รายงาน

เรื่อง โปรแกรม SINGLY CIRCULAR LINKED LIST

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 SINGLY CIRCULAR LINKED LIST

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 SINGLY CIRCULAR LINKED LIST รวมถึงอธิบายหลักการทำงานของโปรแกรม SINGLY CIRCULAR LINKED LIST และอธิบายผลลัพธ์การใช้งานโปรแกรม SINGLY CIRCULAR LINKED LIST

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้ที่สนใจ หรือนักศึกษาทุกท่าน ที่กำลังหาศึกษาในหัวข้อของโปรแกรม SINGLY CIRCULAR LINKED LIST หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 28/08/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	ข
โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม SINGLY CIRCULAR LINKED LIST	8
ผลลัพธ์การใช้งานโปรแกรม SINGLY CIRCULAR LINKED LIST	17
บรรณานุกรม	20

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย

```
#include <stdio.h> //ใช้ printf

#include <conio.h> //ใช้ getch

#include <stdlib.h> //ใช้ malloc

#define HeadData -999 //ข้อมูลพิเศษของโหนดหัว

struct Node //ประกาศโครงสร้างของโหนด

{

    int info;

    struct Node *link;

};

struct Node *H, *H1, *p, *q; //ประกาศพอยเตอร์โหนด

int i,j,k,n,data;

char ch;

Node *Allocate() //จัดสรร 1 โหนดจากพื้นที่เก็บ

{

    struct Node *temp;

    temp=(Node*)malloc(sizeof(Node)); //จัดสรรโหนดตามขนาดที่ประกาศ

    return(temp);

}

void CreateNNode(int n) //สร้าง N โหนด ใส่ข้อมูลและเชื่อมโยงมัน

{
```

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย (ต่อ)

```
    int i,temp;
    H=p;H1=p;
    for (i=1;i<=n;i++) //นับจำนวน N โหนด
    {
        p=Allocate(); //จัดสรรโหนด
        temp=1+rand() % 99; //สุ่มเลขต่าง ๆ ตั้งแต่ 1..99
        p->info=temp; //ใส่ข้อมูลที่สุ่มในโหนด
        H1->link=p; //ให้โหนดสุดท้ายชี้ไปที่โหนดใหม่
        H1=p; //ให้ H1 ชี้ไปที่โหนดใหม่
        H1->link=H; //ตั้งลิงก์ของ H1 ให้ชี้ไปที่โหนดหัวเพื่อสร้างลิสต์แบบวงกลม
    }
}
void ShowAllNode()
{
    printf("H = %x\n",H); //แสดงที่อยู่ของพอยเตอร์ H
    p=H->link; //ตั้งจุดเริ่มต้นของพอยเตอร์ p ที่โหนดแรก
    i=1; //ตั้งค่าเริ่มต้นของตัวนับ
    while (p->info != HeadData) //ตราบใดที่ข้อมูล (INFO) ไม่ใช่ข้อมูลของโหนดหัว
    {
        printf("%d) : %x\t",i,p); //แสดงตัวนับและพอยเตอร์
        printf("INFO : %d\t",p->info); //แสดงข้อมูล (INFO)
```

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย (ต่อ)

```
        printf("LINK : %x\n",p->link); //แสดงลิงก์ (LINK)
        p=p->link; //ข้ามไปโหนดถัดไป
        i++; //เพิ่มตัวนับ
    } //สิ้นสุด While
} //สิ้นสุดฟังก์ชัน
void InsertAfter(int data1)
{
    int temp; //ตัวแปรชั่วคราว
    if (H->link == H) //ถ้าลิงก์ชี้กลับไปที่โหนดหัว
        printf("Circular Linked List have no node!!..\n");
    else
    {
        H1=H->link; //ให้ H1 ชี้ไปที่โหนดแรก
        while (H1->info != HeadData) //ค้นหาข้อมูลตราบใดที่ข้อมูลไม่เท่ากับข้อมูลโหนดหัว
        {
            if (H1->info == data1) //ถ้าพบข้อมูล
            {
                p=Allocate(); //จัดสรรหนึ่งโหนดจากพื้นที่เก็บ
                printf("\nใส่ข้อมูล : " ); //รับข้อมูลสำหรับการแทรก
                scanf("%d",&temp); //อ่านข้อมูลจากคีย์บอร์ด
                p->info=temp; //นำข้อมูลชั่วคราวใส่ในข้อมูลของโหนด
```

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย (ต่อ)

```
                p->link=H1->link; //เปลี่ยนลิงก์ที่ 1 สำหรับการแทรกโหนด (ไกล)
                H1->link=p; //เปลี่ยนลิงก์ที่ 2 สำหรับการแทรกโหนด (ใกล้)
            } //สิ้นสุด if
            H1=H1->link; //ข้าม H1 ไปที่โหนดถัดไป
        } //สิ้นสุด while
    } //สิ้นสุด IF
} //สิ้นสุดฟังก์ชัน
void DeleteAfter(int data1)
{
    int temp; //ตัวแปรชั่วคราว
    if (H->link == H) //ถ้าลิงก์ชี้กลับไปที่โหนดหัว
        printf("Circular Linked List have no node!!..\n");
    else
    {
        H1=H->link; //ให้ H1 ชี้ไปที่โหนดเริ่มต้น
        while (H1->info != HeadData) //ค้นหาข้อมูลตราบใดที่ข้อมูลไม่เท่ากับข้อมูลโหนดหัว
        {
            if (H1->info == data1) //ถ้าพบข้อมูล
            {
                if (H1->link==H) //ถ้าไม่มีโหนดอีกต่อไป
                    printf ("This is the HEAD Node,Can't delete it!!!\n");
```

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย (ต่อ)

```
                else
                {
                    p=H1->link; //ทำเครื่องหมายที่โหนดสำหรับลบ
                    if(p->link==H) //ถ้า p เป็นโหนดสุดท้าย
                        H1->link=H; //ตั้งลิงก์ของ H1 ให้ชี้ไปที่โหนดหัว
                    else
                        H1->link=p->link; //ถ้าไม่ ให้ลิงก์ของ H1 ชี้ไปยังที่อยู่เดียวกันกับ p
                    free(p); //ปล่อยโหนดกลับไปยังพื้นที่เก็บ
                } //สิ้นสุด if2
            } //สิ้นสุด if1
            H1=H1->link; //ข้าม H1 ไปที่โหนดถัดไป
        } //สิ้นสุด while
    } //สิ้นสุด IF
} //สิ้นสุดฟังก์ชัน
int main() //ฟังก์ชันหลัก
{
    p=Allocate(); //สร้างโหนดหัว
    p->info=HeadData; //กำหนดข้อมูลพิเศษ
    p->link=p; //ลิงก์กลับไปที่โหนด
    n=10; //ตั้งจำนวนโหนด
    CreateNNode(n); //เรียกฟังก์ชันสร้างโหนด N
```

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย (ต่อ)

```
printf("PROGRAM SINGLY CIRCULAR LINKED LIST \n");
printf("========================================= \n");
printf("All Data in Linked List \n");
ShowAllNode(); //เรียกฟังก์ชันแสดงโหนดทั้งหมด
ch=' ';
while (ch != 'E')
{
    printf("MENU : [I:Insert D:Delete E:Exit]");
    ch=getch();
    switch (ch)
    {
        case 'I' : printf("\nInsert After data : " ); //รับข้อมูลสำหรับแทรกหลัง
            scanf("%d",&data);
            InsertAfter(data); //เรียกฟังก์ชันแทรกหลังข้อมูล
            printf("\nข้อมูลทั้งหมดในลิสต์หลังจากแทรก\n");
            ShowAllNode(); //เรียกฟังก์ชันแสดงโหนดทั้งหมด
            break;
        case 'D' : printf("\nDelete After data : " ); //รับข้อมูลสำหรับลบหลัง
            scanf("%d",&data);
            DeleteAfter(data); //เรียกฟังก์ชันลบหลังข้อมูล
            printf("\nAll Data in Linked List AFTER DELETED\n");
```

โค้ดของโปรแกรม SINGLY CIRCULAR LINKED LIST พร้อมคำอธิบาย (ต่อ)

```
                ShowAllNode(); //เรียกฟังก์ชันแสดงโหนดทั้งหมด

                break;

        } // สิ้นสุด Switch...case

    } //สิ้นสุด While

    return(0);

} //สิ้นสุดฟังก์ชันหลัก
```

หลักการทำงานของโปรแกรม SINGLY CIRCULAR LINKED LIST

โปรแกรมนี้เป็นการสร้างและจัดการ Singly Circular Linked List (ลิสต์แบบวงกลมเชื่อมต่อทางเดียว) ซึ่งเป็นโครงสร้างข้อมูลที่มีลักษณะเฉพาะคือ โหนดสุดท้ายของลิสต์จะชี้กลับไปที่โหนดหัว ทำให้เกิดการเชื่อมต่อเป็นวงกลม โดยในโปรแกรมนี้เราสามารถเพิ่มข้อมูลใหม่ในลิสต์และลบข้อมูลออกได้ โดยมีข้อแม้คือไม่สามารถลบโหนดหัวได้ นอกจากนี้โปรแกรมยังสามารถแสดงที่อยู่ของโหนดแต่ละโหนดในลิสต์ได้ด้วย

1. การประกาศไลบรารีของโปรแกรม

```
#include <stdio.h> //ใช้ printf

#include <conio.h> //ใช้ getch

#include <stdlib.h> //ใช้ malloc

#define HeadData -999 //ข้อมูลพิเศษของโหนดหัว
```

ในส่วนของการประกาศไลบรารีของโปรแกรม มีรายละเอียดดังต่อไปนี้

- #include <stdio.h> : โปรแกรมนี้ใช้ไลบรารีมาตรฐาน stdio.h ซึ่งมีฟังก์ชันสำหรับการรับและแสดงผลข้อมูลในโปรแกรม ตัวอย่างเช่น การใช้ฟังก์ชัน printf เพื่อแสดงข้อมูลต่าง ๆ ออกทางหน้าจอ
- #include <conio.h> : ฟังก์ชัน conio.h เป็นไลบรารีที่มีฟังก์ชันที่เกี่ยวข้องกับการจัดการอินพุตและเอาต์พุตของคอนโซล โดยโปรแกรมนี้ใช้ฟังก์ชัน getch จากไลบรารีนี้ ฟังก์ชัน getch ใช้สำหรับรับอักขระจากผู้ใช้หนึ่งตัวโดยไม่ต้องรอการกด Enter และอักขระที่รับเข้ามาจะไม่แสดงบนหน้าจอ
- #include <stdlib.h> : ฟังก์ชัน stdlib.h เป็นไลบรารีมาตรฐานที่ใช้สำหรับการจัดการหน่วยความจำ, การแปลงข้อมูล, การสุ่มเลข, และฟังก์ชันยูทิลิตีอื่น ๆ โปรแกรมนี้ใช้ malloc จาก stdlib.h ซึ่งเป็นฟังก์ชันที่ใช้จัดสรรหน่วยความจำให้กับโหนดใหม่ในลิสต์
- #define HeadData -999 : การประกาศ #define เป็นการประกาศมาโครที่กำหนดค่าคงที่ให้กับชื่อในโปรแกรม ในโปรแกรมนี้ HeadData ถูกกำหนดเป็น -999 เพื่อใช้เป็นข้อมูลพิเศษสำหรับโหนดหัว (Head Node) โดยค่า -999 นี้ถูกใช้เพื่อบ่งบอกว่าโหนดนั้นเป็นโหนดหัว ซึ่งจะช่วยในการตรวจสอบว่าโหนดใดเป็นโหนดหัว และไม่อนุญาตให้ลบโหนดนี้

หลักการทำงานของโปรแกรม SINGLY CIRCULAR LINKED LIST (ต่อ)

2. การประกาศโครงสร้างของโหนด

```
struct Node //ประกาศโครงสร้างของโหนด
{
    int info;
    struct Node *link;
};
```

โครงสร้างของโหนด (struct Node) ประกอบด้วยสองส่วนหลัก ๆ คือ

- info: เก็บข้อมูลที่อยู่ในโหนด
- link: พอยน์เตอร์ที่ชี้ไปยังโหนดถัดไปในลิสต์

3. การประกาศพอยน์เตอร์โหนดและตัวแปรต่าง ๆ

```
struct Node *H, *H1, *p, *q; //ประกาศพอยเตอร์โหนด
int i,j,k,n,data;
char ch;
```

ในส่วนของการประกาศพอยน์เตอร์โหนดและตัวแปรต่าง ๆ มีรายละเอียดดังต่อไปนี้

- การประกาศพอยน์เตอร์โหนด
    - struct Node *H: พอยน์เตอร์ H ถูกใช้เพื่อเป็นพอยน์เตอร์ไปยังโหนดหัว (Head Node) ในลิสต์
    - struct Node *H1: พอยน์เตอร์ H1 ถูกใช้เป็นพอยน์เตอร์ช่วยในการเดินทางผ่านโหนดต่าง ๆ ในลิสต์ เพื่อทำการแทรกหรือลบโหนด
    - struct Node *p: พอยน์เตอร์ p ถูกใช้เป็นพอยน์เตอร์ชั่วคราวสำหรับโหนดปัจจุบันที่กำลังถูกสร้างหรือลบ
    - struct Node *q: พอยน์เตอร์ q อาจใช้ในส่วนอื่นของโปรแกรมสำหรับการจัดการโหนด (แต่ในโค้ดที่ให้มา q ยังไม่ถูกใช้งานจริง)

- การประกาศตัวแปรอื่น ๆ
  - int i, j, k, n, data: ตัวแปรประเภท int เหล่านี้ใช้สำหรับเก็บข้อมูลตัวเลขทั่วไปในโปรแกรม:
    - i, j, k: ใช้เป็นตัวนับในลูปต่าง ๆ
    - n: ใช้เก็บจำนวนของโหนดที่ต้องการสร้างในลิสต์
    - data: ใช้เก็บข้อมูลที่ผู้ใช้ป้อนเข้ามาสำหรับการแทรกหรือลบข้อมูลในลิสต์
  - char ch: ตัวแปรประเภท char ใช้เก็บตัวอักษรที่ผู้ใช้กดในเมนูเพื่อเลือกว่าจะทำการแทรก (Insert) ลบ (Delete) หรือออกจากโปรแกรม (Exit)

4. การจองพื้นที่หน่วยความจำด้วยฟังก์ชัน Allocate

```
Node *Allocate() //จัดสรร 1 โหนดจากพื้นที่เก็บ
{
    struct Node *temp;
    temp=(Node*)malloc(sizeof(Node)); //จัดสรรโหนดตามขนาดที่ประกาศ
    return(temp);
}
```

ฟังก์ชันนี้ใช้คำสั่ง malloc เพื่อจัดสรรหน่วยความจำสำหรับโหนดใหม่ในหน่วยความจำ และส่งคืนพอยน์เตอร์ไปยังโหนดนั้น

5. การสร้างโหนดหลายโหนดด้วยฟังก์ชัน CreateNode

```
void CreateNNode(int n) //สร้าง N โหนด ใส่ข้อมูลและเชื่อมโยงมัน
{
    int i,temp;
    H=p;H1=p;
    for (i=1;i<=n;i++) //นับจำนวน N โหนด
    {
```

5. การสร้างโหนดหลายโหนดด้วยฟังก์ชัน CreateNode (ต่อ)

```
        p=Allocate(); //จัดสรรโหนด
        temp=1+rand() % 99; //สุ่มเลขต่าง ๆ ตั้งแต่ 1..99
        p->info=temp; //ใส่ข้อมูลที่สุ่มในโหนด
        H1->link=p; //ให้โหนดสุดท้ายชี้ไปที่โหนดใหม่
        H1=p; //ให้ H1 ชี้ไปที่โหนดใหม่
        H1->link=H; //ตั้งลิงก์ของ H1 ให้ชี้ไปที่โหนดหัวเพื่อสร้างลิสต์แบบวงกลม
    }
}
```

ฟังก์ชันนี้ทำหน้าที่สร้างโหนด n โหนด และเชื่อมโยงโหนดเหล่านั้นเข้าด้วยกันเพื่อสร้างลิสต์วงกลม โดยแต่ละโหนดจะถูกจัดสรรหน่วยความจำใหม่ด้วยฟังก์ชัน Allocate และถูกสุ่มข้อมูลที่มีค่าอยู่ระหว่าง 1 ถึง 99 จากนั้นโหนดสุดท้ายจะถูกเชื่อมโยงกลับไปยังโหนดหัว (H) ทำให้โครงสร้างข้อมูลเป็นวงกลม

6. การแสดงข้อมูลโหนดทั้งหมดในลิสต์ด้วยฟังก์ชัน ShowAllNode

```
void ShowAllNode()
{
    printf("H = %x\n",H); //แสดงที่อยู่ของพอยเตอร์ H
    p=H->link; //ตั้งจุดเริ่มต้นของพอยเตอร์ p ที่โหนดแรก
    i=1; //ตั้งค่าเริ่มต้นของตัวนับ
    while (p->info != HeadData) //ตราบใดที่ข้อมูล (INFO) ไม่ใช่ข้อมูลของโหนดหัว
    {
        printf("%d) : %x\t",i,p); //แสดงตัวนับและพอยเตอร์
        printf("INFO : %d\t",p->info); //แสดงข้อมูล (INFO)
```

6. การแสดงข้อมูลโหนดทั้งหมดในลิสต์ด้วยฟังก์ชัน ShowAllNode (ต่อ)

```
        printf("LINK : %x\n",p->link); //แสดงลิงก์ (LINK)

        p=p->link; //ข้ามไปโหนดถัดไป

        i++; //เพิ่มตัวนับ

    } //สิ้นสุด While

} //สิ้นสุดฟังก์ชัน
```

ฟังก์ชันนี้แสดงข้อมูลโหนดทั้งหมดในลิสต์แบบวงกลม โดยเริ่มจากโหนดแรกหลังโหนดหัวและแสดงที่อยู่ (p), ข้อมูล (p->info), และลิงก์ (p->link) ของแต่ละโหนด

7. การเพิ่มข้อมูลใหม่หลังโหนดที่กำหนดด้วยฟังก์ชัน InsertAfter

```
void InsertAfter(int data1)

{

    int temp; //ตัวแปรชั่วคราว

    if (H->link == H) //ถ้าลิงก์ชี้กลับไปที่โหนดหัว

        printf("Circular Linked List have no node!!..\n");

    else

    {

        H1=H->link; //ให้ H1 ชี้ไปที่โหนดแรก

        while (H1->info != HeadData) //ค้นหาข้อมูลตราบใดที่ข้อมูลไม่เท่ากับข้อมูลโหนดหัว

        {

            if (H1->info == data1) //ถ้าพบข้อมูล

            {

                p=Allocate(); //จัดสรรหนึ่งโหนดจากพื้นที่เก็บ
```

7. การเพิ่มข้อมูลใหม่หลังโหนดที่กำหนดด้วยฟังก์ชัน InsertAfter (ต่อ)

```
                printf("\nใส่ข้อมูล : " ); //รับข้อมูลสำหรับการแทรก
                scanf("%d",&temp); //อ่านข้อมูลจากคีย์บอร์ด
                p->info=temp; //นำข้อมูลชั่วคราวใส่ในข้อมูลของโหนด
                p->link=H1->link; //เปลี่ยนลิงก์ที่ 1 สำหรับการแทรกโหนด (ไกล)
                H1->link=p; //เปลี่ยนลิงก์ที่ 2 สำหรับการแทรกโหนด (ใกล้)
            } //สิ้นสุด if
            H1=H1->link; //ข้าม H1 ไปที่โหนดถัดไป
        } //สิ้นสุด while
    } //สิ้นสุด IF
} //สิ้นสุดฟังก์ชัน
```

ฟังก์ชันนี้ทำหน้าที่เพิ่มโหนดใหม่ลงในลิสต์หลังจากโหนดที่มีข้อมูลตรงกับค่าที่กำหนด (data1) โดยข้อมูลใหม่จะถูกนำเข้าโดยผู้ใช้ผ่านคีย์บอร์ด โหนดใหม่จะถูกเชื่อมโยงเข้าลิสต์โดยการปรับเปลี่ยนลิงก์ของโหนดก่อนหน้าและโหนดใหม่

8. การลบโหนดหลังจากโหนดที่กำหนดด้วยฟังก์ชัน DeleteAfter

```
void DeleteAfter(int data1)
{
    int temp; //ตัวแปรชั่วคราว
    if (H->link == H) //ถ้าลิงก์ชี้กลับไปที่โหนดหัว
        printf("Circular Linked List have no node!!..\n");
    else
    {
```

8. การลบโหนดหลังจากโหนดที่กำหนดด้วยฟังก์ชัน DeleteAfter (ต่อ)

```
        H1=H->link; //ให้ H1 ชี้ไปที่โหนดเริ่มต้น
        while (H1->info != HeadData) //ค้นหาข้อมูลตราบใดที่ข้อมูลไม่เท่ากับข้อมูลโหนดหัว
        {
            if (H1->info == data1) //ถ้าพบข้อมูล
            {
                if (H1->link==H) //ถ้าไม่มีโหนดอีกต่อไป
                    printf ("This is the HEAD Node,Can't delete it!!!\n");
                else
                {
                    p=H1->link; //ทำเครื่องหมายที่โหนดสำหรับลบ
                    if(p->link==H) //ถ้า p เป็นโหนดสุดท้าย
                        H1->link=H; //ตั้งลิงก์ของ H1 ให้ชี้ไปที่โหนดหัว
                    else
                        H1->link=p->link; //ถ้าไม่ ให้ลิงก์ของ H1 ชี้ไปยังที่อยู่เดียวกันกับ p
                    free(p); //ปล่อยโหนดกลับไปยังพื้นที่เก็บ
                } //สิ้นสุด if2
            } //สิ้นสุด if1
            H1=H1->link; //ข้าม H1 ไปที่โหนดถัดไป
        } //สิ้นสุด while
    } //สิ้นสุด IF
} //สิ้นสุดฟังก์ชัน
```

8. การลบโหนดหลังจากโหนดที่กำหนดด้วยฟังก์ชัน DeleteAfter (ต่อ)

ฟังก์ชันนี้ทำหน้าที่ลบโหนดที่อยู่หลังจากโหนดที่มีข้อมูลตรงกับค่าที่กำหนด (data1) ฟังก์ชันจะไม่อนุญาตให้ลบโหนดหัว (Head Node) หากโหนดถัดไปเป็นโหนดสุดท้ายก่อนกลับไปที่โหนดหัว โหนดนั้นจะถูกลบออก และลิงก์จะถูกปรับให้กลับไปยังโหนดหัว

9. ฟังก์ชันหลัก (main)

```
int main() //ฟังก์ชันหลัก
{
    p=Allocate(); //สร้างโหนดหัว
    p->info=HeadData; //กำหนดข้อมูลพิเศษ
    p->link=p; //ลิงก์กลับไปที่โหนด
    n=10; //ตั้งจำนวนโหนด
    CreateNNode(n); //เรียกฟังก์ชันสร้างโหนด N
    printf("PROGRAM SINGLY CIRCULAR LINKED LIST \n");
    printf("======================================== \n");
    printf("All Data in Linked List \n");
    ShowAllNode(); //เรียกฟังก์ชันแสดงโหนดทั้งหมด
    ch=' ';
    while (ch != 'E')
    {
        printf("MENU : [I:Insert D:Delete E:Exit]");
        ch=getch();
        switch (ch)
          {
```

9. ฟังก์ชันหลัก (main) (ต่อ)

```
            case 'I' : printf("\nInsert After data : " ); //รับข้อมูลสำหรับแทรกหลัง
                scanf("%d",&data);
                InsertAfter(data); //เรียกฟังก์ชันแทรกหลังข้อมูล
                printf("\nข้อมูลทั้งหมดในลิสต์หลังจากแทรก\n");
                ShowAllNode(); //เรียกฟังก์ชันแสดงโหนดทั้งหมด
                break;
            case 'D' : printf("\nDelete After data : " ); //รับข้อมูลสำหรับลบหลัง
                scanf("%d",&data);
                DeleteAfter(data); //เรียกฟังก์ชันลบหลังข้อมูล
                printf("\nAll Data in Linked List AFTER DELETED\n");
               ShowAllNode(); //เรียกฟังก์ชันแสดงโหนดทั้งหมด
                break;
        } // สิ้นสุด Switch...case
    } //สิ้นสุด While
    return(0);
} //สิ้นสุดฟังก์ชันหลัก
```

ฟังก์ชัน main ทำหน้าที่สร้างลิสต์เริ่มต้นและจัดการกับการทำงานทั้งหมดของโปรแกรม โดยจะสร้างลิสต์ที่มี 10 โหนดเริ่มต้นและแสดงข้อมูลทั้งหมดในลิสต์ จากนั้นจะให้ผู้ใช้เลือกเมนูเพื่อแทรกหรือลบข้อมูลจนกว่าจะเลือกออกจากโปรแกรม (กด E)

ผลลัพธ์การใช้งานโปรแกรม SINGLY CIRCULAR LINKED LIST

โปรแกรมนี้เป็นตัวอย่างการใช้งาน Singly Circular Linked List ซึ่งเป็นโครงสร้างข้อมูลที่ประกอบด้วยโหนด (Node) หลายๆ โหนดที่เชื่อมต่อกันเป็นวงกลม โดยมีโหนดหัว (Head Node) เป็นจุดเริ่มต้นและจุดสิ้นสุดของลิสต์ การทำงานของโปรแกรมนี้ครอบคลุมทั้งการสร้างลิสต์ การแสดงข้อมูลในลิสต์ การแทรกโหนดใหม่หลังจากโหนดที่กำหนด และการลบโหนดที่อยู่หลังโหนดที่กำหนด

ขั้นตอนการทำงานและผลลัพธ์ของโปรแกรม

1. การเริ่มต้นโปรแกรมและการสร้างลิสต์เริ่มต้น

- เมื่อเริ่มต้นโปรแกรม จะมีการสร้างลิสต์ที่ประกอบด้วย 10 โหนดที่มีค่าข้อมูล (Info) เป็นค่าเลขสุ่มตั้งแต่ 1 ถึง 99 โหนดแรกในลิสต์จะเป็นโหนดหัว (Head Node) ซึ่งมีค่าข้อมูลเป็น -999 เพื่อบ่งบอกว่าเป็นโหนดพิเศษ โหนดอื่นๆ จะเชื่อมต่อกันเป็นวงกลม โดยโหนดสุดท้ายจะชี้กลับไปยังโหนดหัว
- ผลลัพธ์ที่แสดงจะเป็นรายการของโหนดในลิสต์ เริ่มจากโหนดแรกที่มีที่อยู่ (Address) ของโหนด ข้อมูล (Info) และตัวชี้ (Link) ที่ชี้ไปยังโหนดถัดไปในลิสต์

```
E:\ENGCE124\Coding 9 SINGL'
PROGRAM SINGLY CIRCULAR LINKED LIST
========================================
All Data in Linked List
H = 631470
1) : 631490     INFO : 42       LINK : 6314b0
2) : 6314b0     INFO : 54       LINK : 6314d0
3) : 6314d0     INFO : 98       LINK : 6314f0
4) : 6314f0     INFO : 68       LINK : 631510
5) : 631510     INFO : 63       LINK : 631530
6) : 631530     INFO : 83       LINK : 631550
7) : 631550     INFO : 94       LINK : 631570
8) : 631570     INFO : 55       LINK : 631590
9) : 631590     INFO : 35       LINK : 6315b0
10) : 6315b0    INFO : 12       LINK : 631470
MENU : [I:Insert D:Delete E:Exit]
```

2. การแสดงเมนูและการรับคำสั่งจากผู้ใช้

- หลังจากสร้างลิสต์แล้ว โปรแกรมจะเข้าสู่ลูปที่ให้ผู้ใช้เลือกทำงานกับลิสต์ผ่านทางเมนู โดยผู้ใช้สามารถเลือกได้ว่าจะ
    - แทรกโหนดใหม่หลังจากโหนดที่มีข้อมูลที่กำหนด (Insert After)
    - ลบโหนดที่อยู่หลังโหนดที่มีข้อมูลที่กำหนด (Delete After)
    - หรือออกจากโปรแกรม (Exit)

3. การแทรกโหนดใหม่หลังจากโหนดที่กำหนด

- เมื่อผู้ใช้เลือกแทรกโหนดใหม่ โปรแกรมจะขอให้ผู้ใช้ป้อนข้อมูลของโหนดที่ต้องการแทรกโหนดใหม่ หลังจากนั้นจะให้ป้อนข้อมูลที่ต้องการแทรกเข้าไป โปรแกรมจะทำการสร้างโหนดใหม่และแทรกเข้าไปในลิสต์หลังจากโหนดที่กำหนด
- ผลลัพธ์จะแสดงลิสต์ที่มีโหนดใหม่แทรกเข้าไป พร้อมที่อยู่ของโหนด ข้อมูล และลิงก์

```
E:\ENGCE124\Coding 9 SINGL'

PROGRAM SINGLY CIRCULAR LINKED LIST
========================================
All Data in Linked List
H = b41470
1) : b41490     INFO : 42       LINK : b414b0
2) : b414b0     INFO : 54       LINK : b414d0
3) : b414d0     INFO : 98       LINK : b414f0
4) : b414f0     INFO : 68       LINK : b41510
5) : b41510     INFO : 63       LINK : b41530
6) : b41530     INFO : 83       LINK : b41550
7) : b41550     INFO : 94       LINK : b41570
8) : b41570     INFO : 55       LINK : b41590
9) : b41590     INFO : 35       LINK : b415b0
10) : b415b0    INFO : 12       LINK : b41470
MENU : [I:Insert D:Delete E:Exit]
Insert After data : 42

Insert data : 99

All Data in Linked List AFTER INSERTED
H = b41470
1) : b41490     INFO : 42       LINK : b415d0
2) : b415d0     INFO : 99       LINK : b414b0
3) : b414b0     INFO : 54       LINK : b414d0
4) : b414d0     INFO : 98       LINK : b414f0
5) : b414f0     INFO : 68       LINK : b41510
6) : b41510     INFO : 63       LINK : b41530
7) : b41530     INFO : 83       LINK : b41550
8) : b41550     INFO : 94       LINK : b41570
9) : b41570     INFO : 55       LINK : b41590
10) : b41590    INFO : 35       LINK : b415b0
11) : b415b0    INFO : 12       LINK : b41470
MENU : [I:Insert D:Delete E:Exit]
```

4. การลบโหนดที่อยู่หลังโหนดที่กำหนด

- เมื่อผู้ใช้เลือกที่จะลบโหนด โปรแกรมจะขอให้ผู้ใช้ป้อนข้อมูลของโหนดที่ต้องการลบโหนดที่อยู่หลังจากมัน โปรแกรมจะทำการลบโหนดถัดไปที่เชื่อมต่ออยู่กับโหนดที่มีข้อมูลตามที่กำหนด
- โปรแกรมจะไม่อนุญาตให้ลบโหนดหัว (Head Node) หากโหนดที่ต้องการลบเป็นโหนดสุดท้ายในลิสต์ โปรแกรมจะปรับให้โหนดก่อนหน้านี้ชี้กลับไปที่โหนดหัวแทน
- ผลลัพธ์จะแสดงลิสต์ที่มีการลบโหนดออกไปแล้ว พร้อมที่อยู่ของโหนด ข้อมูล และลิงก์

4. การลบโหนดที่อยู่หลังโหนดที่กำหนด (ต่อ)

```
All Data in Linked List AFTER INSERTED
H = b41470
1) : b41490     INFO : 42       LINK : b415d0
2) : b415d0     INFO : 99       LINK : b414b0
3) : b414b0     INFO : 54       LINK : b414d0
4) : b414d0     INFO : 98       LINK : b414f0
5) : b414f0     INFO : 68       LINK : b41510
6) : b41510     INFO : 63       LINK : b41530
7) : b41530     INFO : 83       LINK : b41550
8) : b41550     INFO : 94       LINK : b41570
9) : b41570     INFO : 55       LINK : b41590
10) : b41590    INFO : 35       LINK : b415b0
11) : b415b0    INFO : 12       LINK : b41470
MENU : [I:Insert D:Delete E:Exit]
Delete After data : 99

All Data in Linked List AFTER DELETED
H = b41470
1) : b41490     INFO : 42       LINK : b415d0
2) : b415d0     INFO : 99       LINK : b414d0
3) : b414d0     INFO : 98       LINK : b414f0
4) : b414f0     INFO : 68       LINK : b41510
5) : b41510     INFO : 63       LINK : b41530
6) : b41530     INFO : 83       LINK : b41550
7) : b41550     INFO : 94       LINK : b41570
8) : b41570     INFO : 55       LINK : b41590
9) : b41590     INFO : 35       LINK : b415b0
10) : b415b0    INFO : 12       LINK : b41470
MENU : [I:Insert D:Delete E:Exit]
```

5. การออกจากโปรแกรม

- เมื่อผู้ใช้กด E โปรแกรมจะสิ้นสุดการทำงาน และลูปหลักจะหยุดทำงาน ส่งผลให้โปรแกรมจบการทำงานและคืนค่าควบคุมไปยังระบบปฏิบัติการ

```
E:\ENGCE124\Coding 9 SINGL'

PROGRAM SINGLY CIRCULAR LINKED LIST
=====================================
All Data in Linked List
H = be1470
1) : be1490     INFO : 42       LINK : be14b0
2) : be14b0     INFO : 54       LINK : be14d0
3) : be14d0     INFO : 98       LINK : be14f0
4) : be14f0     INFO : 68       LINK : be1510
5) : be1510     INFO : 63       LINK : be1530
6) : be1530     INFO : 83       LINK : be1550
7) : be1550     INFO : 94       LINK : be1570
8) : be1570     INFO : 55       LINK : be1590
9) : be1590     INFO : 35       LINK : be15b0
10) : be15b0    INFO : 12       LINK : be1470
MENU : [I:Insert D:Delete E:Exit]
--------------------------------
Process exited after 1.764 seconds with return value 0
Press any key to continue . . .
```

บรรณานุกรม

ChatGPT. (-). Implementation of a Singly Circular Linked List in C. สืบค้น 28 สิงหาคม 2567, จาก https://chatgpt.com/