Гра Блекджек. Блекджек - класична карткова гра на удачу та стратегію в казино.

1 ВИМОГИ

1.1 Розробник

- Хелемендик Дмитро Олегович;
- студент групи КН-921д;
- 17-липня-2022.

1.2 Загальне завдання

Реалізувати карткову гру Blackjack у консолі.

2 ОПИС ПРОГРАМИ

2.1 Функціональне призначення:

Програма призначена для виконання карткової гри Блекджек. В ній можуть приймати участь від 1 до 7 гравців. Програма працює за допомогою функцій, що задекларовані в hand.h, player.h, game.h, iostream, vector, ctime, cctype та algorithm.

Демонстрація знайдених результатів передбачає виконання програми у вікні консолі.

2.2 Опис логічної структури

За допомогою ключового слова *class* описую карту, що відображає ранг та масть карти. Розроблено клас, вміст якої подано нижче. Карта має два перерахування ранг та масть. А також наявні методи: конструктор, деструктор, отримання значення карти, перевертання карти та перегрузка оператора виводу.

Також описую руку, яка тримає карти та має такі методи: конструктор, деструктор, додавання карти в руку, очищення руки від карт, отримання суми очок карт в руці гравця.

Далі створюю абстрактний класс "загальний гравець", котрий успадковує параметри та методи класа рука, має власне ім'я та методи: конструктор,

деструктор, чи перебрав гравець(чисто віртуальна функція), оператор виводу та вивід гравця, котрий перебрав, на екран.

Тепер відтворюю гравця, який успадковує загального гравця та доповнює себе методами: конструктор, деструктор, чи бажає гравець взяти карту, перемога, поразка та нічия.

Також створюю дилера, спадкоємця загального гравця та представника казино. Було додано такі методи: конструктор, деструктор, чи бажає дилер взяти карту та перевертання першої карти.

Далі відображаю гральну колоду карт, яка є спадкоємцем класа рука. Цей об'єкт створює та тасує колоду, взаємодіє з гравцем та додає йому додаткову карту.

Нарешті створюю клас "гра", яка буде проводити гру. Має метод, котрий буде запускати ігровий процес.

Опис розроблених структур і функцій наводиться на базі результатів роботи системи автодокументування *Doxygen*.

Вступ

Action introduction();

Призначення: зрозуміти чого хоче гравець.

Опис роботи: функція вітає гравця та запитує його подальші дії. Повертає перерахування "дія" (START, RULES або END).

Правила гри

void showRulesOfGame();

Призначення: ознайомити гравця з правилами гри.

Опис роботи: функція друкує правила на екран.

Отримання імен гравців

void getNamesOfPlayers(vector<string> &names);

Призначення: отримати імена гравців для передачи їх в гру.

Опис роботи: функція запитує кількість гравців та їх імена.

Аргументи:

- *names* — вектор, в якому зберігаються імена всіх гравців.

Отримання значення карти

int getValue() const;

Клас : Card.

Призначення: отримати значення карти для рахування загальної суми очок карт.

Опис роботи: функція перевіряє чи відкрита карта, якщо так - повертає її значення.

Перевертання карти

void flipCard();

Клас : Card.

Призначення: сховати першу карту дилера.

Опис роботи: функція перевертає карту: карта, що лежить сорочкою вгору, перевертається вниз і навпаки.

Перегрузка оператора виводу

friend ostream &operator<<(ostream &os, const Card &card);

Клас : Card.

Призначення: вивід карти на екран.

Опис роботи: функція відправляє об'єкт типа Card в стандартний поток виводу. Якщо карта не схована - друкує її ранг та масть, інакше виводить "XX".

Аргументи:

- *os* — оператор виводу;
- *card* — константна посилання на об'єкт класа Card.

Додавання карти в руку

void addCard(Card *pCard);

Клас : Hand.

Призначення: додати об'єкт класа Card в руку.

Опис роботи: функція отримує карту pCard та додає її в вектор cards.

Аргументи:

- *pCard* — карта, об'єкт класа Card, яку потрібно додати в руку.

Очищення руки від карт

void clearHand();

Клас : Hand.

Призначення: звільнити зайняту пам'ять

Опис роботи: функція проходить по вектору, звільнює зайняту пам'ять. Далі звільнює вектор показників.

Отримання суми очок карт в руці гравця

int getTotal() const;

Клас : Hand.

Призначення: підрахувати загальну кількість очок карт в руці гравця.

Опис роботи: функція рахує кількість очок, привласнюючи тузу значення 1 або 11 залежно від ситуації. Повертає загальну кількість очок в руці гравця.

Чи перебрав гравець

bool isBusted() const;

Клас : GenericPlayer.

Призначення: дізнатися, чи був перебор у гравця.

Опис роботи: функція перевіряє чи загальна сума очок карт у руці більше 21. Повертає істину якщо гравець перебрав.

Вивід гравця, котрий перебрав

void playerBusts() const;

Клас : GenericPlayer.

Призначення: повідомити гравця про перебор.

Опис роботи: функція виводить ім'я гравця на екран та те, що він перебрав.

Оператор виводу

friend ostream &operator<<(ostream &os, const GenericPlayer &genericPlayer);

Клас : GenericPlayer.

Призначення: вивід гравця та його карти на екран.

Опис роботи: функція відправляє об'єкт типа GenericPlayer в стандартний поток виводу. Друкує ім'я гравця, його карти та загальн суму очок.

Аргументи:

- os — оператор виводу;
- genericPlayer — абстрактний клас GenericPlayer.

Отримання даних щодо бажання гравця продовжити брати карту

bool isHittingCard() const override;

Клас : Player.

Призначення: взнати чи хоче гравець взяти карту.

Опис роботи: якщо загальна кількість очок карт у руці гравця не дорівнює 21 - функція запитує чи хоче він взяти карту. Повертає істину якщо гравець хоче взяти карту.

Вивід гравця, котрий переміг

void playerWins() const;

Клас : Player.

Призначення: повідомлення гравця про перемогу.

Опис роботи: функція виводить ім'я гравця на екран та те, що він переміг.

Вивід гравця, котрий програв

void playerLoses() const;

Клас : Player.

Призначення: повідомлення гравця про поразку.

Опис роботи: функція виводить ім'я гравця на екран та те, що він програв.

Оголошення нічиї гравця

void playerPushes() const;

Клас : Player.

Призначення: повідомлення гравця про нічию.

Опис роботи: функція виводить ім'я гравця на екран та те, що він зіграв в нічию.

Отримання даних щодо бажання дилера продовжити брати карту

bool isHittingCard() const override;

Клас : Dealer.

Призначення: зрозуміти чи хоче дилер взяти карту.

Опис роботи: функція повертає істину якщо загальна кількість очок карт у руці дилера менше 17.

Перевертання першої карти

void flipFirstCard();

Клас : Dealer.

Призначення: сховати першу карту дилера.

Опис роботи: якщо наявні карти - функція перевертає першу.

Створення колоди

void populateDeck();

Клас : Deck.

Призначення: створити гральну колоду.

Опис роботи: функція створює гральну колоду з 52 карт.

Тасування колоди

void shuffleDeck();

Клас : Deck.

Призначення: потасувати колоду.

Опис роботи: функція тасує колоду карт за допомогою функції random_shaffle.

Взаємодія з гравцем

void dealWithPlayer(Hand &hand);

Клас : Deck.

Призначення: дати карту гравцю.

Опис роботи: якщо наявні карти в руці гравця функція додає карту.

Аргументи:

- *hand* — руку гравця.

Додавання додаткової карти

void additionalCards(GenericPlayer &genericPlayer);

Клас : Deck.

Призначення: дати гравцю карту якщо він не перебрав та хоче взяти її.

Опис роботи: якщо гравець не перебрав та бажає взяти карту, то функція взаємодіє з гравцем функцією dealWithPlayer. Далі виводить гравця на екран. Також якщо гравець перебрав - повідомляє про це.

Аргументи:

- *genericPlayer* — посилання на загального гравця.

Запуск гри

void startGame();

Клас : Game.

Призначення: провести гру.

Опис роботи: спочатку функція роздає кожному по дві гральні карти та ховає першу карту дилера. Далі відкриває руки всіх гравців. Тепер роздає додаткові карти гравцям(якщо гравець перебрав - друкує це). Потім відкриває першу карту дилера та роздає додаткові карти йому. Нарешті, підводить підсумки по грі. В кінці функція звільнює руки всіх гравців.

Чи бажає гравець продовжити грати

bool isPlayAgain();

Призначення: зрозуміти чи хоче користувач зіграти ще раз.

Опис роботи: функція запитує у користувача чи зіграти знову. Повертає істину якщо гравець хоче відновити гру.

Основна функція

int main()

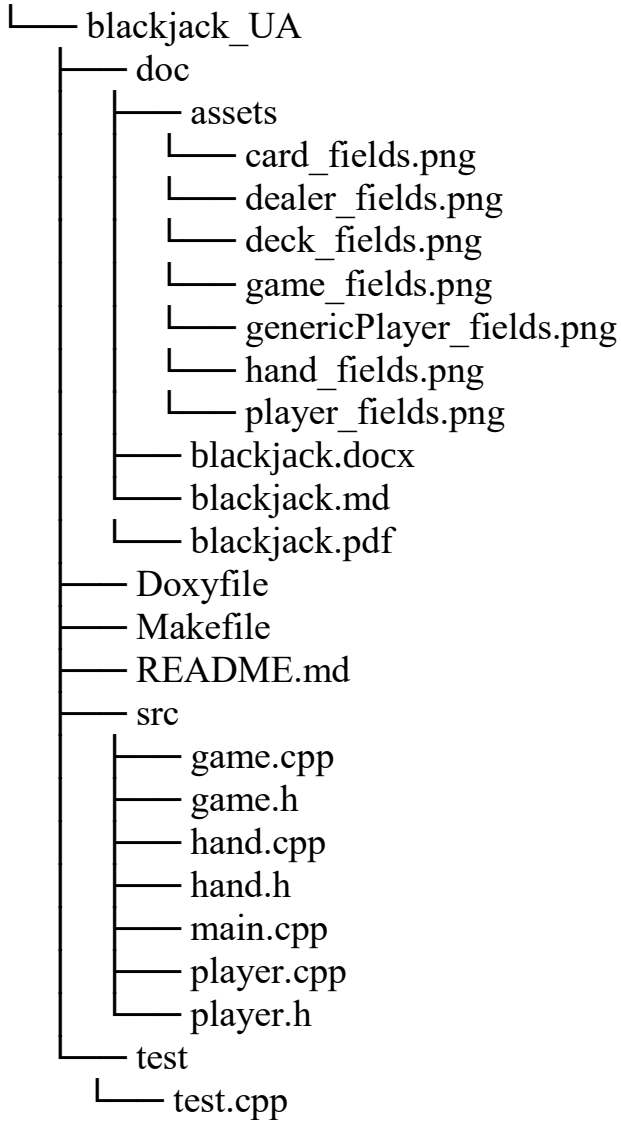
Призначення: головна функція.

Опис роботи:

- спочатку запитую у користувача з чого почати(старт, правила гри чи вихід) за допомогою функції introduction;
- тепер якщо гравец не хоче завершити програму - починаю ігровий процес;
- якщо користувач вирішив ознайомитися з правилами гри - показую їх функцією showRulesOfGame;
- потім запитую у гравців їхні імена шляхом виклику функції getNamesOfPlayers;
- нарешті у циклі while запускаю гру функцією startGame та буду виконувати її доки користувач не відмовиться відновлювати ігровий процес за допомогою функції isPlayAgain;

- успішний код повернення з програми (0).

Структура проекту:



2.3 Важливі фрагменти програми

Додавання додаткової карти

```
void Deck::additionalCards(GenericPlayer &genericPlayer)
{
	cout << endl;
	while (!(genericPlayer.isBusted()) && genericPlayer.isHittingCard()) {
		dealWithPlayer(genericPlayer);
		cout << genericPlayer << endl;
		if (genericPlayer.isBusted())
			genericPlayer.playerBusts();
	}
}
```

Запуск гри

```
void Game::startGame()
{
	// роздає кожному по дві карти
	vector<Player>::iterator iPlayer;
	for (int i = 0; i < 2; i++) {
		for (iPlayer = players.begin(); iPlayer != players.end(); iPlayer++)
			deck.dealWithPlayer(*iPlayer);
		deck.dealWithPlayer(dealer);
	}
	// ховає першу карту дилера
	dealer.flipFirstCard();
	// відкриває руки всіх гравців
	cout << "\tHanding out cards..." << endl;
	for (iPlayer = players.begin(); iPlayer != players.end(); iPlayer++)
		cout << *iPlayer << endl;
	cout << dealer << endl << endl;
	// роздає гравцям додаткові карти
	if (players.size() == 1)
		cout << "\tPlayer's move";
	else
		cout << "\tPlayers's move";
	for (iPlayer = players.begin(); iPlayer != players.end(); iPlayer++)
		deck.additionalCards(*iPlayer);
	cout << "\n\tDealer's move";
	// відкриває першу карту дилера
	dealer.flipFirstCard();
	cout << endl << dealer;
	deck.additionalCards(dealer);
	if (dealer.isBusted()) {
		for (iPlayer = players.begin(); iPlayer != players.end(); iPlayer++) {
			if (!(iPlayer->isBusted()))
				iPlayer->playerWins();
		}
	} else {
		// порівнює загальну кількість очок решти гравців з сумою дилера
		for (iPlayer = players.begin(); iPlayer != players.end(); iPlayer++) {
			if (!(iPlayer->isBusted())) {
				if (iPlayer->getTotal() > dealer.getTotal())
					iPlayer->playerWins();
				else if (iPlayer->getTotal() < dealer.getTotal())
					iPlayer->playerLoses();
				else
					iPlayer->playerPushes();
			}
```

```
            }
        }
        for (iPlayer = players.begin(); iPlayer != players.end(); iPlayer++) {
            iPlayer->clearHand();
        }
        dealer.clearHand();
}
```

3. Варіанти використання

Для демонстрації результатів кожної задачі використовується:

- виконання програми у вікні консолі.

Варіант використання 1: запуск програми у вікні консолі:

- запустити програму у консолі;
- взаємодіяти з початковим меню;
- ввести кількість гравців, далі - їх імена;
- тепер потрібно взяти карти тим гравцям, яким вони потрібні;
- подивитися на результати виконання програми;
- за бажанням є можливість повторити гру.

```
dima@dima-VirtualBox:~/dev/programing-khelemendyk-cpp/blackjack$
./dist/main.bin

                    Welcome to Blackjack!

        0 - START GAME
        1 - RULES OF THE GAME
        2 - QUIT

        Your choice: 0

                    BLACKJACK

                Getting name of players...
        How many players? (1 - 7): 2
```

Enter player name: Dima

Enter player name: Ivan Ivanov

Handing out cards...

Dima:

10h Qd (20)

Ivan Ivanov:

3c Qs (13)

Dealer:

XX 7d

Players's move

Dima, do you want a hit? (Y/N): n

Ivan Ivanov, do you want a hit? (Y/N): y

Ivan Ivanov:

3c Qs Jc (23)

Ivan Ivanov busts.

Dealer's move

Dealer:

2s 7d (9)

Dealer:

2s 7d 2h (11)

Dealer:

2s 7d 2h Kc (21)

Dima loses.

Do you want to play again? (Y/N): n

Goodbye!!!