University of Anbar
College of Computer Science
and Information Technology
Computer Network Systems
Department

Data Structures

Lecture Nine

Second Stage

First Course - 2024-2025

Muhmmad shihab muayad Shihab

Master of Computer Engineering

muhmmad.shihab@uoanbar.edu.iq

Storage Allocation

There are two types of storage allocation depending on the structure of the data:

1- Sequential Allocation Storage

Is the simples way to store lists in memory sequentially, and from the Base address which is the first location of the list, we can know the location of any item in the list.

Advantages

1- Simple in representation
2- Take less memory space
3- Efficient in random access

Disadvantages

1- Hard to apply addition and deletion
2- Number of elements must be predefined

2- Dynamic Allocation Storage

The second way to store lists is to use link (or pointer), each element contain the location of the next element, so elements may not stored sequentially in memory. Each element (node) consist of 2 parts:

1) Data
2) pointer (link) to the next address

Advantages

1- insertion and deletion is easy to implement (not need shifting)
2- Easy to merge and split by only change the pointers

Disadvantages

1- Take more memory space

2- To access any element randomly, we must start from the beginning of the list

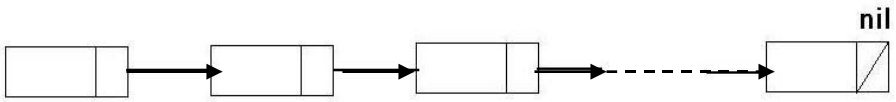
Types of lists

1) Non-Linked List :- do not use pointers it's structure, it use the vectors and array for representing it's structure.
2) Linked Lists :- a list has been defined to contain an ordered list of elements, each element (node) contains a link or pointer to the next node.

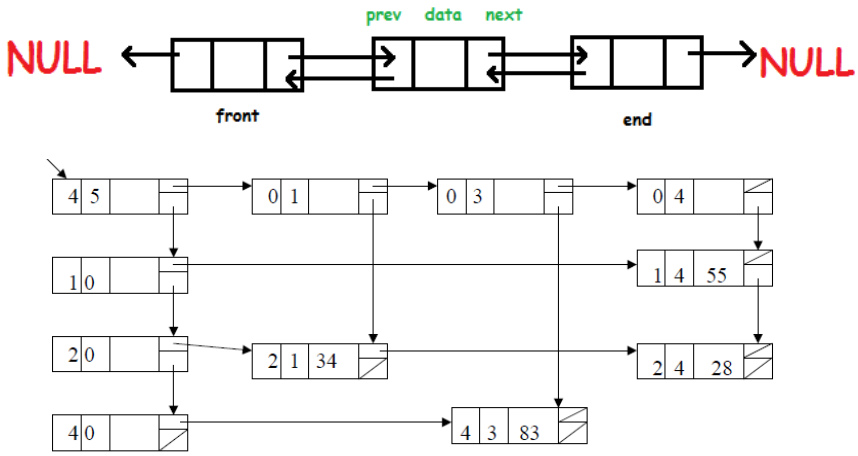
Linked lists:- A list that use pointers or link to refer to the elements of data structures, in a way that element which have logically adjacent need not to be physically adjacent in memory.

Types of linked lists:-

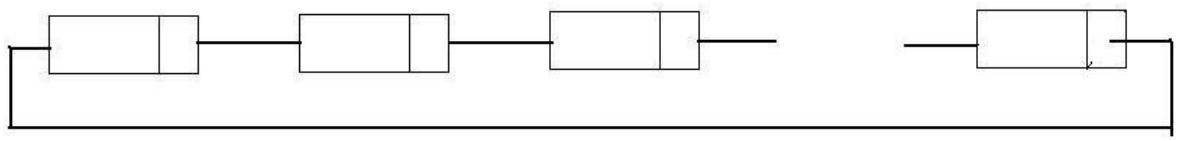
1-**Single Linked List**:- is a list contains set of elements, and each element (node) contain a link or pointer to the next node.



2-**Multi Linked List**:- a list has more than one pointer, like doubly linked list which has two pointers pointing to the previous and next node.



3-**Circuler Linked List**:- a list that last node points to the first node.



Operation on Lists

1- Insertion

2- Deletion

3- Search

4- Change

Examples:

- Print the even number in linked list

```
void print(tnod &f)
{
        nod *p;
        p=f;
        cout<<"\nresult: ";
        while(p!=0)
        {
                if(p->info%2==0)
                cout<<" "<<p->info;
                p=p->next;}
        cout<<"\n ";
}
```

- Print the prime number in linked list

```
void print(tnod &f)
{
        nod *p;
        p=f;
        cout<<"\nresult: ";
        while(p!=0)
        {
                int x = p->info;
                prime(x);
                p=p->next;}
        cout<<"\n ";
```

```
}
void prime(int n)
{
        int n, i, m=0, flag=0;
        m=n/2;
         for(i = 2; i <= m; i++)
        {
            if(n % i == 0)
            {
        flag=1;
        break;
    }
  }
  if (flag==0)
     cout << "Number is Prime:"<< n <<endl;
}
```

- Calculate odd and even numbers in linked list

```
void print(tnod &f)
{
        int ev=0,od=0;
        nod *p;
        p=f;
        cout<<"\nresult: ";
        while(p!=0)
        {
                if (p->info%2==0)
                        ev=ev+1;
                else
                        od=od+1;
                p=p->next;
        }
        cout<<"\n ";
        cout<<"even ="<<ev<<endl<<"odd="<<od;
}
```

- Calculate odd, even, positive and negative numbers in linked list

```
void print(tnod &f)
{
        int sum=0,odd=0,even=0,pos=0,nig=0;
        nod *p;
        p=f;
        cout<<"\nresult: ";
        while(p!=0)
        {
                s=s + p->info;
                if (p->info%2==0)
                  even+=1;
                else
                        odd+=1;
                if (p->info<0)
                 nig+=1;
                 else
                 pos+=1;
                p=p->next;}
        cout<<"\n   "<<odd<<"\n   "<<even<<"\n   "<<pos<<"\n
"<<nig<<"\n "<<sum;
}
```

- Change data node of linked list according to it data:

```
void print(tnod &f)
{
        nod *p;
        p=f;
        int x,y;
        cout<<"Enter data for searching in linked list";
        cin>>x;
        cout<<"Enter data to change its: ";
        cin>>y;
        while(p!=0)
        {
                if (p->info==x)
                        p->info=y;
                p=p->next;
        }
```

```
        cout<<"\n ";
    }
```

References:

- Frank Carrano, D.J. Henry: Data Abstraction and Solving with C++, 2012, 6th edition, Pearson Education, Inc.
- Mark Allen Weiss: Data Structures and Algorithm Analysis in C++, 2014, 4th edition, Pearson Education, Inc.