작성자 : 김기희 (cupid072@naver.com)

변수의 참조범위 (Variable Scope)

- 학습목표
  - 기억 클래스(Storage Class)에 대해 설명할 수 있다.
  - 기억 클래스(Storage Class)의 종류를 나열할 수 있다.
  - 자동 변수의 참조 범위, 존속 기간, 자동 초기화 여부를 설명할 수 있다.
  - 외부 변수의 참조 범위, 존속 기간, 자동 초기화 여부를 설명할 수 있다.
  - 정적(내/외부) 변수의 참조 범위, 존속 기간, 자동 초기화 여부를 설명할 수 있다.
  - 레지스터 변수의 특징을 설명할 수 있다.

기억 클래스(Storage class)

- 기억 클래스(Storage class)란?

  변수의 유효 범위(Variable Scope), 생존기간(Life time), 사용 메모리, 자동 초기화 여부 등 프로그램 내에서 사용되는 변수들의 특성을 결정하는 요소

- 기억 클래스 사용 형식

  Storage_class 자료형 변수명;

- 예)

  auto int age;

  extern double height;

기억 클래스(Storage class)의 종류

- 기억 클래스의 종류

지정자		유효범위	존속기간	사용 메모리	자동 초기화
auto(자동변수)		Block	자동	Stack	X
extern(외부변수)		Program	정적	Data	O
static(정적변수)	internal	Block	정적	Data	O
	external	File			
register(레지스터 변수)		Block	자동	CPU Register	X

자동(auto) 변수

- 함수 내부 또는 블록 내에서 선언되는 변수로 지역(local) 변수라고도 한다.
- 함수 스택 프레임내의 공간을 할당 받으며 함수 종료 시 함수 스택 프레임 삭제와 함께 소멸된다.
- 자동 변수는 자동으로 초기화 되지 않는다.
- 함수 내부 또는 블록 내에서 변수 선언 시 auto 지정자를 지정하여 선언한다.

  auto int age = 25;

- 변수 선언 시 auto 지정자를 생략하여도 auto 변수가 된다.

  int age = 25; // auto int age = 25; 와 동일

- 변수를 참조할 수 있는 범위는 변수가 선언된 함수 또는 블록으로 제한된다.
- 자동 변수를 전역 변수와 동일한 이름으로 선언하는 경우 전역 변수는 자동 변수에 의해 가려진다.

자동(auto) 변수

```
int height = 10;                    // 함수 밖에서 선언되었으므로 extern 변수

void func() {
    printf("%d\n", height);         // 동일 블록 내에 변수 height가 선언되지 않았으므로
}                                   // 함수 블록 외부의 변수 height를 참조

int main() {
    int height = 20;                // 함수 내부에서 선언되었으므로 auto 변수
    printf("%d\n", height);         // 동일 블록 내에서 선언된 변수 height를 참조
    func();
    {
        int height = 100;           // 블록 내에서 선언되었으므로 auto 변수
        printf("%d\n", height);     // 동일 블록 내에서 선언된 변수 height를 참조
    }
    return 0;
}
```

외부(extern) 변수

- 파일 내, 함수 외부에 선언되는 변수로써 프로그램 전체에서 참조가 가능하므로 전역(global) 변수라고도 한다.
- 외부 변수는 DATA영역 또는 BSS영역을 할당 받으며 프로그램 시작 시 할당되고 프로그램 종료 시까지 존재한다.
- 외부 변수는 0 값으로 자동으로 초기화 된다.
- 외부 변수는 변수의 선언(Declare)과 정의(Define)를 구분하여 사용한다.
- extern 지정자는 외부 변수 선언(Declare) 시에만 사용하며 현재 파일 또는 외부 파일에서 선언되고 초기화(define) 된 변수를 참조할 것임을 나타낸다.
- 외부 변수를 선언(Declare)과 동시에 초기화(Define)를 하는 경우 extern 지정자를 사용하지 않는다.
- 외부 변수의 중복 선언(Declare)은 가능하지만 초기값(Define)을 가지는 중복 선언은 허용되지 않는다.
- extern 지정자를 사용하지 않은 외부 변수의 선언이 여러 번 기술되는 경우 선언된 외부 변수 중 오직 하나만 초기값 0인 외부 변수로 선언 및 초기화(Define) 되며 나머지는 선언(Declare)이 된다.

외부(extern) 변수

- 아래 예제의 외부 변수 height는 extern 지정자를 사용하지 않았으므로 extern 변수의 선언이 된다.
- 아래 예제의 외부 변수 height 중 하나 만 0으로 초기화(Define)되는 가정의가 되며 나머지는 외부변수의 선언(Declare)이 된다.

```
#include <stdio.h>
int height;
int height;
int height;

int main() {
  printf("%d\n", height);
  return 0;
}
```

외부(extern) 변수

- extern 지정자를 사용한 외부 변수의 선언(Declare)은 중복이 허용된다.

```
#include <stdio.h>
extern int height;              // Declare
extern int height;              // Declare
int height;                     // Declare   extern이 생략되었으므로 extern
int height = 100;               // Define    초기값이 지정된 변수의 선언

int main() {
  printf("%d\n", height);
  return 0;
}
```

외부(extern) 변수

- extern 지정자를 사용한 외부 변수의 선언(Declare)은 중복이 허용된다.

```
#include <stdio.h>
extern int height = 100;          // extern 변수 선언에 초기 값 지정 (경고 메시지 출력)
int weight = 100;
int weight = 100;                 // extern 변수 선언 및 초기화(Define)가 중복되었으므로 에러

int main() {
  printf("%d\n", height);
  printf("%d\n", weight);
  return 0;
}
```

외부(extern) 변수

main.c

```
extern void print();
extern int a;
int b;
int c = 300;

int main() {
    printf("%d\n", a);
    printf("%d\n", b);
    printf("%d\n", c);
    print();
    return 0;
}
```

sub.c

```
int a = 100;
int b = 200;
extern int c;

void print() {
    printf("%d\n", a);
    printf("%d\n", b);
    printf("%d\n", c);
}
```

정적(static) 변수

- 정적 변수는 파일 내부 또는 함수나 블록 내부에서 선언되는 변수로써 static 지정자를 사용하여 선언한다.
- 정적 변수는 DATA Segment를 할당 받아 사용되며 자동으로 초기화 된다.
- 파일 내, 함수 외부에 선언된 정적 변수를 외부 정적(external static) 변수라고 한다.
- 함수 내부나 블록 내부에서 선언된 변수를 내부 정적(internal static) 변수라고 한다.
- 정적 변수는 프로그램 시작과 함께 공간을 할당 받고 프로그램 종료시까지 유지된다.
- 정적 변수는 공간 할당과 동시에 초기화가 이뤄지며 메모리 할당 시 오직 한 번만 초기화 된다.
- 정적 변수는 외부 변수와 마찬가지로 중복 선언(Declare)은 허용되지만 중복 정의(Define)은 허용되지 않는다.
- 정적 변수가 중복되어 선언(Declare)되는 경우 중복된 선언 중 하나만 가정의(Define)가 되며 나머지는 선언(Declare)이 된다.
- 외부 정적 변수는 변수가 선언된 파일의 모든 곳에서 참조가 가능하며 외부에서는 참조가 불가능하다.
- 내부 정적 변수는 변수가 선언된 함수나 블록 내에서만 참조가 가능하며 변수가 선언된 함수나 블록 외부에서 참조는 불가능하다.

정적(static) 변수

- 중복하여 선언(Declare)하는 경우 하나만 가정의(Define)가 되며 나머지는 선언(Declare)이 된다.

```
#include <stdio.h>

static int height;          // 가정의(Define)
static int height;          // 선언(Declare)
static int height;          // 선언(Declare)

int main() {
  printf("%d\n", height);
  return 0;
}
```

정적(static) 변수

- 초기 값을 지정한 선언은 중복이 허용되지 않는다.

```
#include <stdio.h>

static int height = 10;          // 정의(Define)
static int height = 20;          // 정의(Define) 중복정의가 되므로 에러
static int height;               // 선언(Declare)

int main() {
  printf("%d\n", height);
  return 0;
}
```

정적(static) 변수

main.c

```
#include <stdio.h>
extern void print();
static int a = 10;          // main.c 내에서만 참조 가능
static int b = 20;          // main.c 내에서만 참조 가능

int main() {
    printf("%d\n", a);
    printf("%d\n", b);
    print();
    return 0;
}
```

sub.c

```
#include <stdio.h>
static int a = 100;         // sub.c 내에서만 참조 가능
int b = 200;                // 프로그램 전체에서 참조 가능

void print() {
    printf("%d\n", a);
    printf("%d\n", b);
}
```

정적(static) 변수

main.c

```
#include <stdio.h>
extern void print();
extern int c;
static int a = 10;          // main.c 내에서만 참조 가능
static int b = 20;          // main.c 내에서만 참조 가능
static int c = 100;         // 중복 선언이므로 에러

int main() {
    printf("%d\n", a);
    printf("%d\n", b);
    printf("%d\n", b);
    print();
    return 0;
}
```

sub.c

```
#include <stdio.h>
static int a = 100;         // sub.c 내에서만 참조 가능
int b = 200;                // 프로그램 전체에서 참조 가능
int c = 300;

void print() {
    printf("%d\n", a);
    printf("%d\n", b);
    printf("%d\n", b);
}
```

정적(static) 변수

```
#include <stdio.h>

void func() {
    static int count = 0;                    // static 변수는 메모리 할당 시 오직 한 번만 초기화가 된다.
    count++;
    printf("호출 횟수 : %d\n", count);
}

int main() {
    func();
    func();
    func();
    func();
    return 0;
}
```

아래 함수의 호출 결과는?

```
void func() {
    static int count;
    count = 0;
    count++;
    printf("호출 횟수 : %d\n", count);
}
```

레지스터(register) 변수

- 레지스터(register) 변수는 CPU의 명령 레지스터 중 하나를 할당 받아 사용한다.
- 레지스터 변수는 register 키워드를 사용하여 선언한다.
- 외부 변수(전역 변수)로 선언할 수 없으며 오직 함수나 블록 내에서만 선언할 수 있다.
- register 지정자를 사용하여 레지스터 변수를 선언한다 하여도 실제로 명령어 레지스터를 할당되는가, 스텍 영역을 할당되는 가는 컴파일러에 의해 결정된다.
- register 변수를 사용하면 메모리를 사용하는 변수에 비해 빠른 연산 속도를 얻을 수 있다.
- register 변수를 사용하면 변수에 할당된 레지스터를 다른 연산에 사용할 수 없으므로 전체 적인 프로그램 속도가 저하될 수 있다.
- 그 밖에 register 변수는 다음의 제약이 따른다.
  - 레지스터 변수는 주소를 구할 수 없다.
  - 레지스터 배열에 대하여 주소연산을 할 수 없다.
  - 함수나 블록 범위에서만 사용 가능하다.