TI DSP,,MCU 및 Xilinx zynq FPGA.
프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)
Gccompil3r@gmail.com
학생-윤지완
Yoonjw7894@naver.com

1.DEQUEUE

```
void enqueue(queue **head,int data){

if(*head==NULL)
{
*head = get_node();
(*head)->data=data;
return ;
}
else{
enqueue(&(*head)->link, data);
}
}
queue *dequeue(queue *head,int data)
{
queue *tmp =head;
if(tmp==NULL)
        printf("there are   no data   THAT you delete₩n");

if(head->data!= data)
head->link = dequeue(head->link,data);
else
{
//queue *res = head->link;
printf("now you delete %d₩n",data);
free(tmp);
return head->link;
}
return head;
}


int main(void){
queue *head=NULL;
enqueue(&head,10);
point_queue(&head);
dequeue(head,20);
point_queue(&head);
return 0;
}
```

앞에 QUEUE 는 앞에 말한 거와 같고 QUEUE *DEQUEUE(QUEUE *HEAD,INT DATA)이 부분이 시작이다.지금 상태는 10 20 30 이 차례대로 결과값이 출력된 상태고

*head 가 가리키고 있는 주소를 *tmp 를 넣고 처음 if 문은 tmp 즉 head 가 가리키고 있는데가 “0”이면 발동되는 함수이다. 하지만 지금은 30 까지 출력한 상태라 *head 는 NULL 인 상태가 아니라 발동이 되지않고 그 다음 IF 문으로 내려가 head 가 가리키고 있는 data 값이랑 내가 지우고자하는 값이랑 같지 않으면 발동되는것인데 저는 20 을 지울라고 하기때문에 여기서 if 문이 발동되어 head->link-dequeue(head->link);부분에서 재귀함수의 특성으로 다시 자기자신을 호출해서 *head 에 현재 *head link 가 가리키고있는 주소로 가는것이고 그럼 다시 queue *dequeue(queue *head,int data)로 돌아와서 주소 3000->2000 으로 변경되고 다시 처음 if 문이 돌고 역시 tmp 는 “0”이 아니무로 다음 if 문에 내려가고 이번에는 주소 2000 link 에는 제가 지우고자하는 20 이 들어있기때문에 이번 if 문은 발동되지않고 다음 else 로 넘어간다.그리고 현제 data 값이 free(tmp)로 해제되고 현제 head->link 를 반환한다.

1.트리 순환

```
#include<stdio.h>
#include<malloc.h>

typedef   struct tree{
  int data;
  struct tree *left;
  struct tree *right;
}tree;

tree *real_qtree()
{

tree *first;
first=(tree*)malloc(sizeof(tree));
first->left=NULL;
first->right=NULL;
return first;
}

void left_child (tree **root, int data)
{


if(*root==NULL)
{
*root=real_qtree();
(*root)->data=data;
return ;
}
else if((*root)->data<data){
left_qtree(&(*root)->left,data);

{
left_qtree(&(*root)->right,data);

}

<main)

int number[13]={ 50,45,73,32,48,46,16,37,120,47,130,127,124};
tree *root=NULL;
int i;
for(i=0;number[i];i++)
left_child(&root,number[i]);
return 0;
}
```

오늘한 트리 순환 c 코드 이다.

기준값에서 크면 외쪽으로 배치 크면 오른쪽으로 배치하는 c 코딩이다.

처음에 구조체 tree 만들고 여기서 배열 정수를 받을 data,그리고 오른쪽 왼쪽 담당 포인터 left.right 구조체 포인터 선언하고 main 에서 number 배열에 값들을 넣어준다.그리고 real_qtree 에서 lefr,right 를 NULL 로 초기화 시키고 *first 를 리턴시킨다.main 에서 *root 도 NULL 을 시키

고 for 문을 써서 배열각 주소에 값을 넣어준다.그리고 lefr_child 재귀함수를 호출한다.그럼 void lefr_child 에가서 *root 를 보내주고 각 배열에 넣어둔 값들을 한개씩 부른다.그러면일단 처음에 *root 는 NULL 이기에 if 문이 실행되어 *root 에 first 를 리턴 시킨다. 그 다음 *root data 에 numbeer[0]=50 을 넣는다.그리고 다시 main 으로 돌아와서 그 다음값을 재귀호출에 의해서 다시 보낸다.이 때는 *root 의 값이 NULL 이 아니기에 첫 번째 if 문은 발동되지 않고 두 번째 else if 문이 발동되고 *root 의 data 값이랑 비교해서 크면 ->else if((*root)->data>data)

작으면->else if((*root)->data<data){

발동되고 다음값은 45 이기때문에 작은 else if 문이 발동되고 l eft_qtree(&(*root)->left=data);
재귀함수호출하고 *root->left 의 주소를 보내주고 이때 주소를 2004 번지가 되고 *root 의
는 3000 이 되고 여기서 다시 처음 if 문에 오고 *root->left 를 가리키고 있는데 이때
left 는 NULL 이기 때문에 IF 문이 발동되고 (*root)->data=data 에 45 를 주소 3000 번지에
넣어준다.&(*root)->left 는 3000 이 되고 이번지 data 에 45 를 넣어주고 다시 main 으로
와서 이번에는 73 을 data 로 보낸다.이번 처음 if 문은 발동되지않고 그 다음 else if 문
으로 와서 73 은 기존 data 보크기에 else if((*root)->data>data)가 발동이 된다.그럼

eft_qtree(&(*root)->right=data); 재귀 호출이 이뤄지고 *root 의 right 는 NULL 이기때문에 처음 if 문이 발동이 된다 그래서 전과같이 100 의주소를 malloc 으로 동적으로 할당을받고 주소 4000 에 data=73 을 넣어주고 다시 main 으로 다시오고 이번에는 32 를불러온다.그 다음 32 는 두번을 비교한다. 처음에 50 과 비교했을때 작으니까 left 이기때문에 2004 번지로 그 다음 2004 번지는 data=45 와 연결이 되어있다.그래서 data=45 는 3000 번지에 있으니까 또 여기서 45 와 비교해서 45 보다 작으니까 3000 번지에 left_qtree(&(*root)->left,data);하고 이때 *root 는 NULL 이 기때문에

*root=real_qtree();
(*root)->data=data;에서 동적 메모리 를 할당받고(*root)->data=32 를 넣어준다 그리고 주소를 5000 번지가 된다.이 방식대로 계속 숫자를 넣으면 된다.

AVL_TREE

TYPEDEF ENUM_ROOT

{

RR,

RL,

LL,

LR,

```
void update_level(AVL *root)
{
int left = root->left ?root->left->level:0;
int right = root->right ?root->right->level:0;
if(left>right)
return left+1;
return right+1;
}
```

처음에 1이있다고 가정하고 재귀함수 끝나고 2는 1보다 크니까 root->right에 들어가며 root ->right->level에 1의값을 넣고 .그리고 다음 if문에서 left는 0이고 right는 값을 가지니까 if문 return right+1을 해주면 1의 level값은 2가되고 2의 level값은 1이 된다.그 다음 재귀함수가 끝나고 3이 오는데 3은 그 전에 2에서 int right = root->right ?root->right->level:0; 이 명령어를 통해서 1이 3에게 전달이 된다. 그 다음으로 2도 1과마찬가지로 left에는 값이 없고 right에만 값이

있기에 return right+1을 통해서 1의 right와 2의 right가 각각 1씩 증가하게 되고 결국 1의level 값은='3',2의level 값은='2',3의 level값은='1'이 되는 것이다.

```
void rotation_chek(AVL *root)
{
int left = root->left ?root->left->level:0;
int right = root->right ?root->right->level:0;

return left-right;
{
```

앞서 root->left,right level에 값을 넣어주는 코드를 햇고 처음 두 줄이 그 값들을 불러오는 코드이다.그리고 반환되는 값으로 왼쪽에서 오른쪽 or 오른쪽에서 왼쪽을 빼고 그 값이

2가 되면

```
if(abs(rotation_check(*root))>1)
{
printf("rotation₩n");
*root = rotation(*root, kinds_of_rot(*root,data));
```

요 부분이 실행이 되는 부분이다. 음수를 방지하기 위해서 절대값을 씌어주었다.

```
void kinds_of_rot(AVL   *root,int   data)
}
printf("Data = %D₩n",data);

if(rotation_check(root)>1)

{

if(root->right->data>data)

return RL;

}

return RR;

}

if(rotation_check(root)<-1)

{

else if(root->left->data<data)

return LR;

}

return LL;

}
```