Федеральное государственное автономное образовательное учреждение высшего образования   
"**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "ВЫСШАЯ ШКОЛА ЭКОНОМИКИ**"

Факультет Компьютерных наук

Департамент программной инженерии

**Пояснительная записка**

к домашнему заданию по дисциплине

“Архитектура вычислительных систем”

Симоновича Иван Сергеевича

БПИ 193-2 19 Вариант

Москва 2020

**Формулировка задания:**

Вариант 19. У одной очень привлекательной студентки есть N поклонников. Традиционно в день св. Валентина очень привлекательная студентка проводит романтический вечер с одним из поклонников. Счастливый избранник заранее не известен. С утра очень привлекательная студентка получает N «валентинок» с различными вариантами романтического вечера. Выбрав наиболее заманчивое предложение, студентка извещает счастливчика о своем согласии, а остальных – об отказе. Требуется создать многопоточное приложение, моделирующее поведение студентки. При решении использовать парадигму «клиент-сервер» с активным ожиданием.

**Решение задания:**

При решении использовалась парадигма «клиент-сервер» с активным ожиданием. Согласно этой парадигме несколько клиентов и один сервер совместно с промежуточным программным обеспечением и средой взаимодействия образуют единую систему, обеспечивающую распреде­ленные вычисления, анализ и представление данных. Если точнее, то клиентский поток запрашивает сервер и после ожидает ответа, в то время как сервер ждет запроса от клиента и после этого выполняет свою логику в зависимости от запроса и отправляет ответ на запрос клиенту. Иногда клиент может «исполнить» роль сервера, если сам будет получать запросы. Аналогично сервер будет выступать в роли клиента, если ему потребуется обращаться с запросами к другим программам.

Источники, из которых брал информацию:

[**https://pro-prof.com/forums/topic/parallel-programming-paradigms**](https://pro-prof.com/forums/topic/parallel-programming-paradigms) **- про парадигму клиент-сервер**

[**https://bagdar.info/modele-klient-server-razdelenie-prilojenij-po-urovnyam-tipi-kl.html**](https://bagdar.info/modele-klient-server-razdelenie-prilojenij-po-urovnyam-tipi-kl.html) **- про парадигму клиент-сервер**

[**http://www.cplusplus.com/reference/condition\_variable/condition\_variable/**](http://www.cplusplus.com/reference/condition_variable/condition_variable/) **- просмотр примеров**

[**https://habr.com/ru/post/182626/**](https://habr.com/ru/post/182626/) **- про работу с потоками**

[**https://science-pedagogy.ru/ru/article/view?id=1962**](https://science-pedagogy.ru/ru/article/view?id=1962) **– про методы синхронизации**

При запуске в программу вводится положительное число больше нуля верхнее ограничение -максимальное число типа int. В программе присутствует проверка и повтор ввода, при неверных значениях. Это положительное число n – количество клиентов сервера.

Для корректной реализации парадигмы клиент-сервер в программе были реализованы два класса Valentine, где хранятся данные об клиентах, и Server, где разработана логика сервера.

После успешного ввода создается массив из 8 фраз, указатель сервера и указатели потоков клиентов. После этого создается и запускается поток сервера, с методом serverMain класса Server.

thread serverThread([server] { server->serverMain(); });

В этом методе выведется информация, что начался новый день и сервер будет спать 3 секунды, принимая сообщения, эмулируя ожидание студентки.

В это же время в main в цикле определяются и запускаются для каждого клиента потоки содержащие метод ClientMain, , куда передаются номер клиента, случайная фраза и сервер, на который он должен отправлять запросы.

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

{

clientThreads[i] = new thread(clientMain, i + 1, phrases[rand() % 8], server);

}

В этом методе клиент сообщает что начал писать письмо, после чего создается объект Valentine содержащий информацию о письме и его номере, после чего засыпает на случайное время, эмулируя реального клиента и его трату времени на написание этого письма.

this\_thread::sleep\_for(chrono::milliseconds(100+rand() % 1000));

Если он уже поспал это время, то отправляет свой запрос с ссылкой на письмо на сервер. А сам сервер при вызове запроса enqueueRequest(Valentine\* letter) заносит это письмо в вектор vector<Valentine\*> letters и увеличивает своё количество принятых писем.

server->enqueueRequest(&letter);

Дальше происходит **активное ожидание** потока, то есть пока сервер не определит победителя все запущенные потоки будут ожидать.

while (!server->isResponseReady())

this\_thread::sleep\_for(chrono::milliseconds(5));

Если сервер выбрал победителя (а он его всегда выбирает победителя случайно из поступивших валентинок за время ожидания в 3 секунды: winner = rand() % letters.size();)

, то поток запускается дальше, так как метод сервера isResponseReady() вернет true,и принимает ответ на свой запрос с сервера с помощью метода getResponse, а именно является ли он победителем, и печатает информацию об этом в консоль. Сам сервер перед этим пишет количество полученных писем, а также письмо и номер комнаты победителя (номер самого клиента).

При большом количество клиентов они могут отправлять запросы на сервер случайно, так как будет работать множество потоков и у каждого будет случайное время ожидания ( см. в тестах пример для n=28). В примерах видно, что принимают значения о победители после активного ожидания от сервера клиенты случайно, даже при маленьких значениях n (6,8).

В методах getResponse, enqueueRequest и serverMain, вызываемых в потоках , используется мьютекс mu (методы lock(), unlock()), для синхронизации потоков, чтобы доступ к ресурсу получало по одному потоку.

В конце программы в цикле вызываются методы join для потоков клиента и потока сервера отдельно. Это нужно, чтобы потоки успели отработать до завершения программы без ошибок. В конце также удаляются сервер и потоки, это необходимо для очищения памяти.

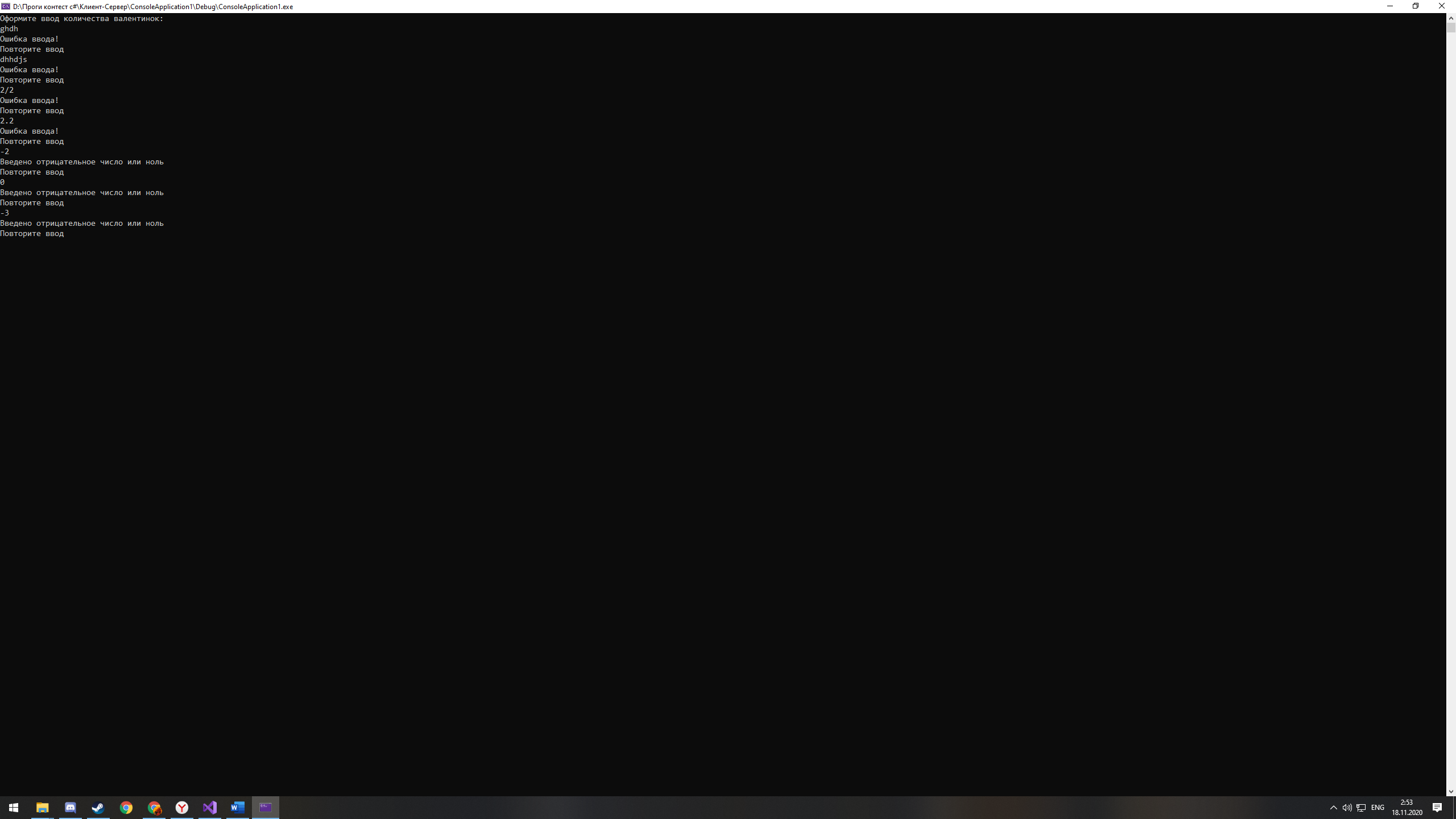
Все подробные комментарии для каждого метода содержатся в коде программы.

Вся необходимая информация выводится в консоль. Если вывод происходит на сервере, то в начале предложения будет выведено [SERVER], если вывод происходит в клиенте, то в начале выведется [CLENT Ni], где i – номер клиента. После конца ожидания сервера (3 секунды) выводится информация о поступивших от клиентов валентинок и опобедителе.

**Тестирование программы:**

**Случаи некорректного ввода:**

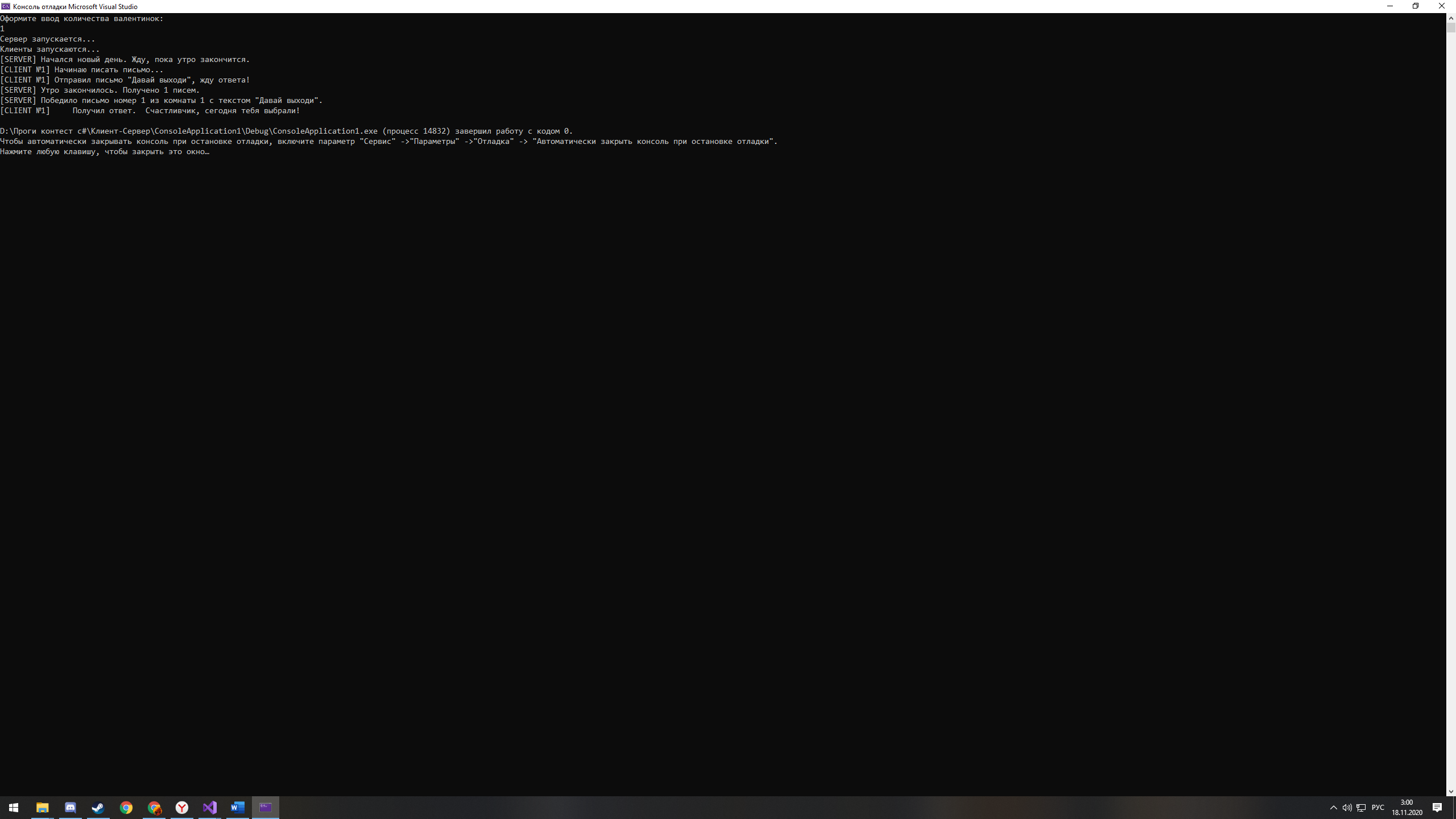
Все возможные случаи некорректного ввода разобраны на одном скрине. При вводе неподходящего числа программа запросит повтор ввода:



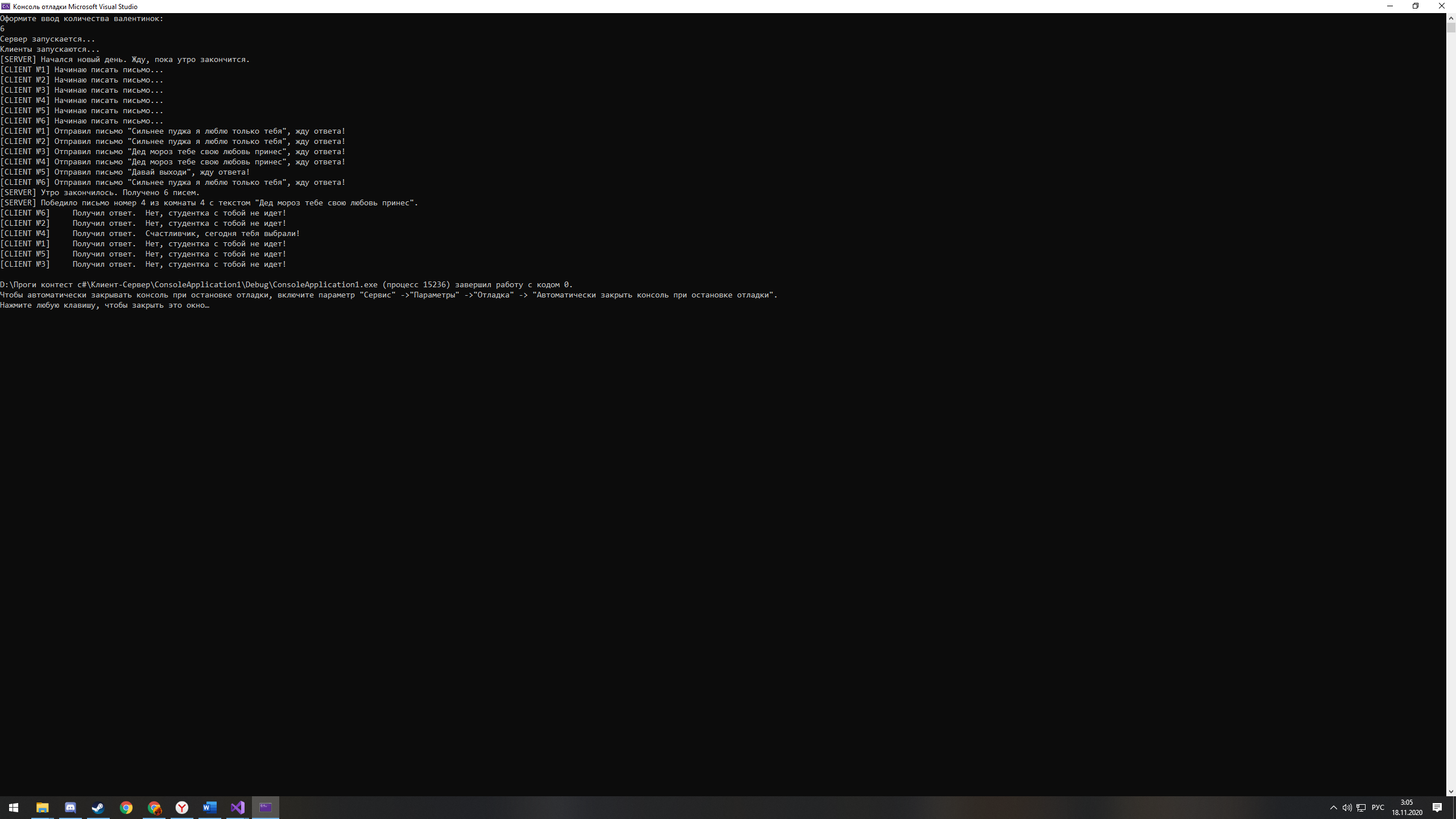
**Примеры корректного ввода:**

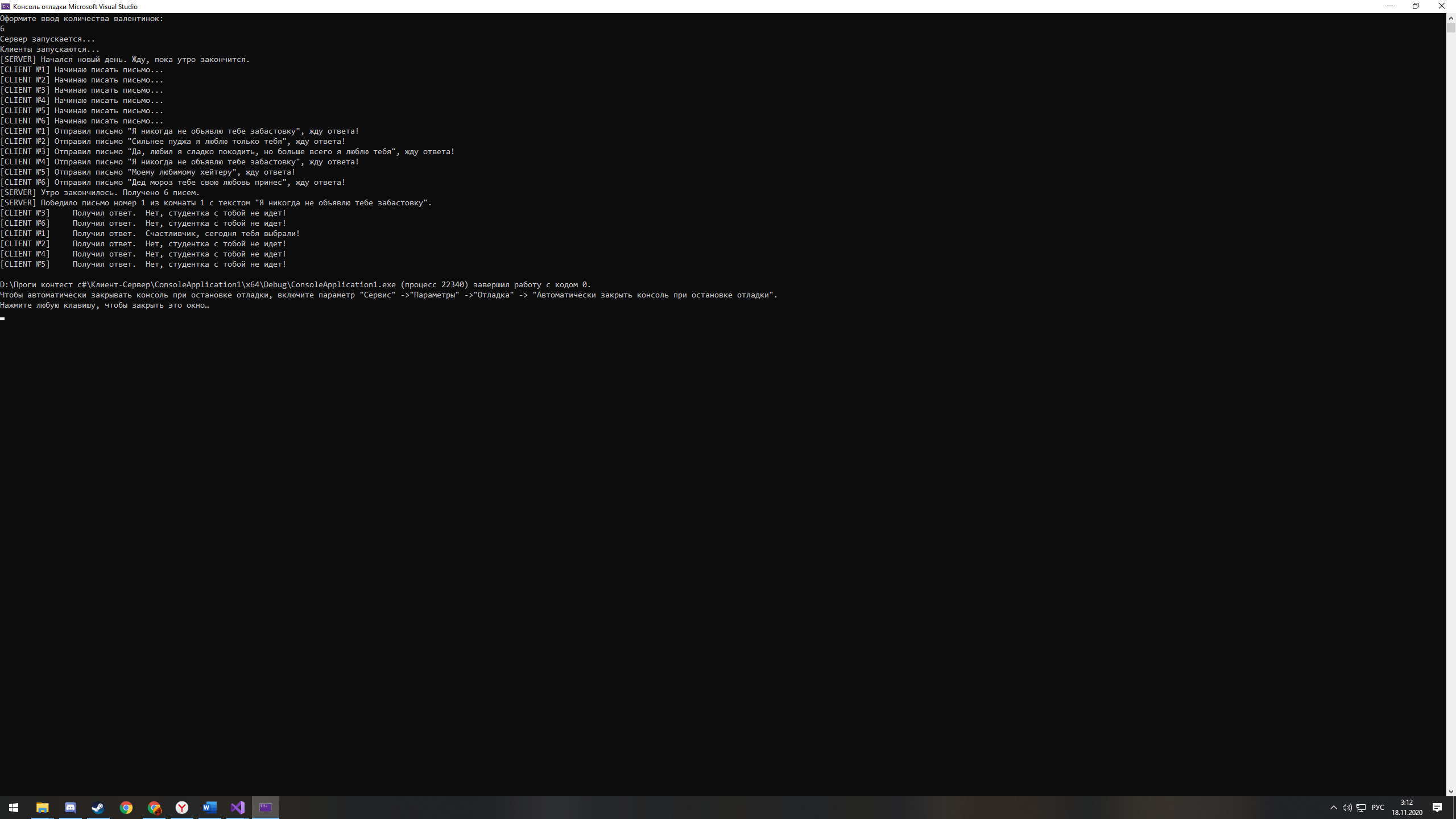
Ниже приведены примеры тестов для n>0

**Для n=1**

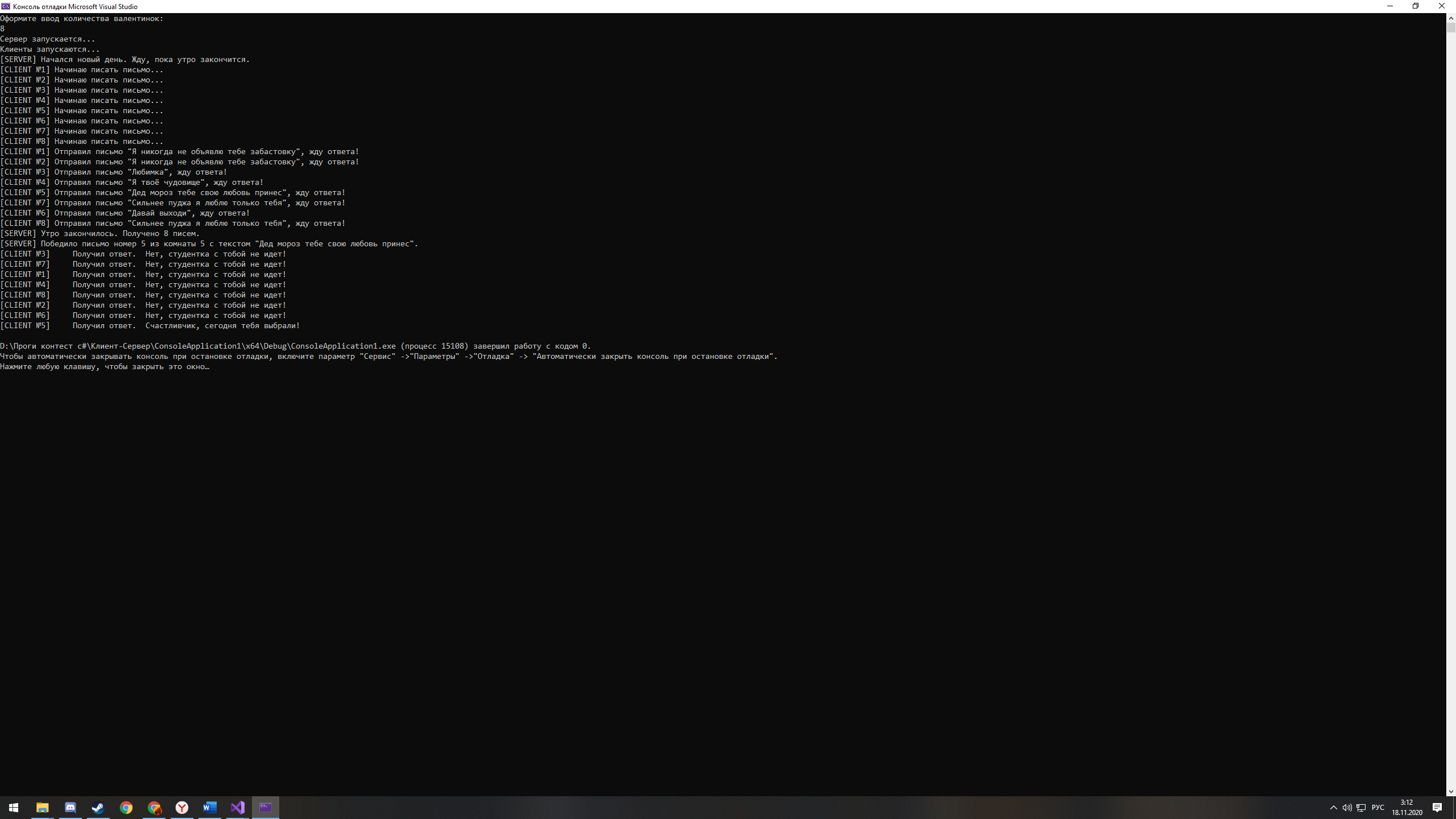


**Для n=6**





**Для n=8**



**Для n=28 Видно, что каждое письмо может отправится в любом порядке**

**Например, клиент 17 отправил до 4-ого, также как и 19 до 5-ого. При большем количество потоков (клиентов) порядок отправки будет еще более рандомным.**

