МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ПРАКТИЧЕСКАЯ РАБОТА №3**

**«Алгоритмы и итераторы»**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-324», «АВТФ»  *Аврамчук И.В.*  *«3» июня 2024г*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | Проверил:  Доцент кафедры ЗИ  *Архипова А.Б.*  *«3» июня 2024г*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* |

Новосибирск 2024

**Цели и задачи работы:** Изучение работы библиотеки алгоритмов и работы с итераторами .   
**Задание к работе:** Составить 10 задач ,которые можно решить используя алгоритмы и итераторы. Самостоятельно решить задачи на языке C++.   
**Задача 1 "Перемешивание карт".** Программа, которая будет перемешивать колоду карт и выводить результат.

**Задача 2 "Игра в крестики-нолики с компьютером".** Программа, которая будет играть с пользователем в игру в крестики-нолики, используя простую логику для ходов компьютера.

**Задача 3 "Игра в угадывание чисел".** Программа, которая будет загадывать число от 1 до 100 и предлагать пользователю угадать его. Программа должна давать подсказки, чтобы пользователь мог угадать число за наименьшее количество попыток.

**Задача 4 "Виселица".** Игрок должен угадать слово, задуманное компьютером, за ограниченное количество попыток.

**Задача 5 "Блек-джек".** Игроки играют против дилера, пытаясь набрать 21 очко или как можно ближе, но не больше.

по стобалльной системе. Необходимо преобразовать оценки в пятибальную систему.  **Задача 6 "Казнь".** На лекцию по теории вероятности опоздало несколько человек, которых ждёт наказание в 8 задаче. Необходимо вычислить имена этих счастливчиков.

**Задача 7 "Штурм 502 кабинета".** Студенты не успевают сдавать лабораторные работы и они начинают приходить заранее и занимать очередь. Необходимо найти количество таких студентов. **Задача 8 "Ограбление".** Студенты из АБс-322 провинившись перед преподавателем и теперь должны по 10 биткойнов с человека. Определить сколько человек провинились и посчитать общую сумму долга.

**Задача 9 "Рейтинг студентов".** В НГТУ проводятся ежемесячные экзамены. Нужно определить средний балл каждого студента и найти студента с наивысшим средним баллом.

**Задача 10 "Голодные игры".** В начале года были 4 группы по 35 человек. Но после неуспешно сданной сессии, в каждой группе осталось разное количество студентов. Нужно посчитать среднее количество человек в каждой группе.

**Задача №1**

**Код программы:**

//Задача "Перемешивание карт". Программа, которая будет перемешивать колоду карт и выводить результат.

#include <iostream> // Подключаем библиотеку для работы с вводом и выводом

#include <vector> // Подключаем библиотеку для работы с вектором

#include <algorithm> // Подключаем библиотеку для использования алгоритмов

#include <random> // Подключаем библиотеку для генерации случайных чисел

#include <chrono> // Подключаем библиотеку для работы с временем

#include <map> // Подключаем библиотеку для работы с ассоциативными контейнерами

using namespace std; // Используем пространство имен std, чтобы не писать std:: перед каждым вызовом

// Определение структуры карты

struct Card {

string suit; // Масть карты (например, Черви)

string rank; // Тип карты (например, Туз)

};

// Функция для печати колоды

void printDeck(const vector<Card>& deck) {

for (const auto& card : deck) { // Для каждой карты в колоде

cout << card.rank << " " << card.suit << endl; // Выводим тип и масть карты

}

cout << endl; // Печатаем пустую строку для разделения

}

int main() {

// Определение мастей и типа

vector<string> suits = {"Черви", "Бубны", "Трефы", "Пики"}; // Вектор строк для мастей карт

vector<string> ranks = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Валет", "Дама", "Король", "Туз"}; // Вектор строк для типа карт

// Создание колоды карт

vector<Card> deck; // Вектор для хранения колоды карт

for (const auto& suit : suits) { // Для каждой масти

for (const auto& rank : ranks) { // Для каждого типа

deck.push\_back(Card{suit, rank}); // Добавляем карту в колоду

}

}

// Печать исходной колоды

cout << "Исходная колода:" << endl; // Выводим сообщение о том, что сейчас будет исходная колода

printDeck(deck); // Вызываем функцию для печати колоды

// Перемешивание колоды

auto seed = chrono::system\_clock::now().time\_since\_epoch().count(); // Получаем текущее время в качестве зерна для генератора случайных чисел

shuffle(deck.begin(), deck.end(), default\_random\_engine(seed)); // Перемешиваем колоду, используя случайное зерно

// Печать перемешанной колоды

cout << "Перемешанная колода:" << endl; // Выводим сообщение о том, что сейчас будет перемешанная колода

printDeck(deck); // Вызываем функцию для печати колоды

// Дополнительная задача: Подсчет карт по мастям

map<string, int> suitCount; // Создаем map для подсчета количества карт каждой масти

for (const auto& card : deck) { // Для каждой карты в колоде

suitCount[card.suit]++; // Увеличиваем счетчик для соответствующей масти

}

// Печать результата подсчета

cout << "Подсчет карт по мастям:" << endl; // Выводим сообщение о начале подсчета карт по мастям

for (const auto& entry : suitCount) { // Для каждого элемента в map

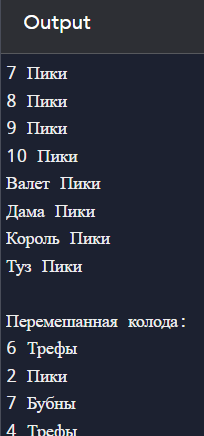
cout << entry.first << ": " << entry.second << endl; // Выводим масть и количество карт этой масти

}

return 0; // Завершаем выполнение программы

}

**Результат работы программы:**



**Задача №2**

**Код программы:**

//Задача "Игра в крестики-нолики с компьютером".

//Программа, которая будет играть с пользователем в игру в крестики-нолики, используя простую логику для ходов компьютера.

#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

using namespace std;

const int SIZE = 3; // Размер игрового поля (3x3)

void initBoard(vector<vector<char>>& board) {

for (int i = 0; i < SIZE; ++i) { // Проход по строкам

vector<char> row(SIZE, ' '); // Создание строки размером SIZE, заполненной пробелами

board.push\_back(row); // Добавление строки в доску

}

}

void printBoard(const vector<vector<char>>& board) {

cout << " 1 2 3 " << endl; // Вывод верхней строки с номерами столбцов

for (int i = 0; i < SIZE; ++i) { // Проход по строкам

cout << i + 1 << " "; // Вывод номера строки

for (int j = 0; j < SIZE; ++j) { // Проход по столбцам

cout << board[i][j]; // Вывод текущего элемента доски

if (j < SIZE - 1) cout << " | "; // Вывод разделителя между элементами (кроме последнего)

}

cout << endl; // Переход на новую строку

if (i < SIZE - 1) cout << " ---+---+---" << endl; // Вывод разделителя между строками (кроме последней)

}

cout << endl; // Пустая строка для удобства чтения

}

bool checkWin(const vector<vector<char>>& board, char player) {

for (int i = 0; i < SIZE; ++i) { // Проверка выигрыша по строкам и столбцам

if (all\_of(board[i].begin(), board[i].end(), [player](char c) { return c == player; }) || // Проверка строки

all\_of(board.begin(), board.end(), [i, player](const vector<char>& row) { return row[i] == player; })) { // Проверка столбца

return true; // Если есть выигрышная строка или столбец, возвращаем true

}

}

if (all\_of(board.begin(), board.end(), [player, n = 0](const vector<char>& row) mutable { return row[n++] == player; }) || // Проверка диагонали 1

all\_of(board.begin(), board.end(), [player, n = 0](const vector<char>& row) mutable { return row[SIZE - 1 - n++] == player; })) { // Проверка диагонали 2

return true; // Если есть выигрыш по диагонали, возвращаем true

}

return false; // Если нет выигрышной комбинации, возвращаем false

}

bool checkDraw(const vector<vector<char>>& board) {

for (const auto& row : board) { // Проход по каждой строке

if (any\_of(row.begin(), row.end(), [](char c) { return c == ' '; })) { // Если есть хотя бы один пустой символ в строке

return false; // Игра не окончена ничьей

}

}

return true; // Если нет пустых ячеек, игра окончена ничьей

}

void computerMove(vector<vector<char>>& board, char symbol) {

srand(time(0)); // Инициализация генератора случайных чисел

int row, col;

do {

row = rand() % SIZE; // Генерация случайной строки

col = rand() % SIZE; // Генерация случайного столбца

} while (board[row][col] != ' '); // Повторять, пока не будет найдена пустая ячейка

board[row][col] = symbol; // Установка символа компьютера в выбранную ячейку

}

void countSymbols(const vector<vector<char>>& board, int& xCount, int& oCount) {

xCount = oCount = 0; // Обнуление счетчиков

for (const auto& row : board) { // Проход по каждой строке

xCount += count(row.begin(), row.end(), 'X'); // Подсчет количества крестиков

oCount += count(row.begin(), row.end(), 'O'); // Подсчет количества ноликов

}

}

int main() {

vector<vector<char>> board; // Доска

initBoard(board); // Инициализация доски

int choice; // Выбор пользователя (кто начнет игру)

cout << "Выберите, кто будет ходить первым (1 - вы, 2 - компьютер): "; // Запрос выбора пользователя

cin >> choice; // Ввод выбора пользователя

char playerSymbol, computerSymbol; // Символы игроков

if (choice == 1) { // Если выбрано начать игру с пользователем

cout << "Выберите символ (X - крестик, O - нолик): "; // Запрос выбора символа

cin >> playerSymbol; // Ввод символа пользователя

computerSymbol = (playerSymbol == 'X') ? 'O' : 'X'; // Выбор символа для компьютера

} else { // Если выбрано начать игру с компьютером

srand(time(0)); // Инициализация генератора случайных чисел

computerSymbol = (rand() % 2 == 0) ? 'X' : 'O'; // Случайный выбор символа для компьютера

playerSymbol = (computerSymbol == 'X') ? 'O' : 'X'; // Определение символа для пользователя

cout << "Компьютер играет за " << (computerSymbol == 'X' ? "крестики" : "нолики") << endl; // Вывод выбранной стороны компьютера

cout << "Вы играете за " << (playerSymbol == 'X' ? "крестики" : "нолики") << endl; // Вывод выбранной стороны пользователя

}

char currentPlayer = (choice == 1) ? playerSymbol : computerSymbol; // Определение, чей ход первый

bool gameOver = false; // Переменная для отслеживания завершения игры

int moveCount = 0; // Счетчик ходов

while (!gameOver) { // Главный цикл игры

printBoard(board); // Вывод текущего состояния доски

moveCount++; // Увеличение счетчика ходов

cout << "Ход №" << moveCount << ". Сейчас ходит: " << (currentPlayer == playerSymbol ? "вы" : "компьютер") << endl; // Вывод текущего игрока

if (currentPlayer == playerSymbol) { // Если ходит игрок

int row, col; // Переменные для хода игрока

cout << "Введите ваш ход (строка и столбец): "; // Приглашение к вводу хода

cin >> row >> col; // Ввод строки и столбца

row--; // Приведение номера строки к индексу массива

col--; // Приведение номера столбца к индексу массива

if (row >= 0 && row < SIZE && col >= 0 && col < SIZE && board[row][col] == ' ') { // Проверка корректности хода

board[row][col] = currentPlayer; // Установка символа игрока на доску

if (checkWin(board, currentPlayer)) { // Проверка на победу

printBoard(board); // Вывод окончательного состояния доски

cout << (currentPlayer == playerSymbol ? "Вы победили!" : "Компьютер победил!") << endl; // Вывод победителя

gameOver = true; // Игра завершена

} else if (checkDraw(board)) { // Проверка на ничью

printBoard(board); // Вывод окончательного состояния доски

cout << "Ничья!" << endl; // Вывод сообщения о ничьей

gameOver = true; // Игра завершена

}

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X'; // Смена игрока

} else { // Если ход некорректен

cout << "Некорректный ход. Попробуйте снова." << endl; // Вывод сообщения об ошибке

moveCount--; // Уменьшение счетчика ходов

}

} else { // Если ходит компьютер

computerMove(board, currentPlayer); // Ход компьютера

if (checkWin(board, currentPlayer)) { // Проверка на победу компьютера

printBoard(board); // Вывод окончательного состояния доски

cout << (currentPlayer == playerSymbol ? "Вы победили!" : "Компьютер победил!") << endl; // Вывод победителя

gameOver = true; // Игра завершена

} else if (checkDraw(board)) { // Проверка на ничью

printBoard(board); // Вывод окончательного состояния доски

cout << "Ничья!" << endl; // Вывод сообщения о ничьей

gameOver = true; // Игра завершена

}

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X'; // Смена игрока

}

}

int xCount, oCount; // Переменные для подсчета количества символов на доске

countSymbols(board, xCount, oCount); // Подсчет символов на доске

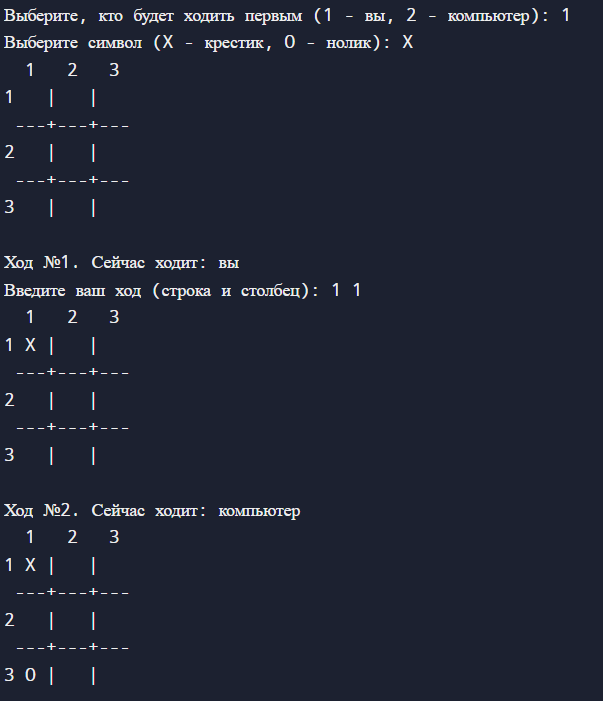
cout << "Количество крестиков (X): " << xCount << endl; // Вывод количества крестиков

cout << "Количество ноликов (O): " << oCount << endl; // Вывод количества ноликов

return 0; // Возврат нуля, как признак успешного завершения программы

}

**Результат работы программы:**



**Задача №3**

**Код программы:**

//Задача "Игра в угадывание чисел"

//Программа, которая будет загадывать число от 1 до 100 и предлагать пользователю угадать его.

//Программа должна давать подсказки, чтобы пользователь мог угадать число за наименьшее количество попыток.

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <algorithm>

using namespace std;

int main() {

srand(time(0)); // Инициализация генератора случайных чисел

int secretNumber = rand() % 100 + 1; // Загаданное число от 1 до 100

int guess, attempts = 0; // Попытка пользователя и счетчик попыток

cout << "Компьютер загадал число от 1 до 100. Попробуй угадать." << endl;

do {

cout << "Введите ваше предположение: ";

cin >> guess;

++attempts; // Увеличиваем счетчик попыток

if (guess < secretNumber) {

cout << "Загаданное число больше." << endl;

} else if (guess > secretNumber) {

cout << "Загаданное число меньше." << endl;

} else {

cout << "Поздравляю! Вы угадали число " << secretNumber << " за " << attempts << " попыток." << endl;

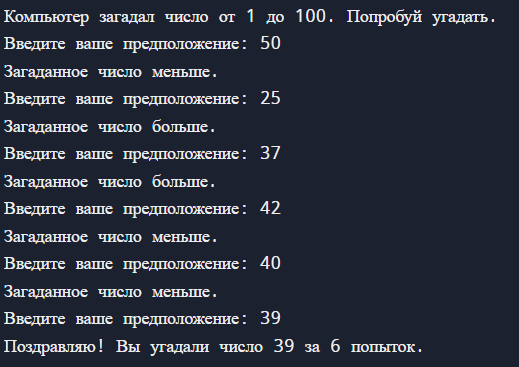
}

} while (guess != secretNumber);

return 0;

}

**Результат работы программы:**



**Задача №4**

**Код программы:**

//Виселица: игрок должен угадать слово, задуманное компьютером, за ограниченное количество попыток.

#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

#include <cstdlib>

using namespace std;

// Функция для выбора случайного слова из списка

string chooseWord() {

vector<string> words = {"apple", "banana", "orange", "grape", "kiwi", "peach"};

srand(time(0));

return words[rand() % words.size()];

}

// Функция для отображения угаданных и неугаданных букв слова

void displayWord(const string& word, const vector<char>& guessedLetters) {

for (char letter : word) {

if (find(guessedLetters.begin(), guessedLetters.end(), letter) != guessedLetters.end()) {

cout << letter << " ";

} else {

cout << "\_ ";

}

}

cout << endl;

}

int main() {

const int maxAttempts = 6; // Максимальное количество попыток

string word = chooseWord(); // Выбираем слово для угадывания

int attemptsLeft = maxAttempts; // Количество оставшихся попыток

vector<char> guessedLetters; // Вектор для хранения угаданных букв

cout << "Добро пожаловать в игру 'Виселица'!" << endl;

while (attemptsLeft > 0) {

cout << "Угадайте слово: ";

displayWord(word, guessedLetters);

// Проверяем, было ли угадано всё слово

if (find\_if(word.begin(), word.end(), [&](char letter) {

return find(guessedLetters.begin(), guessedLetters.end(), letter) == guessedLetters.end();

}) == word.end()) {

cout << "Поздравляем! Вы угадали слово '" << word << "'!" << endl;

break;

}

// Запрос ввода буквы от пользователя

char guess;

cout << "Введите букву: ";

cin >> guess;

// Проверяем, была ли уже введена эта буква

if (find(guessedLetters.begin(), guessedLetters.end(), guess) != guessedLetters.end()) {

cout << "Вы уже вводили эту букву." << endl;

continue;

}

// Добавляем введенную букву в список угаданных

guessedLetters.push\_back(guess);

// Проверяем, есть ли введенная буква в загаданном слове

if (word.find(guess) == string::npos) {

cout << "Такой буквы нет в слове. Осталось попыток: " << --attemptsLeft << endl;

}

}

if (attemptsLeft == 0) {

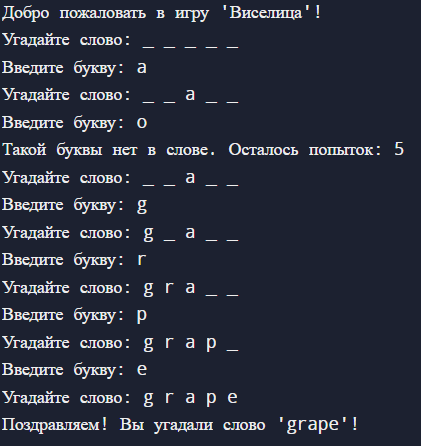
cout << "Извините, вы проиграли. Загаданное слово было: " << word << endl;

}

return 0;

}

**Результат работы программы:**



**Задача №5**

**Код программы:**

//Игра "Блек-джек": Игроки играют против дилера, пытаясь набрать 21 очко или как можно ближе, но не больше.

#include <iostream> // Подключение библиотеки для ввода-вывода

#include <vector> // Подключение библиотеки для работы с векторами

#include <algorithm> // Подключение библиотеки алгоритмов

#include <ctime> // Подключение библиотеки для работы со временем

#include <cstdlib> // Подключение стандартной библиотеки C

using namespace std; // Использование стандартного пространства имен

// Функция для подсчета очков в руке

int countPoints(const vector<int>& hand) {

int sum = 0; // Инициализация суммы очков

int aces = 0; // Инициализация количества тузов

// Использование алгоритма for\_each для подсчета очков

for\_each(hand.begin(), hand.end(), [&](int card) {

if (card >= 10) {

sum += 10; // Карты с номиналом 10 и выше дают 10 очков

} else if (card == 1) {

aces++; // Увеличиваем количество тузов

sum += 11; // Сначала считаем туз как 11

} else {

sum += card; // Добавляем номинал карты к сумме

}

});

// Если перебор и есть тузы, уменьшаем сумму

while (sum > 21 && aces > 0) {

sum -= 10; // Уменьшаем сумму на 10

aces--; // Уменьшаем количество тузов

}

return sum; // Возвращаем сумму очков

}

// Функция для вывода карты

string cardToString(int card) {

if (card == 1) return "A"; // Если карта - туз, возвращаем "A"

if (card == 11) return "J"; // Если карта - валет, возвращаем "J"

if (card == 12) return "Q"; // Если карта - дама, возвращаем "Q"

if (card == 13) return "K"; // Если карта - король, возвращаем "K"

return to\_string(card); // Возвращаем номинал карты как строку

}

// Функция для вывода руки игрока или дилера

void printHand(const vector<int>& hand, const string& name) {

cout << name << " карты: "; // Выводим имя игрока

// Используем итераторы для обхода вектора карт

for (auto it = hand.begin(); it != hand.end(); ++it) {

cout << cardToString(\*it) << " "; // Выводим карту

}

// Выводим сумму очков в руке

cout << "(Очки: " << countPoints(hand) << ")" << endl;

}

int main() {

// Инициализируем генератор случайных чисел текущим временем

srand(static\_cast<unsigned int>(time(0)));

// Колода карт

vector<int> deck;

for (int i = 1; i <= 13; ++i) { // Для каждого номинала карт от 1 до 13

for (int j = 0; j < 4; ++j) { // Для каждой из четырех мастей

deck.push\_back(i); // Добавляем карту в колоду

}

}

// Перемешиваем колоду с помощью random\_shuffle

random\_shuffle(deck.begin(), deck.end());

vector<int> playerHand; // Рука игрока

vector<int> dealerHand; // Рука дилера

// Раздаем по две карты игроку и дилеру

playerHand.push\_back(deck.back()); // Добавляем карту игроку

deck.pop\_back(); // Удаляем карту из колоды

dealerHand.push\_back(deck.back()); // Добавляем карту дилеру

deck.pop\_back(); // Удаляем карту из колоды

playerHand.push\_back(deck.back()); // Добавляем карту игроку

deck.pop\_back(); // Удаляем карту из колоды

dealerHand.push\_back(deck.back()); // Добавляем карту дилеру

deck.pop\_back(); // Удаляем карту из колоды

// Выводим карты игрока

printHand(playerHand, "Ваши"); //printHand - итератор

// Выводим первую карту дилера

cout << "Первая карта дилера: " << cardToString(dealerHand[0]) << endl;

// Ход игрока

while (true) {

cout << "Хотите взять еще карту или остановиться? (Y/N): ";

char choice; // Переменная для хранения выбора игрока

cin >> choice; // Ввод выбора

if (choice == 'Y') {

playerHand.push\_back(deck.back()); // Добавляем карту игроку

deck.pop\_back(); // Удаляем карту из колоды

printHand(playerHand, "Ваши"); // Выводим карты игрока

if (countPoints(playerHand) > 21) { // Проверяем перебор

cout << "Перебор! Вы проиграли." << endl;

return 0; // Завершаем игру

}

} else if (choice == 'N') {

break; // Игрок остановился, выходим из цикла

}

}

// Ход дилера

printHand(dealerHand, "Карты дилера"); // Выводим карты дилера

while (countPoints(dealerHand) < 17) { // Дилер берет карту, если меньше 17 очков

dealerHand.push\_back(deck.back()); // Добавляем карту дилеру

deck.pop\_back(); // Удаляем карту из колоды

printHand(dealerHand, "Карты дилера"); // Выводим карты дилера

}

int playerPoints = countPoints(playerHand); // Очки игрока

int dealerPoints = countPoints(dealerHand); // Очки дилера

// Определение победителя

if (dealerPoints > 21 || playerPoints > dealerPoints) {

cout << "Вы выиграли!" << endl; // Игрок выиграл

} else if (playerPoints < dealerPoints) {

cout << "Дилер выиграл!" << endl; // Дилер выиграл

} else {

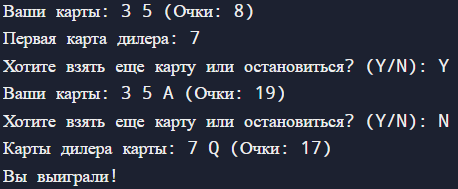
cout << "Ничья!" << endl; // Ничья

}

return 0; // Завершаем программу

}

**Результат работы программы:**



**Задача №6**

**Код программы:**

//Задача "Казнь": На лекцию по терверу опоздало несколько человек, которых ждёт расстрел, бан и долг 10 биткоинов.

//Необходимо вычислить имена этих счастливчиков.

#include <iostream> // Для ввода/вывода

#include <vector> // Для использования векторов

#include <algorithm> // Для использования алгоритмов

#include <ctime> // Для инициализации генератора случайных чисел

#include <cstdlib> // Для генерации случайных чисел

using namespace std; // Использование пространства имен std

int main() {

// Имена студентов

vector<string> names = {"Ilya", "Vlad", "Sasha", "Nikita", "Andrey", "Petya"};

// Инициализация генератора случайных чисел

srand(static\_cast<unsigned int>(time(0)));

// Выбираем случайное количество опоздавших (от 1 до 6)

int numLate = rand() % names.size() + 1;

// Перемешиваем имена студентов

random\_shuffle(names.begin(), names.end());

// Определяем опоздавших студентов

vector<string> lateStudents(names.begin(), names.begin() + numLate);

// Считаем количество опоздавших студентов

int lateCount = count\_if(lateStudents.begin(), lateStudents.end(), [](const string& name) { return !name.empty(); });

// Выводим имена всех студентов

cout << "Студенты: ";

for (const auto& name : names) { // Использование итераторов для перебора всех студентов

cout << name << " ";

}

cout << endl;

// Выводим имена опоздавших студентов

cout << "Опоздавшие студенты: ";

for (const auto& name : lateStudents) { // Использование итераторов для перебора опоздавших студентов

cout << name << " ";

}

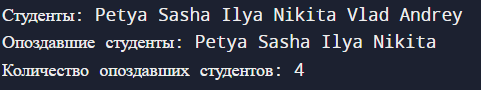
cout << endl;

// Выводим количество опоздавших студентов

cout << "Количество опоздавших студентов: " << lateCount << endl;

return 0;

}

**Результат работы программы:**

**Задача № 7**

**Код программы:**

//Задача "Штурм 502 кабинета": Студенты не успевают сдавать лабораторные работы и они начинают приходить заранее и занимать очередь.

//Необходимо найти количество таких студентов.

#include <iostream> // Для ввода/вывода

#include <vector> // Для использования векторов

#include <algorithm> // Для использования алгоритмов

#include <ctime> // Для инициализации генератора случайных чисел

#include <cstdlib> // Для генерации случайных чисел

#include <numeric> // Для использования функции accumulate

using namespace std; // Использование пространства имен std

int main() {

const int totalStudents = 150; // Всего студентов

const int days = 10; // Количество дней

vector<int> studentsPerDay(days); // Вектор для хранения количества студентов по дням

// Инициализация генератора случайных чисел

srand(static\_cast<unsigned int>(time(0)));

// В 1 день приходит 2 человека

studentsPerDay[0] = 2;

// Генерация количества студентов для последующих дней

for (int i = 1; i < days; ++i) {

// Количество студентов в текущий день должно быть больше, чем в предыдущий день

studentsPerDay[i] = studentsPerDay[i - 1] + rand() % ((totalStudents - studentsPerDay[i - 1]) / (days - i) + 1);

}

// Вывод количества студентов по дням

cout << "Количество студентов по дням:" << endl;

for (int i = 0; i < days; ++i) {

cout << "День " << i + 1 << ": " << studentsPerDay[i] << " студентов" << endl;

}

// Подсчет общего количества студентов за все дни

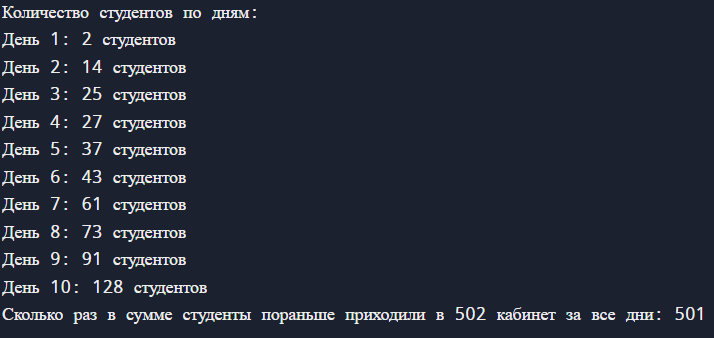
int totalCount = accumulate(studentsPerDay.begin(), studentsPerDay.end(), 0);

cout << "Сколько раз в сумме студенты пораньше приходили в 502 кабинет за все дни: " << totalCount << endl;

return 0;

}

**Результат работы программы:**



**Задача №8**

**Код программы:**

//Задача "Ограбление". Студенты из АБс-322 провинившись перед преподавателем задолжали по 10 биткойнов с человека.

//Определить сколько человек провинились и посчитать общую сумму долга.

#include <iostream> // Для ввода/вывода

#include <vector> // Для использования векторов

#include <algorithm> // Для использования алгоритмов

#include <ctime> // Для инициализации генератора случайных чисел

#include <cstdlib> // Для генерации случайных чисел

#include <numeric> // Для использования функции accumulate

#include <iomanip> // Для управления форматированием вывода

using namespace std; // Использование пространства имен std

int main() {

const int totalStudents = 32; // Общее количество студентов

const int debtPerStudent = 10; // Долг на одного студента в биткойнах

const double bitcoinToUSD = 67663.0; // Стоимость 1 биткойна в долларах

// Инициализация генератора случайных чисел

srand(static\_cast<unsigned int>(time(0)));

// Случайное определение количества провинившихся студентов

int numGuiltyStudents = rand() % totalStudents + 1;

// Создание вектора с долгами студентов

vector<int> debts(totalStudents, 0);

fill\_n(debts.begin(), numGuiltyStudents, debtPerStudent);

// Подсчет общего долга в биткойнах

int totalDebtBitcoin = accumulate(debts.begin(), debts.end(), 0);

// Подсчет общего долга в долларах

double totalDebtUSD = totalDebtBitcoin \* bitcoinToUSD;

// Вывод результатов

cout << "Количество провинившихся студентов: " << numGuiltyStudents << endl;

cout << "Общий долг в биткойнах: " << totalDebtBitcoin << " BTC" << endl;

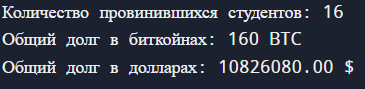
cout << fixed << setprecision(2); // Установка формата вывода для чисел с плавающей точкой

cout << "Общий долг в долларах: " << totalDebtUSD << " $" << endl;

return 0;

}

**Результат работы программы:**



**Задание №9**

**Код программы:**

//Задача "Рейтинг студентов": В НГТУ проводятся ежемесячные экзамены.

//Нужно определить средний балл каждого студента и найти студента с наивысшим средним баллом.

#include <iostream> // Подключаем библиотеку для ввода/вывода

#include <vector> // Подключаем библиотеку для использования векторов

#include <algorithm> // Подключаем библиотеку для использования алгоритмов

#include <numeric> // Подключаем библиотеку для использования функции accumulate

using namespace std; // Используем пространство имен std

// Структура для хранения информации о студенте

struct Student {

string name; // Имя студента

vector<int> scores; // Вектор оценок студента

};

int main() {

// Создаем вектор студентов с их именами и оценками

vector<Student> students = {

{"Налитов", {85, 90, 78, 92, 88}},

{"Андриянов", {80, 85, 84, 79, 90}},

{"Плисова", {90, 92, 88, 91, 85}}

};

string topStudent; // Имя студента с наивысшим средним баллом

double highestAverage = 0.0; // Наивысший средний балл

// Перебираем всех студентов

for (const auto& student : students) {

// Вычисляем средний балл студента

double average = accumulate(student.scores.begin(), student.scores.end(), 0.0) / student.scores.size();

// Если средний балл текущего студента выше наивысшего среднего балла, обновляем значения

if (average > highestAverage) {

highestAverage = average;

topStudent = student.name;

}

}

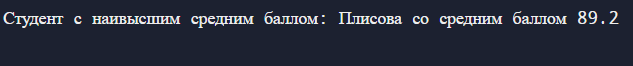
// Выводим имя студента с наивысшим средним баллом и сам средний балл

cout << "Студент с наивысшим средним баллом: " << topStudent << " со средним баллом " << highestAverage << endl;

return 0; // Завершаем программу

}

**Результат работы программы:**



**Задача №10**

**Код программы:**

//Задача "Голодные игры": В начале года были 4 группы по 35 человек.

//Но после неуспешно сданной сессии, в каждой группе осталось разное количество студентов.

//Нужно посчитать среднее количество человек в каждой группе.

#include <iostream> // Подключаем библиотеку для ввода/вывода

#include <vector> // Подключаем библиотеку для использования векторов

#include <algorithm> // Подключаем библиотеку для использования алгоритмов

#include <numeric> // Подключаем библиотеку для использования функции accumulate

#include <ctime> // Подключаем библиотеку для инициализации генератора случайных чисел

#include <cstdlib> // Подключаем библиотеку для генерации случайных чисел

using namespace std; // Использование пространства имен std

int main() {

const int initialStudentsPerGroup = 35; // Количество студентов в начале года в каждой группе

const int totalGroups = 4; // Общее количество групп

// Инициализация генератора случайных чисел

srand(static\_cast<unsigned int>(time(0)));

// Вектор для хранения количества студентов в каждой группе

vector<int> studentsPerGroup(totalGroups, initialStudentsPerGroup);

// Случайное определение количества покинувших студентов для каждой группы

int totalLeft = 0; // Переменная для хранения общего числа покинувших студентов

for (auto& students : studentsPerGroup) {

int left = rand() % (students / 2); // Генерируем количество покинувших (до половины группы)

students -= left; // Уменьшаем количество студентов на количество покинувших

totalLeft += left; // Увеличиваем общий счетчик покинувших студентов

}

// Вывод количества студентов в каждой группе после сессии

cout << "Количество студентов в каждой группе после сессии:" << endl;

for (int i = 0; i < totalGroups; ++i) {

cout << "Группа " << i + 1 << ": " << studentsPerGroup[i] << " студентов" << endl;

}

// Подсчет среднего количества студентов в каждой группе

double averageStudents = accumulate(studentsPerGroup.begin(), studentsPerGroup.end(), 0.0) / totalGroups;

// Вывод среднего количества студентов

cout << "Среднее количество студентов в каждой группе: " << averageStudents << endl;

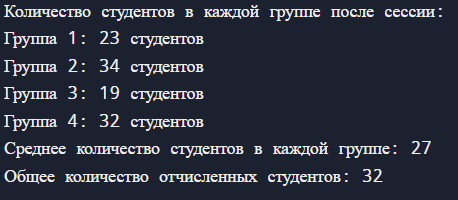
// Вывод общего количества отчисленных студентов

cout << "Общее количество отчисленных студентов: " << totalLeft << endl;

return 0;

}

**Результат работы программы:**



**Вывод**

В ходе практической работы была выполнена поставленная задача на изучение работы библиотеки алгоритмов и работы с итераторами. Были составлены 20 задач и приведена их реализация на языке C++.