

Computer Programming

C Programming Assignment #2, 2005

Brief:

You are to write a game program that is centered around the guessing of numbers.

Detail:

Your computer program should begin by making up a random number (more on this later). It should then ask the user to guess the number. Any guess from the user will have one of three states:

- Too low
- Correct
- Too high

Your program should then inform the user which state his/her guess falls in to. Use a loop to repeat this process until the user gets the right number. Since the number guessing range is from 1-1000 the user may get bored and wish to exit the game. If they 'guess' -1 then the program should terminate, printing out how many guesses they made and what the number was.

If they guess correctly, the program should display a message to that effect, the number of guesses that were made and then exit.

Further below shows a sample of how the program might look. You are not required to produce identical output but attractiveness on screen gains more marks, as does the use of comments within the program to explain what individual pieces of code do.

Due Date:

200500408, 15:15

Submission Mechanism:

Paper & online at the program submission link at <http://webmail.cti-clonmel.ie>

Random Numbers & Computers

This sample shows how poor computers naturally are at choosing random numbers. It seems only humans and the weather are truly random – and even we can be predicted. This program shows the same 'random' number being generated each time by the 'rand' function provided by C.

```
-bash-2.05b$ gcc a2.c
-bash-2.05b$ ./a.out
1804289383
-bash-2.05b$ ./a.out
1804289383
-bash-2.05b$ ./a.out
1804289383
-bash-2.05b$ ./a.out
1804289383
-bash-2.05b$ ./a.out
1804289383
```

The source code of this program is here:

```
#include <stdio.h>
main ()
{
  int x;
  x = rand (1000);
  printf ("%d\n", x);
}
```

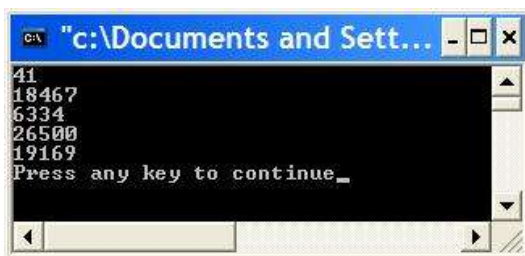
This program utilizes the **rand** function to generate its 'random' function. As you can see the number is not random – it's the same every time. That's because the random number generator has to be 'seeded' before use.

This program tries to generate 5 'random numbers:

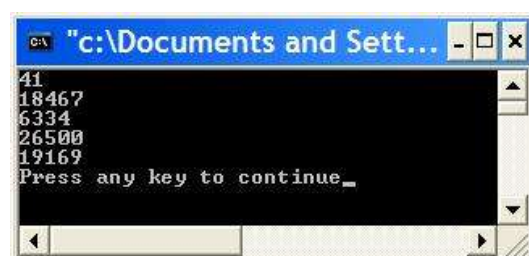
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int i;
  for (i = 0; i < 5; i++)
  {
    printf ("%d\n", rand());
  }
}
```

But each time produces:



Run 1



Run 2

'Seeding' The Random number generator:

If the program is modified to seed the random number generator with a value from the computer clock a near-random number can be generated. This line achieves this seeding process:

```
srand( (unsigned)time( NULL ) );
```

So that this program...

```
#include <stdio.h>
#include <time.h>
main ()
{
    int x;
    srand( (unsigned)time( NULL ) );
    x = rand ();
    printf ("%d\n", x);
}
```

yields this output:

```
-bash-2.05b$ gcc a2.c
-bash-2.05b$ ./a.out
404737816
-bash-2.05b$ ./a.out
1175092725
-bash-2.05b$ ./a.out
1947658440
-bash-2.05b$ ./a.out
262886088
-bash-2.05b$ ./a.out
2116293136
-bash-2.05b$ ./a.out
736575626
-bash-2.05b$ ./a.out
123708310
-bash-2.05b$
```

At this point it is clear that another mechanism might be beneficial, especially since none of the numbers shown above are below 1000. The numbers are random, just not in the range 1-1000.

This is achieved by using the modulus operator (%). If we say `x = rand()%1000`; we start to get more acceptable values.

So this program:

```
#include <stdio.h>
#include <time.h>
main ()
{
    int x;
    srand( (unsigned)time( NULL ) );
    x = rand ()%1000;
    printf ("%d\n", x);
}
```

gives output in a more acceptable range. As can be seen from the next output.

```
-bash-2.05b$ ./a.out
99
-bash-2.05b$ ./a.out
3
-bash-2.05b$ ./a.out
720
-bash-2.05b$ ./a.out
635
-bash-2.05b$ ./a.out
984
-bash-2.05b$ ./a.out
718
-bash-2.05b$ ./a.out
130
-bash-2.05b$ ./a.out
416
-bash-2.05b$ ./a.out
743
-bash-2.05b$ ./a.out
805
-bash-2.05b$ ./a.out
397
```

Below is a sample run of the program operating correctly.

Sample Run:

```
I have chosen a number in the range 1 to 1000. See can you guess it.
Your guess?
254
Too Low. Try Again.
Your Guess?
567
Too High. Try Again.
Your guess?
512
Too Low. Try Again.
Your Guess?
550
Correct!!! In only 4 guesses.
Play again? Y
I have chosen a number in the range 1 to 1000. See can you guess it.
Your guess?
700
Too High. Try Again.
Your guess?
-1
You have given up after 1 guess. The number was 299. Goodbye.
```