**Лабораторная работа №2**

**«Технологии промышленной разработки программного обеспечения. Требования»   
(2 семестр)**

# Постановка задачи

Выполнить реинжиниринг базового варианта прикладной программы «Регистратор ресурсов», далее Регистратора. Реинжиниринг понимается как переработка программы с целью изменения её функциональных и/или эксплуатационных параметров в указанных границах. Пояснения относительно процесса проектирования базового варианта Регистратора, включая основные документы проектирования, а также исходные тексты на языках C# и C++, приведены в качестве Приложения №1 к и предоставляются студенту.

Студент выбирает тему реинжиниринга из списка, в котором темы различаются параметрами, сформулированными в терминах функций базового варианта Регистратора.

# Минимальные требования к выполнению заданий

1) Должны быть разработаны концептуальная модель выбранного варианта Регистратора, включающая таблицу прецедентов, диаграммы прецедентов, концептуальных классов и системных взаимодействий. Разработку можно вести на основе соответствующих таблиц и диаграмм, содержащихся в описании базового варианта Регистратора.

2) Должно быть проведено архитектурное проектирование выбранного варианта Регистратора в форме разработки диаграмм(ы) проектных классов. Диаграмма классов должна быть доведена до уровня зависимости от платформы.

3) Должно быть проведено детальное проектирование выбранного варианта Регистратора в форме локальной переработки текста базового варианта Регистратора с обоснованием проведённых изменений исходного текста по каждому изменённому фрагменту текста.

4) Спроектированный вариант Регистратора должен быть отлажен.

5) Должен быть составлен план тестирования разработанного варианта Регистратора

6) Должно быть проведено тестирование разработанного варианта Регистратора по составленному плану и протокол тестирования включён в отчёт по лабораторной работе.

7) В рамках детального проектирования должен быть обеспечен подсчёт среднего времени использования ресурса, среднего времени ожидания запроса (если оно предусмотрено).

# Методические указания.

Студенту рекомендуется согласовать концептуальную модель разрабатываемого варианта Регистратора (таблицу прецедентов, диаграммы прецедентов, концептуальных классов, системных взаимодействий) прежде, чем приступать к архитектурному проектированию, чтобы убедиться в корректном понимании темы.

# Выбор темы проектирования

Конкретный вариант Регистратора ресурсов характеризуется сочетанием следующих основных независимых параметров:

c - «ёмкость» ресурса,

v – цена ресурса (за единицу времени),

s - «размер» запроса,

pr – приоритет запроса,

wt –время ожидания запроса на обслуживание,

tu – продолжительность обслуживания запроса (владения ресурсом),

pbu – вероятность отказа ресурса во время использования,

pbi - вероятность отказа свободного ресурса.

Примечание. Базовый вариант Регистратора ресурсов может быть характеризована следующим сочетанием параметров:

c = s = 1, v = 0, pr = 0, wt = 0, tu >=0, зависит от действий оператора , pbu = 0, pbi = 0.

# Выбранная тема проектирования (реинжиниринга)

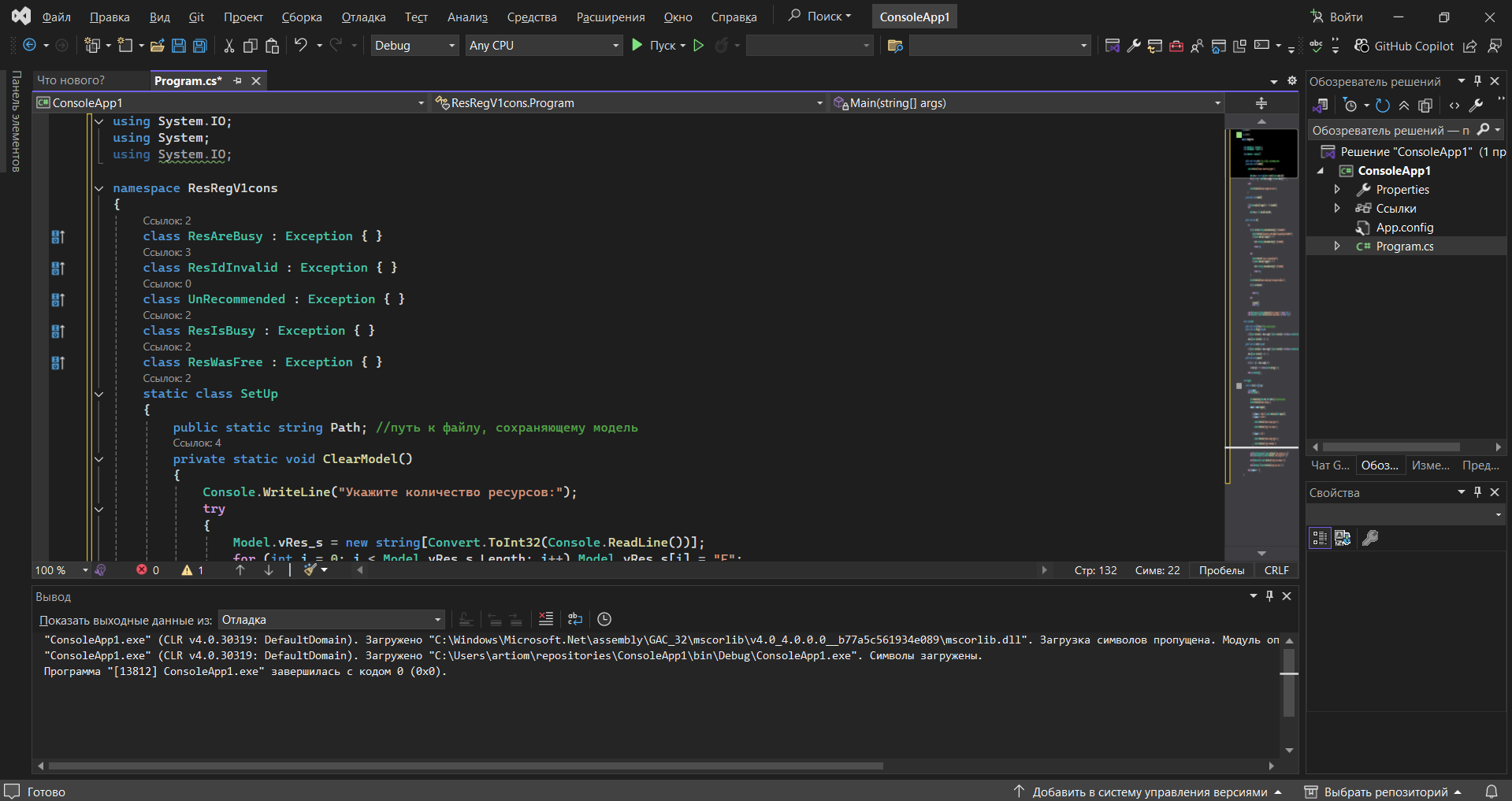
|  | Тема (вариант) | Характеристика варианта \*) | Балл |
| --- | --- | --- | --- |
| 1 | Диверсификация ёмкости ресурса и размера запроса при неделимости запроса | ресурсы и запросы разные, но для использования д. б. c >= s,  v = 0, pr = 0, wt = 0, pbu = 0, pbi = 0  Запрос не может быть разделён, чтобы пользоваться несколькими ресурсами. | 3 |

# Ход выполнения над лабораторной работой

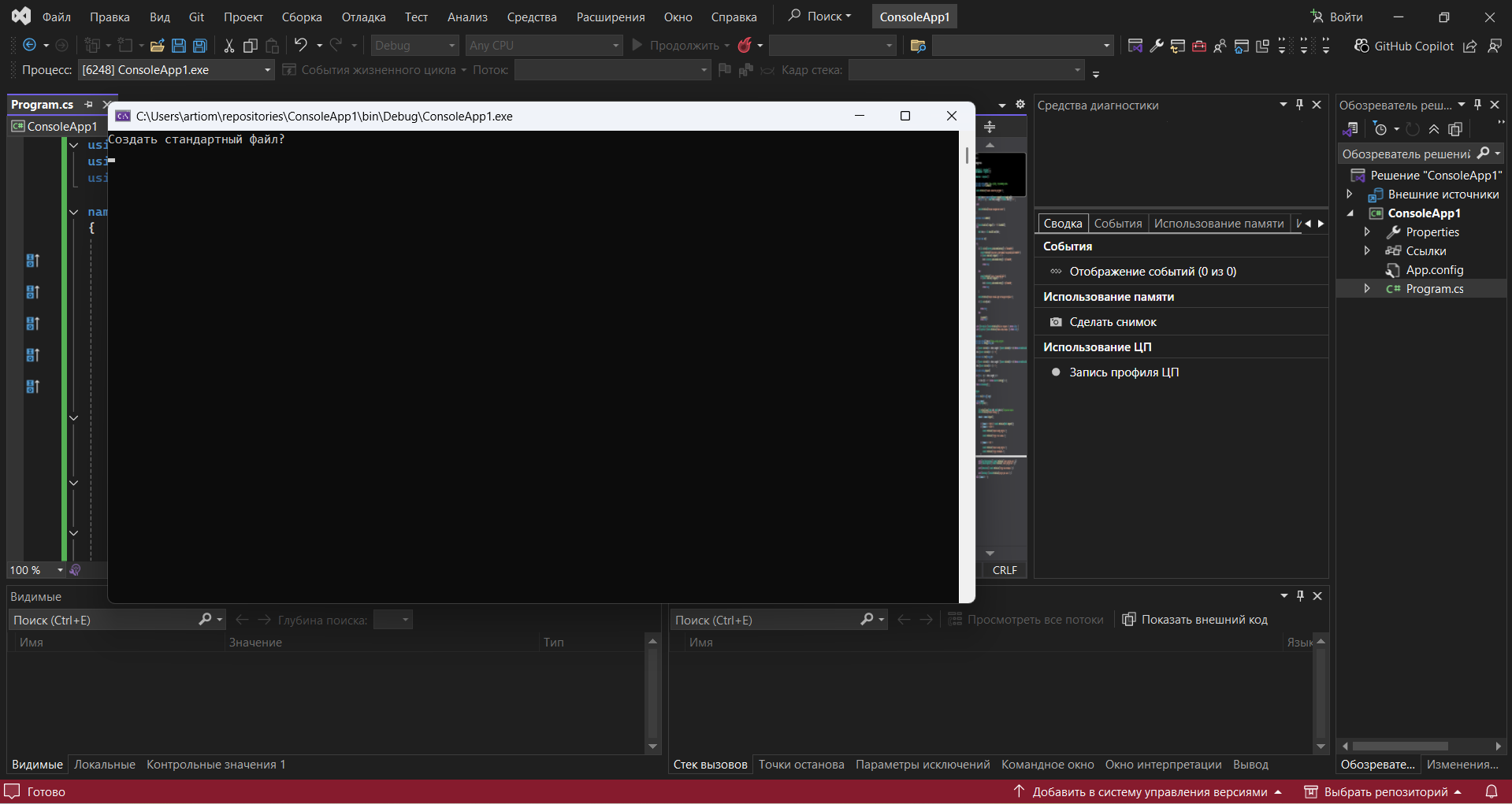
| номер | действие | предпосылка | результат |
| --- | --- | --- | --- |
| 1 | создать документ содержащий отчёт по выполнению лабораторной работы “ЛР №2 ТПР НосовАИ М24-534” | требование сдачи лабораторной работы | документ создан |
| 2 | Добавить заголовок в “ЛР №2 ТПР НосовАИ М24-534” и разделы из условия: “Постановка задачи”, “Минимальные требования к выполнению заданий”, “Методические указания” | метаинформация о лабораторной работы для целостного понимания | разделы добавлены |
| 3 | Добавить раздел “Выбранная тема проектирования (реинжиниринга)” | отобразить какая тема выбрана для выполнения в рамках лабораторной работы | раздел с выбранной темой добавлен |
| 4 | Ознакомиться с условием лабораторной работы “Регистратор ресурсов (обзор базового варианта как проекта)” | для понимания предметной области и проблематики - решаемой системой задачей и основ на которых она построена | ознакомился |
| 5 | Создать раздел с обзором на “Регистратор ресурсов (обзор базового варианта как проекта)” где по ходу будут даваться комментарии возможным изменениям варианта при реализации выбранной темы | для систематизации вводных по проекту и предложений по изменениям в рамках в рамках темы | в процессе |
|  | Установить Visual Studio |  |  |
| 6 | запустить программу и поработать с ней | требование | в процессе. раздел добавлен |
|  | Разобрался что есть в приложении к лабораторной работе, всех диаграммах | Нужно, чтобы в дальнейшем понимать что в принципе есть, чтобы суметь понять как изменить для обеспечения новых функций | сделано |
| 7 | определить “что в программе изменить чтобы обеспечить новые функции” |  |  |
| 8 | добавиться размер у ресурса и атрибута на рисунке 2 |  |  |
| 9 | в команде запроса будет размер ресурса теперь |  |  |
| 10 | сверять запрос с размером ресурса и если не совпадает то по ретраю вернуть соответствующее сообщение |  |  |
|  | сформулировать: что для того чтобы совершить этот реинжиниринг то будут затронуты следующие блоки |  |  |
|  | Риснок 6 можно не трогать |  |  |
|  | Программа StarUML lkz UML |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Запустить программу и поработать с ней

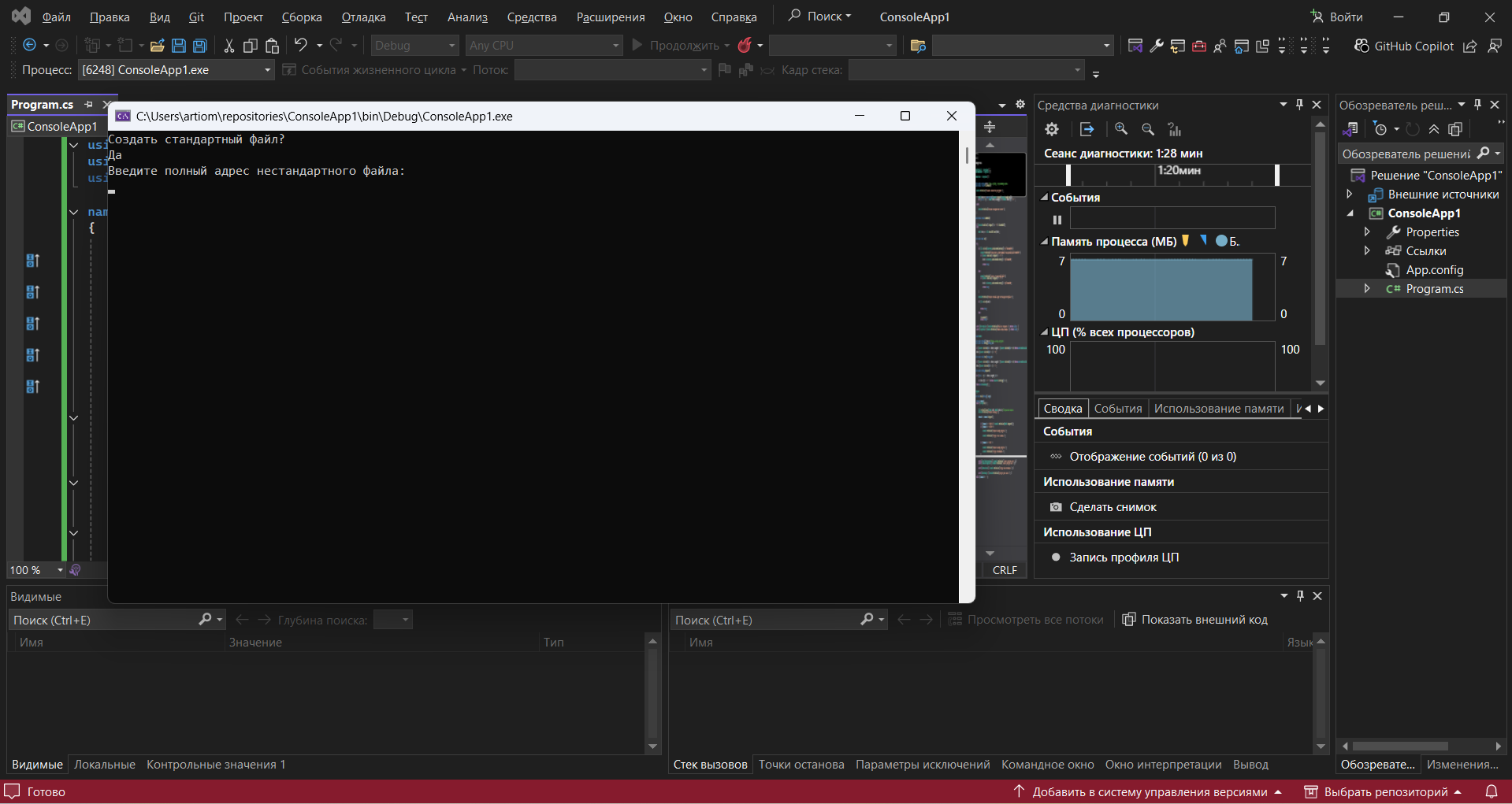
1 Скопирован код в IDE



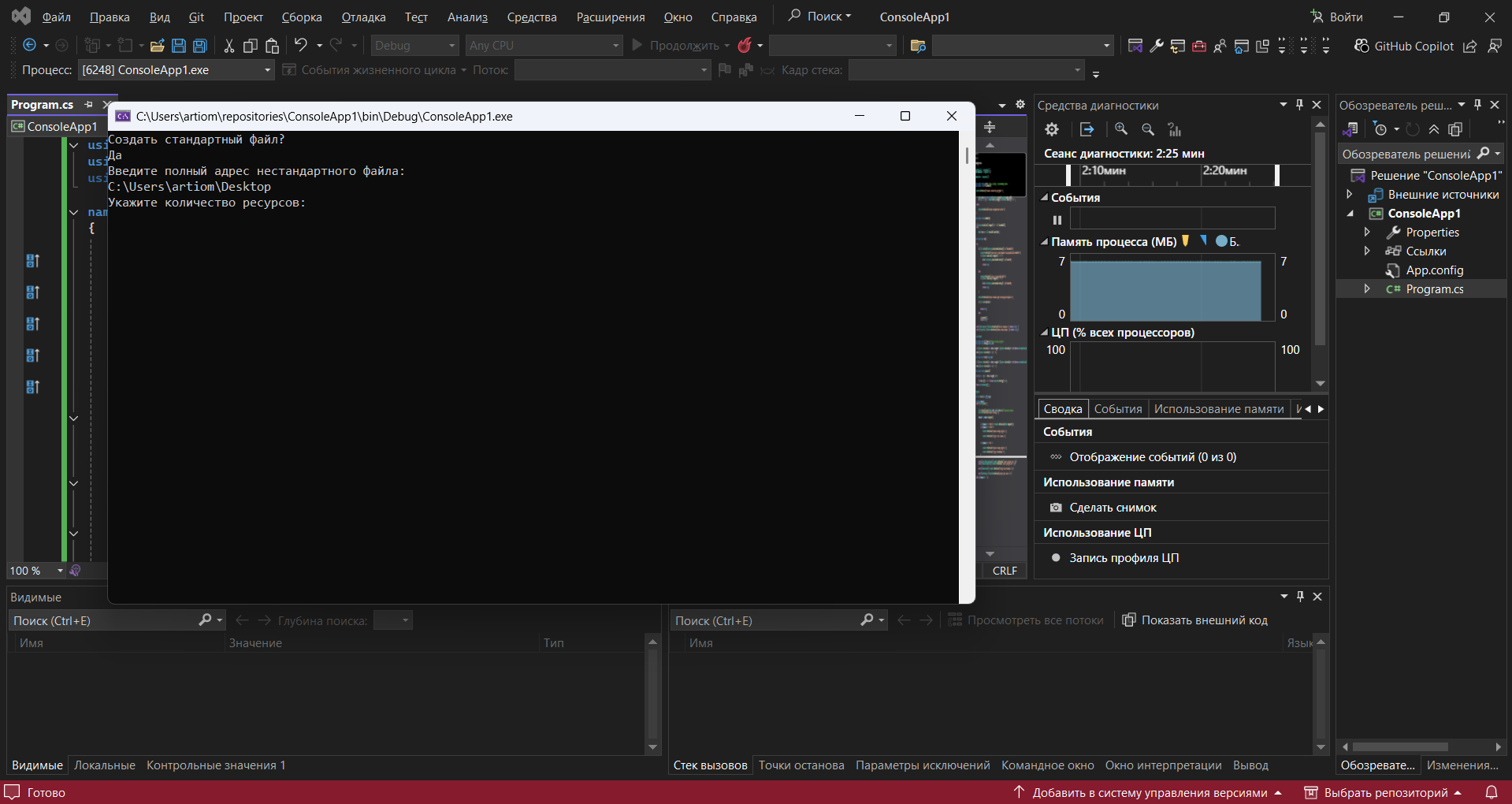
2 Произведён запуск после чего в консоли появилось “Создать стандартный файл?”



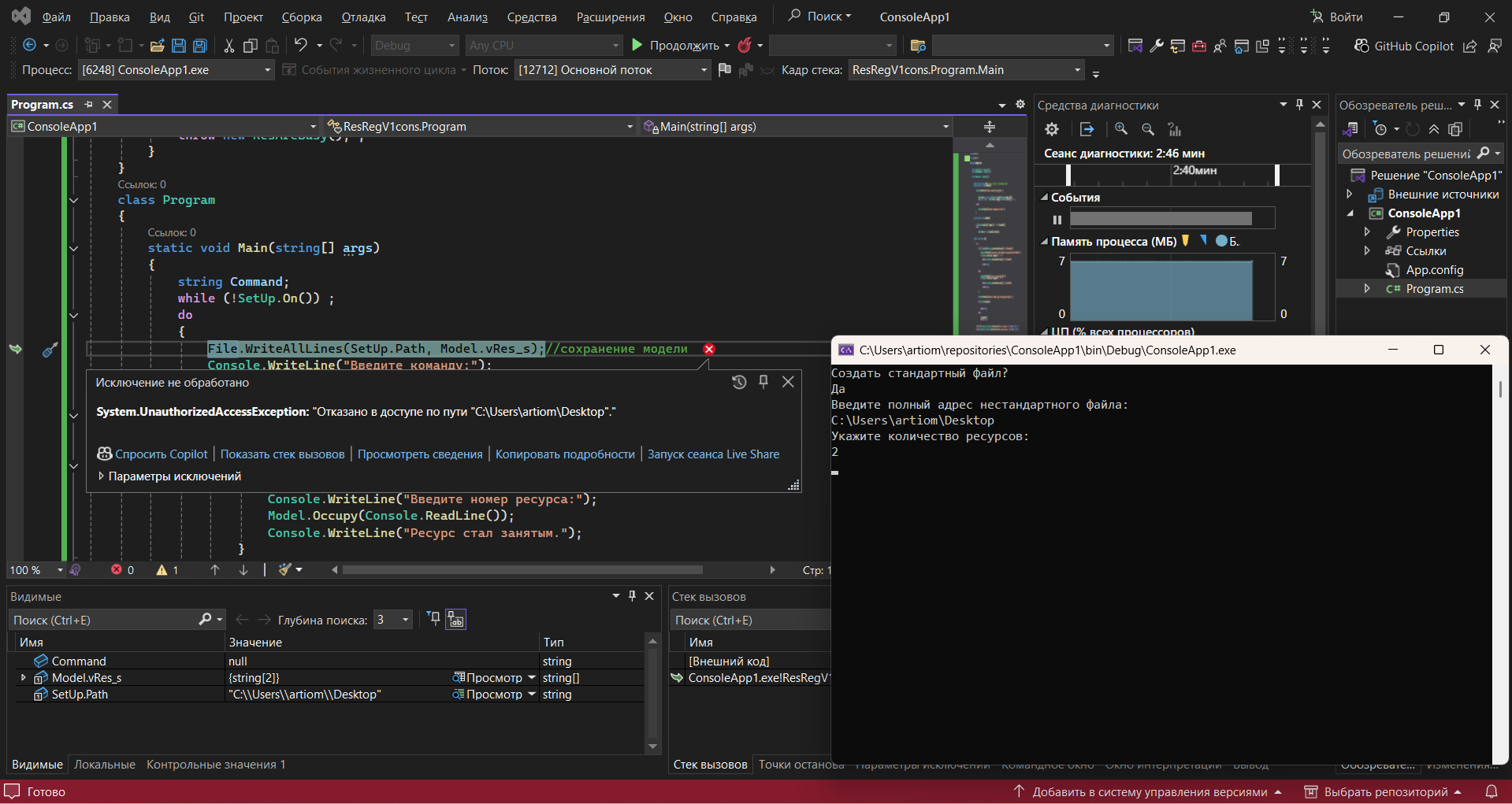
3 После того как я ввёл “Да”



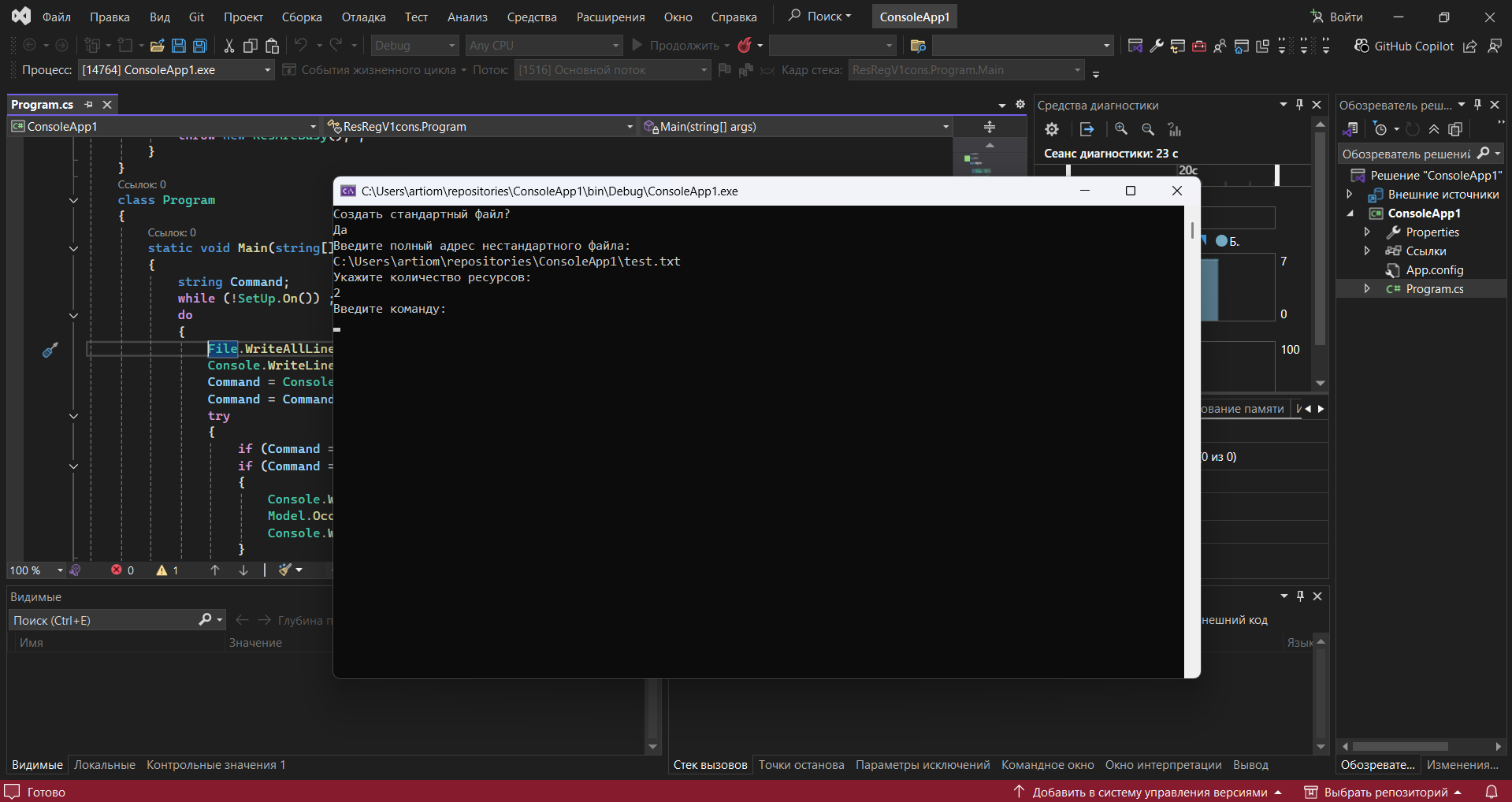
4 Ввёл “C:\Users\artiom\Desktop”



5 Указал что ресурсов 2 - после чего получена ошибка - указать директорию проекта



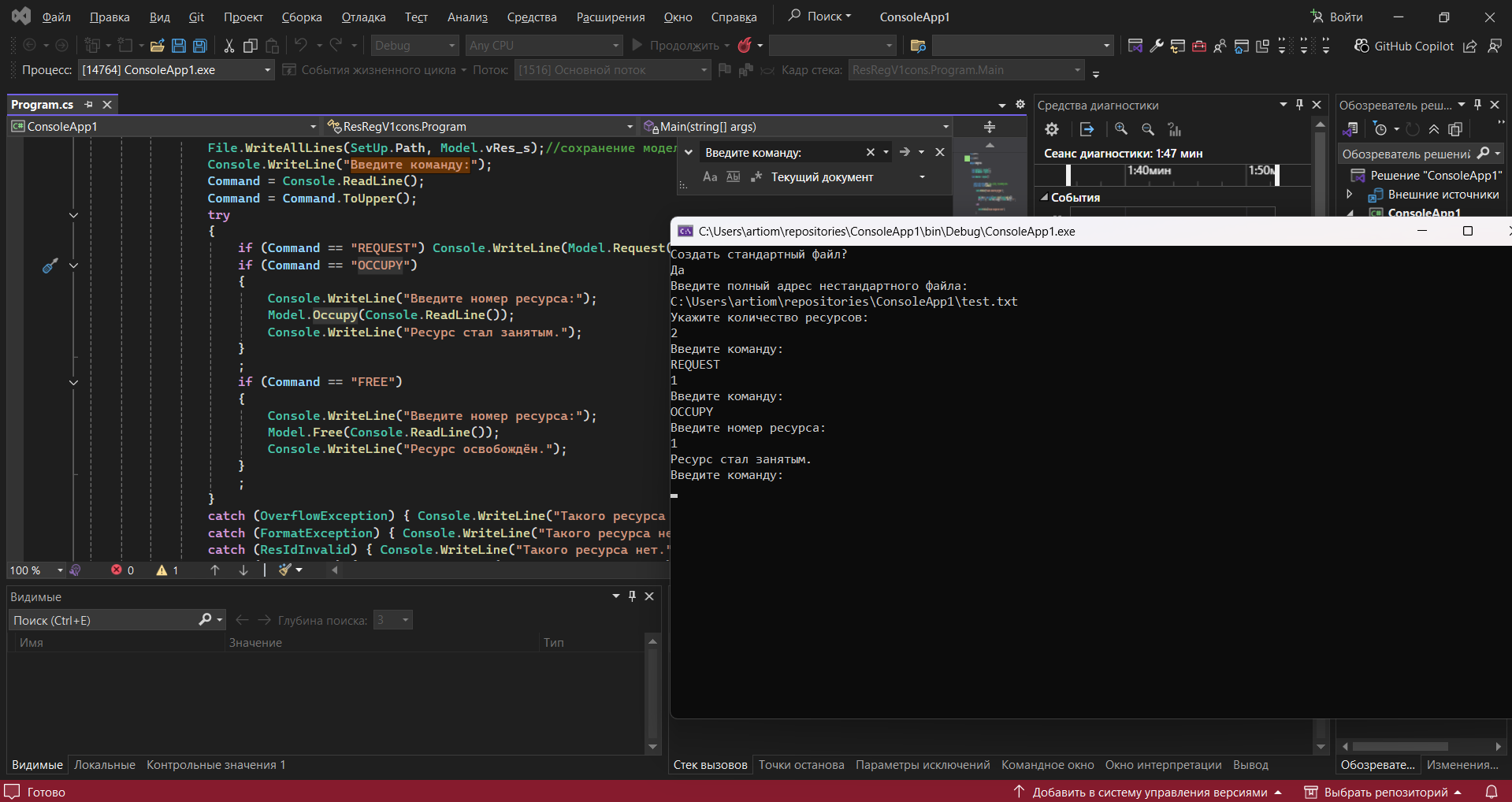
Скорее всего проблема в том что нет доступа к папке рабочего стола попробую указать путь до директории проекта



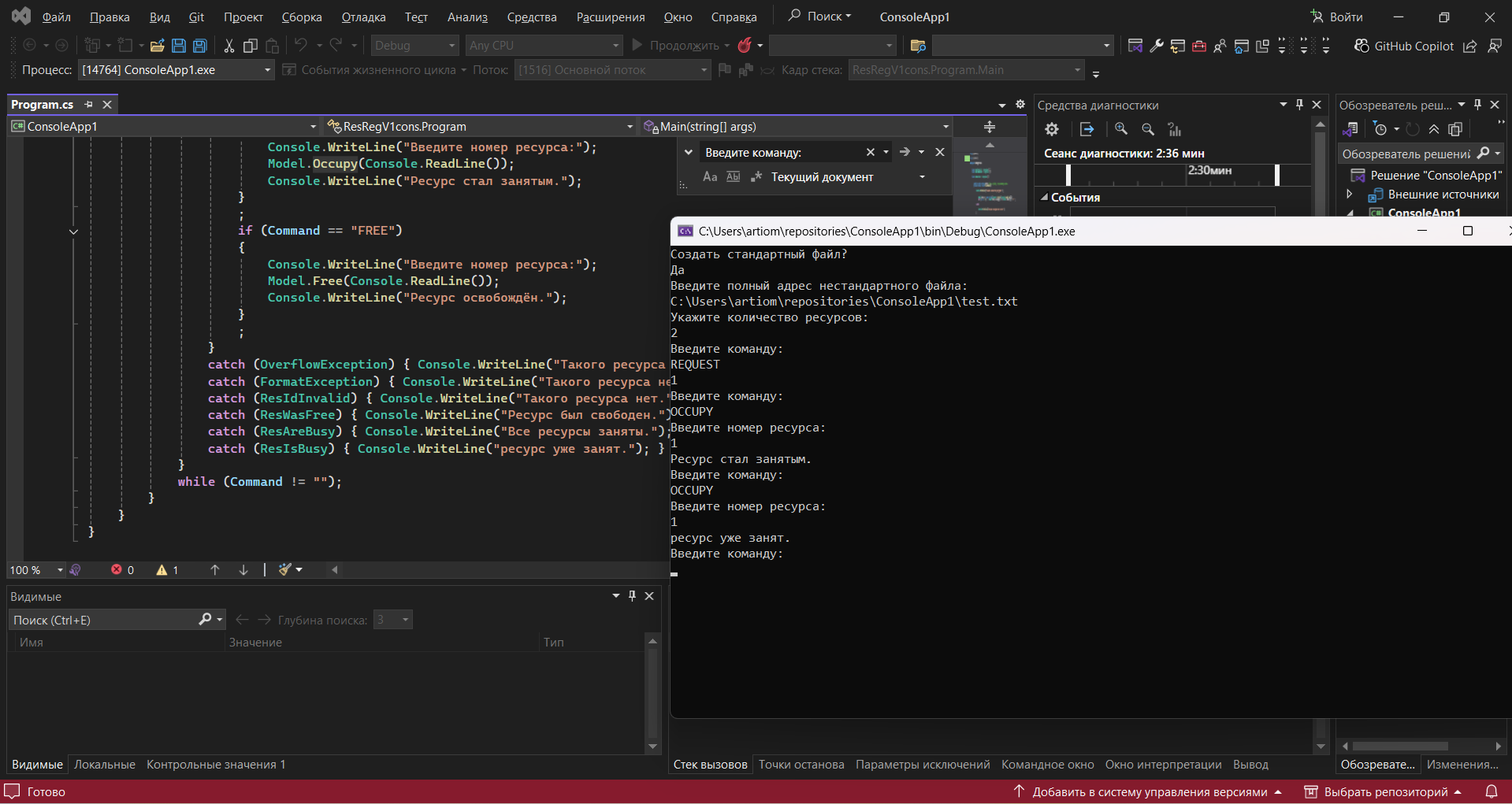
Теперь вводим команду REQUEST



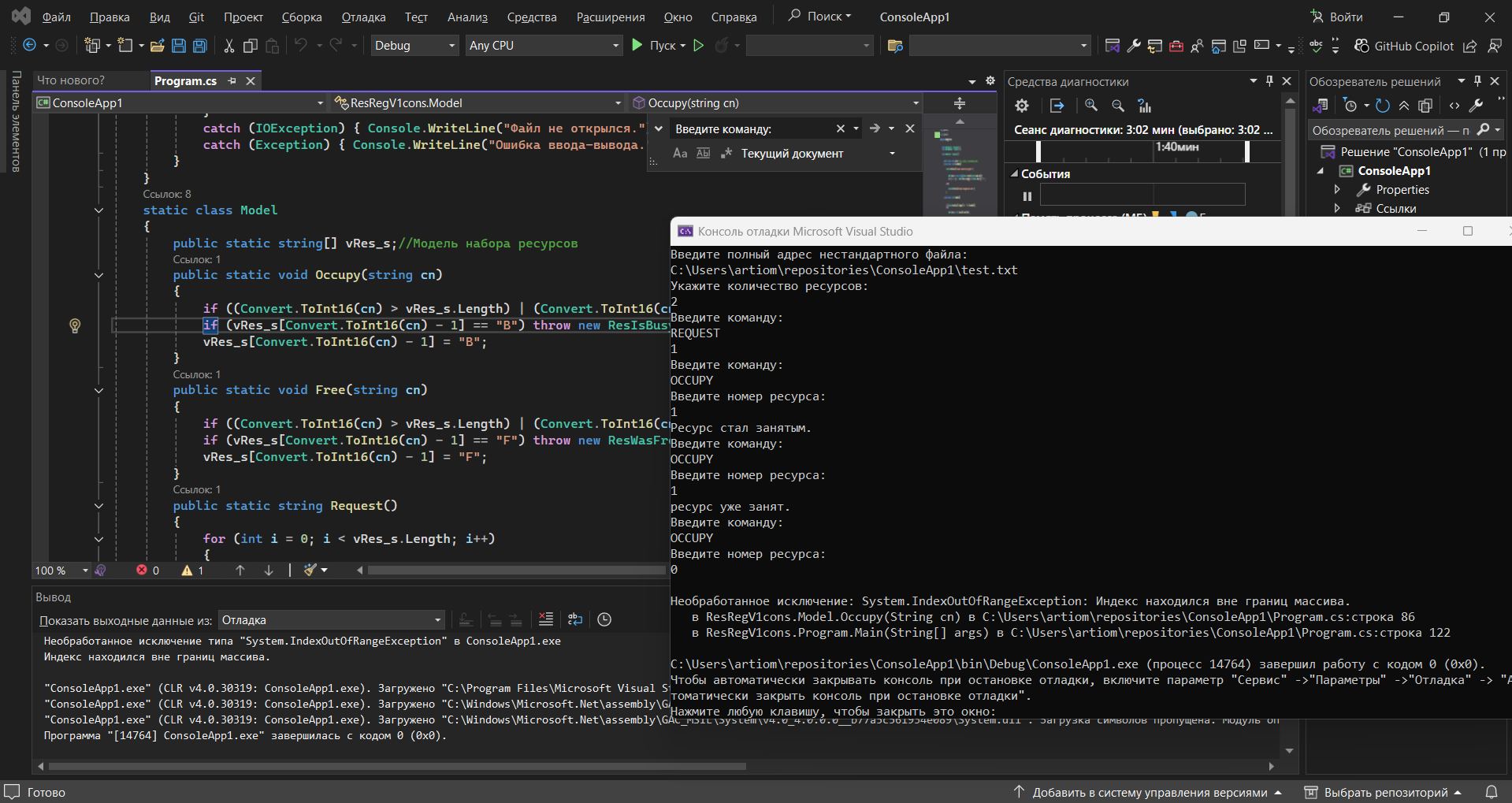
Командой OCCUPY Занимаем первый ресурс



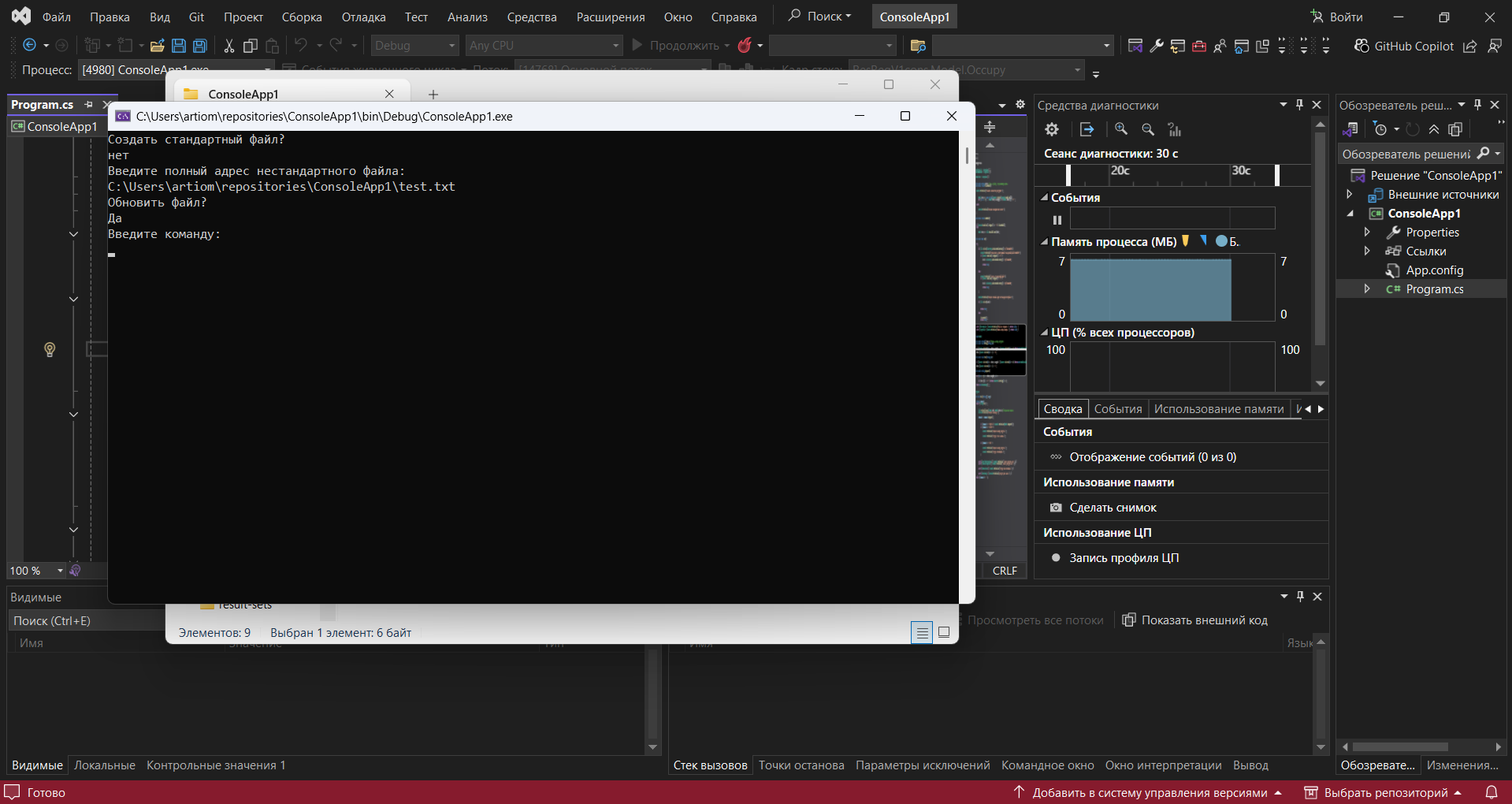
Занятый ресурс нельзя повторно занять



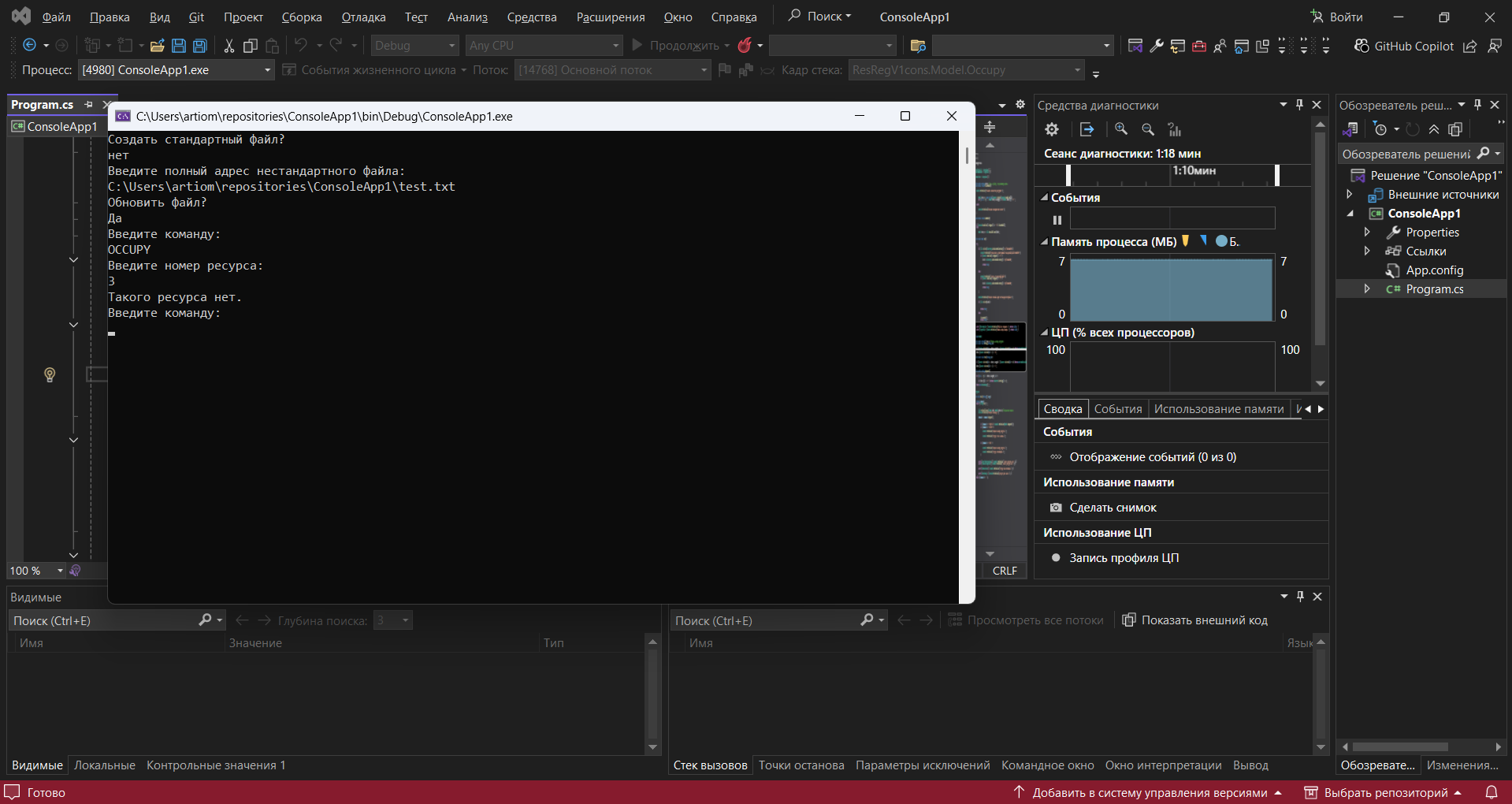
После попытки занять ресурс 0 появилась ршибка



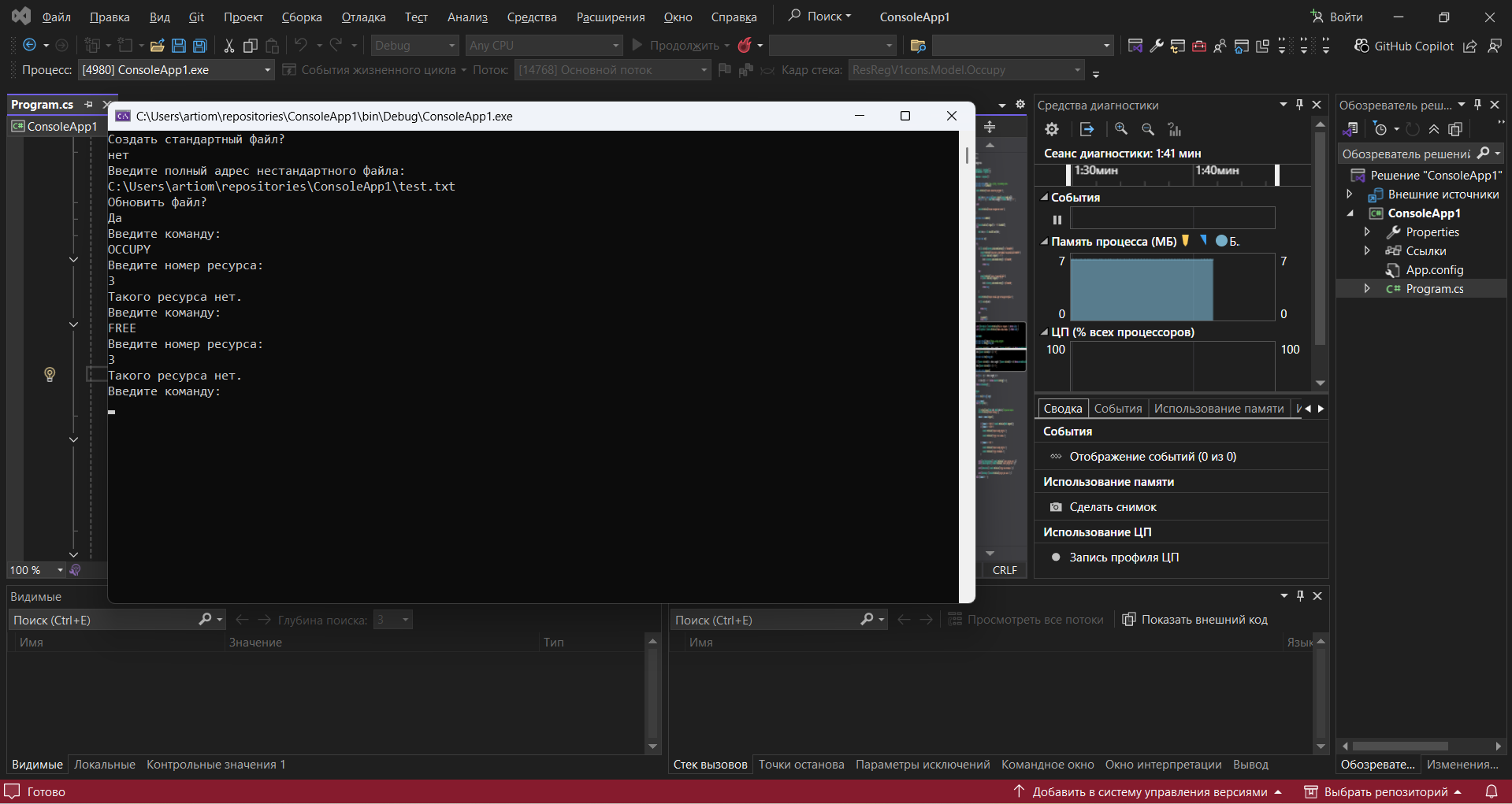
ПОсле перезахода попробовал обновить файл



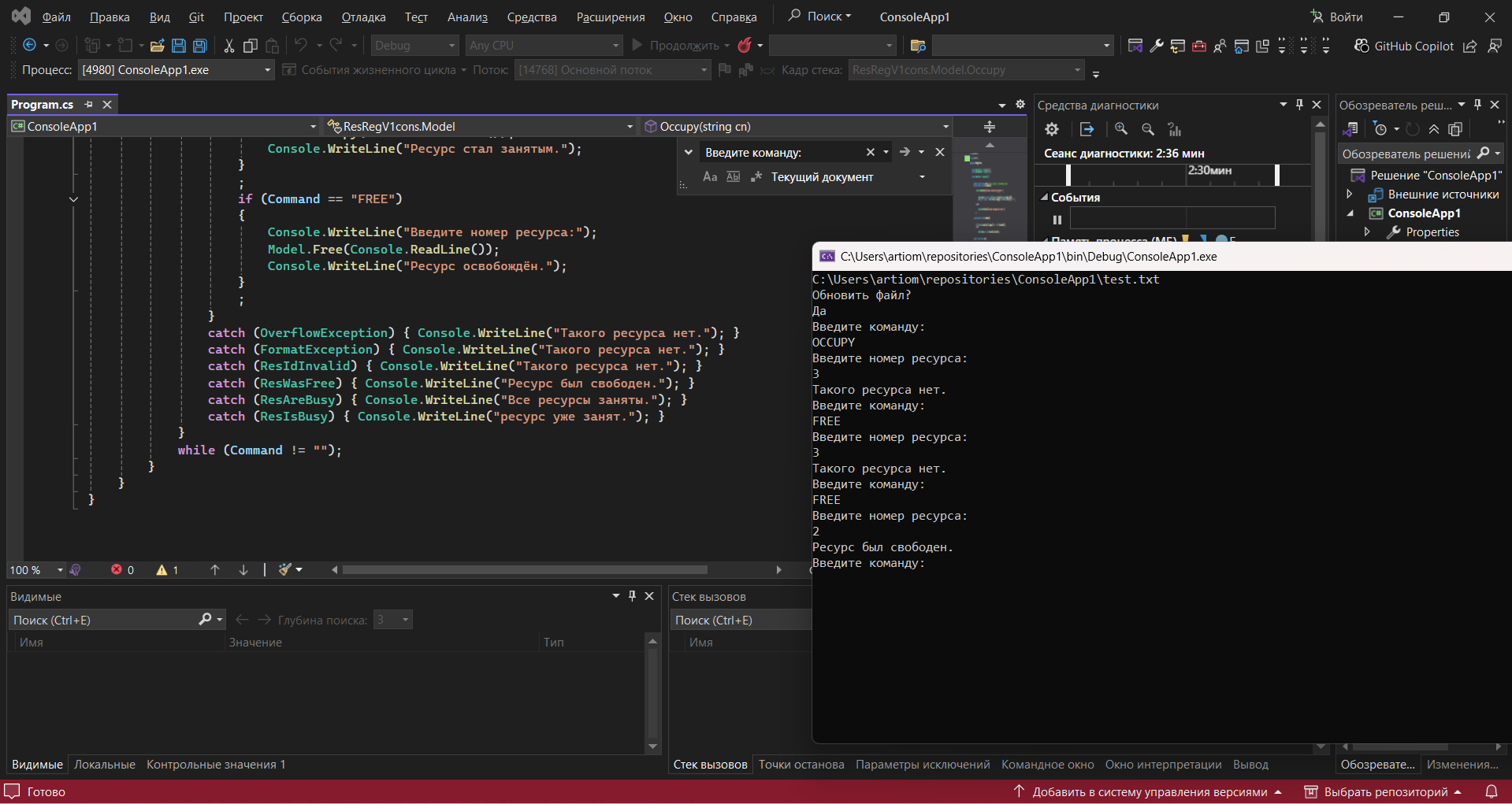
Ресурса 3 нет



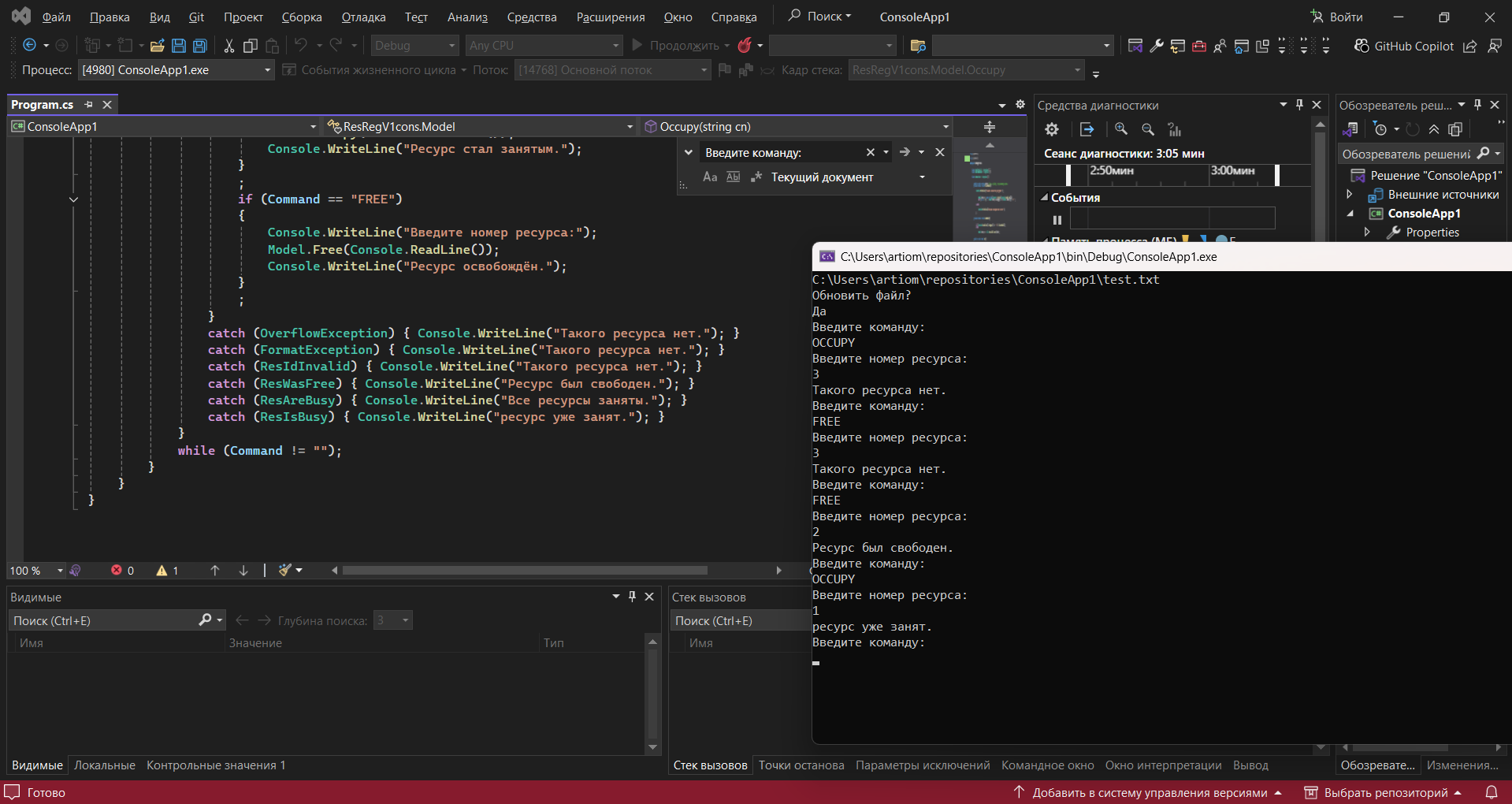
C FREE ресурса 3 тоже нет



Нельзя освободить уже свободный ресурс



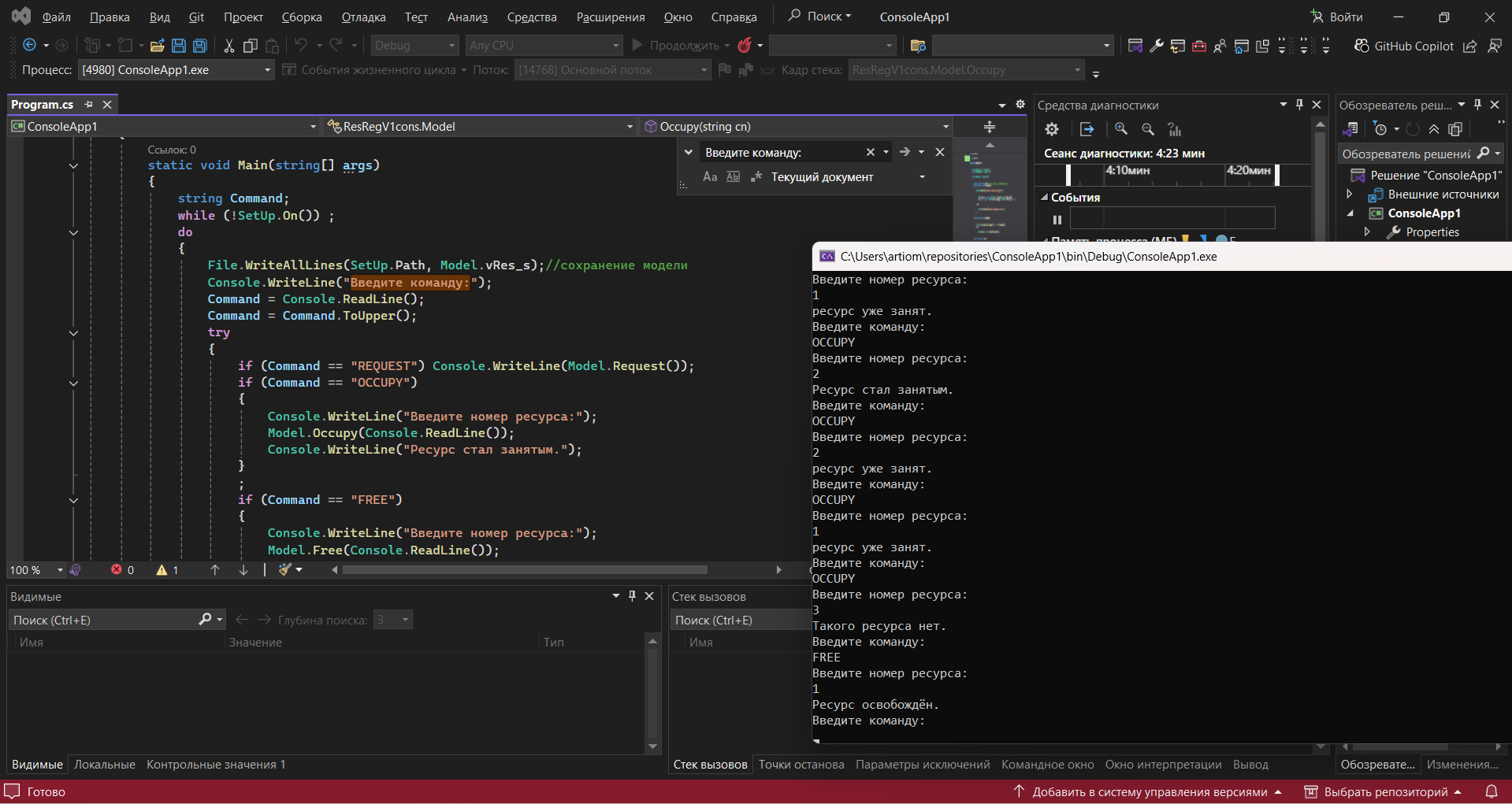
Нельзя занять уже занятый ресурс



Не понял как получить ошибку все ресурсы заняты



Освобождение ресурса



# Что в программе изменить чтобы обеспечить новые функции

Чтобы адаптировать базовую версию программы "Регистратор ресурсов" под вариант "Диверсификация ёмкости ресурса и размера запроса при неделимости запроса", потребуются следующие ключевые изменения:

### 1. Модификация модели данных

Класс `Resource`

Добавить поле ёмкости ресурса (`Capacity`):

class Resource {

public int Id { get; set; }

public int Capacity { get; set; } // Новая характеристика

public bool IsOccupied { get; set; }

public Resource(int id, int capacity) {

Id = id;

Capacity = capacity;

IsOccupied = false;

}

// Проверка возможности обработки запроса

public bool CanHandle(int requestSize) {

return !IsOccupied && Capacity >= requestSize;

}

}

Класс `Request`

Добавить поле размера запроса (`Size`):

class Request {

public int Size { get; set; } // Размер запроса

// Другие параметры (если нужны)

}

### 2. Изменение логики обработки запросов

Метод `Request`

Теперь метод должен находить ресурс с достаточной ёмкостью (`Capacity >= Size`):

// Метод поиска подходящего ресурса

public static string Request(int requestSize)

{

for (int i = 0; i < vRes\_s.Length; i++)

{

if (vRes\_s[i].CanHandle(requestSize))

vRes\_s[i].IsOccupied = true;

return (i + 1).ToString(); // Возвращаем номер ресурса (с 1)

}

throw new ResAreBusy();

}

Метод `Occupy`

Добавить проверку размера запроса:

// Метод занятия ресурса

public static void Occupy(string cn, int requestSize)

{

int index = Convert.ToInt32(cn) - 1; // Индексация с 0

if (index < 0 || index >= vRes\_s.Length)

throw new ResIdInvalid();

Resource resource = vRes\_s[index];

if (resource.IsOccupied || !resource.CanHandle(requestSize))

throw new ResIsBusy();

resource.IsOccupied = true; // Занимаем ресурс

}

### 3. Обновление интерфейса пользователя

Ввод размера запроса

При запросе на занятие ресурса пользователь должен указать размер:

if (Command == "OCCUPY")

{

Console.WriteLine("Введите размер запроса:");

int size = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите номер ресурса:");

Model.Occupy(Console.ReadLine(), size);

Console.WriteLine("Ресурс стал занятым.");

}

### 4. Сохранение и загрузка модели

Модифицировать методы сохранения/загрузки для учёта ёмкости:

// Сохранение

File.WriteAllLines(path, Resources.Select(r => $"{r.Id},{r.Capacity},{r.IsOccupied}"));

// Загрузка

var lines = File.ReadAllLines(path);

foreach (var line in lines) {

var parts = line.Split(',');

int id = int.Parse(parts[0]);

int capacity = int.Parse(parts[1]);

bool isOccupied = bool.Parse(parts[2]);

Resources.Add(new Resource(id, capacity) { IsOccupied = isOccupied });

}

### 5. Обработка ошибок

Добавить исключения для случаев:

- Запрос превышает ёмкость всех ресурсов.

- Попытка разделить запрос.

if (!Resources.Any(r => r.CanHandle(requestSize))) {

throw new Exception("Невозможно обработать запрос: нет ресурса с достаточной ёмкостью.");

}

6. Исправим ClearModel

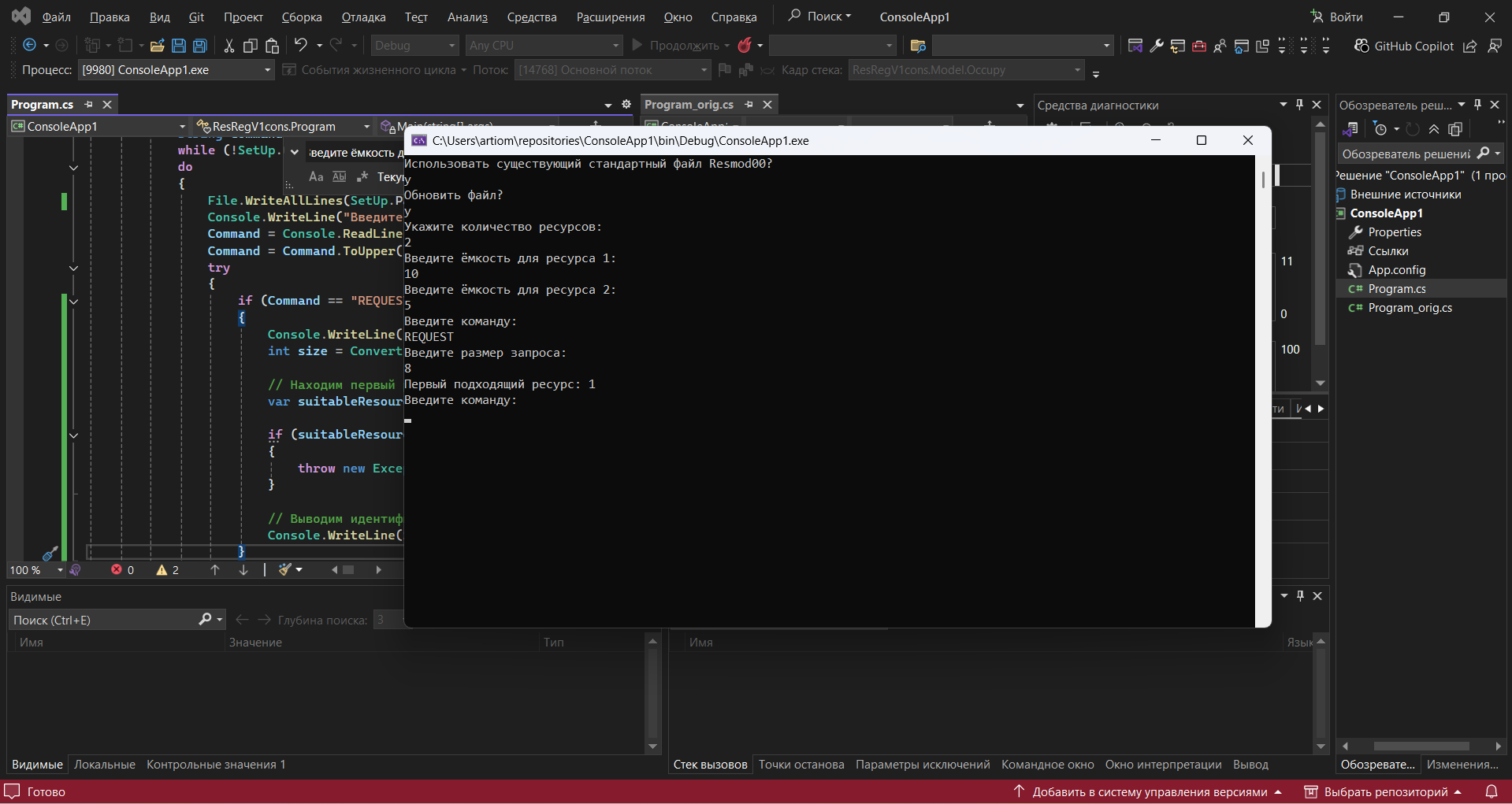
### 6. Тестирование

План тестирования:

1. Позитивный сценарий:

- Ресурсы: `[{Id=1, Capacity=10}, {Id=2, Capacity=5}]`.

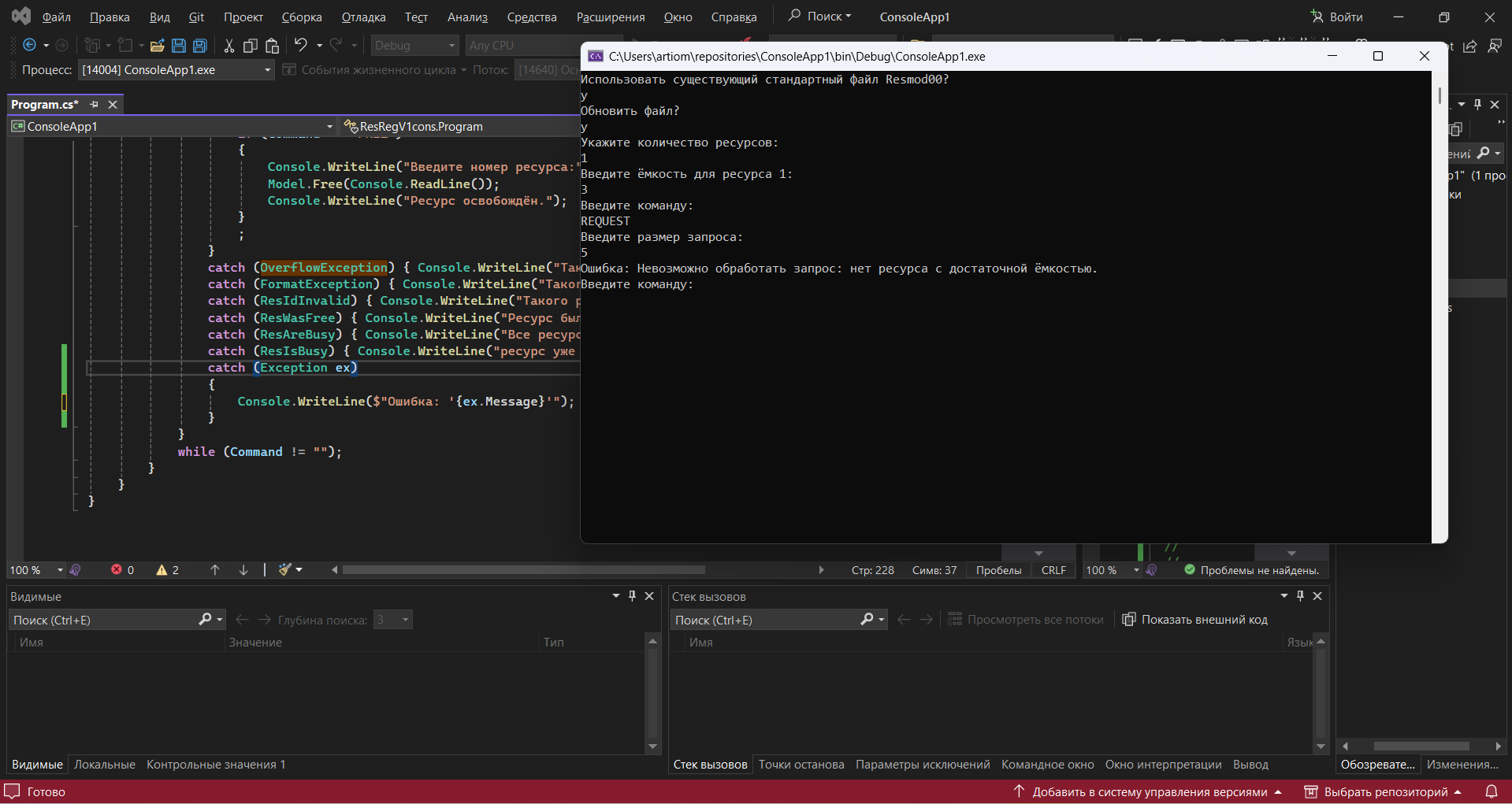
- Запрос: `Size=8` → должен выбрать ресурс 1.



2. Негативный сценарий:

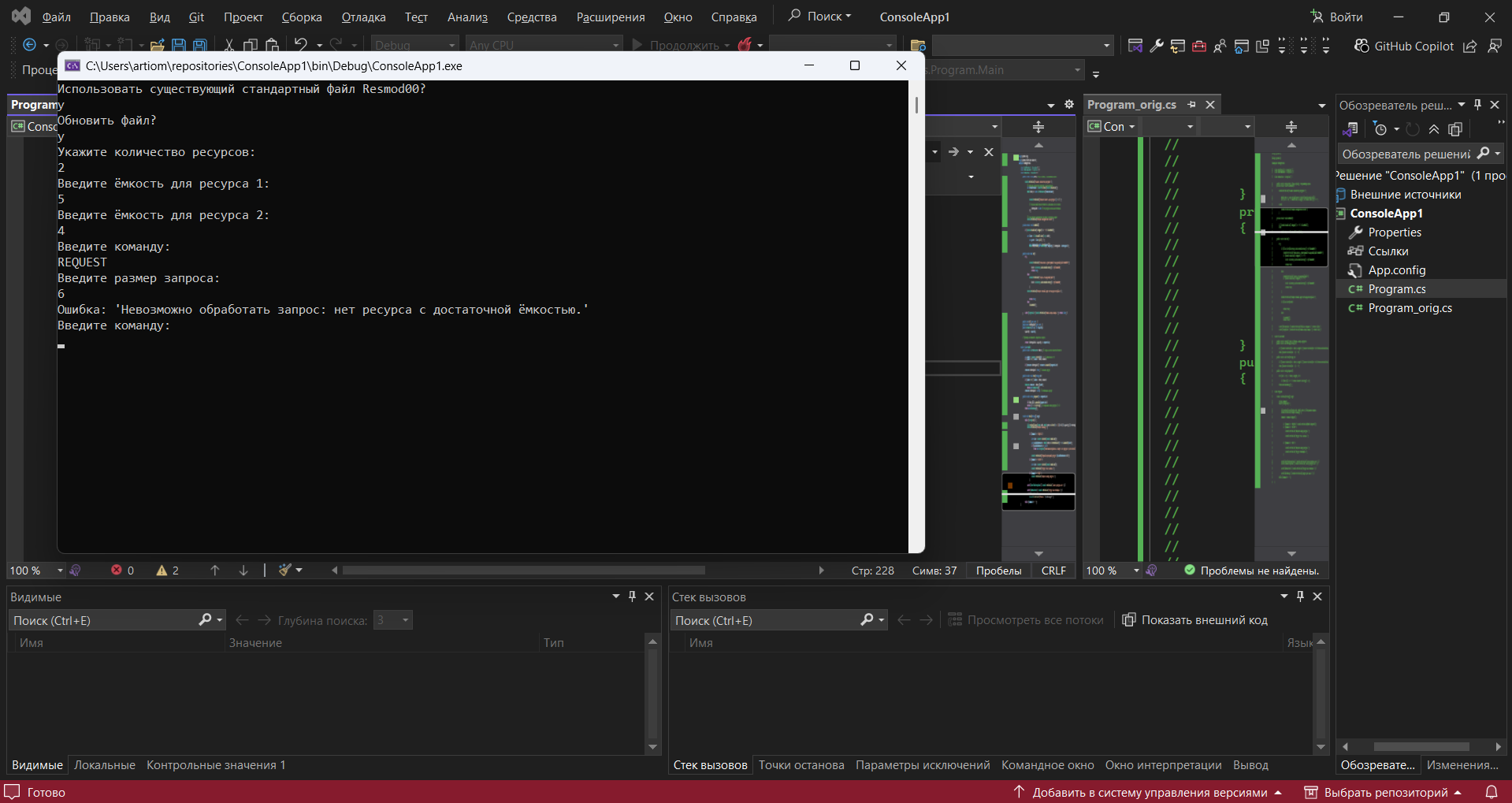
- Ресурсы: `[{Id=1, Capacity=3}]`.

- Запрос: `Size=5` → ошибка.



3. Проверка неделимости:

- Запрос `Size=6` при ресурсах `Capacity=5` и `Capacity=4` → ошибка.



### 7. Диаграммы UML

Диаграмма классов

Добавить атрибуты:

- `Resource`: `+Capacity: int`

Диаграмма последовательности

Обновить методы:

- `HandleRequest(size)` → проверка `Capacity >= Size`.

Как мы видим после этих изменений программа поддерживает разные ёмкости ресурсов и размеры запросов, гарантируя, что запросы обрабатываются целиком без разделения.

Таблица прецедентов

Ссылка на таблицу прецедентов: <https://docs.google.com/spreadsheets/d/1LtgziAT2gaoTo6JkAV0HbDuqImsd6qeGRJfmyq30eHQ/edit?usp=sharing>

## Таблица ручного тестирования

| действие | ожидаемый результат | фактический результат | статус |
| --- | --- | --- | --- |
| 1. Запустить программу и выбрать создание новой модели с 5 ресурсами. | Программа запросит количество ресурсов (5) и ёмкость для каждого ресурса. Все ресурсы будут свободны (IsOccupied = false). | Программа запросила количество ресурсов и ёмкость. Все ресурсы отображены как свободные. | Пройдено |
| 2. Ввести команду REQUEST с размером запроса, который может быть обработан. | Программа найдет первый подходящий ресурс и выведет его номер. Состояние ресурса не изменится. | Программа вывела номер первого подходящего ресурса. Состояние ресурса осталось свободным. | Пройдено |
| 3. Ввести команду OCCUPY с номером ресурса и размером запроса. | Программа проверит возможность занятия ресурса и выведет сообщение "Ресурс стал занятым". Состояние ресурса изменится на IsOccupied = true. | Программа вывела сообщение "Ресурс стал занятым". Состояние ресурса изменилось. | Пройдено |
| 4. Ввести команду FREE с номером занятого ресурса. | Программа проверит состояние ресурса и выведет сообщение "Ресурс освобождён". Состояние ресурса изменится на IsOccupied = false. | Программа вывела сообщение "Ресурс освобождён". Состояние ресурса изменилось. | Пройдено |
| 5. Ввести команду REQUEST с размером запроса, превышающим ёмкость всех ресурсов. | Программа выведет сообщение об ошибке: "Невозможно обработать запрос: нет ресурса с достаточной ёмкостью." | Программа вывела сообщение об ошибке. | Пройдено |
| 6. Сохранить модель и перезапустить программу. | После перезапуска программа предложит использовать сохранённый файл. При выборе "Y" модель загрузится с теми же состояниями ресурсов. | Программа предложила использовать сохранённый файл. После выбора "Y" модель загрузилась с сохранёнными состояниями. | Пройдено |
| 7. Ввести команду OCCUPY с номером уже занятого ресурса. | Программа выведет сообщение об ошибке: "ресурс уже занят." | Программа вывела сообщение об ошибке. | Пройдено |
| 8. Ввести команду FREE с номером уже свободного ресурса. | Программа выведет сообщение об ошибке: "Ресурс был свободен." | Программа вывела сообщение об ошибке. | Пройдено |
| 9. Ввести некорректную команду (например, "INVALID"). | Программа проигнорирует команду или выведет сообщение об ошибке. | Программа вывела сообщение об ошибке. | Пройдено |
| 10. Проверить цикличность работы программы. | Программа продолжает работать, пока пользователь не введёт пустую строку для выхода. | Программа продолжала работать после выполнения всех команд. | Пройдено |