Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2024), B.Sc. in CSE (Day)

Expression Translator

Course Title: Compiler Lab
Course Code: CSE 306
Section: 222 D2

Students Details

Name	**ID**
Md. Moshiur Rahman	221902324

Submission Date: 8/07/2024
Course Teacher's Name: Tasnim Tayiba Zannat

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

The motivation in the back of this venture is to create a tool which can translate easy C mathematics expressions into equal Python and Java expressions. This may be beneficial for programmers transitioning between languages or for academic functions.

1.1 Problem Statement

1.1.1 Problem Definition

Develop a program which could parse a simple C arithmetic expression and translate it into equal Python and Java expressions.

1.2 Complex Engineering Problem

This assignment addresses the complex engineering trouble of language translation in programming. It requires understanding of syntax variations between C, Python, and Java, as well as parsing and string manipulation strategies.

Design Goals:

Create a person-friendly interface for input Accurately parse C expressions Correctly translate expressions to Python and Java Handle simple mathematics operations (+ , -, *, /). Provide clean error messages for invalid inputs.

1.2.1 Application:

This tool may be applied in academic settings, code migration initiatives, or as a brief reference for developers working across a couple of languages.

Chapter 2

Development

2.1 Introduction

The improvement process concerned developing a C program that may manage consumer enter, parse expressions, and output translations.

2.2 Project Details

The project includes a first-rate characteristic and several helping functions:

i) Parse expression: Determines which translation function to call.

ii) Translate to python: Converts the expression to Python

iii) Translate to java: Converts the expression to Java.

2.3 Implementation

The implementation makes use of C preferred libraries and focuses on string manipulation and parsing. Key techniques encompass:

i) Using fgets for secure input

ii) Tokenizing the enter string with strtok-r

iii) Switch statements for managing exceptional operators

iv) Error checking at various degrees of the procedure.

Chapter 3

Performance Evaluation

3.0.0 Implementation in C

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>


// func() prototypes

void parse_expression(char *expression, char *target_language);

void translate_to_python(char *expression);

void translate_to_java(char *expression);


int main() {

    char expression[100];

    char target_language[10];


    printf("Enter a C expression: ");

    fgets(expression, sizeof(expression), stdin);
```

```
    expression[strcspn(expression, "\n")] = '\0'; // remove newline chara

    printf("Enter target language (python or java): ");
    fgets(target_language, sizeof(target_language), stdin);
    target_language[strcspn(target_language, "\n")] = '\0'; // remove newline char

    parse_expression(expression, target_language);

    return 0;
}

void parse_expression(char *expression, char *target_language) {
    if (strcmp(target_language, "python") == 0) {
        translate_to_python(expression);
    } else if (strcmp(target_language, "java") == 0) {
        translate_to_java(expression);
    } else {
        printf("Error: Invalid target language\n");
    }
}

void translate_to_python(char *expression) {
    char *token;
```

```
char *rest = expression;
int operand1, operand2;
char operation;

// get the first operand
token = strtok_r(rest, " ", &rest);
if (token == NULL) {
    printf("Error: Invalid expression\n");
    return;
}
operand1 = atoi(token);

// get the operation
token = strtok_r(rest, " ", &rest);
if (token == NULL || strlen(token) != 1) {
    printf("Error: Invalid operation\n");
    return;
}
operation = token[0];

// get the second operand
token = strtok_r(rest, " ", &rest);
if (token == NULL) {
```

```
    printf("Error: Invalid expression\n");
    return;
}
operand2 = atoi(token);

// translate to Python
printf("Python equivalent: ");
printf("%d ", operand1);
switch (operation) {
    case '+':
        printf("+ ");
        break;
    case '-':
        printf("- ");
        break;
    case '*':
        printf("* ");
        break;
    case '/':
        printf("// ");  // int division in Python
        break;
    default:
        printf("Error: Invalid operation\n");
```

```
            return;
        }
        printf("%d", operand2);
        printf("\n");
    }

    void translate_to_java(char *expression) {
        char *token;
        char *rest = expression;
        int operand1, operand2;
        char operation;

        // get the first operand
        token = strtok_r(rest, " ", &rest);
        if (token == NULL) {
            printf("Error: Invalid expression\n");
            return;
        }
        operand1 = atoi(token);

        // get the operation
        token = strtok_r(rest, " ", &rest);
        if (token == NULL || strlen(token) != 1) {
```

```
    printf("Error: Invalid operation\n");
    return;
}
operation = token[0];

// get the second operand
token = strtok_r(rest, " ", &rest);
if (token == NULL) {
    printf("Error: Invalid expression\n");
    return;
}
operand2 = atoi(token);

// Translate to java
printf("Java equivalent: ");
printf("%d ", operand1);
switch (operation) {
    case '+':
        printf("+ ");
        break;
    case '-':
        printf("- ");
        break;
```

```
        case '*':
            printf("* ");
            break;
        case '/':
            printf("/ ");  // int division in java
            break;
        default:
            printf("Error: Invalid operation\n");
            return;
    }
    printf("%d", operand2);
    printf("\n");
}
```

3.0.1 Discussion

The application efficiently translates easy C expressions to Python and Java. It handles the four simple arithmetic operations and offers suitable error messages for invalid inputs.

3.0.2 Analysis Outcomes:

i) Correctly translates valid expressions

ii) Properly handles integer department differences between languages

iii) Provides clear blunders messages for invalid expressions or unsupported target languages.

Chapter 4

Snapshots

4.0.1 Snapshots of Output

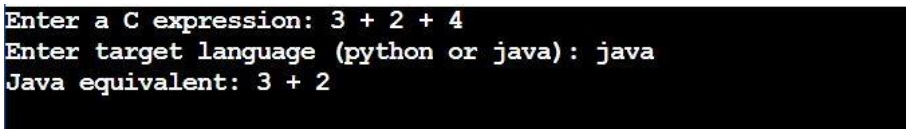
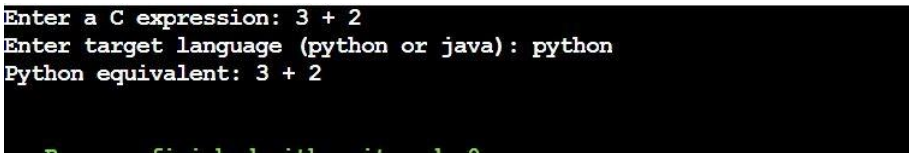


Fig: Addition operation java



Addition operation python

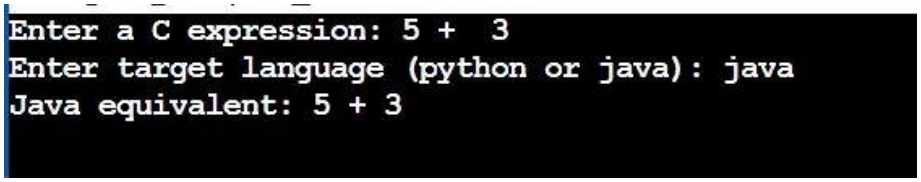


Fig: Addition operation in java

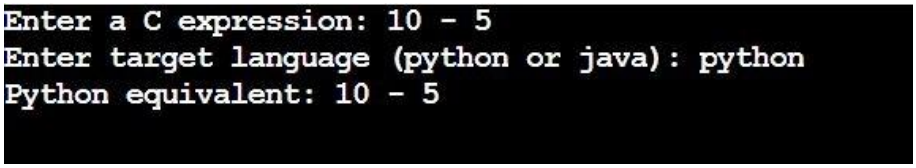


Fig-03: Subtraction operation

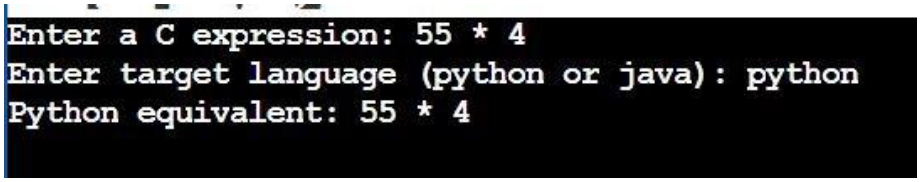


Fig: Multiplication operation

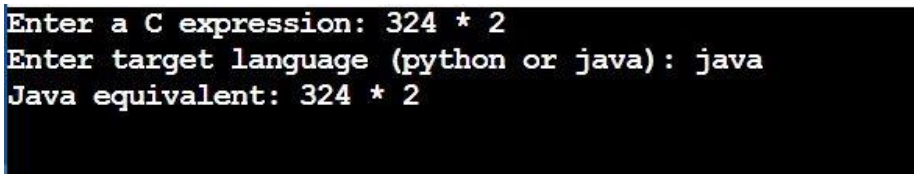


Fig : Java equivalent

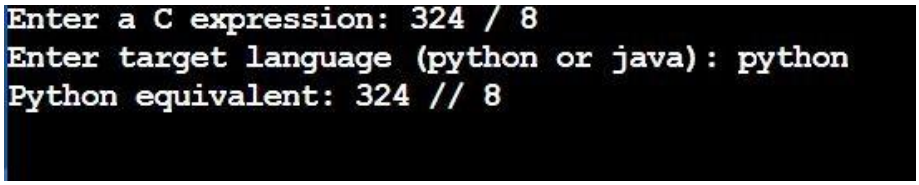


Fig : Division operation

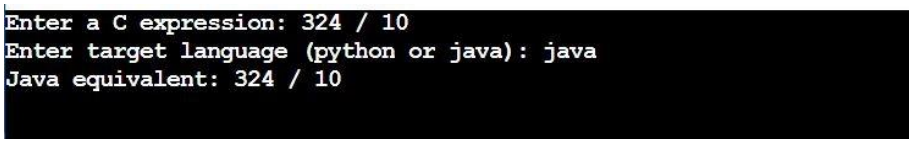


Fig:Java equivalent operation

Fig-03: Error cases

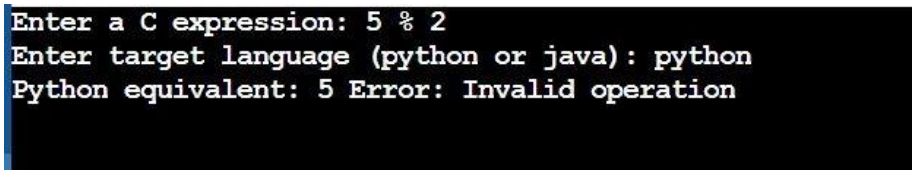


Fig-03: Error cases

Chapter 5

Conclusion

The C expression translator demonstrates the feasibility of creating a simple go-language translation tool for mathematics expressions.

5.0.1 Future Scope:

Potential upgrades and extensions include:

i) Supporting more complicated expressions (parentheses, more than one operations)

ii) Adding extra target languages (like C++, C, Javascript)

iii)Implementing a graphical consumer interface

iv)Handling floating-factor operations

v)Supporting variables and feature calls

5.1 References

[1] C Standard Library header files

[2] Error Reporting