**КУРСОВАЯ РАБОТА**

Создание игры «Blackjack» с использование STL и графической библиотеки SFML

по дисциплине: «Объектно-ориентированное программированиe»

Выполнил

студент гр. 3530904/80003 Коршунов К. В.

Руководитель Александрова О. В.

**Оглавление**

[**Цель** 2](#_Toc28197080)

[**Структура** 3](#_Toc28197081)

[**Пример работы** 5](#_Toc28197082)

[**Вывод** 8](#_Toc28197083)

# **Цель**

Создание игры «Blackjack» на языке С++ с использованием STL и графической библиотеки SFML.

# **Структура**

Файлы:

* card.hpp
* card.cpp
* hand.hpp
* hand.cpp
* dealer.hpp
* dealer.cpp
* deck.hpp
* deck.cpp
* main.cpp
* game.hpp – служит для вынесения функций отрисовки из файла main.cpp

Классы

Deck – содержит два вектора номеров карт. Первый - это карты в колоде, а второй – это карты на столе. Также содержит методы извлечения карты из колоды, возвращение карт со стола в колоду и метод перемешивания колоды.

class Deck {  
public:  
 Deck();  
  
 int getCard();  
 void collectDeck();  
 void mixDeck();  
  
  
 ~Deck() = default;  
private:  
 std::vector<int> deck\_;  
 std::vector<int> deckReturn\_;  
};

Card – содержит в себе номер и ранг карты.

class Card {  
public:   
 Card(const int);  
  
 int rang\_;  
 int num\_;  
  
 ~Card() = default;  
};

Hand – Рука представляет собой вектор карт (умных указателей). Класс содержит методы добавления карты в руку, добавление двух карт, очистки руки, получения приоритета, возврата ставки в зависимости от приоритетов руки и дилера. x\_ и y\_ - координаты руки на столе, points – это сумма рангов карт руки, bet\_ - ставка на данную руку, insurance – страховка, count\_ - количество карт в руке, countT\_ - количество тузов в руке (необходимо для реализации двойного ранга тузов, так как ранг туза может быть 1 и 11).

class Hand {  
public:  
 Hand(const int);  
  
 std::vector<std::shared\_ptr<Card>> cards\_;  
 void addCard(const int);  
 void addTwoCard(const int, const int);  
 void clearHand();  
 int getPriority();  
 int getBet(const int);  
  
 int x\_;  
 int y\_;  
  
 int points\_;  
 int bet\_;  
  
 bool insurance;  
  
 ~Hand() = default;  
  
private:  
 int count\_;  
 int countT\_;  
};

Dealer – дилер представляет собой вектор карт (умных указателей). Класс содержит методы добавления карты, добавление двух карт, очистки, получения приоритета. Константы x\_ и y\_ - координаты на столе, points – это сумма рангов карт руки, count\_ - количество карт в руке, countT\_ - количество тузов.

class Dealer {  
public:  
 Dealer();  
  
 std::vector<std::shared\_ptr<Card>> cards\_;  
 void addCard(const int);  
 void addTwoCard(const int, const int);  
 int getPriority();  
 void clearDealer();  
  
 const int x\_ = 700;  
 const int y\_ = 100;  
  
 int points\_;  
 bool begin\_;  
  
 ~Dealer() = default;  
  
private:  
 int count\_;  
 int countT\_;  
};

# **Пример работы**

Для начала игры необходимо сделать ставку хотя бы на один бокс.



После произойдет раздача карт на боксы с ненулевой ставкой.



**Ранг колоды -**

**- Ставка**

**- Указатель на активный бокс**

Hit – взять карту

Stand – оставить имеющиеся карты

Double – взять всего одну карту, но при этом удвоить ставку

Surrender – сбросить карты этой руки, вернув половину ставки

После выполнения действий с каждым боксом, дилер открывает свою вторую карту и берет из колоды, пока сумма очков будет не меньше 17. Когда дилер доберет карты, по нажатию на клавишу мыши происходит подсчет очков на каждом боксе, и возвращается сумма в зависимости от победы или поражения. Переход к первой стадии.



Также существует отдельная стадия, если открытая карта дилера по рангу равна 10 или 11.

Ранг = 10: Если вторая карта по рангу равна 11, то сразу происходит подсчет очков, без возможности добрать карты. В обратном случае игра продолжается.

Ранг = 11: Игроку будет предложено застраховать его боксы. Страховка равна половине ставки.

Если вторая карта дилера по рангу равна 10, то с застрахованных боксов возвращается вся ставка, а для незастрахованных происходит подсчет очков. Если же вторая карта дилера по рангу не равна 10, то все страховки сгорают, и игра продолжается.

# **Вывод**

Цель моей работы была достигнута. Была получена игра “Blackjack” с использованием графической библиотеки SFML и библиотеки STL, в виде vector, accumulate, for\_each.