

Forum:[Hack Forums](#)

Thread: [List Steam Games + Sizes](#)

Bunnyslippers69 - July 25, 2019, 7:58 am

Not sure if anyone will find this useful, but i thought i'd at least give back to the community somewhat. It lists all steam games installed inside the default download location. If you have multiple game drives, this isn't going to list all of them.

I'd appreciate some constructive criticism\feedback, but just being negative isn't going to benefit anyone. Thanks.

Code using System;
using System.IO;
using Microsoft.Win32;

```
namespace Steam_Game_List
{
    class Program
    {
        static void Main(string[] args)
        {
            var path = Registry.GetValue(@"HKEY_LOCAL_MACHINE\SOFTWARE\Valve\Steam",
            "InstallPath", null);

            if (File.Exists(Convert.ToString(path + @"\steam.exe")))
            {
                var dirs = Directory.GetDirectories(Path.Combine(path + @"\steamapps\common\"));
                foreach (var dir in dirs)
                {
                    double dirSizeB = DirSize(new DirectoryInfo(dir));
                    double dirSizeKB = ScaleUp(dirSizeB);
                    double dirSizeMB = ScaleUp(dirSizeKB);
                    double dirSizeGB = ScaleUp(dirSizeMB);

                    string fileName = dir.Replace(Path.Combine(path + @"\steamapps\common\"), "");
                    ColoredConsoleWrite(ConsoleColor.White, fileName);
                    ColoredConsoleWrite(ConsoleColor.Yellow, "\n\tSizes{" + dirSizeMB + "MB\t");
                    ColoredConsoleWrite(ConsoleColor.Yellow, dirSizeGB + "GB}\n\n");
                }
            }
            else{Console.WriteLine("Couldn't find steam folder");}

            Console.Read();
        }

        public static long DirSize(DirectoryInfo d)
        {
            long size = 0;
            // Add file sizes.
```

```

        FileInfo[] fis = d.GetFiles();
        foreach (FileInfo fi in fis)
        {
            size += fi.Length;
        }
        // Add subdirectory sizes.
        DirectoryInfo[] dis = d.GetDirectories();
        foreach (DirectoryInfo di in dis)
        {
            size += DirSize(di);
        }
        return size;
    }

    public static void ColoredConsoleWrite(ConsoleColor color, string text)
    {
        ConsoleColor originalColor = Console.ForegroundColor;
        Console.ForegroundColor = color;
        Console.Write(text);
        Console.ForegroundColor = originalColor;
    }

    public static double ScaleUp(double defSize)
    {
        return Math.Round(defSize / 1024, 2);
    }
}

```

EDIT: I've remade this in a Windows Form. Pretty much the same idea, but using lists.
[i.imgur.com/sKSXb9g.png](https://imgur.com/sKSXb9g.png)

MavenGradle - January 7, 2020, 11:05 am

What you should have done is at least implemented proper error handling. OOP principles would be nice as well, but seems like a far fetched set of principles for some people (including you). What happens if you run this code on a computer that doesn't even have Steam installed in the first place? I'll tell you: It will fail, but not after some serious overhead..

Have you considered implementing the following check to reduce IO operations and trapped exceptions?

```
Code var path = Registry.GetValue(@"HKEY_LOCAL_MACHINE\SOFTWARE\Valve\Steam",
"InstallPath", null);
```

```
if (path == null || File.Exists(Convert.ToString(path + @"\steam.exe")))
```

What about exceptions that might get raised by running [Registry.GetValue](#)? Those will definitely crash your application without any second thoughts.

If anyone seriously considers adding this to their code base, you should also be aware that the value of InstallPath located in the registry path:

```
Code HKEY_LOCAL_MACHINE\SOFTWARE\Valve\Steam
```

Does not contain any library folders that people can use to install games in directories, other than the default one, using steam. So in addition to the code being absolute trash, it won't even work properly. Allow me to elaborate.

First of all, it appears like you never heard of the `libraryfolders.vdf`, `appmanifest_*.acf` and `config.vdi` files that are utilized by Steam. You, not utilizing said files is reason enough to realize that your implementation is very superficial and that you did not take your time to implement this properly.

```
Code public static long DirSize(DirectoryInfo d)
```

Why is this, and so many other methods in your code snippet public if you're not using OOP principles and are only calling this in your Program class? Do you know what access modifiers are? <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/access-modifiers>

In the same method, you do not check if the `DirectoryInfo d` variable is even null or set at all. This makes it extremely hard to reuse your code, but then again, you don't follow OOP principles either. You could have at least used a proper variable name instead of "d". Another fun thing is your lack of error handling, that could cause issues in regards to folder authorisations and your application could crash at any point..

Another example of a bad practice is the following piece of code:

```
Code public static void ColoredConsoleWrite(ConsoleColor color, string text)
{
    ConsoleColor originalColor = Console.ForegroundColor;
    Console.ForegroundColor = color;
    Console.Write(text);
    Console.ForegroundColor = originalColor;
}
```

You call the `ColoredConsoleWrite` method three times in your code. That's three times that the `originalColor` object has to be created and three times that it's set to the same value. Again, I would seriously suggest that you look into OOP standards.

```
Code public static double ScaleUp(double defSize)
{
    return Math.Round(defSize / 1024, 2);
}
```

The name `ScaleUp` is not indicative of what this method actually does. Once again, implement proper variable and method names. Again, why is this a public method?

Next up:

```
Code Path.Combine(path + @"\\steamapps\\common\\")
```

You use the `Path.Combine` method twice using the same variables. Why are you so keen on implementing overhead into your code? Would it be that awful to utilize a separate variable to store the return value of this method? Example:

```
Code string installPath = Path.Combine(path + @"\\steamapps\\common\\");
```

But hold up a second before you start using the example above;

```
Code Convert.ToString(path + @"\\steam.exe")
```

Here you're not even calling `Path.Combine`, you're just [converting a string to a string](#). The `Convert.ToString()` call is absolutely pointless because when you add a string to an object, the `ToString()` method of said object will be called, which will result in the install path of the steam executable plus `steam.exe`. The same thing actually applies to the single variable you're passing to the `Path.Combine()` method. What exactly would be the point of that, except for useless code overhead?

Overall, your code is not very well thought out. It requires serious changes and you should seriously rewrite this so that it complies to OOP standards. They're not hard to follow.

https://en.wikipedia.org/wiki/Object-oriented_programming

You do not utilize Steam's configuration files and as such you do not use a correct way to retrieve

games and until this is all fixed, I cannot recommend your code snippet to anyone.

Bunnyslippers69 - January 7, 2020, 1:48 pm

(Jan 7, 2020 12:05 PM)MavenGradle Wrote: [\[CITE\]](#) What you should have done is at least implemented proper error handling. OOP principles would be nice as well, but seems like a far fetched set of principles for some people (including you). What happens if you run this code on a computer that doesn't even have Steam installed in the first place? I'll tell you: It will fail, but not after some serious overhead..

Have you considered implementing the following check to reduce IO operations and trapped exceptions?

```
Code var path = Registry.GetValue(@"HKEY_LOCAL_MACHINE\SOFTWARE\Valve\Steam",  
"InstallPath", null);
```

```
if (path == null || File.Exists(Convert.ToString(path + @"\steam.exe")))
```

What about exceptions that might get raised by running [Registry.GetValue](#)? Those will definitely crash your application without any second thoughts.

If anyone seriously considers adding this to their code base, you should also be aware that the value of InstallPath located in the registry path:

```
Code HKEY_LOCAL_MACHINE\SOFTWARE\Valve\Steam
```

Does not contain any library folders that people can use to install games in directories, other than the default one, using steam. So in addition to the code being absolute trash, it won't even work properly. Allow me to elaborate.

First of all, it appears like you never heard of the libraryfolders.vdf, appmanifest_*.acf and config.vdi files that are utilized by Steam. You, not utilizing said files is reason enough to realize that your implementation is very superficial and that you did not take your time to implement this properly.

```
Code public static long DirSize(DirectoryInfo d)
```

Why is this, and so many other methods in your code snippet public if you're not using OOP principles and are only calling this in your Program class? Do you know what access modifiers are?

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/access-modifiers>

In the same method, you do not check if the DirectoryInfo d variable is even null or set at all. This makes it extremely hard to reuse your code, but then again, you don't follow OOP principles either. You could have at least used a proper variable name instead of "d". Another fun thing is your lack of error handling, that could cause issues in regards to folder authorisations and your application could crash at any point..

Another example of a bad practice is the following piece of code:

```
Code      public static void ColoredConsoleWrite(ConsoleColor color, string text)  
      {  
          ConsoleColor originalColor = Console.ForegroundColor;  
          Console.ForegroundColor = color;  
          Console.Write(text);  
          Console.ForegroundColor = originalColor;  
      }
```

You call the ColoredConsoleWrite method three times in your code. That's three times that the originalColor object has to be created and three times that it's set to the same value. Again, I would seriously suggest that you look into OOP standards.

```
Code public static double ScaleUp(double defSize)  
      {
```

```
    return Math.Round(defSize / 1024, 2);  
}
```

The name ScaleUp is not indicative of what this method actually does. Once again, implement proper variable and method names. Again, why is this a public method?

Next up:

```
Code Path.Combine(path + @"steamapps\common\");
```

You use the Path.Combine method twice using the same variables. Why are you so keen on implementing overhead into your code? Would it be that awful to utilize a separate variable to store the return value of this method? Example:

```
Code string installPath = Path.Combine(path + @"steamapps\common\");
```

But hold up a second before you start using the example above;

Code Convert.ToString(path + @"steam.exe") Here you're not even calling Path.Combine, you're just [converting a string to a string](#). The Convert.ToString() call is absolutely pointless because when you add a string to an object, the ToString() method of said object will be called, which will result in the install path of the steam executable plus steam.exe. The same thing actually applies to the single variable you're passing to the Path.Combine() method. What exactly would be the point of that, except for useless code overhead?

Overall, your code is not very well thought out. It requires serious changes and you should seriously rewrite this so that it complies to OOP standards. They're not hard to follow.

https://en.wikipedia.org/wiki/Object-oriented_programming

You do not utilize Steam's configuration files and as such you do not use a correct way to retrieve games and until this is all fixed, I cannot recommend your code snippet to anyone.

Thanks for the constructive criticism. I appreciate the time you spent responding to my thread. I'd completely forgotten about this project, and if i'm honest, was still fairly new to the language when i posted this. I'd like to think i've improved over the months though. I'll definitely go over your response at a later date and possibly make some changes, but i'm currently swamped by revision for my GCSEs so i've got no free time.

MavenGradle - January 7, 2020, 1:49 pm

(Jan 7, 2020 02:48 PM)Bunnyslippers69 Wrote: [\[CITE\]](#) Thanks for the constructive criticism. I appreciate the time you spent responding to my thread. I'd completely forgotten about this project, and if i'm honest, was still fairly new to the language when i posted this. I'd like to think i've improved over the months though. I'll definitely go over your response at a later date and possibly make some changes, but i'm currently swamped by revision for my GCSEs so i've got no free time.

Thanks for taking it like a champ. Good luck with your projects and feel free to post any code here as many members here will be happy to help you out ? .