CPP Base

CHAPTER 1
* 표준 C++ 프로그램의 중요성

모든 C++ 컴파일러에 의해 컴파일이 가능하고, 동일한 실행 결과를 보장해야 하기에 표준의 중요성은 높아지고 있다.

* C++언어의 주요한 설계 목적
-호환성
1) 기존의 C프로그램을 그대로 사용할수 있게 C언어의 문법적 체계를 계승함
2) C언어로 작성된 object file이나 라이브러리를 c++프로그램에서 링크하여 사용 가능해야 함

- 객체 지향 개념 도입
1) 캡슐화 : 데이터를 캡슐로 싸서 외부의 접근으로부터 데이터를 보호
2) 상속 : 객체 정의 클래스 사이에 상속 관계를 두어 자식 클래스의 객체 생성시 자식클래스에 선언된 멤버뿐 아니라 부모 클래스에 선언된 멤버들도 함께 가지고 탄생함.
3) 다형성 : 하나의 기능이 상황에 따라 다양한 기능을 할 수 있도록 함. ex) 연산자 중복.

- 타입 체크를 통해 오류를 줄이고 디버깅을 도움

- 실행 시간의 효율성 저하를 최소화

* C 언어에 추가한 기능
-함수 중복, 디폴트 매개 변수, 참조, 참조에 의한 호출, new와 delete연산자(동적 메모리), 연산자 재정의, 제네릭 함수와 클래스

*C++ 컴파일러
-C++ 소스 프로그램이 문법에 맞게 작성되었는지 검사하고, 기계어 코드로 변환하여 Object file을 생성함. (문법에 맞지 않으면 오류 반환)

*링킹
컴파일후 생성된 Object 파일은 다른 표준 라이브러리나 다른 c++프로그램에 대한 함수, 객체, 데이터에 대한 참조 표시만 존재한다. 이러한 또 다른 파일의 함수, 객체, 데이터를 포함하여 실행에 필요한 모든 기계어 코드를 확보 후 하나의 실행 파일로 만드는 과정

*실행 및 디버깅
링킹을 통한 exe파일은 컴퓨터에서 바로 실행 가능하다. 그리고 프로그램에서 발생된 오류 위치를 발견하거나 문제를 찾아 수정하는 과정이 디버깅이다.

*C++ 표준 라이브러리
- 개발자들이 불러 쓸 수 있는 다양한 종류의 함수와 클래스가 컴파일된 Object file들임. 따라서 c++ 표준 컴파일러 사이에 호환이 가능하며 c++ 표준 라이브러리는 컴파일된 object file로만 제공되며 원시 소스코드는 제공되지 않음.

-C++ 표준 라이브러리는 다음 3개의 그룹과 기타 기능으로 나뉨
1) C라이브러리 : 기존 C+ 라이브러리를 수용하여 C++에서 사용할 수 있게함
2) C++ 입출력 라이브러리 : 콘솔 및 파일 입출력을 위한 함수의 클래스들이며, 제네릭 프로그래밍을 지원하기 위해 템플릿으로 작성됨.
3) C++ STL 라이브러리 : 제네릭 프로그래밍을 지원하기 위해 템플릿으로 작성된 유용한 함수와 클래스를 포함하는 라이브러리임.