รายงาน

เรื่อง โปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายงาน

เรื่อง โปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ENGCE124 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures and Algorithms) หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่ ในระดับปริญญาตรีปีที่ 2 โดยมีจุดประสงค์ในการอธิบายโค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD รวมถึงอธิบายหลักการทำงานของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD และอธิบายผลลัพธ์การใช้งานโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้ที่สนใจ หรือนักศึกษาทุกท่าน ที่กำลังหาศึกษาในหัวข้อของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 28/08/2567

สารบัญ

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย

```
#include <stdio.h> //ใช้ printf

#include <conio.h> //ใช้ getch

#include <stdlib.h> //ใช้ random

#include <math.h> //ใช้ pow

#define MaxNode 100 // กำหนดจำนวนโหนดสูงสุดของต้นไม้

int N, data[MaxNode]; // ประกาศอาร์เรย์สำหรับเก็บข้อมูลของต้นไม้

char ch;

void CreateTreeNS(int n)

{

    int i, temp;

    for (i = 1; i <= n; i++)

    {

        temp = 1 + rand() % 99; //สุ่มตัวเลขที่แตกต่างกันในช่วง 1..99

        data[i] = temp;

    }

}

void ShowArray()

{

    int i = 1;

    while (data[i] != NULL)

    {
```

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย (ต่อ)

```
        printf("[%i]%d ", i, data[i]); i++;
    }
printf("\n============================================================
==================\n");
}
void ShowTree()
{
    int j, level, start, ends;
    j = 1;
    level = 1; //เริ่มต้นที่ระดับ 1
    printf("\n");
    while (data[j] != NULL)
    {
        start = pow(2, level) / 2; //คำนวณโหนดเริ่มต้นของระดับนี้
        ends = pow(2, level) - 1; //คำนวณโหนดสุดท้ายของระดับนี้
        for (j = start; j <= ends; j++)
        {
            if (data[j] != NULL)
            {
                switch (level)
                {
```

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย (ต่อ)

```
case 1 : printf("%40d", data[j]);
         break;
case 2 : if (j == 2)
         printf("%20d", data[j]);
       else
         printf("%40d", data[j]);
       break;
case 3 : if (j == 4)
         printf("%10d", data[j]);
       else
         printf("%20d", data[j]);
       break;
case 4 : if (j == 8)
         printf("%5d", data[j]);
       else
         printf("%10d", data[j]);
       break;
case 5 : if (j == 16)
         printf("%d", data[j]);
       else
         printf("%5d", data[j]);
```

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย (ต่อ)

```
                        break;
                }
            }
        }
        printf("\n\n"); //ขึ้นบรรทัดใหม่
        level++;
    }
}
void PreOrder(int i)
{
    int info, lson, rson;
    info = data[i]; // ข้อมูลราก
    if (info != NULL) //ถ้า INFO ไม่เป็นค่าว่าง
    {
        printf(" %d", data[i]); //แสดงข้อมูล
        lson = 2 * i; //คำนวณ LSON (โหนดลูกทางซ้าย)
        rson = 2 * i + 1; //คำนวณ RSON (โหนดลูกทางขวา)
        PreOrder(lson); //เรียกโหนดลูกทางซ้ายด้วย PreOrder
        PreOrder(rson); //เรียกโหนดลูกทางขวาด้วย PreOrder
    }
}
```

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย (ต่อ)

```
void InOrder(int i)
{
    int info, lson, rson;
    info = data[i]; // ข้อมูลราก
    if (info != NULL) //ถ้า INFO ไม่เป็นค่าว่าง
    {
        lson = 2 * i; //คำนวณ LSON
        rson = 2 * i + 1; //คำนวณ RSON
        InOrder(lson); //เรียกโหนดลูกทางซ้ายด้วย InOrder
        printf(" %d", data[i]); //แสดงข้อมูล
        InOrder(rson); //เรียกโหนดลูกทางขวาด้วย InOrder
    }
}
void PostOrder(int i)
{
    int info, lson, rson;
    info = data[i]; // ข้อมูลราก
    if (info != NULL) //ถ้า INFO ไม่เป็นค่าว่าง
    {
        lson = 2 * i; //คำนวณ LSON
        rson = 2 * i + 1; //คำนวณ RSON
```

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย (ต่อ)

```
        PostOrder(lson); //เรียกโหนดลูกทางซ้ายด้วย PostOrder
        PostOrder(rson); //เรียกโหนดลูกทางขวาด้วย PostOrder
        printf(" %d", data[i]); //แสดงข้อมูล
    }
}
int main()
{
    N = 31;
    CreateTreeNS(N); //สร้าง N โหนด
    while (ch != 'E')
    {
        printf("\nTREE (NODE SEQUENCE)\n");
        printf("==================================\n");
        ShowArray();
        ShowTree();
        printf("\nMENU => P:PreOrder I:InOrder O:PostOrder E:Exit");
        printf("\n-------------------------------------------------------------------------\n");
        ch = getch();
        switch (ch)
        {
            case 'P' :  ShowTree();
```

โค้ดของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD พร้อมคำอธิบาย (ต่อ)

```
                printf("PRE ORDER TRAVERSAL : ");
                PreOrder(1);
                printf("\n"); //ขึ้นบรรทัดใหม่
                break;
        case 'I' :  ShowTree();
                printf("IN ORDER TRAVERSAL : ");
                InOrder(1);
                printf("\n"); //ขึ้นบรรทัดใหม่
                break;
        case 'O' :  ShowTree();
                printf("POST ORDER TRAVERSAL : ");
                PostOrder(1);
                printf("\n"); //ขึ้นบรรทัดใหม่
                break;
      } //จบ Switch...case
   } //จบ While
   return(0);
} //จบ MAIN
```

หลักการทำงานของโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD

โปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD เป็นโปรแกรมในการสร้างและการท่องต้นไม้ (Tree) โดยใช้วิธีการ "NODE SEQUENCE" ที่ทำงานได้กับต้นไม้ที่มีระดับสูงสุดถึง 5 ระดับ (N=31 โหนด) ต่อไปนี้จะเป็นการอธิบายการทำงานของแต่ละฟังก์ชันในโปรแกรม รวมถึงการประกาศตัวแปรและการใช้ไลบรารีต่างๆ

1. การประกาศไลบรารีของโปรแกรม

```
#include <stdio.h> // ใช้ printf

#include <conio.h> // ใช้ getch

#include <stdlib.h> // ใช้ malloc

#define HeadInfo -999 // กำหนดข้อมูลของโหนดหัว (Head Node)
```

ในส่วนของการประกาศไลบรารีของโปรแกรม มีรายละเอียดดังต่อไปนี้

- #include <stdio.h>: ใช้สำหรับการทำงานกับการป้อนข้อมูลและการแสดงผล เช่น printf สำหรับพิมพ์ข้อความออกทางหน้าจอ
- #include <conio.h>: ใช้สำหรับการทำงานกับการรับข้อมูลจากแป้นพิมพ์ เช่น getch ซึ่งรับค่าตัวอักษรจากผู้ใช้โดยไม่ต้องกด Enter
- #include <stdlib.h>: ใช้สำหรับการจัดการกับฟังก์ชันที่เกี่ยวข้องกับหน่วยความจำและการสุ่ม เช่น rand สำหรับการสุ่มตัวเลข
- #include <math.h>: ใช้สำหรับการคำนวณทางคณิตศาสตร์ เช่น pow สำหรับการยกกำลัง

2. การประกาศตัวแปรต่าง ๆ ในโปรแกรม

```
#define MaxNode 100 // กำหนดจำนวนโหนดสูงสุดของต้นไม้

int N, data[MaxNode]; // ประกาศอาร์เรย์สำหรับเก็บข้อมูลของต้นไม้

char ch;
```

ในส่วนของการประกาศตัวแปรต่าง ๆ ของโปรแกรม มีรายละเอียดดังต่อไปนี้

2. การประกาศตัวแปรต่าง ๆ ในโปรแกรม (ต่อ)

- #define MaxNode 100 : กำหนดค่าคงที่ MaxNode เป็น 100 ซึ่งเป็นจำนวนสูงสุดของโหนดที่สามารถจัดการได้
- int N, data[MaxNode]; :
    - N: จำนวนโหนดในต้นไม้
    - data: อาร์เรย์ที่ใช้เก็บข้อมูลของโหนดในต้นไม้
- char ch; : ตัวแปรที่ใช้เก็บค่าของตัวอักษรที่ผู้ใช้ป้อนเพื่อเลือกเมนู

3. การสร้างต้นไม้ด้วยฟังก์ชัน CreateTreeNS

```
void CreateTreeNS(int n)
{
    int i, temp;
    for (i = 1; i <= n; i++)
    {
        temp = 1 + rand() % 99; //สุ่มตัวเลขที่แตกต่างกันในช่วง 1..99
        data[i] = temp;
    }
}
```

ฟังก์ชันนี้จะสร้างต้นไม้โดยการกำหนดค่าข้อมูลให้กับแต่ละโหนดในอาร์เรย์ data ด้วยค่าที่สุ่ม โดยการทำงานจะลูปจะทำการสุ่มตัวเลขและเก็บไว้ในอาร์เรย์ data ตั้งแต่ตำแหน่ง 1 ถึง n

4. การแสดงข้อมูลในอาร์เรย์ด้วยฟังก์ชัน ShowArray

```
void ShowArray()
{
    int i = 1;
```

4. การแสดงข้อมูลในอาร์เรย์ด้วยฟังก์ชัน ShowArray (ต่อ)

```
    while (data[i] != NULL)
    {
        printf("[%i]%d ", i, data[i]); i++;
    }
printf("\n============================================================
==================\n");
}
```

ฟังก์ชันนี้จะแสดงข้อมูลในอาร์เรย์ data ซึ่งเก็บค่าของโหนดในต้นไม้ โดยการทำงานจะใช้ลูป while เพื่อแสดงข้อมูลของโหนดในอาร์เรย์จนกว่าจะถึงตำแหน่งที่มีค่าเป็น NULL

5. การแสดงรูปแบบของต้นไม้ด้วยฟังก์ชัน ShowTree

```
void ShowTree()
{
    int j, level, start, ends;
    j = 1;
    level = 1; //เริ่มต้นที่ระดับ 1
    printf("\n");
    while (data[j] != NULL)
    {
        start = pow(2, level) / 2; //คำนวณโหนดเริ่มต้นของระดับนี้
        ends = pow(2, level) - 1; //คำนวณโหนดสุดท้ายของระดับนี้
        for (j = start; j <= ends; j++)
        {
```

5. การแสดงรูปแบบของต้นไม้ด้วยฟังก์ชัน ShowTree (ต่อ)

```
        if (data[j] != NULL)
        {
            switch (level)
            {
                case 1 : printf("%40d", data[j]);
                         break;
                case 2 : if (j == 2)
                         printf("%20d", data[j]);
                       else
                         printf("%40d", data[j]);
                       break;
                case 3 : if (j == 4)
                         printf("%10d", data[j]);
                       else
                         printf("%20d", data[j]);
                       break;
                case 4 : if (j == 8)
                         printf("%5d", data[j]);
                       else
                         printf("%10d", data[j]);
                       break;
```

5. การแสดงรูปแบบของต้นไม้ด้วยฟังก์ชัน ShowTree (ต่อ)

```
                    case 5 : if (j == 16)
                                printf("%d", data[j]);
                             else
                                printf("%5d", data[j]);
                             break;
                }
            }
        }
        printf("\n\n"); //ขึ้นบรรทัดใหม่
        level++;
    }
}
```

ฟังก์ชันนี้จะแสดงรูปแบบของต้นไม้ในรูปแบบที่อ่านง่าย โดยการจัดรูปแบบตามระดับของต้นไม้ โดยการทำงานจะใช้ pow เพื่อคำนวณตำแหน่งเริ่มต้นและตำแหน่งสุดท้ายของโหนดในแต่ละระดับ โดยจะจัดรูปแบบการแสดงผลตามระดับของต้นไม้

6. การท่องต้นไม้ในลำดับ Pre Order ด้วยฟังก์ชัน PreOrder

```
void PreOrder(int i)
{
    int info, lson, rson;
    info = data[i]; // ข้อมูลราก
    if (info != NULL) //ถ้า INFO ไม่เป็นค่าว่าง
```

6. การท่องต้นไม้ในลำดับ Pre Order ด้วยฟังก์ชัน PreOrder (ต่อ)

```
    {
        printf(" %d", data[i]); //แสดงข้อมูล
        lson = 2 * i; //คำนวณ LSON (โหนดลูกทางซ้าย)
        rson = 2 * i + 1; //คำนวณ RSON (โหนดลูกทางขวา)
        PreOrder(lson); //เรียกโหนดลูกทางซ้ายด้วย PreOrder
        PreOrder(rson); //เรียกโหนดลูกทางขวาด้วย PreOrder
    }
}
```

ฟังก์ชันนี้จะท่องต้นไม้ในลำดับ Pre Order (Root -> Left -> Right) โดยการทำงานจะแสดงข้อมูลของโหนดปัจจุบัน และเรียกใช้งาน PreOrder สำหรับโหนดลูกทางซ้ายและขวาตามลำดับ

7. การท่องต้นไม้ในลำดับ InOrder ด้วยฟังก์ชัน InOrder

```
void InOrder(int i)
{
    int info, lson, rson;
    info = data[i]; // ข้อมูลราก
    if (info != NULL) //ถ้า INFO ไม่เป็นค่าว่าง
    {
        lson = 2 * i; //คำนวณ LSON
        rson = 2 * i + 1; //คำนวณ RSON
        InOrder(lson); //เรียกโหนดลูกทางซ้ายด้วย InOrder
        printf(" %d", data[i]); //แสดงข้อมูล
```

7. การท่องต้นไม้ในลำดับ InOrder ด้วยฟังก์ชัน InOrder (ต่อ)

```
        InOrder(rson); //เรียกโหนดลูกทางขวาด้วย InOrder
    }
}
```

ฟังก์ชันนี้จะท่องต้นไม้ในลำดับ In Order (Left -> Root -> Right) โดยการทำงานจะเรียกใช้งาน InOrder สำหรับโหนดลูกทางซ้ายก่อน จากนั้นจะแสดงข้อมูลของโหนดปัจจุบัน และเรียกใช้งาน InOrder สำหรับโหนดลูกทางขวา

8. การท่องต้นไม้ในลำดับ PostOrder ด้วยฟังก์ชัน PostOrder

```
void PostOrder(int i)
{
    int info, lson, rson;
    info = data[i]; // ข้อมูลราก
    if (info != NULL) //ถ้า INFO ไม่เป็นค่าว่าง
    {
        lson = 2 * i; //คำนวณ LSON
        rson = 2 * i + 1; //คำนวณ RSON
        PostOrder(lson); //เรียกโหนดลูกทางซ้ายด้วย PostOrder
        PostOrder(rson); //เรียกโหนดลูกทางขวาด้วย PostOrder
        printf(" %d", data[i]); //แสดงข้อมูล
    }
}
```

8. การท่องต้นไม้ในลำดับ PostOrder ด้วยฟังก์ชัน PostOrder (ต่อ)

ฟังก์ชันนี้จะท่องต้นไม้ในลำดับ Post Order (Left -> Right -> Root) โดยการทำงานจะเรียกใช้งาน PostOrder สำหรับโหนดลูกทางซ้ายและขวาก่อน และแสดงข้อมูลของโหนดปัจจุบัน

9. ฟังก์ชันหลัก (Main)

```
int main()
{
    N = 31;
    CreateTreeNS(N); //สร้าง N โหนด
    while (ch != 'E')
    {
        printf("\nTREE (NODE SEQUENCE)\n");
        printf("==================================\n");
        ShowArray();
        ShowTree();
        printf("\nMENU => P:PreOrder I:InOrder O:PostOrder E:Exit");
        printf("\n--------------------------------------------------------------------\n");
        ch = getch();
        switch (ch)
        {
            case 'P' :  ShowTree();
                        printf("PRE ORDER TRAVERSAL : ");
                        PreOrder(1);
```

9. ฟังก์ชันหลัก (Main) (ต่อ)

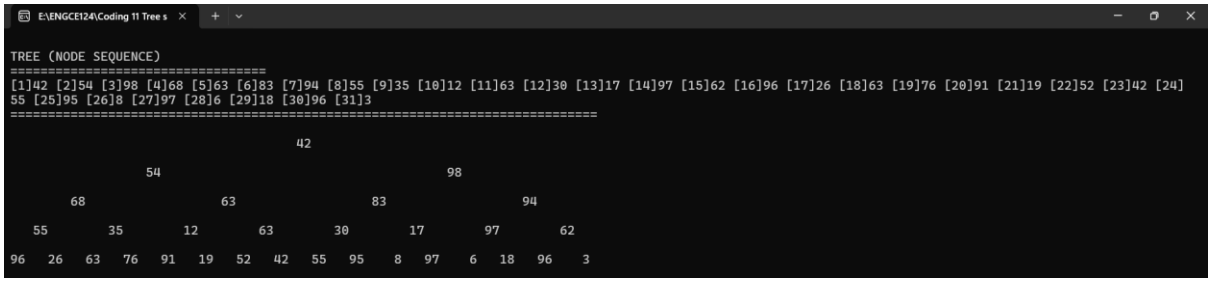
```
                    printf("\n"); //ขึ้นบรรทัดใหม่
                    break;
            case 'I' :  ShowTree();
                    printf("IN ORDER TRAVERSAL : ");
                    InOrder(1);
                    printf("\n"); //ขึ้นบรรทัดใหม่
                    break;
            case 'O' :  ShowTree();
                    printf("POST ORDER TRAVERSAL : ");
                    PostOrder(1);
                    printf("\n"); //ขึ้นบรรทัดใหม่
                    break;
        } //จบ Switch...case
    } //จบ While
    return(0);
} //จบ MAIN
```

ฟังก์ชันหลักของโปรแกรมที่สร้างต้นไม้และให้ผู้ใช้เลือกการท่องต้นไม้ โดยการทำงานจะสร้างต้นไม้ที่มี 31 โหนด ซึ่งใช้ลูป while เพื่อให้ผู้ใช้สามารถเลือกการท่องต้นไม้ (PreOrder, InOrder, PostOrder) หรือออกจากโปรแกรม สุดท้ายจะแสดงผลลัพธ์ของการท่องต้นไม้ตามที่ผู้ใช้เลือก

ผลลัพธ์การใช้งานโปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD

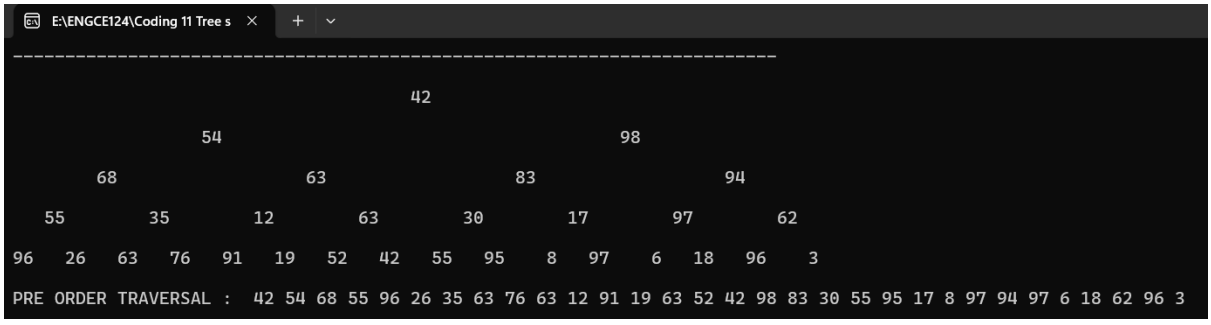
โปรแกรม TREE STRUCTURE BY NODE SEQUENCE METHOD ถูกออกแบบมาเพื่อสร้างต้นไม้และแสดงผลการท่องต้นไม้โดยใช้วิธีการ "NODE SEQUENCE" โปรแกรมสามารถสร้างต้นไม้ที่มีโหนดสูงสุดถึง 31 โหนด (5 ระดับ) และแสดงผลการท่องต้นไม้ในลำดับ Pre Order, In Order, และ Post Order ตามที่ผู้ใช้เลือก ต่อไปนี้เป็นการอธิบายผลลัพธ์ที่เกิดจากการใช้งานโปรแกรม

1. การสร้างต้นไม้ และการแสดงข้อมูลโหนด



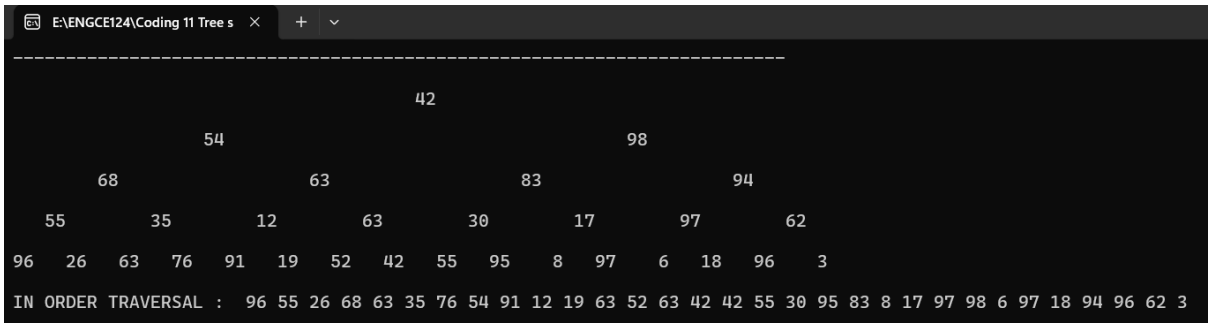
เมื่อเริ่มต้นโปรแกรม ฟังก์ชัน CreateTreeNS จะถูกเรียกใช้เพื่อสร้างต้นไม้โดยการกำหนดค่าข้อมูลให้กับแต่ละโหนดในอาร์เรย์ data ด้วยค่าที่สุ่มจาก 1 ถึง 99 จากนั้นฟังก์ชัน ShowArray จะแสดงข้อมูลในอาร์เรย์ data ซึ่งเก็บค่าของโหนดที่สร้างขึ้น ในที่นี้ [i] แทนตำแหน่งของโหนดในอาร์เรย์ และตัวเลขหลัง [i] คือค่าที่สุ่มมา เช่น [1]42 , [2]54 เป็นต้น และสุดท้ายฟังก์ชัน ShowTree จะแสดงรูปแบบของต้นไม้ในลักษณะที่อ่านง่าย โดยการจัดรูปแบบตามระดับของต้นไม้ ซึ่งในที่นี้ การจัดรูปแบบจะแสดงต้นไม้ในลักษณะระดับ โดยโหนดรากจะอยู่ในตำแหน่งกลางของระดับแรก และโหนดลูกแต่ละระดับจะแสดงตามรูปแบบที่กำหนด

2. การท่องต้นไม้ตามลำดับ Pre Order



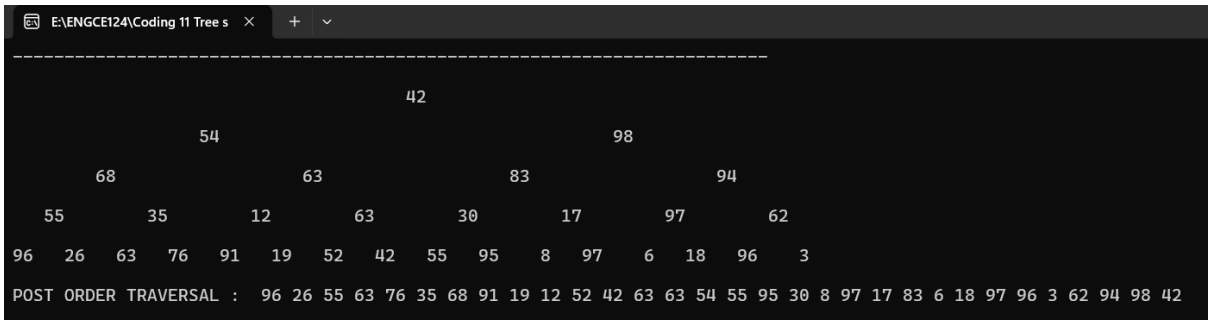
เมื่อเลือกการท่องต้นไม้แบบ Pre Order (Root -> Left -> Right) โดยการป้อนตัวอักษร 'P' ฟังก์ชัน PreOrder จะถูกเรียกใช้และจะแสดงข้อมูลของโหนดในลำดับ Pre Order ในที่นี้ การท่องต้นไม้เริ่มต้นจากโหนดราก (42) และทำการเรียกซ้ำสำหรับโหนดลูกทางซ้ายและขวาตามลำดับ

3. การท่องต้นไม้ตามลำดับ In Order



เมื่อเลือกการท่องต้นไม้แบบ In Order (Left -> Root -> Right) โดยการป้อนตัวอักษร 'I', ฟังก์ชัน InOrder จะถูกเรียกใช้และจะแสดงข้อมูลของโหนดในลำดับ In Order ในที่นี้ การท่องต้นไม้เริ่มต้นจากโหนดลูกทางซ้ายสุด (96) และทำการเรียกซ้ำสำหรับโหนดลูกทางขวาและโหนดรากตามลำดับ

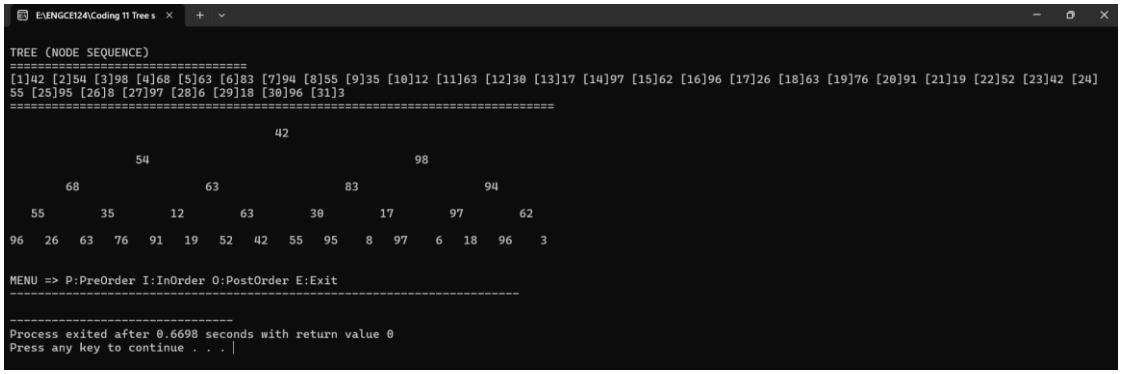
4. การท่องต้นไม้ตามลำดับ Post Order



เมื่อเลือกการท่องต้นไม้แบบ Post Order (Left -> Right -> Root) โดยการป้อนตัวอักษร 'O', ฟังก์ชัน PostOrder จะถูกเรียกใช้และจะแสดงข้อมูลของโหนดในลำดับ Post Order ในที่นี้ การท่องต้นไม้เริ่มต้นจากโหนดลูกทางซ้ายสุดและโหนดลูกทางขวา ก่อนที่จะถึงโหนดราก

5. ออกจากโปรแกรม

โปรแกรมจะทำงานต่อไปเรื่อยๆ จนกว่าผู้ใช้จะเลือกออกจากโปรแกรมโดยการป้อนตัวอักษร 'E'. เมื่อต้องการออกจากโปรแกรม, ฟังก์ชันหลักจะหยุดการทำงานและโปรแกรมจะปิดตัวลง.



บรรณานุกรม

ChatGPT. (-). Tree Traversal Methods of a Tree Structure Program in C. สืบค้น 28 สิงหาคม 2567, จาก https://chatgpt.com/