PES UNIVERSITY

CELEBRATING 50 YEARS

Data Structures

Dilip Kumar Maripuri

Computer Applications

PES UNIVERSITY

Data Structures

Session : Insert Operations: At Position, In Order

Dilip Kumar Maripuri
Computer Applications

Data Structures
Operations on Lists

PES UNIVERSITY

- **Insert a Node at a position in a Singly Linked List**

Algorithm Insert_Position(Head, data, position):

1. Set *curr* ← Head
2. Initialize *i* ← 0
3. Create a new node, *new_node* ← create_node(data)
4. **if** *new_node* = NULL **then**
   - 4.1 **return** Head // Memory allocation failed, return unchanged Head
5. **end if**
6. **if** position = 0 **or** Head = NULL **then**
   - 6.1 Set *new_node.link* ← Head
   - 6.2 **return** *new_node* // New node becomes the new Head
7. **end if**

Data Structures
Operations on Lists

- **Insert a Node at a position in a Singly Linked List**

Algorithm Insert_Position(Head, data, position):

8. **for** *curr* $\neq$ NULL **and** *i* ¡ position - 1 **do**
   - 8.1 Set *curr* $\leftarrow$ *curr.link*
   - 8.2 Increment *i*
9. **end for**
10. **if** *curr* = NULL **then**
    - 10.1 Print "Out of range. Not inserted."
    - 10.2 **return** Head
11. **end if**
12. Set *new_node.link* $\leftarrow$ *curr.link*
13. Set *curr.link* $\leftarrow$ *new_node*
14. **return** Head

End Algorithm

```
1  NODE insert_position(NODE Head, int data, int
       pos) {
2      NODE curr=Head, new_node;
3      int i=0;
4
5      if (new_node == NULL)
6          return Head;
7
8      if (pos == 0 || Head == NULL)
9      {
10         new_node->link = Head;
11         return new_node;
12     }
13
14     for (curr != NULL && i < pos - 1; i++)
```

```
NODE insert_position(NODE Head, int data, int
    pos) {

\\ Continued ....

     if (curr == NULL)
     {
          printf(" Out of range. Not inserted.");
          return Head;
     }

     new_node = create_node(data)
     if(new_node != NULL)
     {
          new_node->link = curr->link;
          curr->link = new_node;}
     }
     return Head;
}
```

Data Structures
Operations on Lists

- **Insert into a Sorted Singly Linked List**

Algorithm Insert_In_Sorted_List(head, data):

1. Set *curr* $\leftarrow$ head
2. Create a new node, *new_node* $\leftarrow$ create_node(data)
3. **if** *new_node* = NULL **then**
   - 3.1 **return** head // Memory allocation failed, return unchanged head
4. **end if**
5. **if** head = NULL **or** head.data $\geq$ data **then**
   - 5.1 Set *new_node.link* $\leftarrow$ head
   - 5.2 **return** *new_node* // New node becomes the new head
6. **end if**

Operations on Lists

- **Insert into a Sorted Singly Linked List**

Algorithm Insert_In_Sorted_List(head, data):

7. Set *curr* $\leftarrow$ head
8. Create a new node, *new_node* $\leftarrow$ create_node(data)
9. **while** *curr.link* $\neq$ NULL **and** *curr.link.data* $<$ data **do**
    - 9.1 Set *curr* $\leftarrow$ *curr.link*
10. **end while**
11. Set *new_node.link* $\leftarrow$ *curr.link*
12. Set *curr.link* $\leftarrow$ *new_node*
13. **return** head

End Algorithm

```
1  NODE insert_in_sorted_list(NODE Head, int data)
   {
2      NODE curr = Head, new_node = create_node(
           data);
3      if (new_node == NULL)
4          return Head;
5
6      if (Head == NULL || Head->data >= data) {
7          new_node->link = Head;
8          return new_node;                    }
9
10     while (curr->link != NULL && curr->link->
           data < data)
11         curr = curr->link;
12
13     new_node->link = curr->link;
14     curr->link = new_node;
15
16     return Head;
17 }
```

Thank You

Dilip Kumar Maripuri
Associate Professor
Department of Computer Applications
dilip.maripuri@pes.edu
8073212026