CHAPTER 1.

**프로그래밍 언어의 종류**

* 기계어(Machine Language)
  + 0과 1의 조합으로 구성된 2진수.
  + 컴퓨터가 이해할 수 있는 유일한 언어.
* 어셈블리어(Assembly Language)
  + 기계어는 너무 복잡하고 어려우니 이를 대신해서 특정 영어 단어를 이용.
  + Add같은 용어를 어셈블러 프로그램으로 기계어로 변환해서 실행하는 방식.
  + CPU가 바뀔 때마다 해당 CPU의 어셈블리어를 학습해야 하는 번거로움.
* 고급 언어(High Level Language)
  + 어셈블리어의 어려움을 극복한 언어.
  + 우리가 배우는 언어. 사람이 이해하기 쉬운 문법
  + 컴파일러나 인터프리터가 고급언어를 기계어로 변환해 실행.
  + BASIC, FORTRAN, COBOL, C, C++, JAVA, Python 등
  + C와 C++은 포인터도 있어서 저급언어의 특징도 있음

\* Colossus(1943): 최초의 전자식 컴퓨터

**프로그래밍 기본 용어\_1**

* 원시 프로그램(source program, source code)
  + 프로그래머가 작성한 프로그램.
  + C언어로 작성하면 .c 확장자.
* 컴파일(compile)
  + 고급 언어들을 기계로 번역하기 위해서 번역기에 해당하는 컴파일러나 통역가에 해당하는 인터프리터가 필요(번역가는 처음부터 끝까지 한번에 결과를 번역, 통역가는 매 대화마다 번역).
  + C언어는 컴파일러 방식(대표적인 컴파일러는 GCC C, Clang, MSVC).
  + 컴파일을 하면 .obj라는 오브젝트 파일이 생성됨, 실행파일이 생성되는게 아님.
* 링킹(linking)
  + 컴파일 후 만들어진 오브젝트 파일(모듈)들을 연결
  + 사용한 라이브러리와 연결하는 작업
  + 링킹을 하면 실행파일(.exe)이 생성됨
* 디버깅(debugging)
  + 컴파일과 링킹 과정에서 소스 코드에 오류가 있을 수 있는데 이를 에러 또는 버그라고 함.
  + 디버깅은 이 버그를 수정해 실행 가능하도록 하는 작업

**C 언어란?**

* 절차적 언어: 프로그램을 순서대로 실행하는 구조.
  + 위->아래, 함수 단위로 짜여 있음
* 시스템 프로그래밍에 적합: 운영체제, 데이터베이스, 임베디드 시스템 등 하드웨어에 가까운 곳에서 잘 쓰임
* 고성능 컴퓨팅 분야에 적함: 다른 프로그래밍 언어의 기반이 되는 언어(낮은 수준의 제어).
* 1970년 Bell 연구소에서 Ken Thompson이 유닉스를 만듦.
* 같은 연구소의 Dennis Ritchie가 어셈블리 언어를 되도록 적게 사용하는 유닉스를 만들기 위해서 C언어를 개발. 유닉스의 90% 이상이 C로 만들어짐.
* 컴파일러 회사마다 다른 c언어를 만들게 되어 표준 문법의 필요성이 제기됨 -> ANSI에서1898년 표준 c언어를 완성 후 1990년에 발표.