

5장. 함수



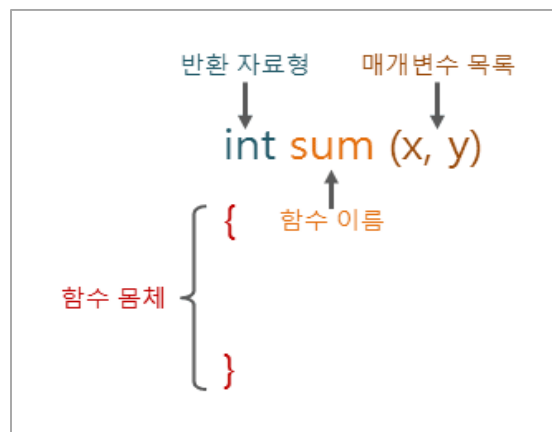
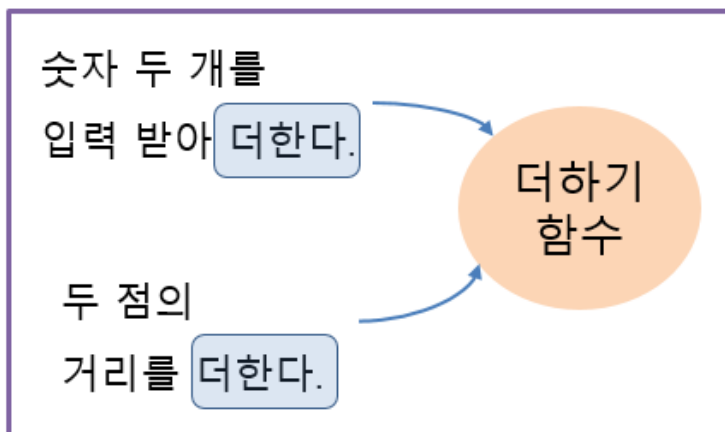
Visualstudio 2019



함수(function)

❖ 함수란?

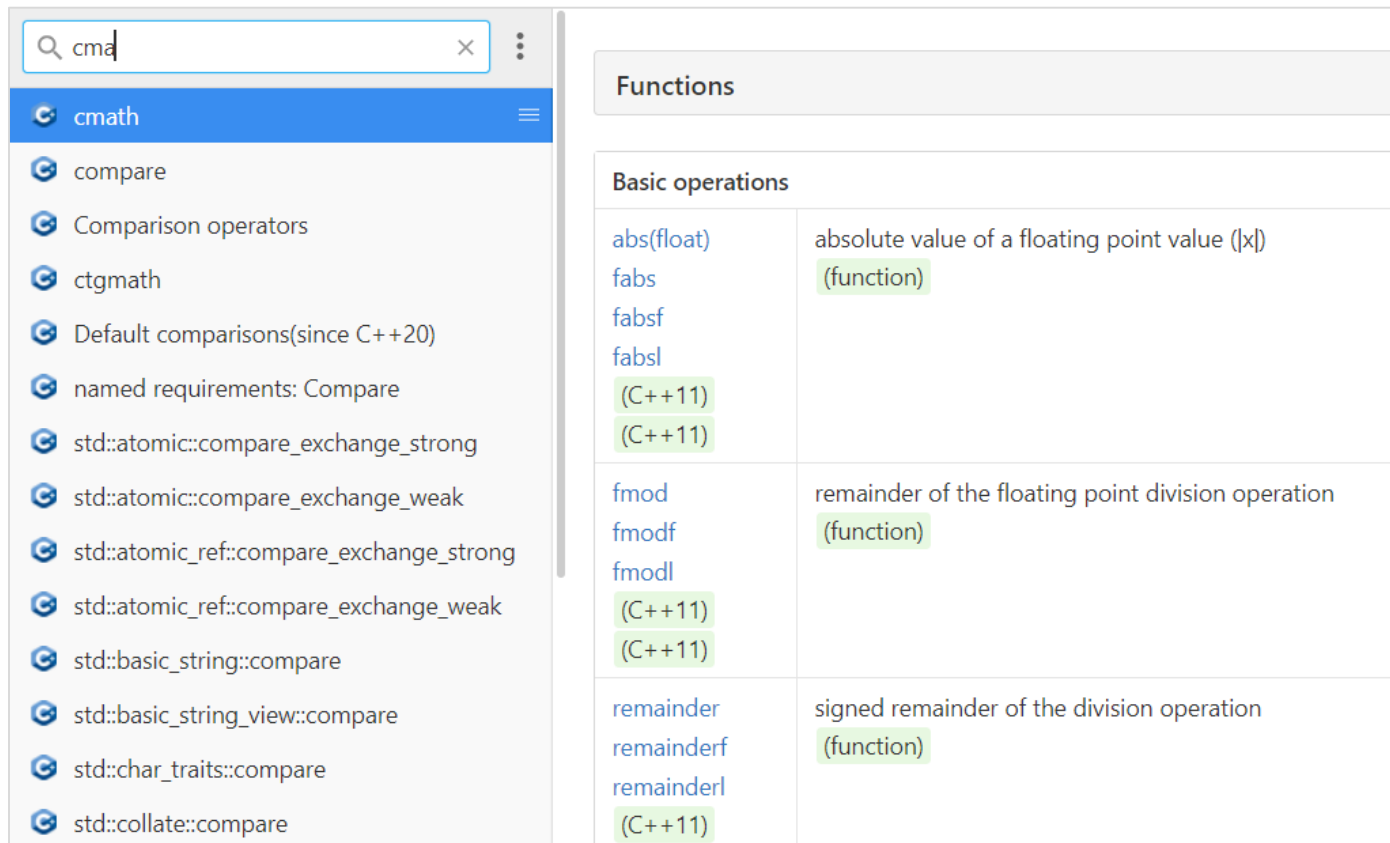
- 하나의 기능을 수행하는 일련의 코드이다.
- 함수는 이름이 있고, 반환값과 매개변수가 있다.(없는 경우도 있음)
- 하나의 큰 프로그램을 작은 부분들로 분리하여 코드의 중복을 최소화하고, 개발은 물론 코드의 수정이나 유지보수를 쉽게 한다.



표준 라이브러리 함수(function)

❖ 내장 함수 – 표준 라이브러리 함수

C++ 언어 Document: <https://devdocs.io/c++>



Search:

- cmath
- compare
- Comparison operators
- ctgmath
- Default comparisons(since C++20)
- named requirements: Compare
- std::atomic::compare_exchange_strong
- std::atomic::compare_exchange_weak
- std::atomic_ref::compare_exchange_strong
- std::atomic_ref::compare_exchange_weak
- std::basic_string::compare
- std::basic_string_view::compare
- std::char_traits::compare
- std::collate::compare

Functions

Basic operations

abs(float) fabs fabsf fabsl (C++11) (C++11)	absolute value of a floating point value (x) (function)
fmod fmodf fmodl (C++11) (C++11)	remainder of the floating point division operation (function)
remainder remainderf remainderl (C++11)	signed remainder of the division operation (function)

표준 라이브러리 함수(function)

❖ 헤더 파일 위치

로컬 디스크 (C:) > Program Files (x86) > Microsoft Visual Studio > 2019 > Community > VC > Tools > MSVC > 14.24.28314 > include

```
#pragma once
#ifndef _CSTDIO_
#define _CSTDIO_
#include <yvals_core.h>
#if _STL_COMPILER_PREPROCESSOR

#include <stdio.h>

#pragma pack(push, _CRT_PACKING)
#pragma warning(push, _STL_WARNING_LEVEL)
#pragma warning(disable : _STL_DISABLED_WARNINGS)
_STL_DISABLE_CLANG_WARNINGS
#pragma push_macro("new")
#undef new

// undef common macro overrides
#undef clearerr
#undef feof
#undef ferror
#undef fgetc
```



표준 라이브러리 함수(function)

❖ 표준 라이브러리 함수

- 수학 관련 함수 – cmath를 include 해야 함

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

int main() {

    //반올림
    cout << round(2.54) << endl;
    cout << round(2.34) << endl;
    cout << round(-2.54) << endl;
    cout << round(-2.34) << endl;

    //버림
    cout << floor(11.7) << endl;
    cout << floor(-11.7) << endl;
```

```
//절대값
cout << abs(-8) << endl;
cout << abs(8) << endl;

//거듭제곱
cout << pow(2, 4) << endl;
cout << pow(4, 2) << endl;

return 0;
}
```



표준 라이브러리 함수(function)

- 시간 관련 함수 – time.h를 include 해야 함

```
#include <iostream>
#include <ctime>
#include <Windows.h>
using namespace std;
```

```
long start, end;

start = time(NULL);
//cout << start << "초" << endl;

for (int i = 0; i < 100; i++) {
    cout << i << "번째 for문" << endl;
    Sleep(100); //0.1초
}

end = time(NULL);

long et = end - start;
cout << "for문 수행 시간 : " << et << "초";
```



표준 라이브러리 함수(function)

- rand() 함수 – srand() 함수 필요함

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

int main() {

    int dice;

    //srand(11); //seed 배정
    srand((unsigned)time(NULL));
    cout << "무작위 수 : " << rand() << endl;

    //주사위 10번 던지기
    for (int i = 0; i < 10; i++) {
        dice = rand() % 6 + 1;
        cout << dice << endl;
    }

    return 0;
}
```



조건문(if문)

▪ UpAndDown 게임(숫자 맞추기 게임)

- 게임이 시작되면 컴퓨터는 난수를 생성하여 저장한다.
- 게이머가 숫자를 예측하면 컴퓨터는 답과 비교하여 '너무 커요!', '너무 작아요.', '정답입니다' 를 출력한다.
- 총 회수는 10회이며, 점수는 남은 회수 x 10으로 계산한다.

```
[1회] 1 ~ 100 사이의 값 예측 -> 50  
너무 작아요!  
[2회] 50 ~ 100 사이의 값 예측 -> 70  
너무 작아요!  
[3회] 70 ~ 100 사이의 값 예측 -> 90  
너무 커요!  
[4회] 70 ~ 90 사이의 값 예측 -> 80  
너무 작아요!  
[5회] 80 ~ 90 사이의 값 예측 -> 85  
너무 작아요!  
[6회] 85 ~ 90 사이의 값 예측 -> 87  
정답입니다.  
정답 : 87  
최종 점수 : 50
```


조건문(if문)

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

int main() {

    int com, guess, i, min = 1, max = 100;

    srand((unsigned)time(NULL));
    com = rand() % 100 + 1;    //컴퓨터에 임의의 수 기억
```



조건문(if문)

```
for (i = 0; i < 10; i++) {  
    cout << "[" << i + 1 << "회]" << min << " ~ "  
        << max << " 사이의 값 예측 -> ";  
    cin >> guess;  
    if (com == guess) {  
        cout << "정답입니다." << endl;  
        break;  
    }  
    else if (com > guess) {  
        cout << "너무 작아요!" << endl;  
        min = guess;  
    }  
    else {  
        cout << "너무 커요!" << endl;  
        max = guess;  
    }  
}  
cout << "정답 : " << com << endl;  
cout << "최종 점수 : " << (10 - i) * 10 << endl;
```



함수(function)의 유형

1. return 값이 없는 함수

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

void printGuguDan(int dan) {
    for (int i = 1; i <= 9; i++) {
        cout << dan << " x " << i << " = " << dan * i << endl;
    }
}

int main() {

    cout << "구구단 " << endl;
    printGuguDan(3);

    return 0;
}
```



함수(function)의 유형

2. return값이 있는 함수

```
int myAbs(int x) {  
    if (x < 0)  
        return -x;  
    else  
        return x;  
}  
  
int main() {  
  
    cout << abs(-9) << endl;  
    cout << myAbs(-9) << endl;  
  
    return 0;  
}
```



함수(function)의 유형

3. 함수 중복(overloading)

함수 이름은 같고 매개 변수의 자료형이 다름

```
int myPow(int x, int y) {  
    int result = 1;  
    for (int i = 0; i < y; i++) {  
        result = result * x;  
    }  
    return result;  
}  
  
double myPow(double x, int y) {  
    double result = 1.0;  
    for (int i = 0; i < y; i++) {  
        result = result * x;  
    }  
    return result;  
}
```

```
int main() {  
  
    int r1 = myPow(2, 5);  
    double r2 = myPow(2.0, 5);  
  
    cout << "r1 = " << r1 << endl;  
    cout << "r2 = " << r2 << endl;  
  
    return 0;  
}
```



기본 매개 변수 (default parameter)

- 기본 매개 변수 (default parameter)

함수의 매개 변수에 대응하는 인자가 함수의 호출 시에 생략되면 매개 변수에 기본값이 복사된다.

주의) 맨 뒤에서부터 앞 쪽으로 순서대로 채워서만 정의 할 수 있음

`void add(int x = 1, int y = 2, int z)` 오류! z에 디폴트 값이 없음

```
void printValue(int x, int y = 1) {  
    cout << "x = " << x << ", y = " << y << endl;  
}  
  
int main() {  
  
    printValue(1);  
  
    printValue(1, 2);  
  
    return 0;  
}
```

x = 1, y = 1
x = 1, y = 2



함수의 사용

- 함수의 매개변수로 배열을 사용

```
int findMax(int a[], int len) {  
    int maxVal = a[0];  
    for (int i = 1; i < len; i++)  
        if (maxVal < a[i]) maxVal = a[i];  
    return maxVal;  
}  
  
char findChar(char s[]) {  
    char maxChar = s[0];  
    for (int i = 1; i < strlen(s); i++)  
        if (maxChar < s[i]) maxChar = s[i];  
    return maxChar;  
}
```



함수의 사용

- 함수의 매개변수로 배열을 사용

```
int main() {  
    int arr[10] = { 2, 71, 59, 33, 94, 25, 85, 9, 24, 11 };  
    char str[] = "game over !";  
  
    int maxVal = findMax(arr, 10);  
    char maxChar = findChar(str);  
  
    //cout << strlen(str) << endl;  
    cout << "max val = " << maxVal << endl;  
    cout << "max char = " << maxChar << endl;  
  
    return 0;  
}
```



인라인 함수

➤ 매크로 함수

실제적인 함수 호출이 아니라 전처리에서 코드를 삽입하는 방법을 제공한다.

C언어 매크로 함수

```
#define SQUARE(x) (x*x)

y = SQUARE(10);

// y = (10*10)과 동일한 문장
```

```
#include <stdio.h>
#define SQUARE(x) (x*x)

int square(int x)
{
    return x * x;
}

int main()
{
    int y = SQUARE(10);
    printf("y = %d\n", y);

    int z = square(10);
    printf("z = %d\n", z);
    return 0;
}
```



인라인 함수

➤ inline 함수

함수 호출 오버헤드로 인한 프로그램의 실행 속도 저하를 막기 위한 기능으로 인라인 함수의 코드를 그대로 삽입하여 함수 호출이 일어나지 않게 한다

```
inline int Square(int x) { return x * x; }
inline int min(int a, int b) { return a > b ? b : a; }
int main() {

    int n = Square(4); //제곱수 반환
    cout << n << endl;

    cout << min(5, 6) << endl; //작은 값 출력
    cout << min(6, 5) << endl;

    return 0;
}
```



변수의 메모리 영역

- **코드 영역** : 프로그램의 **실행 코드** 또는 **함수**들이 저장되는 영역
- **스택 영역** : **매개 변수 및 중괄호(블록)** 내부에 **정의된 변수**들이 저장되는 영역
- **데이터 영역** : **전역 변수**와 **정적 변수**들이 저장되는 영역
- **힙 영역** : **동적으로 메모리 할당하는 변수**들이 저장되는 영역



코드 영역
(실행 코드, 함수)



스택 영역
(지역 변수, 매개 변수)



데이터 영역
(전역 변수, 정적 변수)



힙 영역
(동적 메모리 할당)



변수의 적용 범위

➤ 지역 변수(local variable)

- 하나의 코드 블록에서만 정의되어 사용되는 변수
- 함수 또는 제어문의 중괄호{ } 내부에서 사용
- 지역 변수의 메모리 생성 시점 - 중괄호 내에서 초기화할 때
- 지역 변수의 메모리 소멸 시점: - 중괄호를 벗어났을 때

➤ 전역 변수(global variable)

- 전체 소스 코드를 범위로 적용되는 변수
- 소스 파일 내의 어디서든지 사용 가능한 변수
- "자료를 공유"하는 건 편리하지만 모듈들의 연관의 문제가 발생할 수 있음
- 전역 변수의 메모리 생성 시점 - 프로그램이 시작되었을 때
- 전역 변수의 메모리 소멸 시점: - 프로그램이 종료되었을 때



변수의 적용 범위 – 전역 변수

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

int _count = 0;
//_count 전역 변수, 모든 부분에서 사용 가능
```

```
void xy10(int x, int y) {
    //x, y 는 지역변수, xy10() 블록에서만 사용
    x = x * 10;
    y = y * 10;
}
```

```
int main() {

    int a = 1, b = 2;
    //a,b는 지역변수 main()이 끝날때까지 사용 가능

    xy10(a, b);

    _count++;

    cout << "a = " << a << "b = " << b << endl;
    //cout << "x = " << x << "y = " << y << endl;

    cout << "_count = " << _count << endl;

    return 0;
}
```



변수의 적용 범위 – 정적 변수

➤ 정적 변수(static variable)

- `static` 키워드를 붙임
- 함수 안에서만 사용되지만 프로그램이 실행되는 동안 생존하는 변수임

```
void count() {  
    int x = 0;  
    static int y = 0; //전역 변수, 값을 공유함  
    x = x + 1;  
    y = y + 1;  
    cout << "x = " << x << ", y = " << y << endl;  
}  
int main() {  
  
    count();  
    count();  
    count();  
  
    return 0;  
}
```



변수의 적용 범위 – 외부 변수

➤ 외부 변수(Extern Variable)

- 다른 파일에 있는 변수를 사용할때는 **extern** 키워드를 사용한다.(필수)
- 다른 파일에 있는 함수는 extern 생략 가능

add.h

```
int _count = 0;
int add(int x, int y) {
    int sum;
    sum = x + y;
    return sum;
}
```

```
extern int _count;
int add(int, int);

int main() {
    int x = 3, y = 4, z;

    _count++;
    z = add(x, y);

    cout << "_count = " << _count << endl;
    cout << "z = " << z << endl;

    return 0;
}
```

다른 파일의 함수
사용시 extern사용
(생략가능)



함수(function) 예제

- 성적 처리하기

```
#include "calc_total.h"

int main() {

    int s[6] = { 90, 70, 80, 100, 35, 50 };
    int total = 0;
    float avg = 0.0;

    if (total != -1) {
        total = calcTotal(s, 6);
    }

    avg = (float)total / 6;

    cout << fixed;
    cout.precision(2);

    cout << "총점 : " << total << endl;
    cout << "평균 : " << avg << endl;

    return 0;
}
```



함수(function) 예제

- 성적 처리하기

```
#include "calc_total.h"
```

```
int calcTotal(int score[], int len) {  
    int i, total = 0;  
  
    if (len <= 0)  
        return -1;  
  
    for (i = 0; i < len; i++) {  
        if (score[i] < 0 || score[i] > 100)  
            return -1;  
  
        total += score[i];  
    }  
  
    return total;  
}
```

```
//calc_total.h
```

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

```
extern int calcTotal(int score[], int len);  
//extern 외부변수 키워드로 생략 가능
```

