North South University

Programming Assignment 1

Submitted by:

Arka Karmoker

ID: 2112343642

Course: CSE425

Section: 1

Submitted to:

Dr. Md Shahriar Karim [MSK1]

Department of Electrical and Computer Engineering (ECE)

North South University

Date of submission: 23/02/2025

Question 1

Code:

```
#include <stdio.h>

// Global variable
int x = 10;

// Function prototypes
void outer();
void inner();

void inner() {
    // Inner function uses the nearest variable 'x' (which is
global in this case)
    printf("Inside inner(), x = %d\n", x);
}

void outer() {
    int x = 20; // Local variable inside outer()

    printf("Inside outer(), x = %d\n", x);

    inner(); // Call inner() function, which uses the global x due
to static scoping
}

int main() {
    outer();  // Call outer() function
    printf("Inside main(), x = %d\n", x);  // Uses global x

    return 0;
}

// Output:
// Inside outer(), x = 20
// Inside inner(), x = 10
// Inside main(), x = 10
```

Question 2

Control Structures in C

Control structures in C determine the flow of execution of a program. They help in decision-making, looping, and branching, enabling efficient program control. C has three main types of control structures:

1. **Sequential Control** – Executes statements one after another.
2. **Selection (Decision-Making) Control** – Executes different code blocks based on conditions (e.g., `if`, `switch`).
3. **Iteration (Looping) Control** – Repeats code blocks until a condition is met (e.g., `for`, `while`, `do-while`).

Syntax of Six Control Statements in C

1. **if Statement:**

```
if (condition) {
    // Code to execute if condition is true
}
```

Example:

```
#include <stdio.h>

int main() {
    int num = 10;

    if (num > 5) {
        printf("Number is greater than 5.\n");
    }

    return 0;
}
```

Output:

```
Number is greater than 5.
```

2. **if-else Statement:**

```
if (condition) {
    // Code if condition is true
} else {
    // Code if condition is false
}
```

Example:

```
#include <stdio.h>

int main() {
    int num = 3;

    if (num > 5) {
        printf("Number is greater than 5.\n");
    } else {
```

```
        printf("Number is not greater than 5.\n");
    }

    return 0;
}
```

Output:

```
Number is not greater than 5.
```

3. switch Statement:

```
switch (expression) {
    case value1:
        // Code to execute
        break;
    case value2:
        // Code to execute
        break;
    default:
        // Code if no cases match
}
```

Example:

```
#include <stdio.h>

int main() {
    int day = 2;

    switch (day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        default:
            printf("Invalid day\n");
    }

    return 0;
}
```

Output:

```
Tuesday
```

4. for Loop:

```
for (initialization; condition; update) {
```

```
    // Code to execute in each iteration
}
```

Example:

```
#include <stdio.h>

int main() {
    for (int i = 1; i <= 3; i++) {
        printf("Iteration %d\n", i);
    }

    return 0;
}
```

Output:

```
Iteration 1
Iteration 2
Iteration 3
```

5. **while Loop:**

```
while (condition) {
    // Code executes as long as condition is true
}
```

Example:

```
#include <stdio.h>

int main() {
    int count = 1;

    while (count <= 3) {
        printf("Count: %d\n", count);
        count++;
    }

    return 0;
}
```

Output:

```
Count: 1
Count: 2
Count: 3
```

6. **do-while Loop:**

```
do {
    // Code executes at least once
```

```
} while (condition);
```

Example:

```
#include <stdio.h>

int main() {
    int count = 1;

    do {
        printf("Count: %d\n", count);
        count++;
    } while (count <= 3);

    return 0;
}
```

Output:

```
Count: 1
Count: 2
Count: 3
```

These control structures are fundamental to controlling the execution flow in C programs, ensuring efficient and logical execution.

Question 3

Code:

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

#define ROWS 5  // Minimum 5x5 matrix
#define COLS 5
#define FILE_NAME "circle_data.txt"

// Function to calculate area and perimeter
void calculateCircle(float radius, float *area, float *perimeter)
{
    *area = M_PI * radius * radius;     // Area = πr^2
    *perimeter = 2 * M_PI * radius;     // Perimeter = 2πr
}

int main() {
    int radii[ROWS][COLS];             // 2D array to store radius
values
    float areas[ROWS][COLS];           // 2D array to store
calculated areas
```

```
    float perimeters[ROWS][COLS];      // 2D array to store
calculated perimeters
    FILE *file = fopen(FILE_NAME, "w");

    if (file == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Writing column headers with fixed spacing
    fprintf(file, "%-10s %-10s %-15s %-15s\n", "Serial No.",
"Radius", "Area", "Perimeter");

    printf("Enter %d radius values (for a %dx%d matrix):\n", ROWS
* COLS, ROWS, COLS);

    // Taking user inputs and storing in a 2D array
    int serialNo = 1;
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("Enter radius for [%d][%d]: ", i, j);
            scanf("%d", &radii[i][j]);

            // Calculate area and perimeter
            calculateCircle(radii[i][j], &areas[i][j],
&perimeters[i][j]);

            // Writing results to file with proper formatting
            fprintf(file, "%-10d %-10d %-15.2f %-15.2f\n",
serialNo++, radii[i][j], areas[i][j], perimeters[i][j]);
        }
    }

    fclose(file);

    // Printing the 2D array of radii and areas
    printf("\nStored Radius and Calculated Area Matrix:\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%-10d (%.2f)\t", radii[i][j], areas[i][j]);
        }
        printf("\n");
    }

    printf("\nStored Radius and Calculated Perimeter Matrix:\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
```

```
            printf("%-10d (%.2f)\t", radii[i][j],
perimeters[i][j]);
        }
        printf("\n");
    }

    printf("\nResults have been saved to '%s'.\n", FILE_NAME);

    return 0;
}


// Output:

/*

Enter 25 radius values (for a 5x5 matrix):
Enter radius for [0][0]: 3
Enter radius for [0][1]: 5
Enter radius for [0][2]: 7
Enter radius for [0][3]: 10
Enter radius for [0][4]: 12
Enter radius for [1][0]: 15
Enter radius for [1][1]: 18
Enter radius for [1][2]: 20
Enter radius for [1][3]: 25
Enter radius for [1][4]: 30
Enter radius for [2][0]: 35
Enter radius for [2][1]: 40
Enter radius for [2][2]: 45
Enter radius for [2][3]: 50
Enter radius for [2][4]: 55
Enter radius for [3][0]: 60
Enter radius for [3][1]: 65
Enter radius for [3][2]: 70
Enter radius for [3][3]: 75
Enter radius for [3][4]: 80
Enter radius for [4][0]: 85
Enter radius for [4][1]: 90
Enter radius for [4][2]: 95
Enter radius for [4][3]: 100
Enter radius for [4][4]: 105

Stored Radius and Calculated Area Matrix:
3          (28.27)      5          (78.54)      7
(153.94)     10         (314.16)     12         (452.39)
```

```
15          (706.86)     18          (1017.88)    20
(1256.64)    25          (1963.50)    30          (2827.43)

35          (3848.45)    40          (5026.55)    45
(6361.73)    50          (7853.98)    55          (9503.32)

60          (11309.73)   65          (13273.23)   70
(15393.80)   75          (17671.46)   80          (20106.19)

85          (22698.01)   90          (25446.90)   95
(28352.87)   100         (31415.93)   105         (34636.06)


Stored Radius and Calculated Perimeter Matrix:
3           (18.85)      5           (31.42)      7           (43.98)
10          (62.83)      12          (75.40)

15          (94.25)      18          (113.10)     20
(125.66)     25          (157.08)     30          (188.50)

35          (219.91)     40          (251.33)     45
(282.74)     50          (314.16)     55          (345.58)

60          (376.99)     65          (408.41)     70
(439.82)     75          (471.24)     80          (502.65)

85          (534.07)     90          (565.49)     95
(596.90)     100         (628.32)     105         (659.73)


Results have been saved to 'circle_data.txt'.

Process returned 0 (0x0)   execution time : 64.668 s
Press any key to continue.

*/

/*
Circle_data.txt:

Serial No. Radius     Area            Perimeter
1          3          28.27           18.85
2          5          78.54           31.42
3          7          153.94          43.98
4          10         314.16          62.83
5          12         452.39          75.40
6          15         706.86          94.25
7          18         1017.88         113.10
```

```
8           20          1256.64         125.66
9           25          1963.50         157.08
10          30          2827.43         188.50
11          35          3848.45         219.91
12          40          5026.55         251.33
13          45          6361.73         282.74
14          50          7853.98         314.16
15          55          9503.32         345.58
16          60          11309.73        376.99
17          65          13273.23        408.41
18          70          15393.80        439.82
19          75          17671.46        471.24
20          80          20106.19        502.65
21          85          22698.01        534.07
22          90          25446.90        565.49
23          95          28352.87        596.90
24          100         31415.93        628.32
25          105         34636.06        659.73
*/
```