CC Assignment

Implementation File

Group Members –

Sakshat Prakash – 2019A7PS0227U

Jay Patel – 2019A7PS0119U

Siddharth Bhandary – 2019A7PS0218

Including Libraries

```
add <stdio.h>

int main<>

{

statements ;

}
```

Implementation:

Successfully implemented the user can add stdio.h and it gets converted into the required C form which is #include<stdio.h>.The token is LIB for add part/

Main Part : Successfully implemented and the user can either use int main or void main which will get converted into int main() which is the corresponding C form.

The token is MAIN

Comments - $

A comment is made by starting it with **$**

Eg: $ hi from a new language

Implementation:

Successfully implemented the user can comment any line in the language using the dollar symbol and gets converted to // in C.The token used is COMMENT.

Ending symbol - ;

Just like in C each statement in this language is separated by **;**

Implementation:

Successfully implemented, the user can end any line in the language using the semicolon symbol and gets converted to ; in C. The token used is SC.

Next line - \l post<<\l;

Here \l is used to make a new line. (In C \n is used to make a new line)

Implementation:

Successfully implemented ,the user can add the next line function in any of the printf string statements using the \l symbol and gets converted to \n in printf statement in C.The token used is NL.

Tab - \tab

Here \tab is used to make the cursor move some spaces to the right. (In C \t is used to do the same)

Implementation:

Could not successfully implement since we could not trace in the string where the whole tab character can be matched to only the \t symbol in C.

Declaration Statements –

- **Declaration of a variable - name1,name2,name3 (type);**

  Here variables are declared with their name followed by its specified type which is enclosed by parenthesis.

  Eg: a,b,c(int32);

Implementation:

Successfully implemented ,the user can add declare one or more than one variables with the given type and would successfully get converted into its corresponding C translation.

- **Declaration for array- name1{x},name2{y} (type);**

  Here arrays are declared with their name along with the number of elements in the array being disclosed by curly brackets and its type of array is enclosed by parenthesis.

  Eg: students{10},teachers{5}(int32);

Implementation:

Successfully implemented ,the user can add declare one or more arrays of any data type and would successfully get converted into its corresponding C translation.

Input and Output- post<<varname; get>>varname;

Here for input from the user we type post followed by 2 angle brackets

Here for output from the user we type get followed by 2 angle brackets

Eg:

variable (int32);

get>>variable; (Takes input from the user)

post<<"Hello from our new language";

post>>variable; (Prints the value of the variable in the screen)

Implementation:

Successfully implemented the user can read or write using the post and get commands and gets converted to printf and scanf commands in C.

The token used is post and get.

Operators-

Assignment operator - =:

Addition operator - +

Subtraction operator - -

Multiplication operator - *

Division operator - /

AND Operator - & is equivalent to && in C

OR Operator - | is equivalent to || in C

NOT Operator - ! is equivalent to ! in C

Inequality Operator - !!=:

Increment and Decrement Operators - a++ is a++ and a-- is a--

Equal To Operator - -> is equivalent to ==

Greater Than or Less Than Operator - >=: , <=:

Implementation:

Successfully implemented the user can use these operators which would get converted to C.

Conditional statements-

If Statement -

if <condition> {statments} elif <condtion> {statements} default {statement}

Here we make an if statement by typing If followed by a condition for the if statement enclosed by a parenthesis.

If the if statement is not satisfied an Elif statement is used to execute another condition which is enclosed by a parenthesis.

If neither conditions are satisfied a Default statement is printed which prints a specific statement which is enclosed by curly brackets.

Eg: If<enrolledincc -> {post<<"Make a new language";}

default {post<<"no need to make a new language";}

Implementation:

Successfully implemented the user can use the if, elif block and would get converted into C's if else block. The tokens used are IF,ELIF AND DEFAULT.

Switch Statement -

```
option <expression>
{
 case1:
 statement
 leave;
 case2:
 statement
 leave;
 default:
 statement
}
```

Here the Switch statement is known as option and has an expression enclosed in two angle brackets. Here there are 3 case statements. The last case statement is a default statement where the other cases are not satisfied. They are enclosed by 2 curly brackets.

Implementation:

Successfully implemented the user can use the switch statement block and would get converted into C's switch statement. The tokens used are OPTION and DEFAULT

Goto Statement – reach label1;

reach is equivalent to goto in C.

```
Reach L1;
```

Implementation:

Successfully implemented the user can use the goto statement and the ocde will convert it into goto which is the conversion in C.The token used is REACH

Break Statement – leave

leave is equivalent to break in C

Implementation:

Successfully implemented the user can use the leave statement in case block which will be converted into leave which is the conversion in C.The token used is BREAKK

Continue Statement - skip

skip is equivalent to continue in C

Implementation:

Successfully implemented ,the user can use the skip statement and the code will convert it into continue which is the conversion in C.The token used is CONTINUE_STMT

Return Statement – ret

ret is equivalent to return in C.

Eg:ret 0;

Implementation:

Successfully implemented the user can use the ret statement and the code will convert it into return statement which is the conversion in C.

The token used is RETURN_STMT

<u>Loops -</u>

For loop -

for <;;>{statement} is equivalent to for in C

Implementation:

Successfully implemented the for loop which gets converted into the correspondent for loop in C.

While loop-

till <condition>

do {statement}

till is equivalent to while in C

do is equivalent to do in C

Implementation:

Successfully implemented the while loop which gets converted into the correspondent while loop in C. The tokens for the while loop are TILL and DO,LSB,RSB,DO,LCB,RCB.

Do While loop-

do {statement}

till <condition>

till is equivalent to while in C

do is equivalent to do in C

Implementation:

Successfully implemented the do while loop which gets converted into the correspondent do while loop in C. The tokens for the while loop are DO,LCB,RCB,TILL,LSB AND RSB.

Functions-

function_name <parameters> (type) {statements}

To call function - Call function_name<parameters>;

Eg:

function1<a>(int32)

{post<<"hello from function1";}

Implementation:

Successfully implemented the functions declaration and calling part.The function name is taken as a variable and the arguments are to be stored in FUNC_VAR_LIST. The tokens used ar FUNC-NAME LSB RSB LB TYPE RB.

Also implemented function declared as static

Pointers-

^var_name (type);

var_name = @var_name;

Eg:

^pointer(int32);

pointer=@a; (Pointer pointing to the address of a)

@ for & in c (address)

Implementation:

Successfully implemented the pointers declaration and calling part. The tokens used are POINT LB RB SC and TYPE to denote what kind of pointer it is.

Data Types

int 32-int in C

int 64 -long in C

chr -char in C

fp -float in C

dfp -double in C

bool -bool in C

Implementation:

Successfully implemented the data type’s part which gets successfully converted to their equivalent C data type. The token are the names of data types.