함수(2)

- 지역 변수, 전역 변수, 정적 지역 변수 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

지역 변수

◎ 선언되는 위치에 따른 변수의 구분

- 지역 변수 (local variable)
- 전역 변수 (global variable)
- 지역 변수와 전역 변수는 메모리 상에 존재하는 기간과 변수에 접근할 수 있는 범위가 다름

지역 변수

◎ 지역 변수

- 중괄호로 형성된 블록 구문(혹은 특정 함수 내)에서 선언된 변수
- 선언된 지역 내에서만 유효하며, 같은 블록 안에서 동일한 이름으로 중복으로 선언할 수 없음
- 블록 안에 선언된 지역 변수는 블록 밖을 나오면 메모리 상에서 소멸되고 호출될 때마다 새롭게 할당됨
- 지역변수는 스택이라는 메모리 영역에 할당

지역 변수

◎ 스택 (stack)

- LIFO (Last-In-First-Out) 순으로 데이터를 입출력하는 데이터 구조
- Push : 스택에 데이터를 삽입하는 연산
- Pop : 스택에서 데이터(top)를 삭제하는 연산
- Top: 스택에서 가장 위에 존재하는 데이터

지역 변수

◎ 지역 변수의 예

- LocalVariable.c

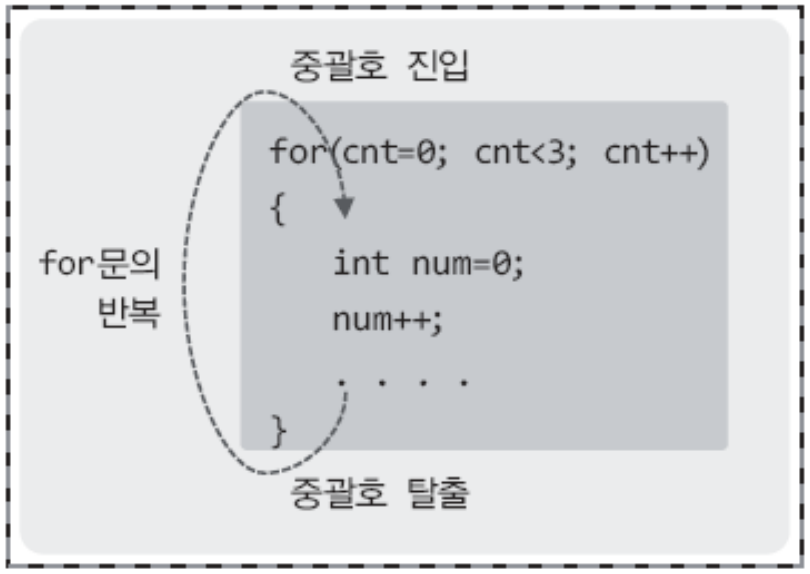
```
int SimpleFuncOne(void)
{
    int num=10;    // 이후부터 SimpleFuncOne의 num 유효
    num++;
    printf("SimpleFuncOne num: %d \n", num);
    return 0;    // SimpleFuncOne의 num이 유효한 마지막 문장
}

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
```

지역 변수

◎ 중괄호 내에서 선언된 지역 변수



- for 문의 중괄호 내에 선언된 지역변수는 for 문의 중괄호를 빠져나가면 메모리 상에서 소멸됨
- for 문의 반복 횟수만큼 지역 변수가 할당되었다가 소멸됨

지역 변수

◎ 중괄호 내에서 선언된 지역 변수의 예

- LocalVaiHideVal.c

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

- 외부에서 선언된 변수를 블록 안에서 같은 이름으로 선언하면 이전에 선언된 변수에 우선하여 처리됨

지역 변수

◎ 함수의 매개 변수도 지역 변수의 일종

- 매개 변수도 선언된 함수 내에서만 접근이 가능
- 선언된 함수가 종료되면, 지역변수와 마찬가지로 매개변수도 함께 소멸

```
A.  B.  C.
int Add (int num1, int num2)
{
    int result = num1 + num2;
D. return result;
}
```

A. 반환형
B. 함수의 이름
C. 매개변수
D. 값의 반환

지역 변수

◎ 함수의 매개 변수도 지역 변수의 일종

- 매개 변수도 선언된 함수 내에서만 접근이 가능
- 선언된 함수가 종료되면, 지역변수와 마찬가지로 매개변수도 함께 소멸
- SimpleAddFunc.c

```
int Add(int num1, int num2)
{
    return num1+num2;
}

int main(void)
{
    int result;
    result = Add(3, 4);
    printf("덧셈결과1: %d \n", result);
    result = Add(5, 8);
    printf("덧셈결과2: %d \n", result);
    return 0;
}
```

```
덧셈결과1: 7
덧셈결과2: 13
```

전역 변수

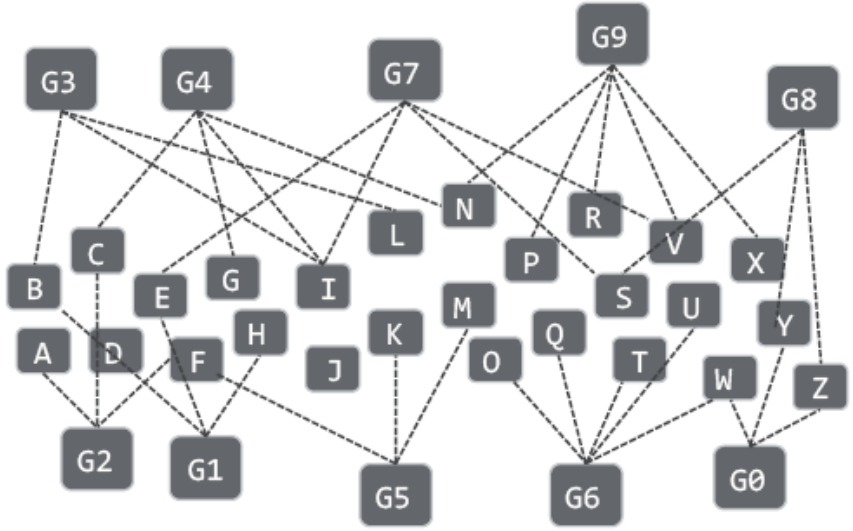
◎ 전역 변수 (global variable)

- 함수 외부에 선언되는 변수로 프로그램 전체 영역 어디서나 접근이 가능한 변수
- 프로그램 시작과 동시에 메모리 공간에 할당되어 프로그램이 마칠 때까지 메모리 상에 존재

  (참고) 지역 변수 : 선언된 영역 내에서면 유효하고, 블록 밖을 나오면 메모리 상에서 소멸
- 일반적으로 0으로 초기화 됨
- 전역 변수와 동일한 이름의 지역 변수를 선언하면, 지역 변수가 선언된 영역에선 지역 변수가 우선하여 인식

◎ 전역 변수의 사용

- 프로그램 실행 중에 소멸되지 않고 어디서든 접근이 가능함
- 전역 변수를 많이 선언하면 프로그램의 복잡도가 증가함
- 전역 변수를 선언해야 하는 상황에서만 신중하게 선언해야 함



전역 변수

◎ 전역 변수 (global variable)

- GlobalVariable.c

```
void Add(int val);
int num;    // 전역변수는 기본 0으로 초기화됨

int main(void)
{
    printf("num: %d \n", num);
    Add(3);
    printf("num: %d \n", num);
    num++;   // 전역변수 num의 값 1 증가
    printf("num: %d \n", num);
    return 0;
}

void Add(int val)
{
    num += val;    // 전역변수 num의 값 val만큼 증가
}
```

```
num: 0
num: 3
num: 4
```

전역 변수

◎ 전역 변수 (global variable)

- LocalValHideGlobalVal.c

```
int Add(int val);
int num=1;

int main(void)
{
    int num=5;
    printf("num: %d \n", Add(3));
    printf("num: %d \n", num+9);
    return 0;
}
int Add(int val)
{
    int num=9;
    num += val;
    return num;
}
```

```
num: 12
num: 14
```

정적 지역 변수

◎ 지역 변수 (local variable)

- 선언된 함수 내에서만 접근이 가능
- 함수 내에서 선언된 지역 변수는 해당 함수가 종료하면 소멸

◎ 정적 지역 변수 (static local variable)

- 선언된 함수 내에서만 접근이 가능
- 프로그램 실행 시 한 번 초기화 되고, 프로그램 종료 시까지 메모리 상에 존재 (전역 변수의 특성)
- 전역 변수와 달리 접근 범위를 제한할 수 있음

정적 지역 변수

◎ 정적 지역 변수 (static local variable)

- StaticLocalVariable.c

```
void SimpleFunc(void)
{
    static int num1=0;    // 초기화하지 않으면 0 초기화
    int num2=0;     // 초기화하지 않으면 쓰레기 값 초기화
    num1++, num2++;
    printf("static: %d, local: %d \n",num1, num2);
}

int main(void)
{
    int i;
    for(i=0; i<3; i++)
        SimpleFunc();
    return 0;
}
```

```
static: 1, local: 1
static: 2, local: 1
static: 3, local: 1
```