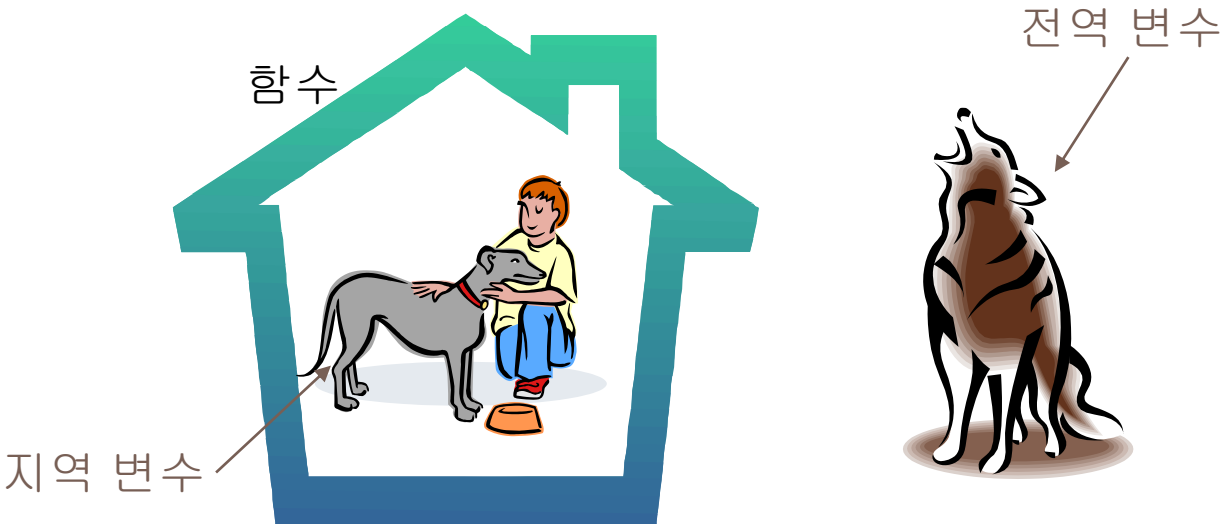
9장 함수

차례

1. 함수를 정의하고 선언하기
2. 변수의 존재기간과 접근범위 1 : 지역변수
3. 변수의 존재기간과 접근범위 1 : 전역변수, static변수
4. 재귀함수에 대한 이해

변수의 접근 범위

- 변수의 접근 범위에 따라 지역 변수, 전역 변수로 구분
- 지역변수 : 함수 안에서 선언된 변수, 함수안에서만 접근 가능
- 전역변수 : 함수 외부에서 선언된 변수, 모든 함수에서 접근 가능



지역 변수

- 지역 변수(local variable): 함수 안에서 선언된 변수, 자동 변수

```
int compute_sum(int n)
{
        int i, result = 0;
        for(i=0; i<=n; i++)
                result += i;
        return result;
}
```

지역 변수

지역 변수의 사용 범위

- 지역 변수는 선언된 함수 안에서만 사용(접근) 가능
- 다른 함수에서 사용시 컴파일 에러 발생

```
int sub1()
{
        int x;
        x=10;
        …
}
void sub2()
{
        printf("%d \n", x);        // 다른 함수의 지역변수 접근 불가능
                                   // 컴파일시 에러 발생!!!
}
```

지역 변수 x의
유효한 범위

지역 변수의 생존 기간

- 함수가 호출되면 메모리에 생성되고 종료되면 자동으로 소멸된다.

```
int sub()
{
      int i=0,result;                    // 지역변수 생성


      …
      return result;            // 지역변수 소멸
}
```

지역 변수의 초기값

- 지역변수는 초기화 하지 않으면 쓰레기값을 가진다.

```
#include <stdio.h>
int main(void)
{
        int temp;                       // 초기화하지않으면 쓰레기값
        printf("temp=%d\n",temp);
        return 0;
}
```

temp=-2523352

함수의 매개 변수

- 매개 변수도 일종의 지역 변수
- 함수를 호출할 때 메모리에 할당되고 인수 값으로 초기화

```
int inc(int counter)   // 매개변수 할당 및 인수로 초기화
{
        counter++;
        return counter;
}

int main(void)
{
        …
        result = inc(value);
        …
}
```

```
int inc( )
{
        int counter = value;
        counter++;
        return counter;
}
```

같은 이름의 지역 변수

- 같은 이름의 지역 변수가 존재해도 선언된 함수가 다르면 문제없음

```
int sub1()
{
      int x=0;          // 이름이 같아도 다른 변수임
      …
}
void sub2()
{
      int x=0;
      …
}
void sub3(int x)
{
      x=0;
      …
}
```

지역변수 정리

	지역(자동)변수	전역(외부)변수	정적지역변수
선언위치	함수(블록) 내부		
접근범위	함수내부		
생존기간	함수호출~종료		
자동초기화	X		
초기값	쓰레기값		
장단점	함수내부에서만 접근 가능하므로 변수 오염방지		

전역 변수

- 전역 변수(global variable): 함수의 외부에서 선언된 변수

```
#include <stdio.h>

int global = 123;                    // 전역 변수

int main(void)
{
        ...
}
```

전역 변수의 접근 범위

- 전역변수는 모든 함수에서 접근(사용)이 가능

```
#include <stdio.h>
void sub1();
void sub2();

int global = 123;                    // 전역 변수

int main(void)
{
      printf("%d\n", global);        // 전역 변수는 모든 함수에서 사용가능
      sub1();
      sub2();
}
void sub1()
{
      printf("%d\n", global);        // 전역 변수는 모든 함수에서 사용가능
}
void sub2()
{
      printf("%d\n", global);        // 전역 변수는 모든 함수에서 사용가능
}
```

```
123
123
123
```

전역 변수의 생존 기간과 초기값

- 전역변수의 생존기간은 프로그램이 실행될때 메모리에 생성되어 프로그램이 종료될때 메모리에서 소멸된다.
- 초기값을 별도로 설정하지 않으면 0으로 자동으로 초기화됨.

전역 변수의 초기값과 생존 기간


```
#include <stdio.h>
int counter;                    // 전역 변수
void set_counter(int i)
{
        counter = i;            // 직접 사용 가능
}
int main(void)
{
        printf("counter=%d\n", counter);        // 직접 사용 가능
        counter = 100;
        printf("counter=%d\n", counter);
        set_counter(20);
        printf("counter=%d\n", counter);

        return 0;
}
```

```
counter=0
counter=100
counter=20
```

같은 이름의 전역 변수와 지역 변수

- 전역 변수와 지역 변수가 같은 이름을 갖는 경우 지역 변수가 전역 변수를 우선한다.
- 먼저 지역변수에서 찾고 없으면 전역 변수를 찾는다.

```
// 동일한 이름의 전역 변수와 지역 변수
#include <stdio.h>

int sum = 123;          // 전역 변수

int main(void)
{
        int sum = 321;                  // 지역 변수
        printf("sum = %d\n", sum);      // sum: 지역 변수가 우선

        return 0;
}
```

```
sum = 321
```

전역 변수의 사용

- 전역변수는 모든 함수에서 접근이 가능하므로 함수 사이에 데이터를 공유할 수 있다.
- 동시에 데이터 오염 가능성이 매우 높으므로 프로그램 오류의 원인이 된다. -> 디버깅이 매우 어려워서 가능하다면 전역변수는 사용하지 말 것!!!!

전역 변수의 사용


```
// 전역 변수를 사용하여 프로그램이 복잡해지는 경우
#include <stdio.h>
void f(void);
int i;
int main(void)
{
        for(i = 0;i < 5; i++)                   // i: 전역변수 -> 지역변수로 선언해야
        {
                f();                            // 함수종료 후 i의 값이 10이 된다.
        }
        return 0;
}
void f(void)
{
        for(i = 0;i < 10; i++)                  // i: 전역변수 -> 지역변수로 선언해야
                printf("#");
}
```

##########

전역변수 정리

	지역(자동)변수	전역(외부)변수	정적지역변수
선언위치	함수(블록) 내부	함수 외부	
접근범위	함수 내부	모든 함수	
생존기간	함수 호출~종료	프로그램시작~종료	
자동초기화	X	O	
초기값	쓰레기값	0	
장단점	함수내부에서만 접근 가능하므로 변수 오염방지	함수 사이에 데이터 교환 편리 버그의 원인됨 디버깅어려움	

저장 유형 지정자 static

- 정적 변수: 자료형 앞에 static을 붙인 변수
- 정적지역변수 : 지역변수 앞에 static 붙임
- 정적전역변수(14장) : 전역 변수 앞에 static 붙임

```
static  int  count1;                    //정적전역변수
void sub()
{
    static  int  count2;                //정적지역변수
    …

    return;
}
```

정적지역변수 특징

- 생존기간: 정적지역변수는 프로그램이 시작할 때 메모리에 생성되고 프로그램이 종료될 때 제거된다.(전역변수와 같음)
- 자동초기화 : 정적지역변수는 자동으로 0으로 초기화된다.(전역변수와 같음)
- 접근범위 : 정적지역변수의 접근 범위는 지역 변수와 같다.(지역변수와 같음)

정적지역변수


```
#include <stdio.h>
void sub(void);
int main(void)
{
        int i;
        for(i = 0;i < 3; i++)  sub();
        return 0;
}
void sub(void)
{
    int auto_count = 0;                 // 지역변수
    static int static_count = 0;        // 정적지역변수, 프로그램 시작시 한
                                        // 번만 할당되고 초기화
    auto_count++;
    static_count++;
    printf("auto_count=%d\n", auto_count);
    printf("static_count=%d\n", static_count);
}
```

```
auto_count=1
static_count=1
auto_count=1
static_count=2
auto_count=1
static_count=3
```

변수 정리

	지역(자동)변수	전역(외부)변수	정적지역변수
선언위치	함수(블록) 내부	함수외부	함수내부
접근범위	함수내부	모든 함수	함수내부
생존기간	함수호출~종료	프로그램시작~종료	프로그램시작~종료
자동초기화	X	O	O
초기값	쓰레기값	0	0
장단점	함수내부에서만 접근 가능하므로 변수 오염방지	함수사이에 데이터 교환편리 버그의 원인됨 디버깅어려움	접근범위를 함수내부로 제한하면서 함수종료후에도 값이 존재한다.

실습과제 1

- 다음을 채워보자

	지역(자동)변수	전역(외부)변수	정적지역변수
선언위치			
접근범위			
생존기간			
자동초기화			
초기값			
장점			
단점			

실습과제2

- 아래 실행 결과를 참고하여 함수 sub의 몸체의 코드를 작성하시오. 정적 변수를 응용할 것.

```
#include <stdio.h>
void sub(void);
int main(void)
{
        int i;
        for(i = 0;i < 4; i++)  sub();
        return 0;
}
void sub(void)
{
    //정의
}
```

```
호출횟수 1회
호출횟수 2회
호출횟수 3회
호출횟수 4회
```

9장 정리문제

1) 함수란 무엇인가?

2) 함수가 왜 필요한지 구체적인 예를 들어 설명하시오.

3) 함수의 정의, 선언, 호출의 의미를 설명하라.

4) 함수의 매개변수의 의미를 구체적인 예를 들어 설명하라.

5) 함수의 리턴값의 의미를 구체적인 예를 들어 설명하라.

6) 매개변수와 자동(지역)변수의 공통점과 차이점를 설명하라.

7) 함수가 호출될 때 컴퓨터에 의해 자동으로 처리되는 2가지 과정을 설명하라.

8) 함수의 선언이 필요한 이유를 설명하라.

9) printf, scanf함수의 선언을 찾아서 설명 하시오.

10)printf, scanf함수의 정의는 어디에 있는가?

9장 정리문제

11)과제게시판의 프로그램 개발과정 및 함수 강의노트를 복습하고 링크과정을 상세히 설명하시오. 링크과정이 왜 필요한가?

12) C언어로 작성된 소스파일을 빌드하면 기계어로 된 실행파일이 생성된다. 소스파일에 있는 문장중에 실행파일로 번역되지 않는 부분은?

13) 소스파일에는 없는데 실행파일에는 포함되는 코드는?

과제제출방법

- 소스코드, 라인단위의 주석, 실행결과를 포함하는 pdf파일을 작성한 후 eclass 과제 게시판에 업로드, **반드시 하나의 pdf파일로 업로드할 것**
- 기한 : 과제 게시판에 마감시간 참조
- 실행결과를 캡쳐할 때 글자를 알아보기 쉽게 확대해서 캡쳐할 것.
- 소스코드의 첫 부분은 아래처럼 제목,날짜,작성자(학번,이름)를 작성할 것

```
// ********************************************
//    제   목 : 정수 4개의 평균을 구하는 프로그램
//    날   짜 : 2023년 9월10일
//    작성자 : 15010101 홍길동
// ********************************************

// 소스코드 작성
```