

Forum: [Hack Forums](#)

Thread: [A Challenge for the bored 2](#)

saltycracker - April 19, 2020, 5:03 am

Write a function that will produce the numbers 1 - 10 randomly and not return the same value twice.

Example:

Say I call my function getNum

The first time I call getNum it may produce:

7

The second time I call getNum it may produce:

3

The third time I call getNum it may produce:

10

The fourth time I call getNum it may produce:

8

The fifth time I call getNum it may produce:

2

The sixth time I call getNum it may produce:

1

The seventh time I call getNum it may produce:

4

The eighth time I call getNum it may produce:

5

The ninth time I call getNum it may produce:

9

The tenth time I call getNum it may produce:

6

Good luck... C++'ers, try to say away from container classes.

Note: What happens at the end of the cycle is up to you. You can reset the function to start again or just do nothing.

FoxBlood - April 19, 2020, 5:28 am

Am I allowed to use rand() function?

saltycracker - April 19, 2020, 5:36 am

(Apr 19, 2020 05:28 AM)FoxBlood Wrote: [\[CITE\]](#) Am I allowed to use rand() function?
Yes and srand too.

I just want people to think about a way to solve this without relying on a container class.

FoxBlood - April 19, 2020, 6:08 am

```
Code int arr[11];
int random_numer()
{
    int number = rand() % 10 + 1;
    if(arr[number] == 1)
        return random_numer();
    else
    {
        arr[number] = 1;
        return number;
    }
}Something like this?
```

HFDone - April 19, 2020, 6:13 am

very simple
with C you can use array for the rand values
with C++ you can use std::array

saltycracker - April 19, 2020, 6:21 am

Very important... You must return the values 1 through 10 randomly and no repeats.

FoxBlood - April 19, 2020, 6:30 am

```
Code void random_numers()
{
    srand (time(NULL));
    int arr[11] = {0};
    for(int i = 1; i<=10;i++)
    {
        int number = rand() % 10 + 1;
        if(arr[number] == 1)
            i--;
        else
        {
            cout<<number<<" ";
            arr[number] = 1;
        }
    }
}My best code I ever write. Isn't it?
```

saltycracker - April 19, 2020, 6:51 am

(Apr 19, 2020 06:30 AM)FoxBlood Wrote: [\[CITE\]](#) Code void random_numers()

```
{
    srand (time(NULL));
    int arr[11] = {0};
    for(int i = 1; i<=10;i++)
    {
        int number = rand() % 10 + 1;
```

```

    if(arr[number] == 1)
        i--;
    else
    {
        cout<<number<<" ";
        arr[number] = 1;
    }
}

```

}My best code I ever write. Isn't it?Wow! I mean Wow! Why would you write it like that? For lol's?

akali - April 19, 2020, 6:55 am

```

Code int getNum(int* array,int index){
    int r = 0;
    do{
        r = rand() % 11 ;
        for(int i = 0; i < index;++i){

            if(r == array[i]){
                r = 0;
            }
        }
    }while(r == 0);
    return r;
}

int main(){
    int array[10] = {0};
    for(int i = 0; i < 10; ++i){
        array[i] = getNum(array,i);
        printf("%i\n",array[i]);
    }
}

```

saltycracker - April 19, 2020, 7:47 am

(Apr 19, 2020 06:55 AM)akali Wrote: [\[CITE\]](#) Code int getNum(int* array,int index){

```

    int r = 0;
    do{
        r = rand() % 11 ;
        for(int i = 0; i < index;++i){

            if(r == array[i]){
                r = 0;
            }
        }
    }while(r == 0);
    return r;
}

int main(){
    int array[10] = {0};
    for(int i = 0; i < 10; ++i){
        array[i] = getNum(array,i);
        printf("%i\n",array[i]);
    }
}

```

So far... You are making it way more difficult than it has to be. You don't need a loop to solve the problem.

HFDone - April 19, 2020, 11:06 am

PHP Code

```
<?php
int getNum(int* array,int index){
    int r = 0;
    r = rand() % 10 + 1;
    for(int i = 0; i < index; ++i){
        if(r == array[i]){
            r = getNum(array, i + 1);
        }
    }
    return r;
}

int main(){
    int array[10] = {0};
    for(int i = 0; i < 10; ++i){
        array[i] = getNum(array, i);
        printf("%i\n", array[i]);
    }
}
```

saltycracker - April 19, 2020, 1:14 pm

(Apr 19, 2020 11:06 AM)HFDone Wrote: [\[CITE\]](#) PHP Code

```
<?php
int getNum(int* array,int index){
    int r = 0;
    r = rand() % 10 + 1;
    for(int i = 0; i < index; ++i){
        if(r == array[i]){
            r = getNum(array, i + 1);
        }
    }
    return r;
}

int main(){
    int array[10] = {0};
    for(int i = 0; i < 10; ++i){
        array[i] = getNum(array, i);
        printf("%i\n", array[i]);
    }
}
```

What is it with the loops? You don't need a loop or recursion to solve this problem.

severos - April 19, 2020, 9:52 pm

PHP Code

```
<?php
```

```
int getNum(){
    static int[] arr = {8,5,1,9,7,6,3,4,2,10}
    static idx = 0;
    if(idx > 9)
        return 0; // already done all numbers
    return arr[idx++];
}
```

Is this considered random enough?

saltycracker - April 19, 2020, 10:06 pm

Here's what I did..

Spoiler ([Click to View](#))

Code #include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARR_SIZE 10

```
int getNum() {
    static size_t size = 10;
    static int arr[ARR_SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int pos = rand() % size;
    int ans = arr[pos];
    int temp = arr[size - 1];
    arr[size - 1] = ans;
    arr[pos] = temp;
    size--; //can reset size here when it reaches 0
    return ans;
}
```

```
int main(int argc, char ** argv) {
    srand((unsigned) time(NULL));
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    return 0;
}
```

0xDEAD10CC - April 21, 2020, 8:02 pm

(Apr 19, 2020 10:06 PM)saltycracker Wrote: ([CITE](#)) Here's what I did..

Spoiler ([Click to View](#))

Code #include <stdio.h>

#include <stdlib.h>

```

#include <time.h>

#define ARR_SIZE 10

int getNum() {
    static size_t size = 10;
    static int arr[ARR_SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int pos = rand() % size;
    int ans = arr[pos];
    int temp = arr[size - 1];
    arr[size - 1] = ans;
    arr[pos] = temp;
    size--; //can reset size here when it reaches 0
    return ans;
}

int main(int argc, char ** argv) {
    srand((unsigned) time(NULL));
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    fprintf(stdout, "%d\n", getNum());
    return 0;
}

```

Why define a constant ARR_SIZE then not use it to assign your size variable? That makes it nearly useless, you could have just omitted the constant and all usages of it to get the same effect. You also have an extra variable in there that you don't need:

saltycracker - April 22, 2020, 12:48 am

(Apr 21, 2020 08:02 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Why define a constant ARR_SIZE then not use it to assign your size variable? That makes it nearly useless, you could have just omitted the constant and all usages of it to get the same effect. You also have an extra variable in there that you don't need:

And your opinion on the other submitted answers?

0xDEAD10CC - April 23, 2020, 11:46 am

There's no opinion to have on the other answers because they didn't solve the intended problem. The solution you provided runs in linear $O(N)$ time, and essentially $O(N)$ space, which is sufficient. I'm sure someone can come up with a way to reduce memory footprint for an array size of only 10 elements (1..10) but that would be splitting hairs.

SIM-3991524 - April 28, 2020, 8:56 pm

lfsr algorithm instead of stdlib rand :)

code

Spoiler [\(Click to View\)](#)

Code #include <stdio.h>

```
int seed = 0x12345678;
```

```
int memory = 0;
```

```
int rand() {  
    seed <=& 1;  
    seed += ((seed >> 19) % 2) ^ ((seed > 7) % 2);  
    return seed;  
}
```

```
int getNum(){  
    int n = (rand() & 0x7fffffff) % 10;  
    if ((memory >> n) % 2 == 0) {  
        memory |= 1 << n;  
        return n;  
    }  
    return getNum();  
}
```

```
int main() {  
    for (int i = 0; i < 10; i++) {  
        printf("%d %d\n", i + 1, getNum());  
    }  
}
```

output

Spoiler [\(Click to View\)](#)

Code > ./run.bat

```
1 2  
2 5  
3 0  
4 4  
5 9  
6 8  
7 7  
8 6  
9 3  
10 1
```

(Apr 23, 2020 11:46 AM)0xDEAD10CC Wrote: [\[CITE\]](#) There's no opinion to have on the other answers because they didn't solve the intended problem. The solution you provided runs in linear $O(N)$ time, and essentially $O(N)$ space, which is sufficient. I'm sure someone can come up with a way to reduce memory footprint for an array size of only 10 elements (1..10) but that would be splitting hairs.

Hey 0xDEAD10CC! Was able to store whether elements 1->10 had been returned in a single int :)

0xDEAD10CC - April 30, 2020, 9:06 pm

(Apr 28, 2020 08:56 PM)SIM-3991524 Wrote: [\[CITE\]](#) Ifsr algorithm instead of stdlib rand :)

code

Spoiler [\(Click to View\)](#)

Code #include <stdio.h>

```

int seed = 0x12345678;
int memory = 0;

int rand() {
    seed <<= 1;
    seed += ((seed >> 19) % 2) ^ ((seed > 7) % 2);
    return seed;
}

int getNum(){
    int n = (rand() & 0x7fffffff) % 10;
    if ((memory >> n) % 2 == 0) {
        memory |= 1 << n;
        return n;
    }
    return getNum();
}

int main() {
    for (int i = 0; i < 10; i++) {
        printf("%d %d\n", i + 1, getNum());
    }
}

```

output

Spoiler [\(Click to View\)](#)

Code > ./run.bat

```

1 2
2 5
3 0
4 4
5 9
6 8
7 7
8 6
9 3
10 1

```

Hey 0xDEAD10CC! Was able to store whether elements 1->10 had been returned in a single int :)
 People use bitmasks and lookup tables for all kinds of things.

SIM-3991524 - April 30, 2020, 9:12 pm

(Apr 30, 2020 09:06 PM)0xDEAD10CC Wrote: [\[CITE\]](#) People use bitmasks and lookup tables for all kinds of things.

Could you be my dad?