**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №8

«Стандартная библиотека шаблонов»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-23Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Правдина А.Д. |
| Подпись и дата: 31.05.2022 |  | Подпись и дата: 31.05.2022 |

Москва, 2022 г.

**Постановка задачи**

**Задание 1**

Пользователь вводит два слова (две переменные). Сколько раз в первом слове встречается последняя буква второго слова.

**Задание 2**

Пользователь вводит некоторый текст. Образовать из него новый, в который включить информацию, заключенную между пробелом и запятой.

**Задание 3**

Дан текстовый файл. Запишите в другой файл содержимое исходного файла, нумеруя каждую строку.

**Задание 4**

Создайте три объекта vector<float> и заполните первые два объекта так, как в предыдущем примере. Напишите цикл for, который суммирует соответствующие элементы первых двух векторов и заносит результат в соответствующий элемент третьего вектора. Выведите содержимое всех трех векторов. Переделать программу: печатать векторы с помощью функции.

**Задание 5**

Создать класс с именем Planet для хранения следующей информации:

название (string), количество спутников.

Создать массив планет с помощью контейнера vector в main().

Занести в вектор информацию о 9 планетах. Данные о планетах:

"Меркурий", 0

"Венера", 0

"Земля", 1

"Марс", 2

"Юпитер", 69

"Сатурн", 62

"Уран", 27

"Нептун", 14

"Плутон", 10

Вывести массив планет на экран, используя функцию.

В цикле определите элемент, у которого максимальное количество спутников.

**Задание 6**

Описать функцию, которая считает количество положительных элементов заданного контейнера-списка list <int>, а затем распечатывает это значение (выдает в стандартный поток cout). Продемонстрировать её работу.

**Задание 7**

Создать класс:

class Card {

string title; // заглавие книги

string author; // автор

int number; // количество имеющихся экземпляров

public:

Card() : title(""), author(""), number(0) {}

Card(string t, string a, int n) : title(t), author(a), number(0) {}

};

В программе создать list из объектов класса Card (5 элементов).

Распечатать список. Пользователь вводит автора, программа выводит книги автора или сообщение об их отсутствии.

**Задание 8**

Создайте отображение map<int,double> и занесите в него пары 1 - 1.1, 2 - 2.2, 3 - 3.3, 4 - 4.4, 5 - 5.5 . . . 9 - 9.9. Выведите содержимое отображения на экран. Удалить пары с четными ключами и снова распечатайте отображение.  
**Задание 9**

В отображении находятся записи о соответствии номеров телефона (7 цифр) и фамилии. Например:

5671234 Ivanov

3214567 Petrov

9871234 Sidorov

В массиве находятся номера телефонов. Например:

string mas[4] = { "1112233", "9871234", "5671234", "5556688" };

Осуществить поиск фамилий из заданного отображения, соответствующих телефонам из массива mas. Найденные фамилии вывести на консоль. В данном примере:

Sidorov

Ivanov

**Текст программы**

**Листинг кода программы:**

**main.cpp**

*#include* "tasks.hpp"  
  
*int* main() {  
 task1();  
 task2();  
 task3();  
 task4();  
 task5();  
 task6();  
 task7();  
 task8();  
 task9();  
 *return* 0;  
}

**tasks.hpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_TASKS\_HPP  
*#define* CODE\_TASKS\_HPP  
  
  
*#include* <iostream>  
*#include* <algorithm>  
*#include* <fstream>  
*#include* <vector>  
*#include* <list>  
*#include* <map>  
*#include* <cmath>  
*#include* "Planet.hpp"  
*#include* "Card.hpp"  
  
*void* printText(*int*);  
  
*void* task1();  
  
*void* task2();  
  
*void* task3();  
  
*void* task4();  
  
*void* task5();  
  
*void* task6();  
  
*void* task7();  
  
*void* task8();  
  
*void* task9();  
  
*#endif //CODE\_TASKS\_HPP*

**task.cpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
#include* "tasks.hpp"  
  
*void* printText(*int* a) {  
 std::cout << "\n\n\t\t\t\tTASK №" << a << '\n';  
}  
  
*template*<*class* T>  
*void* printVector(std::vector<T> vector) {  
 std::cout << '\n';  
 *for* (*auto* element: vector) {  
 std::cout << element << ' ';  
 }  
 std::cout << '\n';  
}  
  
*void* task1() {  
 printText(1);  
 std::string s1, s2;  
 std::cin >> s1 >> s2;  
 *int* q = std::count(s2.begin(), s2.end(), s1[s1.length() - 1]);  
 std::cout << q;  
}  
  
*void* task2() {  
 printText(2);  
 std::string str;  
 std::cin.ignore();  
 getline(std::cin, str); *// не вводит* std::cout << str.substr(str.find(' ') + 1, str.find(',') - str.find(' ') - 1);  
}  
  
*void* task3() {  
*//12. Дан текстовый файл. Запишите в другой файл содержимое исходного файла, нумеруя каждую строку.* std::string s; *// сюда будем класть считанные строки* std::ifstream fin("input.txt");  
 std::ofstream fout("output.txt");  
 size\_t i = 0;  
 *while* (getline(fin, s)) {  
 fout << i++ << ". " << s << '\n';  
 }  
 fin.close();  
 fout.close();  
}  
  
*void* task4() {  
 printText(4);  
 std::vector<*float*> v1, v2, v3;  
 *for* (size\_t i = 0; i < 25; ++i) {  
 v1.push\_back(rand() % 150 \* (*float*) (rand() % 10) / 10);  
 v2.push\_back(rand() % 150 \* (*float*) (rand() % 10) / 10);  
 v3.push\_back(v1[i] + v2[i]);  
 }  
 printVector(v1);  
 printVector(v2);  
 printVector(v3);  
}  
  
*void* task5() {  
  
 printText(5);  
 std::vector<Planet> planets{Planet("Меркурий", 0), Planet("Венера", 0), Planet("Земля", 1), Planet("Марс", 2),  
 Planet("Юпитер", 69), Planet("Сатурн", 62), Planet("Уран", 27), Planet("Нептун", 14),  
 Planet("Плутон", 10)};  
  
 *for* (*auto* planet: planets) {  
 std::cout << planet;  
 }  
 std::cout << "Планета с максимальным количеством спутников:\n";  
 std::cout << \*(std::max\_element(planets.begin(), planets.end(), [](*const* Planet &planet1, *const* Planet &planet2) {  
 *return* planet1.SattelitesNumber < planet2.SattelitesNumber;  
 }));  
}  
  
*void* task6() {  
 printText(6);  
 std::list<*int*> list;  
 *for* (size\_t i = 0; i < 25; ++i) {  
 list.push\_back((*int*) pow(-1, rand() % 2) \* rand() % 145);  
 }  
 std::cout << "list:\n";  
 *for* (*auto* number: list) {  
 std::cout << number << ' ';  
 }  
 std::cout << '\n' << std::count\_if(list.begin(), list.end(), [](*int* number) { *return* number > 0; });  
}  
  
*void* task7() {  
 printText(7);  
 std::list<Card> cards{Card("На западном фронте без перемен", "Эрих Мария Ремарк", 1232),  
 Card("Мартин Иден", "Джек Лондон", 0),  
 Card("Великий Гэтсби", "Фрэнсис Скотт Фицджеральд", 122),  
 Card("Скотный двор", "Джордж Оруэлл", 888),  
 Card("Повелитель мух", "Уильям Голдинг", 123)};  
 *for* (*auto* card: cards) {  
 std::cout << card;  
 }  
 std::string author;  
 std::cout << "Введите имя автора:\n";  
 std::getline(std::cin, author);  
 std::list<Card> searchResult;  
 *for* (*auto* card: cards) {  
 *if* (card.author == author) {  
 searchResult.push\_back(card);  
 }  
 }  
 *if* (searchResult.empty()) {  
 std::cout << "Книг данного автора нет";  
 } *else* {  
 std::cout << "Найдены следующие книги:\n";  
 *for* (*auto* card: searchResult) {  
 std::cout << card;  
 }  
 }  
}  
  
*void* task8() {  
 printText(8);  
 std::map<*int*, *double*> numbers;  
 *for* (*int* i = 0; i < 10; ++i) {  
 numbers.insert(std::make\_pair(i, (i + *static\_cast*<*double*>(i) / 10)));  
 }  
 *for* (*auto* number: numbers) {  
 std::cout << number.first << " - " << number.second << '\n';  
 }  
 *for* (*int* i = 0; i < 10; i += 2) {  
 numbers.erase(numbers[i]);  
 }  
 std::cout << "После удаления:\n";  
 *for* (*auto* number: numbers) {  
 std::cout << number.first << " - " << number.second << '\n';  
 }  
}  
  
*void* task9() {  
 printText(9);  
 std::map<std::string, std::string> Phones{{"5671234", "Ivanov"},  
 {"3214567", "Petrov"},  
 {"9871234", "Sidorov"}};  
 std::string mas[4] = {"1112233", "9871234", "5671234", "5556688"};  
 *for* (size\_t i = 0; i < 4; ++i) {  
 *if* (Phones.find(mas[i]) != Phones.end()) {  
 std::cout << Phones[mas[i]] << '\n';  
 }  
 }  
}

**Planet.hpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_PLANET\_HPP  
*#define* CODE\_PLANET\_HPP  
  
*#include* <string>  
*#include* <iostream>  
*#include* <iterator>  
  
*class* Planet {  
*public*:  
 std::string Name;  
 size\_t SattelitesNumber;  
  
 Planet(std::string name, size\_t sattelites\_number) : Name(name), SattelitesNumber(sattelites\_number){};  
  
 *friend* std::ostream &*operator*<<(std::ostream &, Planet &);  
};  
  
  
  
*#endif //CODE\_PLANET\_HPP*

**Card.cpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_CARD\_HPP  
*#define* CODE\_CARD\_HPP  
  
*#include* <string>  
*#include* <iostream>  
  
*class* Card {  
 std::string title; *// заглавие книги  
 // автор  
 int* number; *// количество имеющихся экземпляров  
public*:  
 Card() : title(""), author(""), number(0) {}  
  
 Card(std::string t, std::string a, *int* n) : title(t), author(a), number(n) {}  
  
 *friend* std::ostream &*operator*<<(std::ostream &, Card &);  
  
 std::string author;  
};  
  
  
*#endif //CODE\_CARD\_HPP*

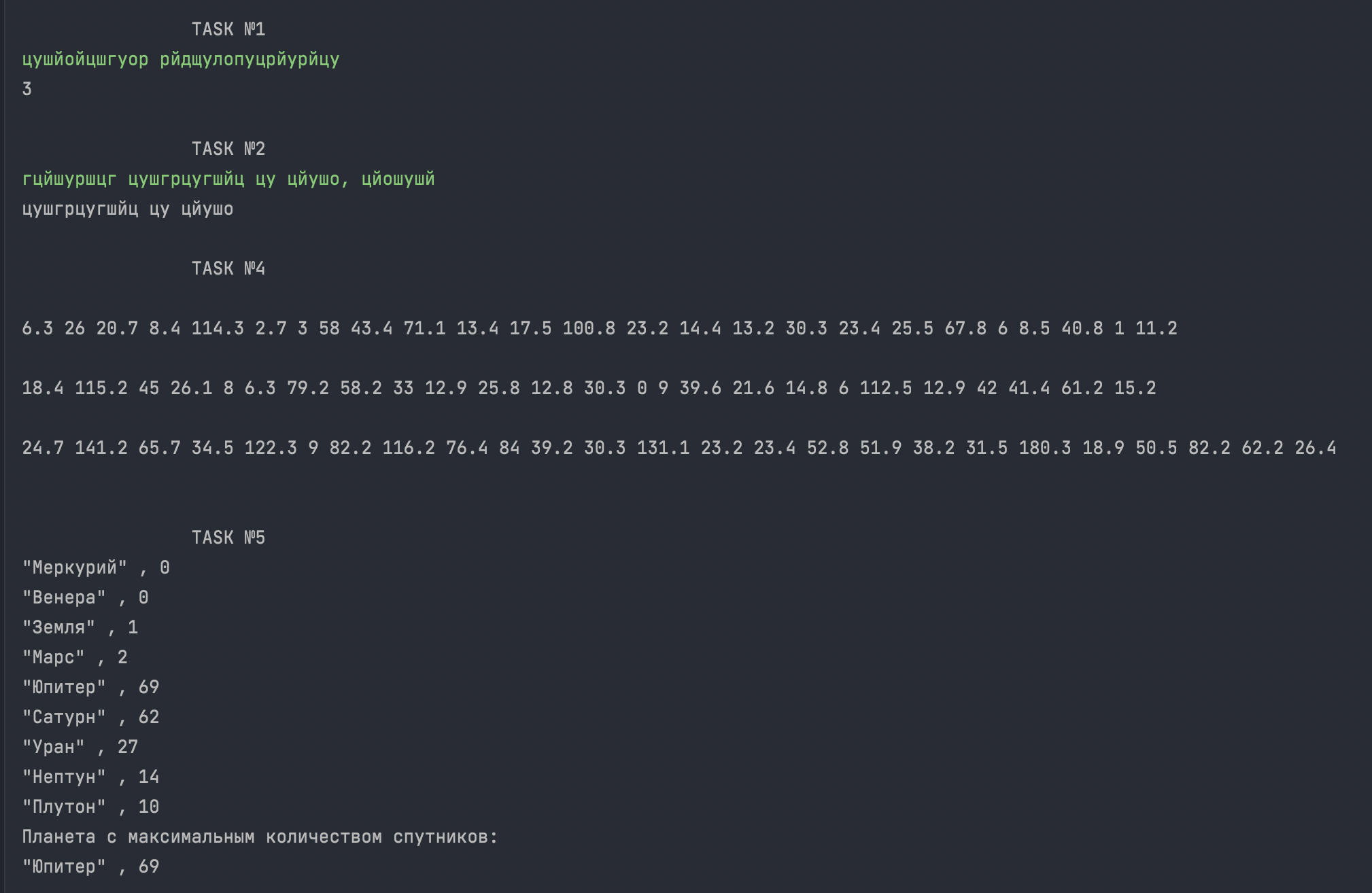
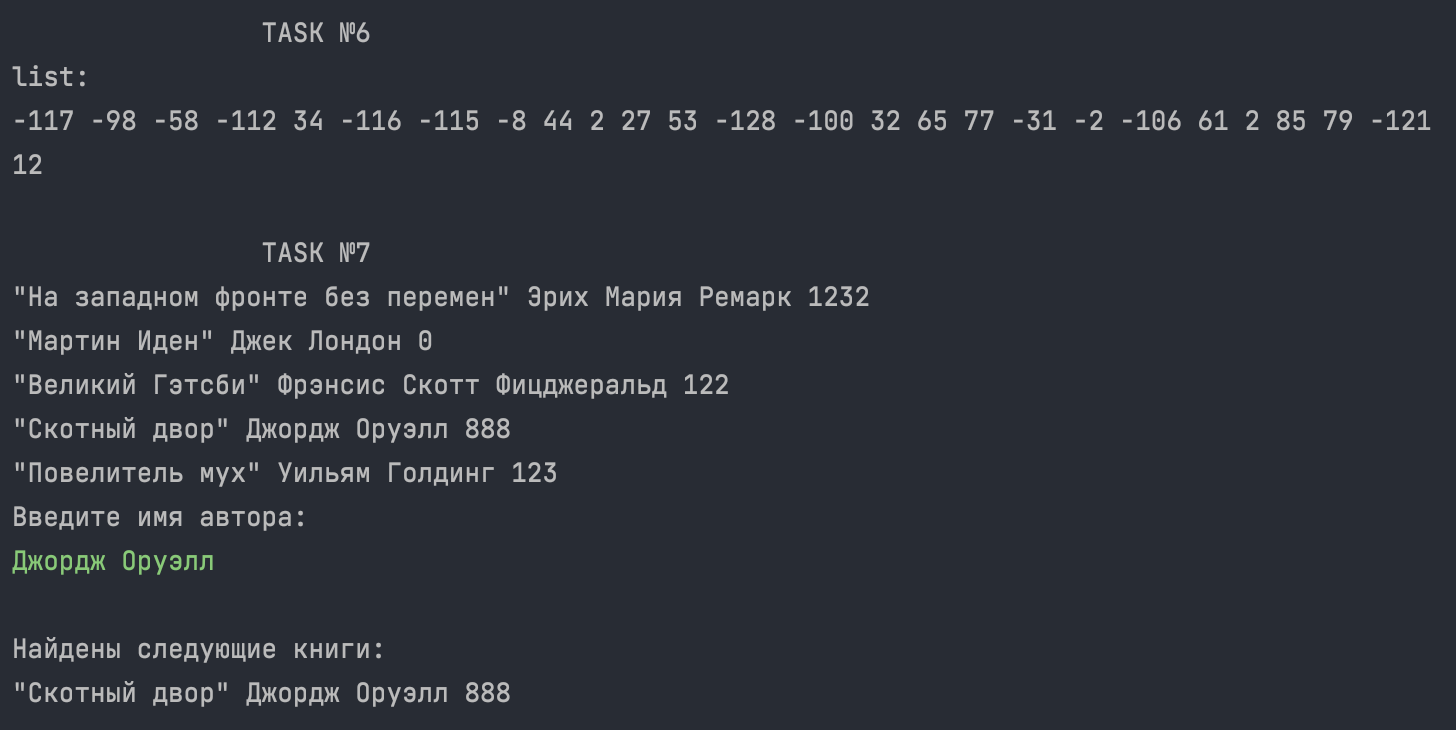
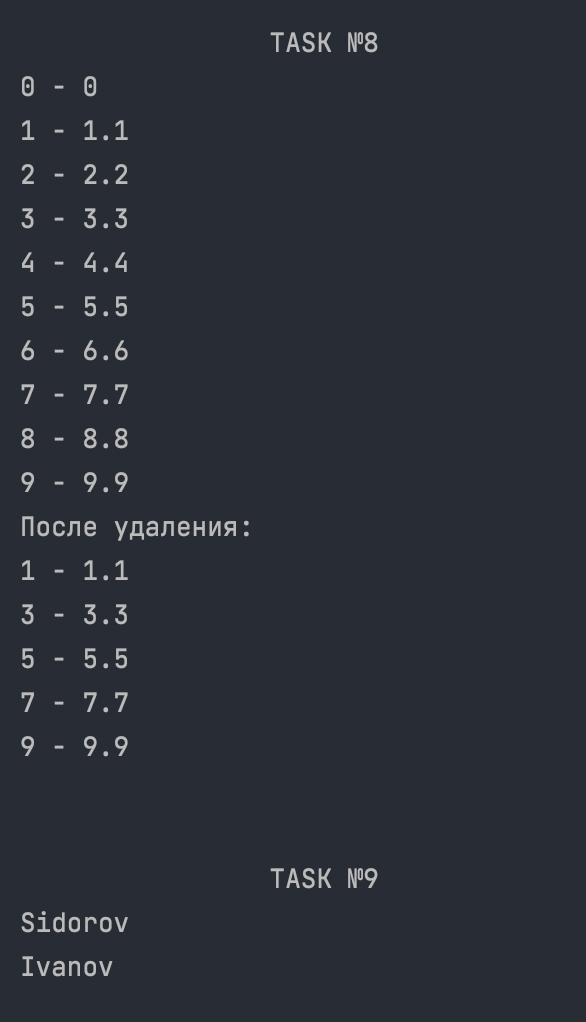
**Planet.cpp**

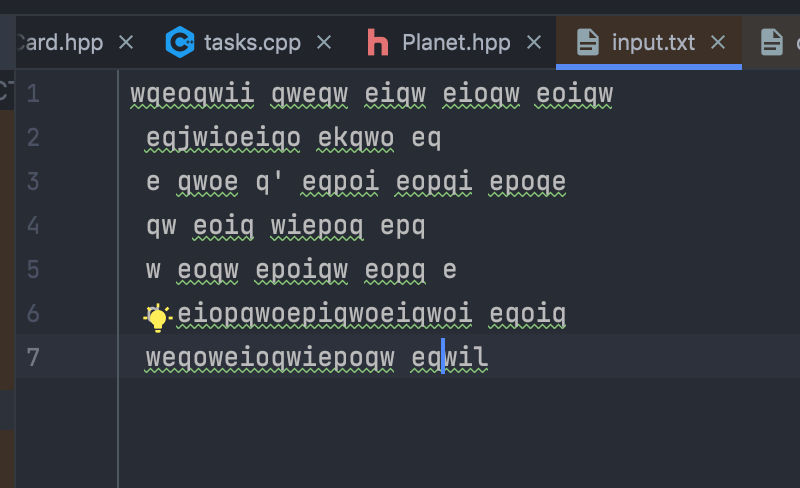
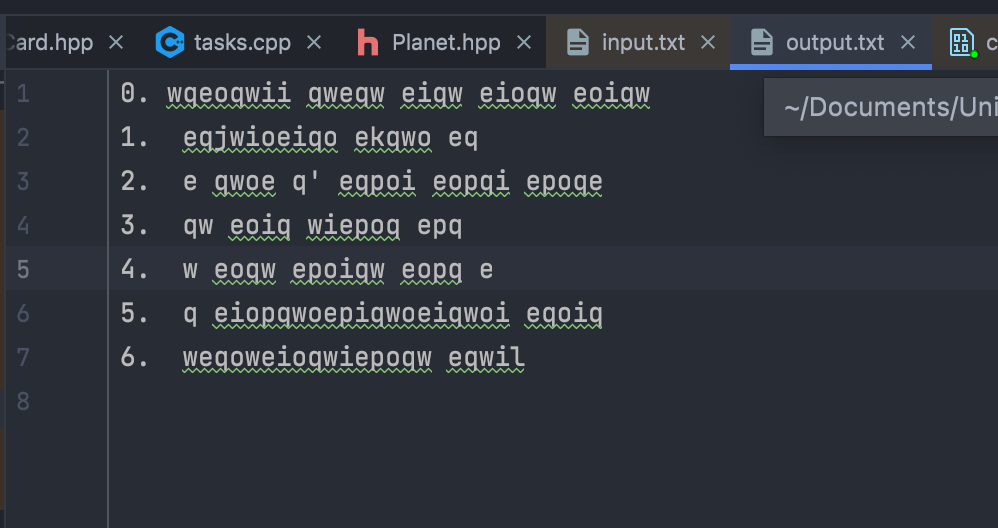
*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#include* "Planet.hpp"  
  
std::ostream &*operator*<<(std::ostream &ostream, Planet &planet) {  
 ostream << '\"' << planet.Name << '\"' << " , " << planet.SattelitesNumber << '\n';  
 *return* ostream;  
}

**Card.cpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#include* "Card.hpp"  
  
std::ostream &*operator*<<(std::ostream &ostream, Card &card) {  
 ostream << '\"' << card.title << '\"' << ' ' << card.author << ' ' << card.number << '\n';  
 *return* ostream;  
}

**Анализ результатов**

**** **** 

**** ****