**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию на тему

«Казино»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  |  |
| Герасимович Н.И. |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г

**Задание:**

Этот код представляет собой текстовую игру, в которой игрок может участвовать в двух карточных играх: блэкджек и дурак. В начале кода импортируется модуль random, который используется для случайного перемешивания колоды карт, и определяются основные функции для игры.

Функция blackjack(username, balance) начинает игру в блэкджек, приветствуя игрока и показывая его текущий баланс. Игроку предлагается ввести ставку, которая должна быть положительной и не превышать его баланс. Затем создается колода карт, и игроку и дилеру раздаются по две карты. Игрок видит свои карты и одну карту дилера. Если у игрока блэкджек (21 очко), он выигрывает. Игрок может выбрать действие: "Хит" (взять карту) или "Стенд" (остановиться). Если сумма очков превышает 21, игрок проигрывает. После завершения хода игрока дилер открывает свои карты и добирает карты, пока сумма его карт не станет 17 или больше. Затем определяется победитель на основе значений карт игрока и дилера, и результаты записываются в лог.

Функция calculate\_hand\_value(hand) рассчитывает сумму очков в руке игрока, учитывая, что тузы могут стоить 1 или 11 очков. Для работы с колодой определены функции create\_deck(), которая создает и перемешивает колоду, и card\_value(card), которая возвращает числовое значение карты. Также есть функции sort\_hand(hand, trump\_suit) и display\_hand(hand, trump\_suit), которые сортируют и отображают карты.

Функция fool(username, balance) начинает игру в дурака, предлагая игроку выбрать ставку. Определяется козырная масть, и игроку и компьютеру раздается по 6 карт. Игрок и компьютер по очереди атакуют и защищаются, пока один из них не останется без карт. Учитывается количество карт, и обновляется баланс игрока.

Функция main\_menu() представляет основное меню, где игрок может выбрать игру (блэкджек или дурак) или выйти из игры. Игрок может продолжать играть, пока у него есть баланс. При выходе из игры функция log\_exit(username, balance) записывает имя игрока и его текущий баланс в файл "game\_log.txt".

Таким образом, код реализует простую текстовую версию двух карточных игр, позволяя игрокам делать ставки, играть против компьютера и отслеживать свой баланс. Игровая логика включает в себя основные правила этих игр, а также обработку пользовательского ввода и вывод результатов.

Код программы на Python:  
  
import random  
def blackjack(username, balance):  
 print(f"\nНачинаем игру в блэкджек, {username}!")  
  
 # Выбор ставки  
 while True:  
 try:  
 bet = float(input("Введите вашу ставку: "))  
 if bet <= 0:  
 print("Ставка должна быть положительной!")  
 elif bet > balance:  
 print("Ставка не может превышать ваш текущий баланс!")  
 else:  
 break  
 except ValueError:  
 print("Пожалуйста, введите корректное число для ставки.")  
  
 # Создаем колоду  
 deck = create\_deck()  
 player\_hand = [deck.pop(), deck.pop()]  
 dealer\_hand = [deck.pop(), deck.pop()]  
  
 while True:  
 print(f"\nВаши карты: {display\_hand(player\_hand, '')} (сумма: {calculate\_hand\_value(player\_hand)})")  
 print(f"Карты дилера: {dealer\_hand[0][0]} {dealer\_hand[0][1]} и скрыта")  
  
 if calculate\_hand\_value(player\_hand) == 21:  
 print("У вас блэкджек! Вы выиграли!")  
 balance += bet \* 1.5 # Увеличиваем баланс на 1.5 ставки  
 break  
  
 action = input("Выберите действие: (1) Хит, (2) Стенд: ")  
 if action == '1':  
 player\_hand.append(deck.pop())  
 if calculate\_hand\_value(player\_hand) > 21:  
 print(f"Ваши карты: {display\_hand(player\_hand, '')} (сумма: {calculate\_hand\_value(player\_hand)})")  
 print("Вы превысили 21! Вы проиграли.")  
 balance -= bet  
 break  
 elif action == '2':  
 break  
 else:  
 print("Неверный выбор. Пожалуйста, выберите 1 или 2.")  
  
 # Дилер играет  
 while calculate\_hand\_value(dealer\_hand) < 17:  
 dealer\_hand.append(deck.pop())  
  
 print(f"Карты дилера: {display\_hand(dealer\_hand, '')} (сумма: {calculate\_hand\_value(dealer\_hand)})")  
  
 # Определяем победителя  
 player\_value = calculate\_hand\_value(player\_hand)  
 dealer\_value = calculate\_hand\_value(dealer\_hand)  
  
 if (dealer\_value > 21 and player\_value <=21)or (player\_value > dealer\_value and player\_value <= 21):  
 print("Вы выиграли!")  
 balance += bet  
 elif player\_value < dealer\_value:  
 print("Вы проиграли.")  
 balance -= bet  
 elif (player\_value == dealer\_value ) or ( player\_value > 21 and dealer\_value>21):  
 print("Ничья!")  
  
 log\_exit(username, balance)  
 return balance  
  
  
def calculate\_hand\_value(hand):  
 value = 0  
 aces = 0  
 for card in hand:  
 card\_val = card\_value(card)  
 if card\_val >= 10: # J, Q, K  
 value += 10  
 else:  
 value += card\_val + 2 # 2-10  
 if card[0] == 'A':  
 aces += 1  
  
 # Обработка тузов  
 while value > 21 and aces:  
 value -= 10  
 aces -= 1  
  
 return value  
  
  
# Определяем карты и их значения  
def create\_deck():  
 suits = ['♠', '♥', '♦', '♣']  
 ranks = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']  
 deck = [(rank, suit) for suit in suits for rank in ranks]  
 random.shuffle(deck)  
 return deck  
  
  
def card\_value(card):  
 ranks = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']  
 return ranks.index(card[0])  
  
  
def sort\_hand(hand, trump\_suit):  
 return sorted(hand, key=lambda card: (card[1] != trump\_suit, card\_value(card)))  
  
  
def display\_hand(hand, trump\_suit):  
 sorted\_hand = sort\_hand(hand, trump\_suit)  
 return ", ".join([f"{i + 1}. {rank} of {suit}" for i, (rank, suit) in enumerate(sorted\_hand)])  
  
  
def find\_smallest\_trump(hand, trump\_suit):  
 trump\_cards = [card for card in hand if card[1] == trump\_suit]  
 if trump\_cards:  
 return min(trump\_cards, key=card\_value)  
 return None  
  
  
def log\_exit(username, balance):  
 with open("game\_log.txt", "a") as log\_file:  
 log\_file.write(f"{username}: {balance:.2f}\n")  
  
  
def draw\_cards(hand, deck, max\_cards=6):  
 while len(hand) < max\_cards and deck:  
 hand.append(deck.pop())  
  
  
def fool(username, balance):  
 print(f"\nНачинаем игру в дурака, {username}!")  
  
 # Выбор ставки  
 while True:  
 try:  
 bet = float(input("Введите вашу ставку: "))  
 if bet <= 0:  
 print("Ставка должна быть положительной!")  
 elif bet > balance:  
 print("Ставка не может превышать ваш текущий баланс!")  
 else:  
 balance -= bet  
 break  
 except ValueError:  
 print("Пожалуйста, введите корректное число для ставки.")  
  
 # Случайный выбор козырной масти  
 suits = ['♠', '♥', '♦', '♣']  
 trump\_suit = random.choice(suits)  
 print(f"Козырная масть: {trump\_suit}")  
  
 deck = create\_deck()  
 player\_hand = [deck.pop() for \_ in range(6)]  
 computer\_hand = [deck.pop() for \_ in range(6)]  
  
 while True:  
 print(f"\nКоличество карт в колоде: {len(deck)}")  
 print("Ваши карты:", display\_hand(player\_hand, trump\_suit))  
 print(f"Количество карт у вас: {len(player\_hand)}")  
 print(f"Количество карт у дилера: {len(computer\_hand)}")  
  
 if not player\_hand:  
 print("Вы выиграли! У вас больше нет карт.")  
 break  
  
 # Определяем, кто ходит первым  
 player\_trump = find\_smallest\_trump(player\_hand, trump\_suit)  
 computer\_trump = find\_smallest\_trump(computer\_hand, trump\_suit)  
  
 if player\_trump and (not computer\_trump or card\_value(player\_trump) < card\_value(computer\_trump)):  
 attacking\_player = 'player'  
 else:  
 attacking\_player = 'computer'  
  
 # Процесс атаки  
 while True:  
 if attacking\_player == 'player':  
 # Игрок выбирает карту для атаки  
 while True:  
 try:  
 print("Вы можете атаковать следующими картами:")  
 print(display\_hand(player\_hand, trump\_suit))  
 card\_index = int(input(f"Выберите карту для атаки (1-{len(player\_hand)}): ")) - 1  
 if 0 <= card\_index < len(player\_hand):  
 attack\_card = player\_hand[card\_index]  
 break  
 else:  
 print("Неверный индекс. Попробуйте снова.")  
 except ValueError:  
 print("Пожалуйста, введите число.")  
  
 print(f"Вы атаковали картой: {attack\_card[0]} {attack\_card[1]}")  
 player\_hand.pop(card\_index) # Удаляем карту из руки игрока  
  
 # Компьютер отвечает  
 if computer\_hand:  
 defending\_card = None  
 for card in computer\_hand:  
 # Защита: карта той же масти с большим значением или козырная карта  
 if (card[1] == attack\_card[1] and card\_value(card) > card\_value(attack\_card)) or (  
 card[1] == trump\_suit and card[1] != attack\_card[1]):  
 defending\_card = card  
 break  
  
 if defending\_card:  
 computer\_hand.remove(defending\_card)  
 print(f"Компьютер защищается картой: {defending\_card[0]} {defending\_card[1]}")  
 attacking\_player = 'computer' # Меняем ход  
 else:  
 print("Компьютер не может защититься, он берет карту атаки.")  
 computer\_hand.append(attack\_card) # Компьютер берет карту атаки  
 attacking\_player = 'computer' # Компьютер пропускает ход атаки  
 else:  
 print("Компьютер не может защититься, он берет карту атаки.")  
 computer\_hand.append(attack\_card) # Компьютер берет карту атаки  
 attacking\_player = 'computer' # Компьютер пропускает ход атаки  
  
 else:  
 # Компьютер выбирает карту для атаки  
 if computer\_hand:  
 attack\_card = computer\_hand.pop(0) # Компьютер атакует первой картой  
 print(f"Компьютер атакует картой: {attack\_card[0]} {attack\_card[1]}")  
  
 # Игрок отвечает  
 if player\_hand:  
 print("Выберите карту для защиты:")  
 defendable\_cards = [card for card in player\_hand if (  
 card[1] == attack\_card[1] and card\_value(card) > card\_value(attack\_card)) or (  
 card[1] == trump\_suit)]  
  
 if defendable\_cards:  
 for i, card in enumerate(defendable\_cards):  
 print(f"{i + 1}. {card[0]} {card[1]}")  
  
 while True:  
 try:  
 defend\_index = int(  
 input(f"Выберите карту для защиты (1-{len(defendable\_cards)}): ")) - 1  
 if 0 <= defend\_index < len(defendable\_cards):  
 defending\_card = defendable\_cards[defend\_index]  
 break  
 else:  
 print("Неверный индекс. Попробуйте снова.")  
 except ValueError:  
 print("Пожалуйста, введите число.")  
  
 player\_hand.remove(defending\_card)  
 print(f"Вы защищаетесь картой: {defending\_card[0]} {defending\_card[1]}")  
 attacking\_player = 'player' # Меняем ход  
 else:  
 # Игрок не может защититься, предлагает взять карту  
 take\_card = input(  
 "Вы не можете защититься. Хотите взять атакующую карту? (да/нет): ").strip().lower()  
 if take\_card == 'да':  
 player\_hand.append(attack\_card) # Игрок берет карту атаки  
 print(f"Вы взяли карту: {attack\_card[0]} {attack\_card[1]}")  
 attacking\_player = 'computer' # Игрок пропускает ход атаки  
 else:  
 print("Вы отказались брать карту. Вы проиграли!")  
 balance -= bet # Игрок теряет ставку  
 log\_exit(username, balance)  
 return balance # Возвращаем текущий баланс  
 else:  
 print("У вас нет карт для защиты, игра окончена.")  
 break  
 else:  
 print("Компьютер не может атаковать, игра окончена.")  
 break  
  
 # Добор карт после атаки  
 draw\_cards(player\_hand, deck)  
 draw\_cards(computer\_hand, deck)  
  
 # Отображаем обновленную руку игрока и количество карт в колоде после добора  
 print("Ваши карты после добора:", display\_hand(player\_hand, trump\_suit))  
 print(f"Количество карт в колоде после добора: {len(deck)}")  
  
 log\_exit(username, balance)  
 return balance # Возвращаем текущий баланс  
  
  
def main\_menu():  
 print("Добро пожаловать в казино!")  
 username = input("Введите ваше имя: ")  
 initial\_balance = 100 # Начальный баланс прописан в коде  
 balance = initial\_balance  
  
 while balance > 0:  
 print(f"\nВаш текущий баланс: {balance:.2f}")  
 print("Выберите игру:")  
 print("1. Блэкджек")  
 print("2. Дурак")  
 print("3. Выход")  
  
 choice = input("Ваш выбор: ")  
  
 if choice == '1':  
 balance = blackjack(username, balance)  
 elif choice == '2':  
 balance = fool(username, balance)  
 elif choice == '3':  
 log\_exit(username, balance)  
 print(f"{username}, вы вышли из игры. Ваш баланс записан.")  
 break  
 else:  
 print("Неверный выбор, попробуйте снова.")  
  
 if balance <= 0:  
 print("Вы исчерпали ваш баланс. Игра окончена.")  
 log\_exit(username, balance)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main\_menu()