ใบงานที่ 10

เรื่อง Doubly Circular Linkedlist

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นายกฤษฎา วิริยา

65543206041-7

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

- แสดงโค้ดโปรแกรมเป็นส่วนๆพร้อมทั้งอธิบาย
- แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
- สรุปผลการทดลอง

ลำดับขั้นการทดลอง

```c
#include <stdio.h>    //use printf
#include <conio.h>    //use getch
#include <stdlib.h>   //usemolloc
#define HeadInfo -999 // Define data of Head Node

typedef struct Node { // Declare structure of node
  int info;
  struct Node *llink;
  struct Node *rlink;
} Node;

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

#include <stdio.h>: เป็นการเรียกใช้ไลบรารีของภาษา C สำหรับการทำงานกับข้อมูลและการแสดงผลบนหน้าจอ

#include <conio.h>: เป็นการเรียกใช้ไลบรารีของภาษา C สำหรับการทำงานที่เกี่ยวข้องกับคีย์บอร์ด

#include <stdlib.h>: เป็นการเรียกใช้ไลบรารีของภาษา C สำหรับการจัดการกับหน่วยความจำ (memory allocation) และฟังก์ชั่นที่เกี่ยวข้อง

#define HeadInfo -999: กำหนดค่าคงที่ HeadInfo เป็น -999 ซึ่งจะใช้เป็นข้อมูลพิเศษในโหนดแรก (HEAD NODE)

typedef struct Node: ประกาศโครงสร้างข้อมูล Node ที่มี info เป็นตัวแปรเก็บข้อมูล และ llink, rlink เป็นตัวแปรเก็บพอยน์เตอร์ที่เชื่อมโยงกับโหนดก่อนหน้าและถัดไปในรายการ

struct Node *H, *H1, *p, *q;: ประกาศตัวแปรพอยน์เตอร์ที่ใช้ในโปรแกรม

```c
Node *Allocate() { // Allocate 1 node from storage pool
  struct Node *temp;
  temp = (Node *)malloc(sizeof(Node)); // Allocate node by size declare
  return (temp);
}
```

Node *Allocate(): ฟังก์ชัน Allocate ใช้สำหรับจองพื้นที่หน่วยความจำสำหรับโหนดใหม่ โดยใช้ malloc เพื่อจองพื้นที่ขนาด sizeof(Node) และส่งกลับเป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังโหนดใหม่

```c
void CreateNNode(int n) { // Create N Node put data and link its
  int i, temp;
  H1 = H; // Start H1 at here
  for (i = 1; i <= n; i++) {// Count N Node
    p = Allocate();          // Allocate New Node
    temp = 1 + rand() % 99; // random difference number 1..99
    p->info = temp;          // Put random data in to node
    H1->rlink = p;           // Link first node to second node
    p->llink = H1;           // LLink point back to predecessor node
    H1 = p;                  // Let H1 point to last node
    H1->rlink = H;           // Set rlink of H1 point to HEAD NODE
    H->llink = H1;           // Set LLink of H point to H1
  }
}
```

void CreateNNode(int n): ฟังก์ชัน CreateNNode ใช้สำหรับสร้าง Doubly Circular Linked List โดยรับจำนวน n โหนดที่ต้องการสร้าง และใส่ข้อมูลที่สุ่มได้ในแต่ละโหนด โดยเริ่มต้นที่โหนด HEAD

```c
void ShowAllNode() {
  printf("H = %x\n", H); // Display address of pointer H
  p = H;                 // Set start point of p pointer
  i = 1;                 // set start value of counter
  if (p->rlink != H) {   // if have more node
    p = p->rlink;  // Skip pointer to first node
    while (p != H) { // While if P <> H
      printf("%d) : %x\t", i, p);       // Show COUNTER and POINTER
      printf("LLINK : %x\t", p->llink); // Show lLINK
      printf("INFO : %d\t", p->info);   // Show INFO
      printf("RLINK : %x\n", p->rlink); // Show RLINK
      p = p->rlink;                     // Skip to next node
      i++;                              // Skip Counter
    }                                   // End While
  }                                     // End if
} // Enf Fn.
```

void ShowAllNode(): ฟังก์ชัน ShowAllNode ใช้สำหรับแสดงข้อมูลในโหนดทั้งหมดในรายการ โดยเริ่มต้นที่โหนด HEAD และทำการวนลูปผ่านโหนดทั้งหมดในรายการ แสดงข้อมูลในแต่ละโหนด และลิงก์ที่ชี้ไปยังโหนดถัดไปและโหนดก่อนหน้า

```c
void InsertAfter(int data1) {
  int temp; // Temporary variable
  if (H->rlink == H)
    printf("Linked List have no node!!..\n");
  else {
    H1 = H->rlink; // Let H1 point at 1st node
    while (H1->info != HeadInfo) {  // Search for the data while H1 loop back to HAED Node
      if (H1->info == data1) { // if Found
        p = Allocate();           // Allocat one node from storage pool
        printf("\nInsert data : "); // Input data for insert
        scanf("%d", &temp);       // Read from KBD
        p->info = temp;           // Entry temporary data into INFO of node
        if (H1->rlink == H) {       // IF H1 is Last Node
          p->rlink = H; // Let p Point to HEAD Node
          H->llink = p; // Let H Point to Last Node
        } else {
          p->rlink = H1->rlink; // Change pointer 1st for insert node (FAR to NEAR)
          H1->rlink->llink = p; // LLINK(RLINK(H1))=p
        }
        p->llink = H1; // LLINK(P)=H1
        H1->rlink = p;    // RLINK(H1)=p101
      }                   // End if
      H1 = H1->rlink;     // Skip H1 to next node
    }                     // End while
  }                       // End IF
} // End Fn.
```

void InsertAfter(int data1): ฟังก์ชัน InsertAfter ใช้สำหรับแทรกโหนดใหม่หลังจากโหนดที่มีข้อมูลเท่ากับ data1 โดยให้ผู้ใช้ป้อนข้อมูลใหม่ที่ต้องการแทรกหลังจากนั้นสร้างโหนดใหม่และแก้ไขลิงก์ให้เป็นอย่างถูกต้อง

```
void InsertBefore(int data1) {
  int temp; // Temporary variable
  if (H->rlink == H)
    printf("Linked List have no node!!..\n");
  else {
    H1 = H->rlink;              // Let H1 point at 1st node
    while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
      if (H1->info == data1) { // if Found
        p = Allocate();          // Allocate one node from storage pool
        printf("\nInsert data : "); // Input data for insert
        scanf("%d", &temp);      // Read from KBD
        p->info = temp;          // Entry temporary data into INFO of node
        if (H1->llink == H) {     // First Node
          p->llink = H; // LLINK(p) Point to HEAD Node
          H->rlink = p; // RLINK(H) Point to p
        } else {
          H1->llink->rlink = p; // RLINK(LLINK(H1))=p
          p->llink = H1->llink; // LLINK(p)=LLINK(H1)
        }
        H1->llink = p; // LLINK(H1)=p
        p->rlink = H1; // RLINK(P)=H1
      }              // End if
      H1 = H1->rlink;  // Skip H1 to next node
    }                // End while
  }                  // End IF
} // End Fn
```

void InsertBefore(int data1): ฟังก์ชัน InsertBefore ใช้สำหรับแทรกโหนดใหม่ก่อนโหนดที่มีข้อมูลเท่ากับ data1 โดยให้ผู้ใช้ป้อนข้อมูลใหม่ที่ต้องการแทรกก่อนหน้านั้นและสร้างโหนดใหม่และแก้ไขลิงก์ให้เป็นอย่างถูกต้อง

```
void DeleteBefore(int data1) {
  int temp; // Temporary variable
  if (H->rlink == H)
    printf("Linked List have NO NODE!!..\n");
  else {
    H1 = H->rlink;              // Let H1 point at 1st node
    while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
      if (H1->info == data1) { // if Found
        if (H1->llink == H) // If no more node
          printf("No more node before here,Can't delete it!!!\n");
        else {
          p = H1->llink;        // Mark at node for Delete
          H1->llink = p->llink; // If not set link of H1 point same address of p
          p->llink->rlink = H1;
          free(p);     // Free node to storage pool
        }              // End if2
      }                // End if1
      H1 = H1->rlink; // Skip H1 to next node
    }                // End while
  }                  // End IF
} // End Fn.
```

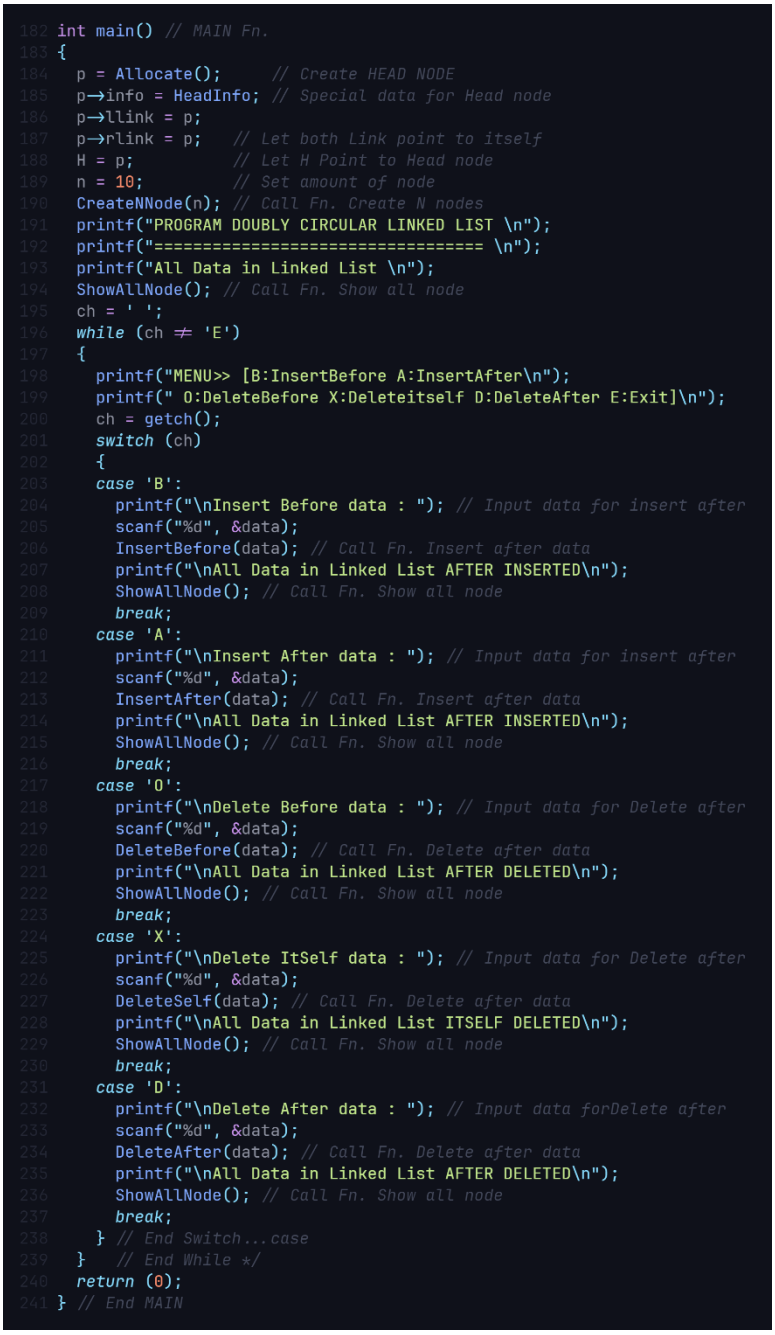
void DeleteBefore(int data1): ฟังก์ชัน DeleteBefore ใช้สำหรับลบโหนดก่อนโหนดที่มีข้อมูลเท่ากับ data1 โดยลบโหนดนั้นออกจาก Doubly Circular Linked List ถ้าไม่มีโหนดก่อนหน้า จะแสดงข้อความแจ้งเตือนว่าไม่สามารถลบได้

```
void DeleteSelf(int data1) {
  int temp; // Temporary variable
  if (H->rlink == H)
    printf("Linked List have NO NODE!!..\n");
  else {
    H1 = H->rlink;                // Let H1 point at 1st node
    while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
      if (H1->info == data1) { // if Found
        p = H1;                             // Mark at node for Delete
        if (p->llink == H && p->rlink == H) { // If only one node
          H->llink = H; // Set HEAD Pointer point it self
          H->rlink = H;
        } else {
          p->llink->rlink = p->rlink;
          p->rlink->llink = p->llink;
        }
        free(p);       // Free node to storage pool
      }                // End if1
      H1 = H1->rlink; // Skip H1 to next node
    }                  // End while
  }                    // End IF
} // End Fn.
```

void DeleteBefore(int data1): ฟังก์ชัน DeleteBefore สำหรับลบโหนดก่อนโหนดที่มีข้อมูลที่ตรงกับค่า data1 โดยลบโหนดนั้นออกจาก Doubly Circular Linked List หากไม่มีโหนดก่อนหน้า จะแสดงข้อความแจ้งเตือนว่าไม่สามารถลบได้

```
void DeleteAfter(int data1) {
  int temp; // Temporary variable
  if (H->rlink == H)
    printf("Linked List have NO NODE!!..\n");
  else {
    H1 = H->rlink;                // Let H1 point at 1st node
    while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
      if (H1->info == data1) { // if Found
        if (H1->rlink == H) // If no more node
          printf("No more node from here,Can't delete it!!!\n");
        else {
          p = H1->rlink;         // Mark at node for Delete
          H1->rlink = p->rlink; // If not set link of H1 point same address of p
          p->rlink->llink = H1;
          free(p);     // Free node to storage pool
        }              // End if2
      }                // End if1
      H1 = H1->rlink; // Skip H1 to next node
    }                  // End while
  }                    // End IF
} // End Fn
```

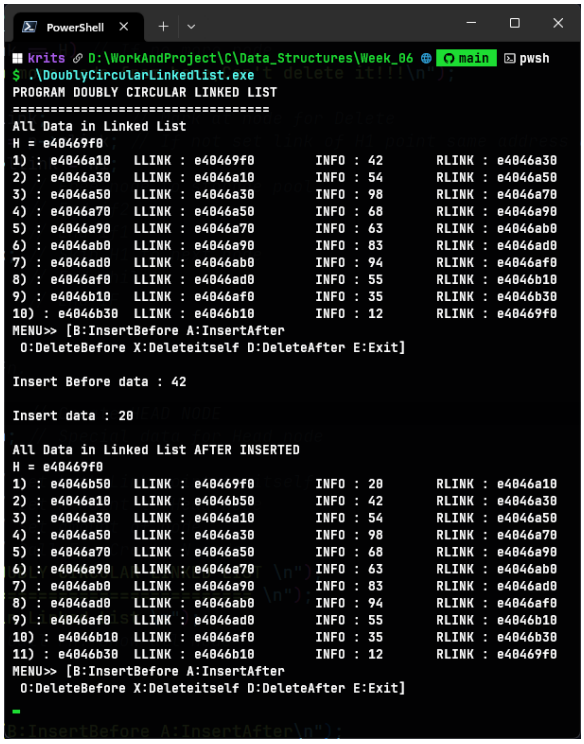
void DeleteAfter(int data1): ฟังก์ชันที่ใช้ลบโหนดในลิงค์ลิสต์หลังจากโหนดที่มีข้อมูลเท่ากับ data1 ในกรณีที่ไม่มีโหนดตามหลัง จะแสดงข้อความว่า "No more node from here,Can't delete it!!!"



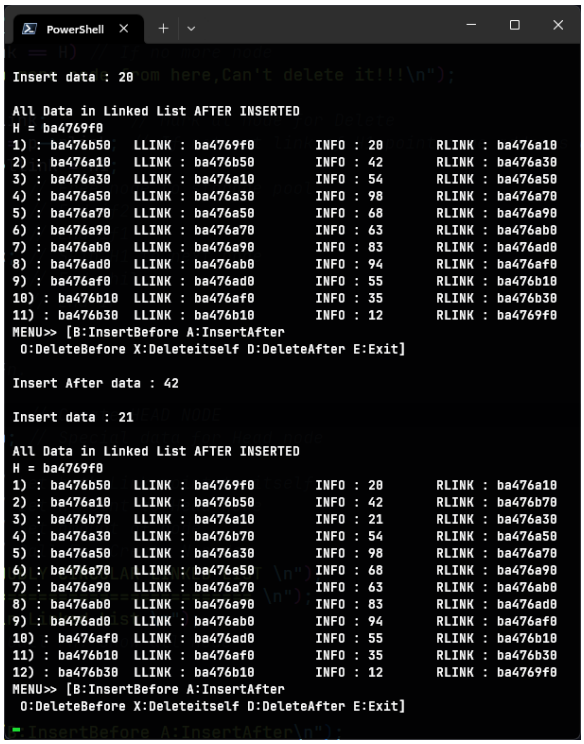
```c
int main() // MAIN Fn.
{
  p = Allocate();      // Create HEAD NODE
  p->info = HeadInfo; // Special data for Head node
  p->llink = p;
  p->rlink = p;    // Let both Link point to itself
  H = p;           // Let H Point to Head node
  n = 10;          // Set amount of node
  CreateNNode(n); // Call Fn. Create N nodes
  printf("PROGRAM DOUBLY CIRCULAR LINKED LIST \n");
  printf("================================= \n");
  printf("All Data in Linked List \n");
  ShowAllNode(); // Call Fn. Show all node
  ch = ' ';
  while (ch != 'E')
  {
    printf("MENU>> [B:InsertBefore A:InsertAfter\n");
    printf(" O:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]\n");
    ch = getch();
    switch (ch)
    {
    case 'B':
      printf("\nInsert Before data : "); // Input data for insert after
      scanf("%d", &data);
      InsertBefore(data); // Call Fn. Insert after data
      printf("\nAll Data in Linked List AFTER INSERTED\n");
      ShowAllNode(); // Call Fn. Show all node
      break;
    case 'A':
      printf("\nInsert After data : "); // Input data for insert after
      scanf("%d", &data);
      InsertAfter(data); // Call Fn. Insert after data
      printf("\nAll Data in Linked List AFTER INSERTED\n");
      ShowAllNode(); // Call Fn. Show all node
      break;
    case 'O':
      printf("\nDelete Before data : "); // Input data for Delete after
      scanf("%d", &data);
      DeleteBefore(data); // Call Fn. Delete after data
      printf("\nAll Data in Linked List AFTER DELETED\n");
      ShowAllNode(); // Call Fn. Show all node
      break;
    case 'X':
      printf("\nDelete ItSelf data : "); // Input data for Delete after
      scanf("%d", &data);
      DeleteSelf(data); // Call Fn. Delete after data
      printf("\nAll Data in Linked List ITSELF DELETED\n");
      ShowAllNode(); // Call Fn. Show all node
      break;
    case 'D':
      printf("\nDelete After data : "); // Input data forDelete after
      scanf("%d", &data);
      DeleteAfter(data); // Call Fn. Delete after data
      printf("\nAll Data in Linked List AFTER DELETED\n");
      ShowAllNode(); // Call Fn. Show all node
      break;
    } // End Switch...case
  }   // End While */
  return (0);
} // End MAIN
```

int main(): ฟังก์ชันหลักที่ใช้เรียกใช้ฟังก์ชันด้านบนเพื่อสร้างและจัดการกับ Doubly Circular Linked List โดยมีเมนูให้เลือกในการเพิ่มหรือลบโหนดจากรายการ และมีการแสดงรายการหลังจากแก้ไขทุกครั้งที่มีการเปลี่ยนแปลง

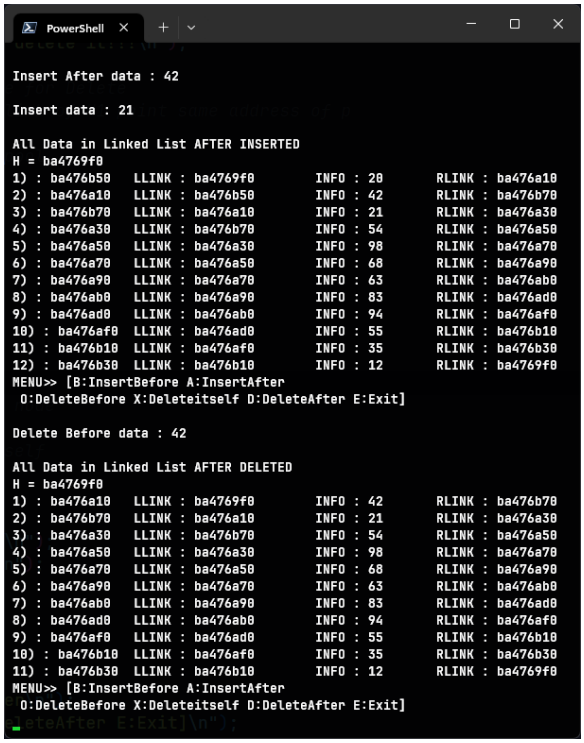
Run Program



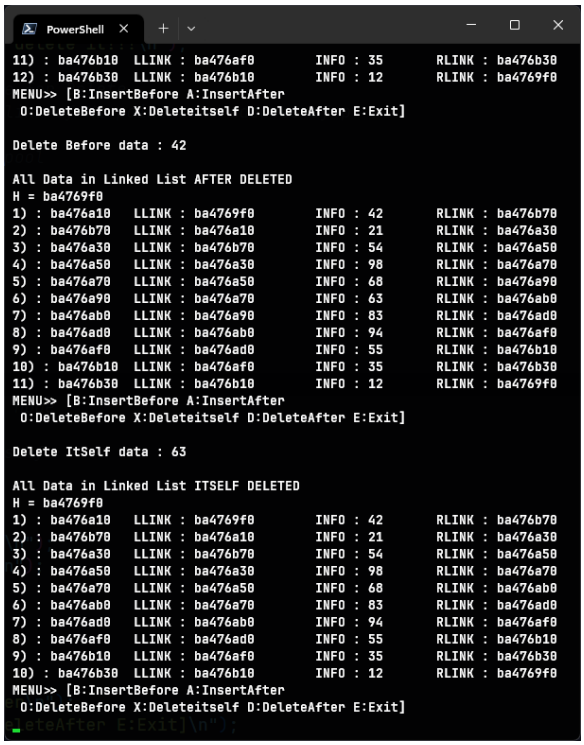
Insert Before



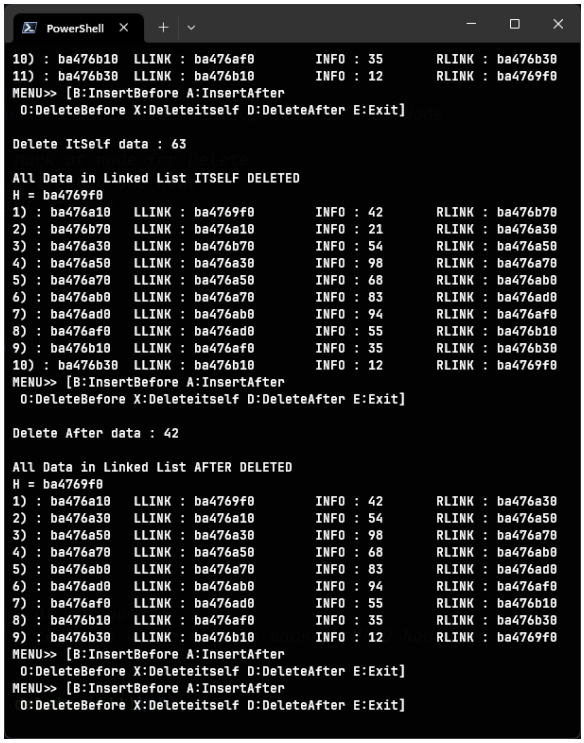
Insert After



Delete Before



Delete Itself



Delete After

สรุปผลการทดลอง

โปรแกรมนี้ช่วยให้ผู้ใช้สามารถสร้าง Doubly Circular Linked List, แสดงข้อมูลในโหนดทั้งหมด, แทรกโหนดใหม่หลังจากโหนดที่ต้องการ, ลบโหนดที่ตั้งหลังจากโหนดที่ต้องการ และทำการออกจากโปรแกรมเมื่อต้องการสิ้นสุดการทำงาน