Example: Write a program to check prime number by using function idea.

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

int prime(int n);

int main()
{
    int num, flag = 0;
    cout << "Enter positive integer to check: ";
    cin >> num;

    // Argument num is passed to check() function
    flag = prime(num);

    if(flag == 1)
        cout << num << " is not a prime number.";
    else
        cout<< num << " is a prime number.";
    return 0;
}

/* This function returns integer value.  */
int prime(int n)
{
    for(int i = 2; i <= n/2; ++i)
    {
        if(n % i == 0)
            return (1);
    }

    return (0);
}
```

Example: A group of numbers ending with zero. Write a program to check each number whether it is odd or even.

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

void odd (int);
void even (int);

int main()
{
  int value;
  cout << "Please, enter number (0 to exit): ";
  cin >> value;

  while (value != 0) {
     odd(value);
     cout << "Please, enter number (0 to exit): ";
     cin >> value;

  }
  return 0;
}

void odd (int x)
{
  if ((x % 2) !=0) cout << "It is odd.\n";
  else even(x);
}

void even(int x)
{
  if ((x % 2) == 0)
    cout << "It is even.\n";
}
```

<u>Example:</u> Write a C++ Program to compute the value of y in the following series.

$$y = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + \frac{x^n}{n!}$$

```
#include <iostream>
#inlclude <cmath>
using namespace std;

int main() {
  float x, y=0.0;
  int n;
  cout<<"Enter x = ";   cin>> x;
  cout<<"Enter n =";    cin>> n;
  for (int i=1; i <= n;  i +=2) {
        int f = 1;
        for (int j=1; j <= i; ++j)
              f  *= j;
       y += pow(x, i) / f ;
  }
 cout<<" \n The value of y = "<<y ;
 return 0;
}
```

Another solution by using functions:

```
#include <iostream>
#inlclude <cmath>
float series(float, int);
int factv(int);

int main() {
  float x, y;
  int n;
  cout<<"Enter x = ";  cin>> x;
  cout<<"Enter n =";   cin>> n;
  y = series(x, n);
  cout<<" \n The value of y = "<< y ;
 return 0;
}

float series(float x, int n)
{   float R = 0;
    for (int i=1; i <= n;  i +=2)
```

```
        R += pow(x, i) / factv(i) ;
   return (R);
}

int factv(int v)
{
  int f = 1;
  for (int j=1; j <= v; ++j)
        f  *= j;
  return (f);
}
```

Example : A sequence of positive integer, terminated by -1, is read from keyboard. Write a C++ program to determine whether these number are in ascending order.

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

int main() {
  int x, y;
  bool Check = true;
  cout<<"Enter x = ";    cin>> x;
  while (Check == true  && x != -1 ) {
       y = x ;
       cout<<"Enter next number = ";     cin>> x;
       if  (y > x)
          Check = false;
    }
   if (Check == true)   cout<<" The numbers are in the ascending order " ;
   else                 cout<<" The numbers are not in the ascending order " ;
   return 0;
}
```

Another solution by using function:

```
#include <iostream>
using namespace std;
bool check_asc(bool);
void printing( bool);

int main()
{  bool Check = true;
   Check = check_asc(Check);
   printing(Check);
   return 0;
}

bool check_asc(bool ch)
{  int x, y;
   cout<<"Enter x = ";   cin>> x;
   while (ch == true  && x != -1 ) {
         y = x ;
         cout<<"Enter next number = ";    cin>> x;
         if  (y > x)
            ch = false;
      }
  return (ch);
}

void printing(bool ch)
{   if (ch == true)
        cout<<" The numbers are in the ascending order " ;
    else
        cout<<" The numbers are not in the ascending order " ;
}
```

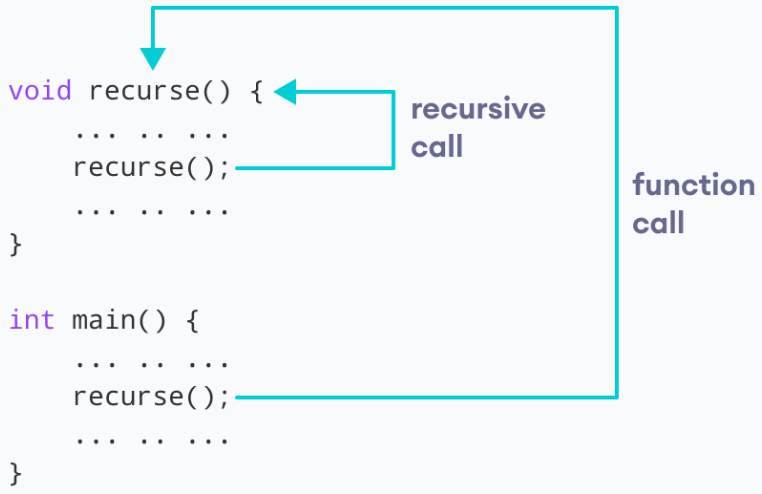
C++ Recursion

A function that calls itself is known as a recursive function, and this technique is known as recursion.

```
void recurse()
{
   ... .. ...
   recurse();
   ... .. ...
}

int main()
{
   ... .. ...
   recurse();
   ... .. ...
}
```

The figure below shows how recursion works by calling itself over and over again.



The recursion continues until some condition is met. To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and the other doesn't.

Example 1: Factorial of a Number Using Recursion

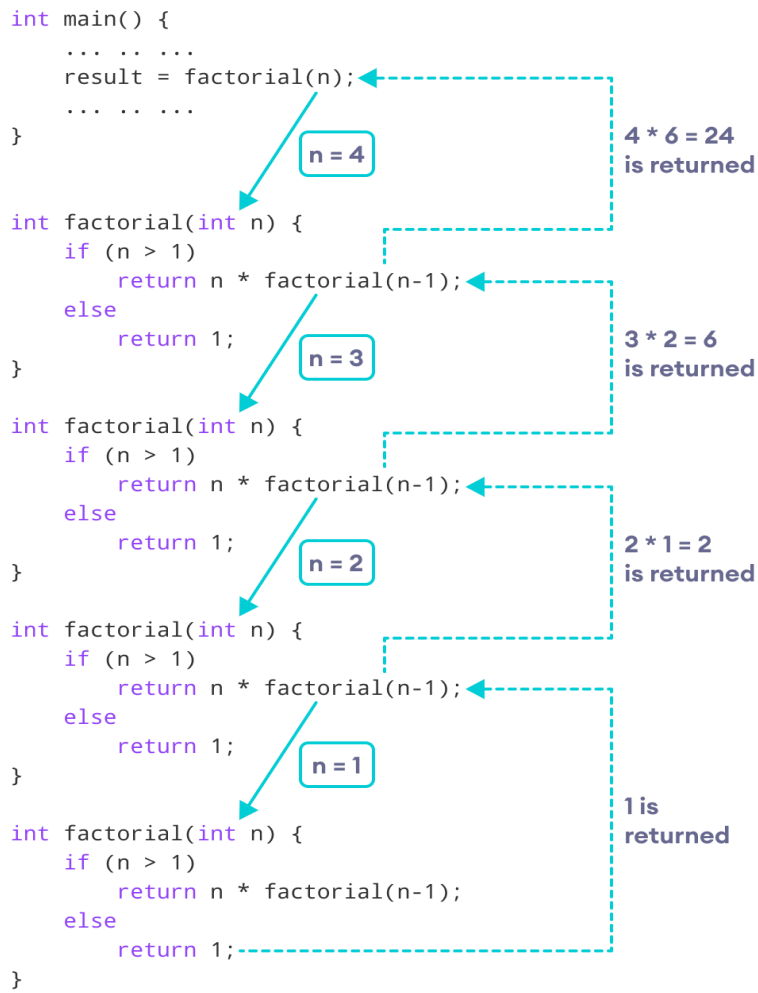
```
#include <iostream>
using namespace std;
int factorial(int);

int main() {
    int n, result;
    cout << "Enter a non-negative number: ";
    cin >> n;

    result = factorial(n);
    cout << "Factorial of " << n << " = " << result;
    return 0;
}

int factorial(int n) {
    if (n > 1)
        return (n * factorial(n-1));
    else
        return (1);

}
```



Example: Write a program to achieve and represent the following mathematical equation by using function recursion idea:

$$sum(x) = \sum_{i=x}^{0} i, \ldots\ldots where\ x \geq 0$$

```
#include <iostream>
using namespace std;
int sum(int);

int main()

{
  int result = sum(10);
  cout << result;
  return 0;
}

int sum(int k) {
  if (k > 0)
    return (k + sum(k-1));
  else
    return (0);
}
```

Example Explained

When the `sum()` function is called, it adds parameter `k` to the sum of all numbers smaller than `k` and returns the result. When k becomes 0, the function just returns 0. When running, the program follows these steps:

```
10 + sum(9)
10 + ( 9 + sum(8) )
10 + ( 9 + ( 8 + sum(7) ) )
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```