NORTH SOUTH UNIVERSITY

Department of Electrical and Computer Engineering

Assignment – 01

Name	: Joy Kumar Ghosh
Student ID	: 2211424 6 42
Course No.	: CSE 425
Course Title	: Concepts of Programming Language
Section	: 1
Date	: 28 September 2024

Ans. to the ques. no. : 1

Static scoping is a method introduced by ALGOL 60 to binding names to nonlocal variables, which has been adopted by many languages. Scoping is necessary to allow reuse of variables.

An acceptable norm is to use shorter variable names. But if we have a program containing millions of lines of code, then shorter variable names may fall short. Therefore, we must reuse variable names and scoping just allow it.

In programming language 'C' scoping works like:

If we declare a global variable, we can access it's value from any defined function. But if we declare a local variable inside a defined function with the same name of global variable, then the function will use the values of local variable.

Example code written and compiled in an online compiler "programiz.com" are given bellow:

main.c

Share Run

```c
#include <stdio.h>

int number;

void function1(){
    //this will use the value from main or global variable
    printf("From function1: %d\n", number);
}

void function2(){
    int number = 8;

    //this function still work as it own rules,
    //will print global variable
    function1();

    //this will print the local value of this function
    printf("From function2: %d\n", number);
}

int main() {
    number = 5;
    function1();
    function2();
    return 0;
}
```

Output Clear

```
/tmp/MmEryEYLtN.o
From function1: 5
From function1: 5
From function2: 8


=== Code Execution Successful ===
```

Ans. to the ques. no. 02

In imperative-language programs, computations involve evaluating expressions and assigning values to variables. To make computations flexible and powerful, control statements are necessary. Control statements control the flow of execution. Programming language 'C' supports several control structures that manage the flow of the program.

Selection or decision statements:

A selection statements provides the means of choosing between two or more execution paths in a program. It could be two way or multiple way selection.

Two way selection statements:

statements of if, if-else:

```
if (a > b){
    printf ("a is greater than b");
}
else {
    printf ("b is greater than a");
}
```

Multiple way selection statements : switch-case

```
switch (operator) {
    case '+':
        result = a + b;
        break;
    case '-':
        result = a - b;
        break;
    case '*':
        result = a * b;
        break;
    case '/':
        result = a / b;
        break;
    default :
        printf ("Invalid Operator");
}
```

Iterative Statements :

An iterative statement is one that causes a statement or collection of statements to be executed zero, one, or more times. It also called as loop.

Counter-controlled Loops:

A counting iterative control statement maintains a count value through a variable called the loop variable, with initial, terminal, and step-size specifications. In PL 'C' 'for' loop used as counter-controlled loops.

```
for (i=0; i<10; i++){
        sum += i;
}
```

Logically Controlled loops:

Logically controlled loops are useful for repeatedly executing collections of statements, as they are more general than counter-controlled loops and are essential for expressing control structure in flowchart. Actually, when we don't know exactly how many times the loops need to run, then we use logically controlled loops. In PL 'C' while and do-while used as logically controlled loops.

```
while (i<10){
    sum += i;
    i++;
}
```

```
do {
    sum += i;
    i++;
} while (i<10);
```

Programming Language C also offen some user located loop control mechanisms, such as break, continue, return.

```
while (true){
    sum += i;
    i++;
    if (i>=10){
        break;
    }
}
```

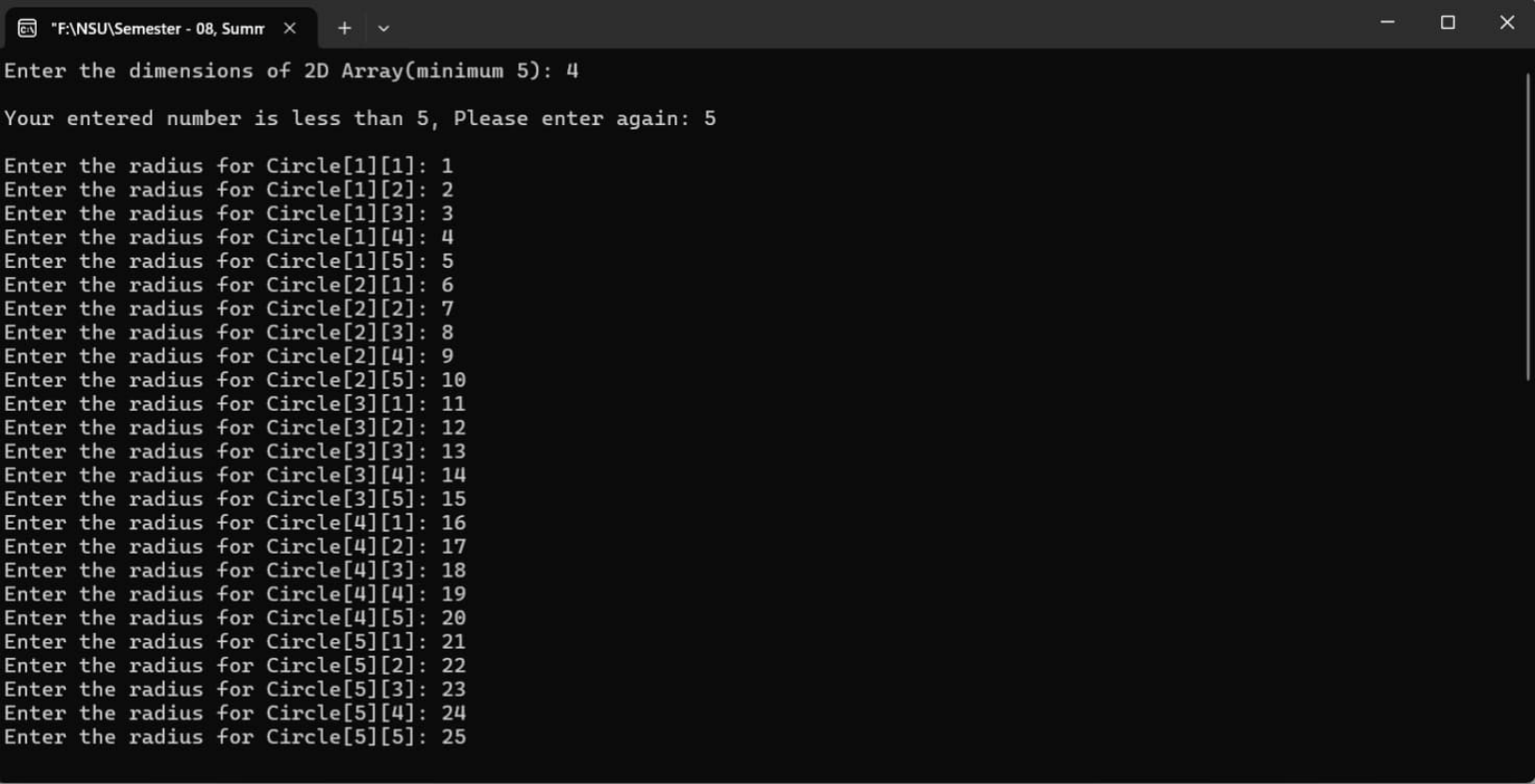
```
while (true){
    sum += i;
    i++;
    if (i<10){
        continue;
    }
    break;
}
```

Assignment-01_Code-03.c

```c
#include <stdio.h>
#include <math.h>

double calculateArea(int radius)
{
    return M_PI * radius * radius;
}

double calculatePerimeter(int radius)
{
    return 2 * M_PI * radius;
}

int main()
{
    int dimensions, count = 1;
    double perimeter;

    printf("Enter the dimensions of 2D Array(minimum 5): ");
    scanf("%d", &dimensions);

    while(dimensions < 5){
        printf("\nYour entered number is less than 5, Please enter again: ");
        scanf("%d", &dimensions);
    }

    printf("\n");

    //declaring 2D array
    int radiusArray2D[dimensions][dimensions];
    double areaArray2D[dimensions][dimensions];

```

Assignment-01_Code-03.c

```c
    FILE *fptr;
    fptr = fopen("Circle-Data.txt", "w");
    if(fptr == NULL){
        printf("Error opening file!");
        return 0;
    }

    fprintf(fptr, "Serial No.\tRadius\t\tArea\t\tPerimeter\n\n");

    for(int i = 0; i < dimensions; i++){
        for(int j = 0; j < dimensions; j++){
            printf("Enter the radius for Circle[%d][%d]: ", i+1, j+1);
            scanf("%d", &radiusArray2D[i][j]);

            areaArray2D[i][j] = calculateArea(radiusArray2D[i][j]);
            perimeter = calculatePerimeter(radiusArray2D[i][j]);

            fprintf(fptr, "%d\t\t%d\t\t%.2f\t\t%.2f\n", count, radiusArray2D[i][j], areaArray2D[i][j], perimeter);
            count++;
        }
    }

    fclose(fptr);

    printf("\n\nTwo dimensional array of radius and corresponding Circle Areas:\n\n");
    printf("Radius\tArea\n");

    for(int i = 0; i < dimensions; i++){
        for(int j = 0; j < dimensions; j++){
            printf("%d\t%.2f\n", radiusArray2D[i][j], areaArray2D[i][j]);
        }
    }
    printf("\nCircle Data successfully written to 'Circle-Data.txt'.\n");
    return 0;
}
```


"F:\NSU\Semester - 08, Summ
Enter the dimensions of 2D Array(minimum 5): 4
Your entered number is less than 5, Please enter again: 5
Enter the radius for Circle[1][1]: 1
Enter the radius for Circle[1][2]: 2
Enter the radius for Circle[1][3]: 3
Enter the radius for Circle[1][4]: 4
Enter the radius for Circle[1][5]: 5
Enter the radius for Circle[2][1]: 6
Enter the radius for Circle[2][2]: 7
Enter the radius for Circle[2][3]: 8
Enter the radius for Circle[2][4]: 9
Enter the radius for Circle[2][5]: 10
Enter the radius for Circle[3][1]: 11
Enter the radius for Circle[3][2]: 12
Enter the radius for Circle[3][3]: 13
Enter the radius for Circle[3][4]: 14
Enter the radius for Circle[3][5]: 15
Enter the radius for Circle[4][1]: 16
Enter the radius for Circle[4][2]: 17
Enter the radius for Circle[4][3]: 18
Enter the radius for Circle[4][4]: 19
Enter the radius for Circle[4][5]: 20
Enter the radius for Circle[5][1]: 21
Enter the radius for Circle[5][2]: 22
Enter the radius for Circle[5][3]: 23
Enter the radius for Circle[5][4]: 24
Enter the radius for Circle[5][5]: 25

Two dimensional array of radius and corresponding Circle Areas:

Radius	Area
1	3.14
2	12.57
3	28.27
4	50.27
5	78.54
6	113.10
7	153.94
8	201.06
9	254.47
10	314.16
11	380.13
12	452.39
13	530.93
14	615.75
15	706.86
16	804.25
17	907.92
18	1017.88
19	1134.11
20	1256.64
21	1385.44
22	1520.53
23	1661.90
24	1809.56
25	1963.50

```
Circle Data successfully written to 'Circle-Data.txt'.

Process returned 0 (0x0)   execution time : 26.371 s
Press any key to continue.
```