

Forum:[Hack Forums](#)

Thread: [Generating numbers and letters in a formatted string.](#)

Pawnhub - October 24, 2019, 5:24 pm

Hello it's time to drop a thread about generating your own license keys for your own services and or generating a string to a specific format. I'll explain the process with comments in code.

```
Code    static void Main(string[] args)
        {
```

```
        string[] letters = { "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s",
        "t", "u", "v", "w", "x", "y", "z" }; //this is defining letters in an array I will be converting this ToUpper()
        for this example.
```

```
        Random r = new Random(); //instantiating the Random() class and assigning it's name r.
        This class is used for generating random numbers.
```

```
        // these lines below is making an implicit variable (var) that will be passed along the
        string array "letters" and using the random "Next" method identifier (r.Next) we can generate
        between certain numbers it'll be using for the array. If I wanted to find a random number between
        0-10, you'd do r.Next(0,10) but for this example were passing in the array of letters which max would
        be 26, so 0-25 would be used since arrays start at position 0, not 1.
```

```
        var _letter1 = (letters[r.Next(0, 25)]);
        var _letter2 = (letters[r.Next(0, 25)]);
        var _letter3 = (letters[r.Next(0, 25)]);
        var _letter4 = (letters[r.Next(0, 25)]);
        var _letter5 = (letters[r.Next(0, 25)]);
        var _letter6 = (letters [r.Next(0, 25)]);
        var _1 = r.Next(0, 9);
        var _2 = r.Next(0, 9);
        var _3 = r.Next(0, 9);
        string dash = "-";
```

```
        //if we wanted to create the format: ABC-123-ABC and inserting random values to this
        format we'd do.
```

```
        var result = (_letter1 + _letter2 + _letter3 + dash + _1 + _2 + _3 + dash + _letter4 +
        _letter5 + _letter6).ToString().ToUpper();
```

```
        //you could also do it without the dash variable and just use "-" and I'm also converting
        to a string and invoking the ToUpper() method for all capital letters.
```

```
        Console.WriteLine(result);
        Console.ReadLine();
```

```
    }
```

A good way to use this is to put it into a loop and it'll generate different values each time.

MavenGradle - October 24, 2019, 5:27 pm

Use a char array instead of a string array and make all characters uppercase. That way you can remove the ToUpper call.

Other than that, solid contribution.

Pawnhub - October 24, 2019, 5:32 pm

(Oct 24, 2019 05:27 PM)MavenGradle Wrote: [\[CITE\]](#) Use a char array instead of a string array and make all characters uppercase. That way you can remove the ToUpper call.

Other than that, solid contribution.

Yes I missed that, I'll change the array type thank you. And I'm leaving in the conversion since I explained it's part of the process in my example. :)

MavenGradle - October 24, 2019, 5:34 pm

(Oct 24, 2019 05:32 PM)UnknownPawn Wrote: [\[CITE\]](#) Yes I missed that, I'll change the array type thank you. And I'm leaving in the conversion since I explained it's part of the process in my example. :)

You can completely get rid of the integer array since the random generator can generate numbers for you:

Code `r.Next(10);`

Pawnhub - October 24, 2019, 5:37 pm

(Oct 24, 2019 05:34 PM)MavenGradle Wrote: [\[CITE\]](#) You can completely get rid of the integer array since the random generator can generate numbers for you:

Code `r.Next(10);`

Again thank you maven, I have fixed that as well. :)

(Oct 24, 2019 05:27 PM)MavenGradle Wrote: [\[CITE\]](#) Use a char array instead of a string array and make all characters uppercase. That way you can remove the ToUpper call.

Other than that, solid contribution.

Update on the char array, you might want to test the code but it's not outputting correctly. Do you know why? Until then it's a string array.

0xDEAD10CC - October 24, 2019, 7:42 pm

I wouldn't call this a solid contribution, it doesn't even work as designed.

Random Next() -

https://docs.microsoft.com/en-us/dotnet/api/system.random.next?view=netframework-4.8#System.Random.Next_System_Int32_System_Int32

Quote:Returns

Int32

A 32-bit signed integer greater than or equal to min**Value** and less than max**Value**; that is, the range of return values includes min**Value** but not max**Value**. If min**Value** equals max**Value**, min**Value** is returned.

The code doesn't follow standard naming conventions, and for a fixed length output you could have used a char array of the specified length to convert to a string rather than concatenating variables. The variable use demonstrates that you haven't understood how to apply the concept of arrays where it makes sense to do so. You also have a char array defined using string values as elements.

The ToString() call is also redundant: In order to concatenate an int and a string, an implicit

ToString() is called behind the scenes so the value you're calling ToString() on explicitly is already a string...

You don't really need any array for the characters from A-Z if you took the time to understand the ASCII table.

Pawnhub - October 24, 2019, 8:05 pm

(Oct 24, 2019 07:42 PM)0xDEAD10CC Wrote: [\[CITE\]](#) I wouldn't call this a solid contribution, it doesn't even work as designed.

Random Next() - [Random.Next Method \(System\) | Microsoft Docs](#)

The code doesn't follow standard naming conventions, and for a fixed length output you could have used a char array of the specified length to convert to a string rather than concatenating variables. The variable use demonstrates that you haven't understood how to apply the concept of arrays where it makes sense to do so. You also have a char array defined using string values as elements.

The ToString() call is also redundant: In order to concatenate an int and a string, an implicit ToString() is called behind the scenes so the value you're calling ToString() on explicitly is already a string...

You don't really need any array for the characters from A-Z if you took the time to understand the ASCII table.

I see where I went wrong thanks. Can't update thread since I've updated this alot.

0xDEAD10CC - October 24, 2019, 8:19 pm

Code static void Main(string[] args)

```
{
    var rnd = new Random();
    var arr = new char[10]; // probably don't care about the magic numbers
                          // that follow for such a small program.

    arr[6] = '-';
    for (int i = 0; i < 6; i++) arr[i] = 'A' + r.Next(26);
    for (int i = 7; i < 10; i++) arr[i] = '0' + r.Next(10);

    var result = new string(arr);

    Console.WriteLine(result);
    Console.ReadLine();
}
```

Pawnhub - October 24, 2019, 8:25 pm

(Oct 24, 2019 08:19 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Code static void Main(string[] args)

```
{
    var rnd = new Random();
    var arr = new char[10]; // probably don't care about the magic numbers
                          // that follow for such a small program.

    arr[6] = '-';
    for (int i = 0; i < 6; i++) arr[i] = 'A' + r.Next(26);
    for (int i = 7; i < 10; i++) arr[i] = '0' + r.Next(10);
```

```
var result = new string(arr);

Console.WriteLine(result);
Console.ReadLine();
}Yeah I can see how this would be better.
```

Question if using random(0,9) normally to print a integer it won't print 9 from the quote you posted earlier?

But if it's enumerating through an array for example this array like so
Code (letters[r.Next(0, 26)]); I still get Z as an output why does this happen if it don't count the max value or am I misunderstanding something? It also doesn't call an index exception. very confused

0xDEAD10CC - October 24, 2019, 8:36 pm

(Oct 24, 2019 08:25 PM)UnknownPawn Wrote: [\[CITE\]](#) Yeah I can see how this would be better.

Question if using random(0,9) normally to print a integer it won't print 9 from the quote you posted earlier?

But if it's enumerating through an array for example this array like so
Code (letters[r.Next(0, 26)]); I still get Z as an output why does this happen if it don't count the max value or am I misunderstanding something? It also doesn't call an index exception. very confused
Read it again... Max is exclusive, so how would it return 9?

Your original code used (0, 9) and (0, 25) which means 9 and 25 won't be returned.

You get Z using (0, 26) because 0..25 inclusive is 26 values...

Pawnhub - October 24, 2019, 8:39 pm

(Oct 24, 2019 08:36 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Read it again... Max is exclusive, so how would it return 9?

I'm talking about the string array above. I see when doing a normal Random.Next(0,9) will return max 8.

But the alphabet array I made is 26 chars long, when enumerating through the array, random still uses the last value (26 = Z) in (letters[r.Next(0, 26)]);

(Oct 24, 2019 08:36 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You get Z using (0, 26) because 0..25 inclusive is 26 values...

Oh alright this is starting to make sense I think?

0xDEAD10CC - October 24, 2019, 8:42 pm

(Oct 24, 2019 08:39 PM)UnknownPawn Wrote: [\[CITE\]](#) I'm talking about the string array above. I see when doing a normal Random.Next(0,9) will return max 8.

But the alphabet array I made is 26 chars long, when enumerating through the array, random still uses the last value (26 = Z) in (letters[r.Next(0, 26)]);

You clearly don't understand array indexing... [0] = A, [25] = Z (not index [26])

Next(0, 26) will return values from 0 through 25. Index [25] is the 26th value because index [0] is the first element, not [1].

Pawnhub - October 24, 2019, 8:43 pm

(Oct 24, 2019 08:42 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You clearly don't understand array indexing... [0] = A, [25] = Z (not index [26])

Next(0, 26) will return values from 0 through 25.
??? I get it now. I see what happened.

0xDEAD10CC - October 24, 2019, 8:45 pm

I'm also calling the overload with one parameter:

Code [Random].Next(Int32)

But the following two calls are functionally equivalent:

Code rnd.Next(0, 26);

Code rnd.Next(26);

MavenGradle - October 25, 2019, 9:45 am

(Oct 24, 2019 08:19 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Code static void Main(string[] args)

```
{  
    var rnd = new Random();  
    var arr = new char[10]; // probably don't care about the magic numbers  
                           // that follow for such a small program.
```

```
    arr[6] = '-';  
    for (int i = 0; i < 6; i++) arr[i] = 'A' + r.Next(26);  
    for (int i = 7; i < 10; i++) arr[i] = '0' + r.Next(10);
```

```
    var result = new string(arr);
```

```
    Console.WriteLine(result);  
    Console.ReadLine();
```

```
}
```

For all intents and purposes you forgot the cast and used a wrong random variable:

Code for (int i = 0; i < 6; i++) arr[i] = (char) ('A' + rnd.Next(26));

for (int i = 7; i < 10; i++) arr[i] = (char) ('0' + rnd.Next(10));

This is how it could look like in relation to the format OP used:

```
Code    Random rnd = new Random();  
        char[] arr = new char[11];  
        const char dash = '-';  
        arr[3] = dash;  
        arr[7] = dash;  
        for (int i = 0; i < 3; i++) arr[i] = (char) ('A' + rnd.Next(26));  
        for (int i = 4; i < 7; i++) arr[i] = (char) ('0' + rnd.Next(10));  
        for (int i = 8; i < 11; i++) arr[i] = (char) ('A' + rnd.Next(26));  
        string result = new string(arr);
```

Or

```
Code    Random rnd = new Random();  
        char[] arr =  
        {  
            (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26)), '-',  
            (char) ('0' + rnd.Next(10)), (char) ('0' + rnd.Next(10)), (char) ('0' + rnd.Next(10)), '-',  
            (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26))  
        };  
        string result = new string(arr);
```

if you prefer not to loop. This is harder to maintain in case format changes though.

If you prefer interpolation (degrades performance significantly):

```
Code    private static readonly Random Random = new Random();
```

```
static void Main(string[] args)
{
```

```
    Console.WriteLine($"{GetRandomChar(){GetRandomChar(){GetRandomChar()}-{GetRandomNumber(){GetRandomNumber(){GetRandomNumber()}-{GetRandomChar(){GetRandomChar(){GetRandomChar()}}");
```

```
    }
    private static char GetRandomChar() => (char) ('A' + Random.Next(26));
    private static char GetRandomNumber() => (char) ('0' + Random.Next(10));^ All of the above
```

should be modified to adhere to OOP standards.

0xDEAD10CC - October 25, 2019, 10:49 am

(Oct 25, 2019 09:45 AM)MavenGradle Wrote: [\[CITE\]](#) For all intents and purposes you forgot the cast and used a wrong random variable:

```
Code for (int i = 0; i < 6; i++) arr[i] = (char) ('A' + rnd.Next(26));
```

```
for (int i = 7; i < 10; i++) arr[i] = (char) ('0' + rnd.Next(10));
```

This is how it could look like in relation to the format OP used:

```
Code      Random rnd = new Random();
          char[] arr = new char[11];
          const char dash = '-';
          arr[3] = dash;
          arr[7] = dash;
          for (int i = 0; i < 3; i++) arr[i] = (char) ('A' + rnd.Next(26));
          for (int i = 4; i < 7; i++) arr[i] = (char) ('0' + rnd.Next(10));
          for (int i = 8; i < 11; i++) arr[i] = (char) ('A' + rnd.Next(26));
          string result = new string(arr);
```

Or

```
Code      Random rnd = new Random();
          char[] arr =
          {
              (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26)), '-',
              (char) ('0' + rnd.Next(10)), (char) ('0' + rnd.Next(10)), (char) ('0' + rnd.Next(10)), '-',
              (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26)), (char) ('A' + rnd.Next(26))
          };
          string result = new string(arr);if you prefer not to loop. This is harder to maintain in case
```

format changes though.

If you prefer interpolation:

```
Code      private static readonly Random Random = new Random();
          static void Main(string[] args)
          {
```

```
    Console.WriteLine($"{GetRandomChar(){GetRandomChar(){GetRandomChar()}-{GetRandomNumber(){GetRandomNumber(){GetRandomNumber()}-{GetRandomChar(){GetRandomChar(){GetRandomChar()}}");
```

```
    }
    private static char GetRandomChar() => (char) ('A' + Random.Next(26));
    private static char GetRandomNumber() => (char) ('0' + Random.Next(10));^ All of the above
```

should be modified to adhere to OOP standards.

I wrote it in a text editor quick but the idea is clear.

MavenGradle - October 25, 2019, 3:38 pm

For those of you who are curious (unique permutations for the format of

[A-Z][A-Z][A-Z]-[0-9][0-9][0-9]-[A-Z][A-Z][A-Z]):

26^6 possibilities for the alphabetic characters alone since you have 6 alphabetical characters and 26 possibilities for each of them.

if you add one number, you add 10 possibilities for each of those values so it becomes 26^6 10

If you add a second number, you add an extra 10 possibilities for those values so you have (26^6 10) 10

If you add another number you add an extra 10 possibilities for those values so you have ((26^6 10) 10) 10 possibilities.

This amounts to the total of 308.915.776.000? unique possibilities. Thanks for the fun [Pawnhub](#) !

Humor me and prove me wrong will you ([A-Z][A-Z]-[0-9][0-9]:

```
Code    static void Main(string[] args)
    {
        Random rnd = new Random();
        List<string> unique = new List<string>();
        const char dash = '-';
        for (int y = 0; y < 67601; y++)
        {
            Console.WriteLine("Current run: " + y);
            bool isUnique = false;
            while (!isUnique)
            {
                char[] arr = new char[5];
                arr[2] = dash;
                for (int i = 0; i < 2; i++) arr[i] = (char) ('A' + rnd.Next(26));
                for (int i = 3; i < 5; i++) arr[i] = (char) ('0' + rnd.Next(10));
                string result = new string(arr);

                bool found = unique.Any(s => s == result);
                if (found) continue;
                unique.Add(result);
                isUnique = true;
            }
        }
        Console.WriteLine("MavenGradle is an idiot. Obviously the total count is: " + unique.Count);
    }
```

TyrantKingBen - October 25, 2019, 4:25 pm

It's been a while (years) since I've done anything in C#. If you wanted more control over formatting you could do something like below.

```
Code using System;
using System.Text;
// using System.Linq;
```

```
public class Program {
    public static void Main() {
        Serial serial = new Serial().Add(Serial.Letter(5))
                                   .Add('-')
                                   .Add(Serial.Digit(5))
                                   .Add('-')
```

```

        .Add(Serial.Alphanumeric(5))
        .Add('-')
        .Add(Serial.Letter())
        .Add(Serial.Digit())
        .Add(Serial.Letter())
        .Add(Serial.Digit())
        .Add(Serial.Letter());
    Console.WriteLine(serial.ToString());
}
}

class Serial {
    private static Random random = new Random();
    private StringBuilder stringBuilder = new StringBuilder();

    public static char Letter() {
        return (char)('A' + random.Next(26));
    }

    public static char[] Letter(int count) {
        // return Enumerable.Range(0, count).Select(x => Letter()).ToArray();
        char[] generated = new char[count];
        for (int i = 0; i < count; i++) generated[i] = Letter();
        return generated;
    }

    public static char Digit() {
        return (char)('0' + random.Next(10));
    }

    public static char[] Digit(int count) {
        // return Enumerable.Range(0, count).Select(x => Digit()).ToArray();
        char[] generated = new char[count];
        for (int i = 0; i < count; i++) generated[i] = Digit();
        return generated;
    }

    public static char Alphanumeric() {
        int randomInt = random.Next(36);
        return (char)('0' + (randomInt > 9 ? randomInt + 7 : randomInt));
    }

    public static char[] Alphanumeric(int count) {
        // return Enumerable.Range(0, count).Select(x => Alphanumeric()).ToArray();
        char[] generated = new char[count];
        for (int i = 0; i < count; i++) generated[i] = Alphanumeric();
        return generated;
    }

    public Serial Add(char c) {
        stringBuilder.Append(c);
        return this;
    }
}

```

```

public Serial Add(char[] c) {
    stringBuilder.Append(c);
    return this;
}

public override string ToString() {
    return stringBuilder.ToString();
}

public Serial() {
}
}

```

Pawnhub - October 26, 2019, 5:24 am

(Oct 25, 2019 03:38 PM)MavenGradle Wrote: [\[CITE\]](#) For those of you who are curious (unique permutations for the format of [A-Z][A-Z][A-Z]-[0-9][0-9][0-9]-[A-Z][A-Z][A-Z]):

26^6 possibilities for the alphabetic characters alone since you have 6 alphabetical characters and 26 possibilities for each of them.

if you add one number, you add 10 possibilities for each of those values so it becomes 26^6 10

If you add a second number, you add an extra 10 possibilities for those values so you have (26^6 10) 10

If you add another number you add an extra 10 possibilities for those values so you have ((26^6 10) 10) 10 possibilities.

This amounts to the total of 308.915.776.000? unique possibilities. Thanks for the fun [Pawnhub](#) !

Humor me and prove me wrong will you ([A-Z][A-Z]-[0-9][0-9]:

Maven I'm glad you figured out the math to this. Yesterday was a brutal math session with you!

Glad you made the math step by step it helps a lot.

Mr.36k permutations ;)

MavenGradle - October 26, 2019, 9:29 am

(Oct 26, 2019 05:24 AM)UnknownPawn Wrote: [\[CITE\]](#) Maven I'm glad you figured out the math to this. Yesterday was a brutal math session with you!

Glad you made the math step by step it helps a lot.

Mr.36k permutations ;)

Well you're the one who pointed out the flaw in the initial calculation!

You can calculate it the other way around as well if you like:

3 numbers, each having 10 possibilities: 10^3 unique possibilities

Add a character and you now have 26 extra possibilities for each of those: 10^3 26 unique possibilities

Add a character and you now have 26 extra possibilities for each of those: (10^3 26) * 26 unique possibilities

And so on and so on until you get to 1000×26×26×26×26×26×26 unique possibilities which will hold the same value as calculated above.

You can simplify this by writing: $10^3 \times 26^6 = 26^6 \times 10^3$

SIM-4440595 - November 16, 2019, 4:19 pm

Thanks for the tutorial, this is very helpful.