**Ch01. C언어 기반의 C++**

**Ch01-1. Printf와 scanf를 대신하는 입출력 방식**

헤더파일의 선언 : #include<iostream>

출력의 기본구성 : std::cout<<’출력대상I’<<’출력대상2’<<’출력대상3’;

개행의 진행 : std::endl을 출력하면 개행이 이뤄진다.

+C 언어에서는 출력의 대상에 따라 서식지정을 달리했지만, C++에서는 그러한 과정이 불필요하다.

입력의 기본구성 : std::cin>>’변수’

변수의 선언위치 : 함수의 중간부분에서도 변수의 선언이 가능하다.

+출력에서와 마찬가지로 입력에서도 별도의 서식 지정이 불필요하다.

+C++에서는 변수의 선언 위치에 제한을 두지 않는다.

std::cin을 통해서 입력되는 데이터의 구분은 스페이스바, 엔터, 탭과 같은 공백을 통해서 이뤄진다.

for(int i=val2+1; i<val1; i++)

result+=i;

* C++에서는 for문 안에서 변수의 선언과 동시에 초기화가 가능하다.

문자열의 입력방식도 다른 데이터의 입력방식과 큰 차이가 나지 않는다.

**Ch01-2. 함수 오버로딩(Function overloading)**

C++은 함수호출 시 ‘함수의 이름’과 ‘전달되는 인자의 정보’를 동시에 참조하여 호출할 함수를 결정한다. 따라서 이렇듯 매개변수의 선언이 다르다면 동일한 이름의 함수도 정의 가능하다. 그리고 이러한 형태의 함수를 가리켜 ‘함수 오버로딩(Function Overloading)’이라고 한다.

+반환형의 차이는 함수 오버로딩의 조건을 만족시키지 않는다.

Ch01-3. 매개변수의 디폴트 값

#include <iostream>

int Adder(int num1=1, int num2=2);

int main(void)

{

    std::cout<<Adder()<<std::endl;

    std::cout<<Adder(5)<<std::endl;

    std::cout<<Adder(3, 5)<<std::endl;

    return 0;

}

int Adder(int num1, int num2)

{

    return num1+num2;

}

* 함수의 선언을 별도로 둘 때에는 디폴트 값의 선언을 함수의 선언부에 위치시켜야 한다. 그 이유는 컴파일러의 컴파일 특성에서 찾을 수 있다. 컴파일러는 함수의 디폴트 값의 지정여부을 알아야 함수의 호출문장을 적절히 컴파일 할 수 있다.