KU KONKUK UNIVERSITY

Chapter 8 & 9

DSLAB

DEPENDABLE SOFTWARE LABORATORY

Chapter 8

실습 문제 1

➢ **함수의 호출 (File: 8_1.c)**

```c
#include <stdio.h>

void f1(void);
void f2(void);

int main(void)
{
    int i;

    f1();
    for(i = 0;i < 3; i++)
        f2();
    f1();

    return 0;
}

void f1(void)
{
    printf("/**********************/\n");
}
void f2(void)
{
   printf("/*                    */\n");
}
```

(a) 이 프로그램의 실행 결과를 예측하여 보라. 실제로 실행하여 예측된 실행 결과와 비교하여 보라.

(b) 함수들의 이름을 의미있는 이름으로 변경하여 보라.

(c) 함수가 시작되면 무조건 함수의 이름을 화면에 출력하도록 프로그램을 수정하라. 어떤 출력이 생성되는가?

실습 문제 2

➢ 매개 변수가 없는 함수의 작성 및 호출 (File: 8_2.c)

```c
#include <stdio.h>

void print_header(void);

int main(void)
{
    print_header();
    return 0;
}

void print_header(void)
{
    ____________________
    ____________________
}
```

(a) 다음과 같은 출력 화면을 생성하도록 print_header() 함수의 몸체를 채워서 완성하라.

출력화면

```
/*****************************/
/* 학번: 11111111            */
/* 이름: 홍길동              */
/*****************************/
```

(b) 이 프로그램에서 다음과 같은 출력 화면을 얻으려면 어떻게 수정하면 되는가? 함수를 사용하는 경우와 함수를 사용하지 않는 경우를 비교하여 보라.

출력화면

```
/*****************************/
/* 학번: 11111111            */
/* 이름: 홍길동              */
/*****************************/
/*****************************/
/* 학번: 11111111            */
/* 이름: 홍길동              */
/*****************************/
```

실습 문제 3

➢ **매개 변수가 있는 함수의 작성 및 호출 (File: 8_3.c)**

```c
#include <stdio.h>

//함수 원형 정의

int main(void)
{
    float a, b, c;

    a = 2.0f;
    b = 3.0f;
    c = //a, b를 인수로 average() 호출
    // ①
    printf("평균은 %f입니다.\n", c);
    // ③
}

float average(float x, float y)
{
    // ②
    //평균을 계산하여 반환
}
```

(a) 위의 프로그램의 빈 칸에 필요한 문장들을 채워보자. 위의 프로그램을 컴파일하고 실행하여 보라. 출력을 기록하라.

(b) `scanf()`를 이용하여 사용자에게서 받은 두개의 실수의 평균을 구하도록 위의 프로그램을 수정하여 보라.

실습 문제 4

➢ 매개 변수가 있는 함수의 작성 및 호출 (File: 8_4.c)

```c
#include <stdio.h>

int main(void)
{
    int i, sum = 0;

    for(i = 0;i <= 100; i++)
        sum += i;

    printf("0부터 %d까지의 합은 %d입니다.\n", n, sum);
    return 0;
}
```

```c
#include <stdio.h>

_____________________________        // 함수 원형 정의

int main(void)
{
    int sum = 0;

    _____________________________    // 함수 호출
    printf("0부터 100까지의 합은 %d입니다.\n", sum);
}

_____________________________        // 함수 헤더 정의
{
    int i, result = 0;
    _____________________________
    _____________________________
    _____________________________
    _____________________________
}
```

(a) 함수의 이름은 get_sum()이라고 하고 int형의 정수 n을 받아서 0에서부터 n까지의 합을 구하여 반환한다고 가정하자. 위의 프로그램의 빈칸을 채워서 완성하여 보라. 컴파일하고 실행하여 보라.

(b) 만약 0에서 100까지의 합과 0에서 200까지의 합을 차례대로 계산한다고 할 때, 함수를 사용하는 프로그램과 함수를 사용하지 않는 프로그램은 어떤 차이점이 있는가? 매개 변수를 사용하는 장점은 무엇인가?

(c) get_sum() 함수가 2개의 인수 s, e를 받아서 s에서 e까지의 정수의 합을 계산하여 반환하도록 수정하여 보라. 수정된 get_sum()을 호출하여 10에서 100까지의 합을 계산하여 보라.

실습 문제 5

➢ **함수와 관련된 오류 (File: 8_5.c)**

```c
#include <stdio.h>

double fpower(double r, int n);          // ①

int main(void)
{
    double prod;

    prod = fpower(3.0, 2);               // ②
    printf("%f\n", prod);
}

double fpower(double r, int n)
{
    int i;
    double result = 1.0;

    for(i = 0; i < n; i++)
        result *= r;

    return result;
}
```

(a) 위의 프로그램을 컴파일하고 실행하여 보라. 어떤 출력이 생성되는가?

(b) 문장 ①의 함수 원형 선언에서 끝에 있는 세미콜론을 제거하고 컴파일 하여 보라. 어떤 일이 발생하는가?

(c) 문장 ①의 함수 원형 선언에서 반환형을 나타내는 double을 제거하고 컴파일 하여 보라. 어떤 일이 발생하는가?

(d) 문장 ②에서 함수 호출을 할 때 fpower(3.0); 와 같이 하나의 인수만을 가지고 호출하여 보라. 어떤 일이 발생하는가?

(e) 함수 원형을 완전히 제거한 후에 컴파일하여 실행하여 보라. 어떤 일이 발생하는가?

(f) return 문장을 다음과 같이 변경하면 어떻게 되는가?

return (result);

실습 문제 6

➢ 함수 원형 (File: 8_6.c)

```c
#include <stdio.h>

double divide(double n, double d);  // ①

int main(void)
{
    double result;

    result = divide(3.0, 10.0);             // ②
    printf("나눗셈 결과는 %f입니다.\n", result);
}

double divide(double n, double d)
{
    return n / d;
}
```

(a) 위의 프로그램을 컴파일하고 실행하여 보라. 출력을 기록하라.

(b) 위의 함수 원형이 정의된 문장 ①을 주석 처리한 후에 컴파일하여 보라. 어떤 오류가 발생하는가? 오류 메시지를 해석하고 설명하여 보라. 실습 후 원상태로 복구하라.

(c) 함수 원형 정의를 다음과 같이 변경하고 실행하여 보라. 어떤 차이점이 있는가?

```c
double divide(double, double);      // ①
```

(d) 문장 ②를 `result = divide(3, 10);` 으로 변경하여 실행하여 보라. 함수의 인수로 실수 대신 정수를 주었는데도 같은 결과가 생성되는 이유는 무엇인가?

(e) 문장 ②를 `result = divide(3, 10);` 으로 변경한 상태에서 함수 원형 정의를 다음과 같이 변경하여 보라. 즉 이번에는 함수의 인수에 대해서는 정의하지 않는다. 어떤 결과가 생성되는가? 실습 후 원상태로 복구하라.

```c
double divide();  // ①
```

(f) 위의 함수 원형이 정의된 문장 ①을 `main()` 함수 안으로 이동하여서 컴파일, 실행하여 보라. 차이점은 무엇인가? 실습 후 원상태로 복구하라.

(g) `divide()` 함수 안에서 매개 변수 `d`가 0이 아닌 경우에만 나눗셈이 되도록 코드를 수정하여 보라.

(h) 함수 원형 정의를 제거하고 `divide()` 함수의 위치와 `main()` 함수의 위치를 서로 바꾼 후에 프로그램을 컴파일하고 실행하여 보라. 어떤 차이점이 있는가?

(i) 같은 함수 원형을 여러 번 나열해도 문제가 없는지를 확인하기 위하여 문자 ①을 반복하여 보라. 어떤 결과가 나타나는가? 그리고 그 이유는 무엇인가?

```c
double divide(double n, double d);  // ①
double divide(double n, double d);  // ①
```

실습 문제 7

➢ 함수의 반환 값 (File: 8_7.c)

```c
#include <stdio.h>

int get_max2(int x, int y);
int get_max3(int x, int y, int z);

int main(void)
{
    int max;

    max = get_max3( 1, 2, 3 ); // 정수 1, 2, 3 중에서 가장 큰 수를 반환
    printf("가장 큰 수는 %d입니다.\n", max);

    return 0;
}

int get_max2(int x, int y)
{
    if( x > y )
    _____________________________
    else
    _____________________________
}

int get_max3(int x, int y, int z)
{
    _____________________________
    _____________________________
    _____________________________
}
```

(a) get_max2() 함수의 반환값을 이용하지 않고도, 세 수 중에 가장 큰 값을 찾아낼 수 있도록 get_max3() 함수를 작성하여 보라.

(b) if-else 제어 구조를 이용하고, 몸체 안에서 get_max2() 함수를 사용하는 꼴로 get_max3() 함수를 작성하여 보라.

(c) get_max2() 함수의 반환값을 이용하여서 get_max3() 함수를 작성하는 경우, 한 줄로도 작성이 가능한가?

실습 문제 8

➢ **함수 원형 (File: 8_8.c)**

```c
#include <stdio.h>

void f(int x, int y);

int main(void)
{
    int a, b;

    a = 10;
    b = 20;

    printf("f() 호출전a = %d, b = %d\n", a, b);
    f(a, b);
    printf("f() 호출후a = %d, b = %d\n", a, b);

    return 0;
}

void f(int x, int y)
{
    x = 30;
    y = 40;
    printf("f() x = %d, y = %d\n", x, y);
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 결과를 기록하라.

(b) 위의 프로그램에서 인수와 매개 변수를 지적하라. 인수와 매개 변수의 관계를 설명하라.

(c) main()의 변수 a와 b를 x와 y로 변경한 후에 컴파일하여 실행하여 보라. 결과가 달라지는가?

Chapter 9

실습 문제 1

➢ **함수의 범위 - 전역 변수와 지역 변수 (File: 9_1.c)**

```c
#include <stdio.h>
int a;      // 전역 변수

int main(void)
{
    int b;
            // ①
    {
        int c;
            // ②
    }
    {
        int d;
            // ③
    }
    return 0;
}

void f(void)
{
    int e;
            // ④
}
```

(a) 각각의 변수가 전역 변수인지 지역 변수인지를 말하라.

(b) 각각의 변수의 범위를 표시하라.

(c) ①, ②, ③, ④ 위치에서 사용이 가능한 변수들을 조사하여 보자. 구체적으로 ①, ②, ③, ④ 위치에서 변수들의 값을 출력하는 문장을 삽입하여 어떤 변수들이 접근이 가능한지를 확인하라.

(d) 전역 변수와 지역 변수의 초기값은 무엇인지 살펴보라.

(e) 변수 a, b, c, d, e를 각각 1, 2, 3, 4, 5로 초기화한 후에 모든 변수들의 이름을 x로 통일한다. ①, ②, ③, ④ 위치에서 x의 값을 출력하여 보라. 전역 변수와 지역 변수가 이름이 같은 경우, 어떤 변수가 사용되는가?

실습 문제 2

➢ **전역 변수의 사용 (File: 9_2.c)**

```c
#include <stdio.h>

int add(int x, int y);     // 함수 원형 정의
int main(void)
{
    int result;

    result = add(10, 20);
    printf("%d\n", result);

    return 0;
}

int add(int x, int y)
{
    // ①
    return x + y;
}
```

(a) 각각의 변수가 전역 변수인지 지역 변수인지를 말하라.

(b) 함수의 매개 변수에 다른 값을 대입할 수 있는가? ①번 위치에서 매개 변수 x와 y에 각각 30과 40을 대입하여 보라. 매개 변수는 지역 변수인가?

(c) 전역 변수 sum을 정의하고 이 전역 변수 sum을 이용하여 덧셈의 결과를 add()에서 main()으로 전달하여 보라. 전역 변수를 사용하는 방법의 장점과 단점은 무엇일까?

실습 문제 3

➢ **static 변수 사용 (File: 9_3.c)**

```c
#include <stdio.h>
void f(void);      // 함수 원형 정의

int main(void)
{
    f();
    f();
    f();
    return 0;
}

void f(void)
{
    static int x = 0;
    int y = 0;

    x++;
    y++;
    printf("x = %d, y = %d\n", x, y);
}
```

(a) 위의 프로그램을 실행하고 그 결과를 기록하라. 결과를 설명하여 보라. 지역 변수 앞에 static을 붙이면 어떻게 되는가?

(b) main()에서 for 반복 구조를 이용하여 f()를 20번 호출하도록 코드를 수정하라.

(c) static 지역 변수를 이용하여 어떤 함수가 몇 번이나 호출되었는지 쉽게 알 수 있다. f()가 10번 이상 호출되면 x와 y의 값을 출력하는 printf() 문이 더 이상 수행되지 않도록 위의 코드를 변경하라.

실습 문제 4

➢ **extern 키워드 사용 (File: 9_4.c)**

```c
// s1.c 파일
#include <stdio.h>
extern int x;      // ①

int main(void)
{
    extern int y; // ②
    x = 10;
    y = 20;

    return 0;
}
int y;
```

(a) 위의 프로그램에서는 extern 키워드가 2곳에서 사용되었다. 어떤 목적으로 사용되었는지를 설명하라.

(b) 위의 프로그램을 컴파일하여 보라. 어떤 오류가 발생하는가? 오류의 원인은 무엇인가?

(c) 또 하나의 소스 파일인 s2.c를 만들고 그 곳에 변수 x를 전역 변수로 선언한 후에 s1.c와 같이 컴파일, 링크하여 실행 파일을 생성하여 보라. 오류가 없어지는가? 주의할 점은 같은 프로젝트 안에 소스 파일 s2.c를 생성하여야 한다.

(d) s2.c 파일에서 변수 x를 정의할 때에 앞에 static을 붙이면 어떻게 되는가? s1.c의 extern int x와 연결되는가? 전역 변수 앞에 static을 붙이면 어떤 의미가 되는가?

실습 문제 5

➢ 재귀 호출 (File: 9_5.c)

```c
#include <stdio.h>
int recursive(int n);

int main(void)
{
    int n = 0, sum;

    printf("정수를 입력하시오:");
    scanf("%d", &n);

    sum = recursive(n);
    printf("%d \n", sum);

    return 0;
}
int recursive(int n)
{
    // ①
    if(n <= 1) return 1;   // ②
    else return n + recursive(n-1); // ③
}
```

(a) recursive() 함수가 호출되면, 'recursive' 라는 함수 이름과 매개 변수 n의 값을 출력하는 문장을 ①의 위치에 삽입하라. 그 결과를 설명하라.

(b) ② 문장을 삭제하고 실행하여 보라. 어떤 결과가 얻어지는가?

(c) ② 문장에서 recursive(n-1)을 recursive(n)으로 변경하여 실행하여 보라. 어떤 결과가 얻어지는가?

(d) 재귀 호출을 사용하지 않고 반복 구조를 사용하여 recursive()를 다시 작성하여 보라.

(e) $1^2 + 2^2 + \cdots + n^2$을 재귀적으로 계산하도록 recursive()를 변경하여 보라.

과제 제출

- **과제 제출 & 포맷**
  - E-mail: dslab_yeob@naver.com
  - 메일 제목: [프프]학번_이름_날짜
    ex) [프프]000000000_김재엽_20160311
  - 과제 파일을 메일 제목과 동일하게 압축하여 제출
    ex) [프프]000000000_김재엽_20160311.zip
  - **+ 압축파일과 별도로 본인 사진 (ex) 이름.jpg) 첨부하여 제출!**

- **과제 제출 파일 List** (마지막 항목까지 수행한 파일 제출)
  - (8_1, 8_2, 8_3, 8_4, 8_5, 8_6, 8_7, 8_8).c
  - (9_1, 9_2, 9_3, 9_4, 9_5).c