รายงาน

Recursive

จัดทำโดย

นายธนรัก ชุ่มสวัสดิ์

รหัสศึกษา 68543210018-6

อาจารย์ผู้สอน

นายปิยพล ยืนยงสถาวร

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา Data Structures and Algorithms

สาขาวิชาวิศวกรรมซอฟต์แวร์ มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ภาคเรียนที่ 2 ปีการศึกษา 2568

Code Program Stack

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

int Factorial(int N) {
    int x, y;

    if (N == 0) {
        printf("Roll Back Point (Base Case Reached)\n");
        return 1;
    }
    else {
        x = N - 1;
        printf("%d! = %d * %d!\n", N, N, x);

        y = Factorial(x);

        printf("%d! = %d * %d = %d\n", N, N, y, N * y);
        return (N * y);
    }
}

int main() {
    int Number = 0, ans;

    printf("RECURSIVE (FACTORIAL) PROGRAM\n");
    printf("==============================\n");

    while (Number != -999) {
        printf("\nEnter Number (-999 is END): ");
        scanf("%d", &Number);

        if (Number == -999) {
            break;
        }

        if (Number >= 0) {
            printf("Formula: N! = N * (N-1)!\n");
            printf("-----------------------\n");
```

```
        ans = Factorial(Number);

        printf("------------------------\n");
        printf("Answer %d! = %d\n", Number, ans);
        printf("------ Finished ------\n");
    } else {
        printf("Error: Please enter a positive number.\n");
    }
  }

  return 0;
}
```

ส่วนที่มีการแก้ไขและเพิ่มเติม

1. แก้ไขไวยากรณ์การประกาศตัวแปรและตัวพิมพ์ใหญ่-เล็ก (Syntax & Case Sensitivity)

มีการประกาศตัวแปรผิดรูปแบบ (ขาดเครื่องหมายจุลภาค) และมีการใช้ตัวแปร X ตัวพิมพ์ใหญ่ในบรรทัดคำนวณ ทั้งที่ประกาศไว้เป็นตัวพิมพ์เล็ก x ซึ่งภาษา C ถือว่าเป็นคนละตัวแปรกัน ทำให้คอมไพล์ไม่ผ่าน โค้ดส่วนนี้จึงได้รับการแก้ไขให้ถูกต้องครับ

```
// --- Original (ในเอกสาร) ---
int Number ans;
int x,y;
X=N-1;

// --- Modified (แก้ไขแล้ว) ---
int Number, ans;
int x, y;
x = N - 1
```

2. แก้ไขคำสั่งแสดงผล printf ที่ผิดโครงสร้าง (Printf Syntax)

มีการวางตำแหน่งตัวแปรผิดและมีเครื่องหมายคอมม่าเกินมา รวมถึงวงเล็บปิดไม่ครบถ้วน โค้ดใหม่ได้ปรับปรุงให้จับคู่ตัวแปรกับ Format Specifier (%d) ให้ถูกต้อง เพื่อให้แสดงขั้นตอนการทำงาน (Trace) ได้ชัดเจน

```
// --- Original (ในเอกสาร) ---
printf( "%2d! = %2d * %2d!\n" N,N, ); //  ผิด: parameter ผิดตำแหน่งและมีคอมม่าเกิน
```

```
// --- Modified (แก้ไขแล้ว) ---
printf("%2d! = %2d * %2d!\n", N, N, x); // แก้ไข: เรียงตัวแปร N, N, x ให้ตรงกับ %d
```

3. แก้ไขเงื่อนไขและการรับค่าเริ่มต้น (Loop Condition & Initialization)

โค้ดเดิมตรวจสอบเงื่อนไข while โดยที่ตัวแปร Number ยังไม่มีค่าเริ่มต้น ซึ่งอาจเกิดข้อผิดพลาดได้ และในบรรทัดตรวจสอบค่าบวก if(Number >= 0 ขาดวงเล็บปิด โค้ดใหม่จึงกำหนดค่าเริ่มต้นและแก้ไขไวยากรณ์ให้สมบูรณ์

```
// --- Original (ในเอกสาร) ---
// Number ไม่ถูกกำหนดค่าเริ่มต้น
while(Number !=-999) {
    if(Number >= 0 {    //  ผิด: ขาดวงเล็บปิด )

// --- Modified (แก้ไขแล้ว) ---
int Number = 0;         // แก้ไข: กำหนดค่าเริ่มต้นป้องกัน Error
while (Number != -999) {
    if (Number >= 0) {  // แก้ไข: เติมวงเล็บปิดให้ครบ
```

อธิบายการทำงานของโปรแกรม

1. main() Function

ฟังก์ชันนี้จะเริ่มต้นด้วยการประกาศตัวแปรและกำหนดค่าเริ่มต้นให้ Number เป็น 0 จากนั้นจะเข้าสู่ลูป while เพื่อรอรับค่าจากผู้ใช้ เมื่อผู้ใช้ป้อนตัวเลขเข้ามา ฟังก์ชันจะตรวจสอบว่าเป็น -999 หรือไม่เพื่อจบการทำงาน หรือตรวจสอบว่าเป็นจำนวนเต็มบวกหรือไม่ หากเงื่อนไขถูกต้อง ตัวแปร Number จะถูกส่งต่อไปยังฟังก์ชัน Factorial เพื่อคำนวณ และเมื่อได้คำตอบกลับมา (เก็บใน ans) ก็จะแสดงผลลัพธ์ทางหน้าจอ

```
int main() {
    int Number = 0, ans;

    printf("RECURSIVE (FACTORIAL) PROGRAM\n");
    printf("==============================\n");

    while (Number != -999) {
        printf("\nEnter Number (-999 is END): ");
        scanf("%d", &Number);

        if (Number == -999) {
            break;
```

```
        }

        if (Number >= 0) {
            printf("Formula: N! = N * (N-1)!\n");
            printf("-----------------------\n");

            ans = Factorial(Number);

            printf("-----------------------\n");
            printf("Answer %d! = %d\n", Number, ans);
            printf("------ Finished ------\n");
        } else {
            printf("Error: Please enter a positive number.\n");
        }
    }

    return 0;
}
```

2. Factorial(int N) Function

เมื่อฟังก์ชันนี้ถูกเรียกใช้งานพร้อมกับค่า N มันจะตรวจสอบ Base Criteria ก่อนว่า N เท่ากับ 0 หรือไม่ หากใช่ จะหยุดการเรียกตัวเองและส่งค่า 1 กลับ (return 1) แต่ถ้าไม่ใช่ (กรณี Recursive Case) ฟังก์ชันจะคำนวณค่า x = N - 1 แล้วเรียกฟังก์ชันตัวเองซ้ำ (Factorial(x)) เพื่อหาคำตอบของลำดับถัดไป เมื่อฟังก์ชันลูกส่งค่าผลลัพธ์ (y) กลับมา ฟังก์ชันแม่ก็จะนำค่านั้นมาคูณกับ N ของตัวเอง แล้วส่งผลลัพธ์สุทธิกลับคืนไปยังฟังก์ชันที่เรียกมันมาตามลำดับชั้น

```
int Factorial(int N) {
    int x, y;

    if (N == 0) {
        printf("Roll Back Point (Base Case Reached)\n");
        return 1;
    }
```

```
    else {
        x = N - 1;
        printf("%d! = %d * %d!\n", N, N, x);

        y = Factorial(x);

        printf("%d! = %d * %d = %d\n", N, N, y, N * y);

        return (N * y);
    }
}
```

ผลลัพธ์ของโปรแกรม

```
RECURSIVE (FACTORIAL) PROGRAM
=============================

Enter Number (-999 is END): 10
Formula: N! = N * (N-1)!
------------------------
10! = 10 * 9!
9! = 9 * 8!
8! = 8 * 7!
7! = 7 * 6!
6! = 6 * 5!
5! = 5 * 4!
4! = 4 * 3!
3! = 3 * 2!
2! = 2 * 1!
1! = 1 * 0!
Roll Back Point (Base Case Reached)
1! = 1 * 1 = 1
2! = 2 * 1 = 2
3! = 3 * 2 = 6
4! = 4 * 6 = 24
5! = 5 * 24 = 120
6! = 6 * 120 = 720
7! = 7 * 720 = 5040
8! = 8 * 5040 = 40320
9! = 9 * 40320 = 362880
10! = 10 * 362880 = 3628800
------------------------
Answer 10! = 3628800
------ Finished ------
```