Practical - 8

2CS701 – Compiler Construction

Harshit Gajipara
19BCE059

Aim:

To implement a Type Checker.: Extend experiment 5 to assign Data type to each identifier as per declaration statement. Verify Data type as per each programming construct and report appropriate error message.

Code:

practical8.c

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n, flag = 0;
    char variable[15], type[15], b[15], c, percent='%';

    printf("\nGrammar for given statements: \n");
    printf("E -> E+E | E-E | E*E | E/E | E%cE | -E | (E) |
NUMBER\n\n", percent);

    printf("Enter the number of variables : ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++)
    {
        printf("Enter the variable[%d] : ", i);
        scanf(" %c", &variable[i]);
        printf("Enter the variable data-type[%d] (float-f,
int-i) : ", i);
        scanf(" %c", &type[i]);
        if (type[i] == 'f')
            flag = 1;
    }

    int expr_len = 0;
```

```
    printf("\nEnter the Expression(end with $) : ");
    getchar();
    while ((c = getchar()) != '$')
        b[expr_len++] = c;

    for (int i = 0; i < expr_len; i++)
    {
        if (b[i] == '/')
        {
            flag = 1;
            break;
        }
    }

    for (int i = 0; i < n; i++)
    {
        if (b[0] == variable[i])
        {
            if (flag == 1)
            {
                if (type[i] == 'f')
                    printf("\nThe datatype is correctly
defined!\n\n");
                else
                    printf("\nIdentifier '%c' must be of
float type!\n\n", variable[i]);
            }
            else
                printf("\nThe datatype is correctly
defined!\n\n");
            break;
        }
    }
    return 0;
}
```

Output:

```
Grammar for given statements:
E -> E+E | E-E | E*E | E/E | E%E | -E | (E) | NUMBER

Enter the number of variables : 5
Enter the variable[0] : a
Enter the variable data-type[0] (float-f, int-i) : f
Enter the variable[1] : b
Enter the variable data-type[1] (float-f, int-i) : i
Enter the variable[2] : c
Enter the variable data-type[2] (float-f, int-i) : i
Enter the variable[3] : d
Enter the variable data-type[3] (float-f, int-i) : f
Enter the variable[4] : e
Enter the variable data-type[4] (float-f, int-i) : f

Enter the Expression(end with $) : a=(-b/c)*(d%e)$

The datatype is correctly defined!

PS D:\19BCE059\B.Tech Semester 7\CC\CC Practicals\Practical 8>
```

```
Grammar for given statements:
E -> E+E | E-E | E*E | E/E | E%E | -E | (E) | NUMBER

Enter the number of variables : 3
Enter the variable[0] : x
Enter the variable data-type[0] (float-f, int-i) : i
Enter the variable[1] : y
Enter the variable data-type[1] (float-f, int-i) : i
Enter the variable[2] : z
Enter the variable data-type[2] (float-f, int-i) : i

Enter the Expression(end with $) : x=y/z$

Identifier 'x' must be of float type!
```

```
Grammar for given statements:
E -> E+E | E-E | E*E | E/E | E%E | -E | (E) | NUMBER

Enter the number of variables : 3
Enter the variable[0] : x
Enter the variable data-type[0] (float-f, int-i) : f
Enter the variable[1] : y
Enter the variable data-type[1] (float-f, int-i) : i
Enter the variable[2] : z
Enter the variable data-type[2] (float-f, int-i) : i

Enter the Expression(end with $) : x=y/z$

The datatype is correctly defined!
```