Chapter 4
Function and Program Structures
함수 + 프로그램구조

Part 2

CSE2018 시스템프로그래밍기초
2016년 2학기

한양대학교 ERICA
컴퓨터공학과 => 소프트웨어학부
도경구

Scope Rules

자동(내부) 변수, 파라미터

- 함수의 시작 부분에서 선언한 변수로 사용 영역은 함수 내부

외부 변수, 함수

- 영역은 선언지점부터 파일의 끝까지
- 앞에서 선언한 이름은 뒤에서 쓸 수 없음
- 뒤에서 선언한 이름을 앞에선 모름
- 그러나 정의하기 전에 사용해야 한다면 external을 붙여 선언해야 함
- 다른 파일에서 정의한 외부변수를 써야 한다면 external을 붙여 선언해야 함

선언 vs 정의

정의

- 변수의 값을 저장할 장소를 정한다.
- 한번만 정의함
- 배열의 크기는 명시해야 함
- 초기 값은 정의에서만 지정 가능

```
int sp;
double val[MAXVAL];
```

선언

- 변수의 타입과 특징을 정한다.

```
extern int sp;
extern double val[];
```

Header File

- 프로그램이 길어지는 경우 기능 별로 여러 파일로 나누면 편리
- 이 때 공유하는 정의와 선언은 파일 하나에 따로 모아두면 관리가 편리함
- 이를 헤더파일이라고 하고 파일 확장자 *.h 로 구분함

calc.h:

```
#define NUMBER '0'
void push(double);
double pop(void);
int getop(char []);
int getch(void);
void ungetch(int);
```

main.c:

```
#include <stdio.h>
#include <stdlib.h>
#include "calc.h"
#define MAXOP 100
main() {
    ...
}
```

getop.c:

```
#include <stdio.h>
#include <ctype.h>
#include "calc.h"
getop() {
    ...
}
```

stack.c:

```
#include <stdio.h>
#include "calc.h"
#define MAXVAL 100
int sp = 0;
double val[MAXVAL];
void push(double) {
    ...
}
double pop(void) {
    ...
}
```

getch.c:

```
#include <stdio.h>
#define BUFSIZE 100
char buf[BUFSIZE];
int bufp = 0;
int getch(void) {
    ...
}
void ungetch(int) {
    ...
}
```

Static Variable

stack.c

```
#include <stdio.h>
#include "calc.h"
#define MAXVAL   100    /* maximum depth of val stack */

static int sp = 0;          /* next free stack position */
static double val[MAXVAL]; /* value stack */

void push(double f) {
    if (sp < MAXVAL)
        val[sp++] = f;
    else
        printf("error: stack full, can't push %g\n", f);
}

double pop(void) {
    if (sp >0)
        return val[--sp];
    else {
        printf("error: stack empty\n");
        return 0.0;
    }
}
```

sp와 **val** 이름은
push 와 pop 함수를 사용하는 파일 바깥에서 볼 수 없다

Static Function

함수 선언 앞에 **static**을 붙이면
파일 내부 전용으로 사용하고
파일 외부 노출이 되지 않는다.

Static Internal Variable

내부(지역) 변수 선언 앞에 **static**을 붙이면
함수 호출 종료 후에도
없어지지 않고 남아 있다.
즉, 함수 전용으로 사용하는 영구 변수이다.

Register Variable

변수 선언 앞에 **register**을 붙이면, 그 변수는 레지스터에 할당된다.
빈번히 쓰는 변수는 레지스터에 저장하여 읽고 쓰는 속도를 줄일 수 있다.
즉, 실행 속도를 향상할 수 있다.
내부(지역) 변수 또는 파라미터 만 사용가능

```
register int x;
register char c;
```

```
void f(register unsigned m, register long n) {
    register int i;
    . . .
}
```

사용 횟수 또는 타입에 제한이 있을 수 있다.
하지만 register에 할당되지 않을 뿐이므로 실행에 지장이 있는 건 아니다.

Block Structure

블록 내부에 선언한 변수와 파라미터 변수의 영역

```
if (n > 0) {
    int i;  /* declare a new i */

    for (i = 0; i < n; i++)
        . . .
}
```

```
int x;
int y;

void f(double x) {
    double y;
    . . .
}
```

함수 선언 내부에 함수를 선언할 수 없다.

Initialization

외부external 변수, 정적static 변수

- 명시적으로 언급하지 않아도 저절로 0으로 초기화된다.
- 명시적으로 초기화를 하는 경우, 반드시 상수식이어야 한다.

내부internal 변수, 레지스터register 변수

- 명시적으로 초기화하지 않으면 값이 미정undefined 이다.
- 초기화하는 식이 상수식일 필요는 없다.

선언하면서 바로 초기화 가능

```
int x = 1;
char squote = '\'';
long day = 1000L * 60L * 60L * 24L;
```

```
int binsearch( . . , int n) {
    int low = 0;
    int high = n - 1;
    int mid;
    . . .
```

=

```
int binsearch( . . , int n) {
    int low, high, mid;

    low = 0;
    high = n - 1;
    . . .
```

Initialization

배열array 초기화

```
int days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

- 배열의 크기를 굳이 명시하지 않으면, 초기화 값의 개수만큼으로 배열의 크기를 자동으로 정한다.
- 배열의 크기를 명시할 수도 있다.
- 초기화 값의 개수가 배열의 명시한 크기보다 작으면 남은 부분은 모두 0으로 채운다.
- 초기화 값의 개수가 배열의 명시한 크기보다 많으면 오류이다.
- 일부만 초기화하는 건 불가능하다.

문자 배열array 초기화

```
char pattern[] = "ould";
```

=

```
char pattern[] = { 'o', 'u', 'l', 'd', '\0' };
```

The C Preprocessor

- 컴파일하기 전에 하는 전처리 작업

File Inclusion

`#include "filename"`

- *filename* 파일의 내용으로 대치한다.
- 그 파일은 동일 폴더에서 우선 찾는다.
- 그 폴더에 없으면 파일을 찾는 규칙에 따른다. (컴파일러에 따라 규칙이 다를 수 있음)

`#include <filename>`

- *filename* 파일의 내용으로 대치한다.
- 파일을 찾는 규칙에 따른다. (컴파일러에 따라 규칙이 다를 수 있음)

- 포함할 파일에 넣는 것들
  - 공통으로 쓰는 #define 문
  - 공통으로 쓰는 extern 선언
  - 공통으로 쓰는 함수 프로토타입 선언

The C Preprocessor

- 컴파일하기 전에 하는 전처리 작업

Macro Substitution

```
#define name replacement-text
```

- 이후에 나오는 *name* 은 모두 *replacement-text* 로 대치한다.
  - *name* 작성 요령은 변수와 동일
  - *replacement-text*그은 뭐든지 상관 없음
- 이름 사용가능 영역은 파일의 정한 지점부터 파일의 끝까지

```
#define forever  for (;;)  /* infinite loop */
```