2024년 전기 졸업과제 착수 보고서

Python 변환을 지원하는,
한글 프로그래밍 언어 코베이직의 구현

팀명: 금정골사과
팀번호: 23
팀원: 이석원, 유수민, 김정한

목차

1. 과제 배경 및 목표

1.1 과제 배경

컴퓨터 프로그램은 현대 사회에서 필수적인 요소로 자리 잡았다. 프로그래밍 언어는 프로그램을 구현하는 도구로 여러 가지 종류가 있다. 그러나 프로그래밍 언어는 주로 영어로 되어 있어서 영어에 익숙하지 않으면 프로그래밍을 배우기 어렵다. 이는 특히 프로그래밍 교육 분야에서 문제가 되고 있다. 다음은 기존 한글 프로그래밍 언어이다.

1.1.1. 한베이식

한베이식[1]은 고려대학교에서 개발한 한글 프로그래밍 언어이다. 명령어는 초등학교 저학년이 이해할 수 있는 용어들로 구성되었고, 명령과 오류 메시지를 한글화하였다. 하지만 40여 년 전에 개발된 언어로서 현재 환경에서는 사용할 수 없는 실정이다. 그러나 초기 한글 프로그래밍 언어로 큰 가치가 있는 언어이다.

1.1.2. 스몰베이직 확장

스몰베이직[2]은 Microsoft에서 개발한 프로그래밍 언어로 초보자가 쉽게 프로그래밍할 수 있는 환경을 제공한다. 이 논문에서는 기존 Windows에서만 동작하던 것을 맥, 리눅스, 모바일에서 동작하도록 확장하였다. 하지만 영어로 프로그래밍해야 한다는 단점이 있다.

1.1.3. 새싹

새싹[3]은 부산대학교에서 개발한 한글 프로그래밍 언어로, C와 비슷한 문법을 제공한다. 초보자도 쉽게 프로그래밍할 수 있도록 한글과 비슷한 어순으로 프로그래밍할 수 있다. C를 기반으로 하였으므로 초보자가 배우기 어려울 수 있다.

1.1.4. AppleⅡ BASIC

AppleⅡ BASIC[4]은 AppleⅡ 컴퓨터에 내장되어 있는 BASIC이다. 기존 Microsoft BASIC의 기능 및 성능을 발전시킨 언어로 AppleⅡ에서 동작하는 프로그램 개발에 사용되기도 하였다. 우리는 AppleⅡ BASIC을 기반으로 하여 코베이직을 개발하였다.

1.1.5. Python

Python[5]은 귀도 반 로섬(Guido van Rossum)에 의해 1991년 만들어진 인터프리터 프로그래밍 언어이다. 간결하고 유연한 문법과 풍부한 라이브러리를 바탕으로 여러 분야에서 활용되고 있다. 특히 교육 분야에서 프로그래밍 입문을 위한 표준 언어처럼 여겨져 사용자가 꾸준히 증가하는 중이다.

1.2 과제 목표

본 연구는 한글을 사용하는 초보자가 쉽게 접근할 수 있는 프로그래밍 언어를 제공하고자 한다. 이를 위해 교육용 프로그래밍 언어인 BASIC을 기반으로 하는 한글 프로그래밍 언어 코베이직을 제안한다. 코베이직은 BASIC의 문법과 특징을 유지하면서, 한글 사용자가 편하게 프로그래밍할 수 있게 하는 것을 목표로 한다.

또한, 본 연구는 코베이직으로 프로그래밍 세계에 입문한 이후에 본격적인 영어 프로그래

밍 언어를 배울 때 더 수월할 수 있도록 코베이직으로 작성된 코드를 Python 코드로 변환하는 기능도 제안한다. Python은 유연하고 간결한 문법을 지원하고 다양한 기능을 언어 차원에서 제공하여 초보자에게 가장 인기있는 언어이다. 따라서 코베이직을 Python으로 변환하는 기능은 한글 프로그래밍 언어로 프로그램 세계에 진입한 사용자가 더 수월하게 본격적인 영어 프로그래밍 언어를 학습할 수 있도록 중간 다리를 놓아주는 것을 목표로 한다.

2. 현실적 제약 사항 및 대책

2.1 제약 사항

코베이직을 Python으로 변환하는 데 있어서 추가적인 논의가 필요한 부분은, BASIC 문법에는 존재하지만 Python에는 존재하지 않거나 동일하게 표현되지 않는 부분이다. 대표적으로 BASIC의 GOSUB 문법이 그러하다. GOSUB는 GOTO와 유사하게 특정 라인으로 이동하게 하는 명령문이지만, RETURN 문을 사용하여 GOSUB를 사용한 위치로 복귀시킬 수 있다는 점에서 GOTO와 차이가 있다. 이 문은 Python에 내장되어 있지 않기에, 코베이직을 Python으로 변환할 때에 GOSUB를 어떻게 구현할 것인지에 대한 문제가 대두된다.

2.2 대책

본 연구에서는 Python의 데코레이터와 label 딕셔너리, subroutine 스택을 이용하여 GOSUB를 구현하는 방법을 제안한다. 먼저 특정 함수의 이름을 받아 label 딕셔너리에 함수 이름과 레퍼런스를 저장하는 기능을 수행하는 label 데코레이터를 정의한다. 이후 만들어진 label 딕셔너리를 기반으로 작동하는 GOTO, GOSUB, RETURN 함수를 구현한다.

GOSUB는 함수의 이름을 인수로 전달받아 해당 함수의 이름이 현재 label 딕셔너리 멤버로 존재하는지 확인한다. 존재하는 경우 GOSUB를 호출한 상위 함수의 이름을 sys._getframe(1).f_code.co_name으로 추출하여 subroutine 스택에 저장하고 label 딕셔너리에서 인수와 같은 이름을 가진 함수를 찾아 실행한다.

RETURN 함수는 내부적으로 subroutine 스택에서 추출한 복귀할 함수의 이름을 GOTO 함수에 인수로 전달하여 이전 위치로 돌아갈 수 있도록 구현되어 있기 때문에, RETURN을 호출하면 언제든지 다시 GOSUB가 실행되었던 위치로 돌아갈 수 있다.
  이러한 방법을 통해 Python에서 GOSUB를 에뮬레이팅할 수 있다. 다만 더 효율적인 방식으로 개선할 수 있을지에 관하여 추가적인 토의가 필요하다.

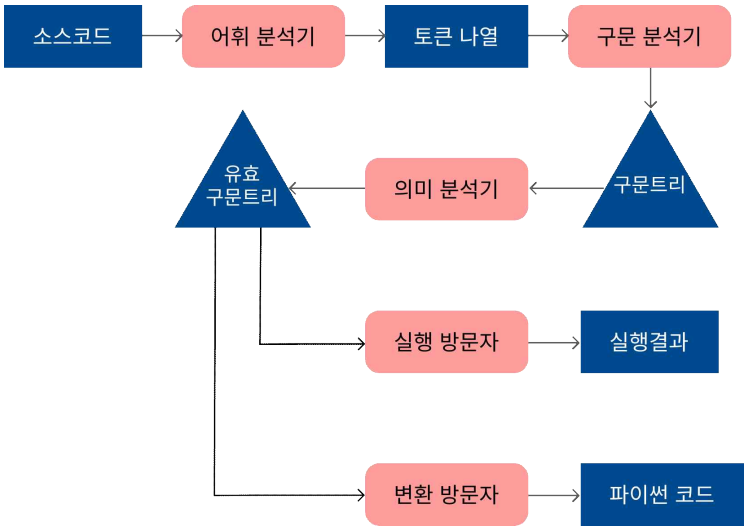
3. 연구 방향 및 내용

3.1 연구 방향

코베이직은 BASIC의 문법을 유지하되, 키워드와 변수명 등을 한글로 제공되는 것을 목적으

로 한다. 한글 키워드를 사용하면 한글을 모태로 하고 있는 사용자가 자연스럽게 프로그래밍할 수 있다. 또한, 주석을 한글로 작성할 수 있게 하여 코드의 이해와 설명을 돕는다.

코베이직의 구현도 전통적인 컴파일러 및 인터프리터 구현 과정을 따른다. 그림 1은 코베이직 실행 구조를 나타낸다.



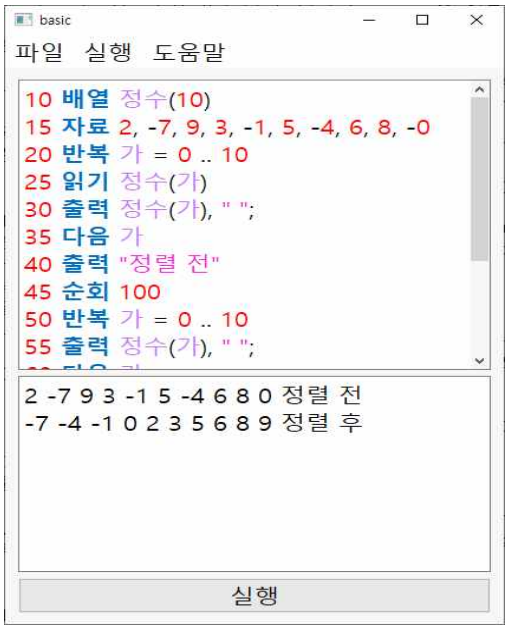
(그림 1) 코베이직 실행 구조

어휘분석 단계에서는 소스 코드를 토큰으로 분해한다. 토큰은 코드의 기본 구성 요소로, 변수, 연산자, 예약어 등과 같은 언어 요소를 나타낸다. 구문분석은 어휘분석에서 생성된 토큰들의 구조와 관계를 파악한다. 이 단계에서는 문법 규칙을 준수하는지 확인하고, 추상 구문 트리를 생성한다. 의미분석은 프로그램의 의미를 이해하고 검증하는 단계이다. 변수의 선언과 사용 등의 검사가 이루어지며, 코드의 의미적 일관성을 보장한다.

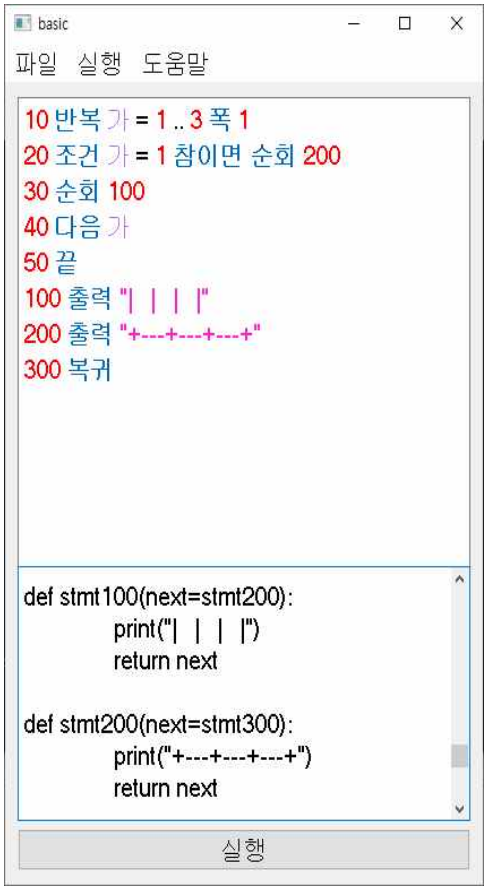
추가적으로, 코베이직 코드를 Python 코드로 변환하는 모듈을 설계하고자한다. 이 모듈은 코베이직으로 작성된 코드를 입력받아 Python 코드로 변환하는 기능을 수행한다. 이를 통해 코베이직 사용자는 학습한 내용을 실무에서도 활용할 수 있다.

3.2 연구 내용

이미 연구가 진행된 관계로 기존에 진행했던 연구의 내용을 소개하고자 한다. 코베이직의 한글 지원은 C++과 Qt의 기능을 이용하여 구현한다. 한글 변수명과 함수명을 처리하기 위해 Unicode 문자열 처리 기능을 사용하였으며, 키보드 입력과 출력을 위한 문자 인코딩 문제를 고려하여 한글 입출력을 지원하도록 개발하였다. 또한, 한글 주석 기능을 구현하여 코드의 가독성을 높이도록 하였다. 그림 2는 코베이직으로 작성한 정렬 프로그램과 그 결과를 나타낸 것이다.



(그림 2) 코베이직을 이용한 정렬 예제



(그림 3) KoBASIC을 Python 코드로 변환 예제

코베이직 UI 상단에는 코드 작성 부분이 있으며, 하단에는 출력 부분이 있다. 코드 작성은 한글로 할 수 있으며, 코드의 가독성을 위해 문법 강조 기능을 적용하였다. 코드 작성 후 실행 버튼을 누르면 실행 결과가 출력된다.

추가적으로, 코베이직 코드를 Python 코드로 변환하는 기능은 변환방문자 클래스를 통해 구현되었다. 이 클래스는 크게 세 가지 연산을 수행하는 메서드로 이루어지는데 (1)앞서 생성된 유효 구문 트리를 순회하면서 Python 코드를 생성하고 (2)생성된 코드를 알맞은 구조로 변환하여 저장한 후 (3)최종 코드를 GUI 프로그램에 출력한다. 정리하면, 코베이직으로 작성한 코드를 바로 실행하려는 사용자는 실행 방문자 모듈을 통해 (그림 2)와 같이 생성된 실행 결과를 얻을 수 있다. 한편 코베이직 코드를 Python 코드로 변환하고자 하는 사용자는 변환 방문자 모듈을 통해 (그림 3)과 같이 동일한 기능을 수행하는 Python 코드를 얻을 수 있다.

4. 개발 일정 및 역할 분담

4.1 개발 일정

개발구분	5				6				7				8			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
베이직 한글지원 구현																
Unicode 문자열 처리 기능 구현																
코베이직 구현																
코베이직 언어 함수 python 코드 변환 함수 구현																
Application 개발 및 배포																

4.2 역할 분담

이름	역할분담
이석원	- 베이직 한글 지원 구현 - 한글 변수명과 함수명을 처리를 위한 Unicode 문자열 처리 기능 구현 - 키보드 입력과 출력을 위한 문자 인코딩 문제를 고려하여 한글 입출력 개발
유수민	- 코베이직 언어 함수 python 코드 변환 함수 구현 - 모델 테스트 및 분석
김정한	- 관련 논문 분석 및 작성 - Application 구현

5. 참고 문헌

[1] 황종선, 원유헌, 곽호영, "한글 베이식 언어의 설계 및 구현", 정보과학회논문지, Vol. 12, No. 1, pp. 52-59, 1985.

[2] 김가영, 정승완, 최광훈, "스몰베이직 프로그램 해석기 설계 및 구현에 관한 연구", 한국정보과학회 학술발표논문집, pp. 1827-1829, 2016.

[3] 천준석, 강도훈, 김건우, 우균, "간결한 한글 프로그래밍 언어 '새싹'", 정보과학회논문지, Vol. 42, No. 4, pp. 496-503, 2015.

[4] Helmers C, "An Apple to Byte," BYTE, Vol. 3, No. 3, pp. 18-46, 1978.

[5] Severance, Charles. "Guido van rossum: The early years of python," Computer, Vol. 48, No. 2, pp. 7-9, 2015.