VIVEK CHAUHAN
590029184

EXPERIMENT 4 : VARIABLES AND SCOPE OF VARIABLES

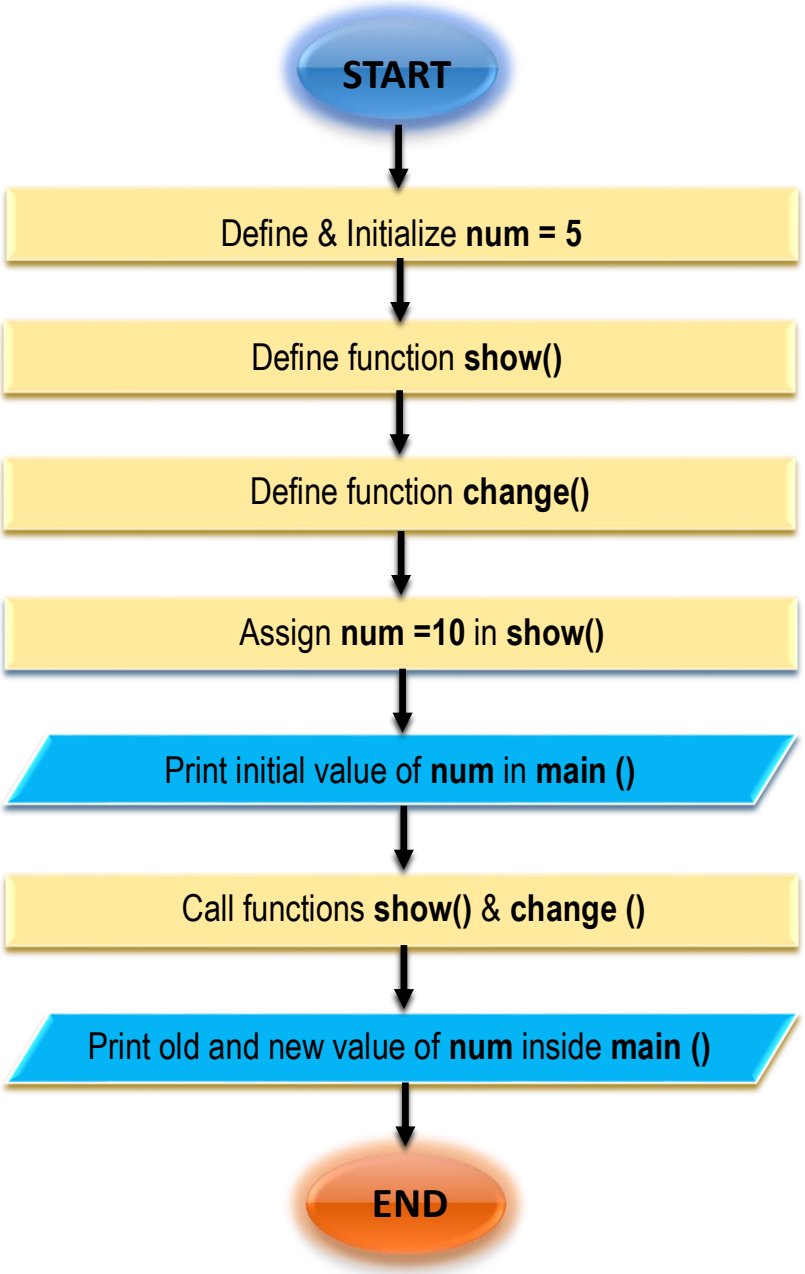
Activity 1: *Declare a global variable outside all functions and use it inside various functions to understand its accessibility.*

ALGORITHM

STEP 1: Start the program.
STEP 2: Declare and initialize a global variable num = 5
STEP 3: Define the function show()
STEP 4: In show(), print the current value of num.
STEP 5: Define the function change()
STEP 6: In change(), assign num = 10 and print new value of num
STEP 7: In main(), print the initial value of num
STEP 8: Call the function show()
STEP 9: Call the function change()
STEP 10: Print value of num in main()
STEP 11: End

VIVEK CHAUHAN
590029184

FLOWCHART :



VIVEK CHAUHAN
590029184

PSEUDOCODE :

```
START

DECLARE global variable num = 5

FUNCTION show
    print "Inside show(): num = ", num
END FUNCTION

FUNCTION change
    SET num = 10
    PRINT "Inside change(): num = ", num
END FUNCTION

MAIN FUNCTION
    print "In main(): num = ", num
    show()
    change()
    print "Back in main(): num = ", num
END MAIN

END
```

CODE :

```c
#include <stdio.h>

// Global variable (accessible in all functions)
int num = 5;

void show() {
    // Using the global variable directly
    printf("Inside show(): num = %d\n", num);
}

void change() {
    // Modifying the global variable
    num = 10;
    printf("Inside change(): num = %d\n", num);
}
```

```c
int main() {
    printf("In main(): num = %d\n", num);
    show();
    change();
    printf("Back in main(): num = %d\n", num);
    return 0;
}
```

OUTPUT:

Activity 2: *Declare a local variable inside a function and try to access it outside the function. Compare this with accessing the global variable from within the function.*

ALGORITHM :

STEP 1: Start
STEP 2: Declare and initialize a global variable *globalvar*
STEP 3: Define the function localexample()

STEP 4: Declare a local variable *localvar* inside localexample() and initialize it.

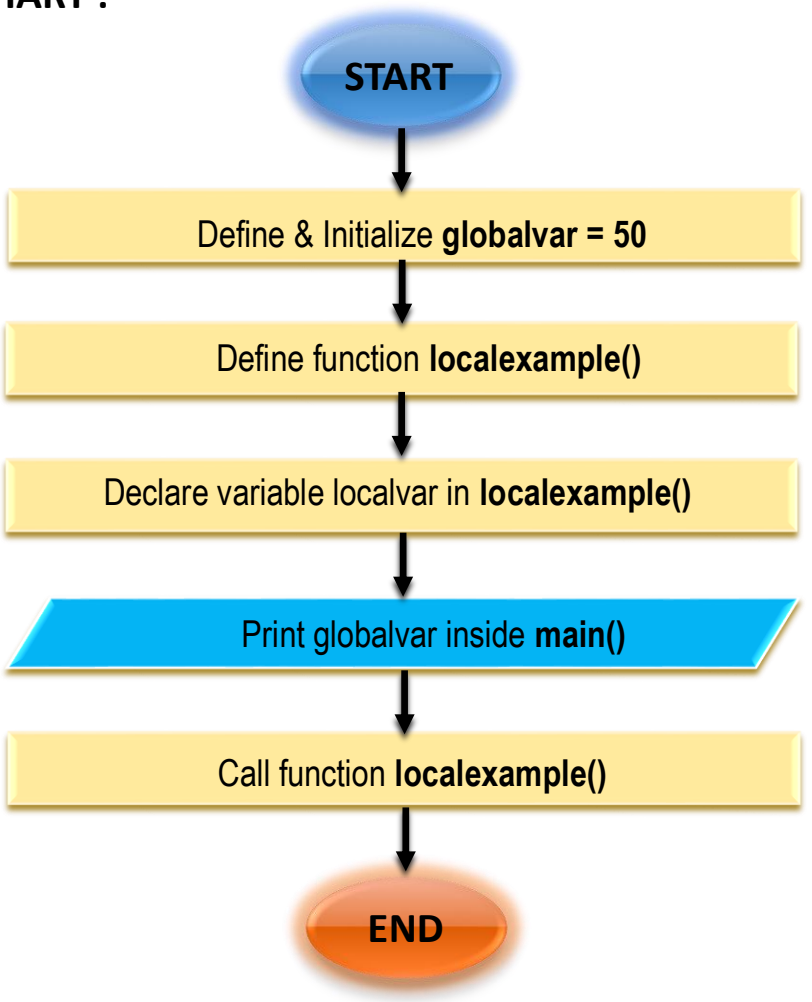
STEP 5: Inside localexample(), print localvar and globalvar.

STEP 6: In main(), print the value of globalvar.

STEP 7: Call the function localexample().

STEP 9: End

FLOWCHART :



VIVEK CHAUHAN
590029184

PSEUDOCODE:

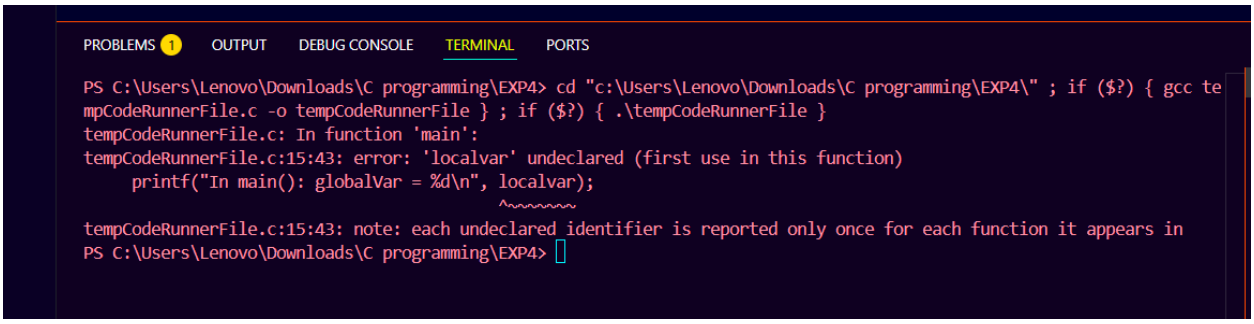
```
START

Declare global variable globalvar = 50

FUNCTION localexample
    DECLARE local variable localvar = 10
    print "Inside localExample(): localvar = ", localvar
    print "Inside localExample(): globalvar = ", globalvar
END FUNCTION

MAIN FUNCTION
    print "In main(): globalvar = ", globalvar
    localExample()

END MAIN

END
```

CODE :

```
#include <stdio.h>

// Global variable
int globalvar = 50;

void localExample() {
    // Local variable
    int localvar = 10;
    printf("Inside localExample(): localVar = %d\n", localvar);
    printf("Inside localExample(): globalVar = %d\n", globalvar);
}

int main() {
    printf("In main(): globalVar = %d\n", globalvar);
    printf("In main(): globalVar = %d\n", localvar);
```

```
    return 0;
}
```

OUTPUT :



Encountered with an error for accessing the localvar inside the main() function

---

Activity 3: *Declare variables within different code blocks and (enclosed by curly braces) and test their accessibility within and outside those blocks.*

ALGORITHM :

STEP 1: Start
STEP 2: Declare and initialize variable x in the main()
STEP 3: Print the value of x in the main block
STEP 4: Enter inner block 1
STEP 5: Declare and initialize variable y in inner block 1
STEP 6: Print the values of x and y inside inner block 1
STEP 7: Enter inner block 2 inside inner block 1
STEP 8: Declare and initialize variable z in inner block 2

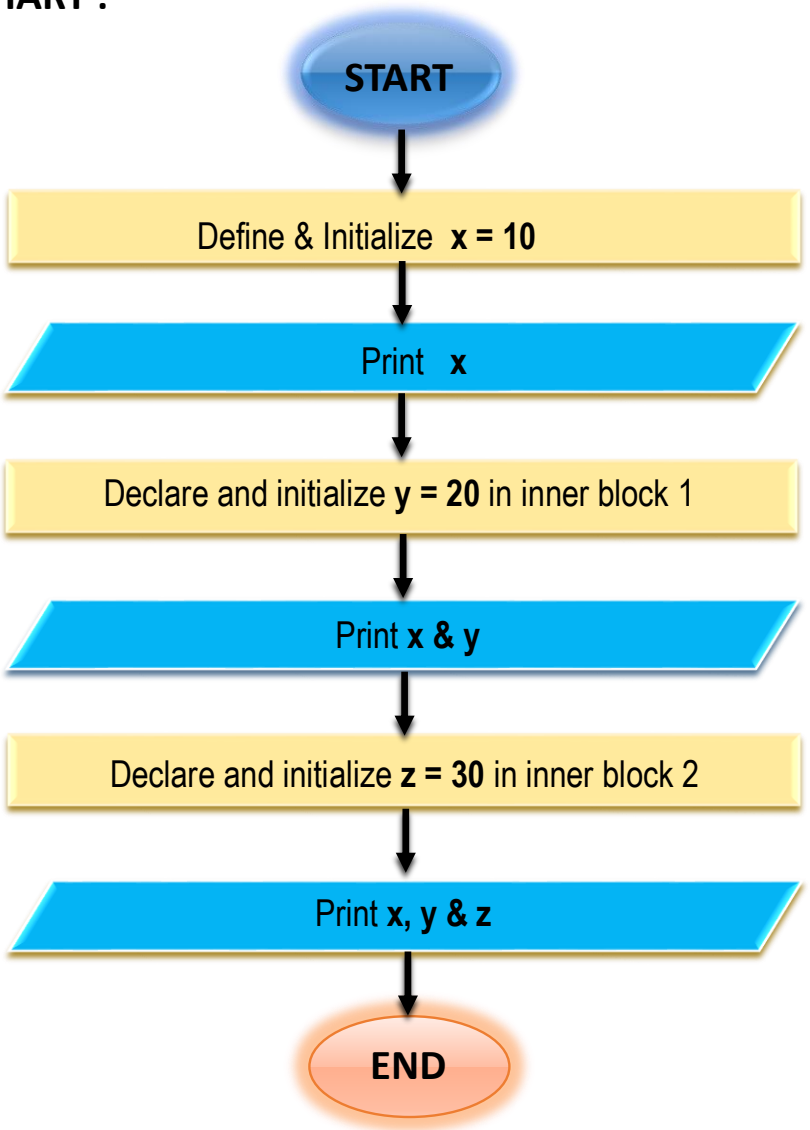
STEP 9: Print the values of x, y, and z inside inner block 2

STEP 10: Exit inner block 2

STEP 11: Exit inner block 1

STEP 12: End

FLOWCHART :



VIVEK CHAUHAN
590029184

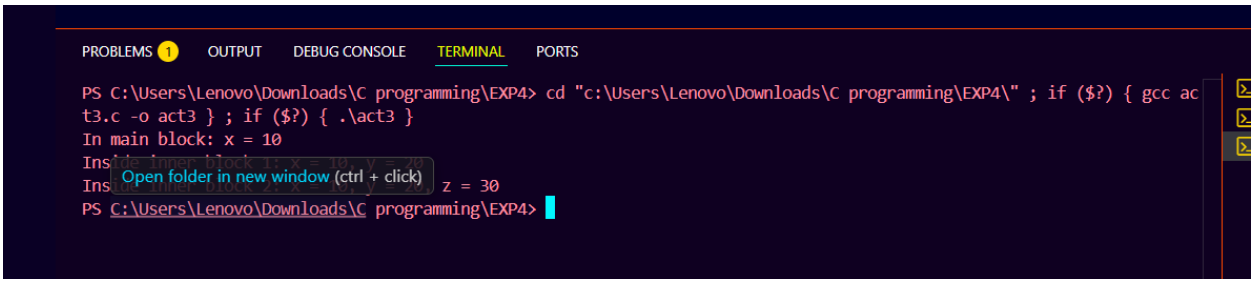
PSEUDOCODE :

```
START

DECLARE variable x in main block
print "In main block: x = ", x

ENTER inner block 1
    DECLARE variable y
    print "Inside inner block 1: x = ", x, ", y = ", y

    ENTER inner block 2
        DECLARE variable z
        print "Inside inner block 2: x = ", x, ", y = ", y, ", z = ", z
    EXIT inner block 2

EXIT inner block 1

END
```

CODE :

```c
int main() {
    int x = 10; // Variable in main block
    printf("In main block: x = %d\n", x);

    {
        int y = 20; // Variable in inner block 1
        printf("Inside inner block 1: x = %d, y = %d\n", x, y);

        {
            int z = 30; // Variable in inner block 2
            printf("Inside inner block 2: x = %d, y = %d, z = %d\n", x, y, z);
        }


    }

    return 0;
}
```

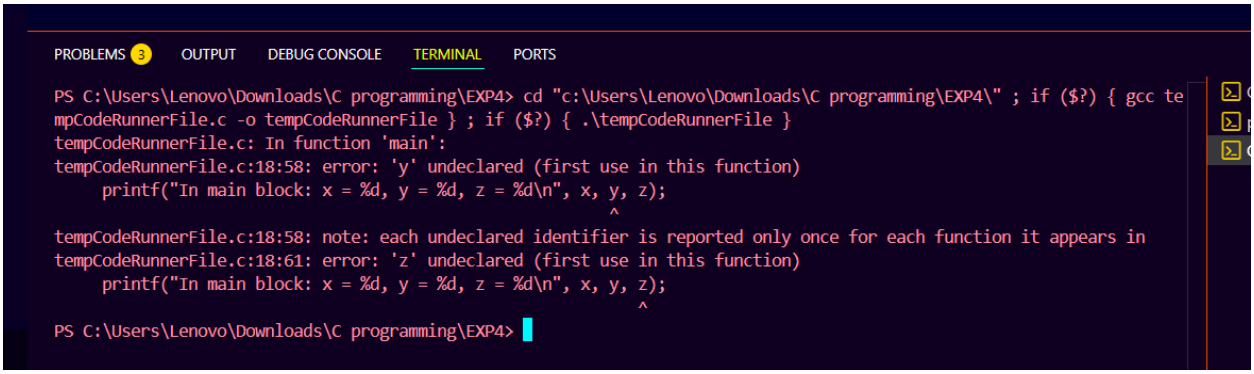
OUTPUT :




```
printf("In main block: x = %d, y = %d, z = %d\n", x, y, z);
```

y and z are not accessible outside inner blocks

This would cause a compilation error



Activity 4: *Declare a static local variable inside a function. Observe how its value persists across function calls.*

VIVEK CHAUHAN
590029184

ALGORITHM :

STEP 1: Start

STEP 2: Define the function counter() with static local variable count=0

STEP 3: Increment count by 1

STEP 4: Print the value of count

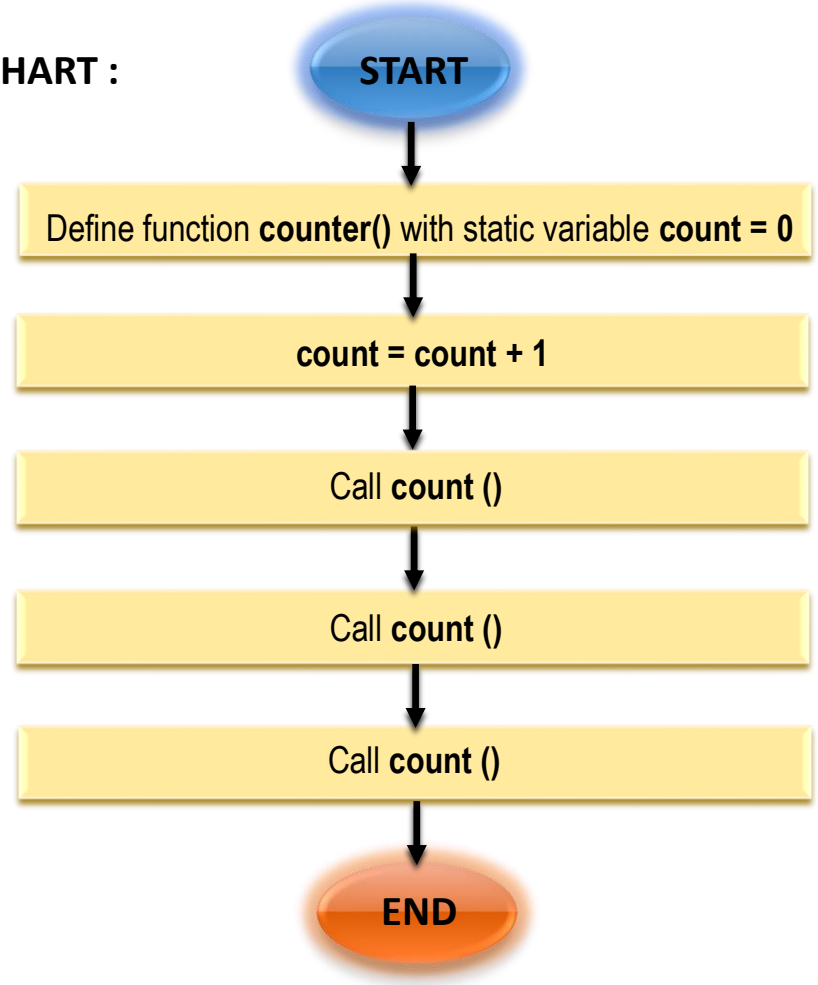
STEP 5: In main(), call counter() for the first time

STEP 6: Call counter() for the second time

STEP 7: Call counter() for the third time

STEP 8: End

FLOWCHART :



VIVEK CHAUHAN
590029184

PSEUDOCODE :

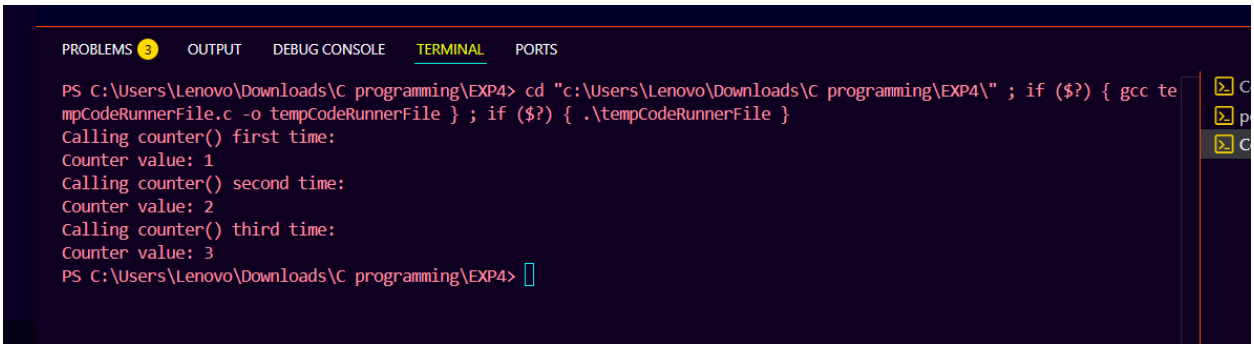
```
START

FUNCTION counter
    DECLARE static local variable count = 0
    Count = count + 1
    print "Counter value: ", count
END FUNCTION

MAIN FUNCTION
    print "Calling counter() first time:"
    Call counter()

    print "Calling counter() second time:"
    Call counter()

    print "Calling counter() third time:"
    Call counter()
END MAIN

END
```

CODE :

```c
#include <stdio.h>

void counter() {
    static int count = 0;  // static local variable
    count++;
    printf("Counter value: %d\n", count);
}

int main() {
    printf("Calling counter() first time:\n");
    counter();

    printf("Calling counter() second time:\n");
    counter();

    printf("Calling counter() third time:\n");
    counter();
    return 0;
}
```

VIVEK CHAUHAN
590029184

OUTPUT :



*Static variable count **retained its value between function calls**, unlike normal local variables which are reinitialized each time.*