Chapter 05.

프로그래밍 언어

❑ 학습목표

- 프로그램 작성 과정에 필요한 구체적인 작업내용과 작업도구들을 학습한다
- 프로그래밍 언어의 세대별 구분을 통해 프로그래밍 언어의 발달과 변천 과정을 학습한다
- 프로그래밍 언어들 중에서 가장 영향력이 크고 널리 쓰여진 언어를 살펴본다
- 프로그래밍 언어의 개발에 많이 사용되는 구현 기법을 학습한다
- 객체지향 프로그래밍 기법을 학습한다

- 프로그래밍
  - 문제를 해결하는 방법
- 컴퓨터 프로그램
  - 컴퓨터 하드웨어에게 일을 시키기 위한 일련의 명령문 집합
- 소프트웨어
  - 소프트웨어 패키지(packaged software)
    - 상품화된 소프트웨어
    - 쉐어웨어(shareware), 프리웨어(freeware)
  - 주문 제작된 소프트웨어(customized software)
    - 사용자를 위해 특별히 주문 제작된 응용 프로그램
- 프로그램 작성 과정

- 프로그램을 작성하여 문제를 해결하는 경우
  - 프로그램의 사용자가 누구인지를 명확히 해야 함
- 출력 정보
  - 문제 해결의 결과
  - 어떠한 내용을 포함해야 하며, 어떤 매체로 출력해야 하고, 어떠한 양식으로 출력(output)되어야 바람직한 것인지 설계
- 입력 데이터 선정 또는 수집
  - 사용자와의 대면 방식은 어떤 스타일로 할 것인가 등에 대한 세부 사항을 명시
- 프로그램으로 구현하기 위한 비용이나 시간 등 실용성 조사를 실시
- 세부적인 사항의 문서화

❍ 알고리즘 설계에서 필요한 기본 개념

❑ 프로그램 논리

- 계층 차트(hierarchy chart)를 이용한 하향식 접근 방식(top-down approach)과 모듈화(modulization)를 통해 이루어짐

❑ 프로그램의 세부적인 형식

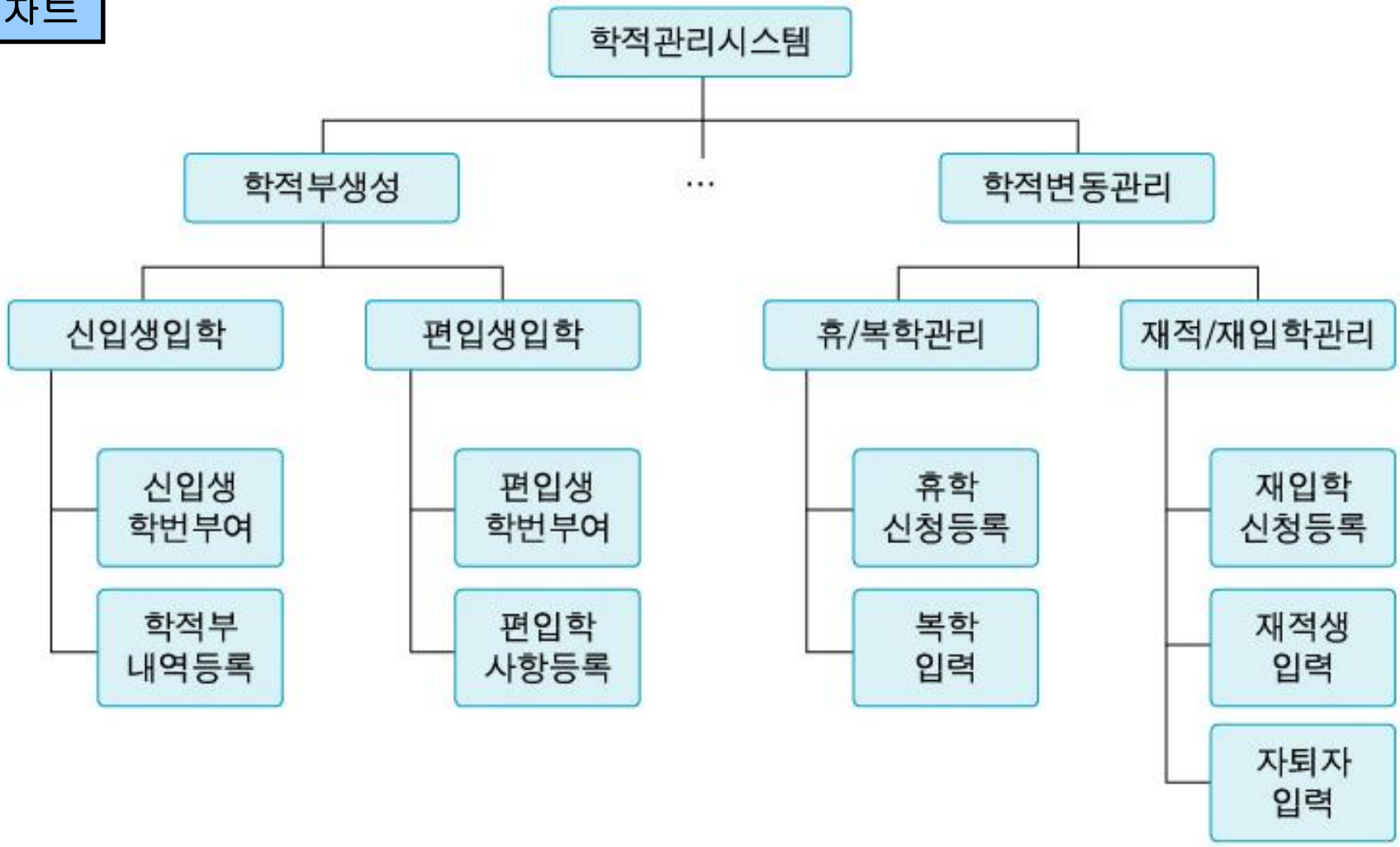
- 유사 코드(pseudo code)를 사용하여 문장으로 서술
- 순서도(flowcharts)를 써서 그래픽으로 나타낼 수도 있음

❑ 구조적 프로그래밍 기법으로 프로그램 작성

- 구조적 프로그래밍 기법(structured programming)
  - 프로그램을 구성 단위인 모듈(module)로 나누는 모듈화와 하향식 접근 방법을 취하고 표준화된 세 가지 기본적인 제어 구조(control structure)만을 사용하여 작성하는 방식
  - 세 가지 기본적인 제어 구조
    - » 서열 구조, 선택 구조, 반복 구조

- 하향식 접근 방법(top-down approach)
  - 크고 복잡한 하나의 문제를 작고 단순한 여러 개의 구성 요소로 나누어 하나씩 처리함으로써 전부를 해결하고자 하는 것
  - 모듈화 개념의 중요성
    - 전체 프로그램이 분할되어 개발되기 때문에 프로그램 작성이나 테스트를 세부적으로 또는 별도로 시행할 수 있음
  - 하향식(top-down) 프로그램 설계
    - 계층 차트 이용
    - 계층 차트
      - 구조 차트(structure chart)
      - 각 모듈의 목적과 그들 사이의 연계성을 나타냄으로써 프로그램의 전체적인 목적을 설명하기 위해 쓰여지는 것

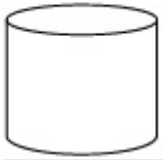
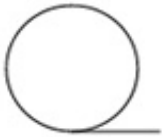
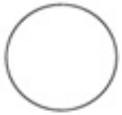
계층 차트


학적관리시스템
학적부생성
…
학적변동관리
신입생입학
편입생입학
휴/복학관리
재적/재입학관리
신입생
학번부여
학적부
내역등록
편입생
학번부여
편입학
사항등록
휴학
신청등록
복학
입력
재입학
신청등록
재적생
입력
자퇴자
입력

❍ 세부 설계

- ❑ 유사 코드(pseudo code)를 사용해서 서술 형식 표현
  - 일상 언어로 프로그램 논리와 처리 순서를 서술하는 방법
- ❑ 순서도(flowchart)를 이용해서 그래픽으로 나타냄
  - 프로그램 순서도
    - 문제 해결을 위한 프로그램을 작성하는 데 필요한 알고리즘 (algorithm) 또는 논리적인 일의 흐름을 그래픽으로 나타내는 것

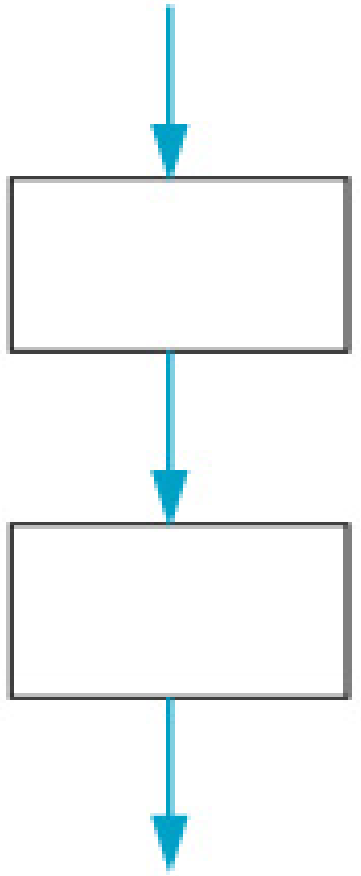
순서도 기호

기호	의미	기호	의미	기호	의미
	시작/끝		작업준비		데이터
	처리		비교/판단		흐름선
	온라인 수동입력		문서(출력)		디스플레이
	자기디스크		자기테이프 (순차파일)		온라인 저장매체
	오프라인 저장매체		연결자		페이지 연결자
	컴뮤니케이션 링크		Manual Operation		Keying peration

- 제어 구조(control structure)
  - 일종의 논리 구조(logic structure)
  - 컴퓨터 프로그램을 구성하는 명령문의 논리적 실행 순서를 제어하는 구조
  - 세 가지 기본적인 제어 구조
    - 순차 구조(sequence structure)
    - 반복 구조(iteration structure)
    - 선택 구조(selection structure)

❑ 순차 구조

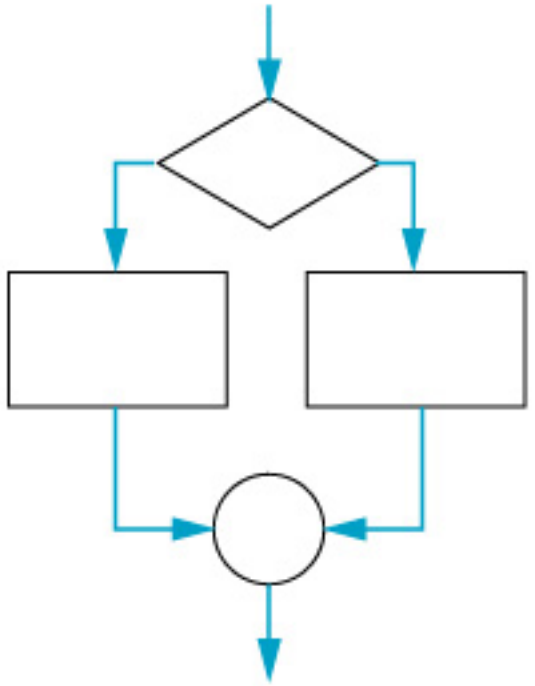
- 프로그램을 구성하는 명령문 (statements)들이 프로그램이 서술된 순서에 따라 차례차례 실행되는 구조



❑ 선택 구조

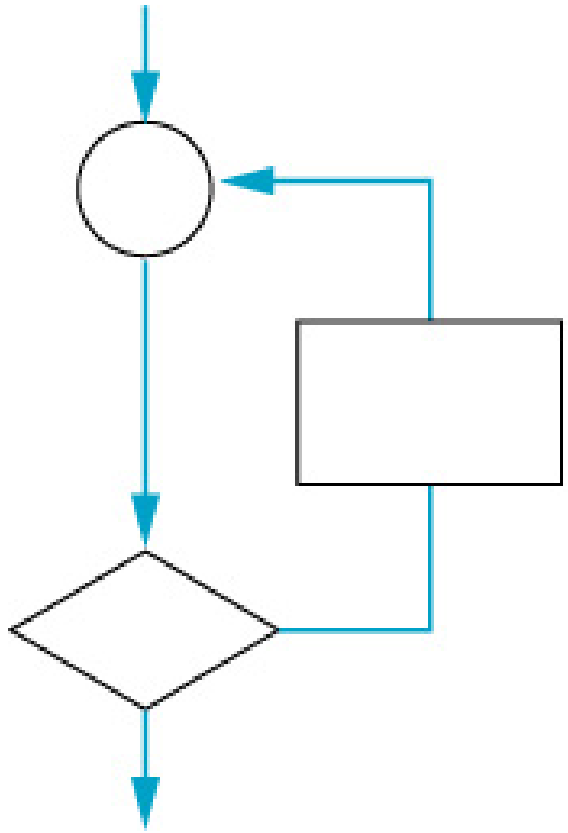
- IF-THEN-ELSE 구조
- IF문에서 주어진 조건이 참이냐 거짓이냐에 따라 실행되는 명령문이 달라지는 경우
- 변형된 선택 구조
  - case나 switch문과 같은 다중 선택 구조

```
switch(expression){
    case constant_expr_1 : statement_1 ;
    ......
    case constant_expr_n : statement_n ;
    [default : statement_n + 1]
}
```



❑ 반복 구조

- 루프(loop)
- 주어진 조건이 맞으면 일련의 명령문을 반복해서 실행하는 구조
- C 언어에서 for문의 일반적인 형식
  - for (expression_1 ; expression_2 ; expression_3) 반복 내용(loop body)
  - 첫 번째 수식(expression_1)은 초기화를 위한 것
  - 두 번째 수식(expression_2)은 반복 제어를 위한 값
  - 세 번째 수식(expression_3)의 값도 반복될 때마다 계산되며 반복 횟수를 증가(또는 감소)시키는 데 사용



- 파스칼, C, C++, 자바에서는 while문
  - while (expression) 반복 내용(loop body)
  - 반복 내용을 실행하기 전에 수식의 값을 계산하여 true(0이 아닌 경우)일 때 실행
- do…while 형식
  - do

    반복 내용(loop body)

    while (expression)
  - 반복 내용이 적어도 1회 이상 실행

- 코딩(coding)
  - 프로그래밍 언어로 프로그램을 작성하는 일
- 프로그램 코딩 단계에서 중요한 일
  - 적당한 프로그래밍 언어를 선정하는 일
    - (ex) 포트란이나 파스칼
      - 수학적 문제 해결이나 통계 처리에 유용한 언어
    - (ex) 코볼
      - 데이터 처리에 강력한 언어
  - 선정된 프로그래밍 언어가 갖춘 구비한 구문(syntax)을 철저하게 지키면서 코딩하는 일

❍ 기본적인 테스트 단계

- ❑ 데스크 체킹(desk-checking)
  - 오류가 없는지 프로그램의 논리 구조가 정확한지 프로그램을 컴퓨터에 실행시키기 전에 점검하는 과정
- ❑ 프로그램 디버깅(program debugging)
  - 프로그램이 포함하고 있을지도 모르는 모든 오류(구문 오류나 논리 오류)를 찾아내어 제거하는 일
- ❑ 실질 데이터에 의한 테스트
  - 실질적인 사용자가 의도적으로라도 결함이 있는 데이터까지 동원해서 다단계 시험 시행을 실시해보는 것

- 프로그램의 문서화(program documentation)
  - 프로그램이 무슨 일을 하며 어디에 어떻게 사용하는지를 문서로 작성하는 일
  - 기록된 자료
    - 추후에 프로그램을 사용하고 유지 보수할 사람을 위한 것
- 유지 보수(maintenance)
  - 프로그램을 항상 사용하기 좋은 조건, 오류가 없는 상태, 최신의(up-to-date) 상태로 유지하기 위한 활동

❑ 프로그램 작성 과정 5단계

1단계 문제 정의와 분석	프로그램 목적, 사용자, 입출력, 데이터 처리에 관한 요구사항 구현상의 문제 분석, 타당성 조사 등 전과정에 대한 문서화
2단계 알고리즘 설계	계층 차트, 순서도 또는 유사코드를 사용해서 프로그램의 논리적인 실행 순서와 내용을 구체화
3단계 프로그램 코딩	적절한 고급 언어를 선택해서 구문에 맞는 프로그램 작성
4단계 프로그램 테스팅	프로그램 디버깅(알파테스팅)과 실질적인 데이터와 실질적인 사용자에 의한 시험 실행(베타테스팅)
5단계 프로그램 문서화/유지보수	하드웨어와 소프트웨어 요구사항, 입출력과 프로그램 파일 관리 등 전체적인 사용 설명서 작성, 프로그램의 효과적인 사용을 위한 유지 및 보수

- 프로그래밍 언어
  - 컴퓨터에서 일을 수행하기 위한 명령문 집합인 프로그램을 작성하기 위해 인위적으로 만들어진 언어
- 프란칼큘(Plankalkul: program calculus)
  - 1936년~1945년 독일 콘라드 주세(Konrad Zuse) 개발
  - 세계 최초의 언어
  - 전쟁으로 소실되었다가 1972년에 발표됨
- 프로그래밍 언어가 이진수로 된 언어에 가까울 때
  - 저급 언어로 분류
- 인간이 사용하는 자연 언어에 가까울 때
  - 고급 언어로 분류

❑ 프로그래밍 언어의 세대별 구분

세대별	출현시기	언어
1세대	1945년	기계어
2세대	1950년대 중반	어셈블리 언어
3세대	1960년대 초	고급 언어(포트란, 코볼, 베이직, C, 에이다)
4세대	1970년대 초	초고급 언어(SQL, Intellect, NOMAD, FOCUS)
5세대	1980년대 초	자연 언어

❑ 1세대 언어: 기계어

- 기계어(Machine language)
  - 프로그래밍 언어 중에 가장 저급
  - 0과 1로 표현되는 이진수로 이루어진 언어
  - 컴퓨터 하드웨어에 의존적
  - 프로그램을 작성하는 시간 많이 걸림
  - 오류 자주 발생
- 어셈블리 언어가 탄생
  - 기계어로 인한 어려움을 해결하려는 노력의 결과

- 어셈블리 언어(assembly language)
  - 명령어를 숫자가 아닌 암기하기 쉬운 문자로 작성할 수 있도록 심볼화
    - ADD Register → AR
    - STORE → ST
  - 언어 번역기(language translator)라는 개념이 탄생
- 언어 번역기
  - 시스템 소프트웨어의 일종
  - 어셈블러(assembler)
    - 어셈블리 언어로 작성된 프로그램을 기계어로 번역하는 프로그램
  - 컴파일러(compiler), 인터프리터(interpreter)

- 고급 언어(High-Level language)
  - 프로그램을 좀더 쉽게 작성하기 위해 개발
  - 자연 언어에 가까운 언어
  - 기계에 비의존적
    - 서로 다른 컴퓨터 하드웨어 사이에 호환 가능
- 컴파일러(compiler) / 인터프리터(interpreter)
  - 고급 언어로 작성된 프로그램을 기계어 코드로 번역시키기 위해서 필요

- 컴파일러(compiler)
  - 고급 언어로 작성된 프로그램을 실행시키기 전에 기계어로 된 코드로 번역
  - 목적 코드(object code) 생성

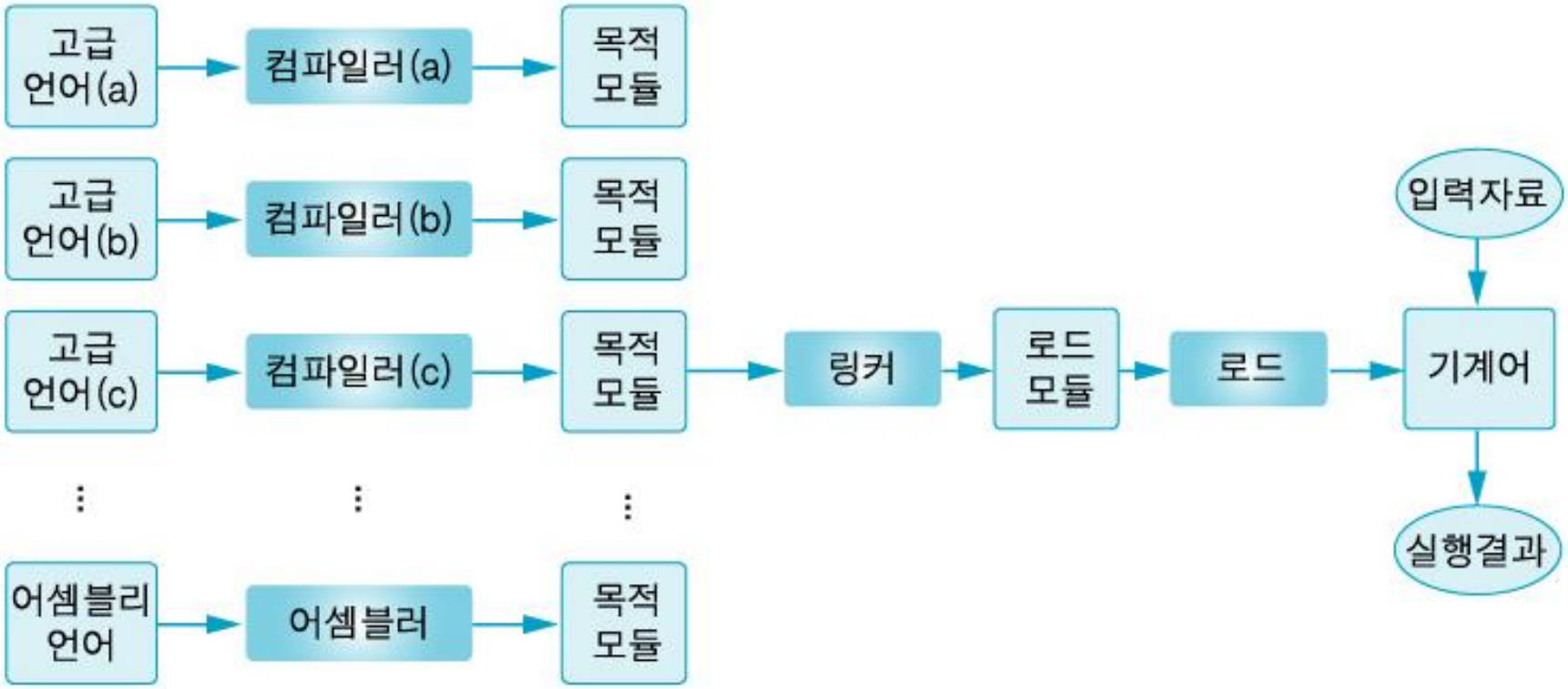
- 인터프리터(interpreter)
  - 고급 언어로 작성된 프로그램을 구성하는 명령문 하나하나를 번역하는 즉시 실행시킴
  - 목적 코드를 생성하지 않음
- 3세대 언어로 알려진 고급 언어
  - 대부분 절차 위주의 언어들(procedural languages)임

❍ 객체지향 언어

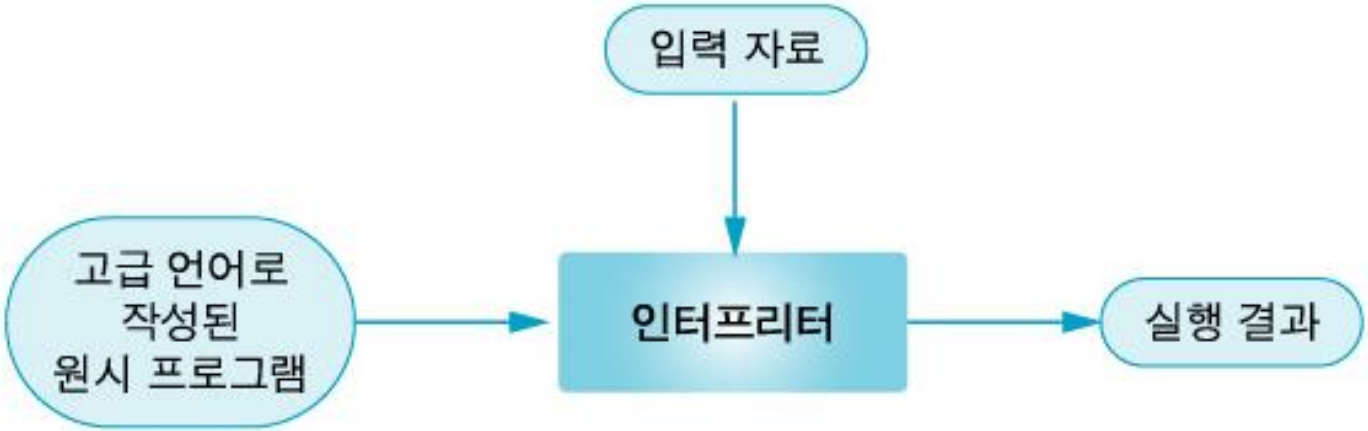
- ❑ 객체 지향 프로그래밍(OOP, Object-Oriented Programming)
- ❑ 캡슐화(encapsulation)
  - 데이터와 이를 처리할 명령들을 객체(object)라는 단위로 묶는 것
- ❑ 상속(inheritance)
  - 객체가 만들어지면 프로그램 내에서 상속을 통하여 재사용 가능
  - 새로운 객체가 기존 객체의 서브클래스(subclass)로 정의되면 기존 객체의 데이터와 그에 행하여지는 연산 기능(method)이 상속되는 것
- ❑ 다형성(polymorphism)
  - 객체가 필요로 하는 데이터 타입이 무엇인지 미리 정하지 않아도 되는, 프로그래밍 기법에서는 진일보한 개념

- 번역 기법(translation)
  - 주어진 고급 프로그래밍 언어로 작성된 프로그램을 실제 주어진 컴퓨터의 기계어로 번역하여 동등한 의미의 기계어 프로그램을 만들어 실행시키는 방법
- 번역기의 종류
  - 컴파일러(compiler)
    - 원시 언어가 고급 언어고, 목적 언어가 저급 언어인 번역기
  - 어셈블러(assembler)
    - 원시 언어가 어셈블리 언어인 번역기
  - 링커(linker, 링키지 에디터)
    - 재배치 형태의 기계어로 된 여러 개의 프로그램을 묶어서 로드 모듈이라는 실행 가능한 하나의 기계어로 번역해 주는 번역기

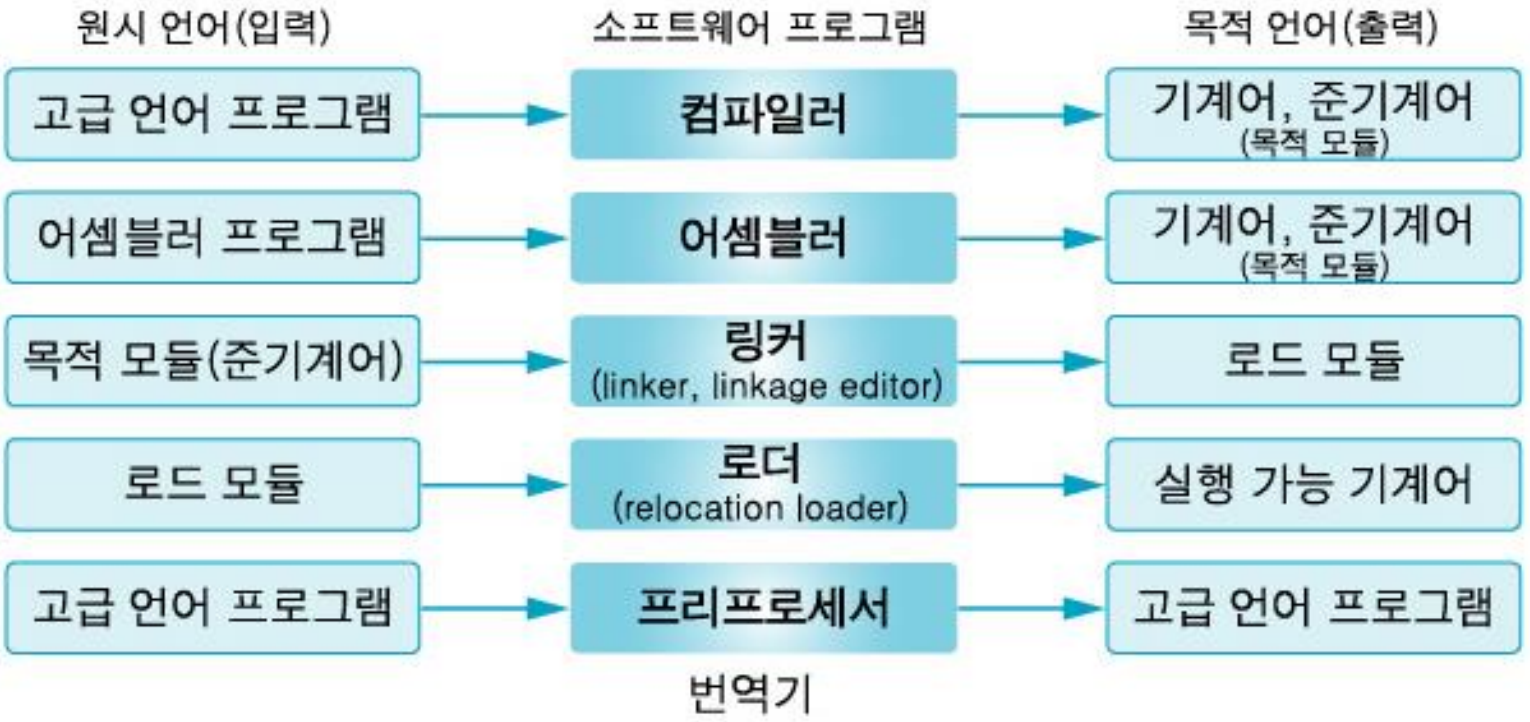
- 로더(loader)
  - 로드 모듈로 된 기계어 프로그램을 실제 실행 가능한 기계어로 번역해서 주기억장치에 적재
- 프리프로세서(preprocessor)
  - 원시 언어와 목적 언어가 모두 고급 언어인 번역기



- 인터프리터 기법(interpretation)
  - 소프트웨어 시뮬레이션 기법
  - 고급 언어로 된 프로그램을 자료로 읽어 들여서 기계어 수행과 동일한 알고리즘으로 프로그램의 각 문장을 디코딩하고 실행시킴으로써 이 고급 언어를 시뮬레이션하는 것
- 인터프리터 실행



❍번역기 종류와 인터프리터



Thank you