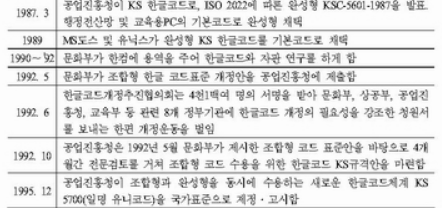
1. 한글 코드의 표준화 과정을 조사한다. 특히 Unicode가 어떻게 표준화되었는지 조사한다.



(출처) KISDI 방송통신동향 통계자료

Window 95 이전, DOS시절에는 결과를 출력하기 위해서 BIOS(Basic Input/Output System)를 직접 제어하거나 ISA(Industry Standard Architecture)인터페이스와 같은 별도의 하드웨어를 사용해야 했다.

이와 같은 것을 방지하여 공통된 표준이 필요하여 유니코드가 만들어지게 되었다.

유니코드는 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준이다. 유니코드의 목적은 현존하 문자 인코딩 방법들을 모두 유니코드로 교체하려는 것이다. 기존의 인코딩들은 그 규모나 범위 면에서 한정되어 있고, 다국어 환경에서는 서로 호환되지 않는 문제점이 있었다. 유니코드가 다양한 문자 집합들을 통합하는 데 성공하면서 유니코드는 컴퓨터 소프트웨어의 국제화와 지역화에 널리 사용되게 되었다

또 KS5601, ISO646, ISO2422, UTF8, UTF16은 무엇인가?

KS5601은 한글 완성형 표준이다(한글 2350자를 표현할 수 있다.)

전각문자/기호/자소/선문자 0xA180 ~ 0xAFFF  
한글영역 0xB0A1 ~ 0xC9FF  
한자영역 0xCA80 ~ 0xFDFF

**ISO/IEC 646**는 7[비트](https://ko.wikipedia.org/wiki/%EB%B9%84%ED%8A%B8_(%EB%8B%A8%EC%9C%84))의 [문자 코드](https://ko.wikipedia.org/wiki/%EB%AC%B8%EC%9E%90_%EC%BD%94%EB%93%9C)를 규정하는 [국제 표준화 기구](https://ko.wikipedia.org/wiki/%EA%B5%AD%EC%A0%9C_%ED%91%9C%EC%A4%80%ED%99%94_%EA%B8%B0%EA%B5%AC)(ISO) 표준이며, 이것을 바탕으로 각 나라 언어판의 문자 코드가 규정된다.

[미국](https://ko.wikipedia.org/wiki/%EB%AF%B8%EA%B5%AD)의 규격인 [ASCII](https://ko.wikipedia.org/wiki/ASCII)를 국제화한 체제이다. [라틴 문자](https://ko.wikipedia.org/wiki/%EB%9D%BC%ED%8B%B4_%EB%AC%B8%EC%9E%90)와 [숫자](https://ko.wikipedia.org/wiki/%EC%88%AB%EC%9E%90)를 중심으로 하는 공통 부분과, 각 나라 규격으로 자유롭게 할당할 수 있는 부분(ASCII에서는 기호에 해당)으로 나뉘어 있어 [유럽](https://ko.wikipedia.org/wiki/%EC%9C%A0%EB%9F%BD) 각 나라에서는 이 부분에 [발음 구별 기호](https://ko.wikipedia.org/wiki/%EB%B0%9C%EC%9D%8C_%EA%B5%AC%EB%B3%84_%EA%B8%B0%ED%98%B8)를 더해 문자를 할당하여 자국어의 문장을 [인코딩](https://ko.wikipedia.org/wiki/%EC%9D%B8%EC%BD%94%EB%94%A9)할 수 있게 되어 있다.

UTF8과 UTF16은 유니코드 문자 인코딩 방식 중 하나이다.

UTF-8 인코딩은 유니코드 한 문자를 나타내기 위해 1바이트에서 4바이트까지를 사용한다. 예를 들어서, U+0000부터 U+007F 범위에 있는 [ASCII](https://ko.wikipedia.org/wiki/ASCII) 문자들은 UTF-8에서 1바이트만으로 표시된다. 4바이트로 표현되는 문자는 모두 [기본 다국어 평면](https://ko.wikipedia.org/wiki/%EA%B8%B0%EB%B3%B8_%EB%8B%A4%EA%B5%AD%EC%96%B4_%ED%8F%89%EB%A9%B4)(BMP) 바깥의 유니코드 문자이며, 거의 사용되지 않는다. [UTF-16](https://ko.wikipedia.org/wiki/UTF-16)과 UTF-8 중 어느 인코딩이 더 적은 바이트를 사용하는지는 문자열에서 사용된 코드 포인트에 따라 달라지며, 실제로 [DEFLATE](https://ko.wikipedia.org/wiki/DEFLATE)와 같은 일반적인 압축 알고리즘을 사용할 경우 이 차이는 무시할 수 있을 정도이다.

UTF-16은 사용되는 [기본 다국어 평면](https://ko.wikipedia.org/wiki/%EA%B8%B0%EB%B3%B8_%EB%8B%A4%EA%B5%AD%EC%96%B4_%ED%8F%89%EB%A9%B4) (BMP, Basic multilingual plane)에 속하는 문자들은 그대로 16비트 값으로 인코딩이 되고 그 이상의 문자는 특별히 정해진 방식으로 32비트로 인코딩이 된다. 기본 다국어 평면은 U+0000 에서 U+FFFF 에 놓인 문자를 담고 있다. 이 영역에는 우리가 쉽게 생각할 수 있는 문자들이 포함되며, 한글, 한자 등은 모두 여기에 포함되어 있다. 이 영역에는 서러게이트 문자(surrogate)들이 준비되어 있어 16비트 이상의 문자를 표현할 때를 대비해 놓았다.

2. 자판을 중심으로 표준화가 낳을 수 있는 문제점을 조사한다. 실제 우리가 쓰는 자판은 최대한 글쇠가 엉키지 않게 만듦으로써 오히려 입력속도가 느리다(QWERT). 이에 따라 새로운 자판(이름을 조사)이 나왔지만 익숙하지 않아서 아직 보급되지 않고 있다. 한글 자판도 2벌식과 3벌식이 있는데 비슷한 문제점이 있다.

퀴티 키보드의 단점을 보안하여 입력속도를 향상시킨 키보드 : **드보락** 키보드

- 어디서 사용되어지는지, 어느 시대에 사용되어 지는지에 따라 표준이 효율적일 수도 비효율적일수도 있다.한번 표준이 확정지어지면 사람들이 이미 쓰던 표준에 익숙해져있어 변화를 수용하기 어렵다.

- 처음에는 컴퓨터의 처리속도가 입력속도를 따라오지 못하여 표준이 정해졌으나 환경이 달라진 지금에서는 다른 표준이 필요하게 되었다. 시대, 환경에 따라 정해져 있는 표준이 비효율적일 수 있다.

3. 현재 XML을 이용하여 각 분야의 문서표준화가 진행되고 있다. 그 예와 현재 진행상황을 설명하라!! (예 eBook, vXML, 전자상거래 ...), 한 분야만 잡아서 조사한다.

eBook에서 사용하는 표준

EPUB

**electronic publication**의 줄임말로, ePub, ePUB, EPub등으로 표기하기도 하는 "epub"는, IDPF(International Digital Publishing Forum) - 국제 전자 출판 포럼 -에 의해서 제정된, ebook(전자책)출판을 위한 공개표준이다. 이 표준을 따르는 파일은 *.epub*확장자를 가진다.

EPUB는 **reflowable** 콘텐츠를 위해서 디자인 되었는데, 이는 reflowable란 디스플레이 장치의 화면에 최적화된 문서출력을 보장함을 의미한다. 일반 데스크탑 컴퓨터에서는 이 reflowable가 중요하지 않겠지만 디스플레이의 크기가 다양한 모바일기기에서는 매우 중요한 기능이다. 예를 들어, 전자문서의 또다른 표준인 PDF는 reflowable를 지원하지 않는데, ebook 단말기나 스마트폰에서 문서를 볼경우 화면 스크롤, 화면 확대에 따른 출력품질 저하등의 문제가 발생할 수 있다.

아마존같은 경우 **AZW**라는 독자적인 포맷을 가지고 ebook 컨텐츠를 개발한다. epub는 (HTML 처럼)공개된 표준으로 자유롭게 사용할 수 있다.

CSS를 이용해서, 문서 스타일을 정의할 수 있다. 이는 epub의 컨텐츠가 XHTML로 만들어지기 때문으로, 유연한 문서 스타일 정의가 가능하다는 장점이 있다.

4. 핸드폰과 스마트폰에서 한글 문자를 입력하는 방법은 표준화되어 있는가? 현재 표준화 상황을 조사하라

현재 가장 널리 사용되고 있는 스마트폰 입력 방식은 쿼티이다. 아이폰은 쿼티 입력 방식 외에는 사용할 수 없고, 안드로이드 스마트폰에서도 기본적으로 쿼티 입력을 제공하고 있다. 2011년 3월에 한손으로 입력하기 쉽도록 방송통신위원회가 스마트폰의 한글 자판에 대한 국가 표준으로 천지인과 나랏글, 스카이 이 세가지 방식 모두를 표준으로 제정하였다.

5. C, C++, Python, PHP, HTML, XHTML의 표준화 과정을 설명하라.

**C)**

1969년에 Bell연구소에서 유닉스에서 동작하는 고급 언어를 만들기 시작하였고 이것이 B언어이다.  
B언어를 만든 데니스 리치는 B언어에 자료형을 추가한 New B언어를 만들었으며  
그리고 1972년과 1973년에 이를 개선한 C언어가 만들어졌다..  
이러한 C언어는 1983년 표준화 과정을 통해 ANSI C 표준이 만들어지고 시대 에 맞게 발전한 표준화 작업을 계속하는 중이다 대표적으로 C99, C11의 표준이 있으며 C++의 발전에 발맞춰 계속 발전하고 있다.

**C++)**

1979년에 Bjarne Stroustrup이 C언어를 바탕으로 만들었다. 속도가 느리지만 객체지향 언어인 Simula의 컨셉을 C언어에 접목시키면서 C with Classes라는 이름으로 만들었다.

1983년에 가상함수, 연산자 오버로딩, 참조, const, new/delete 연산자 등을 포함한 새로운 기능을 추가되면서 이름을 C++로 바꾸었다.

1989년에 다중 상속, 추상 클래스, static 멤버 함수, const 멤버 함수 및 protected 멤버 등의 새로운 기능을 추가하였다

1990년에 C++ 2.0에 추가된 기능 외에 템플릿, 예외, 네임스페이스, new casts 및 bool 타입 등의 새로운 기능들이 추가된 '주석이 달린 C++ 레퍼런스 메뉴얼'이 발표되었고 1991년에 컴파일러 프론트엔드가 Cfront 3.0으로 업그레이드되었으며, The C++ Programming Language의 첫 번째 개정판이 발행되었다.

1998년에 최종 승인된 C++의 첫 공식 표준안으로, C++ 2.0 이후 C++가 안정적으로 정착하고 수요가 폭증함에 따라 여러 회사에서 상업용으로 컴파일러를 만들었고, 이에 따라 코드가 컴파일러에 따라 중구난방이 되는 것을 막기 위해 표준화의 필요성이 제기된다. 그래서 1990년에 ANSI에서 C++ 표준화를 위한 위원회가 설립되었으나 1991년에 [ISO](https://namu.wiki/w/ISO)로 넘기면서 스트로우스트루프를 비롯한 프로그래밍 언어학자 및 컴파일러 제조사들을 모아 표준화 작업을 하기 시작한다.그렇게 C++의 표준안이 ISO/IEC 14882:1998라는 공식 명칭으로 처음 제정되었으며, 이를 줄여 C++98이라 부른다.

C++11) 새로운 기능들이 많이 포함되었던 C++TR1이 정식 표준으로 채택되지 못 해서 대부분의 프로그래머들에게 있어서 C++은 C++98 이후 13년 간 정체된 언어나 다름 없었다.  
  
[2011년](https://namu.wiki/w/2011%EB%85%84)  8월 12일에 최신 컴퓨팅 환경을 지원하기 위해 급격한 변화의 첫 단추를 끊은 새로운 표준안인 C++11이 ISO/IEC 14882:2011라는 정식 명칭으로 최종 승인되었다. 스마트 포인터, 정규표현식, 멀티쓰레드 등등 중요한 신 개념과 편의 기능들이 많이 추가되었다.

C++14) [2014년](https://namu.wiki/w/2014%EB%85%84) 8월에 정식 명칭으로 C++14이 제정되었다. C++11에 추가된 요소들을 다듬고 확장하는 데 치중한 비교적 마이너 표준안이다.

C++17) [2017년](https://namu.wiki/w/2017%EB%85%84) 12월에 C++**1z**로 알려졌던 C++17 표준이 제정되었다.  
C++11의 추가 요소들을 다듬고 향상시키는데 주력했던 C++14에 비해, C++17은 표준 라이브러리의 새로운 기능에 비교적 주력한다 파일 시스템, [STL](https://namu.wiki/w/STL) 병렬 알고리즘 처리 등이 추가되었다

Python)

Python은 1991년 Guido Van Rossum이 발표한 인터프리터, 객체지향, 동적 타이핑 대화형 언어이다.

파이썬은 1980년대 말 고안되어 네덜란드 CWI의 귀도 반 로섬이 [1989년](https://ko.wikipedia.org/wiki/1989%EB%85%84) 12월 구현하기 시작하였다. 이는 역시 SETL에서 영감을 받은 ABC 언어의 후계로서, 예외 처리가 가능하고, 아메바 OS와 연동이 가능하였다. 반 로섬은 파이썬의 주 저자로 계속 중심적 역할을 맡아 파이썬의 방향을 결정하여, 파이썬 공동체로부터 '자선 종신 이사'의 칭호를 부여받았다. 이 같은 예로는 리눅스의 리누스 토발즈 등이 있다.

파이썬 2.0)

파이썬 2.0은 2000년 10월 16일 배포되었고, 많은 기능이 추가되었다. 그중 전면적인 [쓰레기 수집기](https://ko.wikipedia.org/wiki/%EC%93%B0%EB%A0%88%EA%B8%B0_%EC%88%98%EC%A7%91_(%EC%BB%B4%ED%93%A8%ED%84%B0_%EA%B3%BC%ED%95%99))(GC, Garbage Collector)탑재와 [유니코드](https://ko.wikipedia.org/wiki/%EC%9C%A0%EB%8B%88%EC%BD%94%EB%93%9C) 지원이 특징적이다. 그러나 가장 중요한 변화는 개발 절차 그 자체로, 더 투명하고 공동체 지원을 받는 형태가 되었다.

파이썬 3.0)

파이썬3000(혹은 파이썬3k)이라는 코드명을 지닌 파이썬의 3.0버전의 최종판이 긴 테스트를 거쳐 2008년 12월 3일자로 발표되었다. 2.x대 버전의 파이썬과 하위호환성이 없다는 것이 가장 큰 특징이다. 파이썬 3의 주요 기능 다수가 이전 버전과 호환되게 2.6과 2.7 버전에도 반영되기도 하였다.

파이썬 공식 문서에서는 "파이썬 2.x 는 레거시(낡은 기술)이고, 파이썬 3.x가 파이썬의 현재와 미래가 될 것"이라고 요약을 했는데, 처음 배우는 프로그래머들은 파이썬 3으로 시작하는 것을 권장하고 있다.[[8]](https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%B4%EC%8D%AC#cite_note-8)

사전형과 문자열형과 같은 내장자료형의 내부적인 변화 및 일부 구형의 구성 요소 제거.

표준 라이브러리 재배치.

향상된 유니코드 지원. (2.x 에서는 유니코드를 표현하기 위해 u"문자열" 처럼 유니코드 리터럴을 사용했지만 3.0 부터는 모든 문자열이 유니코드이기 때문에 "문자열" 처럼 표현하면 된다.)[[9]](https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%B4%EC%8D%AC#cite_note-9)

한글 변수 사용 가능.

print 명령문이 print() 함수로 바뀌게 되었다.

HTML, XHTML) 출처 : <https://namu.wiki/w/HTML>

