430.211 Programming Methodology

File I/O & Functions

Lab 1 Week 3

Fall 2025

Machine Intelligence & Data science LAB

TA Information

- Main TA
  - Juhyeon Park (parkjh9229@naver.com)

- Support TA
  - Jungyoon Hwang
  - Yongho Kim

Outline

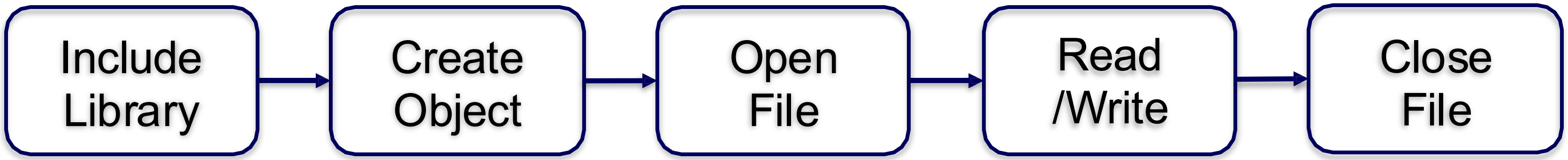
- File I/O
- Functions
  - Predefined functions
  - Programmer-defined functions
  - Iteration vs Recursion
  - Parameters, arguments
  - Call-by-values vs Call-by-references

- Assignment
- Attendance Check

File I/O

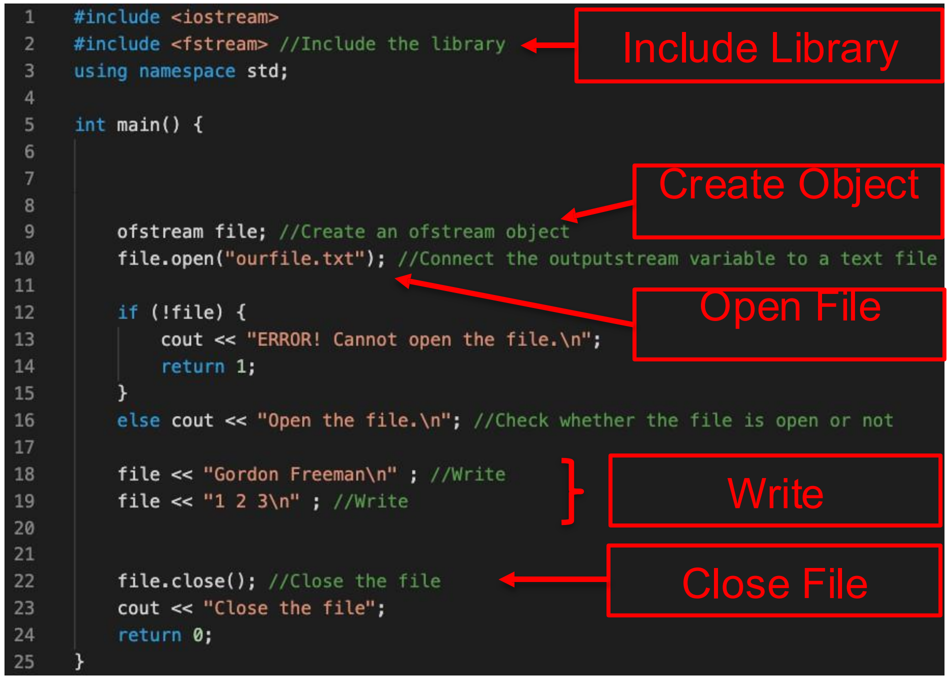
- To read and write a file, we use the **<fstream> library.**
  - The **ifstream** data type is used to **read** a file.
  - The **ofstream** data type is used to **write** a file.
  - The **fstream** data type is used to **read and write** a file.

- The process to read/write a file

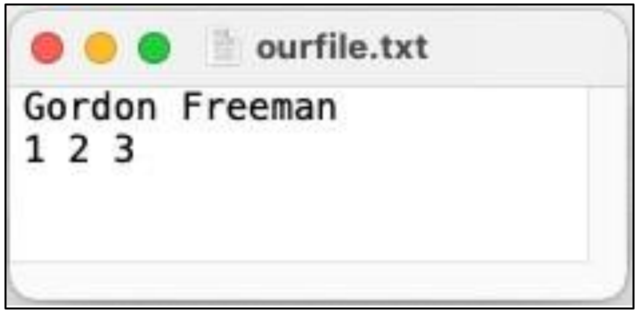


File I/O

- File write example
  - **Write** a new file with a name "ourfile.txt" as below.



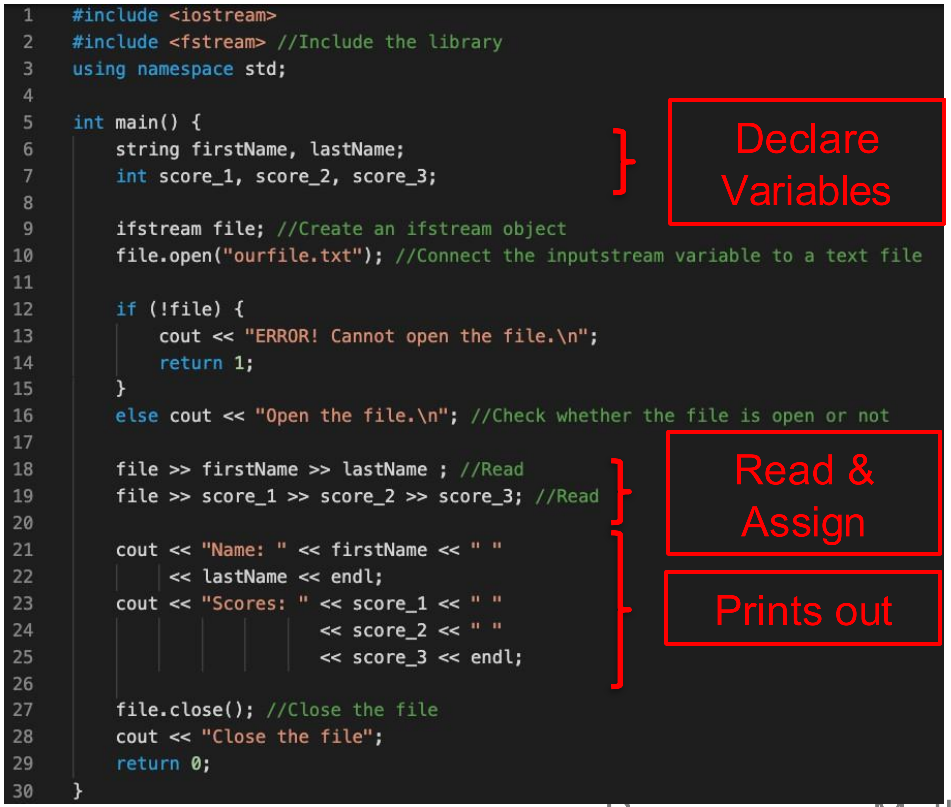
```cpp
#include <iostream>
#include <fstream> //Include the library
using namespace std;

int main() {



    ofstream file; //Create an ofstream object
    file.open("ourfile.txt"); //Connect the outputstream variable to a text file

    if (!file) {
        cout << "ERROR! Cannot open the file.\n";
        return 1;
    }
    else cout << "Open the file.\n"; //Check whether the file is open or not

    file << "Gordon Freeman\n" ; //Write
    file << "1 2 3\n" ; //Write


    file.close(); //Close the file
    cout << "Close the file";
    return 0;
}
```



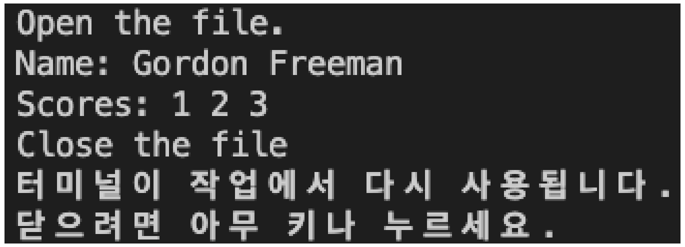
```
Gordon Freeman
1 2 3
```

File I/O

- Ex 1. File read example
  - **Read** the file and **assign** informations to our variables.



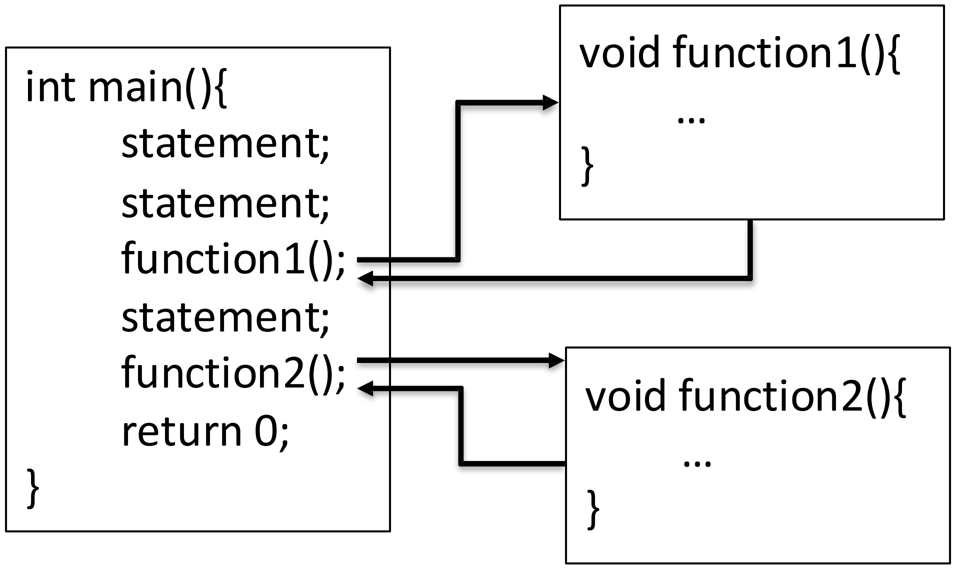
```cpp
#include <iostream>
#include <fstream> //Include the library
using namespace std;

int main() {
    string firstName, lastName;
    int score_1, score_2, score_3;

    ifstream file; //Create an ifstream object
    file.open("ourfile.txt"); //Connect the inputstream variable to a text file

    if (!file) {
        cout << "ERROR! Cannot open the file.\n";
        return 1;
    }
    else cout << "Open the file.\n"; //Check whether the file is open or not

    file >> firstName >> lastName ; //Read
    file >> score_1 >> score_2 >> score_3; //Read

    cout << "Name: " << firstName << " "
         << lastName << endl;
    cout << "Scores: " << score_1 << " "
                       << score_2 << " "
                       << score_3 << endl;

    file.close(); //Close the file
    cout << "Close the file";
    return 0;
}
```



```
Open the file.
Name: Gordon Freeman
Scores: 1 2 3
Close the file
터미널이 작업에서 다시 사용됩니다.
닫으려면 아무 키나 누르세요.
```

Functions

- Building blocks of programs
- Input – Process – Output structure
- Basic subparts to any program



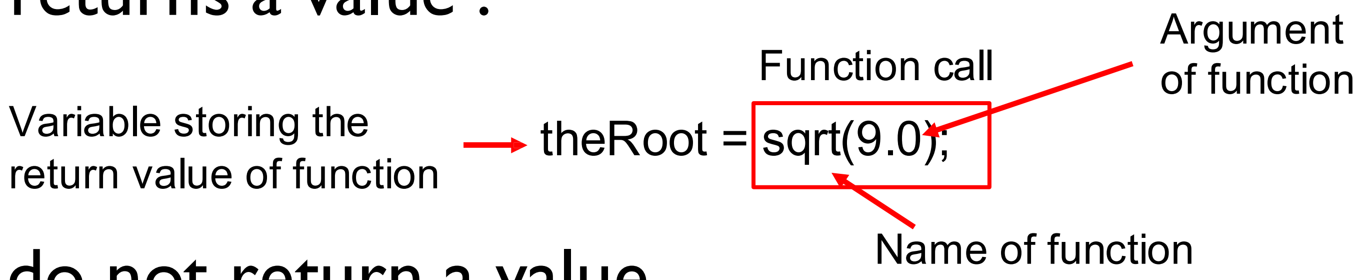
```
int main(){
        statement;
        statement;
        function1();
        statement;
        function2();
        return 0;
}
```

```
void function1(){
        …
}
```

```
void function2(){
        …
}
```

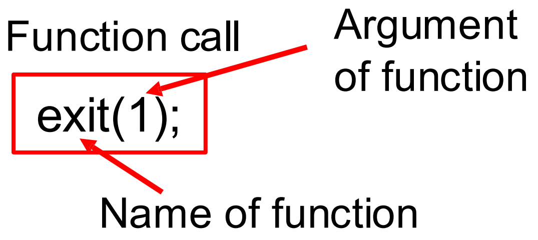
Predefined Functions

- “Ready-to-use” functions
- Must **include** appropriate library
  - #include ~
- Using a predefined function that
- 1) returns a value :



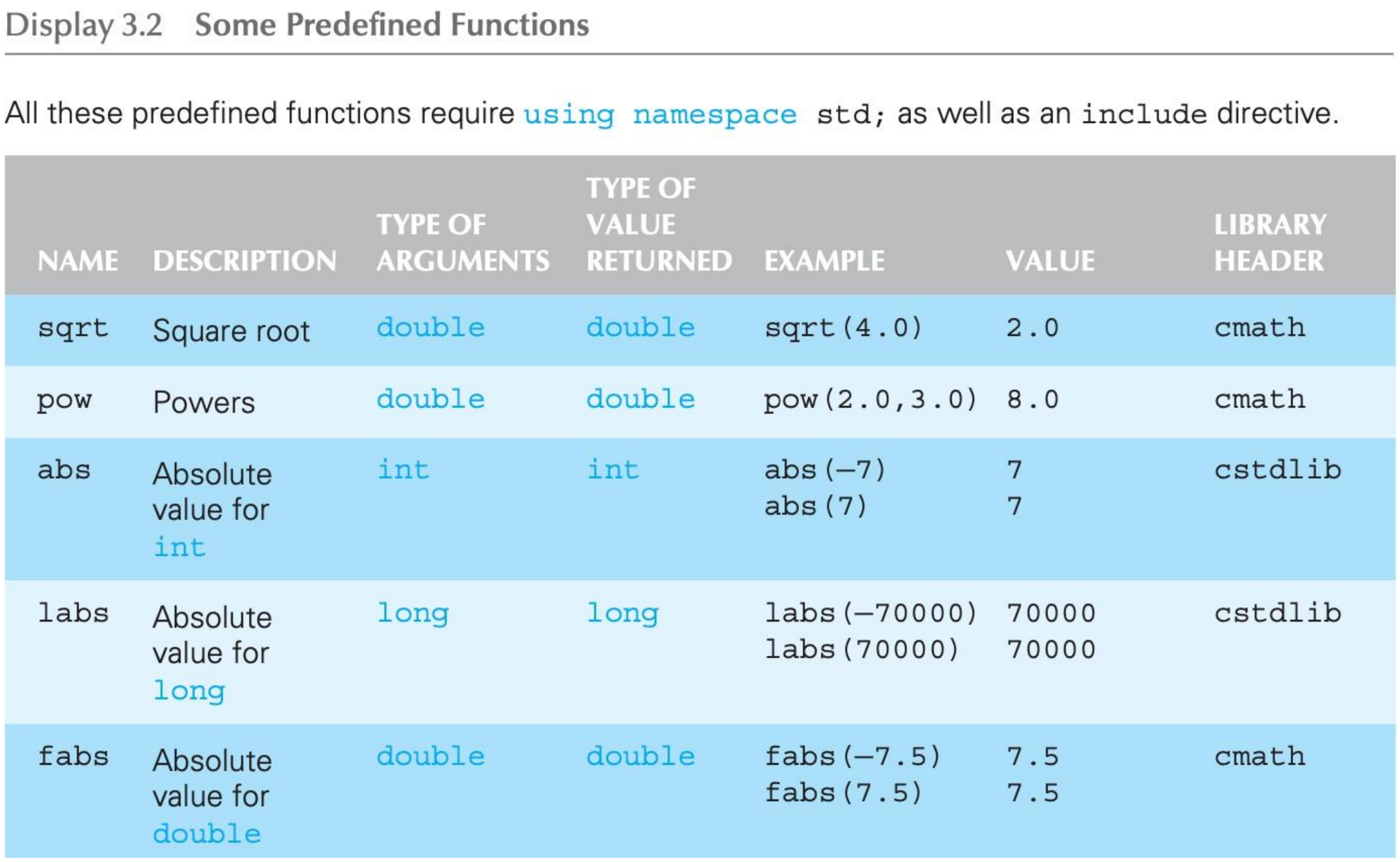
```
theRoot = sqrt(9.0);
```

2) do not return a value



```
exit(1);
```

Predefined Functions



Display 3.2 Some Predefined Functions

All these predefined functions require `using namespace std;` as well as an `include` directive.

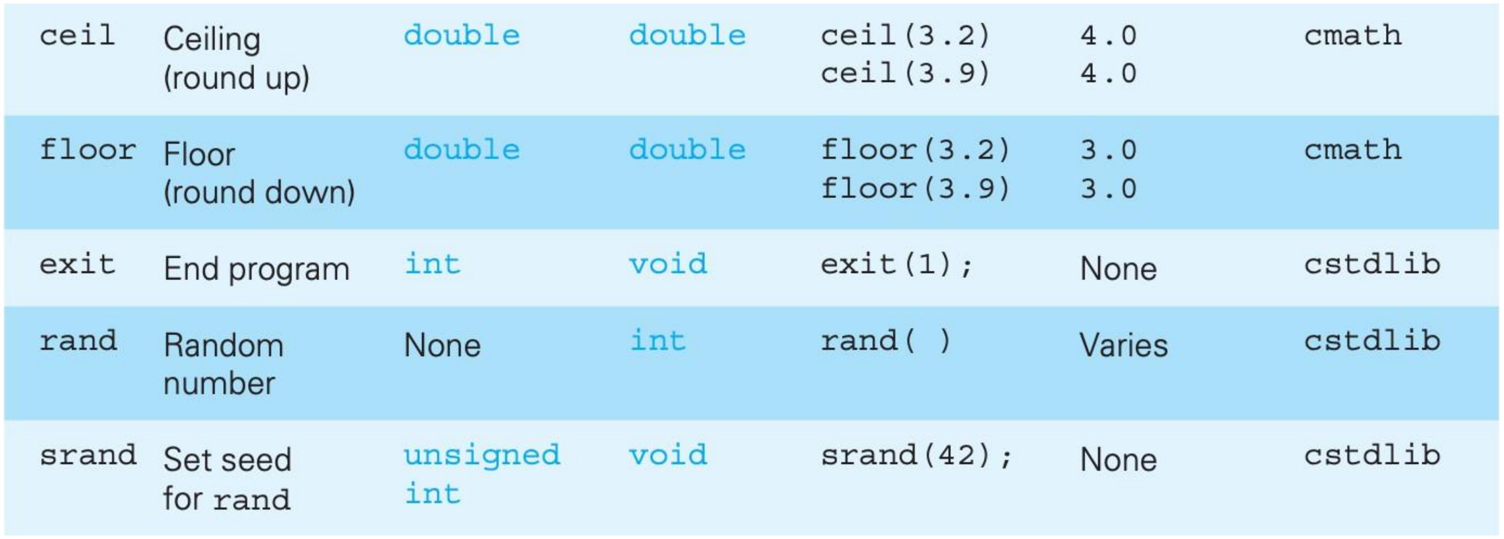
NAME	DESCRIPTION	TYPE OF ARGUMENTS	TYPE OF VALUE RETURNED	EXAMPLE	VALUE	LIBRARY HEADER
`sqrt`	Square root	`double`	`double`	`sqrt(4.0)`	`2.0`	`cmath`
`pow`	Powers	`double`	`double`	`pow(2.0,3.0)`	`8.0`	`cmath`

`abs`	Absolute value for `int`	`int`	`int`	`abs(-7)`
				`abs(7)`
					`7`
					`7`
						`cstdlib`

`labs`	Absolute value for `long`	`long`	`long`	`labs(-70000)`
				`labs(70000)`
					`70000`
					`70000`
						`cstdlib`

`fabs`	Absolute value for `double`	`double`	`double`	`fabs(-7.5)`
				`fabs(7.5)`
					`7.5`
					`7.5`
						`cmath`

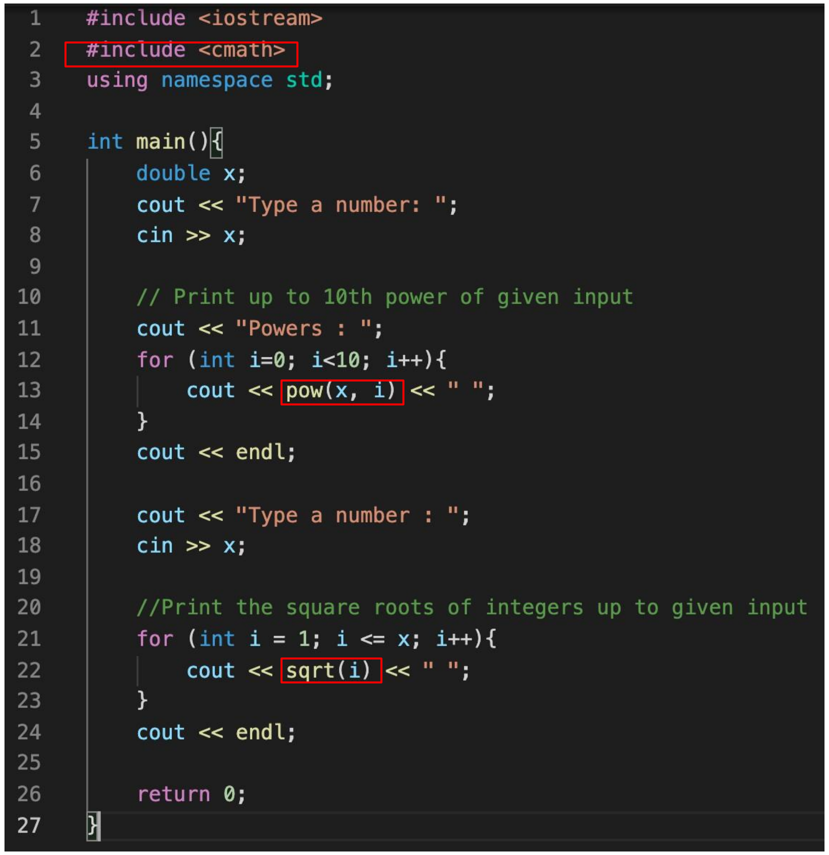
Predefined Functions



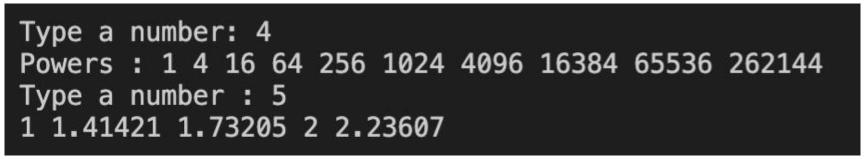
`ceil`	Ceiling (round up)	`double`	`double`	`ceil(3.2)` `ceil(3.9)`	`4.0` `4.0`	`cmath`
`floor`	Floor (round down)	`double`	`double`	`floor(3.2)` `floor(3.9)`	`3.0` `3.0`	`cmath`
`exit`	End program	`int`	`void`	`exit(1);`	None	`cstdlib`
`rand`	Random number	None	`int`	`rand( )`	Varies	`cstdlib`
`srand`	Set seed for `rand`	`unsigned int`	`void`	`srand(42);`	None	`cstdlib`

Predefined Functions

- Ex 2."sqrt" and "pow" functions



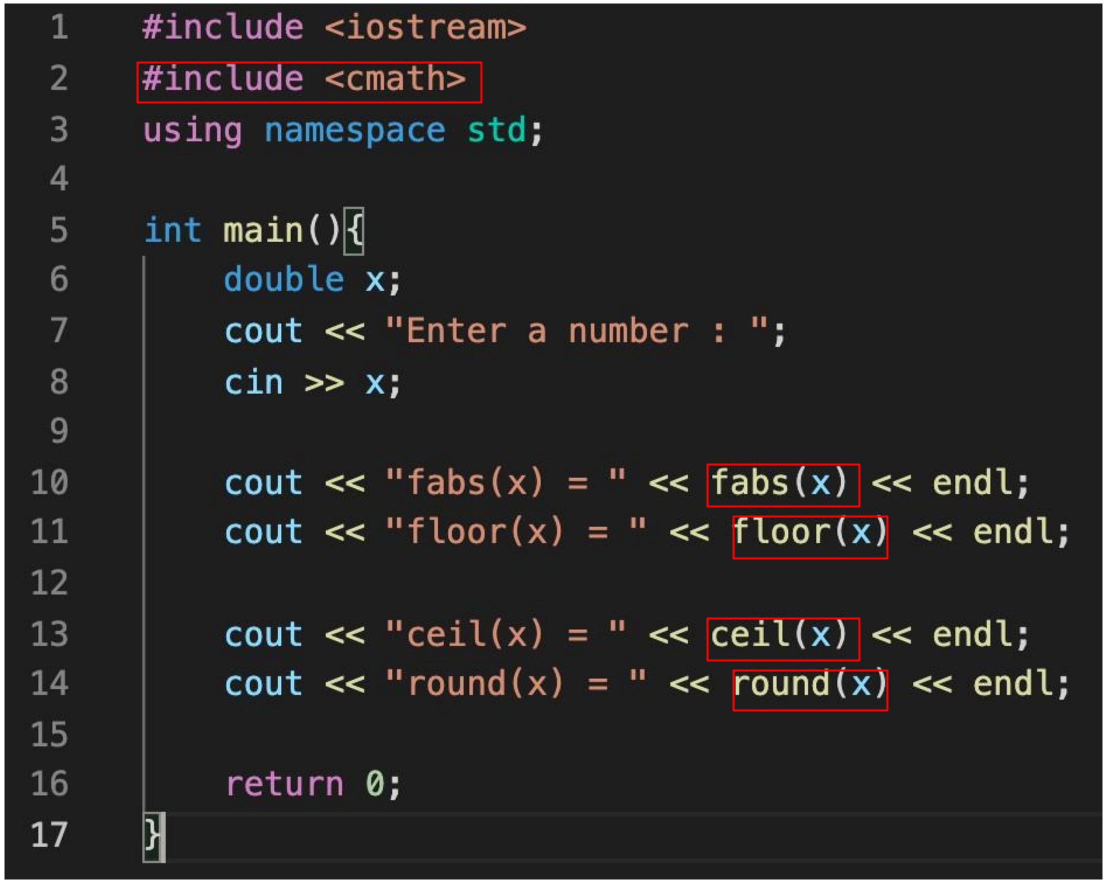
```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main(){
    double x;
    cout << "Type a number: ";
    cin >> x;

    // Print up to 10th power of given input
    cout << "Powers : ";
    for (int i=0; i<10; i++){
        cout << pow(x, i) << " ";
    }
    cout << endl;

    cout << "Type a number : ";
    cin >> x;

    //Print the square roots of integers up to given input
    for (int i = 1; i <= x; i++){
        cout << sqrt(i) << " ";
    }
    cout << endl;

    return 0;
}
```



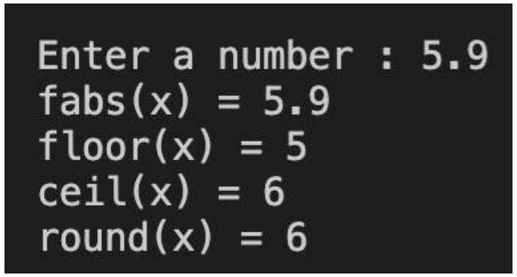
```
Type a number: 4
Powers : 1 4 16 64 256 1024 4096 16384 65536 262144
Type a number : 5
1 1.41421 1.73205 2 2.23607
```

Predefined Functions

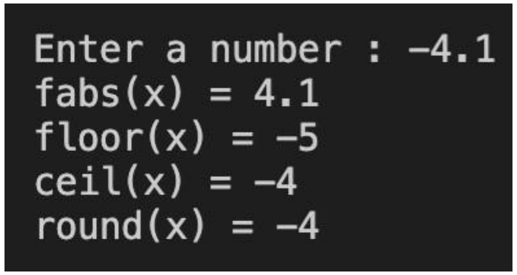
- Ex 3."fabs", "floor", and "ceil" functions



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

int main(){
    double x;
    cout << "Enter a number : ";
    cin >> x;

    cout << "fabs(x) = " << fabs(x) << endl;
    cout << "floor(x) = " << floor(x) << endl;

    cout << "ceil(x) = " << ceil(x) << endl;
    cout << "round(x) = " << round(x) << endl;

    return 0;
}
```



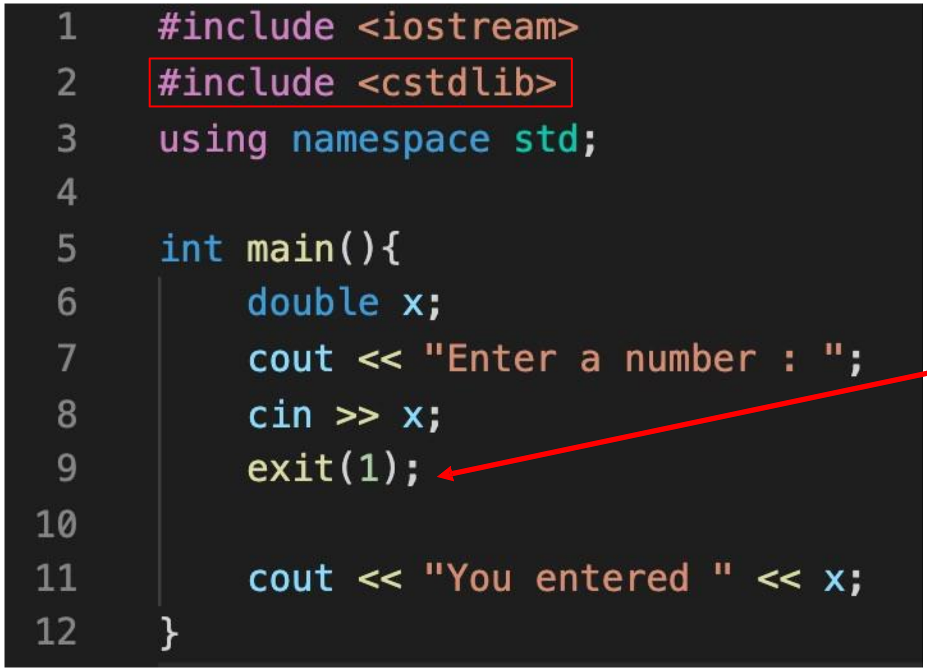
```
Enter a number : 5.9
fabs(x) = 5.9
floor(x) = 5
ceil(x) = 6
round(x) = 6
```



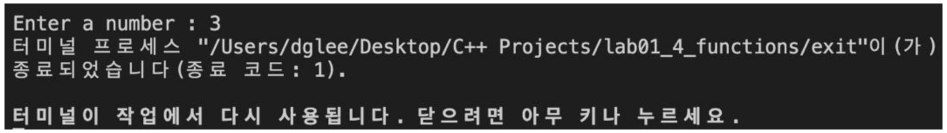
```
Enter a number : -4.1
fabs(x) = 4.1
floor(x) = -5
ceil(x) = -4
round(x) = -4
```

Predefined Functions

- Ex 4."exit" function



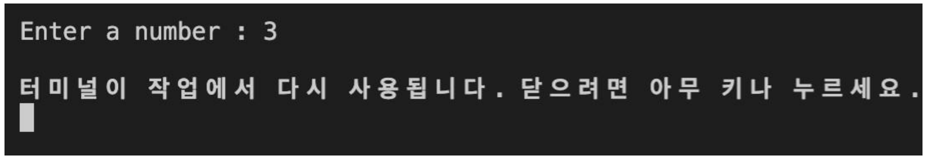
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int main(){
    double x;
    cout << "Enter a number : ";
    cin >> x;
    exit(1);

    cout << "You entered " << x;
}
```



```
Enter a number : 3
터미널 프로세스 "/Users/dglee/Desktop/C++ Projects/lab01_4_functions/exit"이(가)
종료되었습니다(종료 코드: 1).

터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

Any integer can be used, but 1 by convention

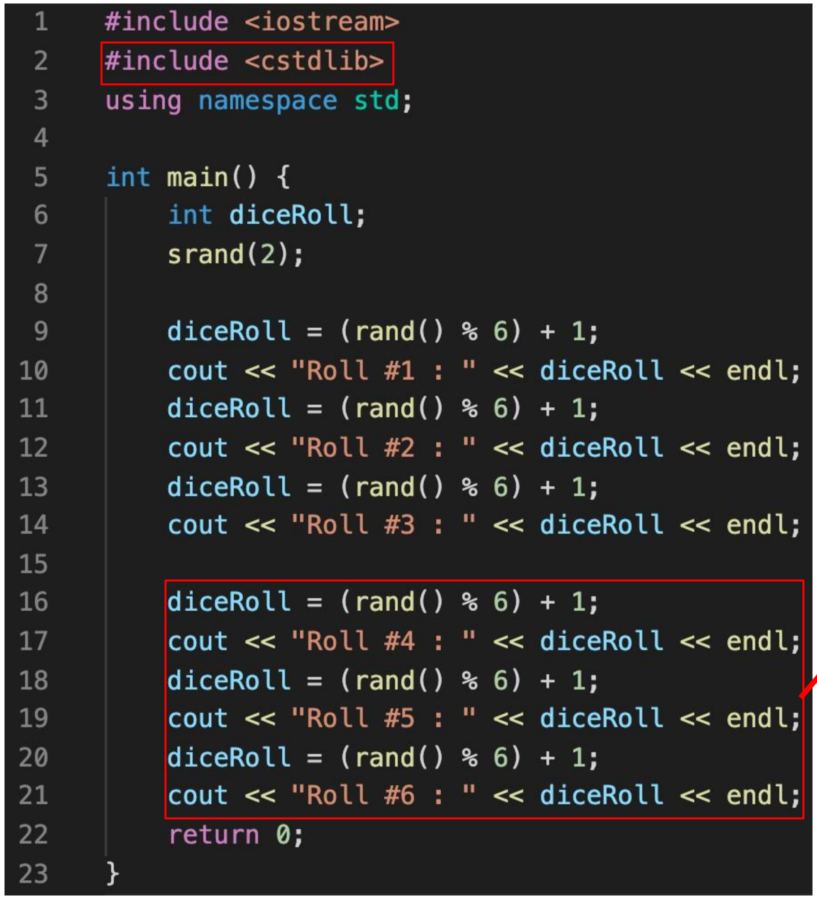
exit(0) ?



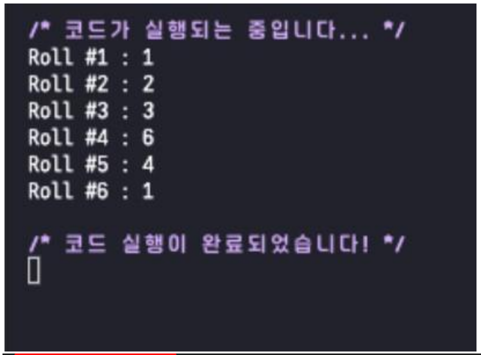
```
Enter a number : 3

터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

Predefined Functions

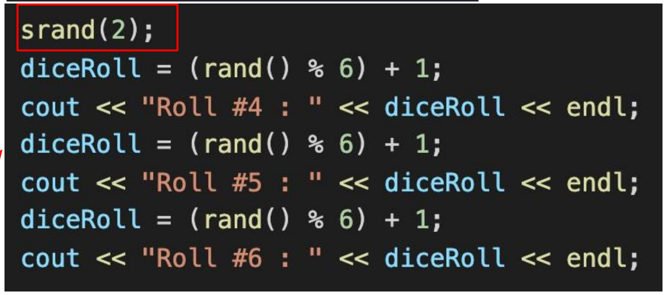
- Ex 5.“rand” and“srand” functions



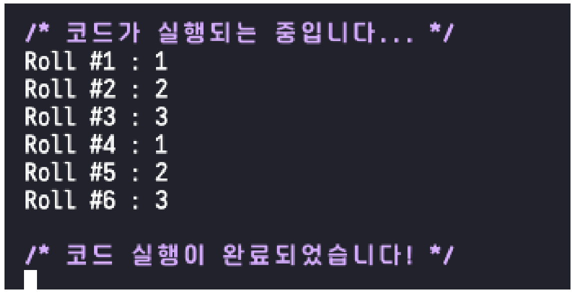
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
    int diceRoll;
    srand(2);

    diceRoll = (rand() % 6) + 1;
    cout << "Roll #1 : " << diceRoll << endl;
    diceRoll = (rand() % 6) + 1;
    cout << "Roll #2 : " << diceRoll << endl;
    diceRoll = (rand() % 6) + 1;
    cout << "Roll #3 : " << diceRoll << endl;

    diceRoll = (rand() % 6) + 1;
    cout << "Roll #4 : " << diceRoll << endl;
    diceRoll = (rand() % 6) + 1;
    cout << "Roll #5 : " << diceRoll << endl;
    diceRoll = (rand() % 6) + 1;
    cout << "Roll #6 : " << diceRoll << endl;
    return 0;
}
```



```
/* 코드가 실행되는 중입니다... */
Roll #1 : 1
Roll #2 : 2
Roll #3 : 3
Roll #4 : 6
Roll #5 : 4
Roll #6 : 1

/* 코드 실행이 완료되었습니다! */
```



```cpp
srand(2);
diceRoll = (rand() % 6) + 1;
cout << "Roll #4 : " << diceRoll << endl;
diceRoll = (rand() % 6) + 1;
cout << "Roll #5 : " << diceRoll << endl;
diceRoll = (rand() % 6) + 1;
cout << "Roll #6 : " << diceRoll << endl;
```



```
/* 코드가 실행되는 중입니다... */
Roll #1 : 1
Roll #2 : 2
Roll #3 : 3
Roll #4 : 1
Roll #5 : 2
Roll #6 : 3

/* 코드 실행이 완료되었습니다! */
```

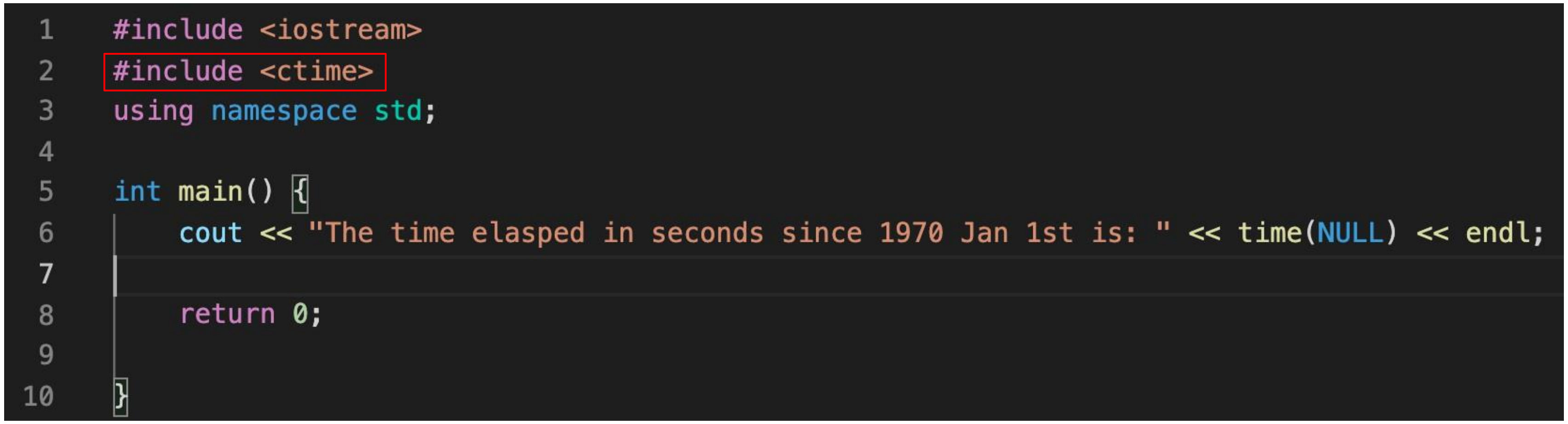
Predefined Functions

- Pseudo Random Number Generator (PRNG)
  - Random Number = (Start Number * a + b) / c
  - Next Random Number = (Random Number * a + b) /c
  - Goes on …

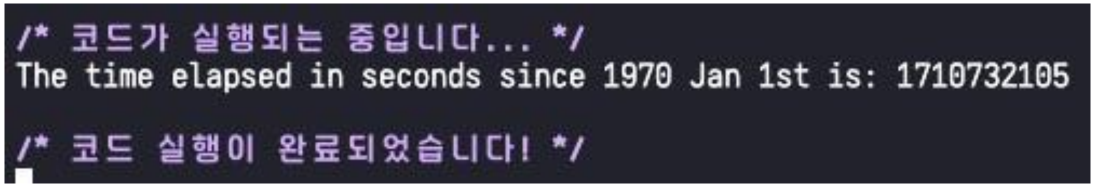
- Random seed corresponds to the "Start Number"

Predefined Functions

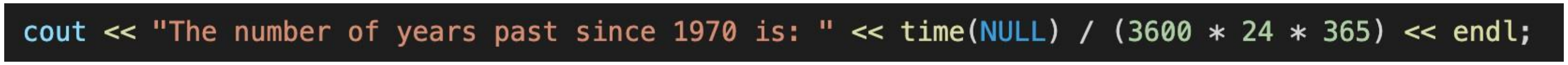
- Ex 6."time" function



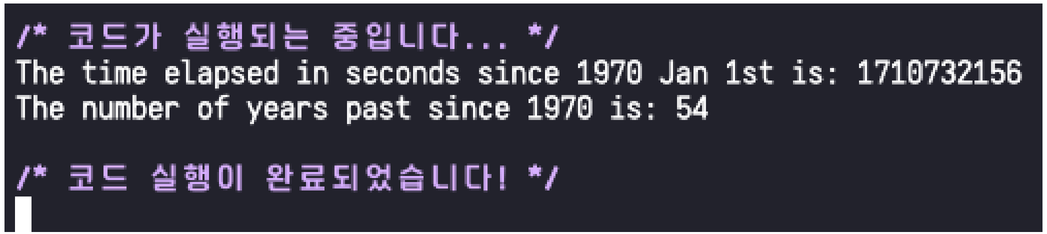
```cpp
#include <iostream>
#include <ctime>
using namespace std;

int main() {
    cout << "The time elasped in seconds since 1970 Jan 1st is: " << time(NULL) << endl;

    return 0;

}
```



```
/* 코드가 실행되는 중입니다... */
The time elapsed in seconds since 1970 Jan 1st is: 1710732105

/* 코드 실행이 완료되었습니다! */
```



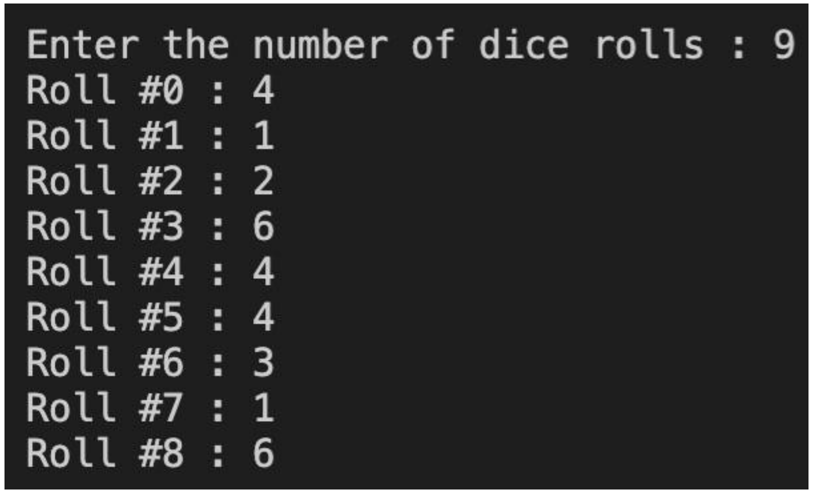
```cpp
cout << "The number of years past since 1970 is: " << time(NULL) / (3600 * 24 * 365) << endl;
```



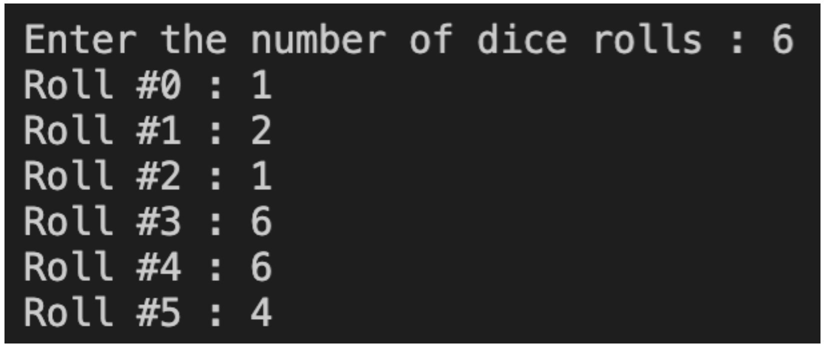
```
/* 코드가 실행되는 중입니다... */
The time elapsed in seconds since 1970 Jan 1st is: 1710732156
The number of years past since 1970 is: 54

/* 코드 실행이 완료되었습니다! */
```

Predefined Functions

- Exercise
  - Using "rand" and "time", create a dice rolling program that outputs differently every time.



```
Enter the number of dice rolls : 9
Roll #0 : 4
Roll #1 : 1
Roll #2 : 2
Roll #3 : 6
Roll #4 : 4
Roll #5 : 4
Roll #6 : 3
Roll #7 : 1
Roll #8 : 6
```



```
Enter the number of dice rolls : 6
Roll #0 : 1
Roll #1 : 2
Roll #2 : 1
Roll #3 : 6
Roll #4 : 6
Roll #5 : 4
```

Programmer-defined Functions

- Programmer-defined Functions
  - We can also define our own functions
  - The reason of defining custom functions
    - Divide & Conquer / Readability / Re-use / …

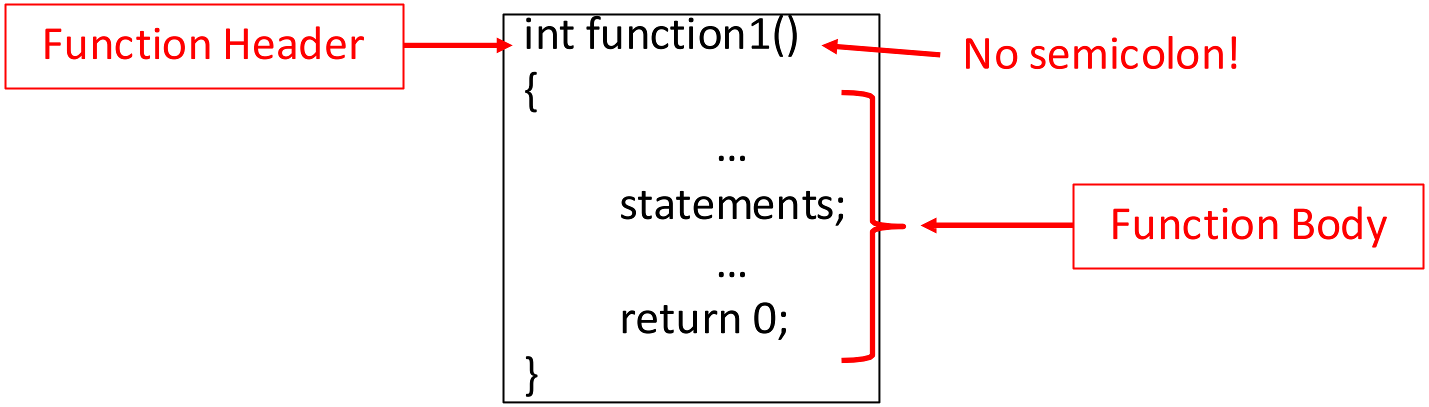
- Components of Function use
  - Function **Declaration/Prototype**
  - Function **Definition**
  - Function **Call**

Programmer-defined Functions

- Function Declaration/Prototype
  - An "informational" declaration for compiler
  - Syntax
    - <return_type> FunctionName(<parameter-list>);
  - Example
    - double totalCost(int number, double price);
  - Should be placed before any calls

Programmer-defined Functions

- Function Definition
  - Implementation of function (similar to implementing main())
  - Usually placed after (or before)   function main()

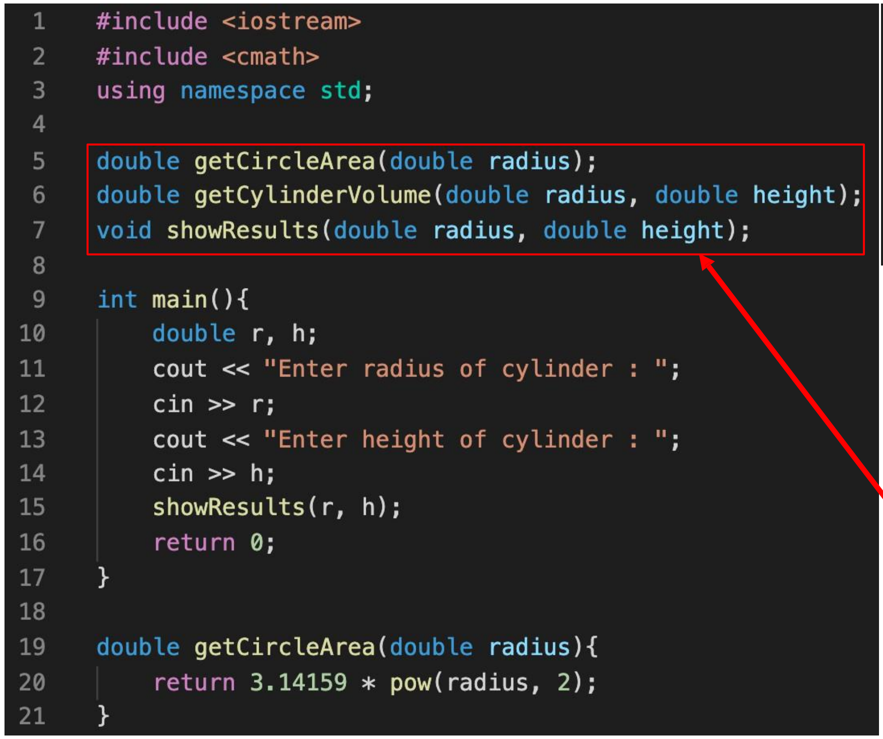


```
int function1()
{
            …
        statements;
            …
        return 0;
}
```

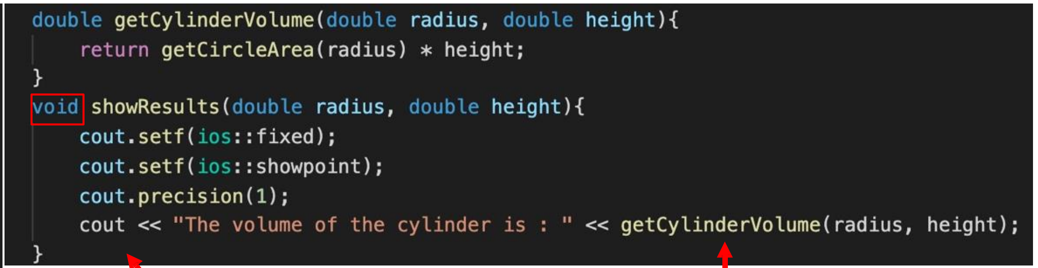
- Function Call
  - Same as calling predefined function

Programmer-defined Functions

- Ex 7. Functions that compute (return)
  - the area of of circle with given radius (parameter)
  - the volume of cylinder with given radius and height (parameter)



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

double getCircleArea(double radius);
double getCylinderVolume(double radius, double height);
void showResults(double radius, double height);

int main(){
    double r, h;
    cout << "Enter radius of cylinder : ";
    cin >> r;
    cout << "Enter height of cylinder : ";
    cin >> h;
    showResults(r, h);
    return 0;
}

double getCircleArea(double radius){
    return 3.14159 * pow(radius, 2);
}
```



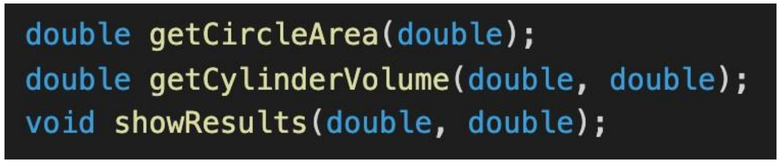
```cpp
double getCylinderVolume(double radius, double height){
    return getCircleArea(radius) * height;
}
void showResults(double radius, double height){
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);
    cout << "The volume of the cylinder is : " << getCylinderVolume(radius, height);
}
```

No return value

Function calls another function !

1) without declaration?

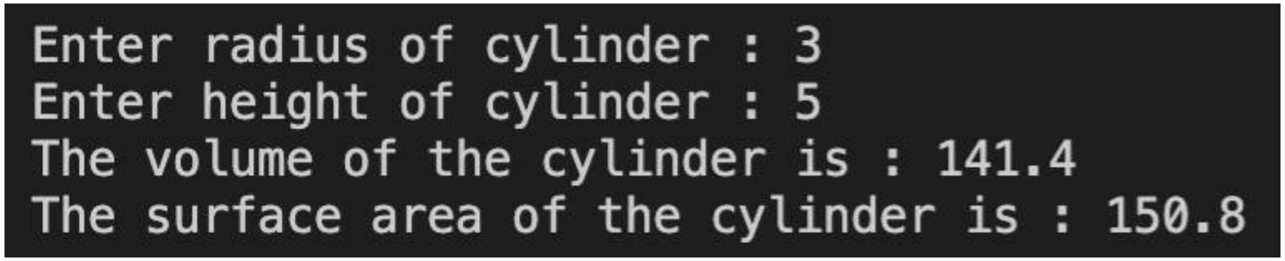
2) without Parameter name?



```cpp
double getCircleArea(double);
double getCylinderVolume(double, double);
void showResults(double, double);
```

Programmer-defined Functions

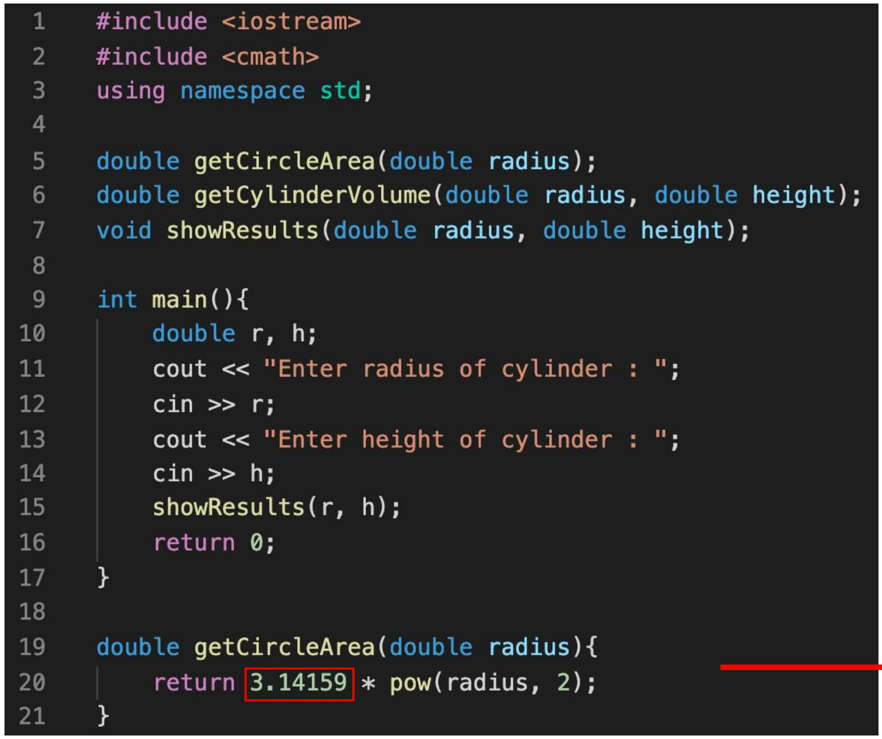
- Exercise.
  - Define two two more functions that compute
    - The circumference of circle as ***getCircleCircumference***
    - The surface area of cylinder as ***getCylinderSurfaceArea***
  - Program should output as follows



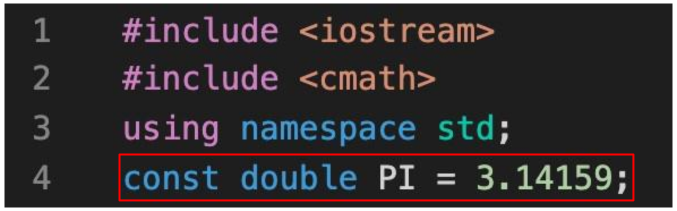
```
Enter radius of cylinder : 3
Enter height of cylinder : 5
The volume of the cylinder is : 141.4
The surface area of the cylinder is : 150.8
```

Programmer-defined Functions

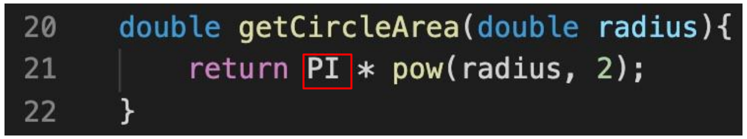
- Ex 7.(cont'd) Global declarations for constants
- – Declare globally so all functions can use it



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

double getCircleArea(double radius);
double getCylinderVolume(double radius, double height);
void showResults(double radius, double height);

int main(){
    double r, h;
    cout << "Enter radius of cylinder : ";
    cin >> r;
    cout << "Enter height of cylinder : ";
    cin >> h;
    showResults(r, h);
    return 0;
}

double getCircleArea(double radius){
    return 3.14159 * pow(radius, 2);
}
```



```cpp
#include <iostream>
#include <cmath>
using namespace std;
const double PI = 3.14159;
```



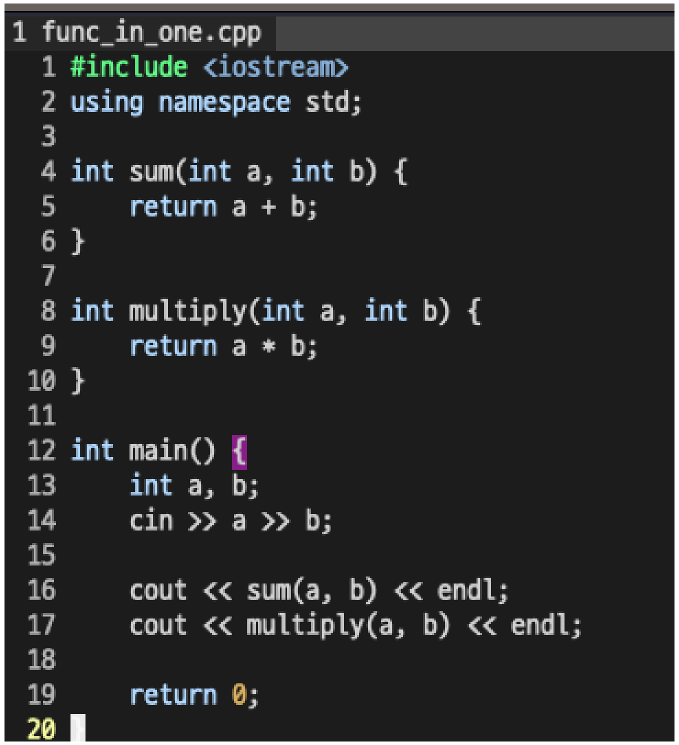
```cpp
double getCircleArea(double radius){
    return PI * pow(radius, 2);
}
```

Programmer-defined Functions

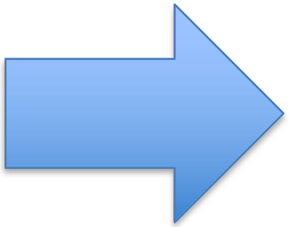
- Separating **declaration & definition** of function
  - For concise, efficient programming
  - For easy debugging

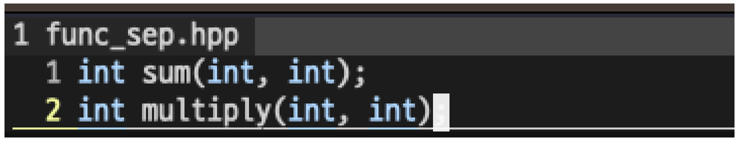
- Declaring prototypes of functions in ".hpp" fille
- Write definition of each function in ".cpp" file

Programmer-defined Functions

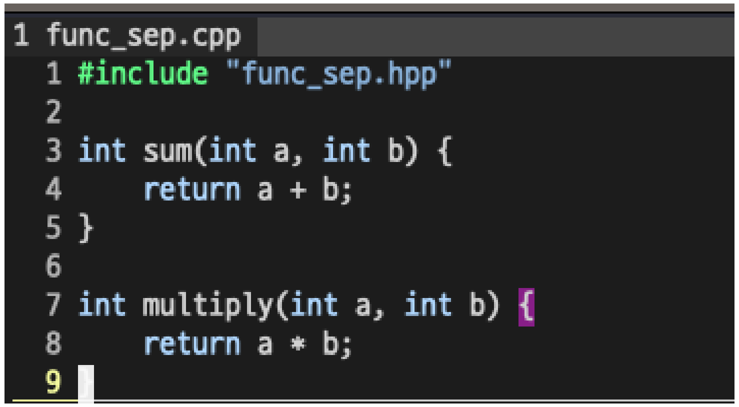


```
1 func_in_one.cpp
#include <iostream>
using namespace std;

int sum(int a, int b) {
    return a + b;
}

int multiply(int a, int b) {
    return a * b;
}

int main() {
    int a, b;
    cin >> a >> b;

    cout << sum(a, b) << endl;
    cout << multiply(a, b) << endl;

    return 0;
}
```

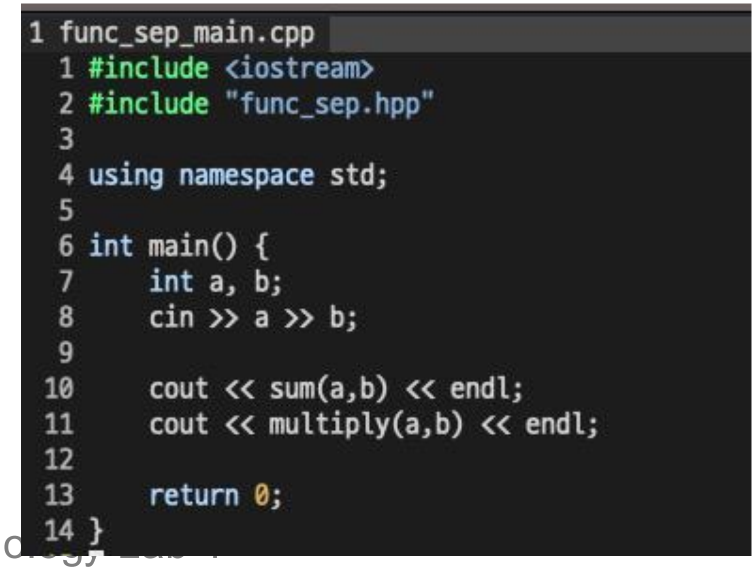




```
1 func_sep.hpp
int sum(int, int);
int multiply(int, int);
```

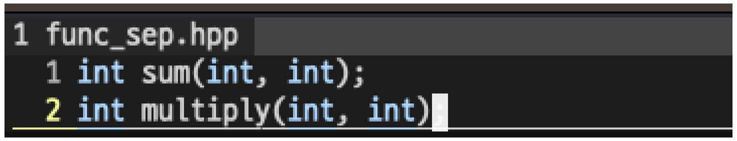


```
1 func_sep.cpp
#include "func_sep.hpp"

int sum(int a, int b) {
    return a + b;
}

int multiply(int a, int b) {
    return a * b;
}
```



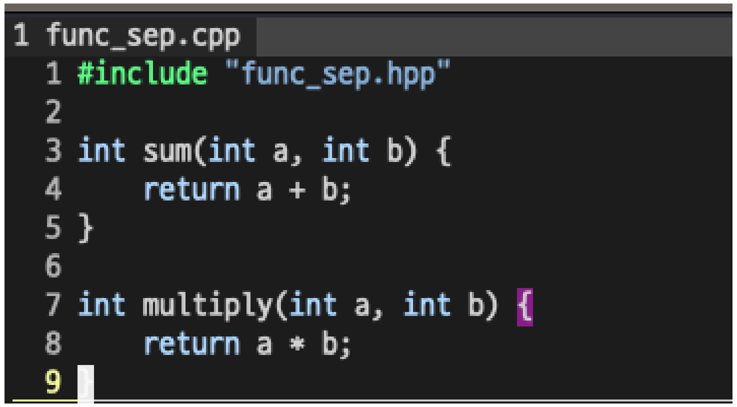
```
1 func_sep_main.cpp
#include <iostream>
#include "func_sep.hpp"

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;

    cout << sum(a,b) << endl;
    cout << multiply(a,b) << endl;

    return 0;
}
```

Programmer-defined Functions



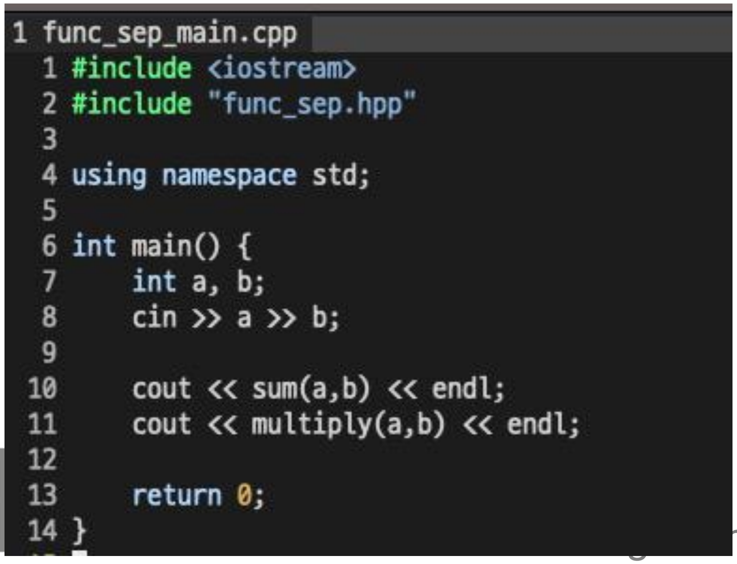
```
1 func_sep.hpp
int sum(int, int);
int multiply(int, int);
```

Declaration in "func_sep.hpp"



```
1 func_sep.cpp
#include "func_sep.hpp"

int sum(int a, int b) {
    return a + b;
}

int multiply(int a, int b) {
    return a * b;
}
```

Definition in "func_sep.cpp"



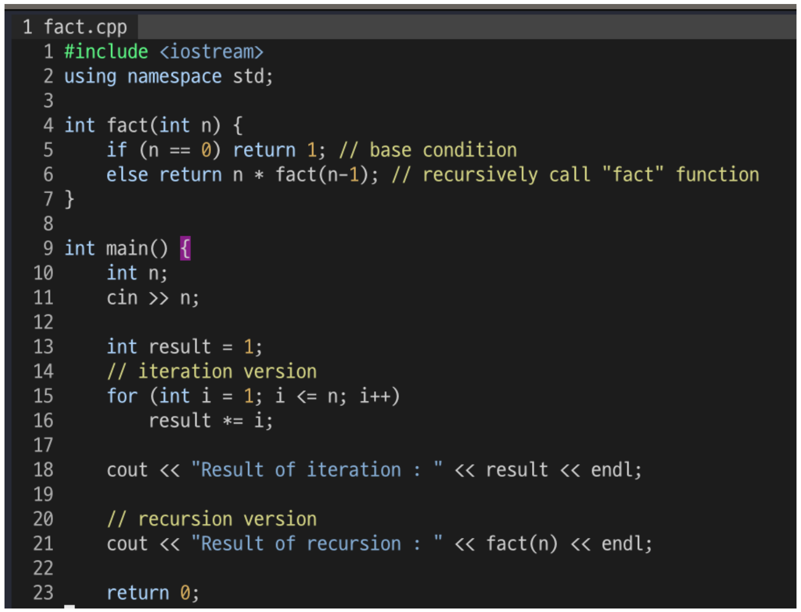
```
1 func_sep_main.cpp
#include <iostream>
#include "func_sep.hpp"

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;

    cout << sum(a,b) << endl;
    cout << multiply(a,b) << endl;

    return 0;
}
```

Include "func_sep.hpp" header &
call functions

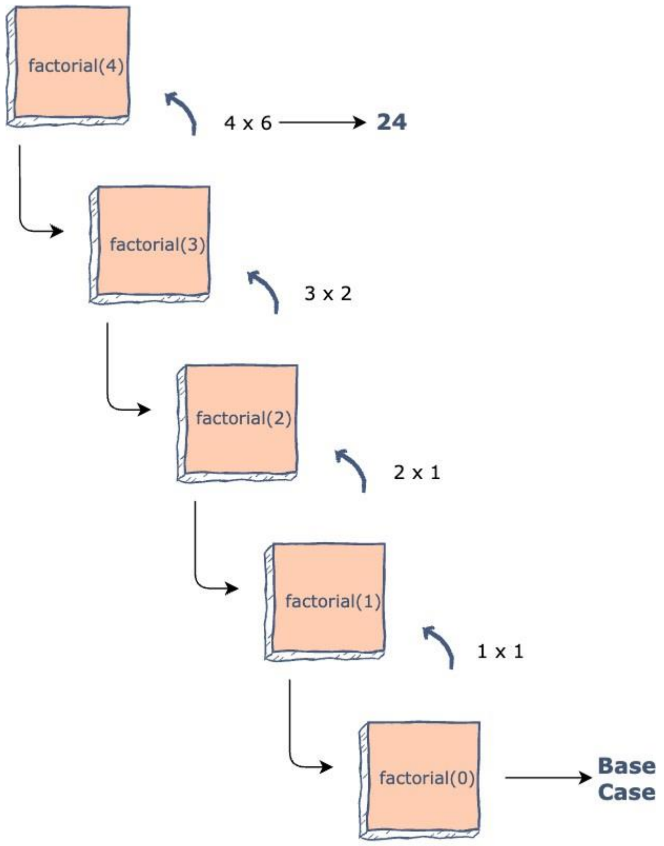
cf)
Compile w/
g++ -std=c++11 func_sep_main.cpp func_sep_cpp –o main

Iteration vs Recursion

- Ex 8. Calculation of n!



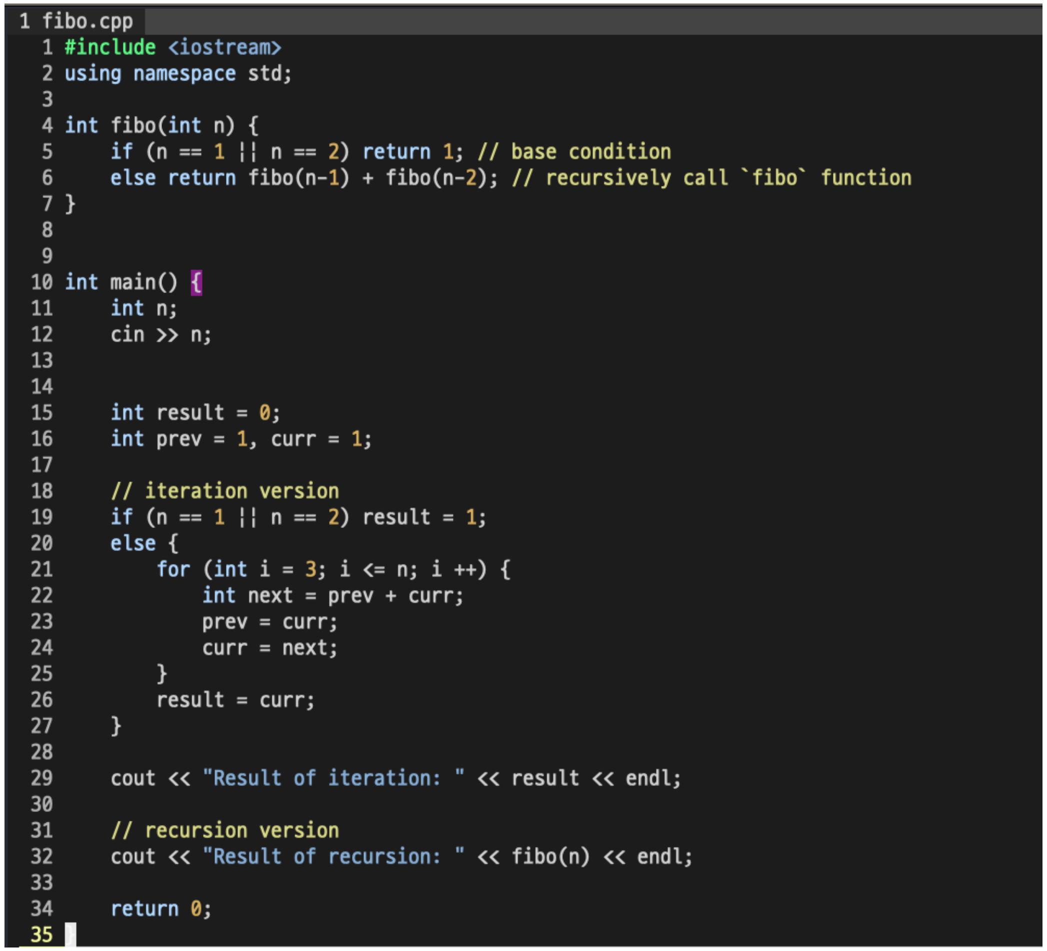
```
1 fact.cpp
#include <iostream>
using namespace std;

int fact(int n) {
    if (n == 0) return 1; // base condition
    else return n * fact(n-1); // recursively call "fact" function
}

int main() {
    int n;
    cin >> n;

    int result = 1;
    // iteration version
    for (int i = 1; i <= n; i++)
        result *= i;

    cout << "Result of iteration : " << result << endl;

    // recursion version
    cout << "Result of recursion : " << fact(n) << endl;

    return 0;
```



Factorial calculation using recursion

Iteration vs Recursion

- Ex 9. Calculation of nth Fibonacci number



1 fibo.cpp

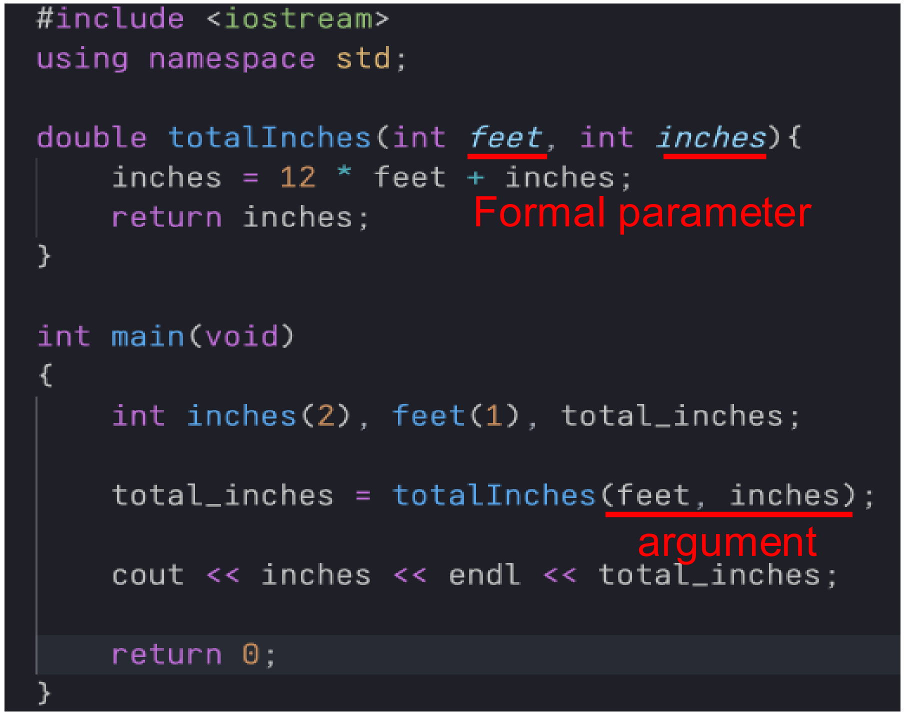
```cpp
#include <iostream>
using namespace std;

int fibo(int n) {
    if (n == 1 || n == 2) return 1; // base condition
    else return fibo(n-1) + fibo(n-2); // recursively call `fibo` function
}


int main() {
    int n;
    cin >> n;


    int result = 0;
    int prev = 1, curr = 1;

    // iteration version
    if (n == 1 || n == 2) result = 1;
    else {
        for (int i = 3; i <= n; i ++) {
            int next = prev + curr;
            prev = curr;
            curr = next;
        }
        result = curr;
    }

    cout << "Result of iteration: " << result << endl;

    // recursion version
    cout << "Result of recursion: " << fibo(n) << endl;

    return 0;

```

Iteration vs Recursion

- Pros of Iteration:
  - More efficient than recursion in terms of speed and memory usage in some cases.
- Pros of Recursion:
  - Can simplify the code for complex problems such as tree-like data structures or nested relationships

Parameters, Arguments

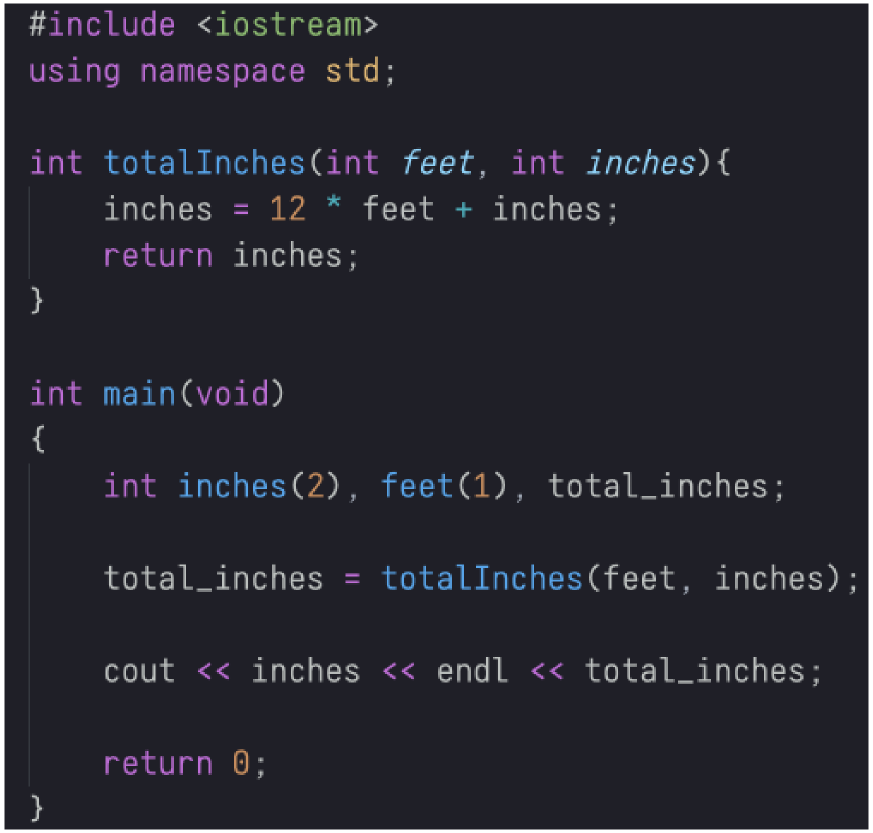
- Parameters, arguments
  - Formal parameter : parameters listed in the function declaration
  - Argument : values passed to the formal parameters



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

double totalInches(int feet, int inches){
    inches = 12 * feet + inches;
    return inches;
}

int main(void)
{
    int inches(2), feet(1), total_inches;

    total_inches = totalInches(feet, inches);

    cout << inches << endl << total_inches;

    return 0;
}
```

Call-by-value

- Ex 10. Call-by-value copies to a local variable
  - Type the code below and guess the result

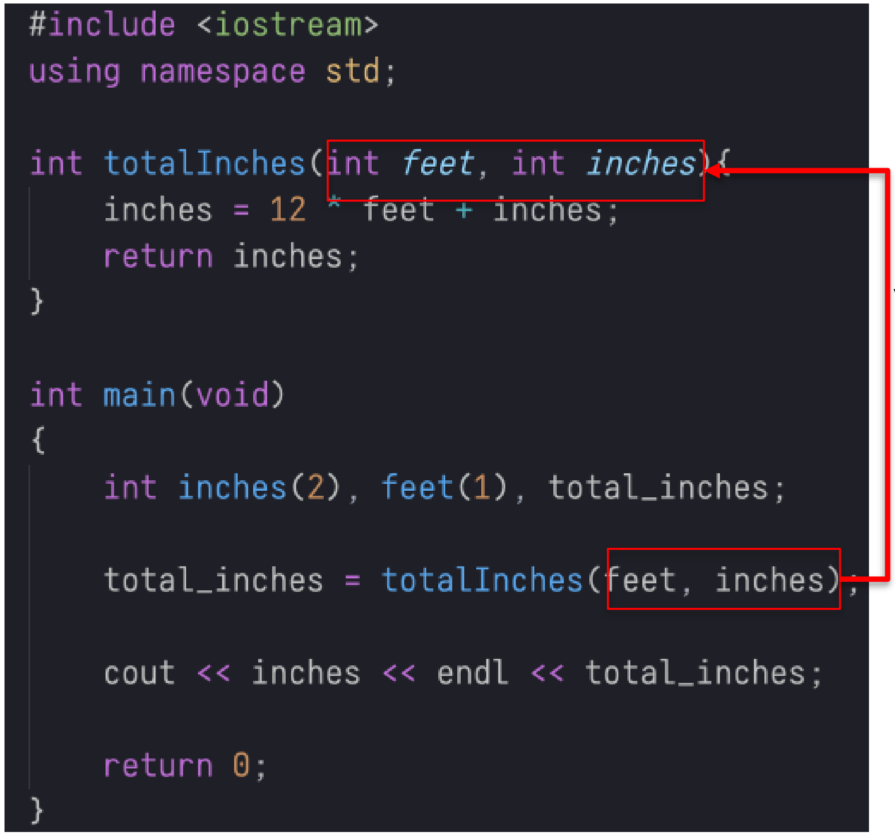


```
#include <iostream>
using namespace std;

int totalInches(int feet, int inches){
    inches = 12 * feet + inches;
    return inches;
}

int main(void)
{
    int inches(2), feet(1), total_inches;

    total_inches = totalInches(feet, inches);

    cout << inches << endl << total_inches;

    return 0;
}
```

Machine Intelligence & Data science LAB

Call-by-value

- Ex 10. Call-by-value copies to a local variable
  - What if we want to modify the argument?

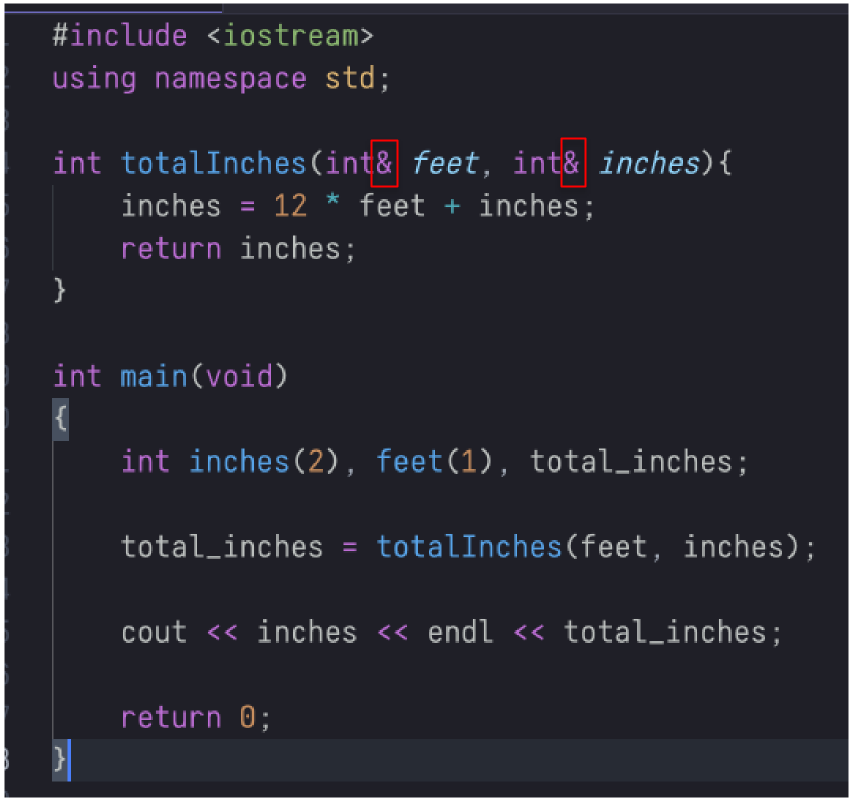


```
#include <iostream>
using namespace std;

int totalInches(int feet, int inches){
    inches = 12 * feet + inches;
    return inches;
}

int main(void)
{
    int inches(2), feet(1), total_inches;

    total_inches = totalInches(feet, inches);

    cout << inches << endl << total_inches;

    return 0;
}
```

Values of the arguments are copied, new local variables are created

Call-by-reference

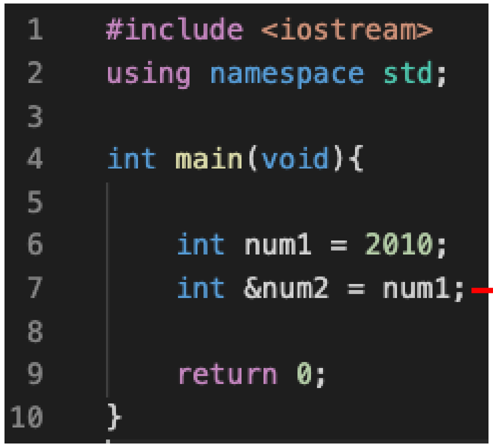
- Ex 10 (cont'd). Call-by-reference passes the variable itself
    - Type code below and guess the result



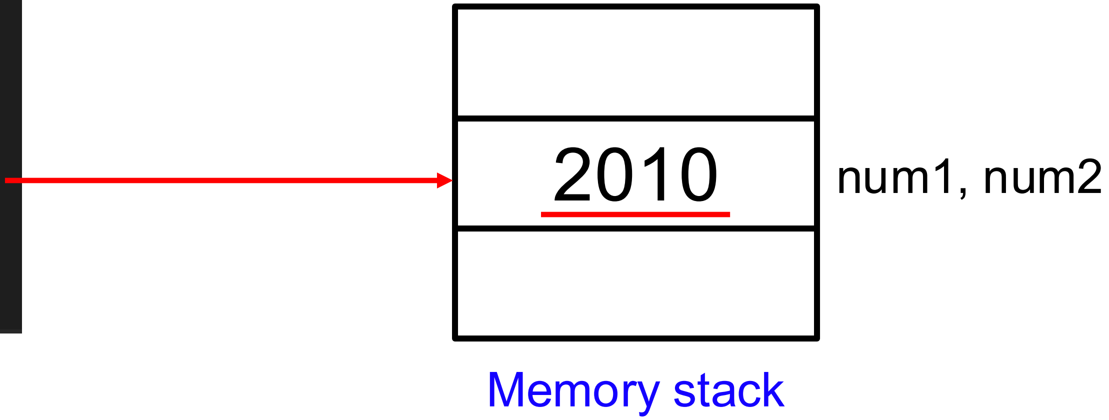
```
#include <iostream>
using namespace std;

int totalInches(int& feet, int& inches){
    inches = 12 * feet + inches;
    return inches;
}

int main(void)
{
    int inches(2), feet(1), total_inches;

    total_inches = totalInches(feet, inches);

    cout << inches << endl << total_inches;

    return 0;
}
```

Reference

- Reference variable
  - Same as giving another name for an already existing variable
  - It must be initialized with a variable



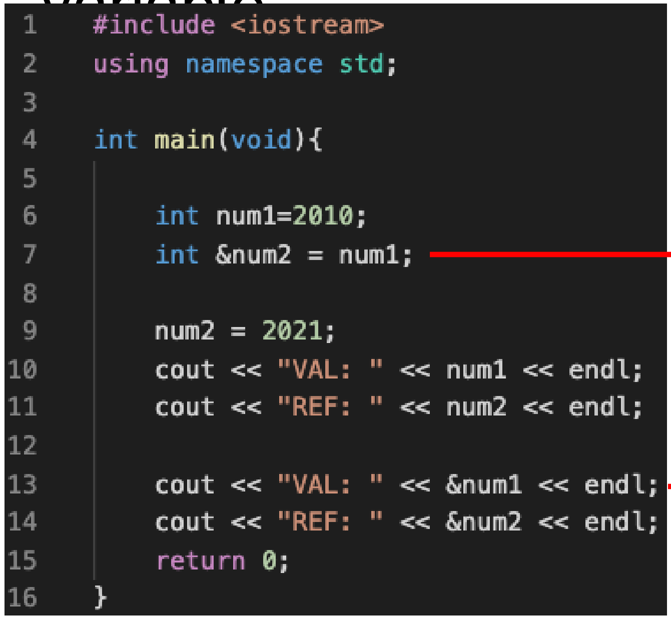
```
#include <iostream>
using namespace std;

int main(void){

    int num1 = 2010;
    int &num2 = num1;

    return 0;
}
```



https://www.tutorialspoint.com/cplusplus/cpp_references.htm

& Operator

- Ex 11. & operator has 2 different uses
  - Declare reference if used during <u>variable declaration</u>
  - Returns address if used in front of an <u>already declared</u> variable



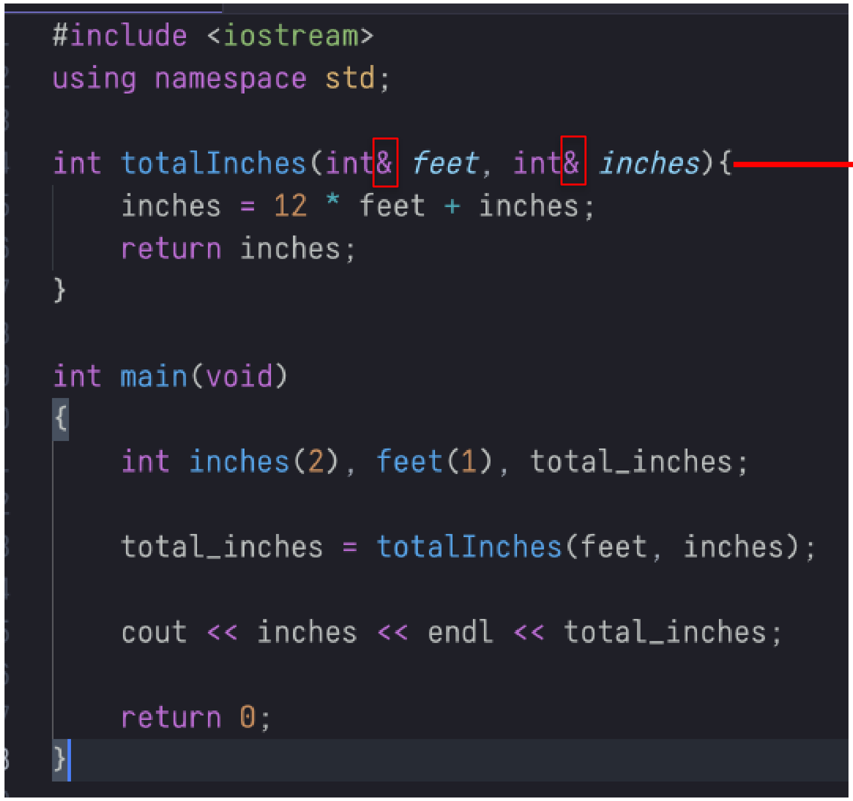
```cpp
#include <iostream>
using namespace std;

int main(void){

    int num1=2010;
    int &num2 = num1;

    num2 = 2021;
    cout << "VAL: " << num1 << endl;
    cout << "REF: " << num2 << endl;

    cout << "VAL: " << &num1 << endl;
    cout << "REF: " << &num2 << endl;
    return 0;
}
```

Declare reference

Returns address

& Operator

- Ex 10 (cont'd). & operator has 2 different uses
  - Declare reference if used during variable declaration

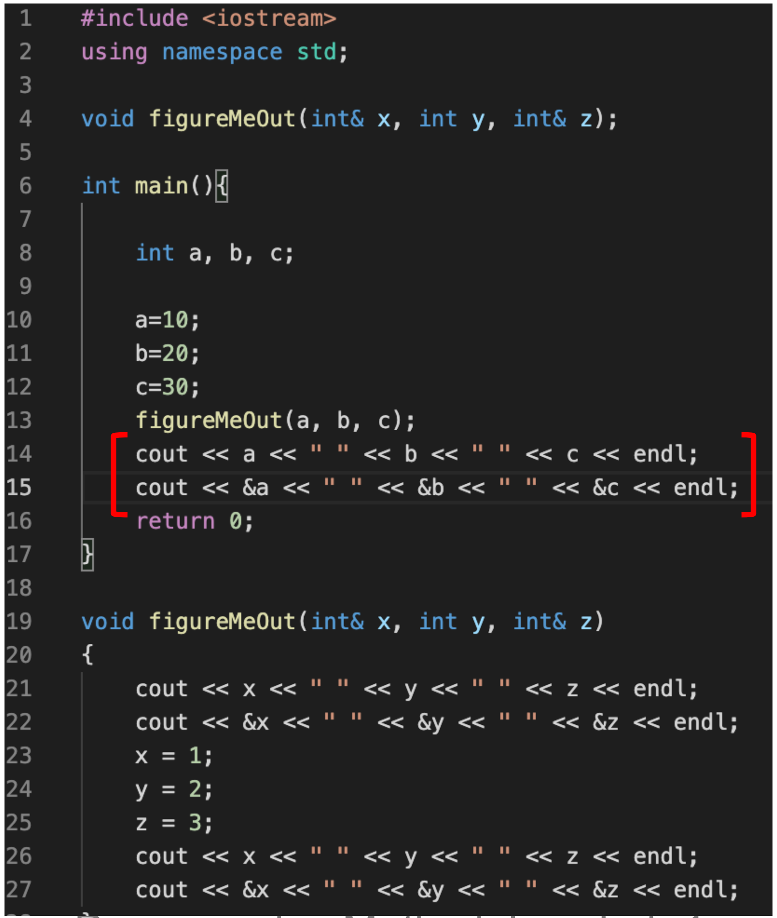


```
#include <iostream>
using namespace std;

int totalInches(int& feet, int& inches){
    inches = 12 * feet + inches;
    return inches;
}

int main(void)
{
    int inches(2), feet(1), total_inches;

    total_inches = totalInches(feet, inches);

    cout << inches << endl << total_inches;

    return 0;
}
```

Declare reference

Call-by-value vs Call-by-reference

- Ex 12. Call-by-value vs call-by-reference
  - Guess the outputs and then see the results.

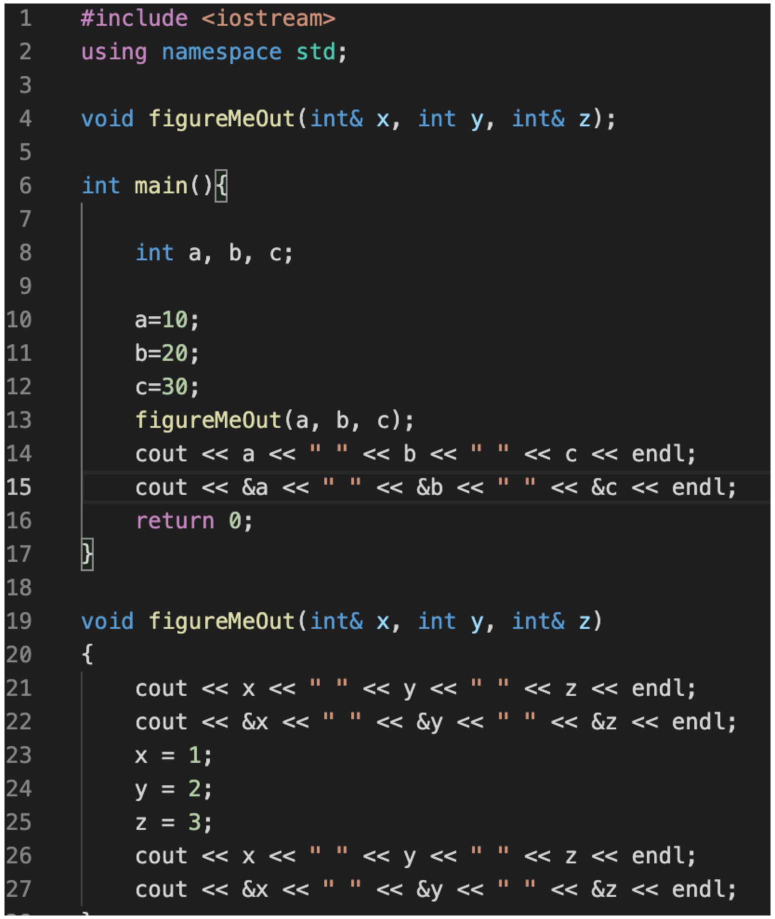


```
#include <iostream>
using namespace std;

void figureMeOut(int& x, int y, int& z);

int main(){

    int a, b, c;

    a=10;
    b=20;
    c=30;
    figureMeOut(a, b, c);
    cout << a << " " << b << " " << c << endl;
    cout << &a << " " << &b << " " << &c << endl;
    return 0;
}

void figureMeOut(int& x, int y, int& z)
{
    cout << x << " " << y << " " << z << endl;
    cout << &x << " " << &y << " " << &z << endl;
    x = 1;
    y = 2;
    z = 3;
    cout << x << " " << y << " " << z << endl;
    cout << &x << " " << &y << " " << &z << endl;
```

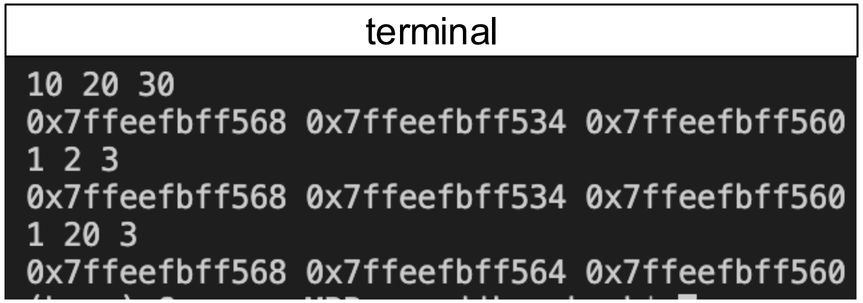
Machine Intelligence & Data science LAB

Call-by-value vs Call-by-reference

- Ex 12. Call-by-value vs call-by-reference



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

void figureMeOut(int& x, int y, int& z);

int main(){

    int a, b, c;

    a=10;
    b=20;
    c=30;
    figureMeOut(a, b, c);
    cout << a << " " << b << " " << c << endl;
    cout << &a << " " << &b << " " << &c << endl;
    return 0;
}

void figureMeOut(int& x, int y, int& z)
{
    cout << x << " " << y << " " << z << endl;
    cout << &x << " " << &y << " " << &z << endl;
    x = 1;
    y = 2;
    z = 3;
    cout << x << " " << y << " " << z << endl;
    cout << &x << " " << &y << " " << &z << endl;
```



terminal

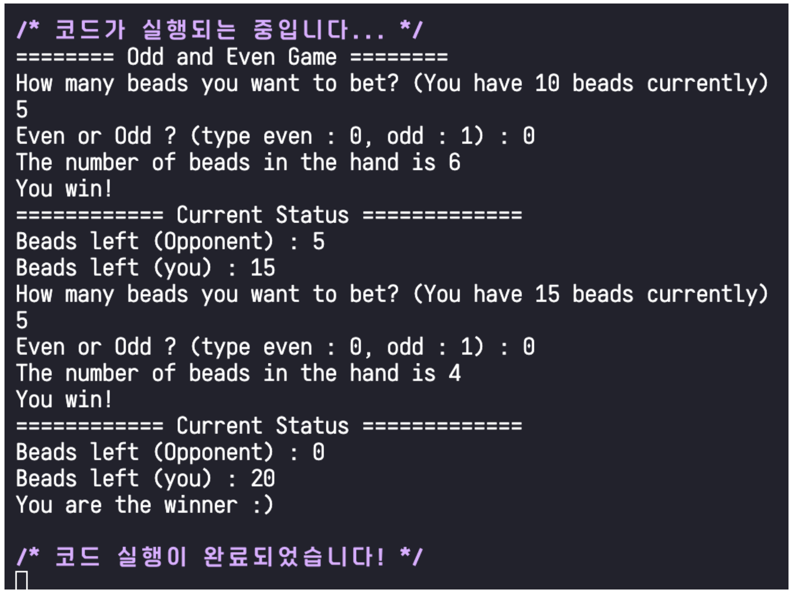
```
10 20 30
0x7ffeefbff568 0x7ffeefbff534 0x7ffeefbff560
1 2 3
0x7ffeefbff568 0x7ffeefbff534 0x7ffeefbff560
1 20 3
0x7ffeefbff568 0x7ffeefbff564 0x7ffeefbff560
```

Call-by-value creates a local variable!
(value of variable b stays unchanged
)

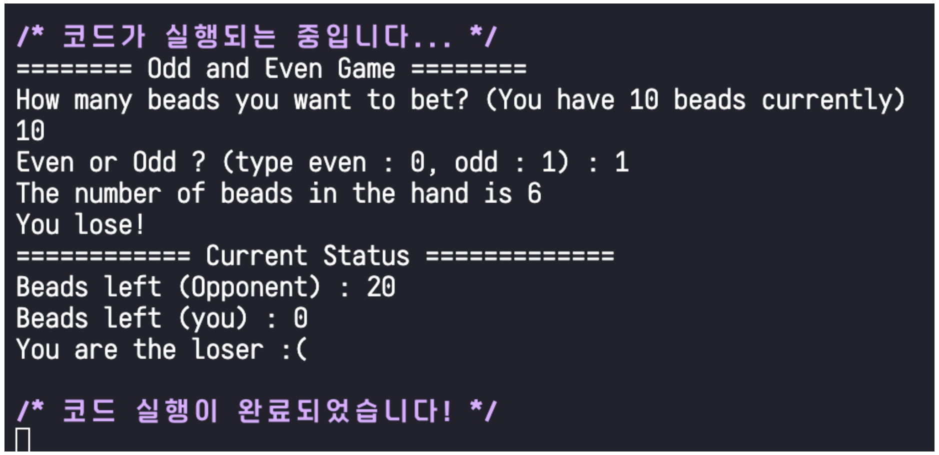
For more information, check out:
https://linuxhint.com/call-by-address-and-call-by-reference-cpp/

Assignment

- Odd and Even game
  - Game objective :The opponent have the beads in the his or her hand randomly, then you guess whether the number of beads in the opponent's hands is even or odd



```
/* 코드가 실행되는 중입니다... */
======== Odd and Even Game ========
How many beads you want to bet? (You have 10 beads currently)
5
Even or Odd ? (type even : 0, odd : 1) : 0
The number of beads in the hand is 6
You win!
============ Current Status =============
Beads left (Opponent) : 5
Beads left (you) : 15
How many beads you want to bet? (You have 15 beads currently)
5
Even or Odd ? (type even : 0, odd : 1) : 0
The number of beads in the hand is 4
You win!
============ Current Status =============
Beads left (Opponent) : 0
Beads left (you) : 20
You are the winner :)

/* 코드 실행이 완료되었습니다! */
```



```
/* 코드가 실행되는 중입니다... */
======== Odd and Even Game ========
How many beads you want to bet? (You have 10 beads currently)
10
Even or Odd ? (type even : 0, odd : 1) : 1
The number of beads in the hand is 6
You lose!
============ Current Status =============
Beads left (Opponent) : 20
Beads left (you) : 0
You are the loser :(

/* 코드 실행이 완료되었습니다! */
```

Machine Intelligence & Data science LAB

Assignment

- Odd and Even game
  - You bet your beads as you want, then
    1) take the opponent's beads as many as you bet when you guess correctly
    2) give the beads you bet to the opponent

  - The game continues until either the opponent or you have no bead at all (*oddEvengame function*)

Assignment

- Skeleton code description
  - *oddEvenGame* should call each of *~Phase* and *showStatus*
  - *betPhase*
    - Parameter : (int) the number of your current beads
    - Return value : (int) the number of beads you want to bet
  - *opponentPhase (given)*
    - Parameter : (int) the number of opponent's current beads
    - Return value : (int) a random number of beads from 0 to the number of opponent's current beads
  - *guessPhase*
    - Parameter : x
    - Return value : (int) 0 or 1 (the user should type either 0 for the even or 1 for the odd and return it)

Assignment

- Skeleton code description
  - *decidePhase*
    - Parameter : (int, int) the number of beads in hand (the integer from the *opponentPhase*) and your guessing (the integer (0 or 1) from the *guessPhase*)
    - Return value : (bool) true if your guess is correct, false otherwise
  - *showStatus (given)*
    - Parameter : (int, int) the number of you and the opponent's beads
    - Return value : x (just print out the current number of you and the opponent's beads)
  - Caution!
    - Do not add additional rand(), srand() function!
    - You don't need to consider the case where you bet more beads than the better or the opponent has.

Assignment

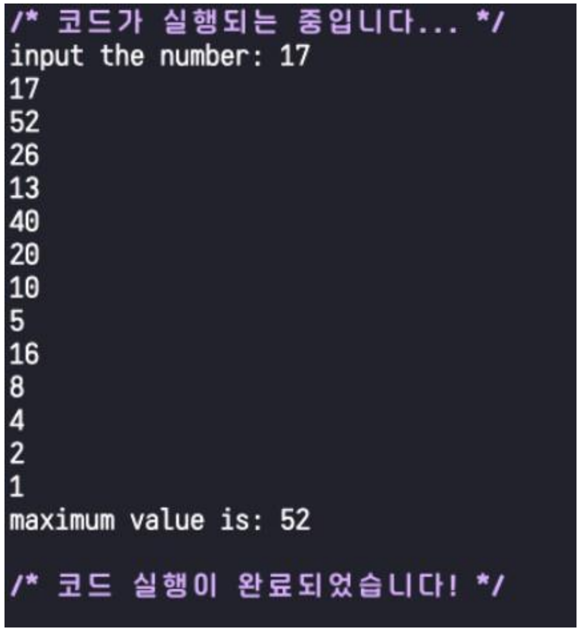
- Submission
  - Due date : 9/26, 14:30 (before the start of next lab 1 session)

Attendance Check

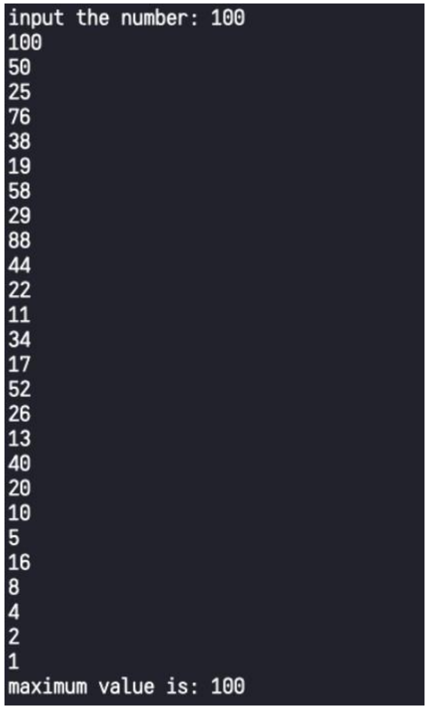
- Collatz conjecture
  - Repeating simple operations will eventually transform every natural number into 1
  - $f(n) = \begin{cases} \frac{n}{2} & if\ n\ is\ even \\ 3n+1 & if\ n\ is\ odd \end{cases}$
  - Ex) if initial value of n is 17 then,
  - 17 -> 52 -> 26 -> 13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1

Attendance Check

- Write a code which
  - Prints the sequence of dropping n to 1
  - Prints maximum value in the sequence



```
/* 코드가 실행되는 중입니다... */
input the number: 17
17
52
26
13
40
20
10
5
16
8
4
2
1
maximum value is: 52

/* 코드 실행이 완료되었습니다! */
```



```
input the number: 100
100
50
25
76
38
19
58
29
88
44
22
11
34
17
52
26
13
40
20
10
5
16
8
4
2
1
maximum value is: 100
```

Machine Intelligence & Data science LAB