UNIVERSITY OF DIYALA

Data Structure

Lecture 2:
Pointer, Structure, Linked list

INSTRUCTOR
ALI A. AL-ANI

UNIVERSITY OF DIYALA

Pointer

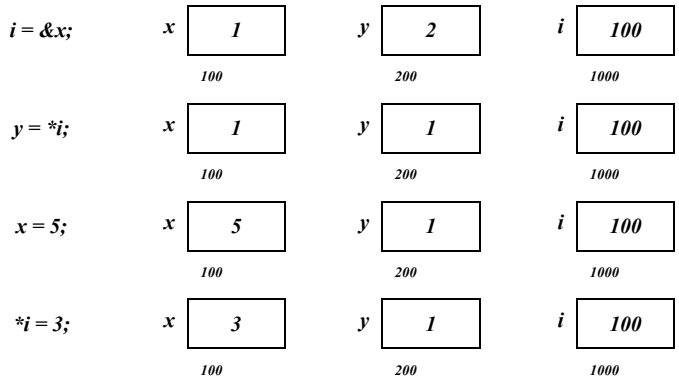
- **A pointer** is a variable whose value is the address of another variable. Like any variable or constant, we must declare a pointer before we can work with it. Some C++ tasks are performed more easily with pointers, and other C++ tasks, such as dynamic memory allocation, cannot be performed without them. The general form of a pointer variable declaration is: ***type *var-name;***
- ***To declare a pointer to a variable do: int A = 5; int *p; *p = &A***
- ***NOTE:*** We must associate a pointer to a particular type: You can't assign the address of a ***int*** to a ***float***, ***For instance:***
  - ***int A=10; float B= 5.2; int *p ; *p = &B;*** // Wrong assignment;
  - ****p = &A;*** // Correct pointer assignment.

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Pointer

- *Consider the effect of the following code:*
  - *int x = 1, y = 2;*
  - *int *i;*
  - *i = &x;*
  - *y = *i;*
  - *x = 5;*
  - **i = 3;*



UNIVERSITY OF DIYALA

Pointer

- ***Note:*** When a pointer is declared it does not point anywhere. You must set it to point somewhere before you use it.
  - *int *i;*
  - **i = 50;* //will generate an error (program crash!!).
- ***The correct use is:***
  - *int *i;*
  - *int x;*
  - *i = &x;* // setting the pointer
  - **i = 100;*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Structures

- A **Structure** in C++ is a group of data elements grouped together under one name. These data elements, known as **members**, can have different types and different lengths. It is a user defined data type which allows us to combine data items of different kinds. They are most commonly used for record-oriented data.
- C/C++ arrays allow us to define variables that combine several data items of the same type, but structure is another user defined data type which allows us to combine data items of different types.

UNIVERSITY OF DIYALA

Structures

- A structure is usually declared before main( ) function. In such cases the structure assumes global status and all the functions can access the structure. The members themselves are not variables, they should be linked to structure variables in order to make them meaningful members.
- ***Example: How to declare a structure***

```
struct Rectangle  // This is name for structure
{                 float Length;
                  int width;
                  double area;
};
```

UNIVERSITY OF DIYALA

Structures

- ***There are two ways to access members within a struct.***

1. ***Use "." dot operator:*** The link between a member and a variable is established using the membership operator '.' Which is also known as dot / membership operator.
2. ***Use " -> " The arrow operator:*** The arrow operator lets us access a member of the structure pointed to by a pointer. This type of structure called ***Self-referential structures.*** Self-referential ***structs*** can be used to create linked data structures:

- The simplest type of self-referential list structure is a sequence of elements. Each element contains some data, and a reference, often called a pointer, to the next element in the list. We can do that in c++ as shown in the following example which will allow us to create a list of integer values.

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


UNIVERSITY OF DIYALA

Linear Data Structure: Linked List

- The array is the natural data structure to use when working with lists. Arrays provide fast access to stored items and are easy to loop through. However, it has at least these limitations:

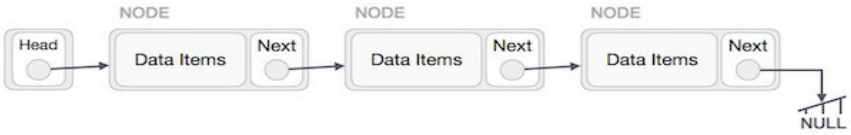
1. ***Fixed size:*** The size of the array is static (specify the array size before using it).
2. ***One block allocation:*** To allocate the array at the beginning itself, sometimes it may not be possible to get the memory for the complete array (if the array size is big).
3. ***Complex position-based insertion:*** To insert an element at a given position we may need to shift the existing elements. If the position at which we want to add an element is at the beginning then the shifting operation is more expensive.

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Linked List

- The limitations of the array can be overcome by using ***linked list structure***. A linked list is a sequence of data structures, which are connected together via links.
- Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after. The link is basically a pointer to another structure that contains a value and another pointer to another structure, and so on.
- A node is made up of two fields as we can see below:



UNIVERSITY OF DIYALA

Linked List

- ***Advantages of Linked Lists***
  1. The linked lists is that they can be *expanded* in constant time.
  2. Linked List is Dynamic data Structure.
  3. Linked List can grow and shrink during run time.
  4. Insertion and Deletion Operations are Easier
  5. Efficient Memory Utilization, i,e no need to pre-allocate memory.
- ***Disadvantages of Linked Lists***
  1. The main disadvantage of linked lists is *access time* to individual elements. Linked lists takes **O(*n*)** for access to an element in the list in the worst case.
  2. linked lists wastes memory in terms of extra reference points.

Department of Computer Science
College of Science

Linked List

- ***The following operations make linked lists:***
- ***Main Linked Lists Operations***
  1. **Insert:** inserts an element into the list.
  2. **Delete:** removes and returns the specified position element from the list.
- ***Auxiliary Linked Lists Operations***
  1. **Delete List:** removes all elements of the list (disposes the list).
  2. **Count:** returns the number of elements in the list.
  3. **Find** *nth* node from the end of the list.

UNIVERSITY OF DIYALA

Linked List: Types

- ***Following are the various types of linked list.***

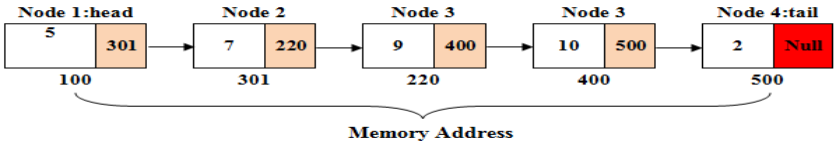
1. ***Simple Linked List:*** *Item navigation is forward only.*
2. ***Doubly Linked List:*** *Items can be navigated forward and backward.*
3. ***Circular Linked List:*** *Last item contains link of the first element as next and the first element has a link to the last element as previous.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Linked List: Types

1. ***Singly Linked Lists:*** The singly linked list consists of a number of nodes in which each node has a ***next*** pointer to the following element. The link of the last node in the list is NULL, which indicates end of the list.



UNIVERSITY OF DIYALA

Linked List

- The structure of a node in a linked list is implemented as **struct** in C++. The following is a type declaration for a linked list of integers:

```
struct node {
int data;
struct node* link;
} *p, *q, *r, *start;
```

- In the above node structure we have defined two fields in structure:

1. ***Data****:* It is Integer part for storing data inside Linked List Node.
2. ***Link :*** It is pointer filed which stores the address of another structure (i.e node).

- Also we add these pointers ***{struct node *p, *q, *r, *start;}*** that will be used in the program.

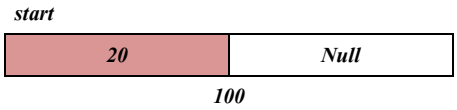
Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Linked List

- *To create a linked list with ONE integer node the steps are:*

```
struct node {
int data;              //Declare the data type for a node.
struct node* link;
} *p,  *start;         //Declare a pointer to the head of the linked list.
P  = new node          //Create the first node as a dynamic variable using the new operator.
P->data = 20;          //Fill in the data for the node.
P->link = Null;
Start = P ;            //Make the next pointer pointing to null, denoting the end of the list.
```



UNIVERSITY OF DIYALA

Linked List

- *The following subprogram create the link list with N nodes. Lastly, store the address in the last field of the last node as NULL.*

```
void  createNnode(struct node*start)
{  int n, i;
p = new node;   start = p;
cout<<"How many elements you like to create"<<endl;    cin >> n;
for ( i = 0; i < n; i++ ){
        cin>>p->data;
         if ( i != n-1 )    q = new node;
           else             q = NULL;
      p->link = q;          p = q;    }          }
```

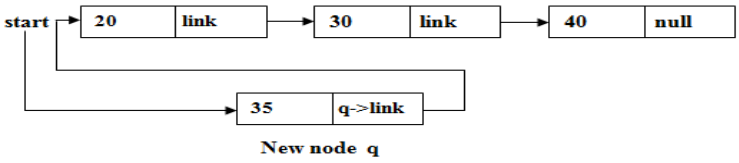
Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Linked List: Insert

- Insertion into a singly-linked list has two special cases. It's insertion a new node before the head (to the very beginning of the list) and after the tail (to the very end of the list). In any other case, new node is inserted in the middle of the list and so, has a predecessor and successor in the list. There is a description of all these cases below.
- ***Sub-program to insert node at the beginning of the linked list***

```
void insertfirst()
{  q=new node;
   q->data = 35;
   q->link=start;
   start=q;
}
```

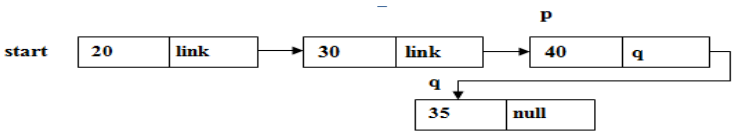


UNIVERSITY OF DIYALA

Linked List: Insert

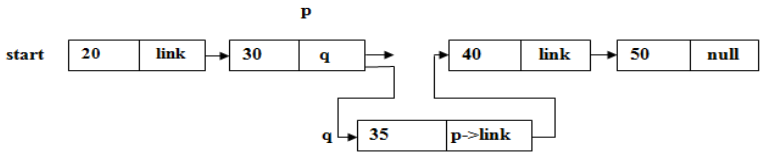
- ***Sub-program to insert node at the end of the linked list***

```
void insertafter()
{  q=new node;
   q->data = 35;
   p->link=q;
   q->link=null;   }
```



- ***Sub-program to insert node at the middle of the linked list***

```
void insertafter(struct node*p)
{  q=new node;
   q->data = 35 ;
   q->link=p->link; [*]
   p->link=q;     }
```



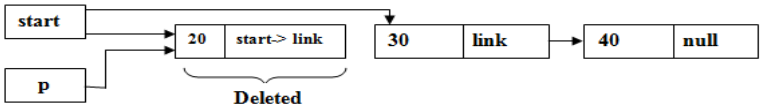
Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Linked List: Delete

- There are three cases, which can occur while removing the node. These cases are similar to the cases in add operation. We have the same three situations, but the order of algorithm actions is opposite.
- ***Sub-program to delete the first node from the linked list***

```
void deletefirst()
{
p=start;
start=(start)->link;
delete(p);
}
```

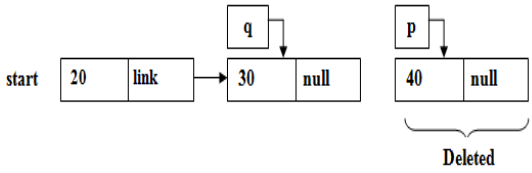


UNIVERSITY OF DIYALA

Linked List: Delete

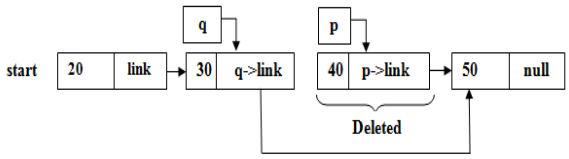
- ***Sub-program to delete the last node from the linked list***

```
void deletelast()
{ p=start;
if(p->link==NULL) { delete(p); start=NULL; }
else { while(p->link!=NULL) { q=p; p=p->link; }
q->link=NULL; delete(p);}}
```



- ***Sub-program to delete the middle node from the linked list***

```
void deletelist(int value) // value = 40
{   p=start;
while(p->data!=value) { q=p; p=p->link; }
q->link=p->link;        delete(p);
}
```



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Linked List: Print

- ***Sub-program to display all nodes in the linked list***

```
void displaylist(struct node*start)
{
p=start;
cout<<"the list:"<<endl;
while(p->link!=NULL)
{
cout<<endl<<p->data;
p=p->link;
}}
```


UNIVERSITY OF DIYALA

Linked List: Assignment

1. *What is the **Doubly Linked List? E**xplain? What are the operations of the **Doubly Linked List** in the computer? **Write all the sub program.***

2. *What is the **Circular linked list? E**xplain? What are the operations of the **Circular linked list** in the computer? **Write all the sub program.***

3. *What are the main difference between **Single linked list**, **Doubly Linked List and Circular linked list** ? List all in detail.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

The End