프로그래밍 언어를 만들자

#0 : 시작에 앞서

사람들은 고수준의 아이디어를 펴내길 원하고, 기계들은 작은 명령 집합을 가지고 있습니다. 따라서, 두 집단 사이 소통에는 번역이 필요하죠. 이때, 프로그래밍 언어가 사람과 기계 사이를 이어주는 징검다리 역할을 합니다. 사람이 쓴 고수준 코드를 기계가 알아듣기 쉽게 번역해주죠.

여기서, 번역하기 전의 고수준 코드를 원본 코드Source Code라고 하며, 번역이 완료되어 기계라는 목표에 더 가까워진 저수준 코드를 목표 코드Target Code라고 부릅니다.

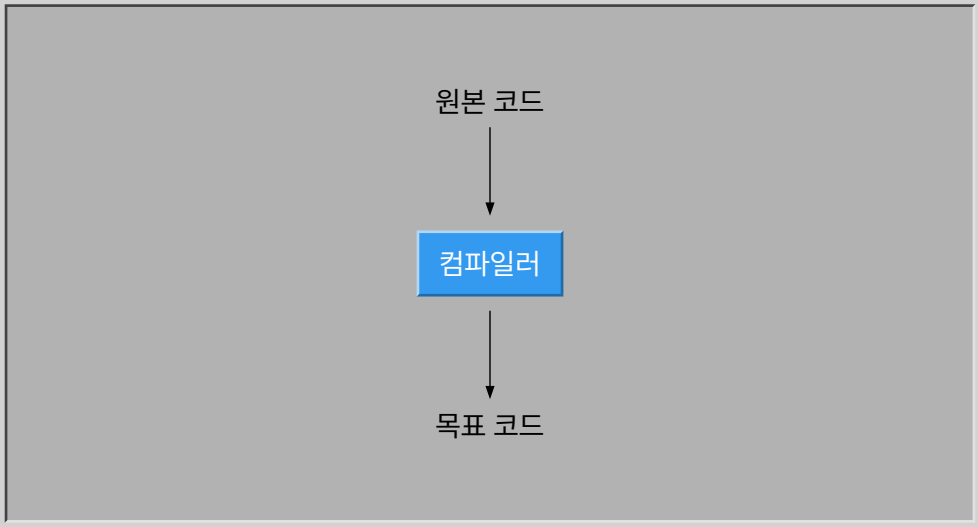

Figure 1: 단순한 컴파일러

연습 문제

간단한 컴파일러를 만들어봅시다. 아래 스펙을 구현해보세요. 모든 명령은 무조건 줄 단위로(다시 말해, `\n` 문자로) 나뉘어 있으며, 올바르지 않은 프로그램 입력은 주어지지 않는다고 가정해도 좋습니다. 목표 코드는 어떤 언어이든 상관 없으나, 실행 가능한 프로그래밍 언어여야 합니다.

A=B `/([a-zA-Z]+)\s*=\s*([0-9]+)/`	변수 이름 **A**에 정수형 값**B**를 대입한다 **B**의 범위는 ± 1000을 넘지 않는다
putchar **A** `/putchar\s*([a-zA-Z]+)/`	변수 **A**에 담긴 값에 해당하는 ASCII 문자를 출력한다

이 언어로 Hello를 출력하는 예제입니다.

```
H=72
e=101
l=108
o=111
putchar H
putchar e
putchar l
putchar l
putchar o
```

다음 페이지에 JavaScript/Python으로 작성한 예시 답안이 있습니다. 편의상 eval로 즉시 실행할 수 있게, 컴파일 대상 언어를 그 자신으로 해두었습니다.

예시 답안

JavaScript

```javascript
function compile(source) {
  let output = '';
  for (const line of source.split("\n")) {
    let match;
    if (match = /([a-zA-Z]+)\s*=\s*([0-9]+)/.exec(line)) {
      [, name, value] = match;
      output += `${name} = ${value};\n`;
    } else if (match = /putchar\s*([a-zA-Z]+)/.exec(line)) {
      [, name] = match;
      output += `process.stdout.write(String.fromCharCode(${name}));\n`
    } else {
      throw new Error(`Malformed Input: ${line}`);
    }
  }
  return output;
}
```

Python

```python
import re

def compile(source):
  output = ""
  for line in source.split("\n"):
    if match := re.match(r"([a-zA-Z]+)\s*=\s*([0-9]+)", line):
      name, value = match.groups()
      output += f"{name} = {value}\n"
    elif match := re.match(r"putchar\s*([a-zA-Z]+)", line):
      name = match.group(1)
      output += f"print(chr({name}), end='')\n"
    else:
      raise Exception(f"Malformed Input: {line}")
  return output
```