글로벌 소프트웨어 교육 현황 및 교육 도구 동향

박효민*

소프트웨어 산업의 중요성이 나날이 높아지고 있는 상황에서, 소프트웨어 인재를 양성하기 위해 어릴 때부터 소프트웨어 교육을 배울 수 있도록 의무화하는 나라가 많아지고 있다. 이러한 세계 각국의 소프트웨어 교육 현황에 대해 살펴보고, 이를 통해 앞으로 소프트웨어 교육을 의무화해야하는데 있어 필요한 사항을 정리해 보고자 한다.

Ⅰ. 서론

Ⅱ. 소프트웨어 교육 개요
1. 소프트웨어 교육의 활성화 배경과 목적
2. 글로벌 소프트웨어 교육 현황

Ⅲ. 소프트웨어 교육 도구
1. 교육용 프로그래밍 언어
2. 주요 소프트웨어 교육 플랫폼

Ⅳ. 결론

* 한국인터넷진흥원 개인정보침해대응팀 주임연구원(phm88@kisa.or.kr)

Ⅰ. 서론

요즘 같은 디지털 시대에서 소프트웨어는 일상생활에서 뿐만 아니라 여러 가지 산업 분야에서 매우 다양하게 이용되고 있다. 일반적으로 소프트웨어를 이용하는 PC나 스마트폰뿐만 아니라 시계, 자동차 등에서도 소프트웨어를 탑재하여 이용하고 있기 때문이다. 이렇게 소프트웨어의 이용이 증가함에 따라 경쟁력이 중요해지면서 소프트웨어 산업의 중요성이 부각되고 있다. 또한 소프트웨어는 전적으로 사람이 개발하기 때문에 소프트웨어 산업에서는 전문 인재의 양성이 가장 중요한 과제가 되었다.

이러한 과제를 성공적으로 수행하기 위해서는 전문 인재 양성을 위한 교육이 필요하다. 소프트웨어 전문 교육 과정은 일반 프로그래밍 언어에 대한 사용법이나 개발을 효율적으로 할 수 있는 자료구조나 알고리즘 등에 대한 내용이다. 하지만 처음부터 전문 교육을 이수하기는 어렵기 때문에 우선적으로 배워야할 기초 소프트웨어 교육에 대한 필요성이 높아졌다. 그로 인해 어릴 때부터 쉽게 소프트웨어를 배울 수 있도록 학생을 대상으로 한 소프트웨어 교육이 활성화되고 있으며, 학생이 쉽게 이해하고 실습할 수 있도록 교육용 프로그램 언어와 교육 플랫폼의 보급 또한 증가하는 추세이다.

이러한 소프트웨어 교육에 대한 상승추세는 세계적으로 확산되고 있다. 이미 미국과 영국, 일본 등 여러 국가에서는 학생대상으로 의무적으로 소프트웨어 교육을 실시하도록 하고 있으며, 정부에서 소프트웨어 교육 커리큘럼을 만들어 제공하고 있다. 우리나라는 현재 의무교육을 실시하지 않고 있지만 2015년 중학교 입학생부터 소프트웨어 교육이 의무화될 예정이다. 또한 초등학교와 고등학교의 경우 소프트웨어 각각 2017년, 2018년부터 소프트웨어 교육이 의무화될 예정이다.

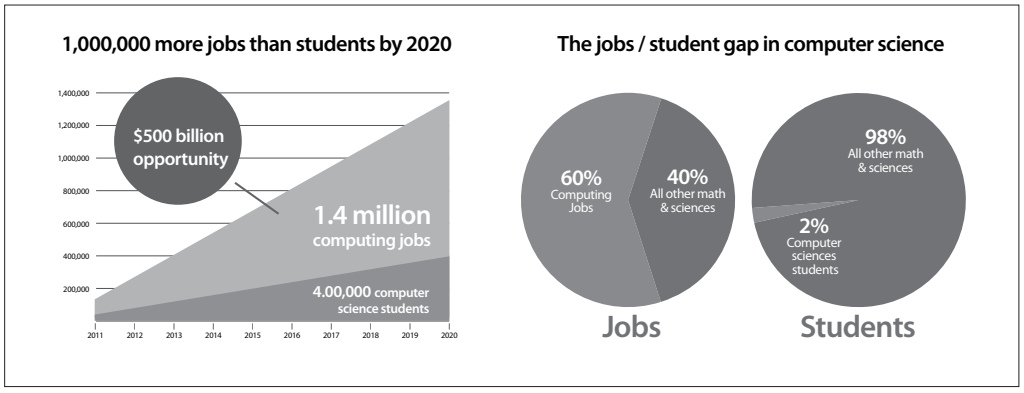
이에 본고에서는 소프트웨어 교육에 대한 내용과 세계 각국의 소프트웨어 교육 현황과 실제 교육에 사용되는 교육용 프로그래밍 언어와 및 소프트웨어 교육 플랫폼에 대해 살펴보고자 한다. 그리고 이를 통해 앞으로 소프트웨어 교육을 추진함에 있어서 필요한 사항들을 정리해 보기로 한다.

Ⅱ. 소프트웨어 교육 개요

1. 소프트웨어 교육의 활성화 배경과 목적

앞서 소개한 것과 같이 소프트웨어 기반 환경이 조성됨에 따라 소프트웨어에 대한 중요성이 높아지고 있다. 그리고 자연스럽게 컴퓨팅과 관련된 직업에 대한 수요가 높아지고 있는 상태이며 앞으로 더욱 증가할 추세이다. 미국의 노동통계국(Bureau of Labor Statistics)의 2010년-2012년 보고서에 따르면 2020년까지 컴퓨팅관련 직업이 컴퓨터 과학을 전공하는 학생의 수보다 훨씬 많아지는 현상이 나타날 것이라고 전망하였다.

이러한 전망은 [그림 1]의 좌측 표로 나타나며, 우측 표를 통해 컴퓨팅관련 직업은 다른 이공계 직업보다 많지만, 학생이 컴퓨터과학 과목을 기피하고 있는 것으로 나타난다. 이와 같은 불균형 문제를 해결하기 위해서는 컴퓨터 과학을 전공하는 학생의 수를 늘려야하기 때문에 학생이 소프트웨어에 대한 흥미를 가질 수 있는 환경을 조성하여야 한다.



출처: Code.org

[그림 1] 향후 2020년까지 컴퓨팅 관련 직업과 컴퓨터 과학 전공 학생 수 차이비교

남녀노소 누구나 소프트웨어에 흥미를 가질 수 있는 환경을 조성하기 위해서 대학교나 비영리 단체가 소프트웨어 교육 현장에 뛰어들어 다양한 소프트웨어 교육용 플랫폼과 교육과정을 개발하고 있다. 또한 소프트웨어 교육을 위해 정부가 직접 교육과정을 만들어 시행하는 나라가 늘어나고 있다.

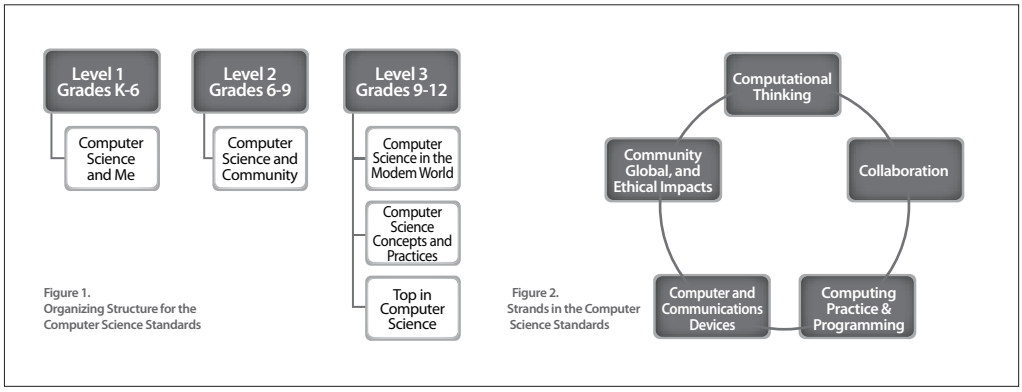
2. 글로벌 소프트웨어 교육 현황

1) 미국

현재 미국은 세계적으로 소프트웨어 교육이 가장 앞서나가고 있다고 평가 받고 있다. 이는 컴퓨터 과학 교사 협회(CSTA, Computer Science Teachers Association)에서 2011년에 발표한 'K-12 컴퓨터 과학 표준(K-12 Computer Science Standards)'을 통해 학생에게 어떻게 컴퓨터 과학을 교육할 것인지에 대한 방향을 제시하고 있기 때문이다. 또한 이를 통해 컴퓨터 과학을 필수과목으로 지정하여 교육하는 학교가 늘어나고 있다.

K-12에서는 컴퓨터 과학을 핵심적인 학과로 인지하고 있으며, 그 이유를 다음과 같이 5가지 제시하고 있다. 첫째, 컴퓨터과학은 지적으로 중요한 학문이다. 둘째, 컴퓨터 과학은 학생에게 다양한 진로를 제공한다. 셋째, 컴퓨터 과학은 문제를 해결하는 방법을 알려 준다. 넷째, 컴퓨터 과학을 통해 이와 연결된 다른 과학 분야에 대해 쉽게 이해할 수 있다. 다섯째, 컴퓨터 과학은 모든 학생이 참여할 수 있다.

그리고 K-12는 전체적인 교육 커리큘럼을 [그림 2]의 좌측 그림과 같이 크게 3가지 단계로 나누어 정의하고 있다. 그리고 우측과 그림과 같이 각 단계를 5가지로 영역으로 나누어 각 영역별로 세부적인 표준을 제시하고 있다.



출처: CSTA K-12 Computer Science Standards(2011)

[그림 2] K-12의 교육과정 분류

1단계인 '컴퓨터 과학과 나(Computer Science and Me)'는 저학년부터 6학년까지의 학생을 교육대상으로 선정하였으며, 학년에 따라 2개의 과정으로 교육내용을 나누어 제시하고 있

다. 그 내용은 〈표 1〉과 같다.

〈표 1〉 K–12 컴퓨터 과학 표준 교육 1단계 교육내용

구분	Grade K–3	Grade 3–6
컴퓨터적 사고 (Computational Thinking)	1. 나이에 맞는 문제를 해결하기 위한 기술적 자원(퍼즐, 논리적 사고 프로그램)을 사용하기 2. 단계별로 생각, 아이디어, 이야기를 설명하기 위해 쓰기 도구, 디지털 카메라, 그리기 도구를 사용하기 3. 컴퓨터를 사용하지 않고 출생날짜와 같은 정보로 학생을 순서대로 정렬할 수 있다는 것을 이해하기 4. 소프트웨어는 컴퓨터의 동작을 제어하기 위해 작성된다는 것을 인식하기 5. 0과 1이 정보를 표현할 수 있다는 것을 보여주기	1. 알고리즘 문제 해결에 필요한 기본 단계를 이해하고 사용하기 2. 컴퓨터 없이 간단한 알고리즘(검색, 정렬 등)을 이해하여 적어보기 3. 비트열이 문자와 숫자정보를 표현하기 위해 사용될 수 있는 지 보여주기 4. 문제를 해결하는데 시뮬레이션이 어떻게 사용될 수 있는 지 기술하기 5. 하나의 큰 문제에 접근하는 동안 고려해야 하는 하위 문제의 목록을 만들기 6. 컴퓨터 과학 및 기타 분야 사이의 연관성을 이해하기
협동 (Collaboration)	1. 선생님이나 가족, 친구의 도움을 받아 다른 사람과 전자적으로 정보를 수집하고 대화 해보기 2. 기술을 이용하여 친구, 선생님, 다른 사람과의 협력 및 공동 활동을 해보기	1. 생산성 기술 도구(워드 프로세싱, 스프레드 시트, 프리젠테이션 소프트웨어)를 사용하여 개인 및 공동 쓰기, 대화, 출판 활동 해보기 2. 온라인 자원(이메일, 온라인 토론, 공동 웹 환경)을 이용하여 솔루션이나 제품을 개발하는 공동 문제해결 활동에 참가해보기 3. 팀워크와 협업이 문제 해결과 혁신에 도움이 된다는 것을 인지하기
컴퓨팅 연습과 프로그래밍 (Computing Practice & Programming)	1. 나이에 맞는 문제를 해결하기 위한 기술 적 자원을 사용하기 2. 교육 과정을 통해 학습에 도움이 되는 적절한 멀티미디어 자원(대화 형 책과 교육용 소프트웨어)을 사용 3. 선생님, 가족, 또는 친구의 도움으로 적절한 멀티미디어 제품을 만들어보기 4. 간단한 작업을 수행하기 위한 명령문을 만들어보기 5. 컴퓨팅 기술을 사용하는 일을 확인해보기 6. 마인드맵 도구를 이용하여 정보를 수집하고 조직하기	1. 기술적 자원(계산기, 휴대 기기, 교육용 소프트웨어 및 웹 툴 등)을 사용하여 문제 해결 및 자율 학습하기 2. 부족한 기술 보충을 위해 범용 생산성 도구 및 주변기기를 사용해보기 3. 기술적 도구(멀티미디어 및 텍스트 작성, 프리젠 테이션, 웹 도구, 디지털 카메라, 스캐너)를 사용하여 개인이나 공동 쓰기, 대화, 출판 활동 해보기 4. 다양항 디지털 기기를 사용하여 데이터를 수집, 조작해보기 5. 일상생활에서의 활동을 단계별 명령의 집합으로 나타내보기 6. 블록기반 비주얼 프로그래밍 언어를 사용하여 문제의 해결책을 제시해보기 7. 개별학습을 위해 원격 정보에 접근하거나 통신하기 위해서 컴퓨팅 기기를 사용해보기 8. 웹페이지에서 하이퍼링크로 이동해보고 검색엔진에서 간단하게 검색해보기 9. 컴퓨팅 지식과 활용을 필요로 하는 직업군을 확인하기

구분	Grade K–3	Grade 3–6
컴퓨터와 통신 장비 (Computers and Communications Devices)	1. 컴퓨터와 관련 장치가 잘 동작하도록 입력 및 출력 장치를 조작하기	1. 키보드 및 다른 입력 및 출력 장치를 사용하는 적절한 수준의 실력 보여주기 2. 일상생활에 보급되어 있는 컴퓨터와 컴퓨팅 기술을 이해해보기(음성 메일, 전자 레인지, 무선 인터넷, 모바일 컴퓨팅 기기, GPS 시스템 등) 3. 컴퓨터 사용 중에 발생할 수 있는 간단한 하드웨어 및 소프트웨어 문제를 확인하기 위한 전략을 만들기 4. 네트워크를 통해 여러 출처에서 정보가 오고 있다는 것을 이해하기 5. 인간과 기계를 구분할 수 있는 요소들을 생각해보기 6. 컴퓨터가 인간의 지능적 행동을 따라할 수 있음을 이해하기(로봇, 음성 및 언어 인식 등)
커뮤니티, 글로벌 및 윤리적 영향 (Community Global, and Ethical Impacts)	1. 기술 시스템 및 소프트웨어를 사용할 때 책임 있는 디지털 윤리의식을 알기 2 기술을 사용하는 동안 긍정적이거나 부정적인 사회적, 윤리적 행동을 구분하기	1. 기술 및 정보의 사용에 있어서 관련된 문제와 부적절한 사용에 대해 토론하기 2. 개인 생활과 사회에 관련된 기술(소셜 네트워킹, 사이버 왕따, 모바일 컴퓨팅 및 통신, 웹 기술, 사이버 보안 및 가상화)의 영향력을 확인하기 3. 전자적 정보 자료에서 발생하는 정확성, 관련성, 적절성, 포괄성, 편견을 평가하기 4. 컴퓨터와 네트워크에 관련된 윤리적 문제를 이해하기(접근성, 보안, 개인 정보 보호, 저작권 및 지적 재산권)

2단계인 '컴퓨터 과학과 커뮤니티(Computer Science and Community)'은 6학년부터 9학년까지를 교육대상으로 선정하였으며, 〈표 2〉와 같은 교육내용을 제시한다.

〈표 2〉 K–12 컴퓨터 과학 표준 교육 2단계 교육내용

구분	교육내용
컴퓨터적 사고 (Computational Thinking)	1 솔루션을 설계하는 문제 해결의 기본 단계를 절차적으로 사용하기 2. 문제 해결에 관한 병렬화 과정을 기술하기 3. 컴퓨터로 처리 할 수 있는 명령으로 알고리즘을 정의하기 4. 같은 문제를 해결하는데 사용되는 여러 가지 알고리즘을 평가해기 5. 검색 및 정렬 알고리즘을 만들어보기 6. 일련의 명령 절차를 설명하고 분석하기 7. 소리, 그림 숫자, 텍스트 등을 포함한 다양한 방법으로 데이터를 표현하기 8. 문제의 구조 및 데이터(그래프, 차트, 네트워크 다이어그램, 플로우 차트)를 시각적으로 표현하기 9. 학습과 연구를 위해서 콘텐츠별로 모델 및 시뮬레이션을 상호작용 시켜보기 10. 모델링 및 시뮬레이션을 사용하여 해결할 수 있는 문제의 종류를 평가하기 11. 어떤 컴퓨터 모델이 현실세계를 정확하게 나타내는 지 분석하기 12. 한 문제를 하위 문제로 분해하기 위해 추상화 사용하기 13. 고급언어, 명령어 세트 및 로직 회로를 포함한 컴퓨터의 계층 구조와 추상화의 개념을 이해하기 14. 숫자, 논리, 집합의 기능과 같은 수학의 요소와 컴퓨터 과학 사이의 관련성을 생각해보기 15. 컴퓨터적 사고에 대한 학문 간에 응용 프로그램의 예를 제공하기

구분	교육내용
협동 (Collaboration)	1. 생산성 / 멀티미디어 도구 및 주변 장치를 적용하여 교육과정에서 협력과 도움에 대해 지원하기 2. 공동 디자인, 개발, 게시 활동을 통해 교육 과정의 개념을 설명하고 통신 기술 자원을 사용하여 표현물(비디오, 팟 캐스트, 웹 사이트) 만들기 3. 페어 프로그래밍, 프로젝트 팀과 같은 그룹 학습 활동에 참여하여 동료, 전문가, 다른 사람과 협력해보기 4. 협력활동에서 다양한 관점에서의 유용한 피드백을 제공하거나 피드백을 통합하고 수용하기
컴퓨팅 연습과 프로그래밍 (Computing Practice & Programming)	1. 다양한 작업을 수행하고 문제를 해결하는 데 적합한 도구 및 기술 자원을 선택하기 2. 개인의 생산성과 교육 과정에 도움을 주는 멀티미디어 도구와 다양한 주변 장치를 사용하기 3. 디자인, 개발, 게시 및 설명과 교육 과정의 개념을 통신 기술 자원을 사용하여 본 제품 (예를 들어, 웹 페이지, 모바일 응용 프로그램, 애니메이션). 4. 알고리즘의 이해 및 실용적인 응용 프로그램 시연하기 5. 반복문, 조건문, 논리식, 변수 및 함수를 포함하는 프로그래밍 언어를 사용하여 문제의 솔루션을 구현하기 6. 암호, 암호화 및 보안 트랜잭션을 사용하여 개인 정보 보안에 대한 모범 사례를 제시하기 7. 컴퓨터 과학을 통해서 향상된 간학문적 직업들을 확인하기 8. 개방형 문제 해결과 프로그래밍을 분석할 수 있는 성향을 보여주기 9. 하나의 프로그램을 다중 실행하여 출력된 데이터를 모으고 분석하기
컴퓨터와 통신 장비 (Computers and Communications Devices)	1. 컴퓨터가 프로그램을 실행할 수 있는 장치임을 인식하기 2. 프로세서를 포함하는 다양한 장치들을 구분하기 3. 하드웨어와 소프트웨어 사이의 관계를 이해하기 4. 기술에 관한 의사소통 시, 정확한 용어를 사용하기 5. 일상생활에서 컴퓨터 사용 중 발생하는 반복적인 문제를 확인하고 해결하기 위한 전략을 세우기 6. 컴퓨터 시스템과 네트워크의 주요 구성요소고 기능을 설명하기 7. 기계와 인간을 구분하고 의사소통하는 방식의 차이를 설명하기 8. 컴퓨터가 사용하는 지적 행동(로봇의 동작, 음성 및 언어 이해 및 컴퓨터 비전)의 모델에 대해 설명하기
커뮤니티, 글로벌 및 윤리적 영향 (Community Global, and Ethical Impacts)	1. 정보기술을 사용할 때의 법적 · 윤리적 행동을 보여주고, 잘못 사용한 결과에 대해 토론하기 2. 시간에 따른 정보 기술의 변화와 이러한 변화에 따른 교육, 직장, 사회에 미치는 영향에 대해 설명하기 3. 인간의 문화에 대해 퓨터 기술의 긍정적이고 부정적인 영향을 분석하기 4. 실제 세계의 문제에 관한 전자 정보 자원의 정확성, 관련성, 적절성, 포괄성, 편견을 평가하기 5. 컴퓨터와 네트워크(보안, 개인 정보 보호, 소유권 및 정보 공유)에 관련된 윤리적 문제를 설명하기 6. 글로벌 경제의 컴퓨팅 자원의 불평등한 분배를 통해 야기되는 평등, 접근, 힘의 문제에 대해 토론하기

3단계인 '개념 적용과 실제 솔루션 생성(Applying Concepts and Creating Real-World Solutions)'은 9학년부터 12학년까지의 학생을 교육대상으로 선정하였으며, 3개의 과정으로 나누어 제시한다. 각 과정은 현대 사회의 정보과학(3A, Computer Science in the Modern World), 정보과학의 원리(3B, Computer Science Principles), 정보과학의 주제(3C, Topic in Computer Science)이다. 3B과목은 선택교과이며, 3A 과정을 선행해야 한다. 자세한 내용은 〈표 3〉 와 같이 정리할 수 있다.

〈표 3〉 K–12 컴퓨터 과학 표준 교육 3단계–3A/3B 교육내용

구분	3A (Computer Science in the Modern World)	3B (Computer Science Principles)
컴퓨터적 사고 (Computational Thinking)	1. 복잡한 문제를 단순한 부분으로 나누기 위해 이미 정의된 함수, 파라미터, 클래스, 메소드를 사용하기 2. 소프트웨어 문제를 해결하기 위해 사용한 소프트웨어 개발 절차(설계, 코딩, 검사, 확인)를 기술하기 3. 절차, 선택, 반복, 재귀는 알고리즘의 블록을 만드는 방법임을 설명하기 4. 수집된 대규모 데이터를 분석하기 위한 기법들을 비교하기 5. 2진수와 16진수의 표현 사이의 관계를 기술하기 6. 디지털 정보의 다양한 형식에 대한 표현과 장단점을 분석하기 7. 각종 데이터가 컴퓨터 시스템에 저장되는 방식을 설명하기 8. 자연 현상을 이해하고 표현하기 위한 모델링과 시뮬레이션을 사용하기 9. 문제의 복잡성을 관리하기 위해 추상화 값에 대해 토론하기 10. 큰 문제를 해결하는 전략으로 병렬 처리의 개념을 설명하기 11. 인간의 의도를 작품으로 나타내는 예술분야에서 컴퓨팅의 주 기능에 대해 기술하기	1. 문제를 다루기 쉬움, 다루기 어려움, 컴퓨터로 해결할 수 없음으로 분류하기 2. 다루기 어려운 문제에 대하여 적절한 해결책으로 휴리스틱 알고리즘의 가치를 설명하기 3. 고전적 알고리즘을 비판적으로 검토하고 독창적인 알고리즘으로 구현하기 4. 알고리즘을 효율성, 정확성, 명확성의 측면에서 평가하기 5. 자연과 인체의 복잡성을 보다 잘 이해하기 위하여 데이터 분석을 활용하기 6. 간단한 데이터 구조들(배열, 리스트)과 그 사용법을 비교하고 대조하기 7. 다양한 2진수 형태(명령, 수, 문자, 사운드, 이미지 등)의 해석에 대하여 토론하기 8. 과학적 가설의 수립, 정교화, 검증을 돕기 위해 모델과 시뮬레이션을 사용하기 9. 모델링과 시뮬레이션을 통하여 데이터를 분석하고 패턴을 구별하기 10. 새로운 함수와 클래스를 정의함으로써 문제를 세분화하기 11. 프로세스를 나누어 쓰레드로 만들고 데이터를 병렬 스트림으로 나누어 동시성을 시연하기
협동 (Collaboration)	1. 소프트웨어 제품을 설계하고 개발하기 위하여 팀으로 작업하기 2. 프로젝트 팀 구성원과 의사소통을 위해 협업 도구(토론 게시판, 위키, 블로그, 버전 관리 등)를 사용하기 3. 컴퓨팅이 어떻게 전통적인 형태를 향상시키고 경험, 표현, 소통, 협동의 새로운 형태를 허용하는 것에 대해 기술하기 4. 협업이 소프트웨어의 설계와 개발에 어떠한 영향을 미치는지 확인하기	1. 협력 소프트웨어 프로젝트를 진행하는 동안에는 프로젝트 협력 도구, 버전 관리 시스템, 통합개발환경(IDE)을 사용하기 2. 소프트웨어 프로젝트 팀에 참여함으로써 소프트웨어 생명 주기를 경험하기 3. 다른 사람이 작성한 프로그램의 가독성과 재사용성을 평가하기
컴퓨팅 연습과 프로그래밍 (Computing Practice & Programming)	1. 다양한 웹 프로그래밍 설계 도구를 이용하여 웹페이지를 생성하고 조직하기 2. 모바일 응용 프로그램을 설계, 개발, 구현하기 위하여 모바일 기기나 에뮬레이터를 사용하기 3. 프로그램의 정확성을 보장하기 위하여 다양한 디버깅 및 검증 방법을 사용하기 4. 문제를 해결할 때마다 분석, 설계, 구현 기술과 같은 일련의 활동을 적용하기 5. 프로그래밍이 용이하도록 응용프로그램 인터페이스(API)와 라이브러리를 사용하기 6. 다양한 유형의 데이터를 사용하기 위하여 적절한 파일 포맷을 사용하기 7. 문제를 해결하고 시스템을 개발하기 위해 사용하는 다양한 프로그래밍 언어를 설명하기	1. 결과물(웹 디자인, 애니메이션, 비디오, 멀티미디어)을 만들기 위하여 고급 도구를 사용하기 2. 대규모 컴퓨터학적 문제(절차적 추상화, 객체지향 설계, 기능적 설계)를 세분화하기 위하여 추상화 도구를 사용하기 3. 프로그래밍 언어 수준과 응용 영역을 기반으로 프로그래밍 언어를 분류하기 4. 규모, 효율성, 보안에 대한 시스템 설계의 원리를 탐구하기 5. 암호화와 인증 전략을 구현할 때 보안의 원리를 적용하기

구분	3A (Computer Science in the Modern World)	3B (Computer Science Principles)
컴퓨팅 연습과 프로그래밍 (Computing Practice & Programming)	8. 프로세스의 실행 과정을 설명하기 9. 암호화, 암호학, 인증 기술을 검증함으로써 보안의 원리를 설명하기 10. 컴퓨팅이 중심인 다양한 직업을 탐구하기 11. 작은 규모 및 큰 규모의 데이터 집합을 수집하기 위한 기법을 설명하기 12. 수학적, 통계적 함수, 집합, 논리가 계산에 사용되는 방법을 설명하기	6. 미래의 직업과 미래에 남아있을 만한 현재의 기술을 예측하기 7. 자연과 인체의 복잡성을 보다 잘 이해하기 위하여 데이터 분석을 활용하기 8. 각각 다른 유형의 문제를 해결하기 위하여 다양한 데이터 수집 기법을 적용하기
컴퓨터와 통신 장비 (Computers and Communications Devices)	1. 모바일 기기와 교통수단(자동차, 비행기)에 탑재된 컴퓨터의 특성을 기술하기 2. 컴퓨터 하드웨어의 구입 및 업그레이드를 위한 범주를 만들기 3. 컴퓨터를 구성하는 기본 요소(입력, 출력, 처리, 저장 등)를 기술하기 4. 입력과 출력의 여러 가지 유형을 비교하기 5. 프로그램 실행에 필요한 하드웨어와 소프트웨어를 계층적으로 설명하기(컴파일러, 인터프리터, 운영체제, 네트워킹 등) 6. 일상생활에서 규칙적으로 발생하는 하드웨어와 소프트웨어의 문제를 진단하고 해결하기 위한 전략을 적용하기 7. 클라이언트와 서버간 통신 그리고 P2P 네트워크를 비교 및 대조하기 8. 컴퓨터 네트워크의 기본적인구성요소를 설명하기(서버, 파일보호, 라우팅, 스풀러와 큐, 공유자원, 오류 허용 범위) 9. 인터넷이 글로벌 커뮤니케이션을 용이하게 하는 이유를 설명하기 10. 인공지능과 로보틱스의 주요 응용분야를 설명하기	1. 응용 프로그램들에서 기능 변형이 미치는 영향을 토론하기 2. 하드웨어를 구분하고 설명하기(예, 물리적 계층, 논리 게이트, 칩, 부품) 3. 정확성, 속도, 구현의 용이성 등을 고려하여 가장 적절한 파일 포맷을 구분하고 선택하기 4. 네트워크 기능(지연시간, 대역폭, 방화벽, 서버 성능)에 영향을 미치는 문제들을 기술하기 5. 컴퓨터 모델링과 로보틱스를 통하여 인지적 행동의 개념을 설명하기
커뮤니티, 글로벌 및 윤리적 영향 (Community Global, and Ethical Impacts)	1. 소셜 네트워크에서의 적절한 행동과 부적절한 행동을 비교하기 2. 사업과 상거래에서 컴퓨터의 영향에 대하여 토론하기(상품의 자동추적, 자동화된 금융 거래, 전자거래, 클라우드 컴퓨팅) 3. 특별한 니즈를 가진 사람들에게 맞춤형 기술의 역할을 설명하기 4. 문화에 관한 기술의 긍정적, 부정적 영향에 대하여 비교하기(소셜 네트워킹, 뉴스와 공공 매체의 전송, 문화 간 상호 교류) 5. 인터넷 상에서 발견되는 정보의 신뢰성을 결정하기 위한 전략을 설명하기 6. 정보의 접근과 배포 권리 사이의 차이를 구분하기 7. 공유하거나 지적재산권을 보호하기 위해 사용되는 소프트웨어 여러가지 라이센스의 종류를 설명하기 8. 해킹과 소프트웨어 불법복제에 관련된 사회적, 경제적 의미를 토론하기 9. 제작되거나 공유된 소프트웨어의 장점과 단점에 대해 기술하기 (상업용 소프트웨어, 퍼블릭 도메인, 공개 소프트웨어 개발) 10. 컴퓨터 네트워크와 관련된 보안 및 사생활 침해 이슈에 대하여 기술하기 11. 중요한 정보에의 접근성에 대한 디지털 격차의 영향을 설명하기	1. 현대의 의사소통 미디어와 장치의 윤리적인 사용을 실천하기 2. 컴퓨터가 가져온 혁신의 장단점을 분석하기 3. 자동화에 의해 금융시장, 거래, 예측이 어떻게 변화될 것인지를 요약하기 4. 전산화로 인하여 사람들이 실제와 가상 조직, 기반 환경을 어떻게 만드는지 요약하기 5. 소프트웨어의 개발과 사용에 영향을 미치는 법과 규제를 이해하기 6. 개인 정보 보호 및 보안에 관한 정부 규제의 영향력을 분석하기 7. 오픈 소프트웨어, 프리웨어, 독점 소프트웨어 라이선스를 구분하고, 라이선스가 다양한 유형의 소프트웨어에 각각 어떻게 적용되는 지 구분하기 8. 국제 사회에서 컴퓨팅 자원의 분배 문제에 대하여 평등, 접근, 힘의 문제를 관련지어 보기

3C의 경우 상단의 교육과정과 다르게 다음과 같은 3가지 영역 중 1개를 선택할 수 있다.

첫째, AP 컴퓨터 과학 A. 고등학교 졸업 후 2~4년제 대학에서 컴퓨터과학 교육을 받고자 하는 학생을 위해 개설(AP, 2010년)된 이 과목은 문제 해결, 알고리즘 개발, 기초 데이터 구조 도입 등을 강조하고 있다. 1, 2단계와 3단계의 3A를 이수하여야 응시 가능하다.

둘째, 단일주제 심화 프로젝트과정. 웹 개발, 게임프로그래밍, 그래픽, 네트워크 기술, 객체지향 설계 및 코딩 등 다양한 과정이 열려 있으며, 반년 또는 1년 과정으로 운영된다. 1, 2단계를 이수한 학생과 일부과정은 3B과정 이수를 필요로 한다.

셋째, 전문인증관련 벤더 공급과정. A+, Quick Security+ 등의 산업자격증 과정이 있으며, 산업자격증을 취득했다는 것은 해당 기술에 대해 적절한 인재임을 증명하는 것이기 때문에, 고등학교 졸업 후 바로 취업을 하고자 하는 학생과 전문학교 졸업한 학생 또는 2년제 대학에 입학하고자 하는 학생을 위한 교육 과정이다.

2) 영국

영국은 2013년 9월에 발표된 National curriculum in England: computing programmes of study에 따르면 key stage를 1부터 4단계까지 나누어 5세부터 16세까지의 학생을 대상으로 소프트웨어 교육 커리큘럼을 제시하고 있다.

〈표 4〉 영국의 컴퓨터 프로그램 교육 내용

단계	교육 내용
Key stage 1	• 디지털 장치에서 프로그램으로 구현되어 있는 알고리즘과 정확하고 모호하지 않은 명령어를 통해 실행되는 프로그램에 대한 이해하기 • 간단한 프로그램을 작성하고 디버깅하기 • 간단한 프로그램이 어떻게 실행될지 예측하는 논리적 추론 사용하기 • 디지털 콘텐츠를 생성, 구성, 저장, 조작, 검색할 수 있는 기술을 활용하기 • 학교 이외에 정보기술을 보편적으로 사용하고 있음을 인식하기 • 기술을 안전하게 사용할 수 있도록 기술지원을 받을 수 있는 연락처를 확인하기
Key stage 2	• 특정 목적을 해결하기 위한 프로그램을 설계, 작성, 디버깅해보기 • 프로그램에서 변수와 다양한 입 · 출력 형태를 사용하여 순서적, 선택적, 반복적 기능 활용하기 • 동일한 알고리즘이 어떻게 실행되는지 설명하고 알고리즘과 프로그램의 에러를 확인할 수 있도록 논리적 사고를 활용하기 • 인터넷 등과 같은 컴퓨터 네트워크 이해하기 • 검색기술을 효과적으로 사용하기 • 특정 디지털 기기에서 프로그램을 설계하고 제작하기 위해서 여러 종류의 소프트웨어를 선택, 사용, 결합하기 • 정보기술을 안전하고 책임감 있게 사용, 용인될 수 없는 없는 행동 인지 등

단계	교육 내용
Key stage 3	• 실제 문제나 물리 시스템에서의 상태와 행동을 모델로 컴퓨터적 추상화를 통해 설계, 사용, 평가하기 • 동일문제에 대해 여러 알고리즘을 비교하여 사용할 수 있도록 주요 알고리즘에 대해 이해하기 • 데이터구조의 적절한 사용, 모듈프로그램 설계 및 개발을 위해 2개 이상의 프로그래밍 언어를 사용하기 • 프로그램에서 사용되는 Boolean 논리와 2진수로 표기 체계를 이해하기 • 컴퓨터 시스템을 구성하는 하드웨어 및 소프트웨어가 어떻게 통신하는 지 이해하기 • 컴퓨터 시스템에서 명령어가 어떻게 저장되고 실행되는 지, 그림, 소리와 같은 다양한 형태의 데이터가 어떻게 2진수로 표현되고 조작되는 지 이해하기 • 디지털 산출물에 대한 신뢰성, 디자인, 사용성을 개선하기 위해 사용자들에게 생성, 재사용, 개정, 용도변경하여 제공하기 • 온라인 개인정보를 보호하는 기술적 보호조치의 범위에 대해 이해하기
Key stage 4	• 컴퓨터과학, 디지털 미디어, 정보기술에 대한 능력, 창의성, 지식 계발시키기 • 개개인의 분석, 문제해결, 설계, 컴퓨터적 사고 능력의 계발 및 적용 • 온라인 개인정보를 보호하는 새로운 방법 등을 포함하여, 기술적 보호조치가 어떻게 변화하고 있는 지 이해하기 ※ 이 수준에서는 높은 수준의 연구 분야 또는 전문 직종으로 진출 할 수 있도록 충분한 수준의 정보 기술 및 컴퓨터 과학을 공부할 수 있는 기회가 있어야 함

Ⅲ. 소프트웨어 교육 도구

1. 교육용 프로그래밍 언어

1) 교육용 프로그래밍 언어의 정의

교육용 프로그래밍 언어란 범용 프로그램을 위해 프로그램을 개발하기 위한 도구라기보다는 우선적으로 프로그래밍을 학습하는 도구로서 설계된 프로그래밍 언어이다(Wikipedia). 명확하게 교육용 언어라고 정해진 범위나 틀은 없다. 일반 프로그래밍 언어도 교육용으로 사용할 수는 있겠지만 언어의 개발 목적이 교육용인지 소프트웨어 제작인지에 따라 교육용 프로그래밍 언어를 구분할 수 있다.

교육용 프로그래밍 언어 통해 얻을 수 있는 장점은 기본적인 알고리즘을 이해할 수 있어 어떤 사건을 해결하는데 있어 알고리즘을 통해 사고하고, 더욱 쉽게 해결할 수 있는 능력을 키워준다. 또한 프로그래밍 언어는 각 언어끼리 영향을 주고받아 문법이 비슷한 언어가 많다. 그렇기 때문에 교육용 프로그래밍 언어를 통한 경험을 통해 일반 프로그래밍 언어를 접할 때

쉽게 습득할 수 있다.

2) 교육용 프로그래밍 언어의 특징

일반 프로그래밍 언어는 각 언어마다 복잡한 문법과 함수의 사용법을 알고 있어야 사용이 가능하다. 하지만 교육용 프로그래밍 언어는 대부분 문법이 간단하거나 텍스트가 아닌 시각적인 방법을 사용함으로써, 나이가 어리거나 컴퓨터 과학 비전공자도 쉽게 접근하여 배울 수 있도록 설계되었다. 하지만 교육을 목적으로 만들어진 언어이기 때문에 지원하는 기능에는 한계가 있다.

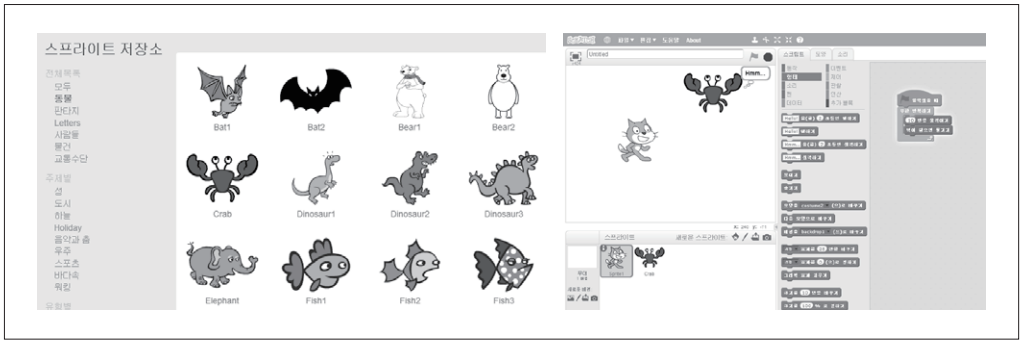
〈표 5〉 일반 프로그래밍 언어와 교육용 프로그래밍 언어와의 특징 차이

구분	일반 프로그래밍 언어	교육용 프로그래밍 언어
인터페이스	대부분 텍스트 기반	대부분 비주얼 기반
문법구조	복잡	간단
적용범위	하드웨어 제어 ~ 어플리케이션	어플리케이션
사용목적	소프트웨어 개발	프로그래밍 교육

3) 주요 교육용 프로그래밍 언어

(1) SCRATCH

MIT 미디어 랩에서 2007년에 개발한 블록형 프로그래밍 언어이다. 만 8세 부터 16세까지의 아동을 대상으로 설계되었으며, 작은 명령 단위인 블록 조각을 서로 조립하여 프로그래밍 할 수 있다. 프로그래밍 내용을 공유할 수 있으며, 2014년 9월 현재 600만 개 이상의 프로젝



출처: Scratch 홈페이지

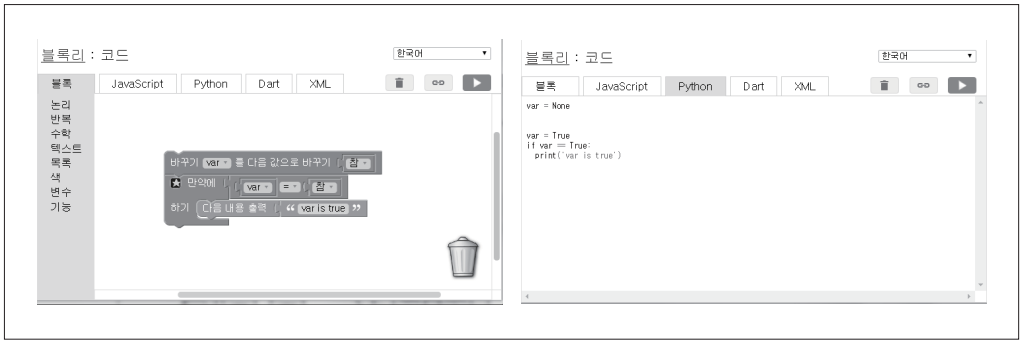
[그림 3] SCRATCH의 웹기반 개발환경

트가 공유되고 있다. 2013년 5월에 2.0버전이 발표되었으며 오프라인 개발환경, 클라우드 저장, 사운드 편집, 그래픽 편집, 스프라이트 복사 기능 등이 추가되었다.

기본적인 변수와 제어문을 사용을 할 수 있으며, 스프라이트를 객체로 사용하여 각각 다른 명령을 넣어 프로그래밍을 할 수 있다.

(2) BLOCKLY

Google에서 개발한 웹기반 블록형 프로그래밍 언어이다. Scratch와 비슷하지만 Blockly로 작성된 프로그램을 javascript, Python, Dart, XML과 같은 일반 프로그래밍 언어로 변환해주는 기능이 있어 일반 프로그래밍 언어에 대한 이해를 도와준다.



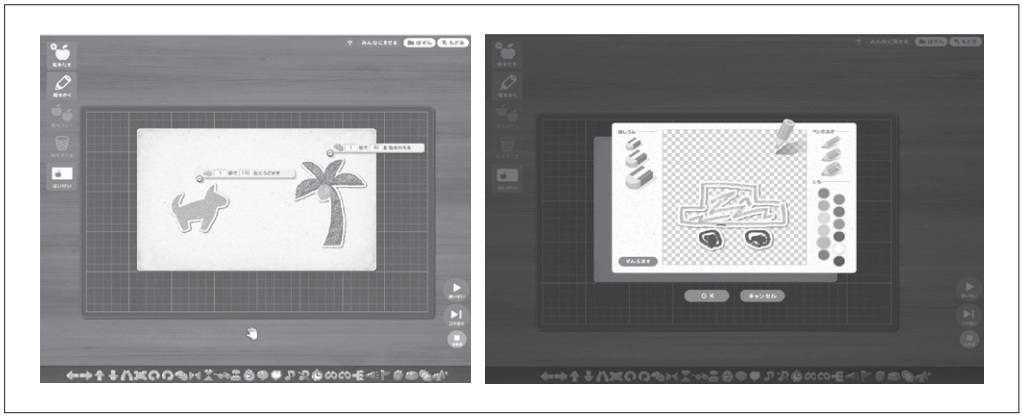
출처: Blockly 홈페이지

[그림 4] BLOCKLY의 언어 변환 기능

Blockly는 다양한 샘플이 준비되어 있으며, 퍼즐, 미로, 거북이 길 찾기, 오리배 게임, 그래프 그리기 등을 통해 Hour of Code와 같은 형태로 문제를 풀어볼 수 있다. 마지막에는 javascript 기반으로 직접 텍스트기반 실습을 해보면서 일반 프로그래밍 언어를 쉽게 접근할 수 있도록 도와준다.

(3) プログラミン(PROGRAMMIN)

일본의 행정기관인 문부 과학성(文部科學省)에서 제작한 교육용 프로그래밍 언어이다. 기본적으로 Scratch를 모티브로 하고 있어 지원하는 기능은 비슷하며, 여기에 일본어 제공 및 일본풍의 아기자기한 인터페이스를 통해 일본의 저 연령대 아이들이 쉽게 접근할 수 있도록 제작된 것이 특징이다.



출처: プログラミン 홈페이지

[그림 5] プログラミン 개발환경

(4) SMALL BASIC

Microsoft에서 2008년에 개발되었으며, 기존의 Basic 언어를 교육용으로 쉽게 배울 수 있도록 개발한 언어이다. Scratch나 Blockly와는 다르게 텍스트 기반 프로그래밍 언어이기 때문에 사용자가 문법을 알고 있어야 하지만 개발환경에서 자동완성 기능을 제공하여 쉽게 접근할 수 있다. 또한 변수는 전역변수로만 선언 및 사용 가능하며, branch와 subroutine기능을 제공한다.

개발사에서 직접 커리큘럼을 제공하고 있어, 프레젠테이션 파일만 다운받아 바로 교육이 가능하다. Small Basic에 숙련이 되면 기능이 더 많은 Visual Basic을 배우도록 하고 있으며, 기존에 Small Basic으로 작성한 코드를 Visual Basic로 바꿔주는 기능을 제공하여 자연스럽게 다음 언어를 배울 수 있도록 해준다.

2. 주요 소프트웨어 교육 플랫폼

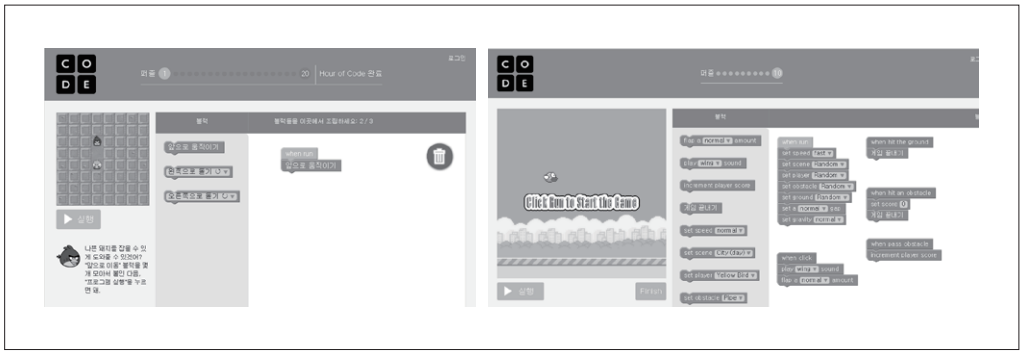
1) Code.org

2012년 8월에 설립된 미국의 비영리 단체로 빌 게이츠, 마크 주커버그, 잭 도로시 등과 같은 소프트웨어 산업의 유명인사와 버락 오바마 미국대통령까지 출연하여 소프트웨어 교육을 장려하는 동영상이 공개되어 급속도로 유명해 졌다.

저 연령대의 학생과 프로그래밍 입문자를 위해 만들어진 튜토리얼 교육(Hour of Code)을

통해 블록형 프로그래밍 언어로 간단하게 프로그래밍할 수 있다. 튜토리얼 교육이 끝나면 수료증을 만들 수 있는 페이지를 제공하며, Beyond Hour를 통해 다양한 프로그래밍 언어에 대한 교육 자료를 확인할 수 있다.

선생님에게 제공되는 교육 자료도 있는데, 교실에서 학생에게 가르치는 방법이나 초 · 중 · 고등학교의 소프트웨어 교육 커리큘럼 등이 그것이다. 그리고 오프라인에서 실행하고 있는 소프트웨어 캠프 활동 정보도 제공하고 있다.



출처: Code.org

[그림 6] Code.org의 Hour of Code와 Flappy Code

2) Codecademy

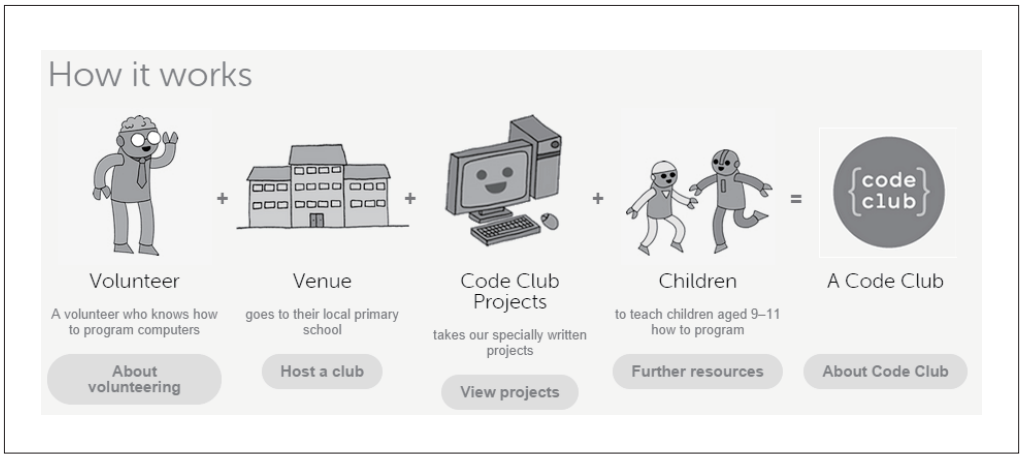
Codecademy는 온라인 인터렉티브 플랫폼으로서 프로그래밍 입문자를 대상으로 Python, PHP, jQuery, javascript, Ruby, HTML, CSS과 같은 일반 프로그래밍 언어 대한 교육 클래스가 준비되어 있다.

홈페이지에서 직접 코드를 작성하여 결과를 바로 확인 할 수 있는 개발환경을 제공하고 있어 프로그램을 설치하지 않아도 실습해볼 수 있으며, 각 단계마다 자세한 설명과 힌트를 통해 일반 프로그래밍 언어를 쉽게 배울 수 있는 것이 장점이다.

3) Codeclub

2012년에 설립된 영국의 비영리 단체로, 9세~11세 아이들을 대상으로 코드 교육을 실시한다. 코드클럽은 온라인상에서 직접 프로그래밍을 하는 것이 아니고 오프라인 상에서 이루어지고 있으며, 소프트웨어 교육을 할 수 있는 사람이 지원하여 방과 후 학교를 찾아가서 학생에게 직접 프로그래밍 교육을 실시한다. 영국 전체에 1,970개의 코드클럽이 만들어져 있으며,

27,000명 이상의 아이들이 교육을 받았다.



출처: www.codeclub.org.uk

[그림 7] 코드클럽 운영 과정

Ⅳ. 결론

앞서 이야기하였듯이 소프트웨어의 필요성은 날이 갈수록 높아지고 있으며, 그와 걸맞게 소프트웨어 교육에 대한 필요성도 높아지고 있다. 하지만 우리나라에서의 초 · 중 · 고등학교 교육과정에서는 컴퓨터에 대한 간단한 내용만 포함되어 있으며, 컴퓨터 과학과 관련된 내용으로 분리된 과목은 없는 상태이다.

이렇게 컴퓨터 과학이 기본 교육과정에 포함되어 있지 않기 때문에 학생들은 대학교에 진학하여 전공으로써 배우거나 전문 교육 학원을 통해 배워야 하는 실정이다. 하지만 대학교의 경우 컴퓨터과학을 전공하는 신입생 수는 매년 감소하고 있는 추세이다. 또한 학원의 경우 프로그래밍 언어에 대해서만 교육하거나 자격증 시험 취득을 위한 문제 풀이 교육이 대부분이기 때문에 컴퓨터과학에 대해 심도 있게 교육하는 곳은 드물다. 이렇게 우리나라는 컴퓨터 과학에 대한 기본적인 교육과정이 부족하기 때문에 어릴 때부터 컴퓨터 과학을 접할 기회가 많지 않아 쉽게 흥미를 가질 수 없는 환경이다.

그러므로 컴퓨터 과학에 대한 바람직한 인식을 확산시키고 흥미를 유발하기 위해 기본 교육과정의 개설과 지속적인 홍보가 필요하다. 그리고 기본 교육과정부터 대학교의 전문 교육

과정으로 쉽게 이어갈 수 있도록 장기적인 교육 계획이 필요할 것이다. 또한 대학교에서도 이러한 장기적인 기본 교육 계획에 맞춘 눈높이 교육과정을 개설하여 학생들이 흥미를 잃지 않고 지속적으로 배울 수 있도록 준비해야 할 것이다.

이러한 교육과정을 개설하기 위해서는 소프트웨어 교육에 대한 인식전환이 필요하다. 앞서 설명했던 미국의 K-12에서는 크게 3단계로 교육과정을 나누어 각 단계마다 컴퓨터적 사고, 협동, 컴퓨터와 통신장비, 사이버 윤리에 대한 교육과정으로 나누어 교육내용을 자세하게 정의하고 있다. 이는 '프로그램을 제작하려면 프로그래밍 언어나 자료구조와 같은 전문 지식만 배우면 된다.'라는 인식을 바꾸고, 컴퓨터과학이 수학, 문학과 같은 기초 학문과 같이 중요한 학문임을 인식한 결과이다. 이와 같이 소프트웨어 교육에 대한 인식전환을 통해 기본 과정을 도출하고 개발하여 전문 인재를 양성하기 위한 노력을 아끼지 말아야 한다.

부가적으로 소프트웨어 교육을 위한 교육 플랫폼이 지속적으로 개발 · 운영되어야 한다. 미국의 Code.org, Scratch와 같이 비영리 단체 및 대학교에서 개발한 교육 플랫폼뿐만 아니라, 일본의 プログラミン(Programmin)과 같이 국가 정부기관에서 직접 개발하고 보급하여 국민이 언제나 접속하여 교육을 받을 수 있는 플랫폼이 개발되어야 할 것이다.

참고문헌

연합뉴스, 내년도 중학교 입학생부터 SW 교육 의무화한다, http://www.yonhapnews.co.kr/economy/2014/07/22/0301000000AKR20140722159600017.HTML

한국경제, KAIST마저 전산전공자 반토막… SW인재 고사직전 〈http://www.hankyung.com/news/app/newsview.php?aid=2014033013761〉

한국교육학술정보원(KERIS). 교육용 프로그래밍 언어 선택전략(2013).

プログラミン 〈http://www.mext.go.jp/programin〉

Blocky 〈https://code.google.com/p/blockly〉

Codeclub 〈https://www.codeclub.org.uk〉

Code.org. 〈http://code.org〉

Codecademy 〈http://codecademy.com〉

CSTA. K-12 Computer Science Standards(2011).

UK gov. National curriculum in England: computing programmes of study(2013).

Scratch 〈http://scratch.mit.edu〉

Small Basic 〈http://smallbasic.com〉