9장 부프로그램 (1)

부 프로그램 사용 이점

- 작성력 향상
- 프로그램 모듈화 용이
- 개별 컴파일 가능

부프로그램 특성

- 각 부프로그램은 단일의 진입점을 갖는다.
- 호출 프로그램 단위는 피호출 부프로그램의 실행 중에 중단된다.
- 부프로그램의 실행이 끝났을 때 제어는 항상 호출 프로그램에게 돌아간다.

부프로그램의 기본 정의

- 부프로그램 정의
  - 부프로그램의 인터페이스와 동작을 서술
- 부프로그램 호출
  - 부프로그램이 수행에 대한 명시적 요청
- 부프로그램 머리부
  - 부프로그램의 첫 줄에 나타나며, 부프로그램의 인터페이스를 명세한다.
- 매개변수 프로파일(프로필)
  - 형식 매개변수의 개수, 순서, 타입이다.
  - 서명(Signature)이라 불리기도 한다.
- 프로토콜
  - 매개변수 프로파일과 반환값 타입(함수인 경우)이며,부프로그램에 대한 인터페이스를 명세
- 형식 매개변수
  - 부프로그램 머리 부상에에 나열한 매개변수
- 실 매개변수
  - 부프로그램 호출문에 사용되는 매개변수
- 부프로그램 선언
  - 부프로그램의 프로토콜만 제공하며, 몸체는 포함하지 않음

매개 변수

- 부프로그램이 처리하는 데이터
  - 매개 변수
  - 지역 변수
  - 비 지역 변수 -> 부작용 초래, 신뢰성 감소

매개변수 대응 2가지 방법

- 위치 기반 매개 변수
  - 단순한 위치에 기반
- 키워드 기반 매개 변수
  - 실 매개변수에 바인딩되는 형식 매개변수의 이름이 실 매개변수의 이름이 실 매개변수와 함께 지정

디폴트 매개변수

- 생략된 매개변수 다음에는 키워드 매개변수가 온다.

부프로그램 유형 * Procedure * 매개 변수화된 계산을 정의하는 문장들의 집합을 추상화 * 사용자 정의 문장 제공 * 반환 값이 없다. * Function * 수학적 함수를 모델리 * 사용자 정의 연산자제공 * 반환 값이 존재한다.

9장 부프로그램 (2) <- 여기 중요

매개변수 전달의 의미적 모델

- 입력 모델
  - 형식 매개변수는 실 매개변수로부터 데이터 전달 받음
- 출력 모드
  - 형식 매개변수는 실 매개변수에 데이터를 전달 함
- 입출력 모드
  - 형식 매개변수는 실 매개변수로부터 데이터를 전달받고, 데이터를 실 매개변수에 전달함

매개변수 전달의 데이터 이동 방법

- Call by Value
- Call by Reference

매개변수 전달의 구현 모델

- 값 전달
- 결과 전달
- 값-결과 전달
- 참조 전달
- 이름-전달

값 전달

값 전달 (Pass by value) 은 매개변수가 값으로 전달되는 입력 모드의 구현

- 형식 매개변수 (Callee) 는 대응 실 매개변수 (Caller)의 값으로 초기화
- 형식 매개변수는 지역변수로 사용
- 데이터의 이동 방식
    - 물리적 값 이동 (값 복사) -> 일반적
    - 접근 패스 전달
    - 쓰기-보호 셀(Read Only) 필요
- 장점 :
    - 스칼라 변수의 경우 연결 비용과 접근 시간이 빠르다
    - 부프로그램의 외부 데이터 접근 제한
- 단점 :
    - 값 복사가 사용될 경우:
    - 형식 매개변수에 대한 기억공간 할당, 값복사에 따른 비용 부담
    - 접근 패스 전달의 경우:
    - 형식 매개변수에 대한 쓰기-보호 요구되고, 간접 주소 지정에 따른 접근 비용 부담

결과 전달

결과 전달 (Pass by result)은 출력 모드의 구현 * 실 매개변수는 형식 매개변수에 값을 전달하지 않는다. * 형식 매개변수는 지역 변수로 사용 * 피호출자 종료 전에 형식 매개변수의 값을 대응 실 매개변수로 전달

값-결과 전달

값-결과 전달 (Pass by value result)은 값이 복사되는 입출력 모드의 구현 * 값 전달과 결과 전달의 혼합 * 복사 전달 (Pass by copy) 라고 명하기도 함 * 처음 : 형식 매개변수를 대응 실 매개변수의 값으로 초기화 * 피호출자 종료 전 : 형식 매개변수의 값을 대응 실 매개변수로 전달

참조 전달

참조 전달 (Pass by reference) 은 입출력 모드의 구현 * 접근 패스 (주로 주소) 를 전달 * 공유 전달 (Pass by sharing) 이라고도 명칭함 * 장점: * 전달 과정 자체가 시간, 기억장소 관점에서 효율적 * 단점: * 별칭 생성 * 신뢰성 저하 우려 존재 * 실 매개변수간의 충돌 * 배열 원소간의 충돌 * 형식 매개변수와 가시적 비 지역변수 간의 충돌

9장 부프로그램 (3)

중복 부프로그램 (Overloaded)

중복 부프로그램은 같은 참조 환경에서 다른 부프로그램과 이름이 같은 부프로그램

- 중복 부프로그램은 **고유한 프로토콜**을 갖는다.
- 그 호출 의미는 **실매개변수 리스트**와 **반환값의 타입**에 의해서 결정
- 중복 부프로그램의 호출은 정적으로 결정
- 지원 언어:
    - C++ : 강제 변환 모두 허용
    - Java : 확장 변환의 경우에만 강제 변환 허용
    - Ada : 타입 강제 변환을 허용하지 않음
    - C#

다형 부프로그램

다형 부프로그램 (Polymorphic Subprogram)은 다른 활성화시에 다른 타입의 매개변수를 취하는 부 프로그램

- 다형 부프로그램 유형
    - 특이 다형성 (Ad hoc Polymorphism) -> 중복 부프로그램
    - 매개변수 다형성 (Parametric Polymorphism) -> OOP
    - 부타임 다형성 (Subtype Polymorphism) -> Parameter 로 함수 생성

특이 다형성 (Ad hoc Polymorphism)

- 특이 (Ad hoc) 은 '특정 목적으로수행되는 무엇' 을 의미하는 그리스어의 어원
- 중복 부프로그램은 다형 부프로그램의 한 유형으로 특이 다형성을 제공
    - 같은 이름을 갖은 여러 개의 부프로그램이 중복
    - 중복된 각 부프로그램은 특정 목적을 가지며, 유사하게 동작할 필요 없음

매개변수 다형성 (Parametric Polymorphism)

- 매개변수 다형성은 부프로그램의 형식 매개변수란에 포괄형 매개변수를 갖는 부프로그램으로 제공

- 포괄형 매개변수는 임의의 타입으로 대체 가능한 타입 매개변수이다.
- 매개변수 다형적 부프로그램을 ** 포괄형 부프로그램 ** (Generic Subprogram) 이라 부른다.
- 매개변수 다형적 부프로그램은 모두 동일하게 행동

부타입 다형성 (Subtype Polymorphism)

- 부타입 다형성은 타입 T의 변수가 T의 객체나 T로부터 파생된 임의 타입의 객체를 접근할 수 있음을 의미
- 객체 지향 프로그래밍 (OOP) 언어에서 지원

포괄형 부프로그램 (Generic)

- 포괄형 함수
    - C++ : 템플리트 함수 (Template Function) 으로 표현
    - Java : 포괄형 타입 이용
    - 포괄형 매개변수는 클래스에 제한
        - 기본 타입은 허용 X
        - 클래스 범위 제한