6주차 1차시 지역변수

【학습목표】

1. 함수의 선언에 대해 설명할 수 있다.
2. 지역변수에 대해 설명할 수 있다.

학습내용1 : 함수의 선언

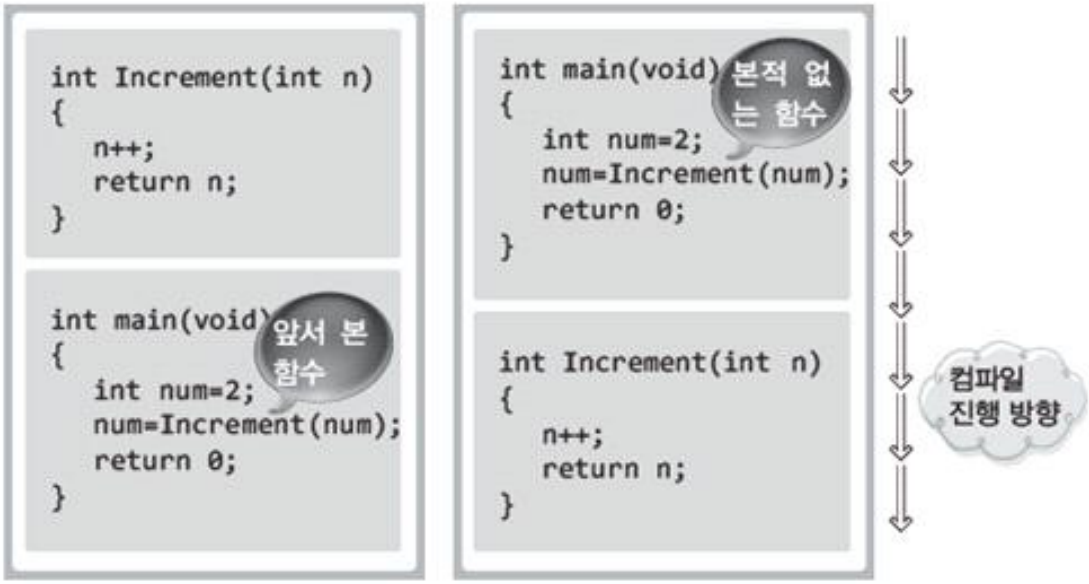
1. 값을 반환하지 않는 return

```
void NoReturnType(int num)
{
    if(num<0)
        return;    // 값을 반환하지 않는 return문!
    . . . .
}
```

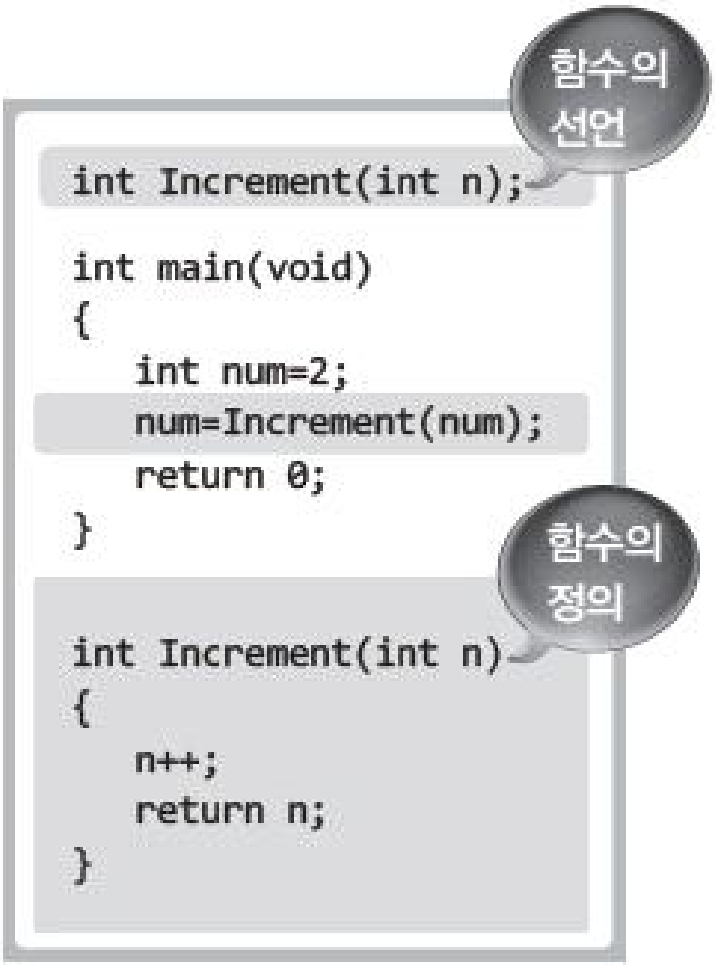
√return문에는 “값의 반환“과 ”함수의 탈출“이라는 두 가지 기능이 담겨있다.

√위의 예제에서 보이듯이 값을 반환하지 않는 형태로 return문을 구성하여 값은 반환하지 않되 함수를 빠져나가는 용도로 사용할 수 있다.

2. 함수의 정의와 그에 따른 원형의 선언



√컴파일이 위에서 아래로 진행이 되기 때문에 함수의 배치순서는 중요하다.
컴파일 되지 않은 함수는 호출이 불가능하다.



✓이후에 등장하는 함수에 대한 정보를 컴파일러에게 제공해서 이후에 등장하는 함수의 호출문장이 컴파일 가능하게 도울 수 있다.
이렇게 제공되는 함수의 정보를 가리켜 "함수의 선언"이라 한이다.

int Increment(int n); //함수의 선언
int Increment(int); // 위와 동일한 함수선언, 매개변수 이름 생략 가능

3. 다양한 종류의 함수 정의1

```
int main(void)
{
    printf("3과 4중에서 큰 수는 %d 이다. \n", NumberCompare(3, 4));
    printf("7과 2중에서 큰 수는 %d 이다. \n", NumberCompare(7, 2));
    return 0;
}

int NumberCompare(int num1, int num2)
{
    if(num1>num2)
        return num1;
    else
        return num2;
}
```

중간에도 얼마든지 return문이 올 수 있다.

```
3과 4중에서 큰 수는 4 이다.
7과 2중에서 큰 수는 7 이다.
```

```
printf("3과 4중에서 큰 수는 %d 이다. \n", NumberCompare(3, 4));
printf("7과 2중에서 큰 수는 %d 이다. \n", NumberCompare(7, 2));
```

위의 두 문장한 NumberCompare 함수호출 이후 아래와 같이 된다.

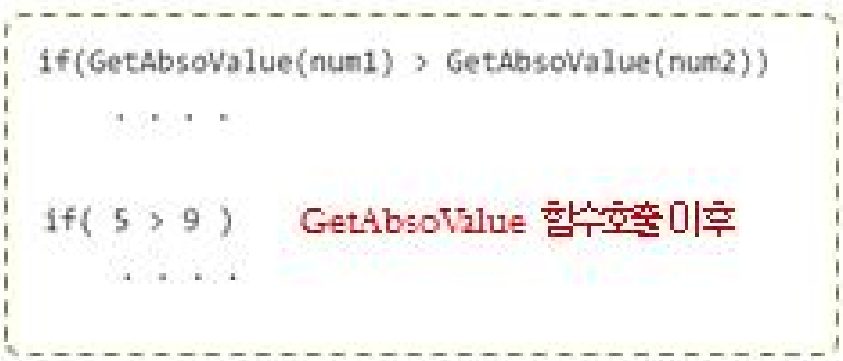
```
printf("3과 4중에서 큰 수는 %d 이다. \n", 4);
printf("7과 2중에서 큰 수는 %d 이다. \n", 7);
```

4. 다양한 종류의 함수 정의2

```
int AbsoCompare(int num1, int num2);   // 절댓값이 큰 정수 반환
int GetAbsoValue(int num);     // 전달인자의 절댓값을 반환

int main(void)
{
    int num1, num2;
    printf("두 개의 정수 입력: ");
    scanf("%d %d", &num1, &num2);
    printf("%d와 %d중 절댓값이 큰 정수: %d \n",
        num1, num2, AbsoCompare(num1, num2));
    return 0;
}
int AbsoCompare(int num1, int num2)
{
    if(GetAbsoValue(num1) > GetAbsoValue(num2))
        return num1;
    else
        return num2;
}
int GetAbsoValue(int num)
{
    if(num<0)
        return num * (-1);
    else
        return num;
}
```

```
두 개의 정수 입력: 5 -9
5와 -9중 절댓값이 큰 정수: -9
```



이 예제에서 보이듯이 함수의 호출 문장은 어디에든 놓일 수 있다.

학습내용2 : 지역변수

1. 함수 내에만 존재 및 접근 가능한 지역변수

```
int SimpleFuncOne(void)
{
    int num=10;    // 이후부터 SimpleFuncOne의 num 유효
    num++;
    printf("SimpleFuncOne num: %d \n", num);
    return 0;    // SimpleFuncOne의 num이 유효한 마지막 문장
}

int SimpleFuncTwo(void)
{
    int num1=20;    // 이후부터 num1 유효
    int num2=30;    // 이후부터 num2 유효
    num1++, num2--;
    printf("num1 & num2: %d %d \n", num1, num2);
    return 0;    // num1, num2 유효한 마지막 문장
}
```

```
int main(void)
{
    int num=17;    // 이후부터 main의 num 유효
    SimpleFuncOne();
    SimpleFuncTwo();
    printf("main num: %d \n", num);
    return 0;    // main의 num이 유효한 마지막 문장
}
```

```
SimpleFuncOne num: 11
num1 & num2: 21 29
main num: 17
```

√함수 내에 선언되는 변수를 가리켜 지역변수라 한다.
√지역변수는 선언된 이후로부터 함수 내에서만 접근이 가능하다.
√한 지역(함수) 내에 동일한 이름의 변수 선언 불가능하다.
√다른 지역에 동일한 이름의 변수 선언 가능하다.
√해당 지역을 빠져나가면 지역변수는 소멸된다. 그리고 호출될 때마다 새롭게 할당된다.

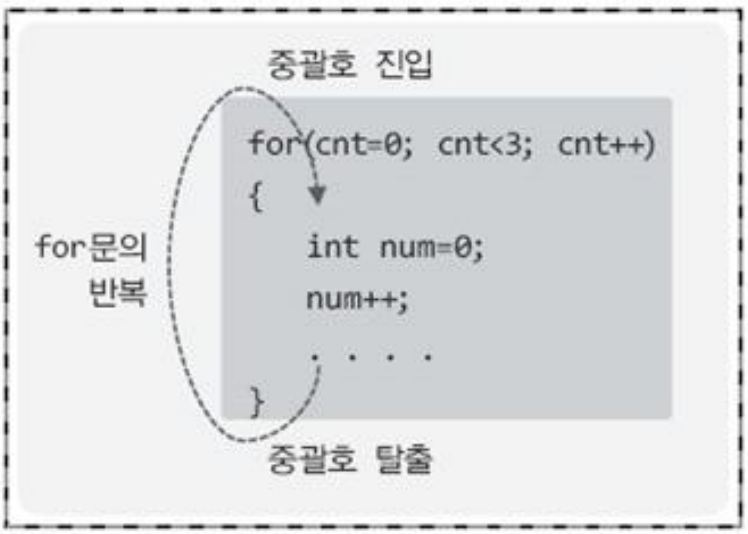
2. 메모리 공간의 할당과 소멸 관찰하기

```
int SimpleFuncOne(void)
{
    int num=10;    // 이후부터 SimpleFuncOne의 num 유효
    num++;
    printf("SimpleFuncOne num: %d \n", num);
    return 0;    // SimpleFuncOne의 num이 유효한 마지막 문장
}

int SimpleFuncTwo(void)
{
    int num1=20;    // 이후부터 num1 유효
    int num2=30;    // 이후부터 num2 유효
    num1++, num2--;
    printf("num1 & num2: %d %d \n", num1, num2);
    return 0;    // num1, num2 유효한 마지막 문장
}
```

```
int main(void)
{
    int num=17;    // 이후부터 main의 num 유효
    SimpleFuncOne();
    SimpleFuncTwo();
    printf("main num: %d \n", num);
    return 0;    // main의 num이 유효한 마지막 문장
}
```

3. 다양한 형태의 지역변수



√for문의 중괄호 내에 선언된 변수도 지역변수이다.
그리고 이 지역변수는 for문의 중괄호를 빠져나가면 소멸된다.
따라서 for문의 반복횟수만큼 지역변수가 할당되고 소멸된다.

```
int main(void)
{
    int num=1;
    if(num==1)
    {
        int num=7;  // 이 행을 주석처리 하고 실행결과 확인하자!
        num+=10;
        printf("if문 내 지역변수 num: %d \n", num);
    }
    printf("main 함수 내 지역변수 num: %d \n", num);
    return 0;
}
```

if문 내에 선언된 변수 num이 main 함수의 변수 num을 가린다.

실행결과

```
if문 내 지역변수 num: 17
main 함수 내 지역변수 num: 1
```

주석처리 후 실행결과

```
if문 내 지역변수 num: 11
main 함수 내 지역변수 num: 11
```

4. 지역변수의 일종인 매개변수

√매개변수는 일종의 지역변수이다.
√매개변수도 선언된 함수 내에서만 접근이 가능하다.
√선언된 함수가 반환을 하면, 지역변수와 마찬가지로 매개변수도 소멸된다.

【학습정리】

1. 키워드 return은 값을 반환하면서 함수를 빠져나갈 때 사용된다. 즉 함수를 빠져나가거나 값을 반환한다.

2. 하나의 함수 내에 둘 이상의 return문이 존재할 수 있다.

3. 중괄호 내에 선언되는 변수는 모두 지역변수다.

4. 지역변수는 해당 지역을 벗어나면 자동으로 소멸된다.

5. 지역변수는 선언된 지역 내에서만 유효하기 때문에 지역이 다르면 이름이 같아도 문제가 되지 않는다.