컴파일러 Project 5-3 – 결과(설계)보고서 –

팀장 : 김석호

조원 : 민윤기, 권현송

## 조원(개인, 팀)

|  |  |  |  |
| --- | --- | --- | --- |
| 학번 | 이름 | 맡은 역할 | 비고 |
| 201502025 | 김석호 | GUI 구현, BytecodeGenListener 관련 추가 구현 |  |
| 201302267 | 민윤기 | PythonGenListener 관련 구현 및 변수명, 타입 에러 체크 기능 구현 |  |
| 201701198 | 권현송 | JavaGenListener 관련 구현 및 변수명, 타입 에러 체크 기능 구현 |  |

Github Repo 주소 – <https://github.com/SeokBA/Compiler_Term>

(현재 private이므로, 추후에 public으로 바꿔야 함)

## 주제 - 지금까지 배운 컴파일러 이론과 실습 지식들을 활용하여 MiniC 파일을 Bytecode 뿐만 아니라 다른 언어로의 변환을 시도하고자 Java Swing을 이용한 GUI 바탕의 MiniC 파일을 입력으로 하고 Antlr 라이브러리를 이용하여 Python, Java, Bytecode로 변환해 파일 형태로 출력하는 프로그램을 제작하는 것을 주제로 선정하였습니다.

## 구현 코드 및 방법 설명(이미지 첨부 가능, 함수 단위로 설명 가능)

1. GUI (TranslatorGUI.java)  
   사용자가 언어를 변환할 경우 사용이 편리하도록 하기 위해서 TranslatorGUI를 구현하였습니다. GUI는 Java GUI Library들을 이용하여 MiniC파일(.c)을 선택해 가져와서 변환할 언어를 선택한 뒤 선택한 저장 위치에 변환된 언어를 저장하는 방식으로 구현했습니다. 또한, walker 실행 중 또는 변환 중 오류가 탐지될 경우 오류 메시지를 GUI 내부의 console에 띄우도록 했습니다.
2. BytecodeGenListener.java 외 bytecode package 내부 코드  
   bytecode는 MiniC.g4에 맞춰 전번 기초구현에 더하여 해당 사항들을 추가하였습니다.  
     
   1) 배열 사용 가능  
   - SymbolTable에 VarInfo를 extends하는 class ArrInfo와 기타 배열 사용에 필요한 메소드들을 추가하여 배열을 저장하는 객체를 구현해 Listener에서 이용하도록 제작하였습니다.  
     
   2) 전역변수 사용 가능 (구현해야함)   
   -  
     
   3) tap 정렬 기능   
   - exitFun\_decl에서 블록안의 개행들을 전부 tap을 추가해주는 기능을 구현하였습니다.  
     
   4) 코드 최적화 - '\_' 명령 변환   
   - iload와 같은 명령어는 0~3까지는 \_명령을 사용하므로, 해당 명령어에 맞춰 변환하는 기능을 구현하였습니다.  
     
   5) 코드 최적화 - not (javap console과 최대한 비슷하게, 구현해야 함)   
   -
3. PythonGenListener.java 외 python package 내부 코드
4. 기본 기능 구현
   * MiniC로 작성된 프로그램을 각 문법에 알맞게 Python으로 변경하는 과정에서

조건문에 사용되는 괄호, 함수 선언에 사용되는 중괄호, 변수 혹은 함수 선언에서

사용되는 타입 spec, 세미콜론 등을 제거 후 indent를 맞추어 정렬하도록

작성하였습니다.

1. 배열 및 초기화되지 않은 변수 선언 기능 구현
   * 배열의 경우 Python에서는 리스트로 표현되는데, 이 리스트의 경우는 일반적인

배열의 개념처럼 초기 크기를 설정할 수 있는 방법이 제한적이기 때문에 [0] \* n과

같은 방식으로 작성하였습니다. 또한 초기화 되지 않은 변수의 경우에는

Python에서 문법 오류를 발생시키기 때문에 MiniC에서 int형 변수 밖에 존재하지

않으므로 0으로 초기화 하여 출력하도록 작성하였습니다.

1. 선언되지 않은 변수 및 함수 사용과 중복 선언 오류 검출
   * 이 기능을 구현하기 위해 기존 Bytecode 과제에서 사용하던 Symbol table의

일부를 차용하였습니다. 전역으로 선언된 에러가 발생하면 GUI 클래스에서 에러 출력 여부를

담당하는 변수를 true로 세팅하고 각 문법을 순회하면서 발생한 에러 메시지를 Dump 빌더에

묶어줍니다. 전역 변수와 지역 변수, 그리고 함수 선언의 enter 메소드를 사용하여 기 선언 여부

를 Symbol table에 들어 있는지의 여부와 전역 혹은 지역변수라면 함수 table에 들어있는지의

여부로 확인하였고, 주로 expr의 exit 메소드에서 변수 혹은 함수가 호출될 때 Symbol table에

존재하지 않는지의 여부로 확인하여 해당 기능을 구현하였습니다. 이렇게 하여 에러가 존재하는

프로그램이라면 GUI에서는 파일로 출력하는 것이 아닌 콘솔에 해당 에러 메시지를 출력하는

형태로 동작하게 됩니다.

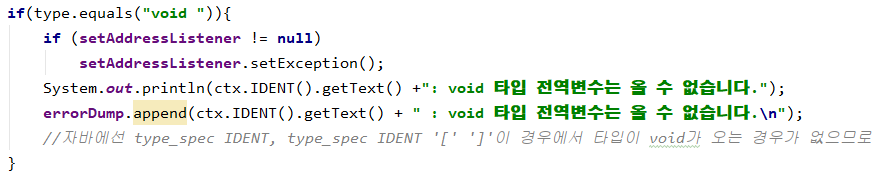
1. JavaGenListener.java 외 java package 내부 코드  
     
   1)기본 기능 구현

- 우선 miniC의 문법이 지원하는 기본적인 기능들이 JAVA언어로 잘 변형되게끔 작성하였습니다. 우선 기본적으로 TestJava라는 클래스(public class TestJava)를 만들어 안에 main함수와 다른 함수들이 작성되도록 하였고, main()함수는 public static void main(String[] args)꼴로 기본 자바 메인 함수 꼴로 바꿔 출력하였습니다. 또한 기본적으로 지원되는 출력 함수는 자바의 System.out.println()으로 변형되도록 하였습니다. 이를 통해 while문, 중첩문, if문, 변수 선언, 함수 호출 등 기본 코드를 자바 형식 그대로 변환하였습니다. 추가로 JavaSymbolTable클래스와 JavaGenListenerHelper클래스를 만들어 추가 기능들을 구현하였고, 줄 맞춤도 자동으로 하여 가독성을 올렸습니다.

2) void변수 유무 확인(지역 변수, 매개 변수, 전역 변수)

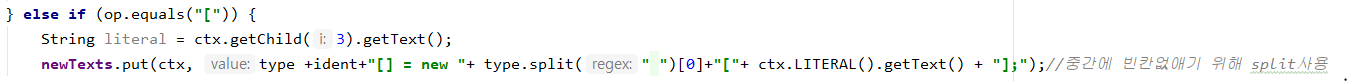
-자바에서는 변수 타입이 void가 될 수 없으므로 void 변수가 나오는 경우 예외처리를 하

였습니다. 매개변수나 지역 변수, 전역 변수가 나올 때 타입을 검사하여 void타입인 경우라면 해당 변수 이름과 에러 문을 출력하였습니다.

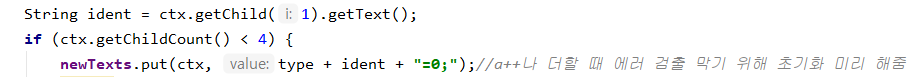


3) 배열 및 초기화되지 않은 변수 선언 기능 구현

- 자바에서는 배열을 선언 할 때, 배열 객체를 생성해 주어야 합니다. 이에 따라 배열을 선언하는 부분에서는 ‘int k[] = new int[3];’이런 식으로 바꿔줘야 합니다. 이에 따라 배열을 선언하는 부분이 나오면 객체를 생성해주도록 추가해주었습니다.

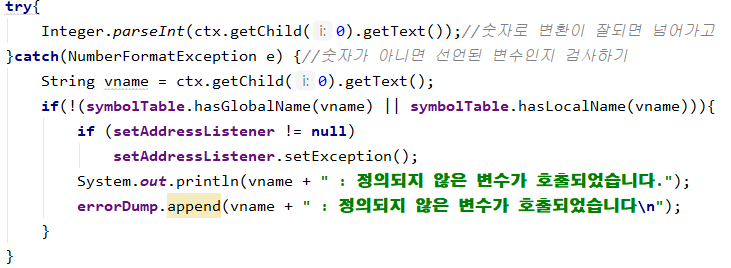


또한 초기화되지 않은 변수가 연산에 이용될 경우 에러가 나므로 선언 시 따로 초기화 기능이 없다면 자동으로 초기화해 주도록 하였습니다. 예를 들어 ‘int a’를 선언하고 ‘++a’나 ’a+1’을 실행할 경우 a의 초기값이 없어서 에러가 나므로 선언시에 0으로 기본 값을 자동 초기화 주어 실행 시 에러가 나지 않도록 하였습니다.



1. 변수 또는 함수 호출 시 선언 유무 확인 기능

Minic를 실행해본 결과 선언된 변수나 함수가 아닌 것들을 호출하여도 문제없이 실행된다는 것을 찾았습니다. 그래서 심볼테이블을 이용하여 변수나 함수를 선언 시 심볼테이블에 전역 변수, 지역변 수, 함수 세 개로 구분해 저장하였습니다. 그 후 호출을하게 되면 심볼 테이블을 검사해 이미 선언된 함수가 맞는지 확인하였습니다. 검사 결과 심볼 테이블에 존재하지 않으면 에러문을 출력하였습니다. 추가로 a=1; 1>a; a>b;이런 모든 경우를 고려하여 해당 expr을 검사하여 숫자가 아니면 변수이므로 심볼테이블에 존재하는지 검사하였습니다. 숫자 유무를 검사한 방법은 문자열은 parseInt함수에 넣어 에러 유무를 확인하여 에러가 난 경우 처리해주었습니다.



이때 a=i/2;인 경우 i/2자체를 변수로 받아들이는 에러가 있어서 자식 수를 세어 하나인 경우만 처리해 주도록 하였습니다.



## 결과 요약

위의 주제를 구현하여 다음과 같이 구현하였습니다.

자바: minic의 기본적인 구현들은 모두 자바로 바꿀 수 있도록 하였고, 추가로 변수 및 배열 초기화 기능, void타입 변수 검사, 변수 및 함수 호출 시 선언 유무 확인 기능들을 더 작업하여 실제 자바 코드가 실행 가능하도록 하였습니다.

## 장, 단점 분석 및 아쉬운 점

이번 과제를 수행하면서 지금까지 배웠던 내용들을 실제로 적용할 수 있는 기회를 갖게 되어 좋았습니다. 조금 아쉬웠던 점은 구현하기에 시간이 부족했던 것 같습니다.

- 기타 내용 추가가능

- 팀장 연락처 (이메일)

- 김석호 : cpswo123@naver.com