ACCESSING LINKLIST (NODES)

Lets go back to prog1

```
typedef struct node{
    int val;
    struct node *next;
}NodeType;
```

SINGLE POINTER

- ⇨ Having the value of the structure **(node)** aka Pointing to the Node itself not the address

When or What's the use?

- ⇨ When you are only changing the value inside of the node or traversing.
  Traversing means visiting every node (ex. Printing)

DOUBLE POINTER

- ⇨ Pointing to a pointer **which has the node** and having the address of the structure (**node**) itself
- ⇨ Always need a Temporary Node so your main Node won't change its node position

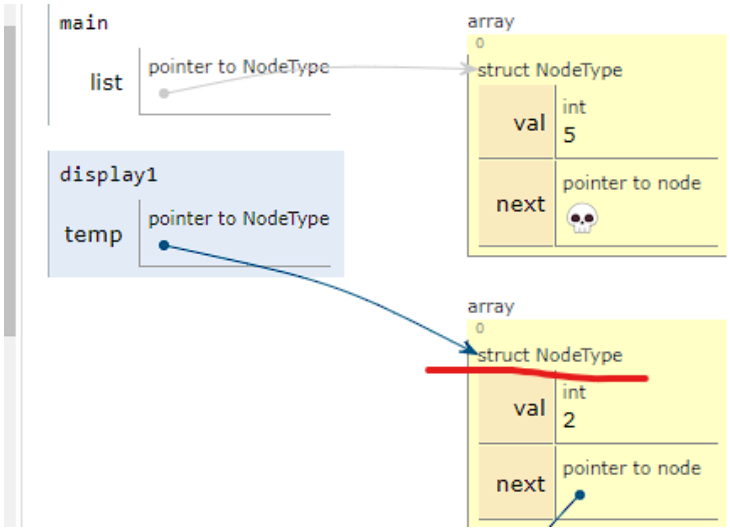
When or What's the use?

- ⇨ When you are MANIPULATING the **node** itself by Traversing
  ex. Deleting the **node**, Adding the **node** and Etc.

- ⇨ Single pointer – Can only manipulate the values inside the structure not the node
- ⇨ Double Pointer – Can manipulate all

TRAVERSING DIFFERENCE

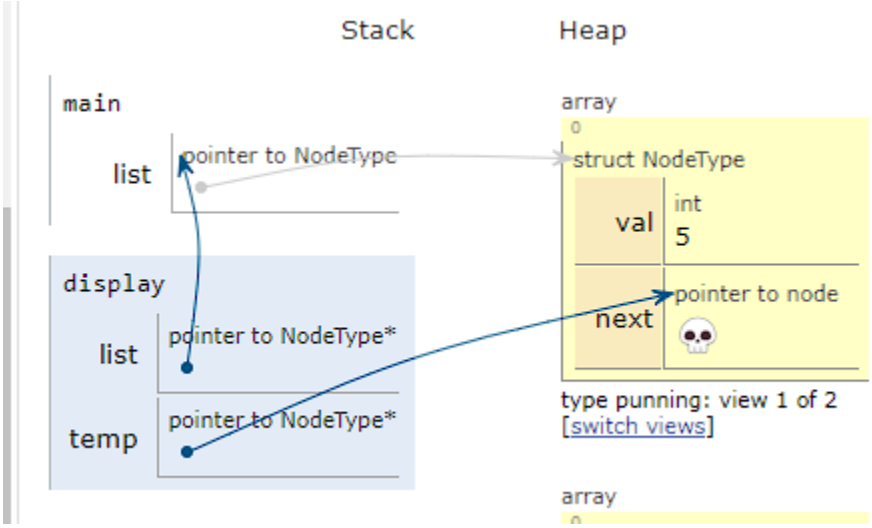
q = q->link; (S.P.)




```
void display1(NodeType *temp){

  while(temp != NULL){
    printf("%d ",temp->val);
    temp = temp->next;
  }

}
```

- ⇨ Jumps to the next node
- ⇨ Doesn't need any temporary as its node wont change except if u are changing a certain value on a variable

p = &(*p)->link; (D.P.)




```
void display(NodeType **list){
  NodeType **temp = list;
  while(*temp != NULL){
    printf("%d ",(*temp)->val);
    temp = &(*temp)->next;
  }

}
```

- ⇨ “&” means the address of that next node which means you can manipulate what’s inside that node or that node itself
- ⇨ Always need a temporary Node so your main Node wont change its node position

HOW DOES DOUBLE POINTER WORKS ?

Example: NodeType **link , *temp;

link => address of what is it pointing to

*link => value

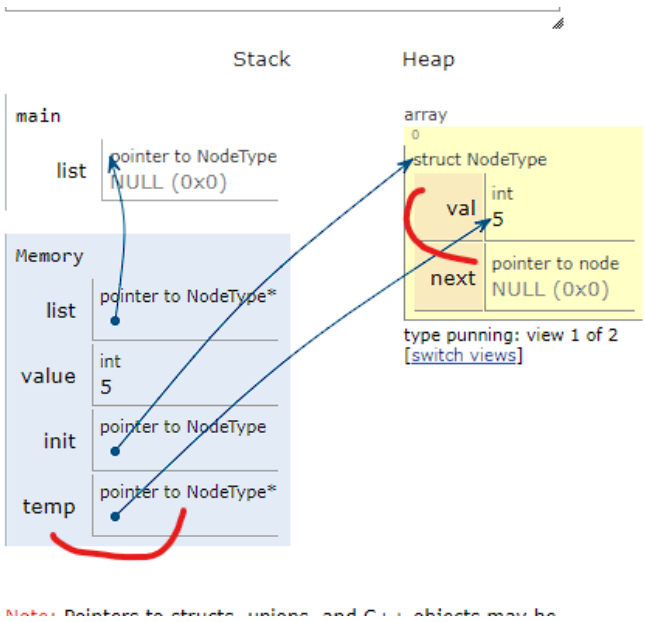
&link => address of link

link = temp;

- ⇨ Link is pointing to what is temp pointing to but link will not be pointing to list anymore which is its main node;
- ⇨ If link = &temp then link is pointing to a pointer

Ex. visualization

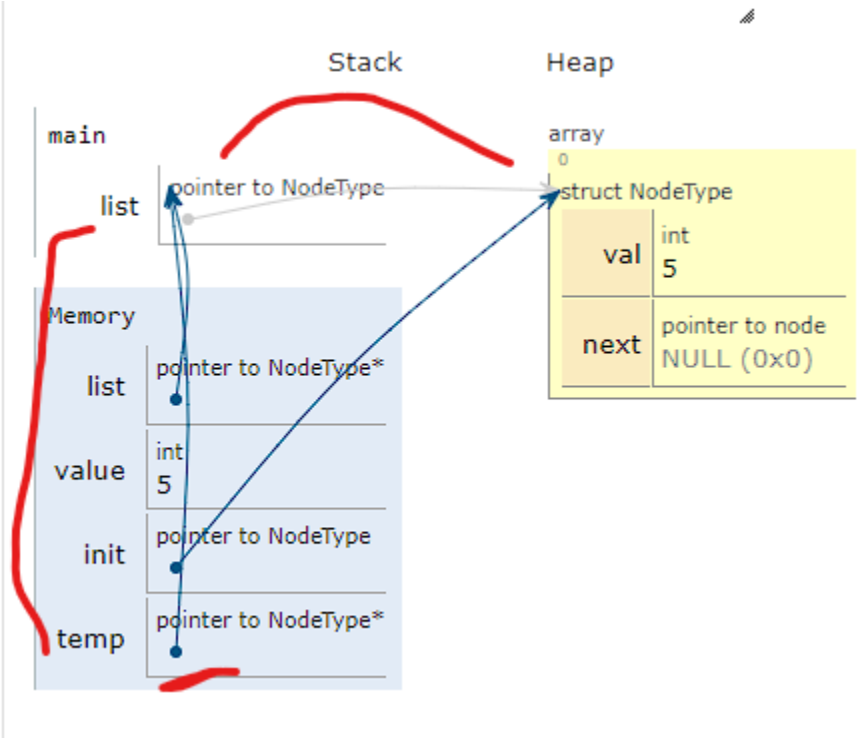
```
void insertRear(NodeType **list, int value){
    NodeType *init = malloc(sizeof(NodeType));
    init->val = value;
    init->next = NULL;

    NodeType** temp;
    for(temp = list; *temp != NULL; temp = &(*temp)->next){}
    temp = init;
}
```



*link = temp;

- ⇨ Link will point to its main node and main node will point to a node which temp is pointing to, then both are pointing to the same node

- ⇨ De – referencing, link getting the value of temp using *




```
void Memory(NodeType **list, int value){
    NodeType *init = malloc(sizeof(NodeType));
    init->val = value;
    init->next = NULL;

    NodeType** temp;
    for(temp = list; *temp != NULL; temp = &(*temp)->next){}
    *temp = init;
}
```

Refer to pythontutor c to visualize, since I cant explain properly ;')