ЛАБОРАТОРНАЯ РАБОТА №3
РАБОТА СО СПИСКОМ ПРОЦЕССОВ ОС И ОПРЕДЕЛНИЕ СОСТАВЛЯЮЩИХ ПК

Цель работы – Приобрести практические навыки получения списка процессов ОС, составляющих ПК и работы с ними.

Теоретическая часть

1. Работа со список процессов ОС

Класс Process (Пространство имен: System.Diagnostics). Этот класс может получать информацию об уже запущенных процесса или создавать новые. Класс Process имеет следующие свойства:

1. Свойство Handle: возвращает дескриптор процесса.
2. Свойство Id: получает уникальный идентификатор процесса в рамках текущего сеанса ОС.
3. Свойство MachineName: возвращает имя компьютера, на котором запущен процесс.
4. Свойство Modules: получает доступ к коллекции ProcessModuleCollection, которая хранит набор модулей (файлов dll и exe), загруженных в рамках данного процесса.
5. Свойство ProcessName: возвращает имя процесса, которое нередко совпадает с именем приложения.
6. Свойство StartTime: возвращает время, когда процесс был запущен
7. Свойство VirtualMemorySize64: возвращает объем памяти, который выделен для данного процесса.

Класс Process имеет следующие методы:

1. Метод CloseMainWindow(): закрывает окно процесса, который имеет графический интерфейс.
2. Метод GetProcesses(): возвращающий массив всех запущенных процессов.

3. Метод GetProcessesByName(): возвращает процессы по его имени. Так как можно запустить несколько копий одного приложения, то возвращает массив.

4. Метод Kill(): останавливает процесс.

5. Метод Start(): запускает новый процесс.

Получение списка запущенных процессов

Для получения списка запущенных процессов необходимо выполнить метод Process.GetProcesses(). Далее вывести информацию в удобочитаемом виде. Каждый элемент полученного массива процессов можно описать набором свойств Id и ProcessName.

```
foreach (Process process in Process.GetProcesses())
{
      // Обработчик
}
```

Пример программы для получения списка процессов представлен на рисунке 1.

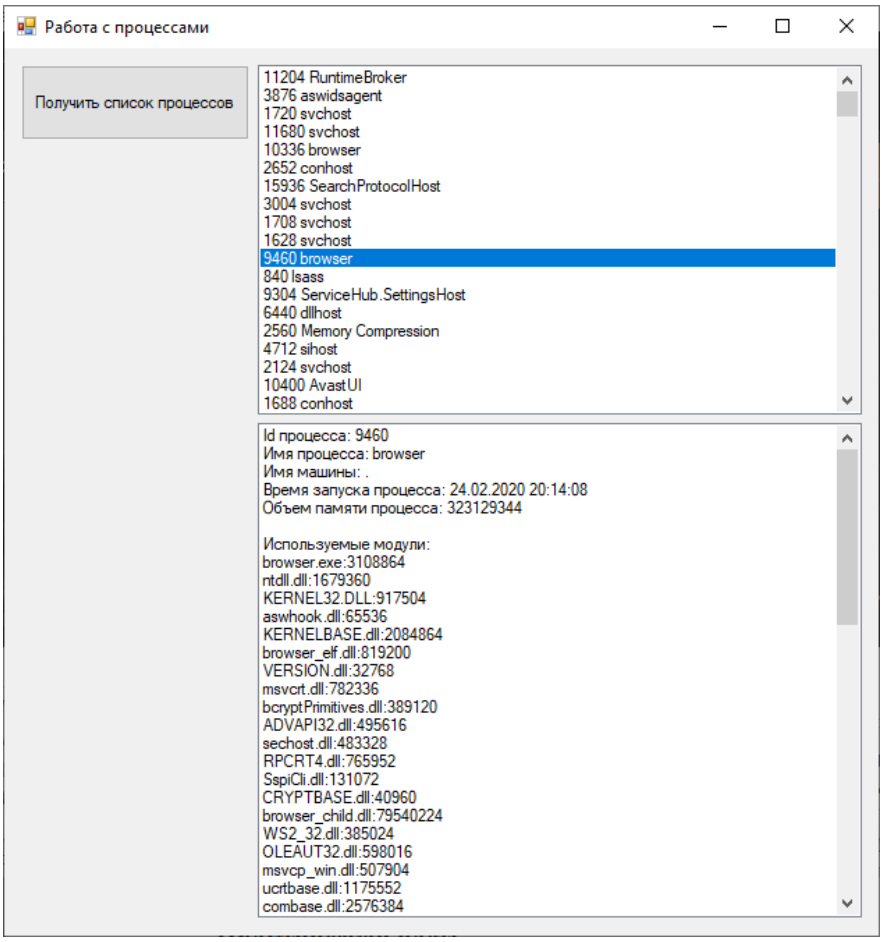
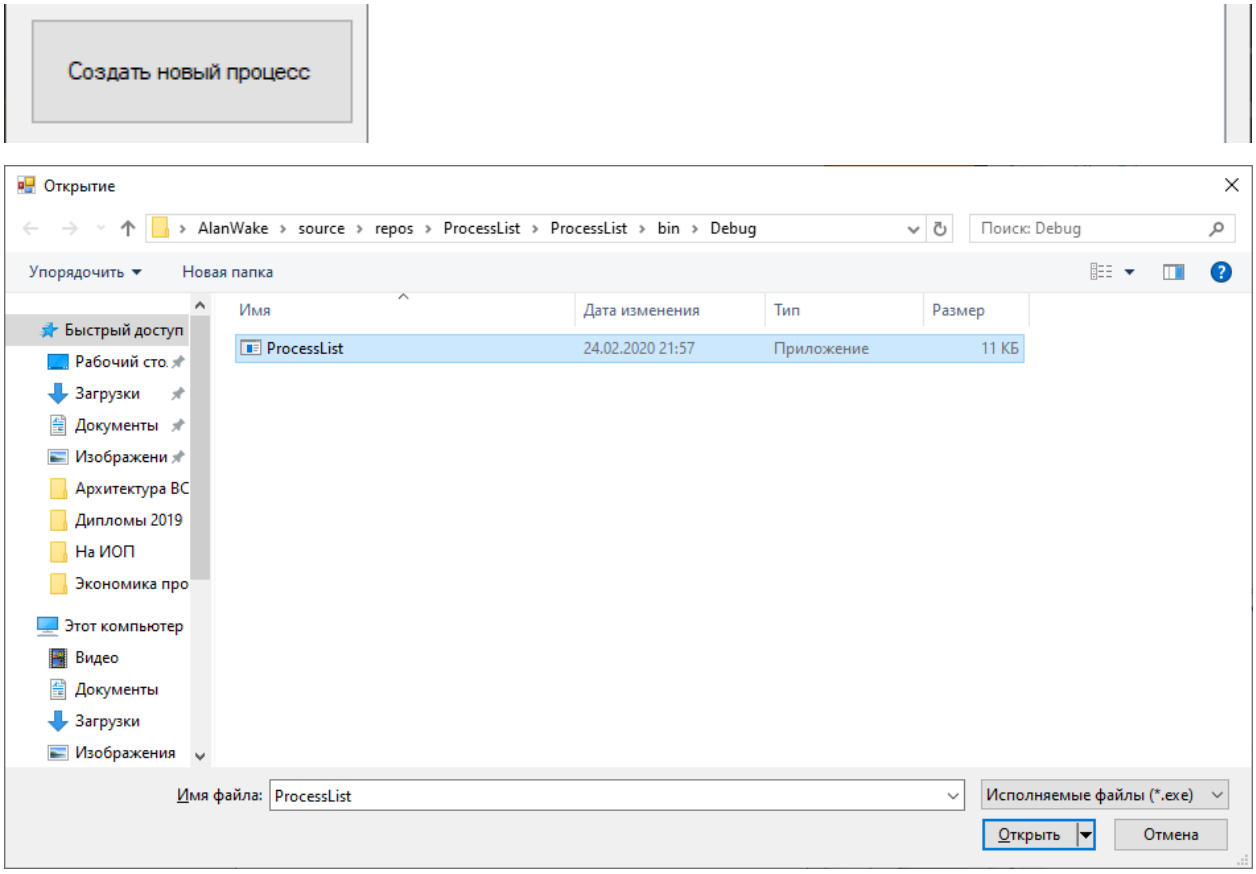


Рисунок 1 – программа, получающая список процессов

Запуск нового процесса

Для запуска нового процесса используется метод Start. Для этого необходимо передать расположение исполняемого файла в качестве входного параметра. Пример создания нового процесса представлен на рисунке 2. Также возможно передавать атрибуты для исполняемых файлов через запятую. Либо можно использовать класс ProcessStartInfo. В нем необходимо указать ProcessStartInfo.FileName – путь к исполняемому файлу и ProcessStartInfo.Arguments – аргументы. Например: запустить браузер с аргументом в виде адреса Интернет-ресурса.



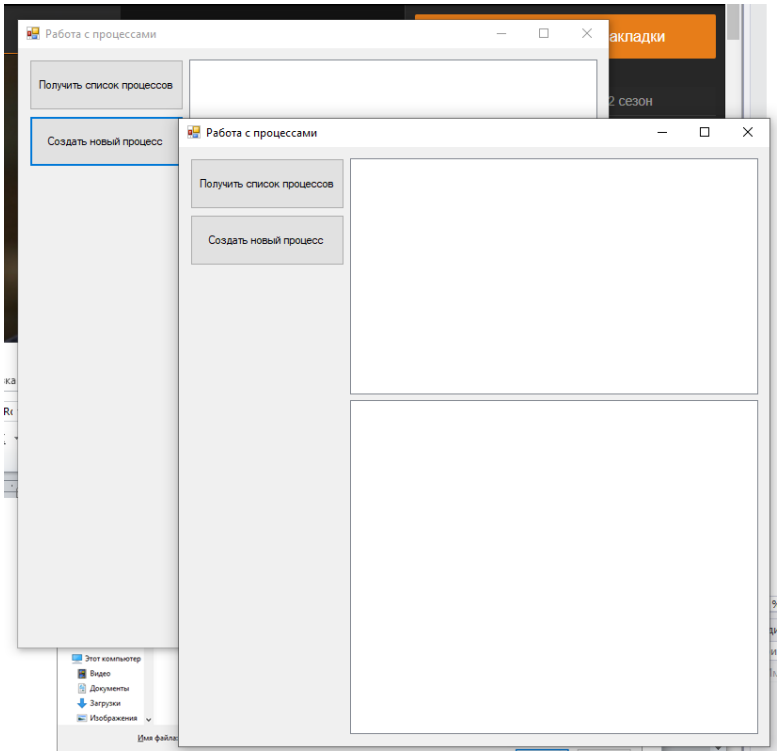


Рисунок 2 – пример создания процесса

2. Определение составляющих ПК

Есть два основных инструмента, с помощью которых можно извлечь информацию из любой версии Windows: System.Environment и System.Management.

Класс Environment - Предоставляет сведения о текущей среде и платформе, а также необходимые для управления ими средства. Этот класс не наследуется.

Свойства и методы класса Environment:

Environment.CurrentDirectory - Получение текущей директории.

Environment.ExitCode - Возвращает или задает код выхода из процесса. 32-битовое целое число со знаком, содержащее код выхода. Значение по умолчанию 0 (нуль), что соответствует успешно выполненному процессу.

Environment.HasShutdownStarted - Возвращает значение, указывающее, выгружается ли текущий домен приложения или среда CLR завершает работу.

Environment.MachineName - Возвращает имя машины.

Environment.OSVersion - Возвращает версию ОС.

Environment.Is64BitOperatingSystem – Возвращает значение типа bool, указывающее является ли операционная система 64-разрядной.

Environment.ProcessorCount – Возвращает число процессоров.

Environment.StackTrace - Возвращает текущие сведения о трассировке стека.

Environment.SystemDirectory - Возвращает путь к системной папке.

Environment.TickCount - Возвращает время, истекшее с момента загрузки системы (в миллисекундах).

Environment.UserDomainName - Возвращает имя сетевого домена, связанное с текущим пользователем.

Environment.UserInteractive - Возвращает значение, позволяющее определить, выполняется ли текущий процесс в режиме взаимодействия с пользователем. Значение true , если текущий процесс выполняется в режиме взаимодействия с пользователем; в противном случае — значение false.

Environment.UserName - Возвращает имя пользователя, сопоставленное с текущим потоком.

Environment.Version - Возвращает объект Version, который описывает основной и дополнительный номера, а также номер построения и редакции среды CLR.

Environment.WorkingSet - Возвращает объем физической памяти, сопоставленной контексту процесса.

Environment.ExpandEnvironmentVariables - Замещает имя каждой переменной среды, внедренной в указанную строку, строчным эквивалентом значения переменной, а затем возвращает результирующую строку.

Environment.SpecialFolder.System - Возвращает путь к особой системной папке, указанной в заданном перечислении.

Environment.GetLogicalDrives – Возвращает массив строк с именами дисков системы.

На рисунке 2 представлен пример работы программы.

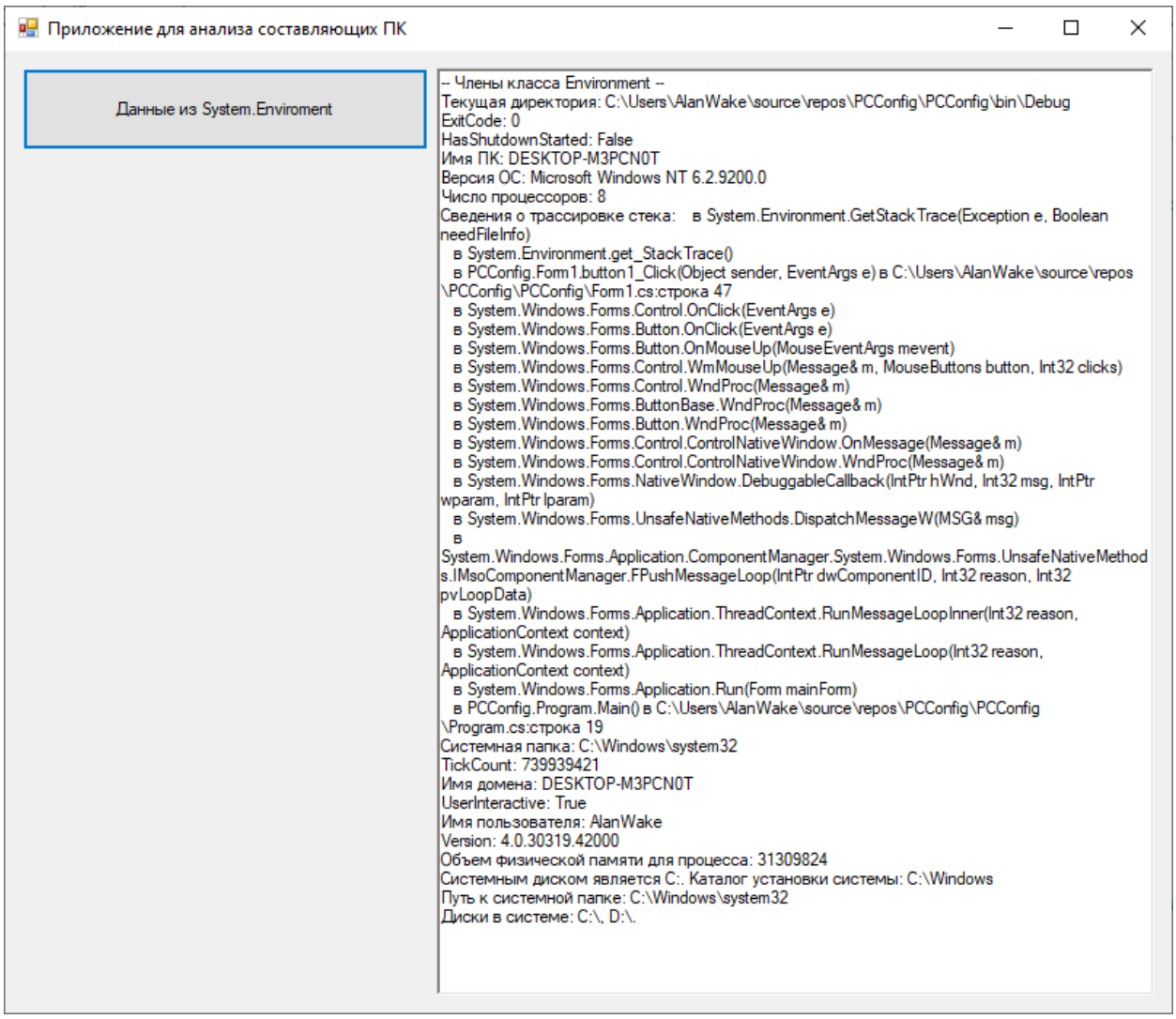


Рисунок 2 - пример работы программы

Пространство имен System.Management

Средства доступа к обширному набору сведений и событий управления, относящихся к системе, устройствам и приложениям, поддерживающим инфраструктуру WMI (Windows Management Instrumentation — инструментарий управления Windows). Приложения и службы могут запрашивать важные сведения об управлении (например, об объеме свободного места на диске, текущем уровне загрузки процессора, о том, к какой базе данных подключено конкретное приложение и т. п.) с помощью классов, производных от ManagementObjectSearcher и ManagementQuery, а также осуществлять подписку на ряд управляющих событий с помощью класса ManagementEventWatcher. Доступные данные могут быть получены в

распределенной среде как от управляемых, так и от неуправляемых компонентов.

Существует 430 вариантов обращения за необходимой информацией. К примеру, если вам требуется узнать, каковы возможности вашего процессора, то необходимо извлечь информацию из **Win32_Processor**.

Получить доступ к информации можно путем создания объекта ManagementClass:

```
ManagementClass myManagementClass = new ManagementClass("Win32_Processor");
```

а затем доступ к свойствам для данного класса:

```
ManagementObjectCollection myManagementCollection =
myManagementClass.GetInstances();
PropertyDataCollection myProperties = myManagementClass.Properties;
```

Перебрать все возможные параметры можно с использованием следующих двух циклов:

```
foreach (var obj in myManagementCollection)
{
      foreach (var myProperty in myProperties)
      {
            // myProperty.Name – имя параметра
            // obj.Properties[myProperty.Name].Value - Значение параметра
      }
}
```

Пример работы с **Win32_Processor** представлен на рисунке 3.

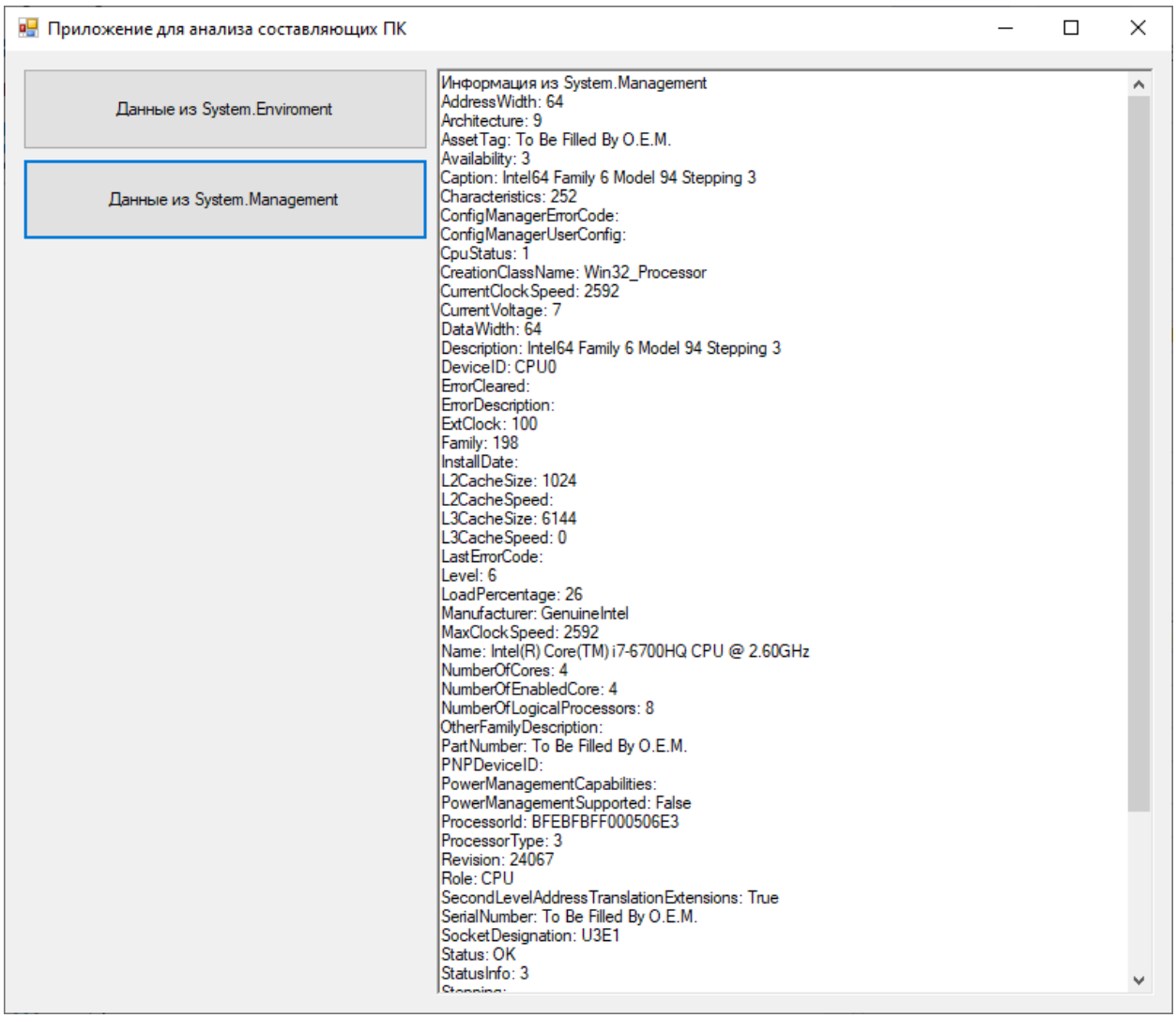


Рисунок 2 – работа с Win32_Processor

Более подробную информацию можно найти на сайте Microsoft https://docs.microsoft.com/ru-ru/windows/win32/cimwin32prov/computer-system-hardware-classes?redirectedfrom=MSDN#cooling-device-classes

3. Работа с PowerShell

Windows PowerShell позволяет системным администраторам автоматизировать большинство рутинных задач. С ее помощью можно менять настройки, останавливать и запускать сервисы, а также производить обслуживание большинства установленных приложений. Windows PowerShell — это в первую очередь командная оболочка с языком сценариев, изначально созданная на основе платформы .NET Framework, а позднее — на .NET Core.

Для работы с Windows PowerShell в Visual Studio необходимо установить пакет NuGet «Microsoft.PowerShell.5.1». Используемые пространства имен:

```
// Пространства имен для испольования PowerShell
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
```

Для доступа к функциональным возможностям Windows PowerShell предлагается объявить глобальную переменную типа Runspace для создания рабочей среды.

```
// Глобальные переменные
Runspace runspace;
```

Для того, чтобы открыть рабочую среду необходимо воспользоваться следующим кодом:

```
// Создаём рабочее пространство для PowerShell
runspace = RunspaceFactory.CreateRunspace();

// Открываем его
runspace.Open();
```

Перед завершением процесса необходимо рабочее пространство закрыть, например, в событии закрытия формы:

```
 // Закрываем рабочее пространство перед выходом
runspace.Close();
```

Пример работы получения рабочего каталога приведен ниже:

```
// Создаём конвейер (pipeline) для наших скриптов
Pipeline pipeline = runspace.CreatePipeline();
// Добавляем новую команду получения текущего рабочего каталога
pipeline.Commands.AddScript("Get-Location");

// Объявляем коллекцию для хранения полученного результата и записываем в него
результат команды
Collection<PSObject> results = pipeline.Invoke();
// Конвертируем коллекцию в единую строку для вывода
StringBuilder stringBuilder = new StringBuilder();
foreach (PSObject obj in results)
{
    stringBuilder.AppendLine(obj.ToString());
}
// Выводим результат
richTextBox1.Text = stringBuilder.ToString();
```

Практическая часть

1. Ознакомиться с теоретической частью.

2. Реализовать программу на языке C# с возможностью вывода списка процессов.

3. Реализовать вывод дополнительной информации при выборе процесса из списка.

Примечание: при работе с информацией о процессах используйте конструкцию try-catch для избегания ошибок отказа в доступе.

4. Реализовать возможность создания новых процессов и ввода аргументов запуска.

5. Реализовать программу на языке C# с возможностью вывода информации из System.Environment на форму в удобочитаемой форме согласно списка из теоретической части.

6. Реализовать программу на языке C# с возможностью вывода информации из пространства имен System.Management на форму в удобочитаемой форме с параметром Win32_Processor.

7. Реализовать программу на языке C# с возможностью вывода информации из пространства имен System.Management на форму в удобочитаемой форме с параметром согласно варианту из таблицы 1.

8. Реализовать программу на языке C# с возможностью вывода информации о рабочем каталоге и списке процессов с использованием пространства имен System.Management.Automation на форму в удобочитаемой форме.

9. Указать основные параметры и их назначение для задания 7.

Таблица 1 – варианты заданий

1	Информация о клавиатуре
2	Информация о мыши или аналогичном устройстве ввода
3	Информацию о физических дисках
4	Информацию о BIOS
5	Информацию о шине
6	Информацию о кеш-памяти
7	Информация об адресах памяти устройств в системе (Device Memory Address)
8	Информация об оперативной памяти
9	Информация о материнской плате

10	Информация о физической памяти
11	Информация о звуковой карте
12	Информация о USB-контроллере
13	Информация о сетевых адаптерах
14	Информация о мониторе
15	Информация о видеокарте