Описание технологий

Для разработки приложения Workers Accounter мы будем использовать язык программирования C# и свод парадигм программирования и проектирования методологии ООП (англ. - Object-oriented programming).

C# - это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он широко используется для создания приложений под платформу .NET Framework.

Объектно ориентированное программирование (сокр. ООП) — методология или стиль программирования на основе описания типов/моделей предметной области и их взаимодействия, представленных порождением из прототипов или как экземпляры классов, которые образуют иерархию наследования.

В приложении будет реализовано 3 основные парадигмы ООП:

1. Инкапсуляция - объединение данных с методами, которые работают с этими данными, или ограничение прямого доступа к некоторым компонентам объекта. Инкапсуляция используется для скрытия значений или состояния структурированного объекта данных внутри класса, предотвращая прямой доступ к ним.
2. Полиморфизм - способность классов с одинаковой спецификацией (интерфейсом) определять различную реализацию, что, в свою очередь, позволяет клиентскому коду абстрагироваться от этой самой реализации и работать с классом, исходя из его спецификации.
3. Наследование - это концепция, согласно которой одни классы, называемые родительскими, могут лежать в основе других — дочерних. При этом, дочерние классы перенимают свойства и поведение своего родителя.

Технологическая часть.

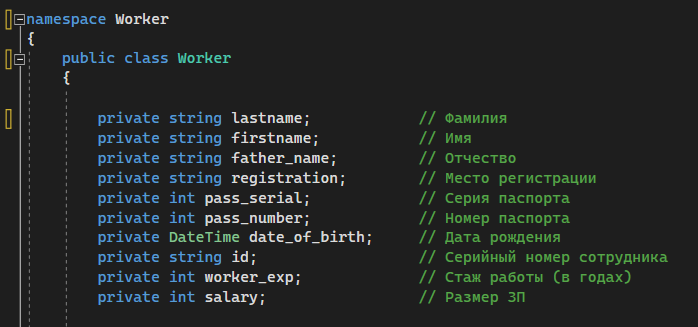
1. Описание базового класса – контейнера - Worker.

Для произведения манипуляций над “карточками” сотрудников, что является основной целью приложения, требуются классы контейнеры, реализующие вышеупомянутые “карточки” сотрудников. В данном случае, компании требуется 3 типа сотрудников, для которых и нужен базовый класс “Worker”.

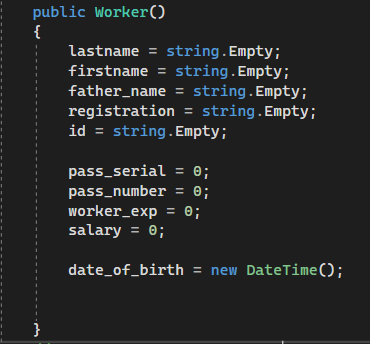
Класс должен содержать основные параметры работника, такие как ФИО, дату рождения, серию и номер паспорта и т.д., которые должны иметь модификаторы доступа private, а так же сеттеры и\или геттеры (модификатор доступа public) для соблюдения 1-ой парадигмы ООП – Инкапсуляции.

Задается пространство имен, чтобы была возможность подключить пространство имен в другом файле

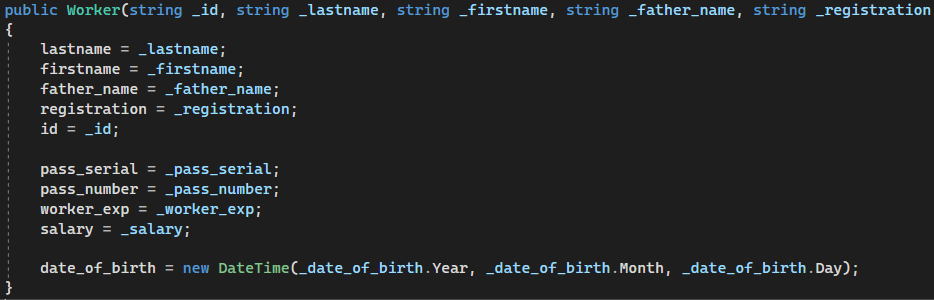
Создается базовый (основной) класс-сотрудник и объявляем переменные



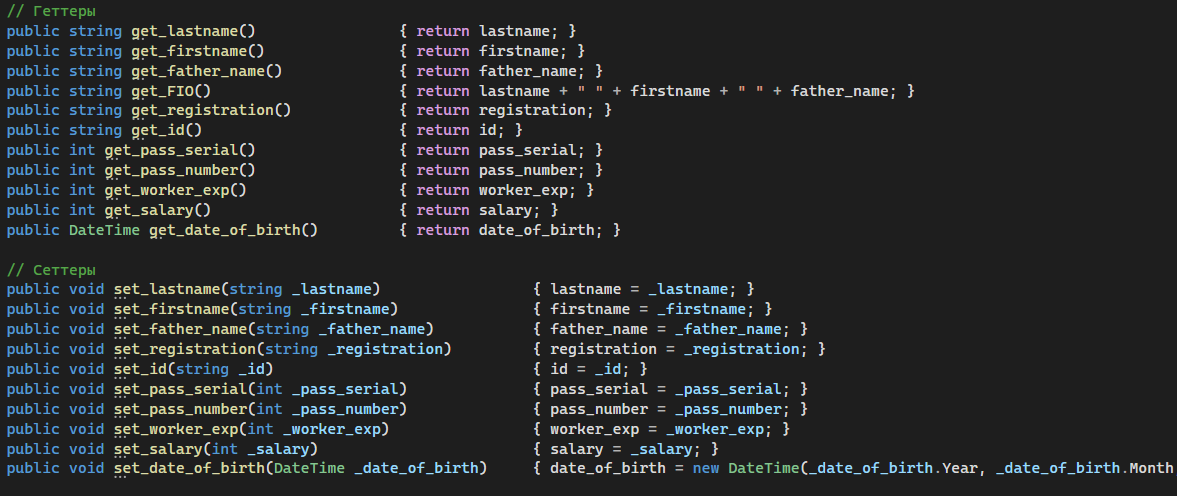
Описываем конструктор по умолчанию (Пустой конструктор) и инициализируем переменные пустыми значениями.



Описываем конструктор с параметрами



Геттеры и Сеттеры

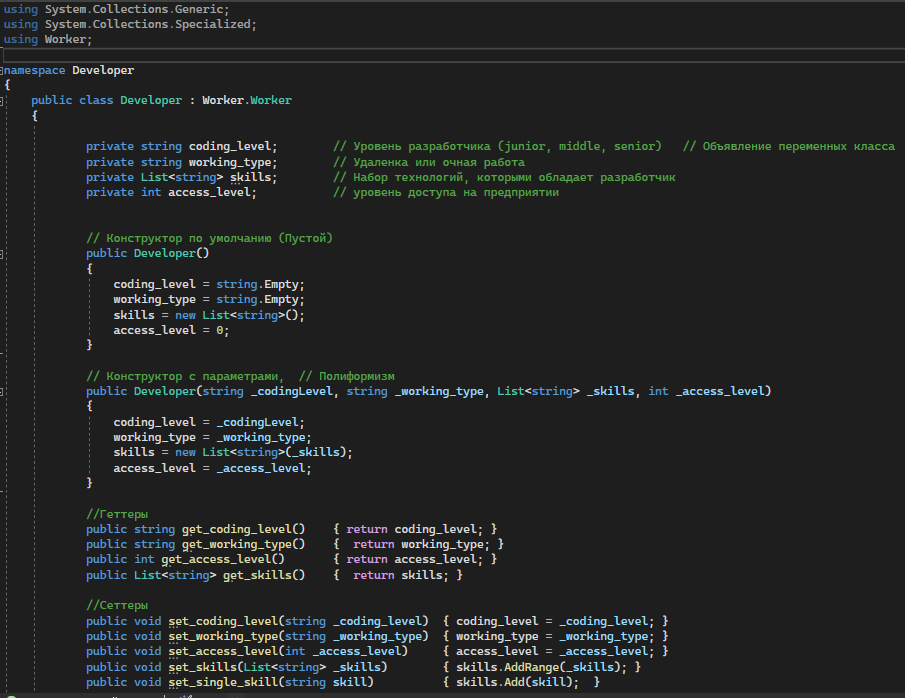


1. Организация дочерних классов – контейнеров специализаций сотрудников.

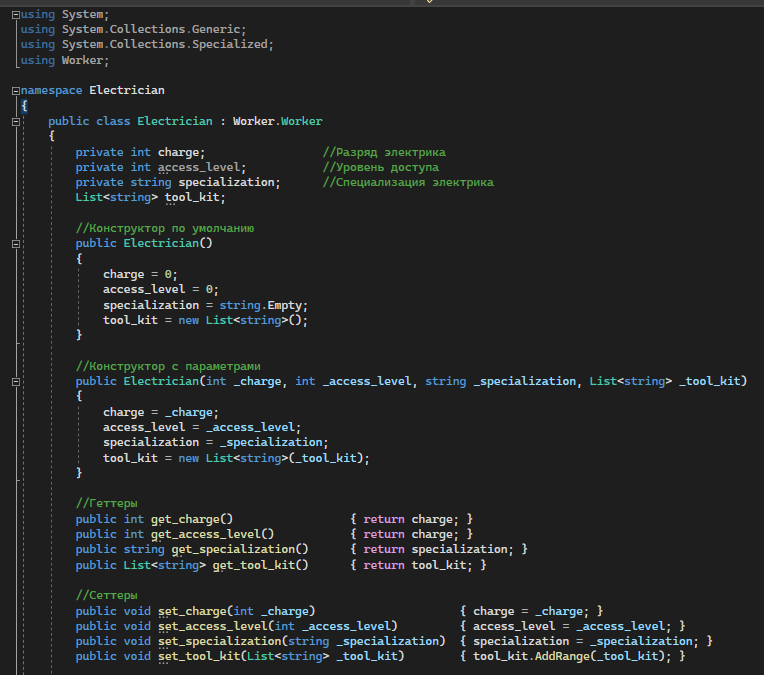
Далее, требуется реализовать несколько дочерних классов, каждый из которых характеризует конкретную профессию. Решение реализовать классы сотрудников отдельно по каждой профессии было принято в связи с тем, что карточки сотрудников разных профессий имели разное содержание (разные поля данных). В данном случае реализуется 3-я парадигма ООП – наследование.

Реализация дочерних классов в проекте:

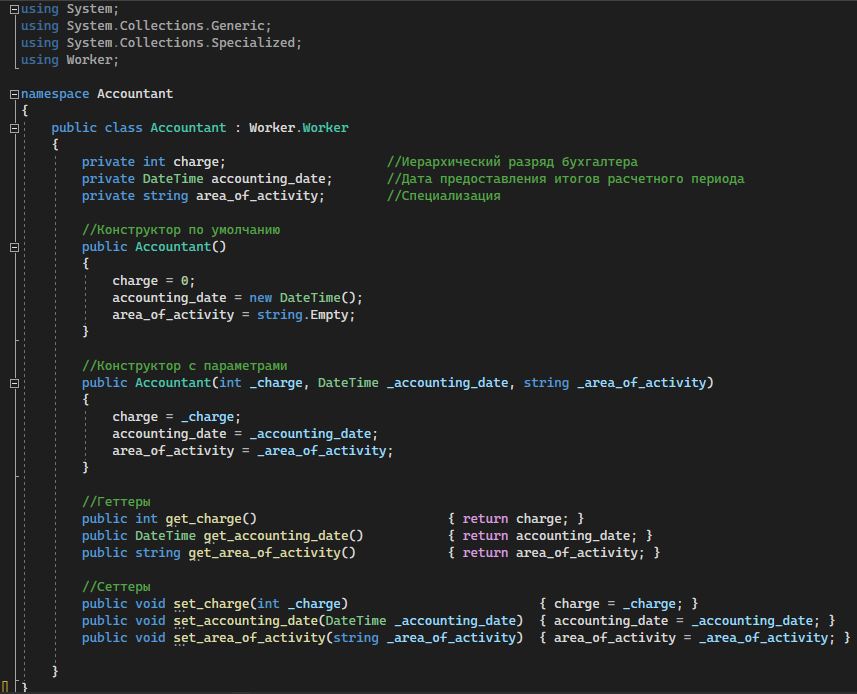
(Класс Разработчик (Developer))



(Класс Электрик (Electrician))



(Класс Бухгалтер (Accountant))



1. В каждой программе должен присутствовать центр программы, там, где сосредоточен основной функционал. В данном случае, это основной класс с названием “WorkersAccounter”. В нем описаны все основные методы, реализующие функционал программы, такие как просмотр списка сотрудников, добавление или удаление карточки сотрудников, и т.д. Именно к этим методам через интерфейс программы будет обращаться пользователь. Отметим, что метод добавления полиморфен – реализован трижды, но с разными входными данными и телами. Этот прием олицетворяет 2-ю парадигму ООП – полиморфизм.

На данном классе следует остановиться поподробнее, так как это единственный класс, в котором методы не ограничиваются конструкторами, сеттерами и геттерами. Соответственно вышеупомянутые методы мы затрагивать не будем.

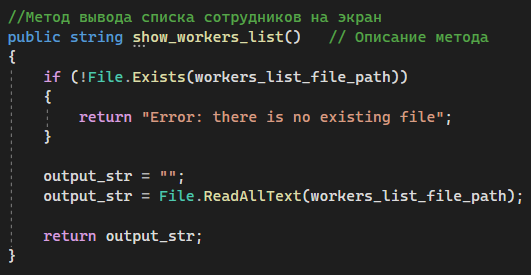
Итак.

1. string show\_workers\_list();

Данный метод возвращает список работников с формате:

[id, полное имя, профессия ]. Считывание списка из файла осуществляется посредством использования вызова метода File.ReadAllText(), в аргументы которого передается путь к файлу списка сотрудников.

Листинг метода string show\_workers\_list();



1. string add\_worker();

Данный метод реализует функцию добавления сотрудника,

и так как он является полиморфным, имеет смысл описать только один из трех полиморфных методов.

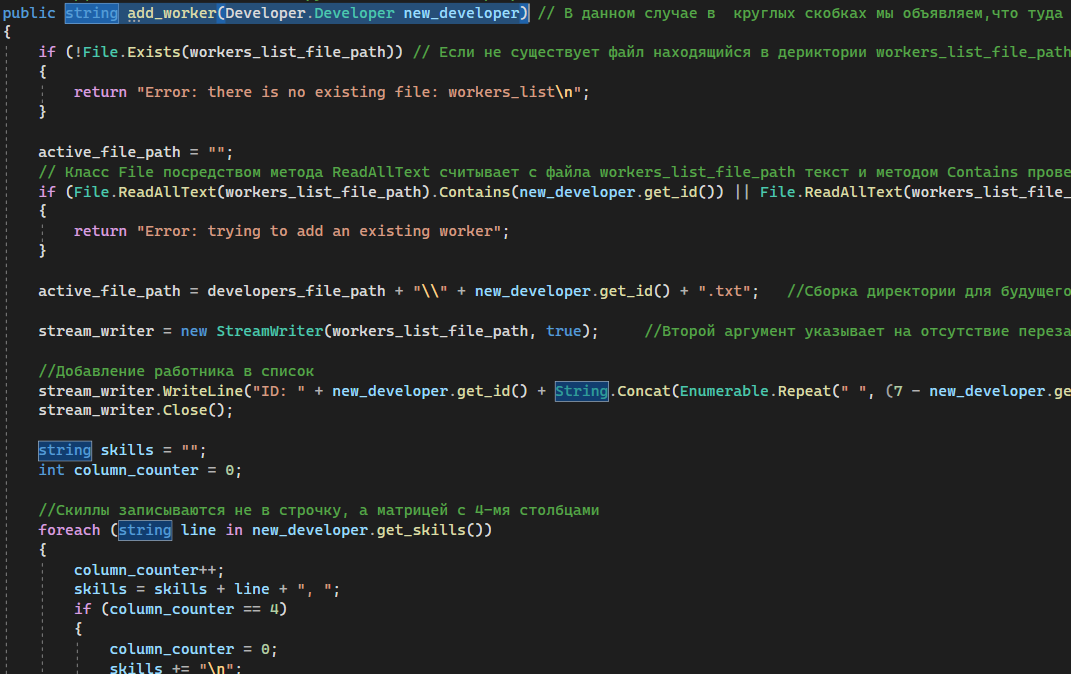
Прежде всего проверяется совпадение нового сотрудника по id и ФИО.

Далее идет “сборка” директории, по адресу которой будет находиться карточка нового сотрудника.

После сборки директории, происходит запись данных сотрудника с список сотрудников.

Далее происходит запись всех данных сотрудника в объект сотрудника,

После чего возвращается метод возвращает строку, сигнализирующую об успешном добавлении работника.

Листинг метода string add\_worker(); 

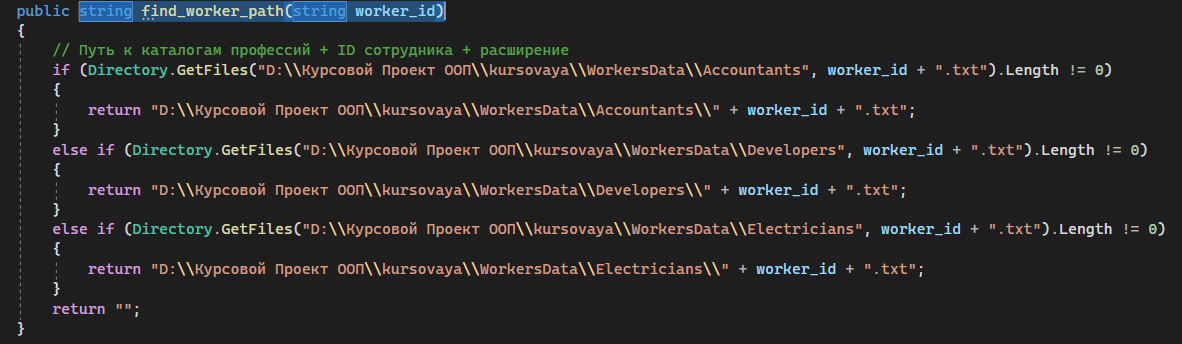


1. string find\_worker\_path(string worker\_id);

Метод, который по id сотрудника ищет его карточку.

Метод проходится по директории каждой профессии, и в каждой проверяет наличие карточки с соответствующим Id. Метод возвращает полный путь к файлу, либо пустую строку, если файл не найден.

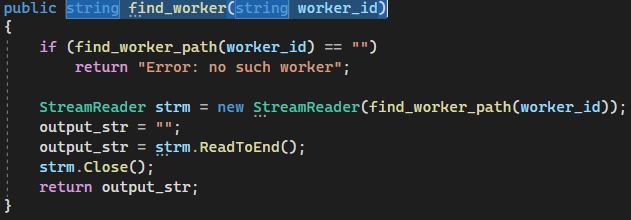
Листинг метода string find\_worker\_path();



1. string find\_worker(string worker\_id);

Метод, возвращающий полную карточку сотрудника, который будет найден с помощью метода find\_worker\_path.

Листинг метода string find\_worker(string worker\_id);



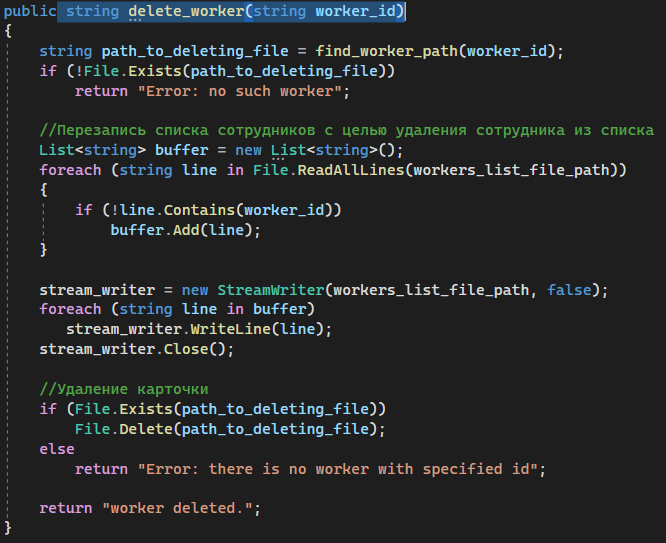
1. string delete\_worker(string worker\_id);

Метод, выполняющий удаление сотрудника из списка сотрудников, а так же удаление карточки сотрудника.

Метод выполняет поиск файла сотрудника по его id, после чего

считывает в буфер весь список сотрудников, пропуская строку, содержащую вышеуказанный id, и перезаписывает список сотрудников с помощью этого буфера, после чего метод спокойно удаляет файл с подходящим id.

Листинг метода string delete\_worker(string worker\_id);



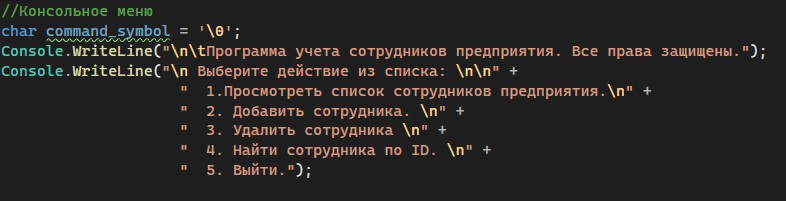
88888888888

1. И наконец, интерфейс программы. В нашем случае, он реализован в классе “Program”, который является дефолтным в любом, даже пустом проекте на C#. В этом классе содержится лишь один метод “Main”, который и является отправной точкой после запуска программы, поэтому в нем и находится весь небольшой интерфейс, через который и будет общаться с программой пользователь.

В данном случае метод Main состоит из вывода меню, в котором пользователю предлагается выбрать одну из функций приложения,

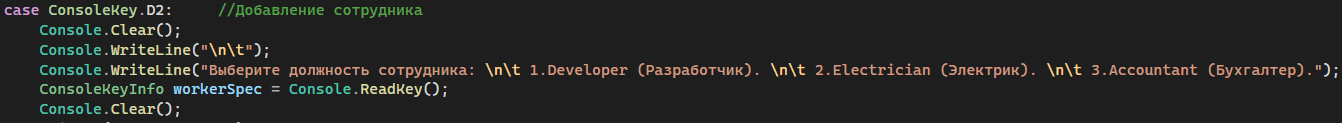
После чего считывается следующая нажатая клавиша, и выбирается, какой метод класса “WorkersAccounter” выбрать (с помощью конструкции switch case).

Листинг вызова меню:



Следует отметить, что при выборе “Добавить сотрудника”, предлагается выбор профессии, которой обладает добавляемый сотрудник, от чего зависит последующая конструкция ввода-вывода создания карточки сотрудника.

Листинг вызова выбора профессии сотрудника:



Рассмотрим добавление сотрудника на примере выбора профессии “Разработчик”.

Ввод запроса программы делятся на два типа:

1. Рукописный ввод.
2. Ввод по нажатию клавиши, и соответственно, выбор варианта из списка.

Первый работает по принципу “Прочитал, что требуется ввести – ввел – подтвердил клавишей ENTER”.

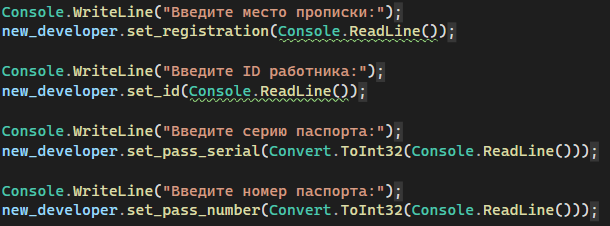
Второй же работает по принципу “Прочитал, выбрал вариант ответа – нажал соответствующую клавишу”.

Первый организует ввод по системе “Вызвать сеттер, в аргументах которого указать считывание строки с подтверждением ввода”.

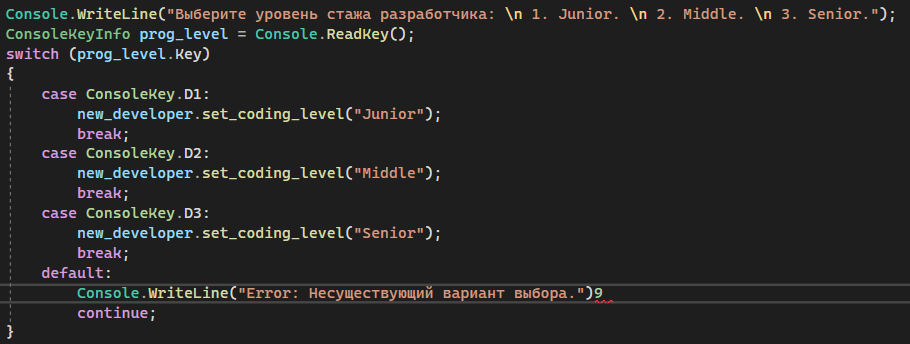
Второй организует ввод по системе “Считать следующий символ, передать его в конструкцию switch case”.

После ввода всех параметров карточки сотрудника, вызывается метод add\_worker(); и выводится текстовое сообщение об успешном добавлении сотрудника.

Пример - листинг ввода-вывода 1 типа:



Пример – листинг ввода – вывода 2 типа:

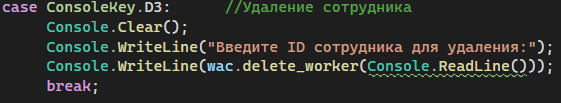


Далее, рассмотрим метод удаления сотрудника.

Здесь все просто:

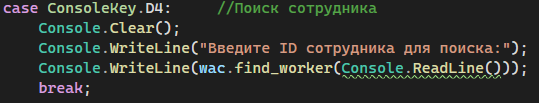
1. Очищается консоль.
2. Происходит вывод сообщения с дальнейшими инструкциями.
3. В консоль выводится результат вызова метода удаления карточки сотрудника.

Листинг вызова метода удаления сотрудника:



Метод поиска сотрудника по ID практически аналогичный:

1. Очищается консоль.
2. Происходит вывод сообщения с дальнейшими инструкциями.
3. В консоль выводится результат вызова метода поиска карточки сотрудника по его ID.

Листинг вызова метода поиска сотрудника:

Крайним в интерфейсе является выход из программы, который очищает строку и прерывает бесконечный цикл, в котором заключен весь интерфейс программы.

Листинг выхода из цикла:

