1. 프로그래밍 언어 개요

1) 프로그래밍이란

- 주어진 어떤 문제를 해결하기 위해 인간과 컴퓨터 사이에서 의사소통을 가능하게 하는 인공적인 언어를 말함

▸ 프로그래밍 언어를 공부해야 하는 이유

- 효율적인 알고리즘을 개발할 수 있는 능력의 향상시킴
- 현재 사용하는 프로그래밍 언어의 능력을 향상시킴
- 주어진 과제를 해결하는 최적의 언어를 선택할 수 있음
- 새로운 언어를 쉽게 배울 수 있음

2) 프로그래밍 언어의 특성

- 간결성(simplicity): 사람이 프로그램을 쉽게 이해하고, 읽을 수 있도록 간결하게 표현할 수 있는 특성
- 직교성(orthogonality): 언어의 각 구성 요소가 상호 독립적이고 어떤 환경에서도 그 구성 요소가 같은 의미로 사용
- 가독성(readability): 누구나 쉽게 프로그램을 읽을 수 있는 특성
- 정확성(preciseness): 엄밀하게 정의된 문법에 따라 작성된 프로그램은 정확성을 보장하며 예측 가능한 번역을 보장
- 기계 독립성(machine independence): 서로 다른 컴퓨터 상에서도 운영이 가능해야 하고, 똑같은 결과를 도출해야 함

3) 저급 언어와 고급 언어

(1) 저급 언어

- 기계어와 어셈블리 언어를 의미함
- 하드웨어에 관련된 직접제어가 가능함

- 프로그램 작성시 상당한 지식과 노력이 필요함

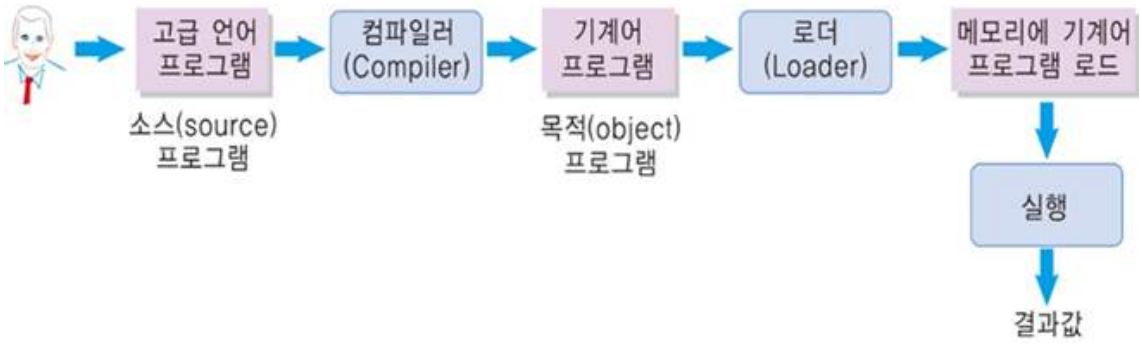
(2) 고급 언어
- 하드웨어에 관련된 지식 없이도 프로그램 작성 가능함
- 일상 적인 언어, 기호 등을 그대로 이용함
- 기억장소를 임의의 기호(symbol)에 저장하여 사용함
- 하나의 명령으로 다수의 동작을 지시할 수 있음(예: A = B + C * D)

프로그래밍 언어	연도	창시자	큰 영향을 준 선행 언어	주요 개발 목적
FORTRAN	1957	J. Backus(IBM)	-	수치 계산
COBOL	1960	위원회	FORTRAN	상업 자료 처리
Lisp	1962	J. McCarthy (MIT)	-	기호연산 리스트 처리
BASIC	1965	Kemeny (Dartmouth)	-	교육용 프로그래밍
SNOBOL	1966	R. Griswold	-	문자 처리
Pascal	1971	N. Wirth (ETH Zurich)	ALGOL68 BCPL	범용 및 교육용, 구조적 프로그래밍
C	1974	D. Ritchie (Bell Labs)	ALGOL68 BCPL	시스템 프로그래밍
Ada	1979	J. Ichbiah 등 (CII Honeywell Bull)	Pascal Simula 67 Modula	범용 및 응용 실시간 프로그래밍
C++	1983	B. Stroustrup	Simula 67 ALGOL 68 C	범용 및 시스템 프로그래밍
Java	1994	James Gosling	C	범용 및 인터넷 프로그래밍
C#	2000	Anders Heilsberg 등 (Microsoft)	Java C++ Visual Basic	객체지향적 웹 응용 프로그램

4) 컴파일러 언어와 인터프리터 언어
- 고급언어를 기계어로 번역해주는 도구, 또는 논리적 장치

방식 / 특징	컴파일러 방식	인터프리터 방식
번역방법	프로그램 전체 번역	실행되는 행 단위로 번역
장점	한번 번역하면 빠른 시간 내에 전체를 실행가능	큰 기억 장치가 필요하지 않으며 번역 과정이 비교적 간단함
단점	프로그램의 일부를 수정하는 경우에도 전체를 컴파일해야 함	반복문이 많은 경우 매 반복 때마다 번역해야 함

결과물	목적 언어로 된 프로그램	실행의 결과
적용언어	FORTRAN, Pascal, COBOL, Ada, C 등	Lisp, SNOBOL 4, Prolog 등



<고급 언어 프로그램의 전체 실행 순서>

2. 주요 프로그래밍 언어

1) 포트란
- 엔지니어, 수학, 과학 등을 위한 수식 계산에 강한 2세대 언어임
- 1966년 ANSI(American National Standard Institute)에 의해 FORTAN IV로 표준화됨
- 1977년 FORTRAN 77로 버전 업 됨

▸ FORTRAN의 특징
- 최초의 고급 언어 중 하나, 다른 언어의 설계에 많은 모델이 됨
- 매우 단순하고 간결하며, 수치와 계산에 강함
- 실행 시 자료의 크기가 고정, 동적 배열이나 재귀 호출 등은 지원하지 않음
- 같은 장소에 서로 다른 변수 이름을 가지게 할 수 있음. 이명효과(aliasing)나 부작용(side effects)등의 가능성을 내포함

```
1        PROGRAM MAIN
2          PARAMETER(MAXSIZ=99)
3          REAL A(MAXSIZ)
4    10    READ(5, 100, END=999) K
5   100    FORMAT(I5)
6            IF(K.LE.0 .OR. K.GT.MAXSIZ) STOP
7            READ *, (A(I), I=1, K)
8            PRINT *, (A(I), I=1, K)
9            PRINT *, 'SUM=', SUM(A, K)
10           GO TO 10
11  999    PRINT *, "ALL Done"
12         STOP
13         END
14  C SUMMATION SUBPROGRAM
15         FUNCTION SUM(V, N)
16          REAL:: V(N) ! New style declaration
17          SUM=0.0
18          DO 20 I=1, N
19              SUM=SUM+V(I)
20   20         CONTINUE
21              RETURN
22              END
```

2) 코볼
- 주된 목적은 사무 처리에 적합하도록 설계됨
- 1960년 COBOL- 60의 최초 버전이 발표됨
- 1968년 ANSI 표준 승인됨
- 1974년 최종 버전인 ANSI COBOL이 완성되어 사용됨

▸ COBOL의 특징
- 컴퓨터와 독립적으로 설계됨
- 사무처리를 목적으로 설계되어, 파일 처리에서 강점을 보임
- 일상적인 영어 문장 구조로 쉬운 가독성을 보임
- 자연어(영어) 문장 구조는 프로그램의 커지는 결과를 초래, 효율성이 떨어짐
- COBOL의 근황: 아직까지도 쓰이고 있는 언어로 Y2K문제로 인한 '코볼 붐'을 초래한 적 있음

3) BASIC(Beginner's All-purpose Symbolic Instruction Code)
- 1965년 케머니와 쿠르츠에 의해 개발된 프로그래밍 언어임
- 1970년대 중반 컴퓨터를 이용한 소규모 업무처리 등에 쓰임
- BASIC의 단점을 보완한 Quick Basic, 마이크로소프트의 Visual Basic이나 EXCEL 등에서 사용됨

▸ BASIC의 특징
- 초보자도 쉽게 배우고, 다양한 작업을 할 수 있음
- 코볼처럼 대화체의 프로그래밍 언어이면서도 작업량이 작음
- 마이크로소프트의 BASIC 지원으로 현재도 계속 발전되고 있음
- 대부분의 OS에서 지원함

4) Pascal
- 스위스에 니클라우스 워스(N. Wirth) 교수에 의해 1971년 탄생함
- 구조적 프로그래밍과 알고리즘 학습에 적합함
- 1990년대 초반까지 대부분의 컴퓨터 관련 교재로 채택됨
- 파스칼 컴파일러로 Borland의 '터보 파스칼'이 유명함

▸ Pascal의 특징
- 교육용으로 적합한 언어이며. 알고리즘과 프로그래밍의 연습에 알맞은 문법임
- 구조적인 프로그램의 작성 가능함
- 컴파일러의 효율성이 좋고, 컴파일러를 만들기가 쉬움
- 객체지향 등의 새로운 개념이나 기술을 채택하여 새로운 언어로 발전함

▸ Pascal의 근황
- C 언어에 비해 엄격한 구조를 요구, 전문적인 프로그래머는 파스칼보다는 C나 C++ 언어를 선호함

5) Lisp
- 1960년 MIT의 존 매카시(McCarthy)에 의해 개발된 언어임
- 자연어 처리와 인공지능 분야에 강함

▸ Lisp의 특징
- 대화식 구성의 인터프리터 방식으로 사용자의 요구에 그 즉시 결과값이 나오는 방식임
- 임의의 자료형을 만들고, 결과 값으로 사용 가능함
- 프로그램과 자료가 동일한 형태로 취급됨
- 메모리를 동적으로 관리하는 기능이 있음

6) Prolog (PROgramming in LOGic)
- 1972년 코왈스키(R. Kowalski) 등에 의해 개발된 논리형 인공지능 언어임
- 비 절차적 논리형 언어임
- 여러 기종의 컴퓨터에서 구형이 가능함
- 일본의 5세대 컴퓨터 프로젝트에 사용된 주요 인공지능 언어임

▸ Prolog의 특징
- 사실(fact), 규칙(rule), 질문(question)들로 프로그램이 구성됨
- 사실과 규칙들을 데이터베이스로 구성, 프로그램 실행은 자료에 대한 질문의 응답 형식으로 진행됨
- 인터프리터 언어이며 대화식의 명령 방식으로 작동함
- 사용자의 질문에 답하기 위해 추론 엔진(inference engine)을 사용하고 사용자가 사실과 규칙 등을 입력함

7) C
- 1972년 데니스 리치가 설계, PDP-11에서 구현시킴
- 기존의 언어에 비해서 신뢰성, 규칙성, 간소성 등을 가짐
- 저급언어의 기능 구현 가능함
- 융통성과 이식성(portability)이 좋아 고급프로그래밍 언어의 개발 속도 향상에 기여함
- 풍부한 연산자와 데이터 형(data type)으로 응용 소프트웨어의 개발 속도를 향상시키는데 기여함

▸ C언어의 특징
- 매우 유연한 구조로 되어 있음
- 대부분의 운영체제에서 기본으로 지원함
- 고급언어와 저급언어 양쪽의 장점을 모두 포함함
- 모든 실행 단위가 함수로 구성됨

8) C++
- 객체지향 프로그래밍(Object-Oriented Programming : OOP)을 지원하기 위해 탄생함
- 대다수의 응용 프로그램을 만들 때 가장 많이 사용함

- 강력함과 편리함의 양쪽 장점을 골고루 내포하여 효율성을 제공함

▸ C++의 특징
- C의 유연성에 객체지향의 편리성을 접목시킴
- 기존의 C언어로 개발된 모든 프로그램을 수정 없이 사용 가능함
- C언어에 익숙해지면 C++도 빠른 적응 가능함
- 대부분의 운영체제에서 C++를 지원함

9) 제 4세대 언어 (Fourth-Generation Programming Language: 4GL)

(1) Power Builder (Power Builder)
- 사이베이스(Sybase)사에서 만든 객체지향 개발 언어임
- 주로 데이터베이스용 응용 프로그램을 작성에 쓰임
- Visual Basic, Delphi 등과 같이 연동 가능함
- 그림을 그리듯이 디자인, 최소한의 프로그래밍으로 효율성을 극대화 할 수 있음

▸ Power Builder의 특징
- 클라이언트/서버 응용 프로그램 개발 환경: 주로 데이터베이스 관련 프로그램으로 서버는 데이터베이스와 관련된 작업, 클라이언트는 사용자 입력/출력 등의 사용자 부분을 처리하는 프로그램을 말함
- 4GL의 RAD(Rapid Application Development) Tool: 그래픽 유저 인터페이스 환경에서 간단한 마우스 동작 만으로 빠른 개발이 가능함
- 다양한 운영체제를 지원하는 개발 환경: Windows, UNIX, Mac 등의 다양한 버전을 지원함

▸ Power Builder의 정리
- 특정 업무용 프로그램을 여러 운영 체제에서 사용 가능함
- 편리한 데이터베이스 관련 기능으로 인기가 많음
- 프로그램 작성 후 배포 시에 여러 가지 불편한 점이 많음
- 데이터베이스 이외의 프로그램에서는 다른 4GL언어보다 성능이 떨어지고 불편함

(2) Delphi
- 볼랜드사의 Object Pascal을 RAD Tool로 변형하여 1995년 출시됨
- 기본적인 내부 구조는 Object Pascal을 이용함
- Windows의 각종 컨트롤이나 도구 등은 VCL(Visual Component Library)이라는 개념으로 지원함

▸ Delphi의 특징
- Object Pascal 언어와 컴파일러를 사용함
- VCL이라는 편리한 컨트롤 제공함
- C나 C++로 만들 수 있는 모든 프로그램을 만들 수 있음. 즉, Windows와 100% 호환이 가능하여 Windows OS하 에서는 강력한 능력을 발휘함

(3) Visual Basic
- 마이크로 소프트에서 만든 BASIC을 사용하는 RAD Tool임
- 다른 RAD Tool과 마찬가지로 사용하기 쉬운 환경과 빠른 시간 안에 원하는 프로그램을 작성하도록 만들어짐

▸ Visual Basic의 특징
- 가장 배우기 쉽고 접하기 쉬운 Basic을 사용함
- 한글 지원이 우수함
- 인터프리터 언어와 컴파일러 언어 양쪽의 장점을 모두 가짐
- 다른 마이크로소프트사의 도구들을 가장 간편하게 적용할 수 있음

(4) Visual C++
- 마이크로소프트사에서 만든 C++ 컴파일러 이름임
- 기존의 C++에 여러 가지 Windows의 기능을 추가함

▸ Visual C++의 특징
- Microsoft Foundation Class: MFC 란 강력하고 방대한 라이브러리를 제공함
- Windows의 모든 기능을 가장 강력하고 자연스럽게 사용 가능함
- Delphi 나 Visual Basic 같은 마우스로 하는 디자인적 요소는 거의 미비한 반면 프로그래밍의 코딩, 디버깅, 프로젝트 관리 면에서 탁월한 기능을 발휘함
- 비주얼 모델러라는 강력한 객체지향 설계도구를 포함함

3. 프로그래밍 개발 절차

1) 사용자 요구 사항 분석
- 사용자의 필요를 파악하고 프로그램을 통해 해결할 문제가 무엇인지 확인하는 단계

2) 프로그램 설계
- 실제 코딩을 시작할 때 사용할 논리를 프로그래머가 대략 그려내는 단계
- 알고리즘 설계라고도 함

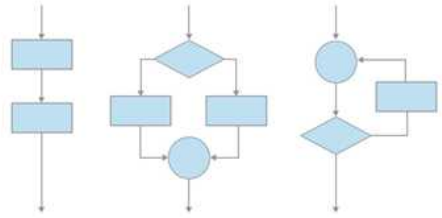
▸ 알고리즘의 특성
- 알고리즘 명령을 수행하면 유한한 횟수를 거친 후 종료해야 함
- 알고리즘의 각 단계와 명령은 명확하게 정의되어야 함
- 알고리즘은 데이터 입력이 0 또는 그 이상이어야 함
- 알고리즘은 한 가지 이상의 결과를 출력
- 알고리즘은 효과적이어야 함

→ 유한한 시간 내에 정확히 수행할 수 있을 정도로 단순해야 함을 의미

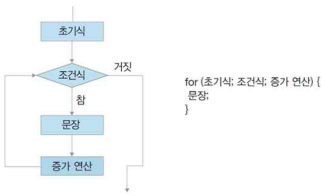
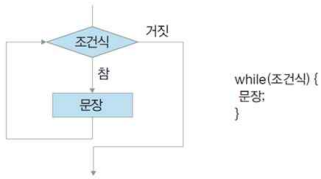
▸ 프로그램 제어 흐름 유형

순차 구조	프로그램 코드 순서대로 실행

선택 구조	프로그램이 다음에 무엇을 해야 하는지를 결정하는 분기 구조
반복 구조	조건이 만족하지 않을 때까지 계속 반복



〈순차 구조〉 〈선택 구조〉 〈반복 구조〉

반복구조 – for	반복구조 – while	반복구조 –do while
		

3) 코딩 및 컴파일

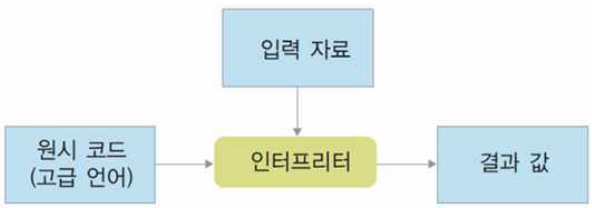
- 코딩: 프로그래밍 언어로 프로그램을 작성하는 단계
- 컴파일: 고급 언어로 작성된 명령문을 기계어로 바꾸는 단계

(1) 컴파일러를 이용한 방식

- 프로그램 전체를 한번에 기계어로 번역하는 방식
- C언어, 코볼, 포트란, 파스칼 등의 언어에서 사용

(2) 인터프리터를 이용한 방식

- 프로그램을 한 행씩 읽어 번역과 실행을 동시에 하는 방식
- 베이직 등의 언어에서 사용



(3) 하이브리드 방식

- 컴파일러와 인터프리터를 함께 이용하는 방식
- 리스프, 스노볼4, APL, 프롤로그, 자바 등의 언어에서 사용

4) 디버깅 및 시험

- 프로그램이 포함하는 모든 오류를 찾아내 제거하는 것
- 오류에는 구문 오류와 논리 오류가 있음

구문 오류	틀린 문자를 입력하거나 문법에 맞지 않는 명령문을 사용했을 때 발생하는 오류
논리 오류	제어 구조의 부적절한 사용으로 발생하는 오류

- 시험은 알파 테스트와 베타 테스트로 구분

알파 테스트	완성된 프로그램을 개발 환경에서 시험하는 방법
베타 테스트	특정 고객이 고객에 쓰는 환경에서 시험하는 방법