



Recursion

Pittsford Sutherland High School

By Yiyou Chen, Yizuo Chen

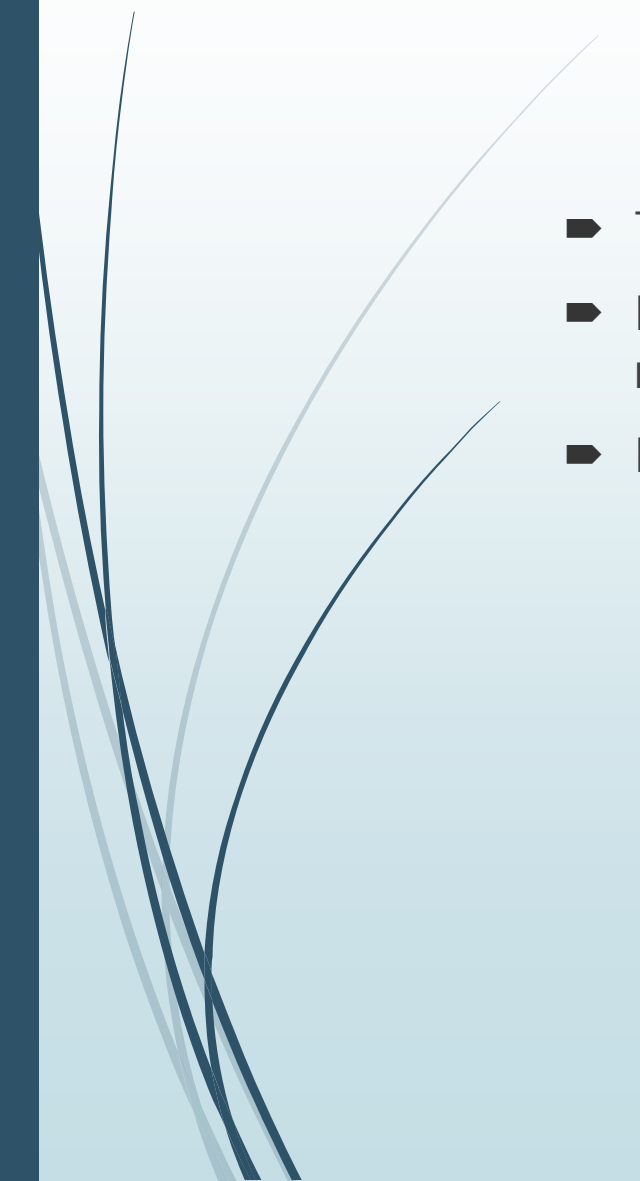


Function

- In C++ grammar section, we introduced the concept of functions. Many people did not quite understand why we want to use functions instead of directly writing all the things in main.
- One minor reason why we use function is to make the codes more clear and easy to read. But the major reason of existence of functions is that functions have powerful usage such as recursions.



What is recursion?

- The repeated application of a recursive procedure.
 - It can be used to find answers by repeatedly use itself, which is the main reason why we introduced C++ functions.
 - It is mainly used to solve problems related to recursive functions.
- 

Recursive functions

- Recursive functions are functions that repeatedly use the known value of itself.
- For example:
- Fibonacci sequences: $f(1) = f(2) = 1; f(x) = f(x - 1) + f(x - 2)$
- $N! : f(x) = f(x - 1) * x$
- A random Sequence: $a_1 = 1$

$$a_n = \begin{cases} \frac{a_{n-1}}{n-1} & \text{if } (n-1) \mid a_{n-1} \\ (n-1) a_{n-1} & \text{otherwise} \end{cases}$$

.....



Recursion in programming

- Also recursion is useful and easy to write, we should be extremely careful when we use it.
- 1. Since the complexity of recursions are usually large, we must notice the range of the input before we use recursions.
- 2. Every time when we use recursions, we need to be careful about the boundary. For example, the boundary for Fibonacci sequences is $f(1) = f(2) = 1$.
- Now we don't have any better way to make recursion faster. However, when we are in the chapter about dynamic programming, we are able to use memorization to make recursion much faster and smarter.

Example 1: Fibonacci sequences

- In this example, we want to use recursion to find the 100th term of Fibonacci sequences.

C++ Code:

```
int find_Fibonacci(int n){  
    if(n == 1 or n == 2) return 1;  
    return (find_Fibonacci(n - 1) + find_Fibonacci(n - 2));  
}  
  
int main(){  
    cout << find_Fibonacci(100) << endl;  
}
```

- The highlighted part uses recursive. As you might have noticed, the time complexity is quite large for this program.

Example 2: N!

- In this example, we want to use recursion to find the value of 10!.

```
int find(int n){  
    if(n == 1) return 1;  
    return (n * find(n - 1));  
}  
int main(){  
    cout << find(10) << endl;  
}
```

- The highlighted part uses recursion. This program also has a quite large time complexity.

Example 3: Random Sequences

$$a_1 = 1$$
$$a_n = \begin{cases} \frac{a_{n-1}}{n-1} & \text{if } (n-1) \mid a_{n-1} \\ (n-1) a_{n-1} & \text{otherwise} \end{cases}$$

- In this example, we want to use recursion to find the value of $a[20]$.

```
int Recaman(int x){
    if(x == 1) return 1;
    if(Recaman(x - 1) % (x - 1) == 0) return Recaman(x - 1) / (x - 1);
    else return Recaman(x - 1) * (x - 1);
}

int main(){
    cout << Recaman(20) << endl;
}
```

- Again, this program is quite slow.



More applications of recursion

- Nearly all functions use recursions.
- Recursion can be widely used in graph theory problems like Breadth first search and depth first search.
- Recursion can also be used to solve math problems or be used to prove a theorem.