout << "Your have " << tockens << " tockens" << endl;

}

}

break;

}

case 2:

{

break;

}

default :

{

cout << "Invalid input" << endl;

goto start;

break;

}

}

}

int CAutomat::get\_achoice()

{

if (choice == 2)

return 1;

if (choice != 2)

return 2;

}

4. ВІДЛАГОЖДЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМИ ЗГІДНО З ІНДИВІДУАЛЬНИМ ЗАВДАННЯМ

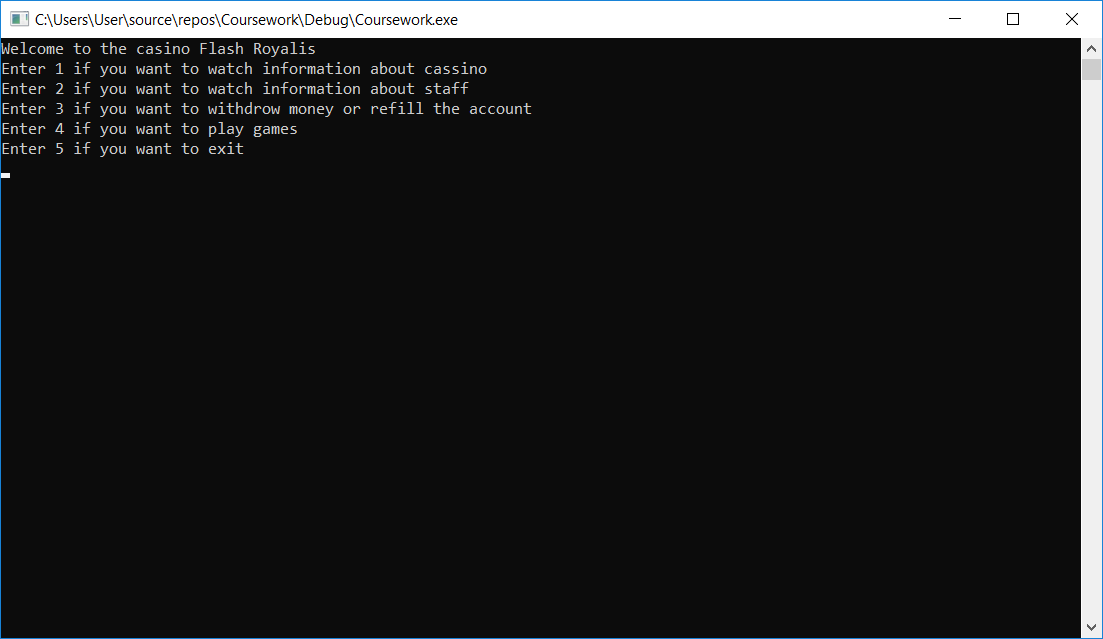
4.1.Відлагодження та тестування певних частин програмою.

Під час тестування окремих частин програм з’являлись певні недоліки та помилки. Щоб все правильно працювало потрібно було не дати можливості користувачу змушувати програму некоректно працювати. Тож всі можливі неполадки були ізольовані і опрацьовані так, щоб з’являлось певне повідомлення з певною інформацією, звідки користувач дізнавався як слід користуватися програмою, щоб уникати таких невизначеностей.

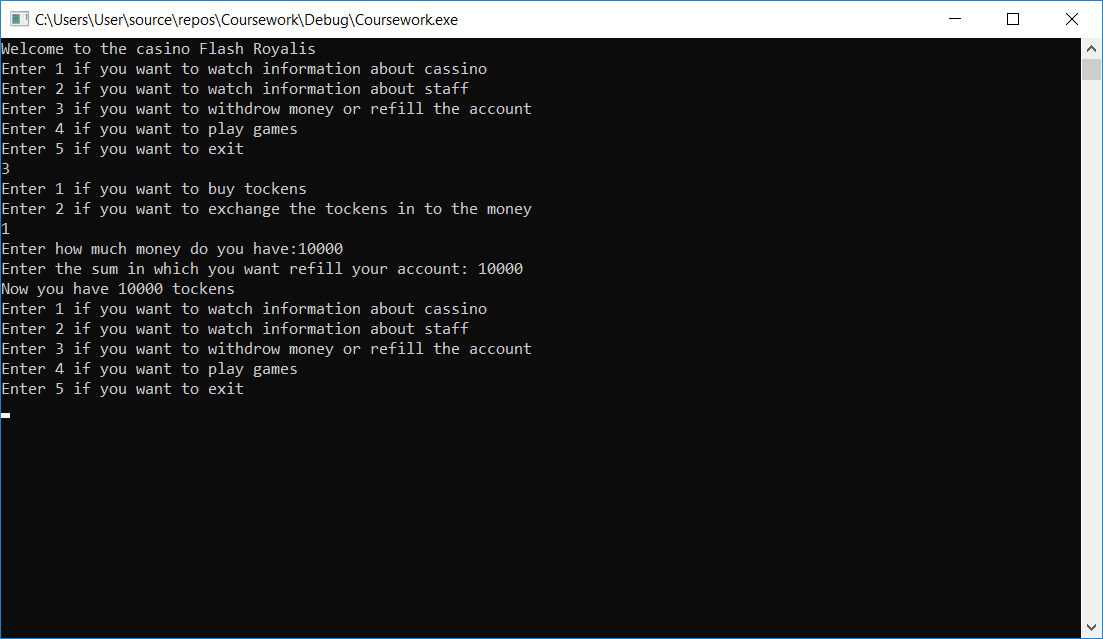
4.1. Відлагождення та тестування програми вцілому.

Подібні кроки були виконані і з перевіркою всієї програми. Нижче наведені скріншоти, які запевнюють в коректній роботі програми:

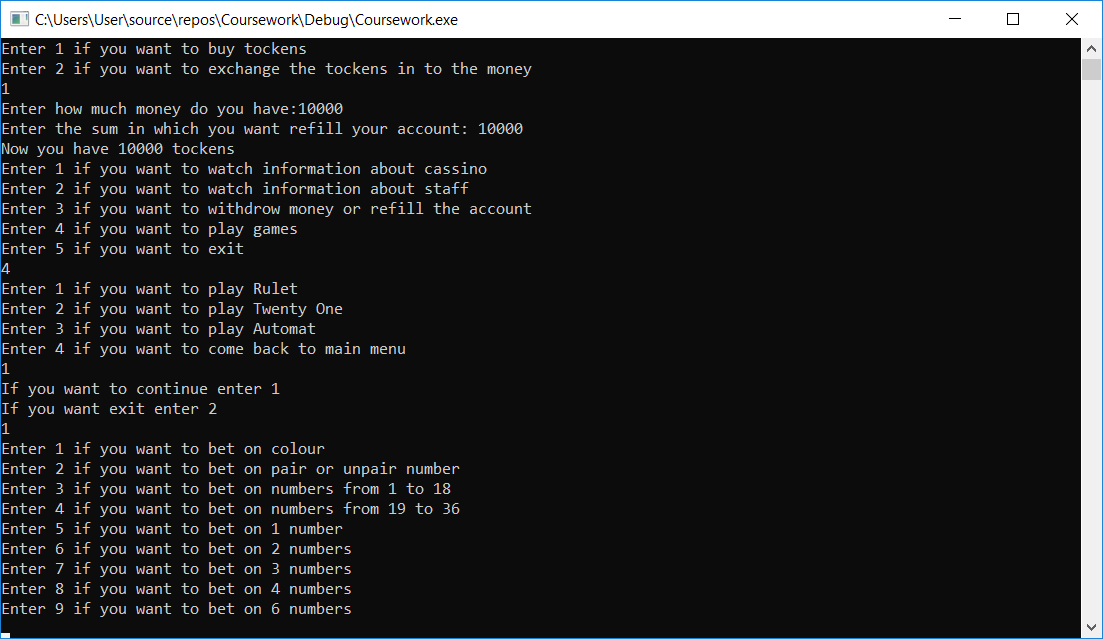
1. Головне меню



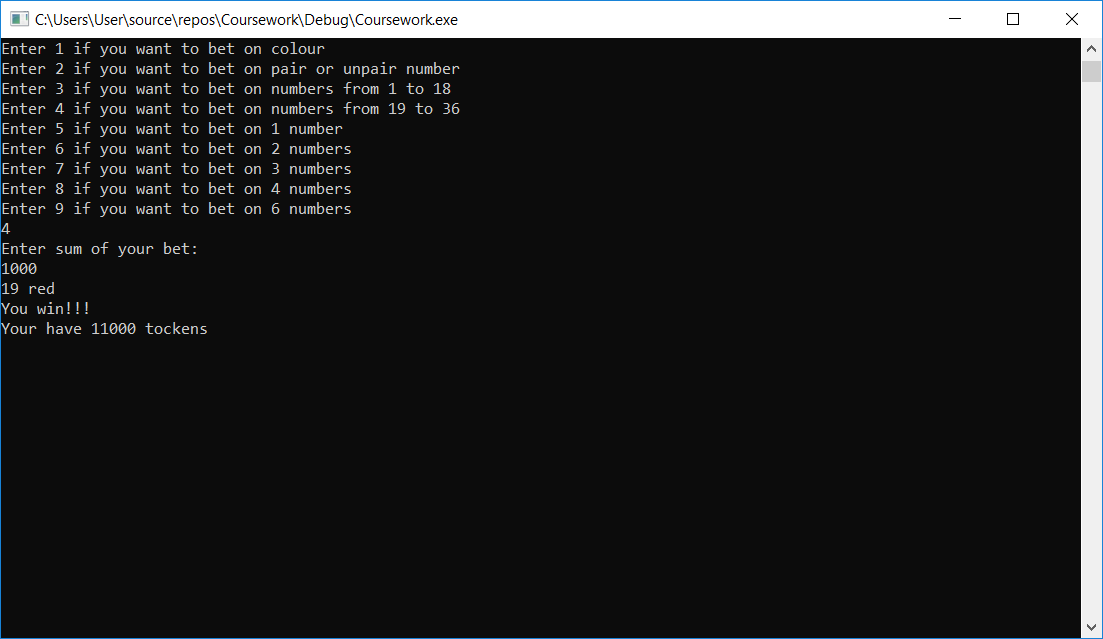
1. Обмін грошей на фішки



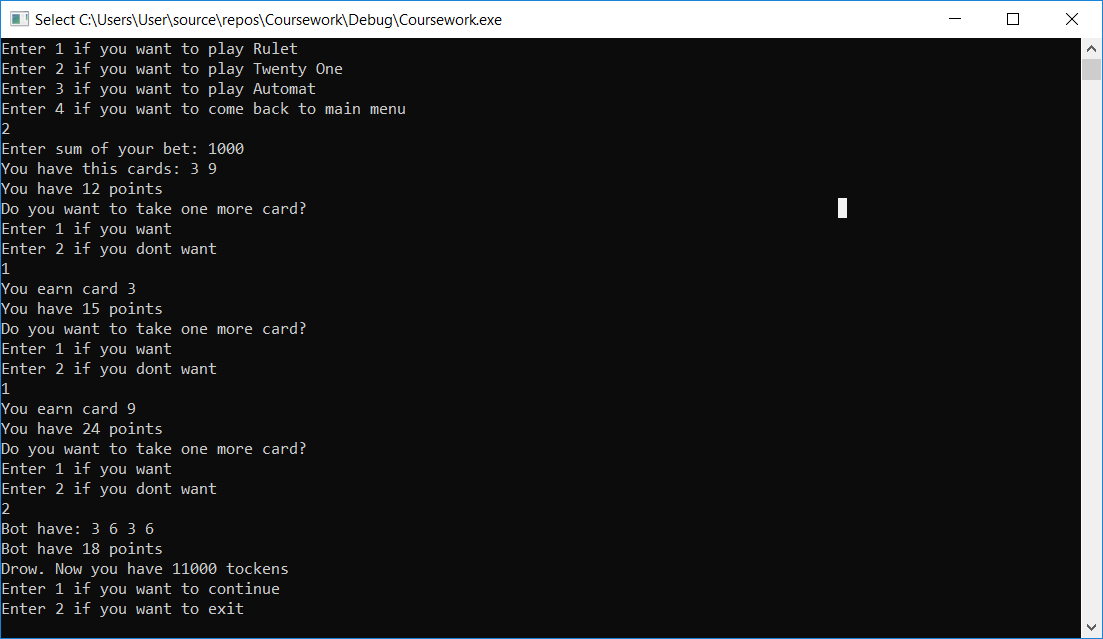
1. Перехід до Рулетки



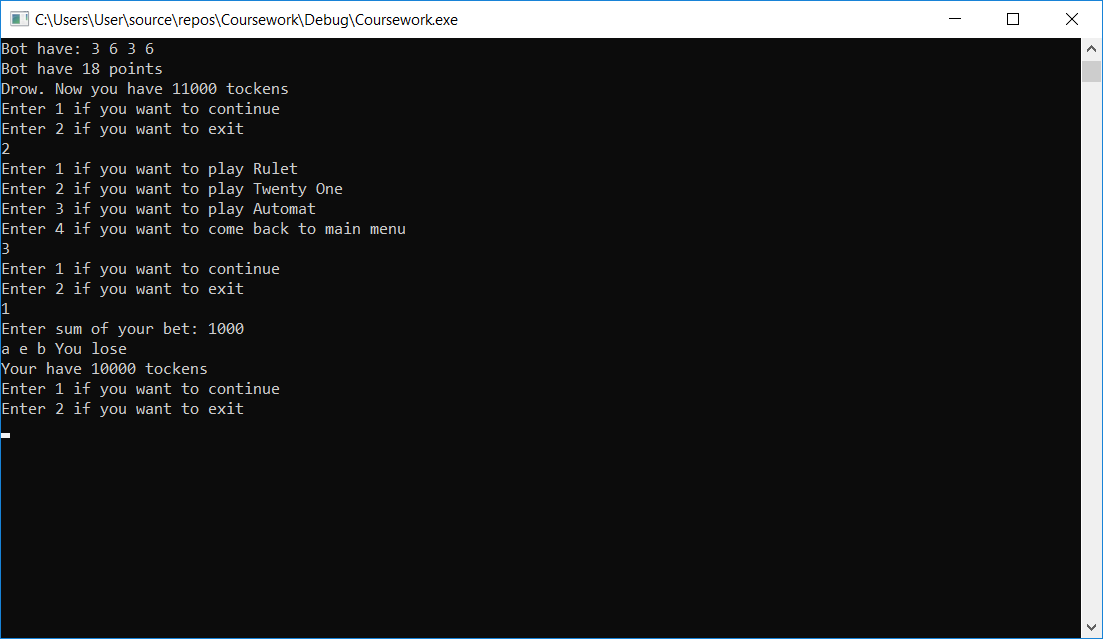
1. Результат гри в Рулетку



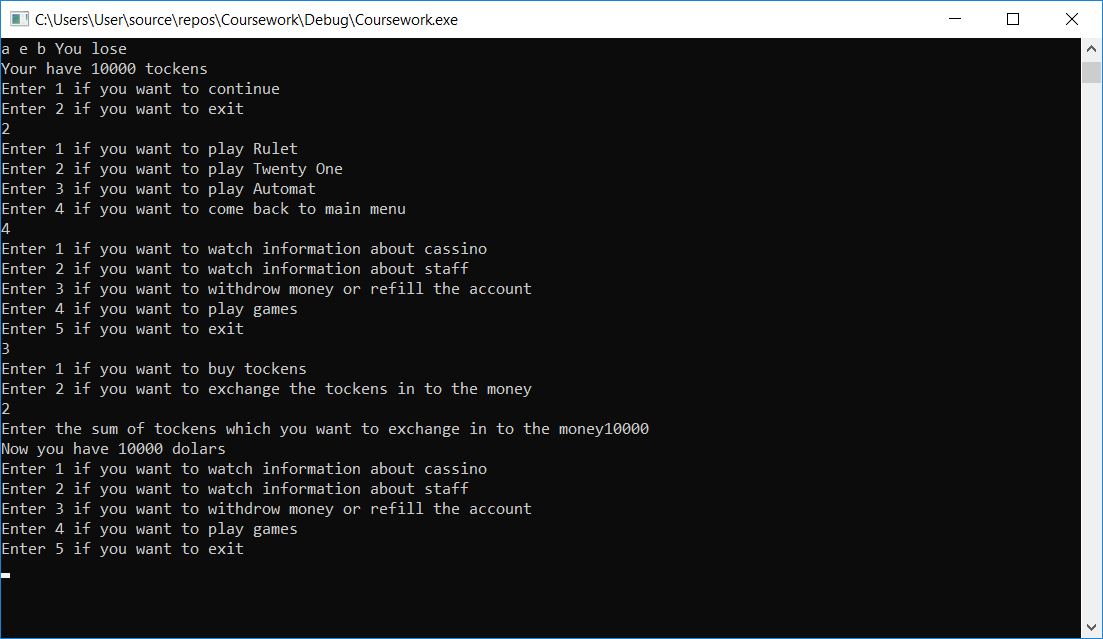
1. Перехід до Двадцять одного та результат гри



1. Перехід до Ігрового автомата та результат гри



1. Зняття коштів



5. АНАЛІЗ ПРОДУКТИВНОСТІ(ЧАСУ РОБОТИ) ТА СТРУКТУРНОЇ СКЛАДНОСТІ РЕАЛІЗОВАНОЇ ПРОГРАМИ ЗГІДНО ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

5.1. Аналіз структурної складності виконання програми.

Оцінюючи структуру програми, потрібно наголосити, що наявні конструкції switch, полегшують не тільки користування програмою в ролі користувача, але як на мене, полегшують власне кодування. Ці конструкції начебто уявно відділяють блок коду від усієї програми, що покращує сприйняття й розуміння виконаних та запланованих цілей (функцій, операцій). Як висновок, можу сказати що, виконання програми на структурному рівні не є складним.

5.2. Всі операції всередині програми виконуються миттєво, немає затримки між переходами по меню та виведенням інформації на консоль.

ВИСНОВОК

При підготовці куросвої роботи я навчився багатьом новим речам. Навчився планувати майбутні дії, маючи якесь певне завдання та створювати блок-схему, яка допомфгає при написанні коду. Навчився, за допомогою конструкції switch робити головне меню програм та зрозумів її корисність у програмувані. Також оволодів базовими елементами об’єктно-орієнтованого програмування.

Загалом мені сподабався процес розробки казино. Мені здається, у мене вийшло зробити заклад, у якому люди моглиб відпочити і випробувати свою вдачу.

СПИСОК ЛІТЕРАТУРИ

1. Прата С. Язык программирования C++. Лекции и упражнения. 6-е издание / Стивен Прата; пер. с англ. – М.: ООО «И.Д. Вильямс», 2012. – 1248 с.: ил. – Парал. тит. англ.

2. Полный справочник по C++, 4-е издание / ГербертШилдт — М.: «Вильямс», 2011. - 800 с.

3. Chris Giametta «Pro Flex on Spring», 2009.- p.445.

4. Роберт Дж. Оберг «ТехнологияCOM + Основы и программирование = Understanding and ProgrammingCOM+: A Practical Guideto Windows 2000 First Edition». — М.: «Вильямс», 2000.- С. 480.

5. Тамре Л. «Введение в тестирование програмного обеспечения ».- М.: Издательский дом «Вильямс», 2003.- С. 368.

6. Липаев В.В. «Обеспечение качества программных средств. Методы и стандарты». - М.: Синтег, 2001.- С. 246.

7. Макгрегор Дж., Сайкс Д. «Тестирование объектно-ориентированного програмного обеспечения».- К: Диасофт, 2002.- С. 432.

8. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft/COM и Java/RMI. Пер. с англ. — М.: Мир, 2002. — С. 510.

9. Макгрегор Дж., Сайкс Д. Тестирование объектно-ориентированного програмного обеспечения.- К: Диасофт, 2002.- С. 432.

10. Тамре Л. Введение в тестирование програмного обеспечения.- М.: Издательский дом―Вильямс‖, 2003.- С. 368.

11. Э. Гамм, Р. Хелм, Р. Джонсон, Д. Влиссидес «Приѐмы объектно-ориентированного проектирования. Паттерны проектирования», Питер, Москва, 2007.- С. 366.

12. Джон Влиссидес «Применение шаблонов проектирования», Москва, 2003.- С. 130.