Assignment 04: String

Date: 2024. 10. 3
Student ID: 201924451
Name: 김태훈

1. str.c 설명

(1) 정수 추출

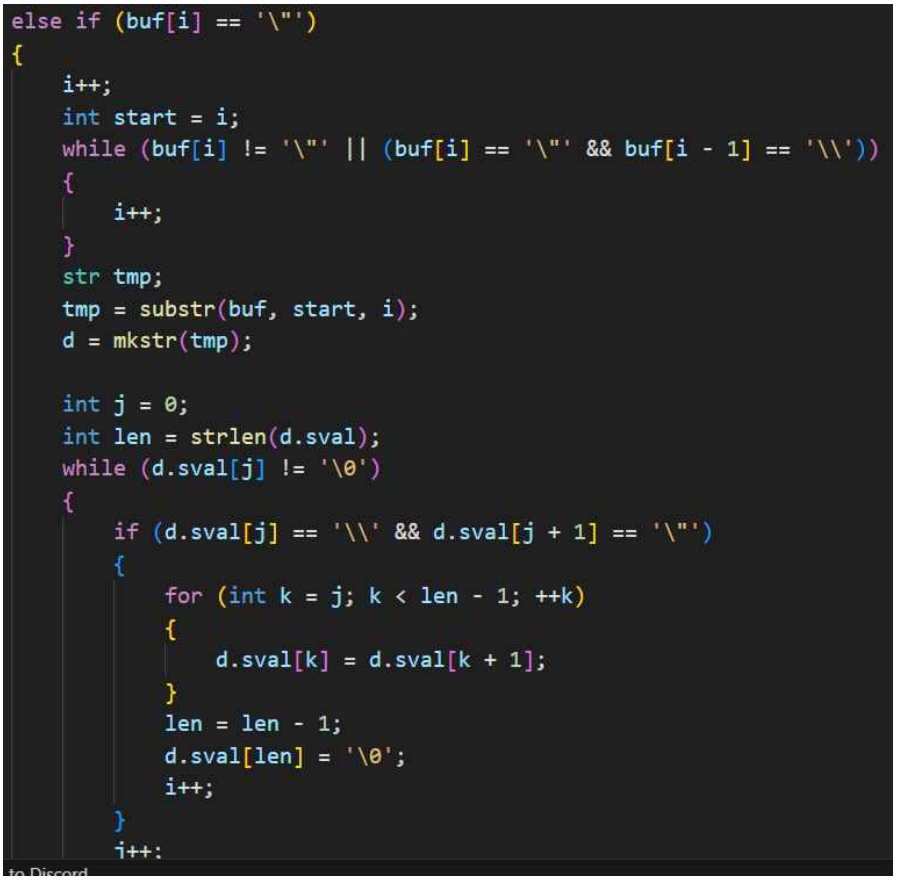
```
int main()
{
    Data d;
    while (custom_gets() > 1)
    {
        buf[strcspn(buf, "\n")] = '\0';
        int i = 0;
        while (i < strlen(buf))
        {
            if (buf[i] == ' ')
            {
                i++;
            }
            else if (isDigit(buf[i]))
            {
                int start = i;
                while (isDigit(buf[i]))
                {
                    i++;
                }
                str tmp;
                tmp = substr(buf, start, i);
                d = mkint(atoi(tmp));
                datarray[sz++] = d;
                free(tmp);
            }
```

문자열을 왼쪽에서 오른쪽으로 보면서 숫자가 나타나면, 숫자가 아닐 때 까지 오른쪽으로 이동하여 처음 숫자가 나타났을 때부터 마지막으로 숫자가 나타난 곳을 잘라 해당 부분을 숫자로 바꾸어 저장한다.

이를 BNF로 표현하면 다음과 같다[3]

<integer> ::= <digit>|<integer><digit>
<digit>::=0|1|2|3|4|5|6|7|8|9

(2) 문자열



```
else if (buf[i] == '\"')
{
    i++;
    int start = i;
    while (buf[i] != '\"' || (buf[i] == '\"' && buf[i - 1] == '\\'))
    {
        i++;
    }
    str tmp;
    tmp = substr(buf, start, i);
    d = mkstr(tmp);

    int j = 0;
    int len = strlen(d.sval);
    while (d.sval[j] != '\0')
    {
        if (d.sval[j] == '\\' && d.sval[j + 1] == '\"')
        {
            for (int k = j; k < len - 1; ++k)
            {
                d.sval[k] = d.sval[k + 1];
            }
            len = len - 1;
            d.sval[len] = '\0';
            i++;
        }
        i++;
```

문자열은 따옴표(“)가 나타냈을 때부터 다시 따옴표가 나타날 때까지를 잘라 저장한다. 이때 문자열을 구분하는 따옴표와 문자열 안에 있는 따옴표를 구분하기 위해 따옴표를 발견한 후, 그 이전 문자가 ‘\’이면 이 따옴표를 건너뛰도록 설정하였다. 이를 BNF로 표현하면 다음과 같다.[3][4]

<string> ::=<quote><string_content><quote>
<string_content>::=<escape_quote><string_content>
|<letter><string_content>
|ε
<escape_quote> ::= \”
<quote< ::= “
<letter> ::= A|B|C|D...|Z|a|b|c|...|z|0|1|2...|9

(3) 식별자

```
else
{
    int start = i;
    while (buf[i] != '\0' && buf[i]!=' ')
    {
        i++;
    }

    str tmp;
    tmp = substr(buf, start, i);
    d = mksym(tmp);
    datarray[sz++] = d;
    free(tmp);
}
```

식별자는 쌍다옴표를 제외한 문자열을 만난 후, null문자나 공백을 만날 때까지를 잘라 부분문자열을 만든 후 식별자로 저장하였다. 이를 BNF로 표현하면 다음과 같다.[3]

<identifier>::=<letter>|<identifier><letter>|<identifier><digit>
<letter>::=A|B|C|D...|Z|a|b|c...|z
<digit>::=0|1|2|3|4|5|6|7|8|9

2. 생성자를 사용하는 방법의 장점과 단점[1]

생성자를 사용하는 방법의 장점은 재사용성, 생산성 향상, 캡슐화가 있다. 재사용성은 상속을 통해 코드를 재사용할 수 있는 것을 말한다. 따라서 공통된 부분은 한번만 작성할 수 있다.
생산성 향상은 객체의 템플릿을 클래스 등으로 정의하고, 생성자를 통해 독립된 객체를 여러개 생성할 수 있다는 것이다.
캡슐화는 객체의 데이터와 코드의 형태를 외부에서 알 수 없게 하고, 수정할 수 없게 하여 안전한 프로그램을 만드는데 도움을 주는 것이다.

생성자를 사용하는 방법의 단점은 코드의 복잡성 증가이다. 상속, 다형성, 동적 바인딩이 포함되면, 코드의 복잡성과 크기를 증가시킬 수 있는 것이다. 또한 더 많은 메모리를 소모할 수 있는데, 이는 객체마다 데이터와 메소드를 저장해야하기 때문이다.

3. 정수, 문자열, 식별자를 Haskell 타입으로 정의하는 방법[2]

(1) 정수

정수는 prelude command에서는 다음과 같이 정의할 수 있다.

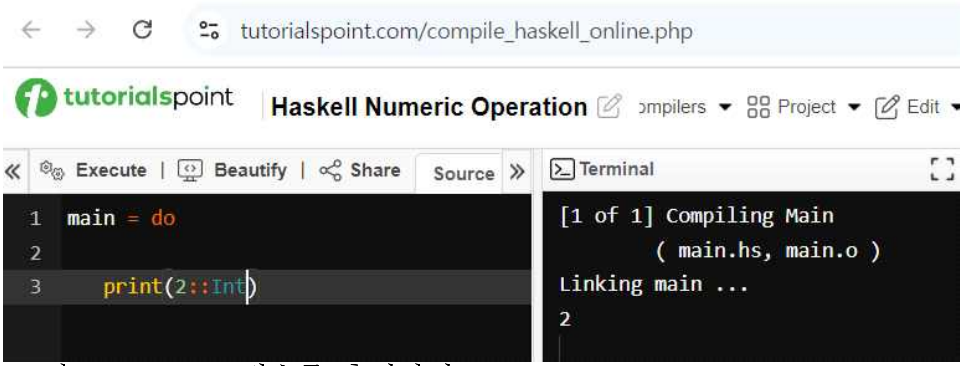
Prelude > 4
4

정수 타입을 코드에서 를 정의하기 위해서는 아래와 같이 할 수 있다.

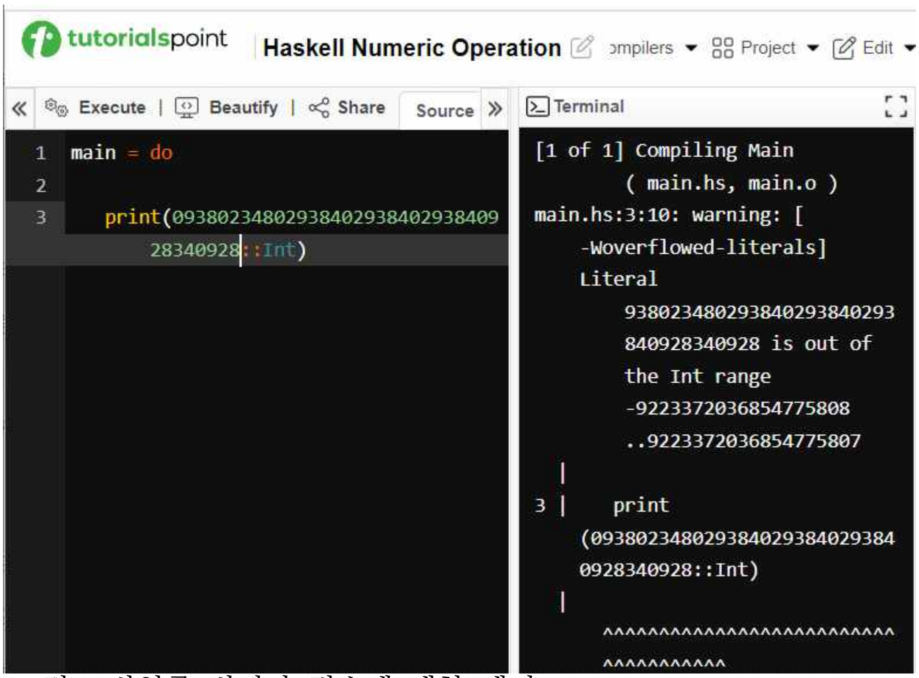
2:: Int

위는 정수 2를 정의한다.

print(2::Int) 는 정수 2를 출력한다.



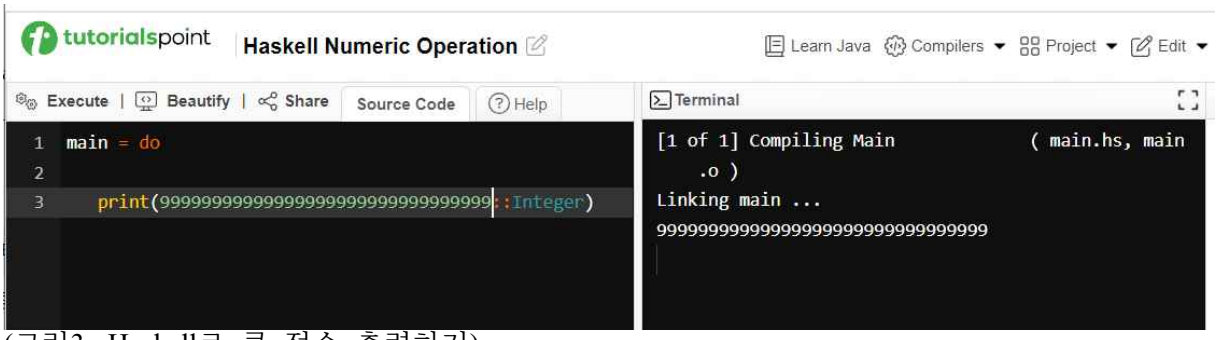
(그림1: Haskell로 정수를 출력하기)



(그림2: 범위를 벗어난 정수에 대한 에러)

Int는 값 범위에 제한이 있지만(-9223372036854775808 ~9223372036854775807) Interger는 제한이 없다. 따라서 큰 정수는 아래와 같이 정의할 수 있다.

2093402938402938409280349802938409293480290:: Integer



(그림3: Haskell로 큰 정수 출력하기)

(2) 문자열

문자열은 prelude command에서 다음과 같이 정의할 수 있다.

Prelude> :t “abcd”

“abcd” ::[char]

:t를 입력하여 문자열을 입력할 것임을 명시해야하고, 쌍따옴표로 묶어야한다.

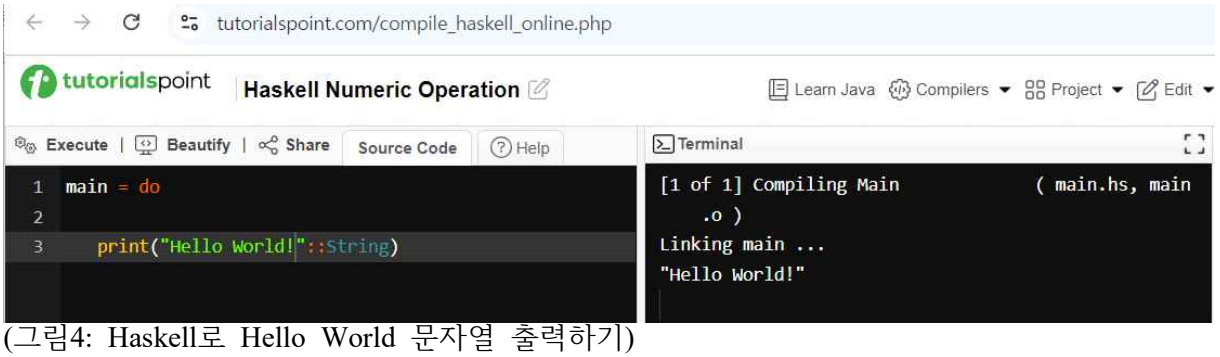
코드에서는 아래와 같이 정의할 수 있다.

“abcd”::[Char]

또는

“abcd”::String

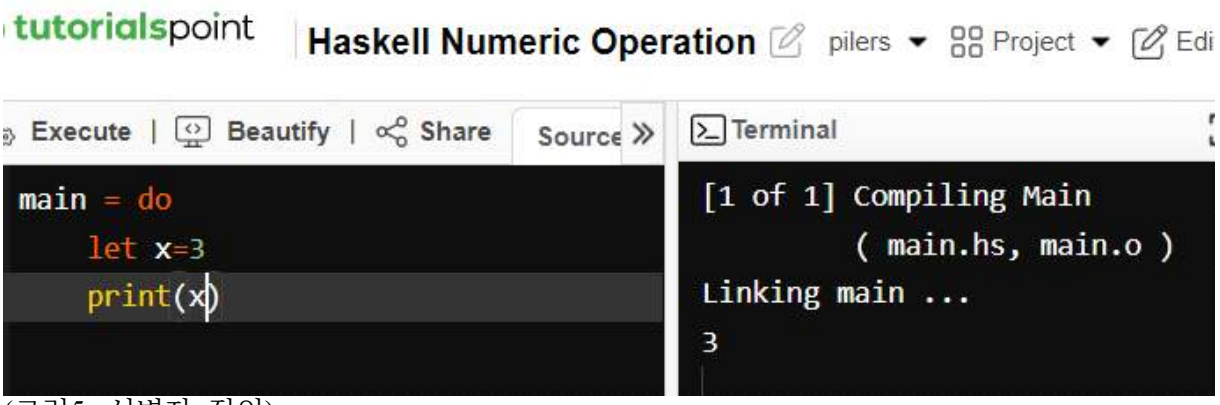
print(“abcd”::String) 은 “abcd를 출력한다.



(그림4: Haskell로 Hello World 문자열 출력하기)

(4) 식별자 정의

Haskeel에서 식별자는 다음과 같이 정의할 수 있다.



(그림5: 식별자 정의)

let x=3

x가 식별자이며, 여기에는 3이 저장되어있다. haskell은 순수 함수형 언어로, 한번 바인딩된 변수는 다시 재정의 될 수 없다.

즉

let x=3
x=4

는 오류를 발생시킨다.

4. C의 타입 정의 방법과 Haskell의 타입 정의 방법의 비교

(1) 정수

C의 정수 정의는 1 2 3 23 등 정수를 입력하여 정의할 수 있다. 즉 printf(“%d”,23)은 23을 출력한다.

Haskell의 정수 정의는 1 2 3 등 정수만 입력할 수 도 있고, 1::Int, 1::Integer 등 뒤에 ::Int, ::Integer를 명시하여 정수임을 명시할 수 있다. C와 다르게 제한이 없는 정수 타입도 가지고 있다.

(2) 문자열

C 문자열은 쌍따옴표 안에 문자열을 나열하여 정의할 수 있다. 즉 “abc”, “def” 등이다. printf(“%s,”abc“)는 abc를 출력한다.

Haskell에서도 마찬가지로 ”abc“, ”def“ 등 쌍따옴표 안에 정의할 수 있다. 또는 ”abc“:String으로 타입을 명시할 수 있다.

(3) 식별자
C에서 식별자는 타입을 명시하여 int x, char y 로 선언할 수 있다. C에서는 기본적으로 식별자에 같은 타입의 값을 여러번 넣을 수 있다.

```
int x=3;
x=4;
x=7;
x="hello" //error
```

Haskell에서는 let으로 식별자를 정의할 수 있다.

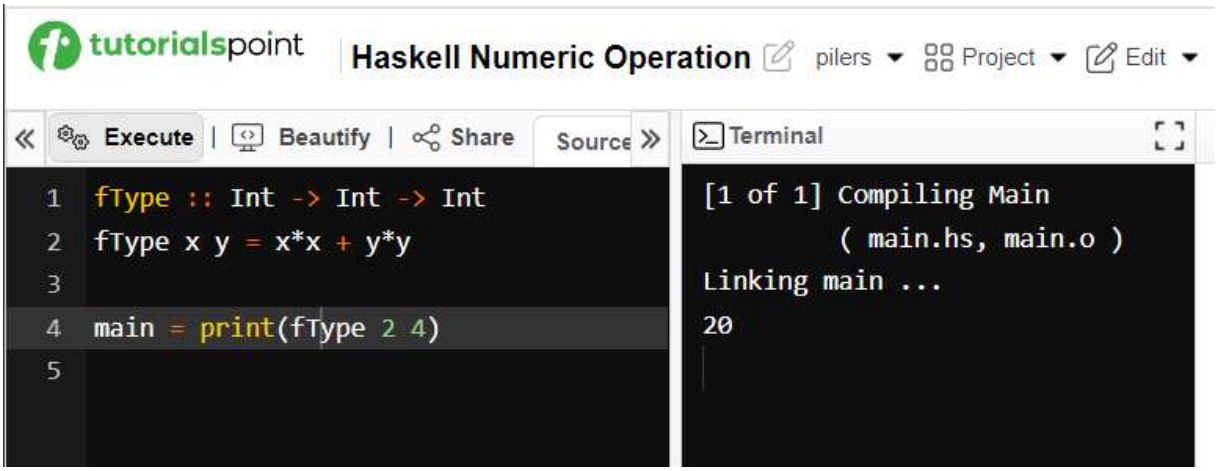
```
let x=3
let y="Hello"
```

Haskell은 순수 함수형 언어로서, 다시 재정의는 할 수 없다. 즉

```
let x=3
x=4
```

는 불가능하다.

번외로 Haskell에서는 함수를 정의하는 방법도 독특하다는 것을 확인하였다.



위와 같이 함수 프로토타입을 선언하고, lambda처럼 함수를 정의하고 사용하는 것을 알게 되었다.

4. 결론

(실험을 통해 배운 것을 정리합니다. 아래 Reference에 참고문헌을 추가합니다.)
BNF를 통해 어휘분석을 할 수 있다는 것을 알게되었고, 이것을 실제로 구현하는 것은 또 다른 문제라는 것을 알았다(구현하는데 많은 버그가 있었다).
여러 언어마다 정수, 문자열, 식별자를 정의하는 방법이 조금씩 다르다는 것을 알았다. 특히 순수 함수형 언어인 Haskell의 독특한 면을 알게 되었다.

References

[1]
geeksforgeeks, *Advantages and Disadvantages of OOP,* https://www.geeksforgeeks.org/benefits-advantages-of-oop/ (visited at 2024-10-03)

[2]
tutorialspoint, *Haskell Tutorial,* https://www.tutorialspoint.com/haskell/index.htm (visited at 2024-10-03)

[3] 박두순 저. “컴파일러의 이해”. 한빛아카데미. 2020

[4] StackOverflow, How to describe a quoted string in EBNF, https://stackoverflow.com/questions/59457364/how-to-describe-a-quoted-string-in-ebnf
(visited at 2024-10-03)