

C 기초 특강

함수

함수의 정의

▶ 일반적인 형식

```
type function_name ( parameter_list )
{
    declarations (선언부)

    statements (문장)
}
```

▶ 예

```
int factorial(int n)          /* header */
{
    int i, product = 1;      /* body */
                             /* declarations */

    for (i = 2; i <= n; i++) /* statements */
        product *= i;
    return product;
}
```

▶ type이 생략되는 경우 C에서는 default로 int

매개변수(parameter)

- ▶ 정의 : 함수를 호출하는 함수와 호출되는 함수 사이에서 정보를 전달하는 매개체 역할을 하는 변수

```
int add_n(int n) ← 매개변수
{
    int i ← 일반 지역변수
    int sum ← 0;

    for (i = 0; i < n; i++) {
        printf("enter a integer : ");
        scanf("%d", &input);
        sum += input;
    }
    return sum;
}
```

void 이용

- ▶ return type의 void
 - ▶ return 되는 값이 없음을 의미
- ▶ parameter_list 의 void
 - ▶ 함수 호출시 괄호 안에 아무것도 없음을 의미

▶ 예

```
void f(void)
{

}
```

함수 호출(call)과 반환(return)

▶ 함수 호출

- ▶ 함수를 호출하는 함수는 현재의 함수 상태(레지스터 상태, 변수들의 값, 다음 코드 주소 등)를 스택에 저장
- ▶ 호출된 함수로 제어를 넘기고 대기

▶ 결과 반환

- ▶ 호출된 함수가 끝나면(몸체의 끝에 도달하거나 return 문장을 만나면) 호출한 함수에게 넘기고 수행에 관련된 정보 삭제
- ▶ return 값이 있는 경우 호출한 함수에게 전달

함수 호출과 반환 예

```
#include <stdio.h>
int prod(int, int);
void main()
{
    int a, b, c;
    a = 165, b=2;
    c= prod(a, b);
    printf("%d * %d = %d\n", a, b, c);
}
```

```
int prod( int x, int y)
{
    int z;

    z = x*y;
}
```

함수 호출 처리 단계

- ▶ 인자 목록의 각 수식 수행
- ▶ 필요한 경우 형식 매개변수의 형으로 변환되어 몸체 시작부분에서 형식 매개변수에 할당
- ▶ 함수 몸체 실행
- ▶ `return`을 만나면 호출한 환경으로 되돌아감
- ▶ `return`이 수식을 가지고 있는 경우 수식 수행, 필요한 경우 함수의 형으로 변환 후 호출한 환경으로 되돌아감
- ▶ `return`이 없는 경우 함수 몸체 끝에 도달하면 호출한 환경으로 되돌아감

함수 인자(argument)

- ▶ 정의 : 함수 호출시 전달되는 자료
 - ▶ 변수, 상수, 포인터 등 다양한 형태

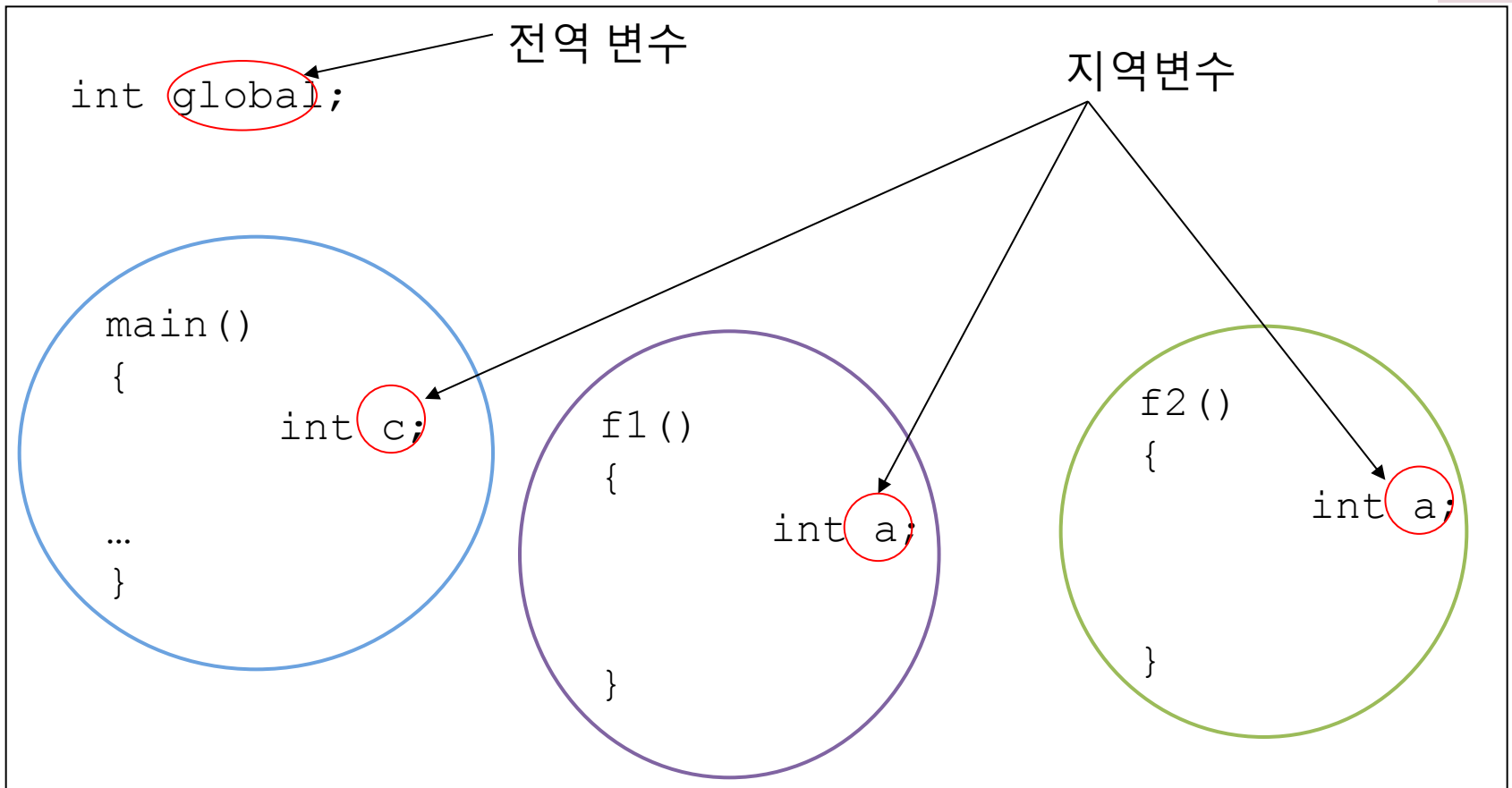
```
int main(void)
{
    int a, b, c, d;
    read_integer(&a, &b);
    c = add(a, b);
    d = average(a, 100);
    output(c, d);
    return 0;
}
```

주소(포인터)

변수

상수

지역(local) 변수와 전역(global) 변수



기억영역 클래스

- ▶ auto
 - ▶ block 내부 선언
- ▶ extern
 - ▶ 함수 밖에서 선언
 - ▶ default 초기화
- ▶ register
- ▶ static
 - ▶ default 초기화
 - ▶ 함수 안 : 변수의 값 유지
 - ▶ 함수 밖 : 비공개(함수나 변수의 가시화 또는 유효범위의 제한)

함수 원형

- ▶ 일반적인 형식

```
type function_name( parameter_type_list );
```

- ▶ 파일 이름과 반환되는 값의 타입, 매개변수 개수와 타입을 컴파일러에게 알려 줌

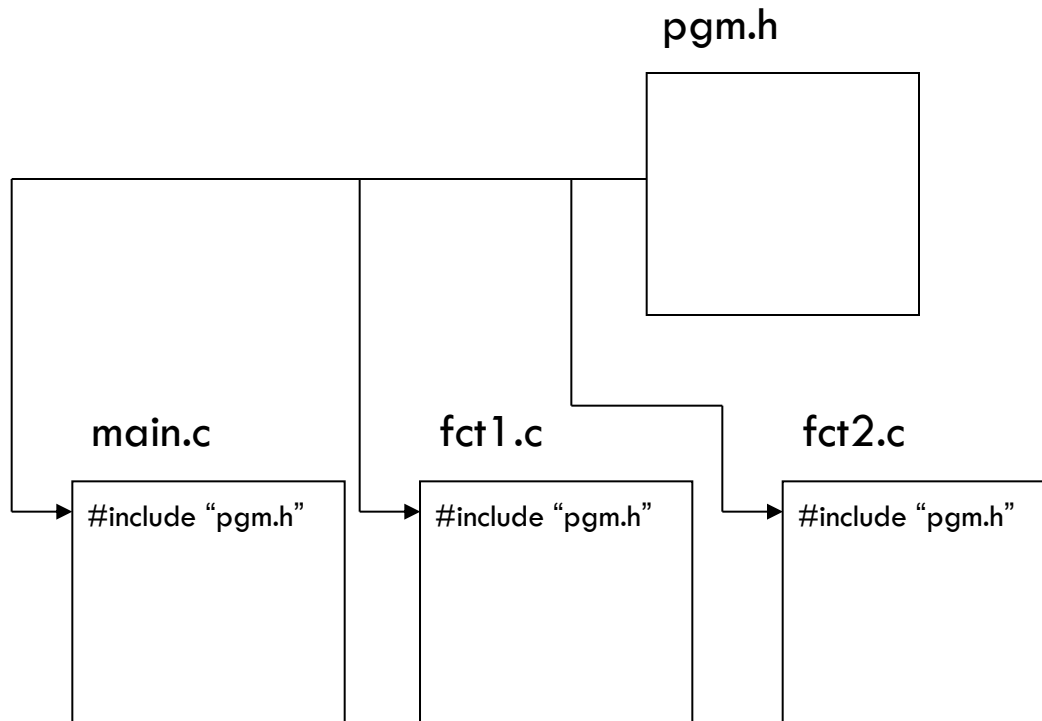
- ▶ 이들 정보를 바탕으로 컴파일러가 코드를 더 철저히 검사

- ▶ 예 (효과는 동일)

- ▶ void f (char c, int i);

- ▶ void f (char, int);

대형 프로그램 개발



Makefile

```
pgm : pgm.h main.o fct1.o fct2.o
    gcc -o pgm main.o fct1.o fct2.o

main.o: pgm.h main.c
    gcc -c main.c

fct1.o: pgm.h fct1.c
    gcc -c fct1.c

fct2.o: pgm.h fct2.c
    gcc -c fct2.c
```

과제

- ▶ 소수 여부를 판정하는 함수 및 테스트 프로그램 작성
 - ▶ 정수를 전달받아 소수이면 1, 소수가 아니면 0을 반환하는 `int is_prime(int)` 함수 작성
 - ▶ 작성한 함수를 확인하는 프로그램 작성
 - ▶ 사용자로부터 반복적으로 정수 입력을 받아 소수 여부를 판정하고, 정수가 아닌 입력이 들어오면 그동안 처리한 횟수를 출력하는 프로그램 작성
- ▶ Goldbach 추측 (“2보다 큰 모든 짝수는 두개의 소수의 합으로 표현될 수 있다.”)을 확인하는 프로그램 작성 (힌트 : `is_prime()` 이용)
 - ▶ 사용자에게 범위(시작값, 종료값)를 입력받는다.
 - ▶ 범위에 속한 모든 짝수에 대해 두 개의 소수의 합으로 표현하여 출력
 - ▶ 예) $700 = 17 + 683$