2.19 Random numbers

Generating a random number

Some programs need to use a random number. Ex: A game program may need to roll dice, or a website program may generate a random initial password.

The ***rand()*** function, in the C standard library, returns a random integer each time the function is called, in the range 0 to RAND_MAX.

Figure 2.19.1: Outputting three random integers.

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
   cout << rand() << endl;
   cout << rand() << endl;
   cout << rand() << endl;

   cout << "(RAND_MAX: " << RAND_MAX << ")" << endl;

   return 0;
}
```

```
16807
282475249
1622650073
(RAND_MAX: 2147483647)
```

Feedback?

Line 2 includes the C standard library, which defines the rand() function and RAND_MAX.

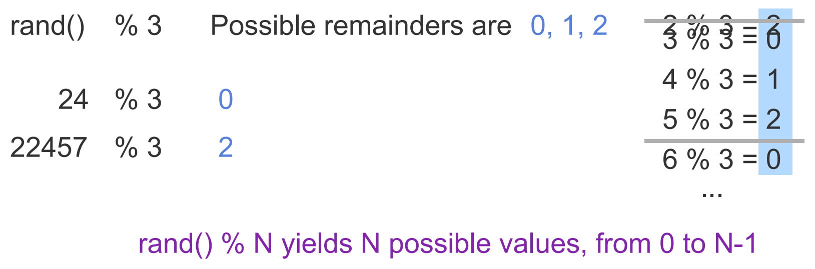
RAND_MAX is a machine-dependent value, but is at least 32,767. Above, RAND_MAX is about 2 billion.

Usually, a programmer wants a random integer restricted to a specific number of possible values. The modulo operator % can be used. Ex: `integer % 10` has 10 possible remainders: 0, 1, 2, ..., 8, 9.

PARTICIPATION ACTIVITY 2.19.1: Restricting random integers to a specific number of possible values.

1 2 3 2x speed

0 % 3 = 0
1 % 3 = 1



```
rand()  % 3   Possible remainders are 0, 1, 2     ~~2 % 3 = 2~~
                                                   3 % 3 = 0
    24  % 3   0                                    4 % 3 = 1
                                                   5 % 3 = 2
 22457  % 3   2                                    6 % 3 = 0
                                                      ...
```

Thus, rand() % 3 yields 3 possible values: 0, 1, and 2.
Generally, rand() % N yields N possible values, from 0 to N-1.

Feedback?

PARTICIPATION ACTIVITY 2.19.2: Random number basics.

1) What library must be included to use the rand() function?

- The C random numbers library
- [✓ The C standard library]

Correct

The C standard library contains useful functions for various purposes, like generating random integers, converting among strings and numbers, and more.

2) The random integer returned by rand() will be in what range?

- 0 to 9
- -RAND_MAX to RAND_MAX
- [✓ 0 to RAND_MAX]

Correct

The range is quite large, starting from 0. RAND_MAX is at least 32,767, but on some machines may be larger (such as about 2 billion on a 32-bit machine).

3) Which expression's range is restricted to 0 to 7?

- rand() % 7
- [✓ rand() % 8]

Correct

% computes the modulo, meaning remainder. Possible remainders when dividing by 8 are 0, 1, 2, 3, 4, 5, 6, and 7.

4) Which expression yields one of 5 possible values?

- rand() % 4
- [✓ rand() % 5]
-

Correct

Possible values are 0, 1, 2, 3, and 4, which is 5 possible values.

rand() % 6

5) Which expression yields one of 100 possible values?

- rand() % 99
- [✓ rand() % 100]
- rand() % 101

Correct

Yields 0 to 99, which is 100 possible values (counting the 0).

6) Which expression would best mimic the random outcome of flipping a coin?

- rand() % 1
- [✓ rand() % 2]
- rand() % 3

Correct

Yields 2 possible values, 0 and 1. Those values can represent the two possible outcomes of a coin flip, heads and tails.

7) What is the smallest *possible* value returned by rand() % 10?

- [✓ 0]
- 1
- 10
- Unknown

Correct

The range of rand() starts with 0, and 0 % 10 is 0.

8) What is the largest *possible* value returned by rand() % 10?

- 10
- [✓ 9]
- 11

Correct

num % 10 can range from 0 to 9.

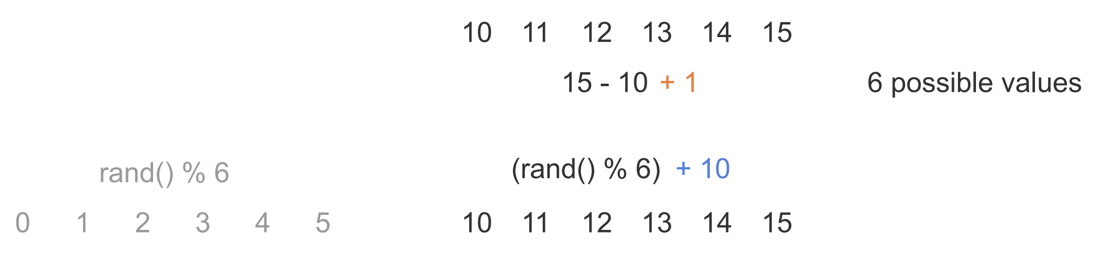
Feedback?

Specific ranges

The technique above generates random integers with N possible values ranging from 0 to N-1, like 6 values from 0 to 5. Commonly, a programmer wants a specific range that starts with some value x that isn't 0, like 10 to 15, or -20 to 20. The programmer should first determine the number of values in the range, generate a random integer with that number of possible values, and then add x to adjust the range to start with x.

PARTICIPATION ACTIVITY 2.19.3: Generating random integers in a specific range not starting from 0.

1 2 3 2x speed



Adding 10 still generates 6 values, but now those values start at 10.
The range thus becomes 10 to 15.

Feedback?

PARTICIPATION ACTIVITY 2.19.4: Generating random integers in a specific range.

1) Goal: Random integer from the 6 possible values 0 to 5.
rand() % ___

Check Show answer

Answer

6

% by 6 has 6 possible remainders: 0, 1, 2, 3, 4, 5.

2) Goal: Random integer from 0 to 4.
rand() % ___

Check Show answer

Answer

5

0 to 4 has 5 possible values: 0, 1, 2, 3, 4.
A common mistake is to assume 0 to 4 has 4 possible values because 4 − 0 = 4. But 1 more value exists in that range.

3) How many values exist in the range 10 to 15?

Check Show answer

Answer

6

15 − 10 + 1 = 6. A common mistake is to forget the + 1. But one can verify the 6 values by listing:

10, 11, 12, 13, 14, 15.

4) How many values exist in the range 10 to 100?

Check Show answer

Answer

91

100 − 10 + 1 = 91. A common mistake is to forget the + 1.

5) Goal: Random integer in the range 10 to 15.
(rand() % 6) + ___

Check Show answer

Answer

10

rand() % 6 yields 6 values from 0 to 5. Adding 10 thus yields 6 values from 0 + 10 = 10 to 5 + 10 = 15.

6) Goal: Random integer in the range 16 to 25.
(rand() % ___) + 16

Check Show answer

Answer

10

25 − 16 + 1 = 10. rand() % 10 generates values from 0 to 9. Adding 16 yields 0 + 16 = 16 to 9 + 16 = 25.

7) How many values are in the range -5 to 5?

Check Show answer

Answer

11

5 − (-5) + 1 = 11. Ranges involving negatives are treated the same as ranges with only positives. The 11 values are -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5.

8) Goal: Random integer in the range -20 to 20.
(rand() % 41) + ___

Check Show answer

Answer

-20

-20 to 20 has 41 values in the range (20 - (-20) + 1 = 41).
(rand() % 41)'s range is 0 to 40.
Adding -20 yields a range of -20 to 20.

Feedback?

PARTICIPATION ACTIVITY | 2.19.5: Specific range.

1) Which generates a random integer in the range 18 ... 30?

- ○ rand() % 30
- ○ rand() % 31
- ○ rand() % (30 - 18)
- ○ (rand() % (30 - 18)) + 18
- [✓ (rand() % (30 - 18 + 1)) + 18]

Correct

rand() % 13 yields a range of 0 to 12. Adding 18 yields a range of 18 ... 30.

Feedback?

The following program randomly moves a student from one seat to another seat in a lecture hall, perhaps to randomly move students before an exam. The seats are in 20 rows numbered 1 to 20. Each row has 30 seats (columns) numbered 1 to 30. The student should be moved from the left side (columns 1 to 15) to the right side (columns 16 to 30).

Figure 2.19.2: Randomly moving a student from one seat to another.

```
#include <iostream>
#include <cstdlib>
using namespace std;

// Switch a student
// from a random seat on the left  (cols  1 to 15)
//   to a random seat on the right (cols 16 to 30)
// Seat rows are 1 to 20

int main() {
   int rowNumL;
   int colNumL;
   int rowNumR;
   int colNumR;

   rowNumL = (rand() % 20) + 1;  // 1 to 20
   colNumL = (rand() % 15) + 1;  // 1 to 15

   rowNumR = (rand() % 20) + 1;  // 1 to 20
   colNumR = (rand() % 15) + 16; // 16 to 30

   cout << "Move from ";
   cout << "row " << rowNumL << " col " << colNumL;
   cout << " to " ;
   cout << "row " << rowNumR << " col " << colNumR;
   cout << endl;

   return 0;
}
```

```
Move from row 8 col 5 to row 14 col 24
```

Feedback?

PARTICIPATION ACTIVITY | 2.19.6: Random integer example: Moving seats. ✓

Consider the above example.

1) The row is chosen using (rand() % 20) + 1. The 20 is because 20 rows exist. The + 1 is _____ .

[✓ necessary]
optional

Correct

The + 1 is necessary because (rand() % 20)'s range is 0 to 19, but the rows are numbered 1 to 20. Thus, adjustment by + 1 is needed.

2) The column for the left is chosen using (rand() % 15) + 1. The 15 is used because the left half of the hall has _____ seats.

[✓ 15]
30

Correct

The seats are in 30 columns numbered 1 to 30. The left half is numbered 1 to 15, and the right half 16 to 30. For 1 to 15, there are 15 − 1 + 1 = 15 values.

3) The column for the right could have been chosen using (rand() % 15) + 15.

True
[✓ False]

Correct

The right columns are numbered 16 to 30. (rand() % 15)'s range is 0 to 14. Adding 15 would have yielded 15 to 29, which is wrong. The starting value of the desired range should be added, in this case 16.

Feedback?

Pseudo-random

The integers generated by rand() are known as pseudo-random. "Pseudo" means "not actually, but having the appearance of". The integers are pseudo-random because each time a program runs, calls to rand() yield the same sequence of values. Earlier in this section, a program called rand() three times and output 16807, 282475249, 1622650073. Every time the program is run, those same three integers will be printed. Such reproducibility is important for testing some programs. (Players of classic arcade games like Pac-man may notice that the seemingly-random actions of objects actually follow the same pattern every time the game is played, allowing players to master the game by repeating the same winning actions).

Internally, the rand() function has an equation to compute the next "random" integer from the previous one, (invisibly) keeping track of the previous one. For the first call to rand(), no previous random integer exists, so the function uses a built-in integer known as the ***seed***. By default, the seed is 1. A programmer can change the seed using the function srand(), as in srand(2) or srand(99).

If the seed is different for each program run, the program will get a unique sequence. One way to get a different seed for each program run is to use the current time as the seed. The function ***time()*** returns the number of seconds since Jan 1, 1970.

Note that the seeding should only be done once in a program, before the first call to rand().

Figure 2.19.3: Using a unique seed for each program run.

```
#include <iostream>
#include <cstdlib>
#include <ctime>    // Enables use of time() function
using namespace std;

int main() {
   srand(time(0));  // Unique seed
   cout << rand() << endl;
   cout << rand() << endl;
   cout << rand() << endl;

   return 0;
}
```

```
636952311
51510682
304122633

(next run)

637053153
1746362176
1450088483
```

Feedback?

PARTICIPATION ACTIVITY | 2.19.7: Using a unique seed for each program run. ✓

1) The s in srand() most likely stands for _____ .

- sequence
- [✓ seed]

✓ **Correct**

srand() sets the seed value that will be used by rand() to generate the first "random" integer in the sequence that rand() generates.

2) By starting a program with srand(15), calls to rand() will yield a different integer sequence for each program run.

- True
- [✓ False]

✓ **Correct**

The sequence will differ than if srand() was not called (which is the same as srand(1)), but that sequence will be the same for every program run. Ex: The sequence following srand(2) may be 33614, 564950498, and 1097816499. Every program run will yield that same sequence.

3) By starting a program ✓

with srand(time(0)), calls to rand() will yield a different integer sequence for each successive program run.

[✓ True] False

Correct

time(0) returns the number of seconds since Jan 1, 1970. Thus, each successive program run will be seeded with a higher value than the previous run, yielding a different sequence.

4) rand() is known as generating a "pseudo-random" sequence of values because the sequence begins repeating itself after about 20 numbers.

True [✓ False]

Correct

The "pseudo" is because the sequence is the same for the given seed, with the next "random" number computed from the previous one. But the sequence does not repeat itself; millions of numbers can be generated without reaching a repeating pattern within the sequence.

Feedback?

Exploring further:

- C++ random number library

CHALLENGE ACTIVITY 2.19.1: Generate a random integer.

Jump to level 1

1
2
3
4

Generate a random integer between -3 and 6 (inclusive)

(rand() % 10) + -3

1 2 3 **4**

Check Next

Done. Click any level to practice more. Completion is preserved.

✓ Expected: (rand() % 10) + -3. The divisor is the range size, so 6 - -3 + 1 = 10, which yields 0 to 9. That range is shifted by -3 to yield -3 to 6.

Feedback?

CHALLENGE ACTIVITY | 2.19.2: rand function: Seed and then get random numbers.

Type a statement using srand() to seed random number generation using variable seedVal. Then type **two statements** using rand() to print two random integers between (and including) 0 and 9. End with a newline. Ex:

```
5
7
```

Note: For this activity, using one statement may yield different output (due to the compiler calling rand() in a different order). Use two statements for this activity. Also, after calling srand() once, do not call srand() again. (Notes)

```
#include <iostream>
#include <cstdlib>   // Enables use of rand()
using namespace std;

int main() {
   int seedVal;

   cin >> seedVal;

   /* Your solution goes here  */
   srand(seedVal);
   cout << rand()%10 << endl;
  cout << rand()%10 << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with seedVal = 4

Your output
```
1
3
```

✓ Testing with seedVal = 55

Your output
```
8
5
```

Feedback?

CHALLENGE ACTIVITY | 2.19.3: Fixed range of random numbers.

Type **two statements** that use rand() to print 2 random integers between (and including) 100 and 149. End with a newline. Ex:

```
101
133
```

Note: For this activity, using one statement may yield different output (due to the compiler calling rand() in a different order). Use two statements for this activity. Also, srand() has already been called; do not call srand() again.

```
#include <iostream>
#include <cstdlib>   // Enables use of rand()
#include <ctime>     // Enables use of time()
using namespace std;

int main() {
   int seedVal;

   cin >> seedVal;
   srand(seedVal);

   /* Your solution goes here  */
   cout << rand()%50 +100<<endl;
   cout <<rand()%50 +100<<endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with seedVal = 4

Your output
```
101
133
```

✓ Testing with seedVal = 55

Your output

```
148
125
```

Feedback?