

## **Project 1 : Interpreter Development**

Saejoon Kim, PH.D.

Dept. of Computer Science and Engineering

Sogang University

### **1. Design goal**

The interpreter is a computer program or environment that directly executes the source code of a programming language. Through this project, we will develop an interpreter that performs simple operations and practice the dynamic allocation, pointer, and string processing that we learned in class.

In this project, students create their own interpreters with simple operations. The program has the following restrictions.

- Development Environment : Linux (cspiro server)
- Use pointer and dynamic allocation to solve a given problem.

The production of this program should follow the basic course of the C programming class and the results should be submitted in accordance with the published format (source files and documents).

## 2. Requirements

### 2.1 Preparation

Define what needs to be implemented in this program and investigate the grammatical knowledge of C language required for production. Learn about the environment and constraints in which production is made, how this affects program development, and what additional features are available.

```
/* Describe what will be required for the development of the program. */  
/* Ex)The command of storing variables requires dynamic allocation,  
because .. */
```

The command of storing variables requires dynamic allocation, because the program must store the variable name and integer value using the associative array for storing variable-value pairs, and key is character array (string), so variable array should be 2-dimension array. And, the array must be released end of the program.

Getting command to the keyboard requires the function `fgets()` included in `<stdio.h>` because when using `scanf()`, `scanf()` reads input until it encounters space so it can't read input after space(' ').

The command of storing variables also requires pointer, because the program stores the variable-value at the user-defined function, the the associative array for storing variable -value pairs at main function also changes. And it can be implemented with call-by-reference.

To perform different user-defined functions such as `PRINT()`, `PRINT_VARS()` according to the command, it requires the function '`strlen()`' ,'`strcmp()`' included in `<string.h>` because after reading the command

To calculate the value with the operator requires function `atoi()` included in `<string.h>` because if the character string (the first or third part of command-Ex)"1" + "3") is a part of number, it should be converted to an integer.

## 2.2 Analysis

Design a function to meet the program's requirements, after the definition of the problem to be implemented and the acquisition of basic knowledge has been completed. To modularize the problem, the function, parameters, return values, etc. of the function are accurately identified and considered how each function is implemented.

*/\* Explain the flow of the program. \*/*

1. The computer reads a command with keyboard and stores it to the character array '\*com'.
  2. If the first part of com is "INT", user-defined function INT() checks if the third part of com is integer value. And if it is, user-defined function INT() stores the value and variable name in the array 'num1' and 'num2'.
  3. If the first part of com is "PRINT", user-defined function PRINT() checks if the second part of command (it is stored at character array var[]) is an element of character array \*\*num1, and if it is, prints the integer value of 'var[ ]'.
  4. If com is "PRINT\_VARS", user-defined function "PRINT\_VARS()" prints the names of all stored variables and their value.
  5. If the second part of com is '\*', '+', '-', '/' user-defined function CAL() check if the first and third part of com is integer value or an element of variable name array( \*\*num1), and if it is, the program calculates the integer values using the operators.
  6. Else, the program prints "error".
- Repeat this until the command is "EXIT".
- If com is "EXIT", the program breaks the loop condition and the program ends.

## 2.3 Development

Once the analysis is over, it goes into production. Implement each function to perform the function defined in the analysis step. If problems occur that were not found beforehand during production, they can be solved flexibly.

```
/* Explain in detail what function you have implemented and how it works */  
/* If you don't explain it, we'll assume you didn't implement it */
```

```
int main()
```

-this function reads command and stored variables  
- it performs different functions according to the command parameters

- i: the number of element of array 'num1','num2'
- n: the length of string 'com'
- num1: the array that stores variable names
- num2: the array that stores variable's value
- com: the command

- op1: the string that is front part of command
- op2: the string that is second part of command

return value: 0(program ends)

In this function, char \*com, char \*\*num1,int \*num2 is dynamically allocated using malloc() included in <stdlib.h>. But because \*\*num1 is 2- dimensional array, \*num1 is also dynamically allocated using for loop. The max size of variable name is 30. And there is no test case that stores more than 32 variables. So the size of allocated memory of \*\*num1 is sizeof(char\*)\*33 , \*num[i] ( i=0,1,2,...,32) is sizeof(char)\* 31 , \*num2 is sizeof(int)\*33, \*com is sizeof(char)\*129.

Using the while loop, the program repeated this.

1.op1,op2 are initialized.

2.The program reads the input with keyboard, and stored it to \*com. And because program uses 'scanf', the last element of \*com is '\0'. So it is changed to '\0'.

3. The program separates a command into three parts, but the last part is ignored. The first part is 'op1', and the second part is 'op2'.

4. If op1 is "INT", i=INT(com,num1,num2,i);

5. If op1 is "PRINT",PRINT(com,num1,num2,i);

6. If op2 is '\*', '/', '+', '-', CAL(op2,com,num1,num2,i);

7. If com is "EXIT", break the loop.

The program releases com,num1[j],num1,num2 and program ends.

(j=0,1,2,...,32)

int INT(char\*,char\*\*,int\*,int)

- this function stores integer value and variable name  
parameters

-com: the command

- num1: the array that stores variable names

- num2: the array that stores variable's value

- i: the last index of stored value of array 'num1','num2'

-num: if 'n' is integer value, num is equal to 'n'

num is integer value that is new variable's value

-n: the string that is the last part of command to check if it is integer value

return value: i; the number of elements of array 'num1','num2'

To check if the last part of com is integer value, the program initializes n, the last part of com. If  $((double)(n - (int)n) == 0)$  is 1, n is not double value. If n is double value, the program prints "error", and returns. Return value is i (the same as i at main function )

Then, to check if n is string value, use 'for loop'.

if  $(num[j] < '0' \&\& num[0] != '-' || (num[j] > '9' \&\& num[0] != '-'))$  is 1, print "error", and break the loop.

(j=0,1,...,strlen(num)-1)

And when  $j==strlen(num)-1$ , store the value using atoi(), and  $i=i+1$  because the number of elements of stored value is increased. Return value is i.

void CAL(char\*,char\*,char\*\*,int\*,int)

-this function calculate and print the value using the operators

parameters

- op2: the middle part of command
  - com: the command
  - num1: the array that stores variable names
  - num2: the array that stores variable's value
  - i: the number of elements of array 'num1','num2'
  - n1: the first part of command
  - n2: the last part of command
  - op: the operator to calculate
  - n3: the integer value that actually calculated
  - n4: the integer value that actually calculated
- return value : none

The program initializes op to op2[0]. And check if num1[j] is same as n1( the first part of com) using 'for loop'.(j=0,1,...,i-1) If it is, n3=num2[j]. Else, to check if n1 is integer value, if (n1[x] < '0' && n1[0] != '-' || (n1[x] > '9' && n1[0] != '-')) is 1, print "error\n" and return to main function. (x=0,1,...,strlen(n1)-1) Else, n3 is n1 using atoi().

This repeats at n2, the third part of com.

If i==0, and n1 and n2 are integer value, then n3=n1, n4=n2. (using if (n1[x] < '0' && n1[0] != '-' || (n1[x] > '9' && n1[0] != '-')) and for loop(x=0,1,...,strlen(n1)-1)) Else, print "error\n" and return to main function.

Then operate and print the value using op,n3,n4 and return. But when n4==0 and op=='/', print "error\n" .

void PRINT\_VARS(char\*\*,int\*,int)

- this function prints the name of all stored variables and their value
- num1: the array that stores variable names
- num2: the array that stores variable's value
- i: the number of elements of array 'num1','num2'

return value: none

using for loop, print the name of all stored variables and their value and return . If i is 0, print "empty" and return.

```
void PRINT(char*,char**,int*,int)
```

-this function searches the name of variable if it is in the arr 'num1' and prints a stored variable

parameters

-com: the command

- num1: the array that stores variable names

- num2: the array that stores variable's value

- i: the number of element of array 'num1','num2'

-var: the last part of the command

return value : none

First, the program initializes var[] (the last part of the command). And using strcmp(), check if var is same as num1[j], print num2[j]. (j=0,1,...,i-1)

Else, if j is n-1, print "not found" and return. And if i is 0, print "not found" and return.

## 2.4 Test

Whether the program produced is actually working and running according to the rules presented is the biggest criterion for evaluation. It is also considered whether all other possible exceptions have been handled, whether the problem has been correctly identified at the design stage, and whether the solutions have been reasonable and clear.

```
/* Describe how you resolved one of the possible exceptions */
```

```
/* Ex) if the program get double or string values, ... */
```

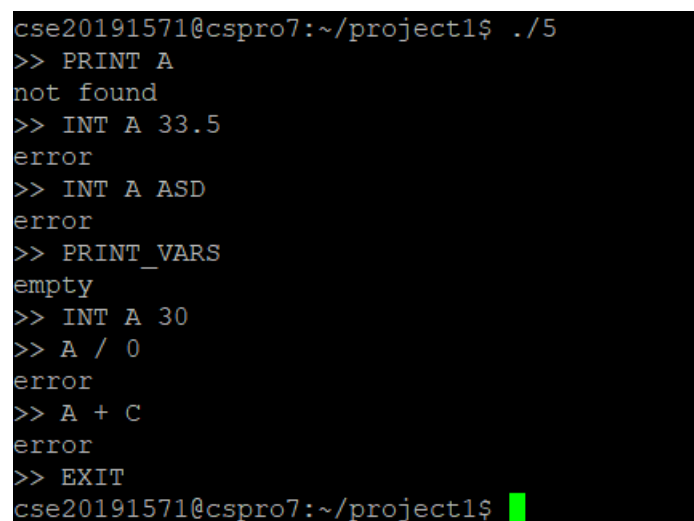
If the program get double or string values, we can separate this exception into two cases.

1. If the program get double value  $n$ ,  $(\text{double})(n - (\text{int})n) == 0$  is 0, so using if-else statement, then when the program gets double value, the program prints "error" and return.
2. If the program get string value,  $(\text{num}[j] < '0' \&\& \text{num}[0] != '-' || (\text{num}[j] > '9' \&\& \text{num}[0] != '-'))$  is 0, so using if-else statement, then when the program gets string value, the program prints "error" and break 'for loop'.

## 2.5 Evaluation

Enter several test cases and evaluate them.

*/\* Execute your program and show the results of your test cases \*/  
/\* and explain it with screenshots. \*/*



```
cse20191571@cspro7:~/project1$ ./5
>> PRINT A
not found
>> INT A 33.5
error
>> INT A ASD
error
>> PRINT_VARS
empty
>> INT A 30
>> A / 0
error
>> A + C
error
>> EXIT
cse20191571@cspro7:~/project1$
```

1. A command that print unsaved variable, the program prints "not found".
- 2,3. The program gets double or string value, it prints "error".
4. A command is "PRINT\_VAR" and there is no stored variable, print "empty".
5. 30 is integer value, so  $\text{num1}[0] = 'A'$  and  $\text{num2}[0] = 30$
6.  $(\text{integer value})/0$  is not integer value, so print "error".
7. C is unsaved variable, so print "error".
8. When a command is "EXIT", program ends.



```

cse20191571@cspro7:~/project1$ ./5
>> INT A 30
>> INT B 32
>> INT C 113
>> A + C
143
>> A + B
62
>> A - 30
0
>> PRINT_VARS
A=30
B=32
C=113
>> EXIT
cse20191571@cspro7:~/project1$ █

```

1. 30 is integer value, so num1[0]='A' and num2[0]=30
2. 32 is integer value, so num1[1]='B', and num2[1]=32
3. 113 is integer value, so num1[2]='C' and num2[2]=113.
4. A and C are elements of num1, so calculate the value with operator '+' and print 143.
5. A and B are elements of num1, so calculate the value with operator '+' and print 62.
6. A is an element of num1, and 30 is integer value, so it can calculate the value with operator '-' and print 0.
7. The command is "PRINT\_VARS", so print the names of all stored variables and their value.
8. The command is "EXIT", so break the loop and program ends.

## 2.6 Stability

It is required to identify various constraints and devise programmatic treatments that can complement situations where errors can be caused (C-language buffer overflow vulnerabilities). For this, students should create separate functions. After the completion of the actual program, the functions shall be possible according to the project details in the test process, and the modifications and completeness of the parts not properly performed shall be continuously carried out.

## **2.7 Durability**

In carrying out the project, students can accumulate basic knowledge of C language through learning about the pointer, dynamic arrangement, and string processing of C language and learn the principle of the operation of the interpreter in the process of coding themselves. Students can also actively handle the various errors that occur during the project and correct them on themselves.

## **2.8 Standard**

Basically, the project is built in compliance with the ANSI C standard of the gcc Compiler(5.4), and students connect to the Linux server and proceed with the project, so they provide a desktop and ssh interface program that can access that server. Sub-accounts are issued to each student on the Linux server to provide an environment where they can freely use their allocated capacity to proceed with the project.

# **3. Note**

## **3.1 Environment**

Students connect to a Linux server and proceed with the project, so they provide a desktop and ssh access program to connect to that server. Sub-accounts are issued to each student on the Linux server that they connect to to provide an environment where they can freely proceed with the project based on the capacity they receive.

## **3.2 Focus**

As a programming class that follows the first semester, the focus should be on generating interest from the sense of achievement in the completion of the project, including an additional simple algorithm in the grammatical part of programming,

and improving the ability to complete fragmentary knowledge that has been learned during the two semesters in a single project.

### **3.3 Team**

Individuals form a team.

### **3.4 Due-date**

Sun. November 10, 23:59:59

You must be able to complete the program and write a report within the stated project period.