08\_함수

학습목표

함수의 개념을 이해한다.

함수를 작성하는 방법을 익힌다.

함수의 반환 값과 매개 변수를 이해한다.

전역 변수, 지역 변수를 이해한다.

1. 함수는 왜 필요한가?

“청소하기”나 “세탁하기”처럼 우리가 일상생활에서 되풀이해야 하는 작업이 있다. 일상생활에서도 되풀이되는 귀찮은 작업이 있듯이 프로그램에서도 되풀이되는 작업들이 있다.

함수란 특정 작업을 수행하여 그 결과를 반환하는 문장들의 집합이다. 함수는 입력 데이터를 받아서 처리한 후에 결과값을 반환한다. 각각의 함수에는 이름이 붙어 있으며 이름을 가지고 함수를 호출하여 사용할 수 있다.

함수는 본래 코드를 반복하지 않기 위해서 고안된 것이지만 최근의 프로그래밍에서는 그 이상의 의미를 지닌다. 함수는 프로그램을 구성하는 기본적인 구성 요소라고 할 수 있다. 하나의 프로그램은 여러 함수들이 모여서 만들어진다. 함수는 레고 블록 같은 존재라 할 수 있다. 함수를 프로그램을 이루는 부품으로 생각하면 이해하기 쉽다. 함수들을 조립하여서 프로그램을 작성하는 것이다.

프로그램의 전체 설계가 끝나고 각 함수들의 사양만 결정되면 각 함수들은 독자적으로 개발될 수 있다. 함수를 사용하게 되면 각 함수들을 독립적으로 업그레이드할 수 있어서 유지 보수가 쉬워지며 다른 프로그램에서도 이 함수들을 재활용할 수 있다.

결론적으로 함수를 사용하게 되면 코드의 재활용, 가독성, 유지 관리의 측면에서 많은 이점이 있다.

1. 함수의 정의

형식:

Int add ( int x, int y ) 반환형/함수 이름/매개변수들

{

Int result;

……

}

가장 먼저 함수가 반환하는 값의 자료형을 먼저 쓰게 되어 있다. 이것을 함수의 반환형(return type)이라고 한다. 이어서 함수의 이름, 매개 변수(parameter)들을 차례대로 적어주면 된다. 이것을 합쳐서 함수 헤더(function header)라고 한다. 함수의 몸체는 중괄호로 둘러싸인 부분으로 함수가 수행하는 작업이 적혀있다.

함수의 반환형은 함수가 처리를 종료한 후에 호출한 곳으로 반환하는 데이터의 유형을 말한다. 만약 값을 반환하지 않는다면 void라고 표시한다. 값을 반환하려면 return문장을 사용해야 한다. Return 다음에 수식을 써주면 이 수식의 값이 반환된다. Return 0;은 0을 반환하는 것이다.

함수 이름은 변수의 이름과 동일한 규칙으로 만들 수 있다. 규칙만 따른다면 어떤 이름이라도 가능하다. 일반적으로 함수의 목적을 설명하는 동사 또는 동사+명사를 사용한다.

함수를 호출할 때는 작업에 필요한 데이터 값들을 보낼 수 있다. 매개 변수(parameter)는 함수가 받는 데이터를 함수 몸체로 전달해주는 변수이다. 매개 변수는 여러 개가 될 수 있으며 각 매개 변수는 쉼표로 분리된다.

기본적으로 하나의 함수는 하나의 작업만을 수행하여야 한다. 일반적으로 하나의 함수는 30행을 넘지 않도록 하는 것이 좋다. 함수의 반환형을 명시하지 않으면 int형으로 가정한다. 만약 반환 값이 void형로 지정된 함수에서 값을 반환하면 문법적인 오류가 발생한다.

함수는 블랙박스(blackbox)와 같아서 외부에서는 함수의 이름과 입력, 출력만 보인다. 함수의 구체적인 코드는 보이지 않는다. 내부에서도 마찬가지이다. 외부 상황을 신경 쓰지 않고 입력을 받아 작업을 한 후에 결과를 내보낸다.

함수를 사용하기 위해서는 함수를 호출해야 한다. 함수를 호출하려면 함수의 이름을 써주고 함수가 필요로 하는 데이터를 나열한 다음, 세미콜론을 붙이면 된다.

Ex) n = get\_integer();

함수를 호출하게 되면 현재 실행하고 있던 코드는 잠시 중단된다. 호출된 함수 안에 있는 문장들이 순차적으로 실행되며 문장들의 실행이 끝나면 호출한 위치로 되돌아간다.

인수와 매개 변수는 함수 호출시에 데이터를 주고받기 위하여 필요하다. 인수(argument)는 호출 프로그램에 의하여 함수에 실제로 전달되는 값이다. 함수가 호출될 때마다 인수는 함수의 매개 변수로 전달된다. 인수를 실제 매개 변수라고도 하고 매개 변수를 형식 매개 변수라고 부르기도 한다. 여기서 주의해야할 점은 매개 변수의 개수와 인자의 개수는 정확히 일치해야 한다는 점이다. 매개 변수의 개수와 인수의 개수가 일치하지 않으면 아주 찾기 어려운 오류가 발생하게 된다. 또한 인수와 매개 변수의 타입은 서로 일치해야 한다.

반환 값(return value)은 호출된 함수가 호출된 곳으로 작업의 결과값을 전달하는 방법이다. return문장 뒤에 있는 수식의 계산 값이 반환 값이 된다.

-여러 개의 반환 값을 전달하기 위해서는 “포인터”나 “구조체” 를 사용해야 한다.

일반적으로 함수를 사용할 때는 미리 컴파일러에게 함수에 대한 정보를 알려야 한다. 이것을 함수 원형(function prototype)이라고 한다. 함수 원형은 함수가 사용되기 전에 정의되어야 한다. 함수 원형은 함수의 이름, 매개 변수, 반환형을 함수가 정의되기 전에 미리 한번 써주는 것이다. 함수 원형은 함수 헤더에 세미콜론(;)만을 추가한 것과 똑같다. 함수 원형이 필요한 이유는 컴파일러에게 미리 함수에 대한 정보를 주어서 함수의 매개 변수 검사, 반환형 검사 등을 하게 하기 위한 것이다. 함수 원형을 적지 않고 main 앞에 정의해도 되지만 함수 호출이 서로 물고 물리는 경우에는 이 방법이 불가능하다. 대부분의 경우에 먼저 함수 원형을 적어주는 것이 좋다.

1. 지역 변수와 전역 변수

함수와 밀접하게 관련이 있는 것이 변수이다. 함수 안에서 정의되는 변수는 지역 변수라고 불리고 해당 함수 안에서만 사용이 가능하다. 함수의 외부에서 선언되는 변수는 전역 변수라고 불린다. 전역 변수는 소스 파일 어디서나 사용이 가능하다.

지역 변수(local variable)는 함수 안에 선언되는 변수이다. 정확하게 말하자면 블록 안에 선언되는 변수이고 블록(block)이란 중괄호로 둘러싸인 영역이다. 지역 변수는 선언된 블록 안에서만 접근과 사용이 가능하다. 지역 변수는 초기화시키지 않으면 쓰레기 값을 가지게 된다. 따라서 지역 변수를 사용할 때는 반드시 초기화시켜야 한다.

전역 변수(global variable)는 함수 외부에서 선언되는 변수이다. 전역 변수의 범위는 소스 파일 전체이다. 즉 전역 변수는 소스 파일 안의 모든 함수에서 사용이 가능한 변수이다. 프로그래머가 전역 변수를 초기화시키지 않으면 컴파일러가 0으로 초기화시킨다. 전역 변수의 생존 기간은 프로그램 시작부터 끝까지 이기 때문에 많이 사용하면 메모리를 많이 사용하게 된다. 또한 어디서나 접근이 가능하다는 점 때문에 코드가 길어질 경우에 코드가 꼬일 수 있다.

같은 이름의 전역 변수와 지역 변수가 존재할 경우 지역 변수가 전역 변수보다 우선시된다. 이것을 지역 변수가 전역 변수를 가린다고 한다.

1. 정적 변수

지역 변수처럼 블록에서만 사용되지만 블록을 벗어나도 자동으로 삭제되지 않는 변수가 있다. 지역 변수를 정적 변수(static variable)로 만들면 이런 변수가 된다.

정적 변수는 키워드 static을 앞에 붙여서 만든다. 정적 변수는 전역 변수와 같이 프로그램이 시작할 때 생성되고 프로그램이 끝나면 삭제된다.

정적 변수는 보통 함수가 가지고 있는 변수의 변화를 유지하기 위해 사용된다.

1. 순환 호출

함수는 자기 자신을 호출할 수도 있다. 이것을 순환(recursion)이라고 부른다. 이것은 처음에는 상당히 이상하게 보이지만 사실은 순환은 가장 흥미롭고 또 효과적인 프로그래밍 기법 중의 하나이다. 문제의 정의가 순환적으로 되어 있는 경우, 대게 순환 형태의 코드가 더 이해하기 쉽다. 그러나 순환 코드의 약점은 수행시간이다. 함수를 빈번하게 호출해야 하기 때문에 실행 시간이 더 걸리는 경우가 많다. 그러나 순환을 사용하지 않으면 도저히 프로그램을 작성할 수 없는 경우가 종종 있다. 따라서 순환은 반드시 익혀두어야 한다.