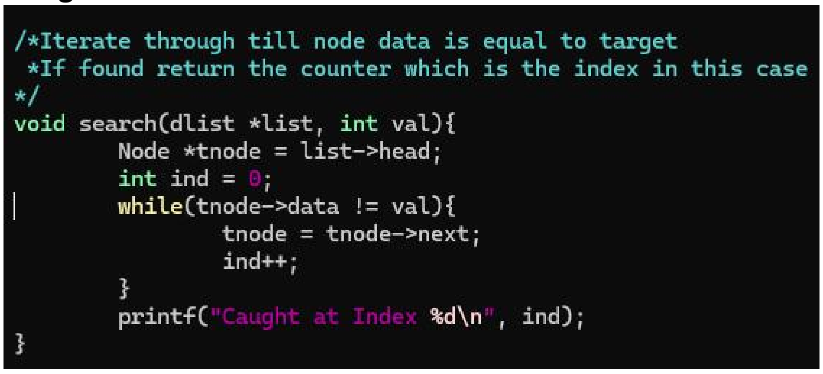
Doubly Linked-List using C Part 3

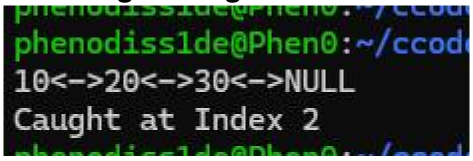
Learning step 5 : Search

For this we just need to iterate until tnode data is the asked value and return index using counter.



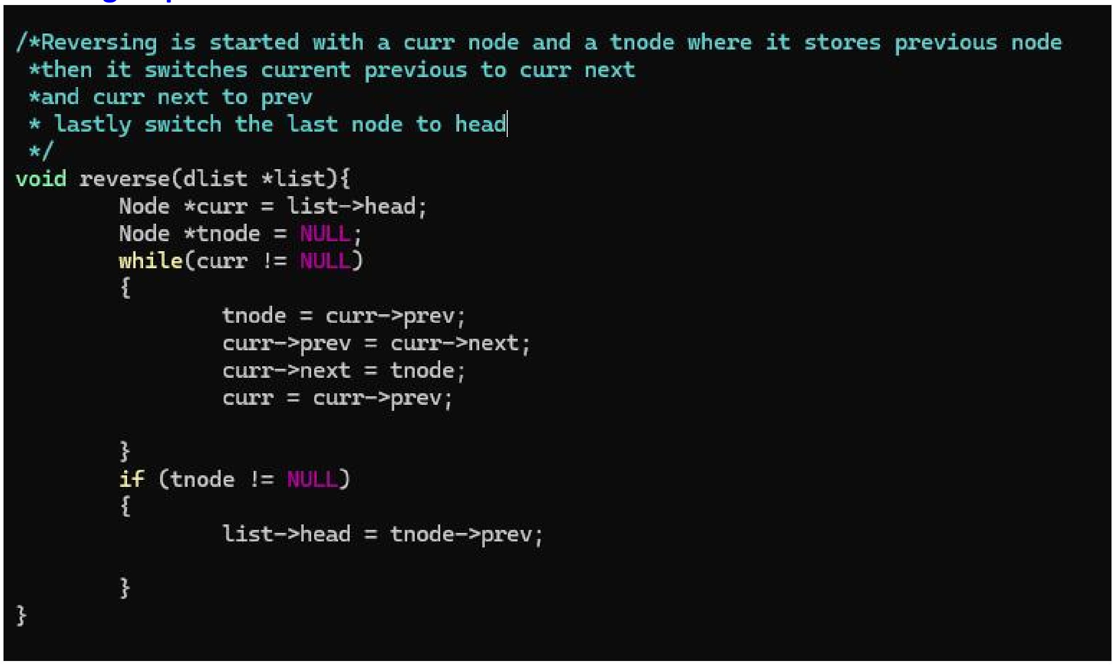
```c
/*Iterate through till node data is equal to target
 *If found return the counter which is the index in this case
*/
void search(dlist *list, int val){
        Node *tnode = list->head;
        int ind = 0;
        while(tnode->data != val){
                tnode = tnode->next;
                ind++;
        }
        printf("Caught at Index %d\n", ind);
}
```

Searching for 30 gives us



```
phenodiss1de@Phen0:~/ccod
10<->20<->30<->NULL
Caught at Index 2
```

Learning step 6 : Reverse



```c
/*Reversing is started with a curr node and a tnode where it stores previous node
 *then it switches current previous to curr next
 *and curr next to prev
 * lastly switch the last node to head
 */
void reverse(dlist *list){
        Node *curr = list->head;
        Node *tnode = NULL;
        while(curr != NULL)
        {
                tnode = curr->prev;
                curr->prev = curr->next;
                curr->next = tnode;
                curr = curr->prev;

        }
        if (tnode != NULL)
        {
                list->head = tnode->prev;

        }
}
```

It’s a bit complicated , here we need a temp pointer to keep track of prev to current node. First save curr prev node to temp, then set current previous to curr next, the we can update curr next to temp and then just increment using curr->previous.