













8.4 함수와 변수의 적용 범위를 이해하고, 변수의 종류에 대해 학습한다.











함수와 변수

- 함수는 특정한 일을 하는 독립된 단위
- 함수 기능 뿐만 아니라 함수에서 사용하는 변수에도 독립성이 적용
 ✓ 즉, 함수에서 선언된 변수들은 그 함수에서만 유효
- 경우에 따라서 특정 함수에만 국한되지 않고, 함수와 무관하게 사용되는 변수가 필요



변수종류

- 지역변수
- 전역변수
- 정적변수





지역 변수

- 선언 위치 : <u>함수 내에서 선언</u>
- 유효 범위: 변수를 선언한 **함수 내에서만(지역적으로) 유효**
- 함수 호출과 동시에 자동으로 생성되고 함수가 종료되면 자동으로 소멸되어 **자동변수**라고도 함
- **함수의 형식 인자도 지역변수**임
- 다음 코드에서 컴파일 오류가 발생하는 이유는?

```
int add (int x, int y) {
    int c;
    int c;
    c = x + y;
    return c;
}

int main()
{
    int c;
    c = add(3,4);
    · · ·
}
```





지역 변수의 독립성: 함수 구현에 독립성을 부여

■ 아래에서 두 함수의 변수 c는 서로 다른 변수

add()의 변수

X 3

y 4

c 7

add()함수
안에서만 사용 가능

main()의 변수
c 10
main()함수
안에서만 사용 가능

실행 결과 3 + 4 = 7 c = 10





전역 변수

- 선언 위치 : <u>함수 밖에서 선언</u>
- 유효 범위 : 프로그램 내 <u>어디서든 사용 가능</u>
- 자동으로 0으로 초기화
 - ✓ But, 모든 변수는 명시적으로 초기화 하는 습관을 가지자.

```
int c = 0; // 전역변수 선언
int add (int x, int y)
{
    c = x + y;
    return c;
}

int main()
{
    c = 10;
    printf("3 + 4 = %d\n", add(3,4));
    printf("c = %d\n", c);
    return 0;
}
```

실행 결과 3





동일한 이름의 전역변수와 지역변수

■ 지역 변수가 우선

```
int c = 0; // 전역변수

void add (int x, int y) {
    c = x + y;
    printf("add: c = %d\n", c);
}

int main() {
    int c = 10; // 지역 변수
    add(3,4);
    printf("main: c = %d\n", c);
    return 0;
}
```

실행 결과

add: c = 7 main: c = 10 add()의 변수

x 3

y 4

add()함수
안에서만 사용 가능

사용 가능

main()의 변수

c 10

main()함수
안에서만 사용 가능

모든 함수에서





전역 변수는 함수 사이의 데이터 전달을 위한 또 하나의 수단

✓ 아래 코드는 설명을 위한 예제로 좋은 방식의 코드는 아님

```
int c = 0; // 전역변수 선언
void add (int x, int y) {
   c = x + y;
int main() {
  add(3,4));
   printf("3 + 4 = %d\n", c);
   return 0;
```

실행 결과

$$3 + 4 = 7$$

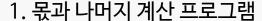
■ 전역 변수를 사용하면 함수 사이의 데이터 전달이 편리하나, **함수의 독립성을 해치므로 신중을 기해야** 함

실습 하기



[예제 8.3] 다음과 같이 함수를 정의하고 사용하시오.





- ✓ div() 함수
 - ✓ 반환형은 int, 인자는 int형 변수 2개
 - ✓ 인자 2개를 나눈 몫을 반환, 나머지는 전역변수에 저장
- ✓ main() 함수
 - ✓ 두 개의 정수를 입력 받고, div 함수를 호출하여 몫과 나머지 계산하여 한 줄에 출력

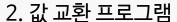
	입력 예시1	출력 예시1
5 3		1 2
	입력 예시2	출력 예시2
4 2		2 0





[예제 8.3] 다음과 같이 함수를 정의하고 사용하시오.





- ✓ swap() 함수
 - ✓ 반환형은 void형, 인자는 없음
 - ✓ 전역변수 a와 b의 값을 교환(즉, 두 개의 변수 값 바꾸기)
- ✓ main() 함수
 - ✓ 두 개의 정수를 입력 받아 전역 변수 a와 b에 저장
 - ✓ swap() 함수를 호출
 - √ 교환된 두 전역 변수의 값 출력

a 입력: 6 b 입력: 8

swap 함수 호출 후

a = 8

b = 6

실행 예시 (붉은 색은 사용자 입력)







변수의 지속시간

- ✓ 지역 변수 : <u>함수</u> 호출 시 생성, 함수 종료 시 소멸
- ✓ **전역 변수** : <u>프로그램</u> 실행 시 생성, 프로그램 종료 시 소멸

```
void inc_L() {
  int c = 0; ⇒ 지역 변수
  ++C;
  printf("%d\n", c);
int main(){
  inc_L(); ⇒ 첫 번째 호출
  inc_L(); ⇒ 두 번째 호출
  inc_L(); ⇒ 세 번째 호출
  return 0;
```

실행 결과 ⇒ 첫 번째 호출 ⇒ 두 번째 호출 ⇒ 세 번째 호출

실행 결과

```
1 ⇒ 첫 번째 호출
2 ⇒ 두 번째 호출
3 ⇒ 세 번째 호출
```

```
int c =0; ⇒ 전역 변수
void inc_G() {
  ++c;
  printf("%d\n", c);
int main(){
  inc G(); ⇒ 첫 번째 호출
  inc_G(); ⇒ 두 번째 호출
  inc_G(); ⇒ 세 번째 호출
  return 0;
```





정적 변수

- static 키워드 사용
 - ✓ 선언 위치와 유효 범위 : 지역 변수와 동일
 - √ 함수내 선언
 - ✓ 선언한 함수 내부에서만 사용 가능
 - ✓ 지속 시간 : 전역 변수와 동일
 - ✓ 프로그램 실행 전체 과정 동안딱 한번만 생성되고 초기화

실행 결과

```
1 ⇒ 첫 번째 호출
2 ⇒ 두 번째 호출
3 ⇒ 세 번째 호출
```

```
void inc_S() {
  static int c = 0; ⇒ 정적 변수
  ++C;
  printf("%d\n", c);
int main(){
  inc_S(); ⇒ 첫 번째 호출
  inc_S(); ⇒ 두 번째 호출
  inc_S(); ⇒ 세 번째 호출
```





함수와 관련된 변수 종류와 특징

	지역 변수	정적 (지역) 변수	전역 변수
선언위치	함수내부		함수외부
사용범위	선언한 함수 내부에서	프로그램 내 어디서든지 사용가능	
자동 <i>초</i> 기화	X (사용자가 직접 초기화)	사용자가 직접 초기화) O (0으로 자동 초기화)	
지속시간	함수호 출될 때마다생성, 해당함수종료시소멸	프로그램이 실행 동안 단 한번 생성, 프로그램 종료 시 소멸	



학습 정<mark>리</mark>



- 함수 내부에서 선언된 변수는 지역 변수이고, 선언된 함수 내에서만 유효함
- 지역 변수는 함수가 호출되면 자동으로 생성되고, 함수가 종료되면 자동으로 사라져 자동 변수라고도 함
- 함수 외부에서 선언된 변수를 전역 변수라 하고, 모든 함수에서 유효함
- 정적 변수의 유효 범위는 지역 변수와 동일하고, 지속 시간은 전역 변수와 동일함