**rand function**

A highly practical function – generates random number.

This function doesn’t take any arguments and generates a random number from 0 to at least $2^{16}$.

What if the random number we’re looking for is much smaller? Say single digit.
- use remainder
- we know any number $\% n$, result 0 to $n - 1$

Thus, print a single digit,
```
cout << rand() % 10;
```
Sample rand

/* simulates dice roll, * print a random value between 1 - 6 */
cout << rand() % 6 + 1 << endl;

General rule:
Get 1 random number between a to b
rand() % how many numbers in range a & b
then + lowest possible random number

Thus, rand() % (b - a + 1) + a
Sample rand

/* print three, 2 digit random numbers */
for (int i = 1; i <= 3; i++){
    cout << rand() % 90 + 10 << endl;
}

But if you just try this in a program, you’ll find it prints out the same sequence (3 number) every single time. Why? Isn’t random number should be random?
rand function

rand() function is not random at all.

It’s actually pseudo-random numbers, which gives a set of numbers using complex algorithm.

To make it gives a different sequence of numbers every time, we have to set an unique random seed (starting number in the sequence)

Every time the program is run, srand(time(0));
/* print three, 2 digit random numbers */
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main(){
    srand(time(0));
    for (int i = 1; i <= 3; i++){
        cout << rand() % 90 + 10 << endl;
    }
    return 0;
}
Write Functions

Why do we need to write our own functions?

- A task might be performed repeatedly through out different parts of the program. Instead of copy and paste the same code, we may reuse the call, which define this part as a function, whenever this code need to be execute in the program, we make a function call.

- It’s easier to make modification in one place than multiple places.

- Keep main function simple and easier to manage while focusing on the task.
User-defined Functions

1. At the beginning of the program declare the function
2. Define the function, usually after main(){}
3. The function may be called from another function

Sample:
#include <iostream>
t
int someTask(int x){
    std::cout << "Do something " << x << std::endl;
    return x - 1;
}

int main(){
    int v = 10;
    std::cout << "Start ." << std::endl;
    v = someTask(v);
    v = someTask(v);
    std::cout << "End."
    return 0;
}
Declare and Define Function

-(Declare / Prototype) This gives the compiler a preview of what your function would look like
• After “using namespace std;”, before “int main()”

-(Definition) Define the actions the function should take
• Usually after “int main{ ... return 0;}”
Declare Function

prototype functions syntax:

return_type function_name(argument_type_list)

- return_type the data type of which the function returns
- function_name name of the function
- argument_list List of argument data types, separate each one with a comma
Return Type

The function can return 1 or none object back to the caller.

return_type specify the data type of this object
Ex:
sqrt(); // returns double
double sqrt(); // returns int
If the function doesn’t return anything, the return type would be “void”.
Parameter/Argument List

The function may need some information/data in order to finish the task.

Pass all the data as it’s argument
Function takes these data as it’s parameter

Ex: sqrt(100.1);
- take a double value to compute it’s square root.

If the no information is needed, the parameter list is empty.
If more than one information is needed, each separated by a comma “,”
Each parameter has it’s own data type and name!
Sample prototypes

double sqrt(double v);

int rand();

A function called avg which takes two double values, compute and return the exact average of two.

double avg(double a, double b);
Sample prototypes

string makeString();

double max(double a, double b, double c);

bool check(string st);

void print(string st);

// 4 functions called below
int main() {
    string st = makeString();

    cout << max(3.4, 3.2, 3.5);  // print 3.5

    if (check(st)) print(st);
    return 0;
}