컴파일러개론 1주차 실습

IMLG TO C 컴파일러 만들기

2024. 09. 06.

TA: 박정필

✉: 202350941@o.cnu.ac.kr

공지, 질문 방법

- 강의자료, 동영상
  - 사이버캠퍼스 업로드 예정
- 공지
  - 카카오톡 오픈채팅, 사이버캠퍼스
    - https://open.kakao.com/o/gkfF7JMg
      - 오픈프로필 사용 가능
      - 참여코드: 2024cp01
- 질문
  - 수업시간
  - 카카오톡 오픈채팅에 질문
  - 이메일
    - 이메일 제목은 "[컴파일러개론][분반] ...", 제목 반드시 준수
    - 교수님 : eschough@cnu.ac.kr
    - TA : 202350941@o.cnu.ac.kr

목차

- 실습 환경 구축
  - Intellij 설치
  - Java 코딩 환경 구축
- 1주차 과제 소개

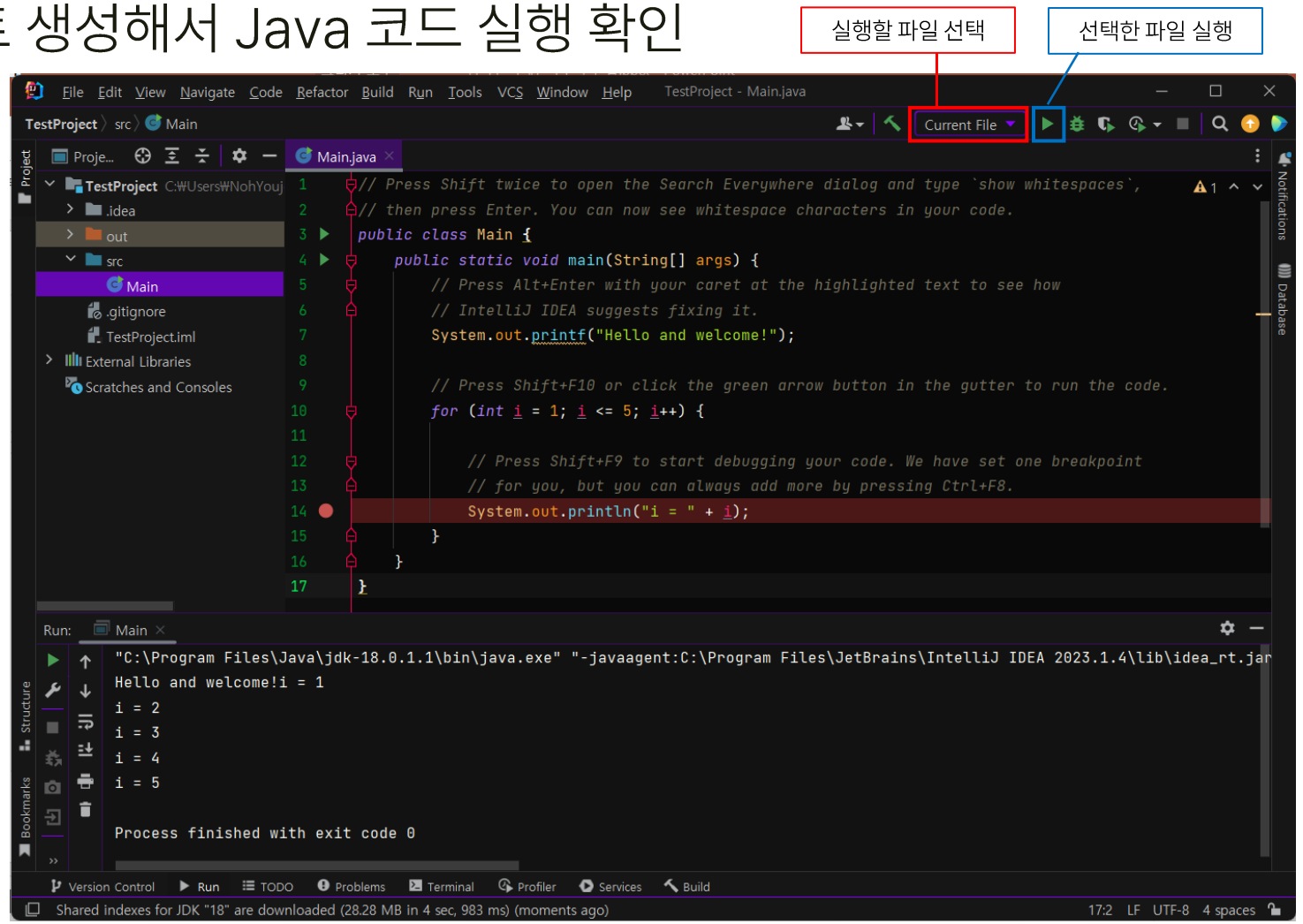
Intellij 설치 및 Java 코딩 환경 구축 (1/2)

- 인터넷을 보고 Intellij 설치
  - https://www.jetbrains.com/ko-kr/idea/download/other.html 에서 운영체제에 맞춰 community edition 다운로드
  - [IntelliJ] Intellij 설치방법 (tistory.com)
    - Mac OS: https://how-can-i.tistory.com/127
  - 이외 다른 포스팅 참고해도 상관 없음

- 프로젝트 생성해서 Java 코드 실행 확인
  - Intellij에서 실습 세팅 진행

Intellij 설치 및 Java 코딩 환경 구축 (2/2)

- 프로젝트 생성해서 Java 코드 실행 확인



1주차 과제 – imolang 컴파일러

- 난해한 프로그래밍 언어의 컴파일러 구현
- Java로 구현
- 입력받은 언어를 C언어로 컴파일
- 정규식 이용해 구현

imolang

- :,], (,), }, ^로 이루어진 난해한 프로그래밍 언어
- 다음으로 구성됨
  - 총 4개의 기능
    - 읽기(scanf) 기능
    - 쓰기(출력, printf) 기능
    - 지정(변수 assignment, a = b) 기능
    - 계산 (덧셈, 뺄셈) 기능
  - 최대 3개 변수까지 지정 가능
  - 0 이상의 정수, 문자열을 다룰 수 있음
  - 한 줄에 한 명령어

imolang

- 문법
  - 변수: ^, ^^, ^^^ 세 가지 까지의 변수를 가짐
  - 읽기 (scanf)
    - :) 0 이상의 정수를 읽어 변수에 저장
      e.g. :) ^ ➔ 0 이상의 정수를 입력 받아 변수 ^에 저장
    - :):] 문자열을 읽어 변수에 저장
      e.g. :):] ^^ ➔ 문자열을 읽어 변수 ^^에 저장

imolang

- 문법
  - 쓰기 (printf)
    - :)) 숫자 출력

      e.g.1. :)) 5 ➔ 5 출력

      e.g.2. :)) ^ ➔ 변수 ^에 저장된 값 출력

    - :)):] 문자열 출력

      e.g.1. :)):] helloworld ➔ helloworld 출력

      e.g.2. :)):] ^ ➔ 변수 ^에 저장된 문자열 출력

    - :)):]] ➔ 줄바꿈 출력

imolang

- 문법
  - 지정
    - :() 0 이상 정수를 변수에 저장

      e.g. :() ^ 5 ➔ 변수 ^에 5를 저장

    - :():] 문자열을 변수에 저장

      e.g. :():] ^ hello ➔ 변수 ^에 hello를 저장

imolang

- 문법
  - 계산
    - :} 첫째 인자에 둘째 인자를 덧셈

      e.g. :} ^ ^^ ➔ 변수 ^의 값과 변수 ^^의 값을 더함

    - :}} 첫째 인자에서 둘째 인자를 뺄셈

      e.g. :}} ^ 5 ➔ 변수 ^의 값에서 5를 뺌

    - 한 줄에 계산 연산자는 하나만 허용(e.g. :} ^ ^^ :} ^^^ 등 불가)
  - 계산 후 지정
    - :() (변수) (계산 명령어)

      e.g. :() ^^ :} ^^ ^ ➔ 두 변수 ^^과 ^의 합을 변수 ^^에 지정

imolang

- 예시 입출력
  - Helloworld\n 출력

```
:)):] Helloworld
:)):]]
```

```
#include <stdio.h>
int main(){
int a;
int b;
int c;
char* as;
char* bs;
char* cs;
printf("Helloworld");
printf("\n");
}
```

imolang은 최대 3개 변수 가지므로
사용가능한 변수를 미리 선언
(변수를 미리 선언할지 imolang에서
실제 사용(지정) 시 선언할 지는 본인 자유)

입력:
출력: Helloworld

  - 입력된 문자열 출력

```
:):] ^ //scanf %s a
:)):] ^  //printf %s a
```

```
#include <stdio.h>
int main(){
int a;
int b;
int c;
char* as;
char* bs;
char* cs;
scanf("%s", as);
printf("%s", as);
}
```

입력: hello
출력: hello

imolang

- 예시 입출력
  - 계산기(덧셈, 뺄셈)

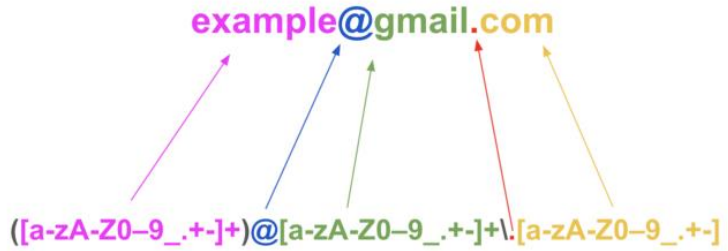
```
:) ^                    //scanf %d a
:) ^^                   //scanf %d b
:() ^^^ :} ^ ^^         //c = a+b
:)) ^^^                 //printf %d c
:() ^^^ :}} ^ ^^        //c = a - b
:)) ^^^                 //printf %d c
```

```
#include <stdio.h>
int main(){
int a;
int b;
int c;
char* as;
char* bs;
char* cs;
scanf("%d", &a);
scanf("%d", &b);
c = a + b;
printf("%d", c);
c = a - b;
printf("%d", c);
}
```

```
입력: 2
      1
출력: 3
      1
```

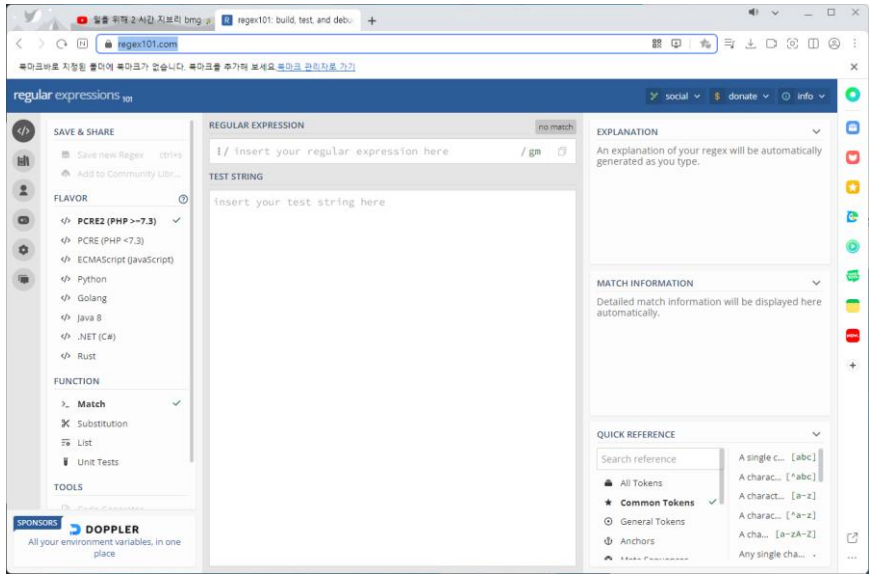
Regular Expression (1/3)

- Regular Expression (정규표현식)
  - 특정한 규칙을 가진 문자열의 집합을 표현하는 형식 언어
  - 문자열에 대한 조건을 표현하는 방법
  - POSIX, PCRE 등의 다양한 표현 방식이 존재



이메일 정규식 패턴매칭

- JAVA Regular Expr Online
  - https://regex101.com/
  - https://regexone.com/



Regular Expression (2/3)

- JAVA 구현
  - https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html
  - https://docs.oracle.com/javase/8/docs/api/java/util/regex/Matcher.html
- Pattern
- Matcher

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3

java.util.regex

Class Pattern

java.lang.Object
  java.util.regex.Pattern

All Implemented Interfaces:

Serializable

public final class **Pattern**
extends Object
implements Serializable

A compiled representation of a regular expression.

A regular expression, specified as a string, must first be compiled into an instance of this cl
that can match arbitrary character sequences against the regular expression. All of the stat
matchers can share the same pattern.

A typical invocation sequence is thus

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

A matches method is defined by this class as a convenience for when a regular expression is
input sequence against it in a single invocation. The statement

```
boolean b = Pattern.matches("a*b", "aaaaab");
```

is equivalent to the three statements above, though for repeated matches it is less efficient

Instances of this class are immutable and are safe for use by multiple concurrent threads. I

Summary of regular-expression constructs

compact1, compact2, compact3

java.util.regex

Class Matcher

java.lang.Object
  java.util.regex.Matcher

All Implemented Interfaces:

MatchResult

public final class **Matcher**
extends Object
implements MatchResult

An engine that performs match operations on a character sequence by interpreting a Pattern.

A matcher is created from a pattern by invoking the pattern's matcher method. Once created, a matcher can be used to perform three different kinds of match operations:

- The matches method attempts to match the entire input sequence against the pattern.
- The lookingAt method attempts to match the input sequence, starting at the beginning, against the pattern.
- The find method scans the input sequence looking for the next subsequence that matches the pattern.

Regular Expression (3/3)

- JAVA 구현
  - Pattern, Matcher API 활용
  - 정규식 이용해 imolang 명령어 매칭

```java
import java.util.regex.Pattern;

public class RegexExample {
        public static void main(String[] args)  {

            String pattern = "^[0-9]*$"; //숫자만
            String val = "123456789"; //대상문자열

            boolean regex = Pattern.matches(pattern, val);
            System.out.println(regex);
    }
}
```

```java
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexExample {
        public static void main(String[] args)  {
            Pattern pattern = Pattern.compile("^[a-zA-Z]*$"); //영문자만
            String val = "abcdef"; //대상문자열

            Matcher matcher = pattern.matcher(val);
            System.out.println(matcher.find());
        }
}
```

Regular Expression - 예시

- Regex 예시
  - 휴대전화: "\d{3}-\d{3,4}-\d{4}"
  - imolang 덧셈연산: ":\} \^{0,3} \^{0,3}"

- 중요
  - (,), }, ^ 등 괄호 및 특수 기호들은 regex에서 문자가 아닌 **집합 기호**로 인식(메타 문자)
  - 예시로, 정규식 ":() ^"은 ()는 빈 집합, ^는 문자의 시작으로 인식되어 정상 작동하지않음
  - 해당 문자를 정규식으로 매칭하고 싶다면 백슬래시(\)를 앞에 붙여 사용해야 함
    - 언어마다 차이가 있으나 Java는 백슬래시를 두 번 붙여야 함

    e.g. ":\\(\\) \\^"

1주차 과제: imolang 컴파일러

- Imolang으로 작성된 .imlg 소스 파일을 C 코드로 바꾸어 .c 파일에 출력하는 imolang 컴파일러를 제작하시오.
  - 입력되는 모든 프로그램은 오류가 없다고 가정 (즉, 오류 처리는 하지 않아도 됨)
  - .imlg 코드에는 빈 줄이 없다고 가정
  - 코드는 Java로 작성
    - Package 사용 x
    - 파일 여러 개로 구현해도 됨
    - Main 함수가 있는 파일 이름은 반드시 Main.java로, 나머지 파일들은 이름 상관 없음
  - Input, output 파일 이름은 test
    - Input file은 test.imlg
    - Output file은 test.c
    - 절대 주소 사용 x, 상대 주소 사용하여 현재 디렉토리의 .imlg 파일을 읽어 현재 디렉토리에 .c 파일 만들도록 함

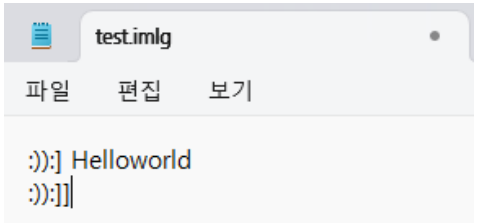
절대 경로: `C:\Users\NohYoujeong\Downloads\test.imlg`
상대 경로: `.\test.imlg`

1주차 과제: imolang 컴파일러

- 추가 안내 사항
  - 너무 어렵게 생각하지 말고 되는 데까지 구현하여 제출
  - 변수는 최대 3개 까지로 제한 (^, ^^, ^^^)
  - 인덴트 구현 안해도 됨. Testcase 기반 채점
    - C 코드가 컴파일/실행했을 때, imolang 코드와 같은 동작을 하면 맞다고 채점할 예정
  - 정규식 이용해 구현하면 됨 (파서 등 구현 필요 X)
- 제출 방식
  - Java 소스 파일을 학번.zip으로 압축해서 사이버캠퍼스 과제 제출물로 제출
  - Java 코드에 주석 잘 달기
- 마감
  - 2024년 9월 13일 금요일 23시59분 (기한 엄수)

imolang 컴파일러 동작 예시

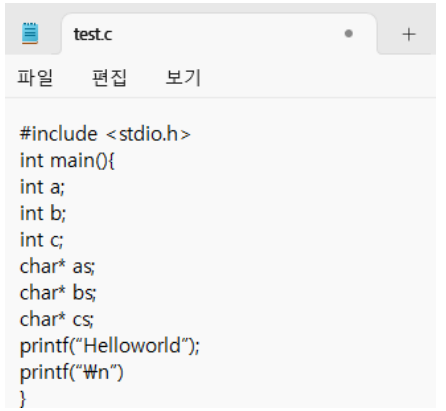
test.imlg



JAVA imolang 컴파일러

Main.java에서 같은 폴더에 있는 imlg
파일 읽어 컴파일
(상대경로 사용)

test.c



Test.imlg와 같은 의미 가지는
c 파일 생성.
c 파일 동작 기준으로 채점

입출력 파일 이름은 test.imlg / test.c로 고정!
test.c 파일이 컴파일 되지 않는다면 0점. 반드시 컴파일 테스트를 해볼 것

그 외 주의사항

- 코드 카피 금지
  - 코드 유사도 검사기 사용
  - 카피 적발 시 해당 과제 0점 처리

- GPT 사용 금지
  - GPT 사용으로 코드카피 적발 시 마찬가지로 해당 과제 점수 0점

- 컴파일 확인
  - test.c 파일이 컴파일 되지 않는다면 0점. 반드시 컴파일 테스트를 해볼 것

	이론/		실습/과제	
	퀴즈/복습	동영상 강의	실습	과제
1	9/3 1. Introduction	9/3~9/10 2. Lexical Analysis	9/6 IntelliJ, Regex	토이 언어 기한 9/13
2	9/10 2. Lexical Analysis	9/10~9/24 3.구문분석. 4.Topdown Parsing I	9/13 휴강 9/20 ANTLR	Rule 인식 기한 9/27
3	9/24 3. 구문분석, 4.Topdown Parsing I (예비군훈련으로 퀴즈 없음)	9/24~10/1 4.Topdown Parsing II	9/24, 9/27 Quiz 대신 손으로 문제풀기 과제 기한 10/4	
4	9/27(금) 4.Topdown Parsing II (10/1이론 보강으로 퀴즈 없음)	10/1~10/8 5. Bottomup Parsing I	10/4 Recursive Decent Parser	Recursive Decent Parser 기한10/11
5	10/8 5. Bottomup Parsing I	10/8~10/15 5. Bottomup Parsing II	10/11, 10/18 리스너 설명, 실습	Pretty print I 기한10/18
6	10/15 5. Bottomup Parsing II	없음		Pretty Print II 기한 11/1
7	10/22 중간고사	10/22~10/29 6. SDD & AST	10/25 휴강	
8	10/29 6. SDD & AST	10/29~11/5 7. IR Basics	11/1, 11/8 JVM, Jasmin, Javap	JVM 손코딩 기한 11/8
9	11/5 7. IR Basics	11/5~11/12 8. IR Translation I		JVM – jasmin 기한 11/15
10	11/12 8. IR Translation I	11/12~11/19 8. IR Translation II	11/15 AST 빌드	AST 빌드 기한 11/22
11	11/19 8. IR Translation II	11/19~11/26 9. Semantic Analysis	11/22, 29 AST에서 Java byte 코드 변환	AST -> Javabytecode
12	11/26 9. Semantic Analysis	11/26~12/3 10. Machine Dependent Analysis		
13	12/3 10. Machine Dependent Analysis	12/3~12/10 11 Analysis & Optimization	12/6 텀프로젝트 소개	텀프로젝트 기한 12/19(목)
14	12/10 11 Analysis & Optimization			
15	12/17 기말고사			12/20 Extra Lecture

퀴즈

- 점수가 거의 들어가지 않는다. 5% 미만
- 매우 쉽게 출제된다.
- 소홀히 할 경우 크게 변별이 될 가능성이 있다.
- 가장 낮은 퀴즈 점수는 1개는 버린다.
- 9/24 (예비군훈련일)과 9/27(10/1일 보강일)의 퀴즈는 손으로 풀어 내는 과제로 대체한다.(추후 공지)

기타

이론 시간에 모각공용(모두 각자 공부용) 이어폰 반드시 가져와주세요