ใบงานที่ 12

เรื่อง Tree structure by NODE POINTER Method

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นาย สารินทร์ อินต๊ะรักษา รหัส 65543206082-1

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

- สร้างโค้ดโปรแกรมตามตัวอย่างในเอกสารประกอบการสอนสัปดาห์ที่7
- แสดงโค้ดโปรแกรมเป็นส่วนๆ พร้อมอธิบาย
- แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
- สรุปผลการทดลอง

ลำดับขั้นการทดลอง

```
struct Node {//Declare structure of Tree node
int info;
struct Node *lson;
struct Node *rson;
};
struct Node *T, *address[MaxNode]; // Declare pointer T of Tree node
int i,N,info[MaxNode];
char ch;
Node *Allocate() {//Allocate 1 node from storage pool and return pointer of node
struct Node *temp;
temp=(Node*)malloc(sizeof(Node)); //Allocate node by size declare
return(temp);
}
```

- ประกาศโครงสร้าง Node
  - info: เก็บข้อมูลที่ต้องการเก็บในแต่ละโหนดของต้นไม้
  - lson: เป็นพอยน์เตอร์ที่ชี้ไปยังโหนดลูกทางซ้าย
  - rson: เป็นพอยน์เตอร์ที่ชี้ไปยังโหนดลูกทางขวา
- ประกาศตัวแปรและพอยน์เตอร์สำหรับต้นไม้:
  - T: เป็นพอยน์เตอร์ที่ชี้ไปยังโหนดรากของต้นไม้
  - address[MaxNode]: เป็นอาร์เรย์ของพอยน์เตอร์ที่ใช้เก็บที่อยู่ของโหนดในต้นไม้
- ประกาศตัวแปรและฟังก์ชัน Allocate:
  - info[MaxNode]: เป็นอาร์เรย์ที่ใช้เก็บข้อมูลที่จะใช้สร้างโหนดในต้นไม้
  - ch: เป็นตัวแปรที่ใช้สำหรับการอ่านข้อมูลจากผู้ใช้
  - Allocate(): เป็นฟังก์ชันที่ใช้ในการจองพื้นที่ในหน่วยความจำเพื่อสร้างโหนดใหม่และส่งคืนพอยน์เตอร์ที่ชี้ไปยังโหนดที่สร้างขึ้น

```c
int main()
{
N=31;
CreateTreeNP(N);
while (ch != 'E') //Loop until 'E' Pressed
{
printf("\nPROGRAM TREE(Node Pointer) \n");
printf("================================= \n");
printf("N : %d\n",N);
printf("Sequence of data : ");
for (i=1;i<=N;i++) //Show Data Sequence
printf("%d ",info[i]);
ShowTree(); //Show tree structure
printf("\nMENU => P:PreOrder I:InOrder O:PostOrder E:Exit");
printf("\n--------------------------------------------------------------------------------\n");
ch=getch();
switch (ch)
{
case 'P' : printf("PRE ORDER TRAVERSAL : ");
PreOrder(T);
printf("\n--------------------------------------------------------------------------------\n");
break;
case 'I' : printf("IN ORDER TRAVERSAL : ");
InOrder(T);
printf("\n--------------------------------------------------------------------------------\n");
break;
case 'O' : printf("POST ORDER TRAVERSAL : ");
PostOrder(T);
printf("\n--------------------------------------------------------------------------------\n");
break;
} //End Switch...case
} //End While
return(0);
}//End MAIN
```

- int main() : เป็นฟังก์ชันหลักของโปรแกรมที่จะทำการควบคุมการทำงานทั้งหมดของโปรแกรม.
- ในลูป while (ch != 'E') : โปรแกรมจะวนลูปรอรับคำสั่งจากผู้ใช้ตราบเท่าที่ไม่ได้ป้อน 'E' โดยลูปนี้จะแสดงเมนูต่าง ๆ และจะตรวจสอบคำสั่งที่ผู้ใช้ป้อนเพื่อดำเนินการต่อ.
- แสดงข้อความและข้อมูลต่าง ๆ :
  - แสดงข้อความ "PROGRAM TREE(Node Pointer)" เพื่อบ่งชี้ถึงโปรแกรมที่ทำงาน.
  - แสดงค่าของตัวแปร N ที่กำหนดไว้.
  - แสดงลำดับข้อมูลที่ใช้สร้างต้นไม้.
  - เรียกใช้ฟังก์ชัน ShowTree() เพื่อแสดงโครงสร้างของต้นไม้.
- เมนูและการจัดการกับต้นไม้ :
  - แสดงเมนูทางด้านล่างสำหรับผู้ใช้ให้เลือกคำสั่ง.
  - อ่านคำสั่งที่ผู้ใช้ป้อนโดยใช้ฟังก์ชัน getch().
  - ใช้คำสั่ง switch เพื่อตรวจสอบคำสั่งที่ผู้ใช้ป้อนและดำเนินการตามคำสั่งที่เลือก.
- ฟังก์ชันการท่องไปในต้นไม้ :
  - สำหรับแต่ละคำสั่งที่ผู้ใช้เลือก (P, I, O) โปรแกรมจะเรียกใช้ฟังก์ชันที่สื่อถึงวิธีการท่องไปในต้นไม้ตามลำดับที่กำหนด (PreOrder, InOrder, PostOrder).

```
void CreateTreeNP(int n) {
int i,temp,Father;
struct Node *p, *FatherPT;
T=NULL; //Set start of T Pointer
for (i=1;i<=n;i++){
p=Allocate(); // Allocate NODE p
temp=1+rand() % 99; //random difference number 1..99
info[i]=temp; //Keep data for Check Correcting of Sequence
address[i]=p; //Keep Address of Node Sequence
p->info=temp; //Keep data to INFO
p->lson=NULL; //Set default LSON=NULL
p->rson=NULL; //Set default RSON=NULL
if (T==NULL) { //Check for T=NULL?
T=p; //Set T point to first node for begining of Treer
}
else{
Father=(i/2); //Calculate FATHER
FatherPT=address[Father]; //Get pointer of Father Node
if(FatherPT->lson == NULL)
FatherPT->lson=p; //Link LSON to new node
else
FatherPT->rson=p; //Link RSON to new node
}
}
}
```

ในฟังก์ชัน CreateTreeNP เป็นฟังก์ชันที่ใช้ในการสร้างต้นไม้โดยมีโหนดจำนวน n โหนด และกำหนดค่าข้อมูลและการเชื่อมโยงโหนดในต้นไม้ดังนี้ :

- int i, temp, Father;: ประกาศตัวแปรที่ใช้ในการวนลูปและเก็บค่าชั่วคราว.
- struct Node *p, *FatherPT;: ประกาศพอยน์เตอร์ที่ใช้เก็บพื้นที่ของโหนดและพอยน์เตอร์ของโหนดบิตของบิตเก่า.
- T=NULL;: กำหนดให้พอยน์เตอร์ T เป็น NULL แสดงว่าต้นไม้เริ่มต้นที่ไม่มีโหนด.
- for (i=1;i<=n;i++): วนลูปสร้างโหนดตามจำนวนที่กำหนด.
- p=Allocate();: จองพื้นที่สำหรับโหนดใหม่โดยใช้ฟังก์ชัน Allocate().
- temp=1+rand() % 99;: สุ่มค่าข้อมูลในโหนดในช่วง 1 ถึง 99.
- info[i]=temp;: เก็บค่าข้อมูลในอาร์เรย์ info เพื่อเช็คความถูกต้องของลำดับข้อมูล.
- address[i]=p;: เก็บพอยน์เตอร์ของโหนดลงในอาร์เรย์ address เพื่ออ้างอิงถึงโหนดตามลำดับ.
- p->info=temp;: กำหนดค่าข้อมูลในโหนด.
- p->lson=NULL;: กำหนดให้ลูกทางซ้ายเป็น NULL.
- p->rson=NULL;: กำหนดให้ลูกทางขวาเป็น NULL.
- if (T==NULL) { ... }: ตรวจสอบว่าต้นไม้ว่างหรือไม่ ถ้าใช่จะกำหนด T เพื่อชี้ไปที่โหนดใหม่ที่สร้าง.

else { ... }: ถ้าต้นไม้ไม่ว่าง จะทำการเชื่อมโยงโหนดใหม่ไปยังโหนดพ่อตามลำดับ โดยเช็คว่าลูกทางซ้ายของโหนดพ่อว่างหรือไม่ ถ้าว่างจะเชื่อมโยงไปทางลูกทางซ้าย ถ้าไม่ว่างจะเชื่อมโยงไปทางลูกทางขวา.

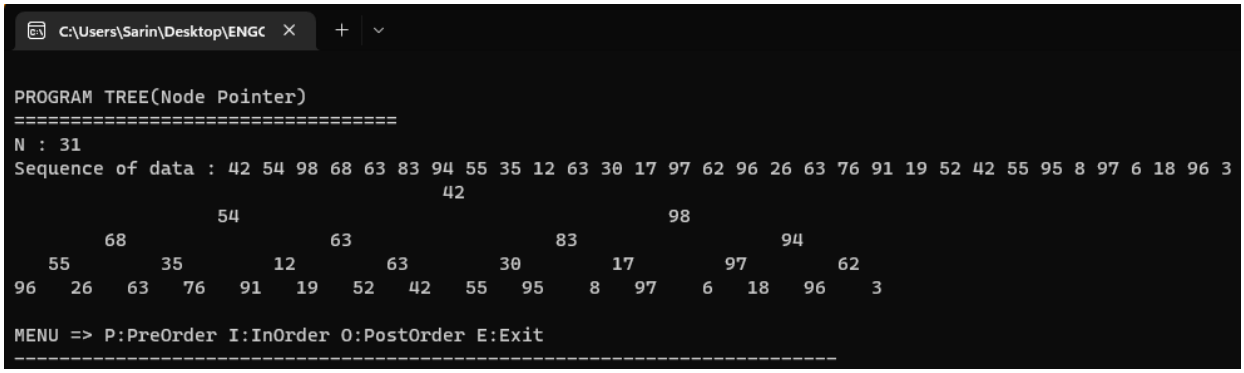
```
void PreOrder(struct Node *i)
{
if (i != NULL) {//if i NOT NULL
printf(" %d",i->info); //Display INFO
PreOrder(i->lson); //Call left Son by PreOrder
PreOrder(i->rson); //Call Right Son by PreOrder
}
}
void InOrder(struct Node *i)
{
if (i != NULL) { //if i NOT NULL
InOrder(i->lson); //Call left Son by InOrder
printf(" %d",i->info); //Display INFO
InOrder(i->rson); //Call Right Son by InOrder
}
}
void PostOrder(struct Node *i)
{
if (i != NULL) {//if i NOT NULL
PostOrder(i->lson); //Call left Son by PostOrder
PostOrder(i->rson); //Call Right Son by PostOrder
printf(" %d",i->info); //Display INFO
}
}
```

- void PreOrder(struct Node *i): ฟังก์ชันสำหรับท่องต้นไม้ด้วยวิธี Pre-order หรือการท่องโหนดรากก่อน.
    - ถ้าโหนด i ไม่เป็น NULL:
        - แสดงค่าข้อมูลในโหนด i.
        - เรียกฟังก์ชัน PreOrder สำหรับลูกทางซ้ายของโหนด i.
        - เรียกฟังก์ชัน PreOrder สำหรับลูกทางขวาของโหนด i.
- void InOrder(struct Node *i): ฟังก์ชันสำหรับท่องต้นไม้ด้วยวิธี In-order หรือการท่องโหนดลูกซ้ายก่อน.
    - ถ้าโหนด i ไม่เป็น NULL:
        - เรียกฟังก์ชัน InOrder สำหรับลูกทางซ้ายของโหนด i.
        - แสดงค่าข้อมูลในโหนด i.
        - เรียกฟังก์ชัน InOrder สำหรับลูกทางขวาของโหนด i.
- void PostOrder(struct Node *i): ฟังก์ชันสำหรับท่องต้นไม้ด้วยวิธี Post-order หรือการท่องโหนดลูกหลัง.
    - ถ้าโหนด i ไม่เป็น NULL:
        - เรียกฟังก์ชัน PostOrder สำหรับลูกทางซ้ายของโหนด i
        - เรียกฟังก์ชัน PostOrder สำหรับลูกทางขวาของโหนด i
        - แสดงค่าข้อมูลในโหนด i

```c
void ShowTree(){
int j,level,start,ends;
j=1;
level=1; //Start at Level 1
printf("\n");
while (info[j] != NULL) {
start=pow(2,level)/2; //Calculate START Node of this level
ends=pow(2,level)-1; //Calculate END Node of this level
for (j=start;j<=ends;j++)
if(info[j] != NULL) {
switch (level) {
case 1 : printf("%40d",info[j]);
break;
case 2 : if (j==2)
printf("%20d",info[j]);
else
printf("%40d",info[j]);
break;
case 3 : if(j==4)
printf("%10d",info[j]);
else
printf("%20d",info[j]);
break;
case 4 : if(j==8)
printf("%5d",info[j]);
else
printf("%10d",info[j]);
break;
case 5 : if(j==16)
printf("%d",info[j]);
else
printf("%5d",info[j]);
break;
}
}
printf("\n"); //Line feed
level++;
}
}
```

- int j, level, start, ends;: ประกาศตัวแปรที่ใช้ในการควบคุมการแสดงผล.
- j=1;: กำหนดค่าเริ่มต้นให้กับตัวแปร j.
- level=1;: กำหนดค่าระดับเริ่มต้นให้เป็น 1.
- while (info[j] != NULL) { ... }: วนลูปตรวจสอบข้อมูลในอาร์เรย์ info ในแต่ละลำดับของโหนด.
- start=pow(2,level)/2;: คำนวณค่าเริ่มต้นของลำดับโหนดที่เป็นไปได้ในระดับนั้นๆ.
- ends=pow(2,level)-1;: คำนวณค่าสิ้นสุดของลำดับโหนดที่เป็นไปได้ในระดับนั้นๆ.
- ในลูป for (j=start;j<=ends;j++) { ... }: วนลูปตามลำดับโหนดในระดับนั้นๆ.
    - ตรวจสอบว่าข้อมูลในโหนดที่ j ไม่เป็น NULL:
        - ใช้ switch สำหรับการแสดงผลโหนดขึ้นอยู่กับระดับของโหนด.
        - แสดงข้อมูลในโหนด j โดยใช้ระยะที่เหมาะสมตามระดับของโหนด.
- level++;: เพิ่มระดับขึ้นไปหนึ่งระดับ.

ผลลัพธ์การทดลอง



สรุปผลการทดลอง

โปรแกรมนี้เป็นโปรแกรมภาษา C ที่ใช้ในการสร้างและแสดงผลข้อมูลของต้นไม้โดยใช้วิธีการ "NODE POINTER Method" และทำการท่องไปในต้นไม้ด้วยวิธีการ Pre-order, In-order, และ Post-order ตามที่ผู้ใช้เลือกจากเมนู

สื่อ / เอกสารอ้างอิง

ไฟล์ประกอบการสอนสัปดาห์ที่ 7 ของ อาจารย์ ปิยพล ยืนยงสถาวร เรื่อง : **Tree structure**