

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Пензенский государственный университет»

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему «Разработка программы сложной структуры методом нисходящего
программирования. Игра «Блэк-джек»

20.05.2026
отмечено
Г. Генералова Н.А.

Выполнил: ст. гр. 25BBB1
Самсонов М.А.

Проверил: к.т.н. доцент
Генералова А.А.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

 М.А. Митрохин

« 18 » февраля 2026 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Самсонову Михаилу Алексеевичу Группа 258ВВ61

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

Игра "Блэкджек".

Генерацию карт осуществляет компьютер

Игрок пытается набрать больше очков, чем у дилера, но не более 21.

В файл загружается и сохраняется таблица рекордов

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы,
иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания " 18 " февраля 2026

Дата защиты проекта " 20 " апреля 2026

Руководитель Гамзатов А. А.

Задание получил " 18 " февраля 2026 г.

Студент Самсонов М. А.

Содержание

Введение.....	5
1 Постановка задачи.....	6
2 Выбор решения.....	7
3 Описание разработки программы.....	8
4 Отладка и тестирование.....	9
5 Описание программы.....	10
5.1 Схема функции main	10
5.2 Описание модулей.....	10
6 Руководство пользователя.....	12
6.1 Заставка	12
6.2 Правила игры	12
6.3 Главное меню.....	13
6.4 Игровой процесс.....	13
6.5 Подсчёт очков.....	16
6.6 Завершение игры и сохранение рекордов	17
Заключение	18
Список используемых материалов	19
Приложение А	19
Листинг программы	20
Файл blackjack.c.....	20
Файл blackjack.h	23
Файл easy.c	24
Файл hard.c	26

Введение

В современном мире компьютерные игры занимают значительное место в сфере развлечений и даже профессиональной деятельности. Программирование подобных игр на языке С подробно рассмотрено в работах Кернигана и Ричи [1], а также в учебнике Павловской [4]. Одной из таких игр можно считать популярную карточную игру «Блэк-джек», за счёт того, что она развивает вероятностное мышление и умение принимать решения в условиях неопределённости. Это игра представляет интерес с точки зрения алгоритмизации и программирования. Изучение принципов работы этой игры позволяет углубить знания в области разработки программного обеспечения, обработки пользовательского ввода и реализации сложной игровой логики с чёткими правилами.

Актуальность работы заключается в том, что реализация игры «Блэк-джек» охватывает целый ряд фундаментальных тем программирования: генерацию случайных чисел, алгоритмы подсчёта очков с учётом гибкой стоимости туза, а также реализацию поведения искусственного противника – дилера, который действует по правилам.

В данной работе используется язык программирования Си, потому что этот язык предоставляет низкоуровневый контроль над памятью и структурами данных, позволяя гибко управлять массивами для хранения колоды и динамически изменять колоду у игрока. Также в работе применяется указатели для передачи данных между функциями и генератор случайных чисел для эмуляции тасования и выдачи карт, а также ассемблерная вставка, для проверки стоимости туза.

1 Постановка задачи

Целью данной работы является разработка консольной версии карточной игры «Блэк-джек» с использованием языков программирования Си и Ассемблер. В ходе выполнения работы необходимо реализовать следующий функционал:

1. Генерацию стандартной колоды из 52 карт
2. Перемешивание колоды
3. Раздачу карт игроку и дилеру
4. Возможность добирать карты или останавливаться у игрока
5. Автоматический ход дилера, набирать до 17 или 18 в

зависимости от сложности

6. Подсчёт очков руки с учётом туза
7. Определение победителя
8. Ведение таблицы рекордов с сохранением в файл
9. Два уровня сложности:
 - a. Легкий: валет = 10, дама = 10, король, туз = 11, дилер берёт карты до 17
 - b. Сложный: валет = 12, дама = 13, король = 14, туз = 11, дилер берёт карты до 18

Во время реализации проекта должны быть использованы сложные типы данных – структуры, массивы, файлы, алгоритмы обработки ввода и вывода и принципы модульного программирования. Особенности работы с динамическими структурами в языке Си подробно описаны в учебном пособии Подбельского [5].

Сохранение результата игр.

Режим работы видеосистемы – текстовый/графический.

Устройство ввода информации – клавиатура и мышь.

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.

Наличие заставки.

2 Выбор решения

Для разработки выбрана среда Visual Studio 2026, поддерживающая язык C и ассемблерные вставки (`__asm`). Она предоставляет все необходимые инструменты для комфортной и эффективной разработки игры «Блэк-джек», делая процесс написания кода, отладки и оптимизации максимально удобным.

Проект структурирован с использованием модулей, что позволяет логически разделить код на функциональные блоки, которые легко сопровождать. Схема данных приведена на рисунке 1.

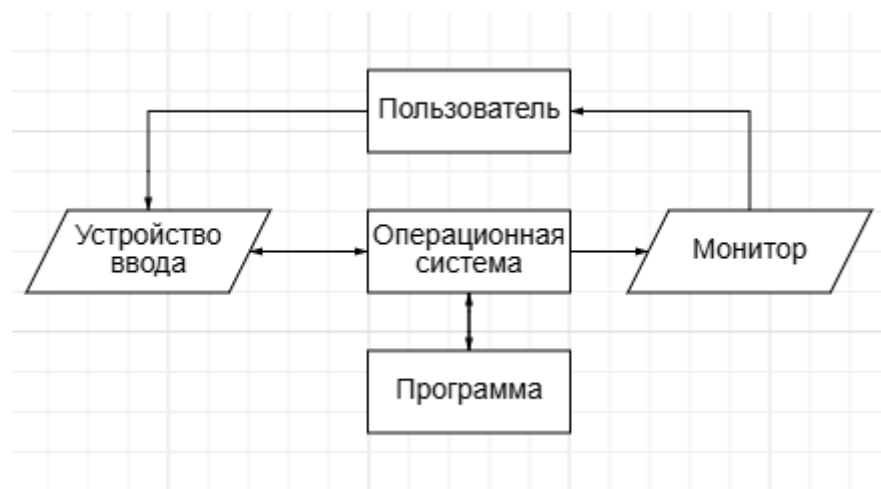


Рисунок 1 – Схема данных

Visual Studio также подходит для работы с языком ассемблера благодаря интеграции низкоуровневых возможностей с удобной средой разработки.

Главным преимуществом является поддержка встроенного ассемблера, что позволяет писать код в теле программы без необходимости использования

дополнительных файлов. Ассемблерная вставка используется в функции `Score` для коррекции суммы при наличии тузов - замена 11 на 1 при переборе.

Данные о рекордах сохраняются в текстовый файл `records.txt` через функции `load_records()` / `save_records()`. По завершению игры в файл записываются результаты сессии: количество игр, побед, поражений и ничьи.

3 Описание разработки программы

Для реализации был выбран нисходящий метод разработки, потому что он позволяет сначала определить общую структуру программы, а затем постепенно детализировать её компоненты.

При запуске программы на экран выводится приветственная заставка, правила пользования и 2 уровня сложности:

1. Лёгкий: валет=10, дама=10, король=10, туз=11, дилер берёт карты до 17.
2. Сложный: валет = 12, дама = 13, король = 14, туз = 11, дилер берёт карты до 18.

Программа разделена на 4 модуля, потому что это оптимальный вариант, который позволяет изменять одну из частей, не затрагивая остальные, а также соответствует принципам структурного программирования [2]:

1. `blackjack.h` – общие структуры и прототипы;
2. `blackjack.c` – главный модуль: инициализация колоды, перемешивание, подсчёт очков с ассемблерной вставкой, вывод таблицы, меню;
3. `easy.cpp` – логика лёгкого уровня.
4. `hard.cpp` – логика сложного уровня.

4 Отладка и тестирование

В качестве среды разработки была выбрана программа Visual Studio 2026. Она оптимальна как для разработки на языке программирования Си, так и для отладки программы благодаря мощному функционалу и удобству работы.

Точки останова позволяют исследовать состояние программы, чтобы проверить, как ведёт себя приложение.

На рисунке 2 показан пример использования точек останова.

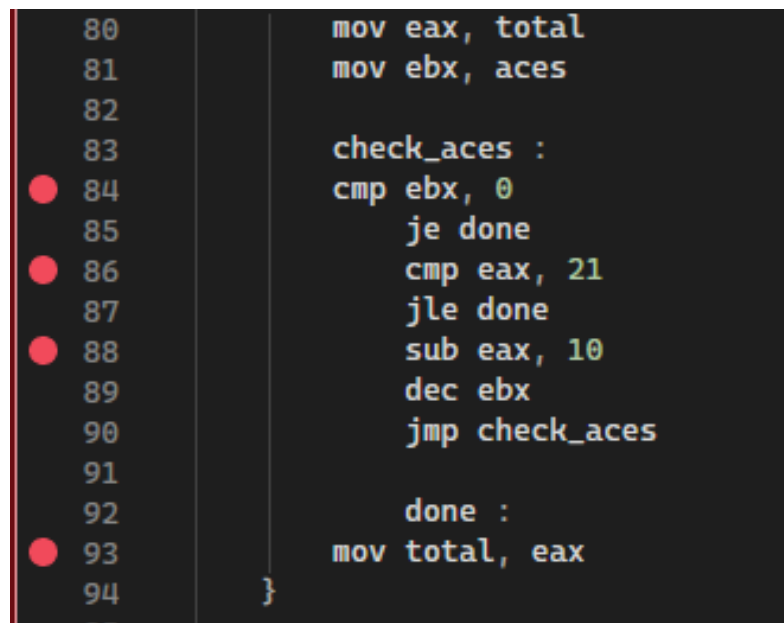


Рисунок 2 – Использование точек останова

Тестирование проводилось в рабочем процессе, в процессе разработки и после завершения написания программы.

5 Описание программы

5.1 Схема функции main

На рисунке 3 представлена схема функции main.

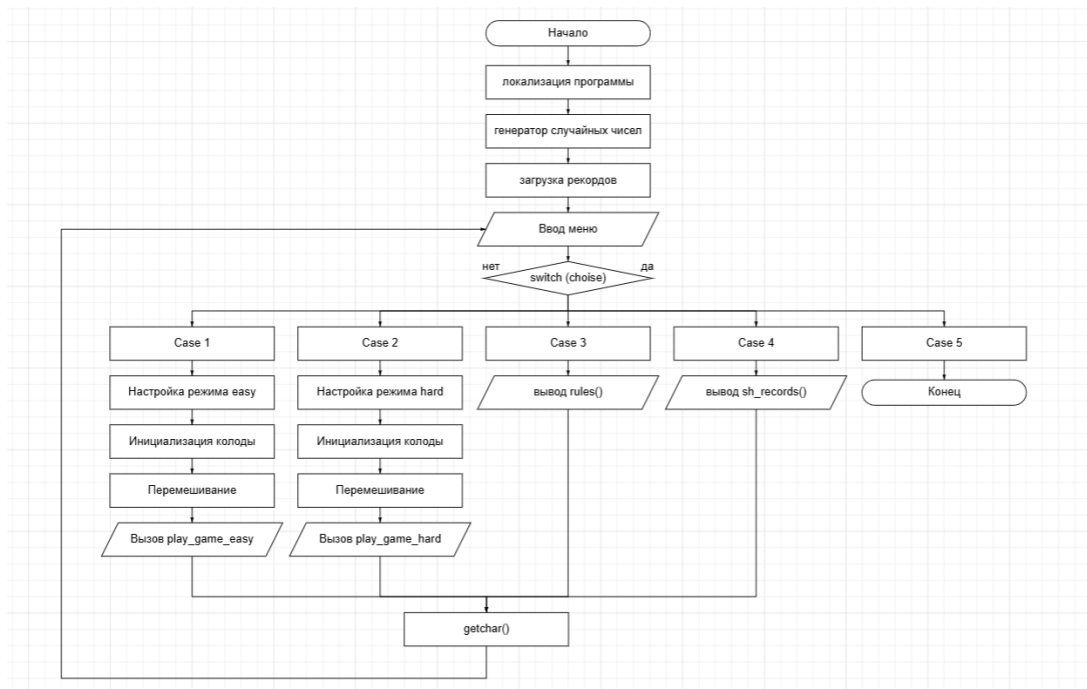


Рисунок 3 – Схема функции main

На схеме представлена последовательность действий при запуске программы: установка локализации, загрузка рекордов, инициализация генератора случайных чисел, затем бесконечный цикл с выводом меню и ветвление по выбору пользователя. Схема наглядно показывает, как организовано управление всей программой.

5.2 Описание модулей

Проект состоит из следующих модулей:

1. `blackjack.h` – заголовочный файл, содержит:
 - Определение структур `Card` (ранг, значение) и `Records` (победы, поражения, ничьи).
 - Объявления глобальных переменных и прототипов функций.
2. `Blackjash.c` – главный модуль:
 - `Init_deck()` – инициализация колоды под выбранный режим

- Shuffle() – перемешивание
 - Draw_card() – выдача карты
 - Score() – подсчёт очков
 - Print_cards(), print_score(), draw_table() – вывод игрового стола.
 - Load_records(), save_records – работы с файлом рекордов
 - Menu(), rules(), sh_records() – служебные функции
 - Main() – управляющая функция
3. Easy.c – логика лёгкого уровня:
- Play_game_easy() – реализует игровой процесс с порогом дилера 17 и классическими значениями карт
4. Hard.c – логика сложного уровня:
- Play_game_hard() – аналогично, но порог дилера 18 и повышенные значения фигур.

Ассемблерная вставка демонстрирует низкоуровневый цикл корректировки тузов[3]:

```
__asm {
    mov eax, total
    mov ebx, aces
check_aces:
    cmp ebx, 0
    je done
    cmp eax, 21
    jle done
    sub eax, 10
    dec ebx
    jmp check_aces
done:
    mov total, eax
}
```

Нисходящий метод проектирования подробно описан у Кернигана и Ричи [1], а фундаментальные принципы работы ассемблера изложены в классическом учебнике Ирвина [6].

6 Руководство пользователя

Программа blackjack.exe является консольной игрой, имитирующей правила Блэк-джека.

6.1 Заставка

На рисунке 4 представлена игровая заставка, которая появляется при запуске приложения.

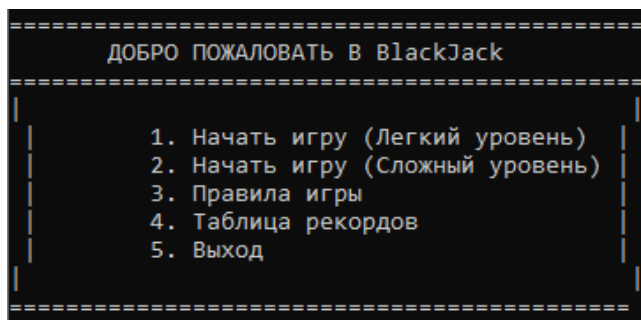


Рисунок 4 – Заставка

Меню оформлено символами псевдографики для наглядности, а заголовок отделен линиями для улучшения читаемости интерфейса.

6.2 Правила игры

На рисунке 5 представлены правила игры.

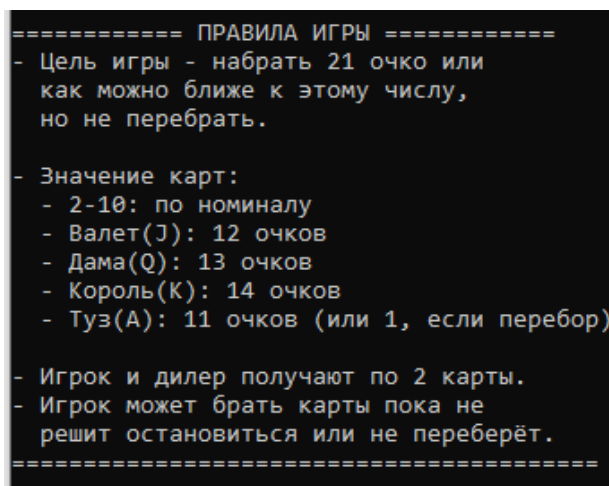


Рисунок 5 – Правила игры

Правила представлены в сжатом виде, чтобы не полностью перегружать пользователя, но при этом дать полное понимание механики.

6.3 Главное меню

На рисунке 6 представлено главное меню игры, в котором предлагают 5 опций:

1. Легкий уровень
2. Сложный уровень
3. Правила
4. Таблица рекордов
5. Выход

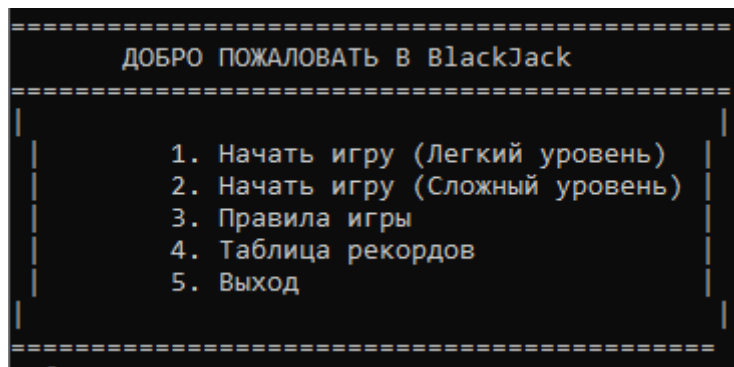


Рисунок 6 – главное меню

Пользователь вводит 1 или 2. После чего программа инициализирует колоду с соответствующими правилами.

6.4 Игровой процесс

После выбора уровня игрок видит свои две карты, одну карту дилера и очки. На рисунке 7 представлен игровой стол с картами и очками. Появляется запрос: взять карту (1) и остаться (2).

```
== Ваши карты ==  
[10] [8]  
Счёт: 18  
  
== Карты дилера ==  
[XX] [K]  
Счёт дилера: ???  
  
1. Взять карту  
2. Остаться  
Введите действие:
```

Рисунок 7 – игровой стол

При взятии карты она добавляется, сумма пересчитывается. Если сумма >21, игрок проигрывает. Бывают и ситуации, когда у игрока или дилера с самого начала игры перебор (>21). На рисунке 8 представлена ситуация, когда у игрока >21

```
== Ваши карты ==  
[10] [8]  
Счёт: 18  
  
== Карты дилера ==  
[XX] [K]  
Счёт дилера: ???  
  
1. Взять карту  
2. Остаться  
Введите действие: 1  
  
Вы взяли карту [Q]  
Счёт: 28  
Перебор! Вы проиграли!
```

Рисунок 8 – Перебор у игрока

После остановки игрока дилер добирает карту до определенного порога (17 или 18). Затем сравниваются очки и объявляется результат (рис. 9).

```
== Ваши карты ==  
[5] [8]  
Счёт: 13  
  
== Карты дилера ==  
[XX] [A]  
Счёт дилера: ???  
  
1. Взять карту  
2. Остаться  
Введите действие: 1  
  
Вы взяли карту [6]  
  
== Ваши карты ==  
[5] [8] [6]  
Счёт: 19  
  
== Карты дилера ==  
[XX] [A]  
Счёт дилера: ???  
  
1. Взять карту  
2. Остаться  
Введите действие: 2  
  
== Ход дилера ==  
  
== Результаты ==  
Ваши карты: [5] [8] [6]  
  
Счёт: 19  
  
Карты дилера: [10] [A]  
  
Счёт дилера: 21  
Дилер победил!  
  
Ещё раунд?_
```

Рисунок 9 – игровой процесс

Данный рисунок показывает полный игровой процесс, для того, чтобы пользователь мог понять, как играть и как выглядит обычная игровая сессия.

6.5 Подсчёт очков

Подсчёт автоматический. Туз даёт 11 очков, но если общая сумма превышает 21, туз автоматически начинает считаться как 1 (рис. 10).

```
== Ваши карты ==  
[5] [3] [3] [2]  
Счёт: 13  
  
== Карты дилера ==  
[XX] [A]  
Счёт дилера: ???  
  
1. Взять карту  
2. Остаться  
Введите действие: 1  
  
Вы взяли карту [8]  
  
== Ход дилера ==  
Дилер взял карту [9]  
Дилер взял карту [8]  
  
== Результаты ==  
Ваши карты: [5] [3] [3] [2] [8]  
  
Счёт: 21  
  
Карты дилера: [2] [A] [9] [8]  
  
Счёт дилера: 20  
Вы победили!  
  
Ещё раунд?
```

Рисунок 10 – пересчёт туза у дилера
Благодаря такой оптимизации программа быстро корректирует счёт.

6.6 Завершение игры и сохранение рекордов

После каждой партии (победа/поражение/ничья) заносится в таблицу рекордов и сохраняется в файл records.txt (рис. 11).

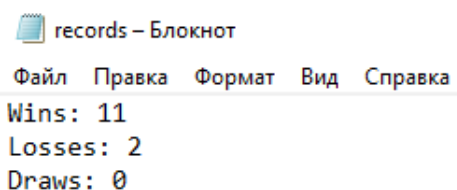


Рисунок 11 – таблица рекордов в файле records.txt

Помимо этого, таблицу рекордов можно посмотреть и в самой игре, выбрав в главном меню «4» (рис. 12).

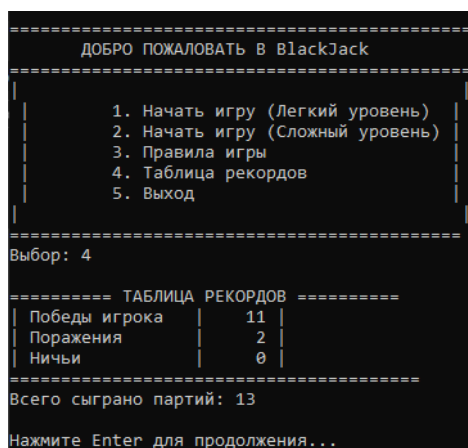


Рисунок 12 – таблица рекордов в игре

Программа автоматически обновляет эти данные после каждой партии и загружает их при запуске. Это позволяет сохранять статистику между сеансами игры. Таким образом, пользователь может отслеживать свой прогресс.

Заключение

В ходе выполнения курсовой работы была разработана консольная игра «Блэк-джек» на языке Си с двумя уровнями сложности и использованием ассемблерной вставки.

Основные достижения:

1. Реализована полная логика классической карточной игры.
2. Применено модульное программирование – это разделение на .h и .сфайлы.
3. Используются динамические структуры – это колода как массив структур и файловый ввод/вывод – сохранение рекордов.
4. В критический алгоритм, в данном случае подсчёт очков с тузами, внедрена ассемблерная вставка для оптимизации и демонстрации низкоуровневых возможностей.
5. Создано удобное текстовое меню и таблица рекордов.

Проект показывает эффективность языка Си для написания игровой логики. Программа успешно реализует все основные функции карточной игры «Блэк-джек». Приобретенные в ходе выполнения работы навыки имеют важное значение для дальнейшего профессионального роста в области программирования.

Список используемых материалов

1. Керниган Б. У. Язык программирования С: пер. с англ. / Б. У. Керниган, Д. М. Ричи. – 3-е изд., испр. – Москва : Вильямс, 2021. – 304 с.
2. Прата, С. Язык программирования С. Лекции и упражнения : пер. с англ. / С. Прата. – 6-е изд. – Москва : Диалектика, 2019. – 896 с.
3. Зубков, С. В. Ассемблер для процессоров Intel x86 : учебное пособие / С. В. Зубков. – Москва : ДМК Пресс, 2020 – 412 с.
4. Павловская Т. А. С/С++. Программирование на языке высокого уровня : учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2019 – 464 с.
5. Подбельский В. В. Язык Си : учеб. Пособие / В. В. Подбельский. – 6-е изд., перераб. и доп. – М. : Финансы и статистика, 2020. – 600 с.
6. Ирвин К. Язык ассемблера для процессоров Intel : пер. с англ. / К. Ирвин. – 7-е изд. – М. : Вильямс, 2017. – 912 с.
7. Фридман А. Л. Основы объектно-ориентированного программирования на С++ : учеб. пособие / А. Л. Фридман. – М. : Интернет-Университет Информационных Технологий, 2017. – 235 с.

Приложение А

Листинг программы

Файл blackjack.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <locale.h>
#include <errno.h>
#include <Windows.h>
#include "blackjack.h"
#include "hard.cpp"
#include "easy.cpp"

Card deck[52];
int deck_index;
Records records = { 0, 0, 0 };
const char* RECORDS_FILE = "records.txt";

void init_deck(int mode) {
    const char* ranks[] = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "J",
"Q", "K", "A" };
    int values[] = { 2,3,4,5,6,7,8,9,10,12,13,14,11 };

    switch (mode) {
        case 1:
            for (int i = 0; i < 10; i++) values[i] = i + 2;
            values[10] = 10;
            values[11] = 10;
            values[12] = 10;
            break;
        case 2:
            for (int i = 0; i < 10; i++) values[i] = i + 2;
            values[10] = 12;
            values[11] = 13;
            values[12] = 14;
            break;
        default:
            printf("Неизвестный режим, использую классический.\n");
            for (int i = 0; i < 10; i++) values[i] = i + 2;
            values[10] = values[11] = values[12] = 10;
    }
    values[12] = 11;

    int index = 0;
    for (int i = 0; i < 13; i++) {
        for (int j = 0; j < 4; j++) {
            strcpy_s(deck[index].rank, sizeof(deck[index].rank),
ranks[i]);

            deck[index].value = values[i];
            index++;
        }
    }
}

void Shuffle() {
    for (int i = 0; i < 52; i++) {
        int j = rand() % 52;
        Card temp = deck[i];
        deck[i] = deck[j];
        deck[j] = temp;
    }
    deck_index = 0;
}
```

```

Card draw_card() {
    if (deck_index >= 52) {
        Shuffle();
    }
    return deck[deck_index++];
}

int Score(Card* cards, int num_cards) {
    int total = 0;
    int aces = 0;

    for (int i = 0; i < num_cards; i++) {
        total += cards[i].value;
        if (cards[i].value == 11) {
            aces++;
        }
    }

    __asm {
        mov eax, total
        mov ebx, aces

        check_aces :
        cmp ebx, 0
            je done
            cmp eax, 21
            jle done
            sub eax, 10
            dec ebx
            jmp check_aces

        done :
        mov total, eax
    }

    return total;
}

void print_cards(Card* cards, int num_cards, int hide_first) {
    for (int i = 0; i < num_cards; i++) {
        if (hide_first && i == 0) {
            printf("[XX] ");
        }
        else {
            printf("[%s] ", cards[i].rank);
        }
    }
    printf("\n");
}

void print_score(Card* cards, int num_cards, int hide_first) {
    if (hide_first) {
        printf("Счёт дилера: ???\n");
    }
    else {
        int score = Score(cards, num_cards);
        printf("Счёт дилера: %d\n", score);
    }
}

void draw_table(Card* player, int player_count, Card* dealer, int dealer_count, int
hide_first, int player_score, int dealer_score) {
    printf("\n===== \n");
    printf("                ИГРОВОЙ СТОЛ                \n");
    printf("===== \n");
    printf("Дилер: \n ");
    for (int i = 0; i < dealer_count; i++) {
        if (hide_first && i == 0) printf("[XX] ");
        else printf("[%s] ", dealer[i].rank);
    }
}

```

```

printf("\n Очки: ");
if (hide_first) printf("???\n");
else printf("%d\n", dealer_score);

printf("\nВы:\n ");
for (int i = 0; i < player_count; i++) {
    printf("[%s] ", player[i].rank);
}
printf("\n Очки: %d\n", player_score);
printf("=====\n");
}
void load_records() {
    FILE* file = NULL;
    errno_t err = fopen_s(&file, RECORDS_FILE, "r");
    if (err != 0 || file == NULL) return;
    fscanf_s(file, "Wins: %d\nLosses^ %d\nDraws: %d",
        &records.wins, &records.losses, &records.draw);
    fclose(file);
}
void save_records() {
    FILE* file = NULL;
    errno_t err = fopen_s(&file, RECORDS_FILE, "w");
    if (err != 0 || file == NULL) {
        printf("Ошибка сохранения рекордов!\n");
        return;
    }
    fprintf(file, "Wins: %d\nLosses: %d\nDraws: %d",
        records.wins, records.losses, records.draw);
    fclose(file);
}

void sh_records() {
    printf("\n===== ТАБЛИЦА РЕКОРДОВ =====\n");
    printf("| %-15s | %5d |\n", "Победы игрока", records.wins);
    printf("| %-15s | %5d |\n", "Поражения", records.losses);
    printf("| %-15s | %5d |\n", "Ничьи", records.draw);
    printf("=====\n");
    printf("Всего сыграно партий: %d\n",
        records.wins + records.losses + records.draw);
    printf("\nНажмите Enter для продолжения...");
    getchar();
}
void menu() {
    printf("\n===== \n");
    printf("        ДОБРО ПОЖАЛОВАТЬ В BlackJack        \n");
    printf("===== \n");
    printf("| \n");
    printf(" |          1. Начать игру (Легкий уровень) |\n");
    printf(" |          2. Начать игру (Сложный уровень) |\n");
    printf(" |          3. Правила игры                  |\n");
    printf(" |          4. Таблица рекордов              |\n");
    printf(" |          5. Выход                        |\n");
    printf("| \n");
    printf("===== \n");
    printf("Выбор: ");
}

void rules() {
    printf("\n===== ПРАВИЛА ИГРЫ =====\n");
    printf("- Цель игры - набрать 21 очко или\n");
    printf("как можно ближе к этому числу,\n");
    printf("но не перебрать.\n\n");
    printf("- Значение карт:\n");
    printf("  - 2-10: по номиналу\n");
    printf("  - Валет(J): 12 очков\n");
    printf("  - Дама(Q): 13 очков\n");
    printf("  - Король(K): 14 очков\n");
    printf("  - Туз(A): 11 очков (или 1, если перебор)\n\n");
    printf("- Игрок и дилер получают по 2 карты.\n");
    printf("- Игрок может брать карты пока не\n");

```

```

printf("  решит остановиться или не переберёт.\n");
printf("=====\\n");
printf("\\nНажмите Enter для продолжения...");
getchar();
getchar();
}
int main() {
    setlocale(LC_ALL, "Rus");
    system("chcp 1251 > nul");
    srand(time(NULL));
    load_records();

    int choise, mode = 1;

    do {
        Shuffle();
        menu();
        scanf_s("%d", &choise);

        switch (choise) {
            case 1:
                init_deck(1);
                Shuffle();
                play_game_easy();
                printf("\\nЕщё раунд?");
                getchar();
                getchar();
                break;
            case 2:
                init_deck(2);
                Shuffle();
                play_game_hard();
                printf("\\nЕщё раунд?");
                getchar();
                getchar();
                break;
            case 3:
                rules();
                break;
            case 4:
                sh_records();
                break;
            case 5:
                printf("\\nСлишком сложно, не так ли?\\n");
                break;
            default:
                printf("\\nНеверный выбор! Попробуйте снова\\n");
                printf("\\nНажмите Enter для продолжения...");
                getchar();
                getchar();
        }
    } while (choise != 5);

    return 0;
}

```

Файл blackjack.h

```

#ifndef blackjack_h
#define blackjack_h

#include <stdio.h>

typedef struct {
    char rank[3];
    int value;
} Card;

```

```

typedef struct {
    int wins;
    int losses;
    int draw;
} Records;

extern Card deck[52];
extern int deck_index;
extern Records records;
extern const char* RECORDS_FILE;

void init_deck(void);
void Shuffle(void);
Card draw_card(void);
int Score(Card* cards, int num_cards);
void print_cards(Card* cards, int num_cards, int hide_first);
void print_score(Card* cards, int num_cards, int hide_first);
void draw_table(Card* player, int p_cnt, Card* dealer, int d_cnt, int hide_first, int
p_score, int d_score);
void load_records(void);
void save_records(void);
void sh_records(void);
void rules(void);
void menu(void);

void play_game_easy(void);
void play_game_hard(void);

#endif // !blackjack_h

```

Файл easy.c

```

#include <stdio.h>
#include <locale.h>
#include "blackjack.h"

void play_game_easy() {
    Card player_cards[10];
    Card dealer_cards[10];
    int player_count = 0;
    int dealer_count = 0;

    player_cards[player_count++] = draw_card();
    dealer_cards[dealer_count++] = draw_card();
    player_cards[player_count++] = draw_card();
    dealer_cards[dealer_count++] = draw_card();

    int player_score = Score(player_cards, player_count);
    int dealer_score = Score(dealer_cards, dealer_count);

    if (player_score > 21) {
        printf("\n== Ваши карты ==\n");
        print_cards(player_cards, player_count, 0);
        printf("Счёт: %d\n", player_score);
        printf("Не повезло :)! Вы проиграли!\n");
        records.losses++;
        save_records();
        return;
    }

    int choice;
    while (player_score < 21) {
        printf("\n== Ваши карты ==\n");
        print_cards(player_cards, player_count, 0);
        printf("Счёт: %d\n", player_score);

        printf("\n== Карты дилера ==\n");
        print_cards(dealer_cards, dealer_count, 1);
    }
}

```

```

        print_score(dealer_cards, dealer_count, 1);

        printf("\n1. Взять карту\n");
        printf("2. Остаться\n");
        printf("Введите действие: ");
        scanf_s("%d", &choice);

        if (choice == 1) {
            player_cards[player_count++] = draw_card();
            player_score = Score(player_cards, player_count);

            if (player_score > 21) {
                printf("\nВы взяли карту [%s]\n",
player_cards[player_count - 1].rank);
                printf("Счёт: %d\n", player_score);
                printf("Перебор! Вы проиграли!\n");
                records.losses++;
                save_records();
                return;
            }
            printf("\nВы взяли карту [%s]\n", player_cards[player_count -
1].rank);
        }
        else if (choice == 2) {
            break;
        }
        else {
            printf("Неверный Выбор\n");
        }
    }

    printf("\n== Ход дилера ==\n");
    dealer_score = Score(dealer_cards, dealer_count);

    if (dealer_score > 21) {
        printf("Перебор у дилера! Вы победили!\n");
        records.wins++;
        save_records();
        return;
    }

    while (dealer_score < 17) {
        dealer_cards[dealer_count++] = draw_card();
        dealer_score = Score(dealer_cards, dealer_count);
        printf("Дилер взял карту [%s]\n", dealer_cards[dealer_count -
1].rank);

        if (dealer_score > 21) {
            printf("Перебор у дилера! Вы победили!");
            records.wins++;
            save_records();
            return;
        }
    }

    printf("\n== Результаты ==\n");
    printf("Ваши карты: ");
    print_cards(player_cards, player_count, 0);
    printf("\nСчёт: %d\n", player_score);

    printf("\nКарты дилера: ");
    print_cards(dealer_cards, dealer_count, 0);
    printf("\nСчёт дилера: %d\n", dealer_score);

    if (player_score > dealer_score) {
        printf("Вы победили!\n");
        records.wins++;
    }
    else if (player_score < dealer_score) {
        printf("Дилер победил!\n");
        records.losses++;
    }
}

```

```

    }
    else {
        printf("Я не знаю, как у тебя это получилось, но ничья\n");
        records.draw++;
    }
    save_records();
}

```

Файл hard.c

```

#include <stdio.h>
#include <locale.h>
#include "blackjack.h"

void play_game_hard() {
    Card player_cards[10];
    Card dealer_cards[10];
    int player_count = 0;
    int dealer_count = 0;

    player_cards[player_count++] = draw_card();
    dealer_cards[dealer_count++] = draw_card();
    player_cards[player_count++] = draw_card();
    dealer_cards[dealer_count++] = draw_card();

    int player_score = Score(player_cards, player_count);
    int dealer_score = Score(dealer_cards, dealer_count);

    if (player_score > 21) {
        printf("\n== Ваши карты ==\n");
        print_cards(player_cards, player_count, 0);
        printf("Счёт: %d\n", player_score);
        printf("Не повезло :)! Вы проиграли!\n");
        records.losses++;
        save_records();
        return;
    }

    int choice;
    while (player_score < 21) {
        printf("\n== Ваши карты ==\n");
        print_cards(player_cards, player_count, 0);
        printf("Счёт: %d\n", player_score);

        printf("\n== Карты дилера ==\n");
        print_cards(dealer_cards, dealer_count, 1);
        print_score(dealer_cards, dealer_count, 1);

        printf("\n1. Взять карту\n");
        printf("2. Остаться\n");
        printf("Введите действие: ");
        scanf_s("%d", &choice);

        if (choice == 1) {
            player_cards[player_count++] = draw_card();
            player_score = Score(player_cards, player_count);

            if (player_score > 21) {
                printf("\nВы взяли карту [%s]\n",
player_cards[player_count - 1].rank);
                printf("Счёт: %d\n", player_score);
                printf("Перебор! Вы проиграли!\n");
                records.losses++;
                save_records();
                return;
            }
            printf("\nВы взяли карту [%s]\n", player_cards[player_count -
1].rank);
        }
    }
}

```

```

        else if (choice == 2) {
            break;
        }
        else {
            printf("Неверный Выбор\n");
        }
    }

    printf("\n== Ход дилера ==\n");
    dealer_score = Score(dealer_cards, dealer_count);

    if (dealer_score > 21) {
        printf("Перебор у дилера! Вы победили!\n");
        records.wins++;
        save_records();
        return;
    }

    while (dealer_score < 18) {
        dealer_cards[dealer_count++] = draw_card();
        dealer_score = Score(dealer_cards, dealer_count);
        printf("Дилер взял карту [%s]\n", dealer_cards[dealer_count -
1].rank);

        if (dealer_score > 21) {
            printf("Перебор у дилера! Вы победили!");
            records.wins++;
            save_records();
            return;
        }
    }

    printf("\n== Результаты ==\n");
    printf("Ваши карты: ");
    print_cards(player_cards, player_count, 0);
    printf("\nСчёт: %d\n", player_score);

    printf("\nКарты дилера: ");
    print_cards(dealer_cards, dealer_count, 0);
    printf("\nСчёт дилера: %d\n", dealer_score);

    if (player_score > dealer_score) {
        printf("Вы победили!\n");
        records.wins++;
    }
    else if (player_score < dealer_score) {
        printf("Дилер победил!\n");
        records.losses++;
    }
    else {
        printf("Я не знаю, как у тебя это получилось, но ничья\n");
        records.draw++;
    }
    save_records();
}

```