

Java를 알고 C배우기

컴퓨터프로그래밍3

week 3-3 함수 기초

2022.1학기
충남대 조은선

추상화 (Abstraction)

▶ Process Abstraction

- ▶ 어떤 목적을 수행하는 일련의 문장들을 subprogram으로 작성함
- ▶ Subprogram의 종류
 - ▶ “Procedure” vs. “Function (함수)”
- ▶ Readability, writability 향상
- ▶ 프로그래밍 초기부터 강조되어 왔음

▶ Data Abstraction

- ▶ 프로그래머가 새로운 data type을 정의하는 것을 허용함
- ▶ 1980년대부터 강조됨
- ▶ 객체지향언어에서 다루게 됨

Subprogram의 필요성

- ▶ 아래 프로그램을 고려

```
(A) distx = x1 - x2;  
    if (distx < 0) distx = -distx;
```

```
(B) disty = y1 - y2;  
    if (disty < 0) disty = -disty;
```

→ 유사함 : 변수 이름을 제외하면 동일

- ▶ 유사한 부분을 모아 한번만 코드를 작성

```
(H) disty = s - t;  
    if (dist < 0) dist = -dist;
```

(A의 x1)과 (B의 y1)은 (H의 s)에 대응
(A의 x2)와 (B의 y2)는 (H의 t)에 대응

- ▶ H부분을 subprogram으로 작성하여 A와 B 부분에서 사용하는 방법이 편리
 - ▶ 대부분의 언어에서 지원: C언어 도!

C 언어의 subprogram 구현

(A) `big = x1;`
`if (big < x2) big = x2;`

...

(B) `big = y1;`
`if (big < y2) big = y2;`

...

(H) `big = s;`
`if (big < t) big = t;`

...

(A) `big = foo(y1,y2);`

...

(B) `big = foo(y1,y2);`

...

(A의 x1)과 (B의 y1)은 (C의 s)에 대응

(A의 x2)와 (B의 y2)는 (C의 t)에 대응

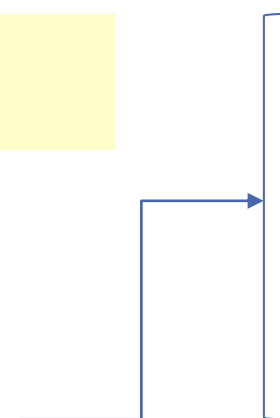
```
int foo(int s, int t)
{ int big;
```

(H) `big = s;`
`if (big < t) big = t;`

```
return big;
```

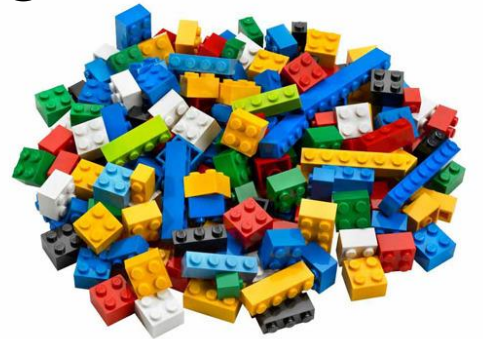
```
}
```

함수!



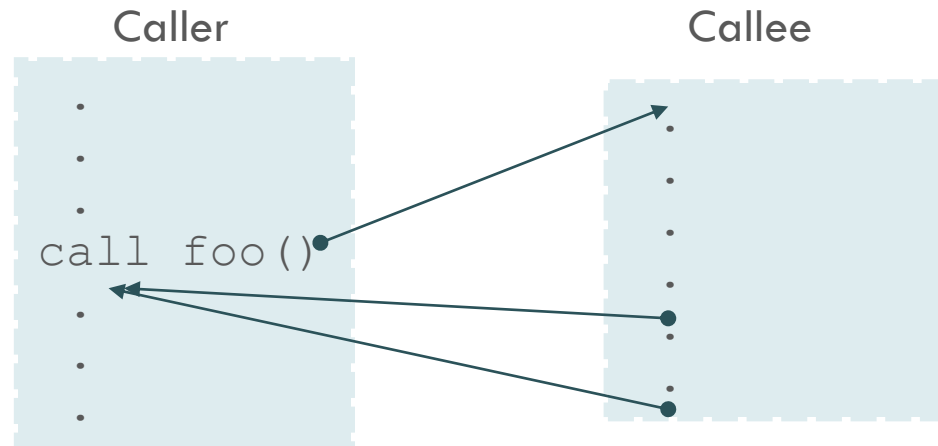
Subprogram 사용 이유

- ▶ 반복되는 코드를 한번만 작성함으로써 **코드 작성 용이 (효율성)**
- ▶ 프로그램 **module화**가 용이
 - ▶ 큰 문제는 작은 문제들로 나누어 해결하는 것이 일반적인 문제 해결 접근방법임
 - ▶ 작은 문제 각각은 subprogram으로 구현하고 한 프로그램을 작은 subprogram들의 집합으로 정의
 - ▶ 각 subprogram은 독립적으로 작성, 수정, 테스트됨
- ▶ **개별 컴파일** (separate compilation) 가능
 - ▶ subprogram 들은 대개 서로 독립적이어서, 각 subprogram은 독립적으로 컴파일 가능
 - ▶ 컴파일된 subprogram들은 linking을 통해서 실행가능한 큰 프로그램이 됨
 - ▶ 한 subprogram이 수정되면, 해당 subprogram만 다시 컴파일하면 됨
 - ▶ 한 subprogram의 수정으로 프로그램 전체를 다시 compile 할 필요가 없고
 - ▶ 규모가 큰 프로그램의 경우 컴파일 시간을 상당히 절약 가능



Subprogram의 특성

- ▶ 한 subprogram의 실행 시작 지점은 하나임 (one-entry)
- ▶ 한 subprogram의 실행이 종료되는 지점은 하나 이상일 수 있음
- ▶ Subprogram의 caller와 callee의 동작 과정
 - ▶ Caller (subprogram을 호출한 모듈)는 Callee(호출된 subprogram)의 실행이 완료될 때 까지 실행이 유보됨
 - ▶ Callee의 실행이 완료되면 프로그램의 실행 제어는 Caller에게로 넘어감
 - ▶ Caller는 subprogram을 호출한 지점의 다음부터 실행을 재개



Subprogram은 하나의 기능만 수행하도록 하자

용어, 개념 정의

Subprogram의 종류

- ▶ 함수(function), procedure

이름

- ▶ 다른 subprogram과 구별

형식인자 (formal parameter)

- ▶ header에 나열된 임시 변수 (dummy), Subprogram 내에서만 사용됨

리턴 타입

- ▶ Subprogram이 함수(function)인 경우에만 사용됨
- ▶ Function을 실행한 결과로 리턴하는 값의 타입

프로토타입 (prototype)

- ▶ Signature 라고도하며, 형식 인자의 개수, 순서, 타입과 리턴 타입

정의(Definition)

- ▶ header와 body를 표현

선언(Declaration)

- ▶ 프로토타입

호출(call)

- ▶ subprogram을 실행하도록 하는 명시적 표현
- ▶ {이름, 실 인자}

실 인자(actual parameter)

- ▶ 호출할 때 사용되는 값이나 주소
- ▶ 형식인자에 대응됨

Header

- ▶ Subprogram의 첫부분으로 사용법(interface)을 표현
- ▶ 즉, 이름, 종류, parameter profiler, 리턴 타입 등

Body

- ▶ Subprogram의 실행과정을 표현

```
int main()
{
    int foo(int, int);

    ...
    bigger = foo(a, b);
    ...
}

int foo(int s, int t)
{
    int big;

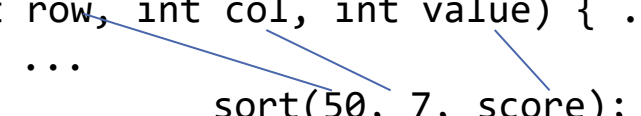
    big = s;
    if (big < t)
        big = t;
    return big;
}
```

실인자와 형식인자의 대응

위치 인자

- ▶ 인자 순서에 따라 대응
- ▶ 대부분의 언어에서 지원

```
sort(int row, int col, int value) { ... }  
...  
sort(50, 7, score); ...
```



가변 인자

- ▶ 인자의 개수가 변함 - printf()

```
void printNumbers(int args, ...) {  
    printf("%d ", args);  
}  
...  
printNumbers(1, 10);  
printNumbers(2, 10, 20);  
printNumbers(3, 10, 20, 30);  
printNumbers(4, 10, 20, 30, 40); // 출력 1 2 3 4
```

고정이어야함

```
#include <stdio.h>  
#include <stdarg.h>
```

```
int sum(int count, ...) {  
    int res = 0;  
    va_list ap;  
    int i;  
  
    va_start(ap, count);  
    for(i=0; i<count; i++)  
        res += va_arg(ap, int);  
    va_end(ap);  
    return res;  
}  
  
int main() {  
    printf("%d\n",  
        sum(10, 1,2,3,4,5  
            6,7,8,9,10));  
    return 0;  
}
```

출력 결과 >> 55

Prototype

- ▶ C는 One-Pass Compiler
 - ▶ 소스코드를 한 번만 보고 컴파일을 끝냄
 - ▶ A가 B보다 먼저 나와야 B가 A를 호출할 수 있음
 - ▶ 함수 Prototype 을 먼저 선언하면 순서가 문제되지 않음

```
#include <stdio.h>
int square(int m);
int square_add(int, int);
```

- ▶ 함수의 선언, 정의, 호출은 다른 의미!

```
int square(int m);      ① 선언
int square(int m){      ② 정의
    return m*m;
}
...
square(num);            ③ 호출
```

여러가지 C 함수의 특성들

- ▶ 리턴값이 없는 함수
 - ▶ 리턴타입을 `void` 라고 적어주고, body에서 `return;` 으로 탈출
 - ▶ 엄밀히 말해서 함수가 아니라 procedure
- ▶ C 표준 라이브러리
 - ▶ `stdio.h`, `stdlib.h`, `ctype.h`, `string.h`, `math.h`, `time.h`, ...
 - ▶ 대략 145개 정도의 함수를 제공

Quiz

▶ Subprogram이나 함수를 사용하는 이유로 적절하지 않은 것은?

- (a) 반복되는 코드를 단지 한번만 작성할 수 있기 때문에
- (b) 프로그램 실행 시간을 줄일 수 있기 때문에
- (c) 큰 문제는 작은 문제들로 나누어 해결하기가 용이하므로
- (d) 프로그램 개발 시 컴파일 시간을 줄일 수 있기 때문에