함수란 ?

특정 목적의 작업을 수행하기 위해 독립적으로 설계된 코드의 집합

C++ 프로그램에서 함수는 특정 작업을 캡슐화하는 데 유용하게 사용됩니다.

함수를 사용하는 이유

1. 프로그램에서 특정 기능을 수행을 하는데 있어 반복적인 프로그래밍을 줄일 수 있기 때문에 함수를 사용한다.
2. 함수를 사용하여 프로그래밍을 하면 전체적인 코드의 가독성이 높아진다.
3. 프로그램의 문제가 발생했을 때 유지보수가 쉬워진다.

Tip) 하나의 함수는 하나의 기능을 하도록 구현하는 것이 좋음.

함수 선언

[리턴 타입] 함수 이름 ( 매개변수 목록 )

{ // 함수 몸체

}

call by value

값에 의한 전달 방식은 함수 호출을 통해 전달되는 인수(argument) 값을 함수 내에서 사용할 수 있도록 매개변수에 복사하는 방식

+ 매개변수의 값들을 함수 내에서 변경하여도 원래 인수로 전달된 변수에 영향을 미치지 않음

call by reference

참조에 의한 전달 방식은 함수 호출을 통해 전달되는 인수 값을 복사하는 것이 아닌, 원본 데이터를 전달하는 것 입니다.

+ 매개변수의 값들을 함수 내에서 변경하면 원래 인수로 전달된 변수에 변경된 값이 들어가게 됨

재귀함수

함수 내에서 자기 자신을 또 다시 호출하는 것을 말함.

주의) 재귀 함수를 사용할 때 작업을 마무리 한 후 중단할 수 있는 코드가 포함되어야함.

함수 포인터 ?

<http://tcpschool.com/cpp/cpp_function_pointer>

(추가적인 내용)

< C++ 만의 기능 >

참조자 ( reference )

기존 변수에 별명을 붙여 주어 사용할 수 있도록 한 것

참조자 선언

[변수 타입] 변수 이름 = 값 ; // 변수 선언

[변수 타입] & 참조자 이름 = 변수 이름 ; // 참조자 선언

(& : 주소를 나타내는 것이 아닌 타입을 식별하기 위해 식별자로 사용된 것)

* 참조자는 대상 변수와 같은 메모리 위치를 참조하게 됨.

참조자 선언시 주의 사항

1. 참조자의 타입은 해당 변수의 타입과 같아야 한다.
2. 참조자는 선언과 동시에 초기화 되어야 한다.
3. 참조자는 한 번 초기화 되면, 참조 대상을 바꿀 수 없다.

즉, 참조자를 사용한 후 참조자가 연산 작업을 할 경우 참조자와 참조 대상의 값이 둘 다 바뀌게 된다.

int x = 20 ;

int &y = x ;

y++;

cout << x << “ “ << y << endl; // 11 11

C++ 에서 참조자는 주로 함수에 인자를 전달할 때 사용됩니다.

#include <iostream>

using namespace std;

void call\_by\_value(int x , int y ){

int tmp = x ;

x = y ;

y = tmp ;

}

void call\_by\_reference ( int &x , int &y ) {

int tmp = x ;

x = y ;

y = tmp ;

}

int main(void){

int x = 10 ;

int y = 20 ;

call\_by\_value(x,y);

cout << x << “ “ << y << endl ; // 10 20

call\_by\_reference(x,y) ;

cout << x << “ “ << y << endl; // 20 10

}

Default Argument ( 디폴트 인수 )

1. 함수의 원형에만 지정할 수 있다.
2. 가장 오른쪽부터 시작하여 순서대로만 지정할 수 있다.
3. 가운데 인수들만 별도로 디폴트 인수로 지정할 수 없다.

void sum (int a , int b , int c , int d );

void sum (int a , int b , int c = 3, int d = 4 ) ;

void sum (int a = 1 , int b = 2, int c , int d ) ; ( X )

<사용시>

#include <iostream>

using namespace std ;

void sum ( int a , int b, int c = 3, int d = 4 ){

int sum = a + b + c + d ;

cout << “sum: “ << sum << endl;

}

int main(void){

sum ( 1, 2 ); // sum(1,2,3,4) 와 같음 ( sum: 10 )

sum ( 1, 2, 6, 8 ); // sum(1,2,6,8) 와 같음 ( sum: 17 )

// sum ( 1, 5, , 9 ); // ERROR

return 0;

}

< C++ 함수 오버로딩 >

함수 오버로딩(Overloading)은 같은 이름의 함수를 중복하여 정의하는 것을 의미함.

즉, 오버로딩(Overloading)은 같은 이름의 함수에 매개 변수의 목록들을 달리하여 하나의 이름으로 사용할 수 있도록 한 것

+함수 시그니처

함수의 원형에 있는 매개변수 목록을 의미

( 함수의 매개 변수의 개수와 그 타입이 모두 같을 경우, 두 함수는 동일하다고 한다. )

즉, 오버로딩은 같은 이름을 가진 함수들이 서로 다른 시그니처를 가질 때 오버로딩이 된다.

1. 매개 변수의 개수가 다른 경우
2. 매개 변수의 개수가 같지만 타입이 다른 경우

void print\_func(int a , int b , char t);

void print\_func(int a , int b , int c );

void print\_func(int a, int b);

void print\_func(char a, char b);

+예제