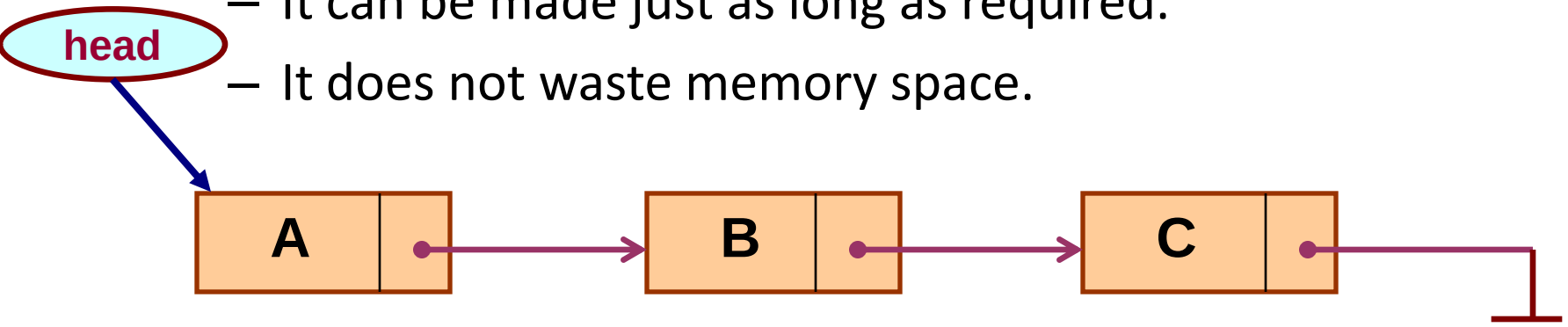
Linked List

Introduction

- A linked list is a data structure which can change during execution.
  - Successive elements are connected by pointers.
  - Last element points to `NULL`.
  - It can grow or shrink in size during execution of a program.
  - It can be made just as long as required.
  - It does not waste memory space.



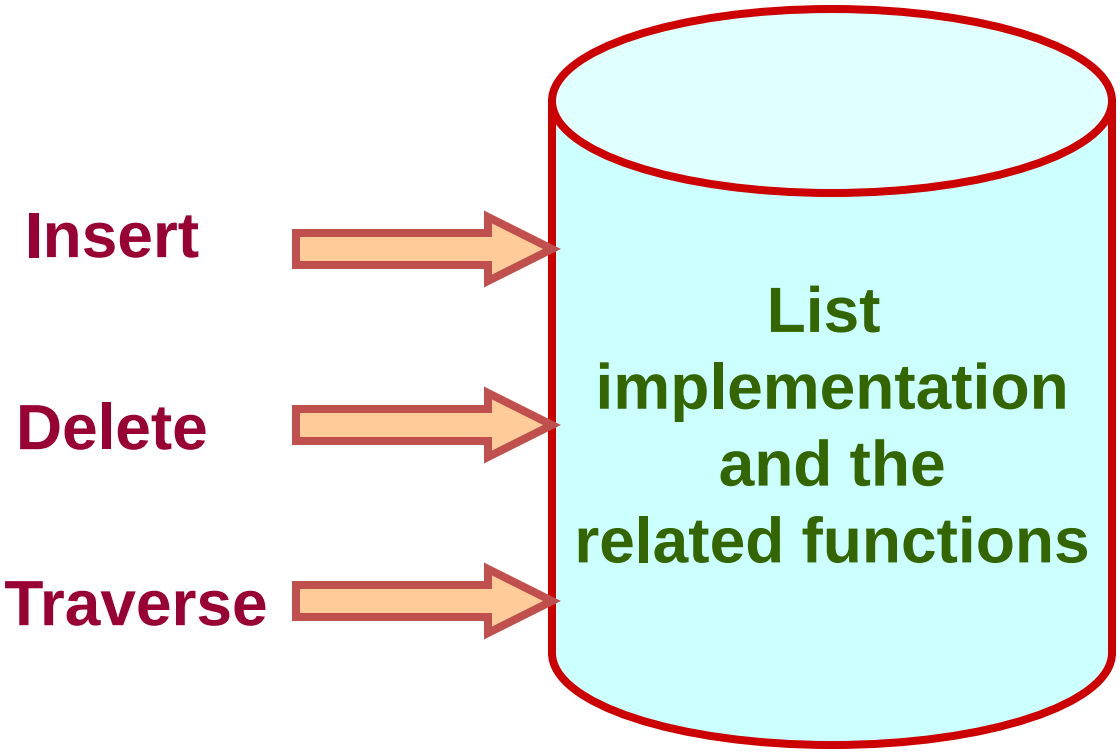
- Keeping track of a linked list:
  - Must know the pointer to the first element of the list (called *start*, *head*, etc.).

- Linked lists provide flexibility in allowing the items to be rearranged efficiently.
  - Insert an element.
  - Delete an element.

Basic Operations on a List

- Creating a list
- Traversing the list
- Inserting an item in the list
- Deleting an item from the list
- Concatenating two lists into one

Conceptual Idea

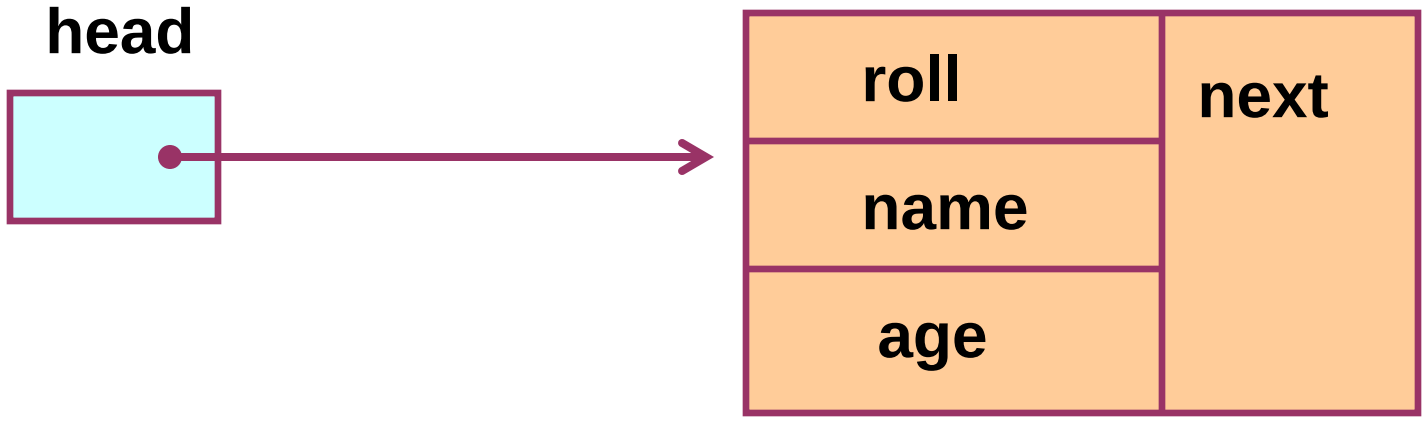


Creating a List

How to begin?

- To start with, we have to create a node (the first node), and make `head` point to it.

```
head = (node *)
  malloc(sizeof(node));
```



Linked List Node

```
struct node {
    int value;
    struct node *next;
};
```

Main function

```
void  main()
{
    struct node *last=NULL;
    struct node *head= (struct node*) malloc(sizeof(struct node));
    head->value=10;
    head->next=NULL;
   last=head;
last=append(last,110);
last=append(last,510);
last=append(last,610);
last=append(last,710);
print(head);
}
```

Append Function(Insert a new node at End)

```
struct node* append(struct node* last,int data)
        {
         struct node *h1= (struct node*) malloc(sizeof(struct node));
            h1->value=data;
            h1->next=NULL;
            last->next=h1;
            last=h1;
         return last;
        }
```

Print/Traverse

- Note: Here, *head* pointer is a local pointer variable within the scope of *print* function. So, traversing through the head pointer does not change the base address or the address of first node in the *main* function.
- So, the *head=head->next* only signifies the movement of head pointer(i.e., local scope) within the print function

```
void print(struct node* head)
{
     while(head !=NULL)
        {
                    printf("%d \n %d \n", head->value,head->next);
                    head=head->next;
        }
}
```

Output:

```
10
 12522544
110
 12522576
510
 12522608
610
 12522640
710
 0

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

Contd.

- If there are n number of nodes in the initial linked list:
  - Allocate n records, one by one.
  - Read in the fields of the records.
  - Modify the links of the records so that the chain is formed.