

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Пермский государственный национальный
исследовательский университет»

Институт компьютерных наук и
технологий

ОТЧЁТ

по индивидуальной работе №2
по дисциплине «Язык программирования C++»
Вариант 7

Работу выполнил
студент группы ИТ/О ИТ-18-2024
1 курса Борисова Мария Евгеньевна
«13» июня 2025 г.

Работу проверил
Кухар Дарья Анатольевна
«13» июня 2025 г.

Пермь 2025

Оглавление

Постановка задачи.....	3
Алгоритм решения	4
Тестирование.....	7
Код программы	12

Постановка задачи

Игра в «пьяницу».

В игре в «пьяницу» карточная колода раздается поровну двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта старше, забирает себе обе вскрытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт — проигрывает. Для простоты будем считать, что все карты различны по номиналу, а также, что самая младшая карта побеждает самую старшую карту ("шестерка берет туза"). Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды). Напишите программу, которая моделирует игру в пьяницу и определяет, кто выигрывает.

В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта со значением 0 побеждает карту 9.

Входные данные. Программа получает на вход две строки: первая строка содержит 5 чисел, разделенных пробелами—номера карт первого игрока, вторая — аналогично 5 карт второго игрока. Карты перечислены сверху вниз, то есть каждая строка начинается с той карты, которая будет открыта первой.

Выходные данные. Программа должна определить, кто выигрывает при данной раздаче, и вывести слово `first` или `second`, после чего вывести количество ходов, сделанных до выигрыша. Если на протяжении 1000000 ходов игра не заканчивается, программа должна вывести слово `botva`.

Примеры входные данные 1 3 5 7 9 2 4 6 8 0 выходные данные `second` 5

Алгоритм решения

Для того, чтобы реализовать задачу на языке C++, воспользуемся динамической структурой данных – очередь, данная структура наиболее удобна для реализации игры, поскольку имеет логику, схожую с логикой игры – новые элементы добавляются в конец очереди, а извлекаются — только из её начала. Так же воспользуемся типом данных string - тип данных, значением которого является произвольная последовательность (строка) символов алфавита, и методами данного типа, сделанный выбор облегчит проверку ввода пользователя.

План ведения игры:

1 этап – Инициализация игры:

1. Создаются 2 игрока (player1 и player2) с пустыми колодами карт (реализованы как Queue).
2. Инициализируется массив usedCards[10] для отслеживания, использованных карт (все значения false).

2 этап – Ввод значений карт:

1. Сброс состояния:
 - a. Очищаются очереди игроков.
 - b. Сбрасывается массив usedCards.
2. Ввод карт первого игрока:
 - a. Демонстрируются правила игры.
 - b. Реализуется ввод карт.
 - c. Реализуется проверка ввода, при ошибке, процесс начинается заново.
3. Ввод карт второго игрока:
 - a. Реализуется ввод карт.
 - b. Реализуется проверка ввода, при ошибке, процесс начинается заново.

3 этап – Процесс реализации игры:

1. Цикл игры (пока не пройдено > 1000000 шагов игры и есть карты):
 - a. Извлекаются верхние карты.
 - b. Определяется победитель раунда.
 - c. Победитель забирает карты (сначала карта проигравшего, потом своя).
2. Проверка условия окончания игры:
 - a. Если у player1 нет карт побеждает – player2.
 - b. Если у player2 нет карт побеждает – player1.

с. Выводится количество партий игры, реализованных до её завершения.

Используемые методы и их пояснение.

Методы класса Queue:

Конструктор и деструктор:

Queue() - конструктор, инициализирует очередь.

~Queue() - деструктор, очищает память.

Основные операции с очередью:

PushNumber(int value) - добавляет элемент в конец очереди.

PopNumber() - удаляет и возвращает элемент из начала очереди.

FrontNumber() const - возвращает первый элемент без удаления.

Вспомогательные методы:

IsEmpty() const - проверяет, пуста ли очередь.

size() const - возвращает количество элементов в очереди.

Методы класса GameDrum:

Ввод и проверка данных:

ValidateCard(int card) - проверяет корректность карты.

InputCards(Queue& player, const string& prompt) - обрабатывает ввод карт игрока.

Управление игровым процессом:

ResetG() - сбрасывает состояние игры.

SetCA() - настраивает игру (ввод карт игроков).

PlayCA() - основной игровой цикл.

Вспомогательные функции:

Работа с консолью:

GetMenu() - отображает и обрабатывает главное меню.

main() - точка входа в программу.

Особенности методов:

Все методы Queue работают с внутренней структурой данных (связным списком).

Методы GameDrun используют функциональность Queue для реализации игровой логики.

InputCards() включает сложную валидацию ввода пользователя.

PlayCA() реализует основной игровой цикл с обработкой специальных правил (0 побеждает 9).

Тестирование

Статическое тестирование:

Проводится без реального выполнения программы. Тестировщики анализируют исходный код, документацию и другие элементы. Цель — найти ошибки, недочёты и несоответствия на начальных этапах разработки.

Цель тестирования в рамках варианта:

Выявить потребности и ошибки кода на этапе разработки.

План:

Код программы был отправлен пользователям разного уровня понимания структуры программирования – начинающим программистам, преподавателям и любителям, по итогам их просмотра были даны следующие комментарии

1. «Сделай комментарии чуть дальше от основного кода, чтобы не было каши при визуальном просмотре»
2. «Добавь красивое оформление на правила игры, так будет интереснее»
3. «Распиши подробно правила игры, мне кажется, немного непонятно в чём суть»

Итог тестирования:

Было добавлено обширное пояснение правил игры и их презентабельный вид.

Демонстрация итоговой вариации вида кода после исправлений:

```
1 #include "Queue.h"
2
3 Queue::Queue() : head(nullptr), tail(nullptr), count(0) {} // Конструктор Queue. Список инициализации, который инициализирует три члена класса:
4
5 Queue::~Queue() // Деструктор класса Queue
6 {
7     while (!IsEmpty()) {
8         PopNumber();
9     }
10 }
11
12 void Queue::PushNumber(int value) // Добавление элемента в очередь
13 {
14     Node* newNode = new Node(value);
15     if (IsEmpty()) {
16         head = tail = newNode;
17     }
18     else {
19         tail->next = newNode; // Реализация добавления элемента
20         tail = newNode;
21     }
22     count++;
23 }
24
25 int Queue::PopNumber() // Реализует удаление элемента из начала очереди и возвращает его значение
26 {
27     if (IsEmpty())
28     {
29         return -1;
30     }
31     Node* temp = head;
32     int value = temp->value;
```

Вывод

Показать выходные данные из: Отладка

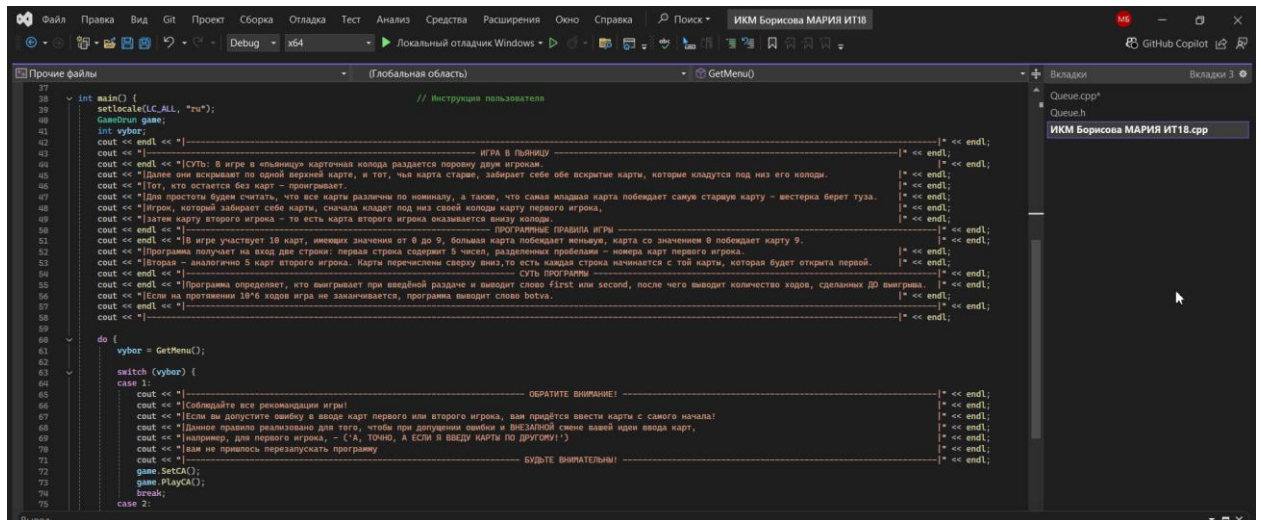
Поток stdout завершился с кодом 3221225786 (0xc000013a).

Поток 19128 завершился с кодом 3221225547 (0xc000004b).

Поток 20900 завершился с кодом 3221225786 (0xc000013a).

Поток 7328 завершился с кодом 3221225786 (0xc000013a).

Программа "[13072] ИИМ Борисова МАРИЯ ИТ18.exe" завершилась с кодом 3221225786 (0xc000013a).



Динамическое тестирование:

Предполагает выполнение кода и анализ его поведения в реальном времени.

Программу запускают и проверяют, как она ведёт себя во время работы. Это помогает обнаружить дефекты, которые проявляются только при запуске программы.

Цель тестирования в рамках варианта:

Выявить дефекты, ошибки программы на ранних этапах.

План:

Код программы был отправлен пользователям продвинутого уровня понимания структуры программирования, по итогам их просмотра, были даны следующие комментарии.

1. «Почему можно вводить одинаковые карты, нельзя же, исправь»
2. «Я случайно в меню нажала 2w, и он работает, так должно быть?»
3. «Я пытаюсь вывести слово botva, какие карты нужно ввести

пользователям, чтобы оно вывелось или оно у тебя никак не выводится и это ошибка?»

Итог тестирования:

Были исправлены несоответствия функций заданию варианта, улучшена проверка на некорректный ввод пользователя.

В случае с выводом слова botva – варианта исхода, при котором партий сделано больше 1000000, возникли сложности. В реализации исправления данной части кода выяснилось, что при условиях «В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта со значением 0 побеждает карту 9», варианта botva не может быть, исход игры всегда будет конечным и количество шагов не столь большим.

Данная проблема решается таким путём:

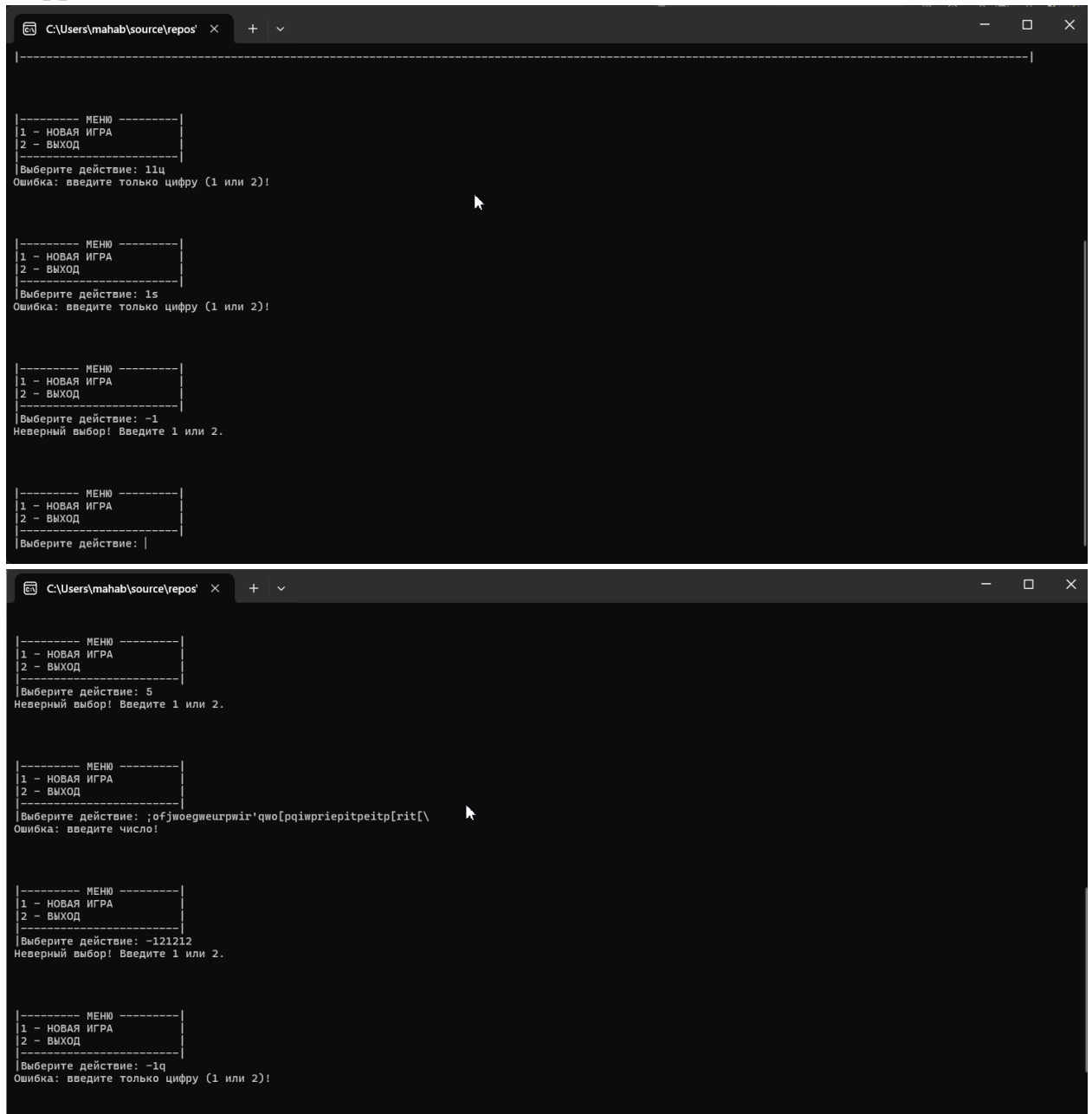
Если изменить правила (например, сделать так, что 0 побеждает все карты, кроме 1), то можно создать бесконечный цикл.

Можно искусственно создать колоды, которые зацикливаются (например, если карты всегда одинаковые после каждого круга).

Но в текущих правилах это невозможно, поэтому проблема вывода слова botva не подлежит исправлению.

Демонстрация итоговой вариации функционала после исправлений:

Корректность ввода пользователя:



```
C:\Users\mahab\source\repos' x + v

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: 11с
Ошибка: введите только цифру (1 или 2)!

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: 1s
Ошибка: введите только цифру (1 или 2)!

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: -1
Неверный выбор! Введите 1 или 2.

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: |

C:\Users\mahab\source\repos' x + v

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: 5
Неверный выбор! Введите 1 или 2.

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: ;ofjwoegweupwir'qwo[pgiwprieptpeitp[ritf\
Ошибка: введите число!

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: -121212
Неверный выбор! Введите 1 или 2.

----- МЕНЮ -----
1 - НОВАЯ ИГРА
2 - ВЫХОД
-----
Выберите действие: -1q
Ошибка: введите только цифру (1 или 2)!
```

```
|----- Ввод карт первого игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений!):
12345
Ошибка:
Нужно ввести ровно 5 УНИКАЛЬНЫХ карт через пробел (ЦИФРЫ ОТ 0 ДО 9)!
Обратите внимание на правила введения карт и попробуйте снова!

|----- Ввод карт первого игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений!):
1a 2d 3 o 5
Нужно ввести ровно 5 УНИКАЛЬНЫХ карт через пробел (ЦИФРЫ ОТ 0 ДО 9)!
Обратите внимание на правила введения карт и попробуйте снова!

|----- Ввод карт первого игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений!):

|----- МЕНЮ -----|
| 1 - НОВАЯ ИГРА
| 2 - ВЫХОД
|-----|
|Выберите действие: 1
|----- ОБРАТИТЕ ВНИМАНИЕ! -----|
|Соблюдайте все рекомендации игры!
|Если вы допустите ошибку в вводе карт первого или второго игрока, вам придётся ввести карты с самого начала!
|данное правило реализовано для того, чтобы при допущении ошибки и ВНЕЗАПНОЙ смене вашей идеи ввода карт,
|например, для первого игрока, - ('А, ТОЧНО, А ЕСЛИ Я ВВЕДУ КАРТЫ ПО ДРУГОМУ!')
|вам не пришлось перезапускать программу
|----- БУДЬТЕ ВНИМАТЕЛЬНЫ! -----|

|----- Ввод карт первого игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений!):
1 2 3 4 5

|----- Ввод карт второго игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений, а так же ЦИФРЫ НЕ МОГУТ СОВПАДАТЬ С ЦИФРАМИ ИГРОКА %1!):
1 2 3 4 5
Ошибка:
Нужно ввести ровно 5 УНИКАЛЬНЫХ карт через пробел (ЦИФРЫ ОТ 0 ДО 9)!
Обратите внимание на правила введения карт и попробуйте снова!

|----- Ввод карт первого игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений!):

|----- Ввод карт первого игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений!):
0 1 2 3 4

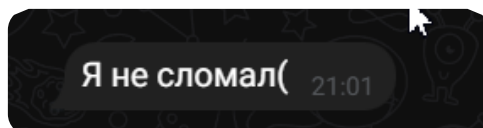
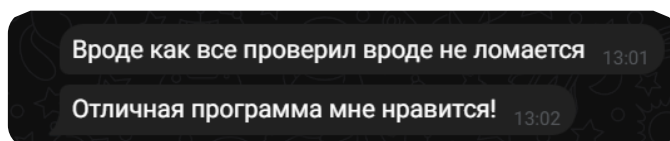
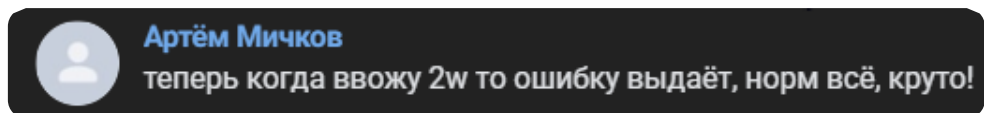
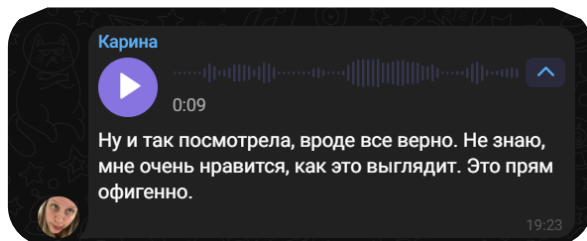
|----- Ввод карт второго игрока -----|
|Введите 5 карт через пробел в одну строку (Введите 5 цифр от 0 до 9 без повторений, а так же ЦИФРЫ НЕ МОГУТ СОВПАДАТЬ С ЦИФРАМИ ИГРОКА %1!):
5 6 7 8 9
second - ПОБЕЖДАЕТ ВТОРОЙ ИГРОК! Количество ходов до победы: 5

|----- МЕНЮ -----|
| 1 - НОВАЯ ИГРА
| 2 - ВЫХОД
|-----|
|Выберите действие:
```

Правильность игры (для скриншота, указанного выше).

	1 игрок 0 1 2 3 4	
	2 игрок 5 6 7 8 9	
	1 партия – побеждает игрок 2	
	Итого:	
	1 игрок: 1 2 3 4	
	2 игрок: 6 7 8 9 5 0	
	2 партия – побеждает игрок 2	
	Итого:	
	1 игрок: 2 3 4	
	2 игрок: 7 8 9 5 0 6 1	
	3 партия – побеждает игрок 2	
	Итого:	
	1 игрок: 3 4	
	2 игрок: 8 9 5 0 6 1 7 2	
	4 партия – побеждает игрок 2	
	Итого:	
	1 игрок: 4	
	2 игрок: 9 5 0 6 1 7 2 8 3	
	5 партия – побеждает игрок 2	
	Итого:	
	1 игрок: пусто	
	2 игрок: 5 0 6 1 7 2 8 3 9 4	

Комментарии тестировщиков после итогового исправления:



Код программы

Ссылка на репозиторий GitHub:

<https://github.com/mariyaborisovait18/MARIYABORISOVAIT18.git>

QR-код на репозиторий GitHub:

