

**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ» (ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

Отечет

Дисциплина: Прикладные информационные технологии

Ф.И.О. студента: Ерохин Артём Александрович

Направление: 09.03.02 Информационные системы и технологии

Номер группы: ИС101

Номер зачетной книжки: 210039

Номер варианта расчетно-графической работы: 9

Проверил: Ковригин А.В.

Новосибирск 2023

СОДЕРЖАНИЕ

[1.Задача 3](#_Toc124374458)

[2.Решение 4](#_Toc124374459)

[3. Вид приложения в консоли 11](#_Toc124374460)

# 1.Задача

Создание приложения «Задачи сотрудникам»

Необходимо создать программу, реализованную на С++, по управлению объектами данных по указанной теме с использованием ООП и реализовывать изученные паттерны.

Необходимо создать минимум 3 объекта сущности разных видов и указать взаимодействие между объектами. Реализовать интерфейсы ввод, вывода и редактирования сущностей. Реализовать ручной ввод сущностей, загрузку из файла и сохранения в файл, удаление выбранных сущностей.

Данные между запусками программы хранить в структурированном текстовом файле.

# 2.Решение

С самого начала я объявляю структуры данных для дальнейшей работы с ними:

struct human

{

string name;

string surname;

};

struct characteristics

{

string name;

string order;

int quantity;

double price;

};

Далее необходимо создать сущности для работы с ними, так как функционал методов у них однообразный, а различаются они лишь структурами данных, я опишу их работу на примере сущности клиентов:

private:

human cli[20] = {

{ "ООО","Рога и Копыта" },

{ "ООО","Однодневочка" },

{ "ИП","Тиньков Олег Борисович" },

};

Для хранения изначальных данных я использовал модификатор доступа “privat”, в связи с тем, что эти данные будут использоваться лишь в методах этого класса.

Для методов класса используется модификатор доступа “public” в связи с тем, что методы класса еще понадобятся для создания меню с помощью паттерна “controller”.

Метод вывода сущностей:

void Print() {

cout << " Клиенты компании << endl;

cout << endl;

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

cout << setw(6) << " Форма организации предприятия:" << setw(10) << (cli + i)->name<< setw(10);

cout << setw(6) << " Название предприятия:" << setw(10) << (cli + i)->surname << endl;

}

}

Метод добавления сущности:

void add() {

if (n == 20) {

cout << "Список клиентов полон!" << endl;

return;

}

cout << "Форма организации предприятия:" << endl;

cin >> cli[n].name;

cout << " Название предприятия:" << endl;

cin >> cli[n].surname;

n++;

}

Метод удаления сущности:

void del() {

if (n == 0) {

cout << "Клиентов нет!" << endl;

return;

}

cout << "Название предприятия:" << endl;

string cliname;

cin >> cliname;

int icli = -1;

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

if (cliname == cli[i].name) {

icli = i;

break;

}

}

if (icli == -1) {

cout << "Такого клиента нет" << endl;

return;

}

cli[icli] = cli[n - 1];

n--;

cout << "Успешное удаление клиента!" << endl;

}

Метод редактирования сущности:

void change() {

if (n == 0) {

cout << "Клиентов нет!" << endl;

return;

}

cout << "Введите клиентское предприятие" << endl;

string cliname;

cin >> cliname;

int icli = -1;

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

if (cliname == cli[i].surname) {

cout << "Форма организации предприятия:" << endl;

cin >> cli[i].name;

cout << "Название предприятия:" << endl;

cin >> cli[i].surname;

icli = i;

break;

}

}

if (icli == -1) {

cout << "Такого клиента нет" << endl;

return;

}

cout << "Успешное редактирование клиента!" << endl;

}

Метод записи данных сущности в файл:

void recording() {

fout.open("клиенты.txt");

fout << n << endl;

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

fout << cli[i].name << endl;

fout << cli[i].surname << endl;

}

fout.close();

}

Метод считывания данных из файла в сущность:

void readData() {

fin.open(“выгрузка.txt");

fin >> n;

fin.ignore();

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

getline(fin, cli[i].name);

getline(fin, cli[i].surname);

fin.ignore();

}

fin.close();

}

Ниже описан класс с паттерном “controller”:

class controller : public Data , public waiter,public client{

private:

bool work;

Таким образом делается класс дочерним для предыдущих сущностей для создания общего меню.

В модификации доступа “private” создается логическая переменная work для работы паттерна.

public:

controller() {

work = 1;

}

bool getwork() {

return work;

}

void showMenu() {

cout << "Меню:" << endl

<< "(1) Сотрудники:" << endl

<< "(2) Клиенты:" << endl

<< "(3) Задачи:" << endl

<< "(0) Выход из программы:" << endl;

while (getMenuItem() == -1);

}

void showMenu1() {

cout << "Меню:" << endl

<< "(1) Вывод всех сотрудников:" << endl

<< "(2) Добавление нового сотрудника:" << endl

<< "(3) Удаление сотрудника с указанным именем:" << endl

<< "(4) Запись списка сотрудников в файл:" << endl

<< "(5) Выгрузка списка сотрудников из файла:" << endl

<< "(6) Редактирование сотрудника с указанным именем"<< endl

<< "(7) Выход в главное меню"<< endl

<< "(0) Выход из программы:" << endl;

while (getMenuItem1() == -1);

}

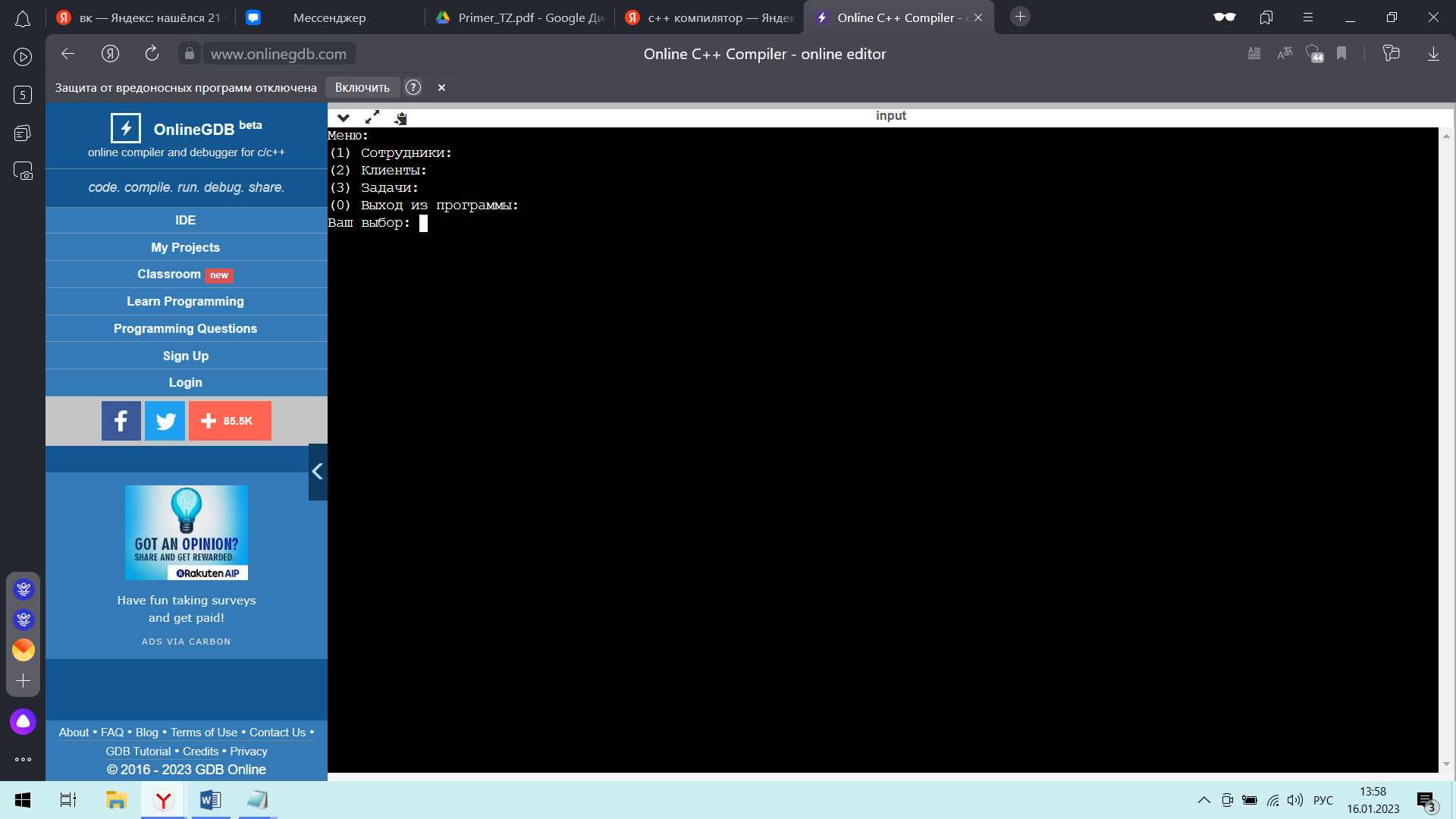
Для создания меню, был создан класс “controller” для предыдущих сущностей для создания общего меню.

В модификации доступа “private” создается логическая переменная “work” для работы паттерна.

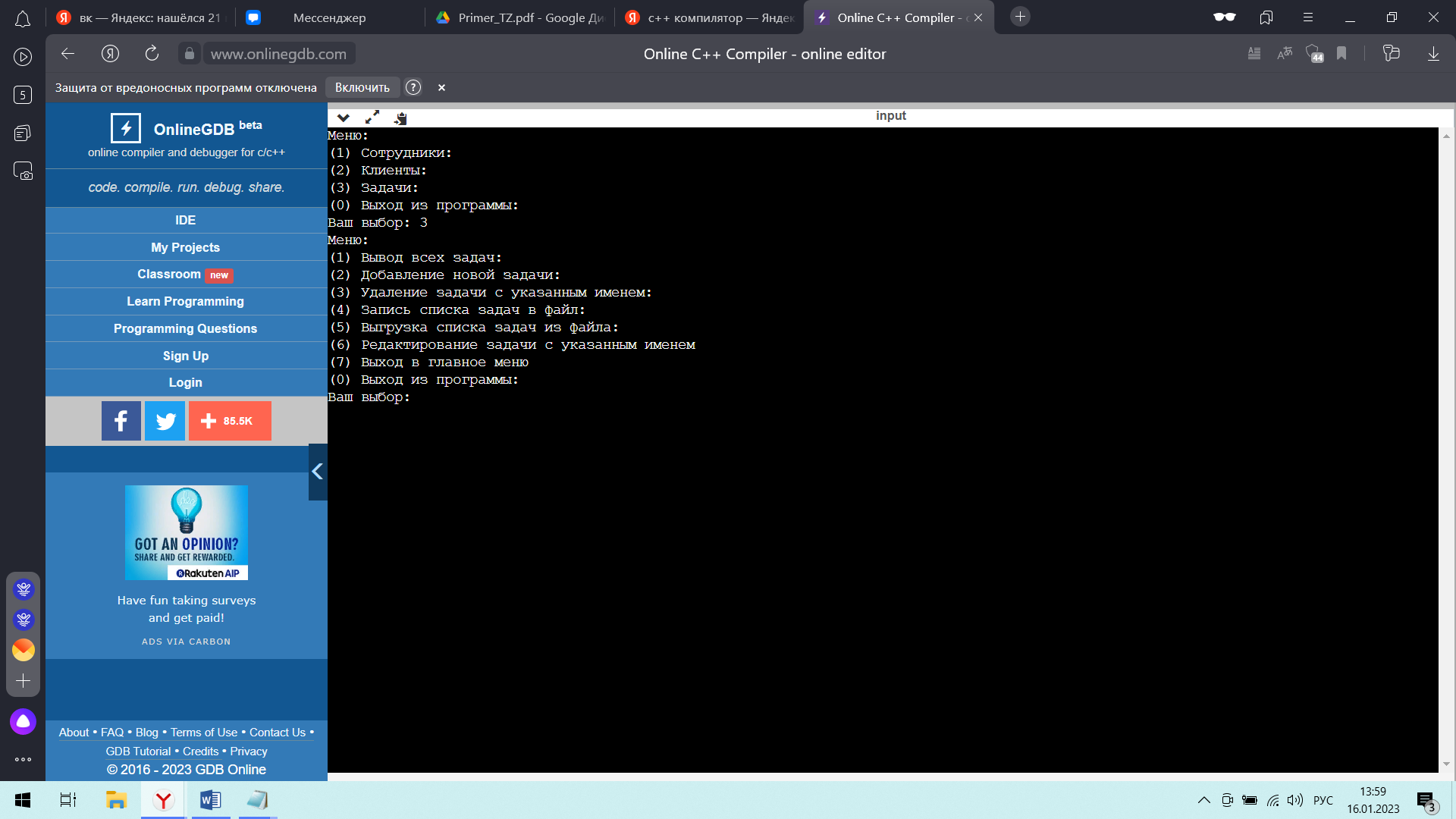
Для создания меню, были прописаны все возможные команды, которые сможет выполнять программа.

# 3. Вид приложения в консоли

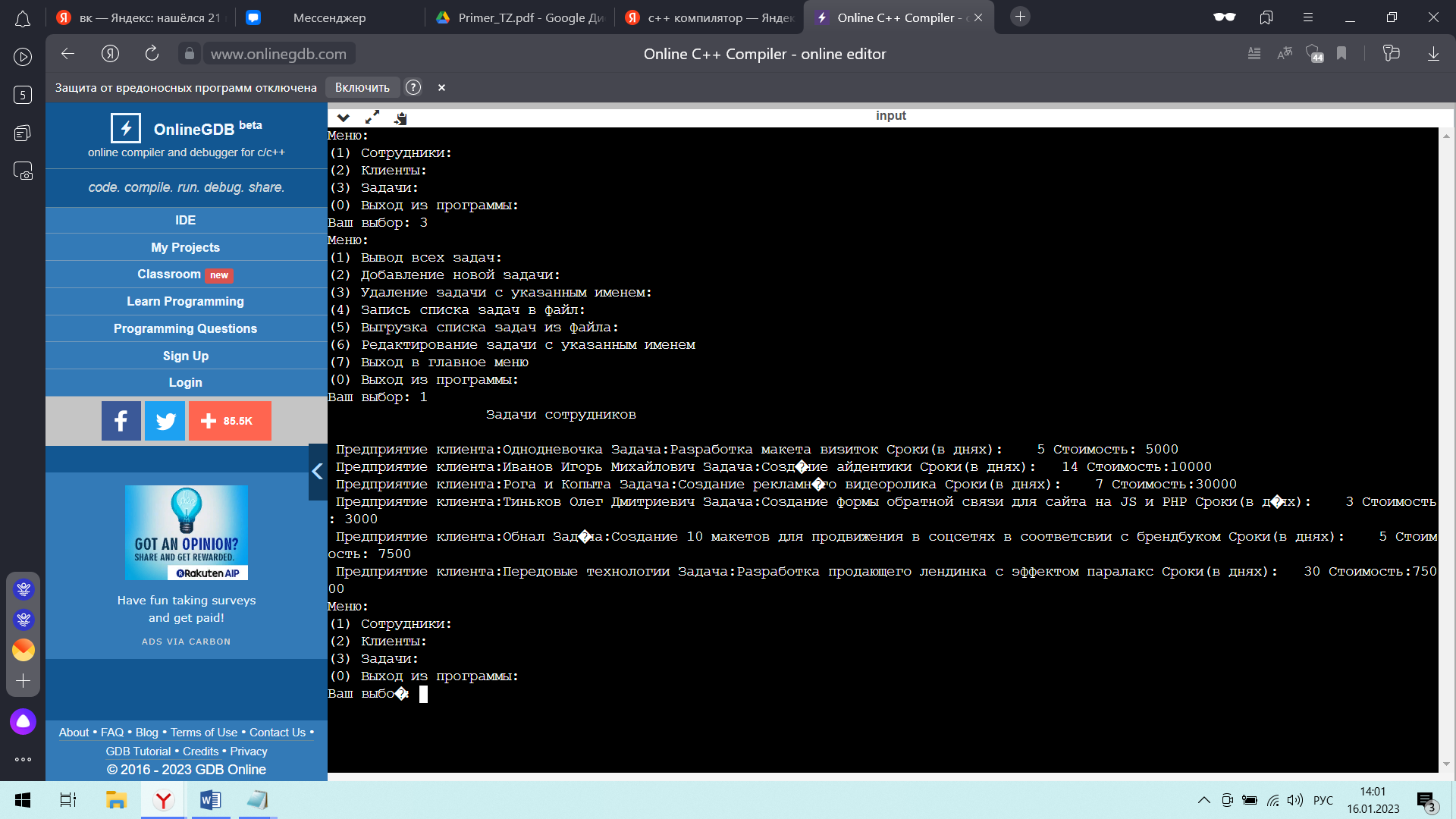
* Главное меню



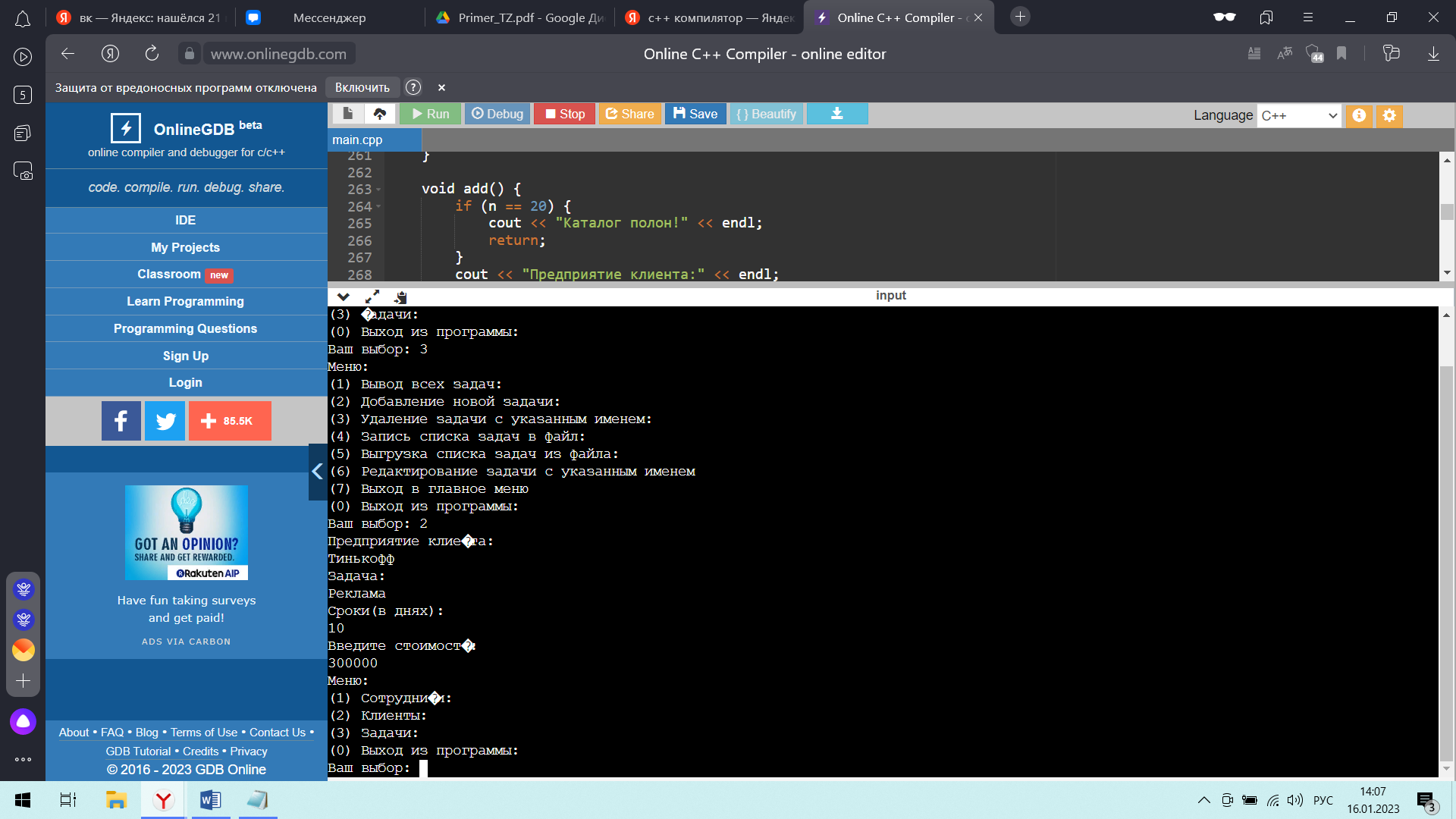
* На примере пункта 3 – Меню задач



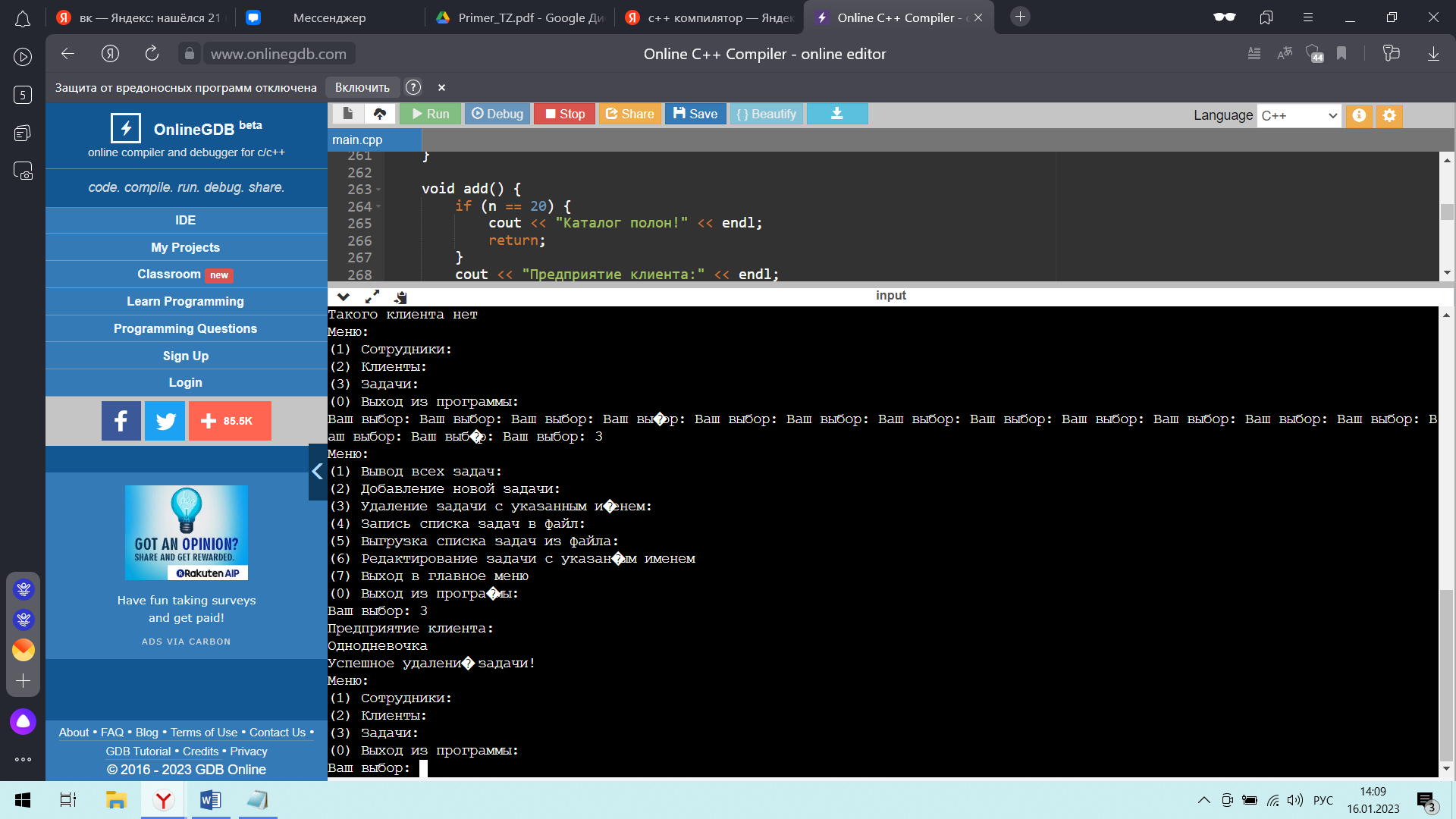
* Пункт 1 Меню



* Пункт 2 Меню



* Пункт 3 Меню



* Пункт 6 меню

