Week 5

Chapter 4
Function and Program Structures

함수 + 프로그램구조

CSE2018 시스템프로그래밍기초
2016년 2학기

한양대학교 ERICA
컴퓨터공학과 => 소프트웨어학부
도경구

함수
Function

Function Abstraction
작업을 작게 함수로 분리하여 부품화하는 작업
(세부 내역은 숨김)

프로그램은 여러 파일로 분산하여 작성하고
파일별로 따로 컴파일(compile)한 다음
미리 컴파일해놓은 라이브러리 함수와 함께 모두 같이 장학(load)하여 실행

Definition
정의

```
return-type function-name ( argument declarations ) {
    declarations and statements
}
```

`return`
statement

```
return expression;
```

```
int dummy() {}
```

특정 문자열 패턴이 있는 줄 프린트하기
File Copying

= grep

while *(there's another line)*
    if *(the line contains the pattern)*
        print it

main.c

```
#include <stdio.h>
#define MAXLINE 1000 /* maximum input line length */

int readline(char line[], int max);
int strindex(char source[], char searchfor[]);

char pattern[] = "ould"; /* pattern to search for */

/* find all lines matching pattern */
int main() {
    char line[MAXLINE];
    int found = 0;

    while (getline(line, MAXLINE) > 0)
        if (strindex(line, pattern) >= 0) {
            printf("%s", line);
            found++;
        }
    return found;
}
```

readline.c

```
#include <stdio.h>

/* readline: read line into s, return length */
int readline(char s[], int lim) {
    int c, i;

    i = 0;
    while (--lim > 0 && (c = getchar()) != EOF && c != '\n')
        s[i++] = c;
    if (c == '\n')
        s[i++] = c;
    s[i] = '\0';
    return i;
}
```

strindex.c

```
/* strindex: return index of t in s, -1 if none */
int strindex(char s[], char t[]) {
    int i, j, k;

    for (i = 0; s[i] != '\0'; i++) {
        for (j = i, k = 0; t[k] != '\0' && s[j] == t[k]; j++, k++)
            ;
        if (k > 0 && t[k] == '\0')
            return i;
    }
    return -1;
}
```

실수문자열을 실수로 바꾸기
Convert a string into its floating-point equivalent

= atof in <stdlib.h>

```c
#include <stdio.h>
#include <ctype.h>

/* atof: convert string s to double */
double atof(char s[]) {
    double val, power;
    int i, sign;

    for (i = 0; isspace(s[i]); i++)
        ;
    sign = (s[i] == '-') ? -1 : 1;
    if (s[i] == '+' || s[i] == '-')
        i++;
    for (val = 0.0; isdigit(s[i]); i++)
        val = 10.0 * val + (s[i] - '0');
    if (s[i] == '.')
        i++;
    for (power = 1.0; isdigit(s[i]); i++) {
        val = 10.0 * val + (s[i] - '0');
        power *= 10.0;
    }
    return sign * val / power;
}
```

```c
#include <stdio.h>

#define MAXLINE 100

/* rudimentary calculator */
int main() {
    double sum, atof(char []);
    char line[MAXLINE];
    int readline(char line[], int max);

    sum = 0;
    while (readline(line, MAXLINE) > 0)
        printf("\t%g\n", sum += atof(line));
    return 0;
}
```

실수문자열을 정수로 바꾸기
Convert a string into its integer equivalent

= atoi in <stdlib.h>

```
/* atoi: convert string s to integer using atof */
int atoi(char s[]) {
    double atof(char s[]);

    return (int) atof(s);
}
```

External Variable

외부 변수

- 외부 변수는 함수 정의 바깥에 선언하는 변수
- 파일 내부 함수 어디에서든지 접근 가능

함수 선언 내부에 함수를 선언할 수 없다.

Reverse Polish notation Calculator

Example

infix	`(1 - 2) * (4 + 5)`
reverse Polish (postfix)	`1 2 - 4 5 *`

Algorithm

```
while (next operator or operand is not end-of-file indicator)
    if (number)
        push it
    else if (operator)
        pop operands
        do operation
        push result
    else if (newline)
        pop and print top of stack
    else
        error
```

Outlines of implementation

```
#includes
#defines

function declarations for main
int main() { . . . }

external variables for push and pop
void push(double f) { . . . }
double pop(void) { . . . }

int getop(char s[]) { . . . }

routines called by getop
```

Let's code!

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
- 다름 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

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.

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