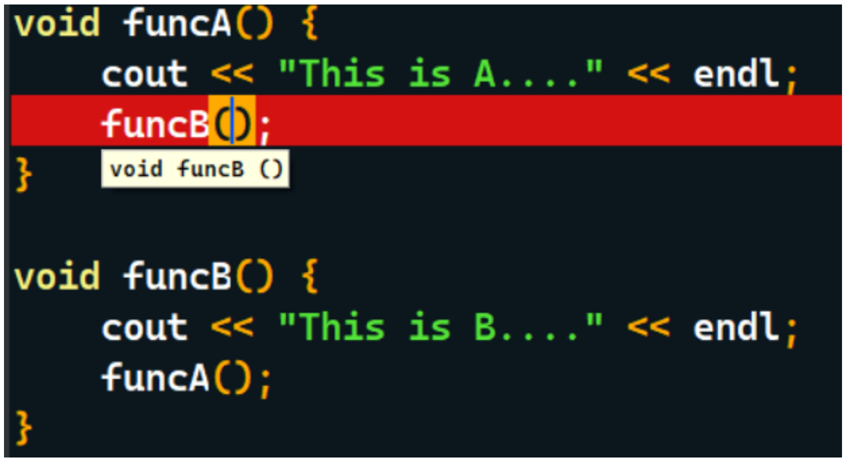
Summary

Sessions (05-01-2023)

- Create a function before we call that function.

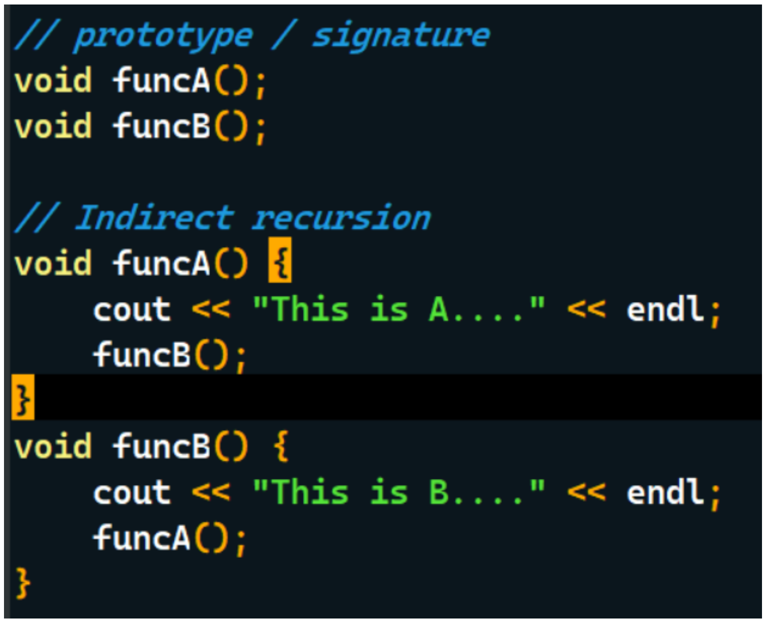
- If we call “funcA()” from the main() function then it will show an error as “*funcB() was not declared in this scope*” means “funcA()” don’t know about “funcB()” because “funcB()” is created after the call.



```
void funcA() {
    cout << "This is A...." << endl;
    funcB();
}

void funcB() {
    cout << "This is B...." << endl;
    funcA();
}
```

- If we want to solve the above problem then we have to tell signatures before calling it, that is known as prototype.



```
// prototype / signature
void funcA();
void funcB();

// Indirect recursion
void funcA() {
    cout << "This is A...." << endl;
    funcB();
}
void funcB() {
    cout << "This is B...." << endl;
    funcA();
}
```

- If a recursive function calling itself and that recursive call is the first statement in the function then it's known as **Head or Non-Tail Recursion**.

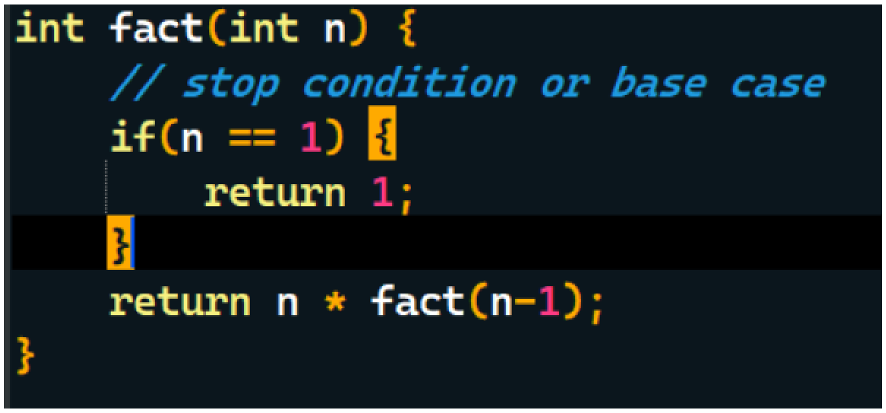
- If a recursive function calling itself and that recursive call is the last statement in the function then it's known as **Tail Recursion**.

- As we know using tail recursion, recursion will convert into iteration and this is known as **TCO(Tail Call Optimization)**. If code is in tail recursion then it will never face stackoverflow kind of issues.

- Example of Head Recursion :

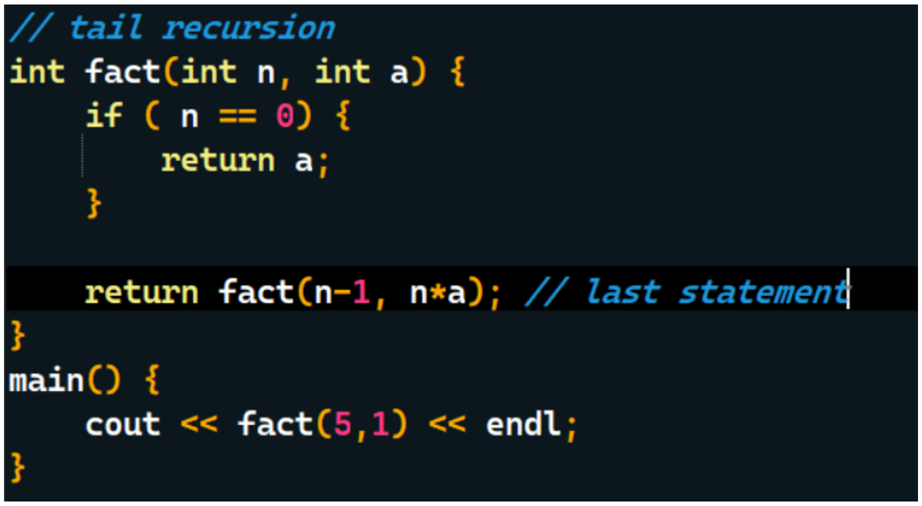
  As we can see in the code, the last operation / statement is not a recursive function therefore below code is in head recursion.

  In the code last statement is "*n * fact(n-1)*" and last operation is "*" not fact(n-1) function.



```
int fact(int n) {
    // stop condition or base case
    if(n == 1) {
        return 1;
    }
    return n * fact(n-1);
}
```

- Example of Tail Recursion :



```
// tail recursion
int fact(int n, int a) {
    if ( n == 0) {
        return a;
    }

    return fact(n-1, n*a); // last statement
}
main() {
    cout << fact(5,1) << endl;
}
```