University of Technology
الجامعة التكنولوجية

Computer Science Department
قسم علوم الحاسوب

Data Structure
هياكل البيانات

Dr. Dena Kadhim Muhsen
د. دينا كاظم محسن

cs.uotechnology.edu.iq

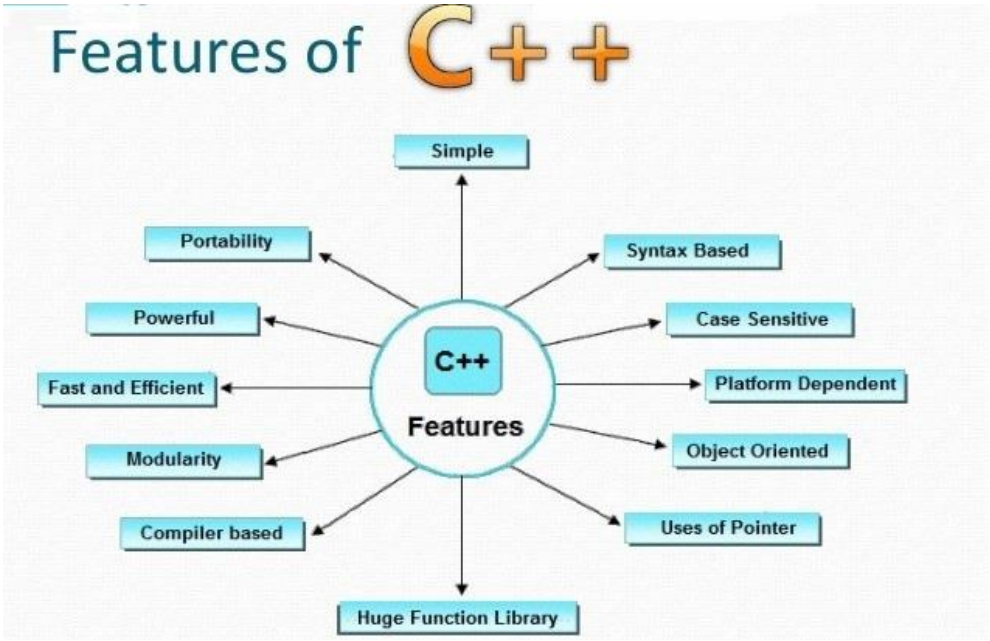
Lecture: 1

Intoduction

C++ Programming Language

C++ is a general-purpose object-oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of the C language. It is therefore possible to code C++ in a "C style" or "object-oriented style." In certain scenarios, it can be coded in either way and is thus an effective example of a hybrid language.

C++ is considered to be an intermediate-level language, as it encapsulates both high- and low-level language features. Initially, the language was called "C with classes" as it had all the properties of the C language with an additional concept of "classes." However, it was renamed C++ in 1983.



Lecture: 1

Simple Programs

A Hello World Program

```
1.  #include <iostream.h>
2.  void main() {
3.          cout << "Hello world";
4.  }
```

The Area of Circle

```
1.  #include<iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  float r,area;
6.          cout<< "\nEnter radius of circle : ";
7.  cin>>r;
8.  area = 3.14*r*r;
9.          cout<<"Area of circle : "<<area;
10. }
```

C++ If Example

If (condition) {
//code to be executed
}

Even Number

```
1.  #include <iostream>
2.  using namespace std;
3.  int main () {
4.       int num = 10;
5.  if (num % 2 == 0)
6.  {
7.     cout<<"It is even number";
8.   }
9.   }
```

Absolute Value of Number

```
1.  #include<iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int a;
6.  cout<<"Enter any number:";
```

```
7.    cin>>a;
8.          if(a>0)
9.             cout<<"The absolute value of number is:"<<a;
10.         else
11.            cout<<"The absolute value of number is:"<<-(a);
12.   return 0;
13.   }
```

C++ switch

```
switch(expression){
case value1:
 //code to be executed;
 break;
case value2:
 //code to be executed;
 break;
.....

default:
 //code to be executed if all cases are not matched;
 break;
}
```

Check the grade of Student

```
1.  #include <iostream>
2.  using namespace std;
3.  int main () {
4.        int num;
5.        cout<<"Enter a number to check grade:";
```

```
6.          cin>>num;
7.             switch (num)
8.             {
9.                case 10: cout<<"It is 10"; break;
10.               case 20: cout<<"It is 20"; break;
11.               case 30: cout<<"It is 30"; break;
12.               default: cout<<"Not 10, 20 or 30"; break;
13.            }
14.      }
```

Lecture: 2

C++ For Loop

```
For (initialization; condition; incr/decr){
//code to be executed
}
```

Print numbers from 1 to 10

```
1.  #include <iostream>
2.  using namespace std;
3.  int main() {
4.      for(int i=1;i<=10;i++){
5.  cout<<i <<"\n";
6.  }
7.  }
```

Find Factorial of a number

Factorial on n = 1*2*3*...*n

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int i, n, factorial = 1;
6.  cout << "Enter a positive integer: ";
7.  cin >> n;
8.  for (i = 1; i <= n; ++i) {
9.  factorial *= i;
10.}
11.cout<< "Factorial of "<<n<<" = "<<factorial;
12.return 0;
```

13.}

C++ While loop

```
While (condition){
//code to be executed
}
```

Print number from 1 to 10

```
1.  #include <iostream>
2.   using namespace std;
3.   int main() {
4.   int i=1;
5.         while(i<=10)
6.        {
7.            cout<<i <<"\n";
8.            i++;
9.          }
10.    }
```

C++ Do-While Loop

```
do {
//code to be executed
}while (condition);
```

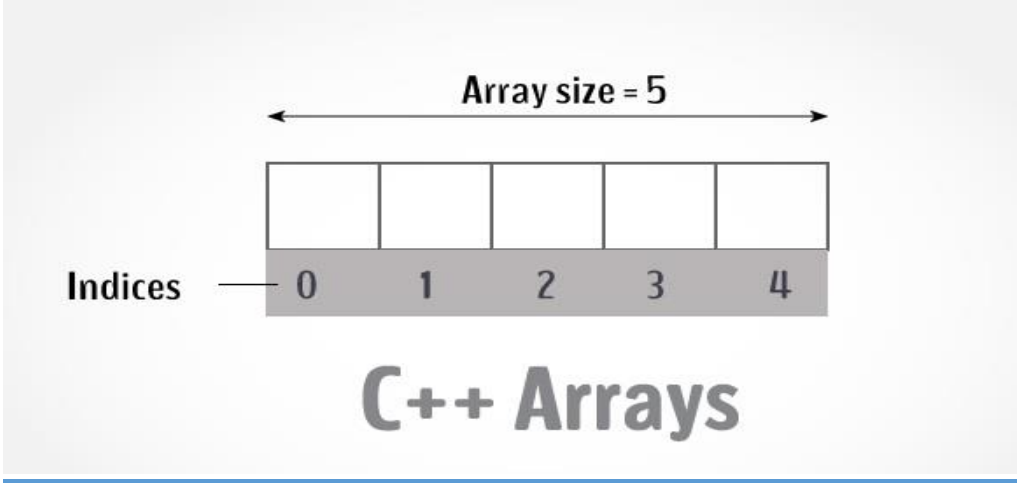
Max and Min number

```
1.   #include<iostream>
2.   using namespace std;
```

```
3.  int main()
4.  {
5.  int n, max=0, min=32767;
6.  char choice;
7.  do
8.  {
9.   cout<<"Enter number : ";
10.  cin>>n;
11.  if(n>max)
12.  max=n;
13.  if(n<min)
14.  min=n;
15.  cout<<"Do you want to Continue(y/n)? ";
16.  cin>>choice;
17.  }while(choice=='y' || choice=='Y');
18. cout<<"Maximum Number :"<<max<<"\nMinimum Number :"<<min;
19. return 0;
20. }
```

C++ Array

Lecture: 2



Sum of numbers in an array

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int numbers[5], sum = 0;
6. cout << "Enter 5 numbers: ";
7. for (int i = 0; i < 5; ++i)
8. {
9. cin >> numbers[i];
10.sum += numbers[i];
11.}
```

```
12.cout << "Sum = " << sum << endl;
13.   return 0;
14.}
```

C++ Function and Array

➢ Passing One-dimensional Array to a Function

Display the marks of students

```
1.  #include <iostream>
2.  using namespace std;
3.  void display(int m[5])
4.  {
5.        cout << "Displaying marks: "<< endl;
6.  for (int i = 0; i < 5; ++i)
7.  {
8.        cout << "Student "<< i + 1 <<": "<< m[i] << endl;
9.  }  }
10.int main()
11.{
12.int  m[5] = {88, 76, 90, 61, 69};
13.display(m);
14.}
```

Lecture: 2

➢ Passing Multidimensional Array to a Function

★ *Display the elements of Multidimensional Array* ★

```
1.  #include  <iostream>
2.  using namespace std;
3.  void display(int n[3][2])
4.   {
5.        cout << "Displaying Values: " << endl;
6.   for(int i = 0;  i < 3; ++i)
7.   {
8.   for(int j = 0; j < 2; ++j)
9.    {
10.  cout << n[i][j] <<"     " ;
11.   }
12.   }
13.   }
14.   int main()
15.  {
16.  int num[3][2] = {
17.            {3, 4},
18.            {9, 5},
```

```
19.             {7, 1}
20.};
21.  display(num);
22. }
```

Data structure:

A collection of data elements whose organization is characterized by accessing operations that are used to store and retrieve the individual data elements.

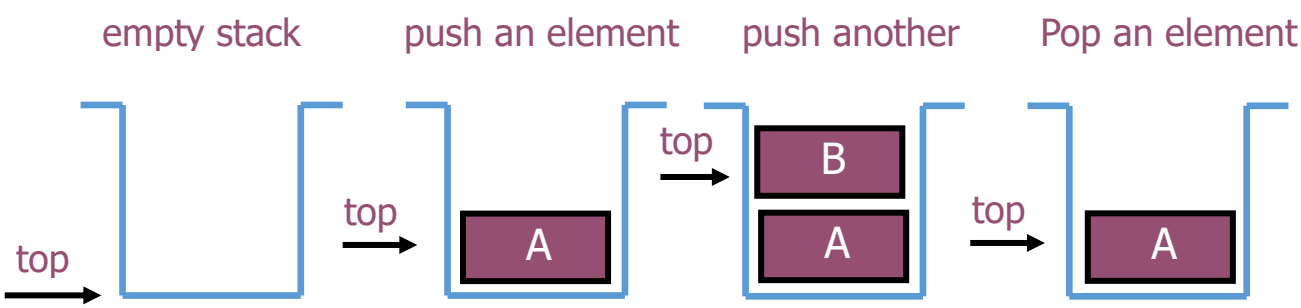
- **A stack** is a basic computer science data structure and can be defined in an abstract, implementation-free manner, or it can be generally defined as a linear list of items in which all additions and deletion are restricted to one end that is Top.

Basic operations in stack

Stacks are a type of container adaptors with LIFO (Last In First Out) type of working, where a new element is added at one end and (top) an element is removed from that end only.

The functions associated with stack are:

empty() – Returns whether the stack is empty
size() – Returns the size of the stack
top() – Returns a reference to the top most element of the stack
push(g) – Adds the element at the top of the stack
pop() – Deletes the top most element of the stack



push, pop, and print in the stack

```
#include<iostream.h>
void push(int st[6],int &top)
{
int item;
if(top==5)
   cout<<”over flow”;
else
{
    cin>>item;
++top;
st[top]=item;
}
}
void pop(int st[6],int &top)
{
int item;
if(top==-1)
cout<<"under flow";
else
{
item=st[top];
--top;
}
}
void printst(int st[6],int top)
{
int i;
```

```
for(i=top;i>=0;--i)
cout<<st[i]<<endl;
}
void main()
{
int x,st[6],top=-1;
do
{
cout<<"1-push\n";
cout<<"2-pop\n";
cout<<"3-printst\n";
cout<<"4-exit\n";
cout<<"enter your choich\n";
cin>>x;
switch(x)
{
case 1:push(st,top);break;
case 2:pop(st,top);break;
case 3:printst(st,top);break;
default:cout<<"error\n";
}
}
while(x!=4);
}
```

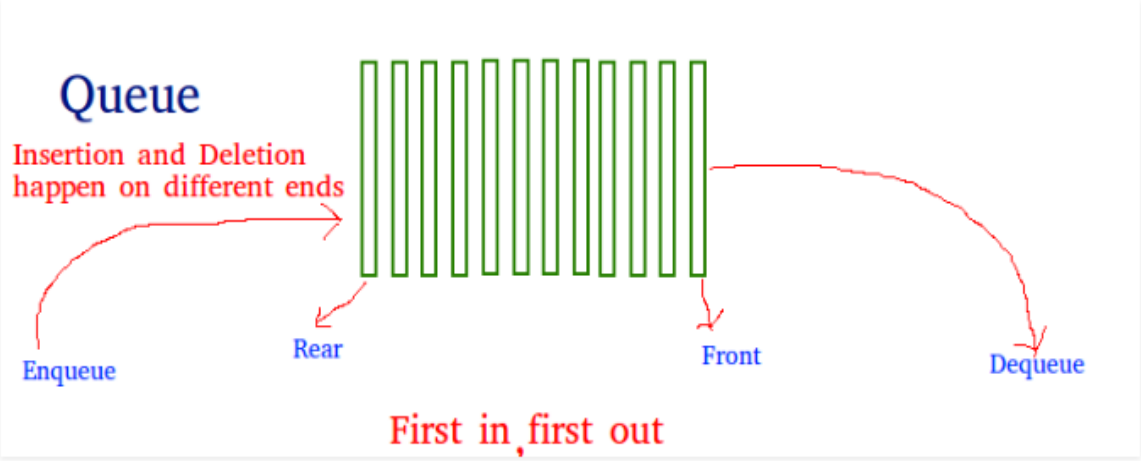
Using tow stack and reverse the elements of first stack

1. #include<iostream.h>
2. void main()

```
3.  {int a, st1[6],st2[6],top1,top2,item,i;
4.  top1=top2=-1;
5.  for(i=1;i<=6;i++)
6.  {   cin>>item;
7.  top1++;
8.  st1[top1]=item;
9.  }
10.    cout<<"the st1=\n";
11.for(i=top1;i>=0;i--)
12.    cout<<st1[i]<<'\n';
13.for(i=0;i<=6;i++)
14.{item=st1[top1];
15.top1--;
16.top2++;
17.st2[top2]=item;
18.}
19.     cout<<"\nthe st2=\n";
20.for(i=top2;i>=0;i--)
21.     cout<<st2[i]<<'\n'; cin>>a;
22.}
```

- **A queue** is a sequence of data elements. In the sequence
  - Items can be removed only at the front
  - Items can be added only at the other end, the back

- **Basic operations in Queue**
  - Construct a queue
  - Check if empty
  - Enqueue (add element to back)
  - Front (retrieve value of element from front)
  - Dequeue (remove element from front)



Add, delete, and print in the queue

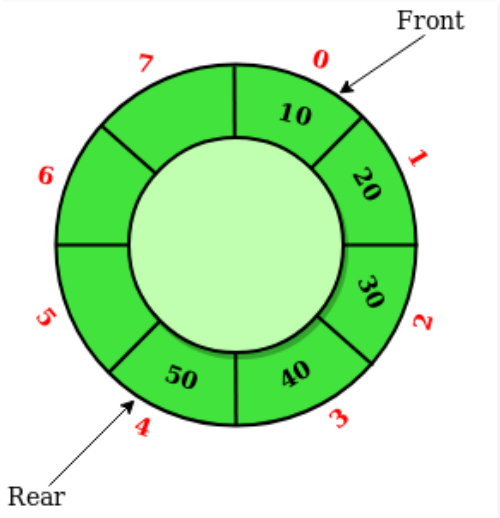
```
1.  #include<iostream.h>
2.  void insq(int q[5],int &f,int &r)          //Add function
3.  {
4.  int item;
5.  if(r==4)
6.      cout<<"over flow\n";
7.  else
8.  {
9.  cin>>item;
10. r++;
11. q[r]=item;
12. }
13. if(f==-1)
14. f=0;
15. }
16. void delq(int q[5],int &f,int &r)          //Delete function
17. {
18. int item;
19. if(f==-1)
20.     cout<<"under flow\n";
21. else
22. if(f==r)
23. {
24. item=q[f];
25. f=-1;
26. r=-1;
```

```
27.  }
28.  else
29.  {
30.  item=q[f];
31.  ++f;
32.  }
33.  }
34.  void printq(int q[5],int  f,int  r)                  //Print function
35.  {
36.  int i;
37.  for(i=f;i<=r ;i++)
38.       cout<<q[i]<<"   ";
39.       cout<<endl;
40.  }
41.  void exit()
42.  { exit(); }
43.  void main()
44.  {
45.  int q[5];
46.  int f=-1,r=-1;
47.  int x;
48.       cout<<"This is queue program! \n";
49.  do
50.  {
51.     cout<<endl;
52.       cout<<"1-Add\n";
53.       cout<<"2-Delete\n";
54.       cout<<"3-Print\n";
55.       cout<<"4-Exit\n";
56.       cout<<"Enter your choise : \n";
57.       cin>>x;
```

```
58.  switch(x)
59.  {
60.     case 1:insq(q,f,r);break;
61.     case 2:delq(q,f,r);break;
62.     case 3:printq(q,f,r);break;
63.     case 4:exit();
64.     default:cout<<"Error , Please enter other Choise!\n";
65.  }  //End of switch
66.  }   //End of do
67.  while(x=4);
68.  }
```

Lecture: 5

- **Circular Queue :** Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **'Ring Buffer'.**



Add, delete, and print in the circular queue

```
1. #include<iostream.h>
2. #include<stdlib.h>
3. void insert(int cq[6],int &f,int &r)
4. {
5. int item;
```

```
6.  if(f==0&&r==5)
7.  cout<<"over flow\n";
8.      else if(f-r==1)
9.      cout<<"over flow\n";
10.     else if(r==5&&f>0)
11.     {
12.     cin>>item;
13.     r=0;
14.     cq[r]=item;
15.     }
16.     else
17.     {
18.     cin>>item;
19.     r++;
20.     cq[r]=item;
21.     }
22.     if(f==-1)
23.     f=0;
24.     }
25.     void del(int cq[6],int &f,int &r)
26.     {
27.     int item;
28.     if(f==-1)cout<<"under flow\n";
29.     else
30.     if(f==r)
31.     {
32.     item=cq[f];
33.     f=-1;r=-1;
34.     }
```

```
35.  else
36.  {
37.  if(f==5&&f>r)
38.  {
39.  item=cq[f];
40.  f=0;
41.  }
42.  else
43.  {
44.  item=cq[f];
45.  f++;
46.  }
47.  }}
48.  void print(int cq[6],int f,int r)
49.  {
50.  int i;
51.  if(r>=f)
52.  {
53.  for(i=f;i<=r;i++)
54.  cout<<cq[i]<<"  ";
55.  cout<<endl;
56.  }
57.  else
58.  {
59.  for(i=0;i<=5;i++)
60.  cout<<cq[i]<<" ";
61.  for(i=0;i<=r;i++)
62.  cout<<cq[i]<<" ";
63.  cout<<endl;
```

```
64.  }
65.  }
66.  void main()
67.  {
68.  int cq[6],x,r=-1,f=-1;
69.  do
70.  {
71.  cout<<endl;
72.  cout<<"1_insert\n"<<"2_delete\n"<<"3_print\n";
73.  cout<<"enter your choice:";
74.  cin>>x;
75.  switch(x)
76.  {
77.  case 1:insert(cq,f,r);break;
78.  case 2:del(cq,f,r);break;
79.  case 3:print(cq,f,r);break;
80.  case 4:exit(0);break;
81.  default:break;
82.  }
83.  }while(x!=4);
84.  }
```

A linked list

A linked list is a dynamic data structure where each element (called a **node**) is made up of two items - the data and a reference (or pointer) which points to the next node. A linked list is a collection of nodes where each node is connected to the next node through a pointer. The first node is called a **head** and if the list is empty then the value of head is NULL.

In simple words, a linked list consists of nodes where each node contains a data field and a reference (link) to the next node in the list.

Basic Operations

Following are the basic operations supported by a list.

- **Insertion** − Adds an element in the list.
- **Deletion** − Deletes an element from the list.
- **Display** − Displays the complete list.
- **Search** − Searches an element using the given key.
- **Delete** − Deletes an element using the given key.

Types of Linked List

Following are the various types of linked list.

- **Simple Linked List** − Item navigation is forward only.
- **Doubly Linked List** − Items can be navigated forward and backward.
- **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.

insert, delete, and print in the linked list

```
#include<iostream.h>
#include<stdlib.h>
#define nill 0 ;
struct node{
int info;
struct node *next;
};
typedef node *nodeptr;
void insbeg(nodeptr &plist)  //add from begin function
{
nodeptr p;
p=new node();
cin>>p->info;
p->next=plist;
plist=p;
}
void insend(nodeptr &plist)          //add from end function
{
nodeptr p,h;
h=plist;
while(h->next !=0)
{
h=h->next;
}
cout<<"Enter the value!\n";
p=new node();
cin>>p->info;
```

```
p->next=nill;
h->next=p;
}

void insbet(nodeptr &plist)          //add from between function
{
int i=1,j;
cout<<"Enter the location of node which you want add it!"<<endl;
cin>>j;
nodeptr a,b,p;
a=plist;
while(i !=j)
{
b=a;
a=a->next;
++i;
}
cout<<"Enter the value of new node!"<<endl;
p=new node();
cin>>p->info;
b->next=p;
p->next=a;
}
void delbeg(nodeptr &plist)                //delete from begin function
{
nodeptr h;
h=plist;
plist=h->next;
delete(h);
}
void delend(nodeptr &plist)          //delete from end function
{
nodeptr h,a;
```

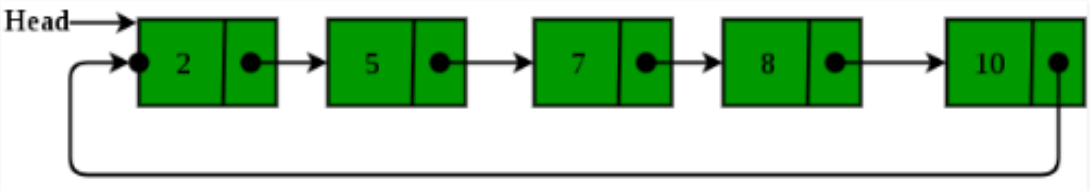
```
h=plist;
while(h->next !=0)
{
a=h;
h=h->next;
}
a->next=nill;
delete(h);
}
void delbet(nodeptr &plist)          //delete from between function
{
int i=1,j;
cout<<"Enter the location of node which you want to delete it!"<<endl;
cin>>j;
nodeptr a,b;
a=plist;
while(i !=j)
{
b=a;
a=a->next;
++i;
}
b->next=a->next;
delete(a);
}
void printpl(nodeptr &plist)          //Print function
{
nodeptr h;
h=plist;
while(h !=0)
{
cout<<h->info<<"     ";
h=h->next;
```

```
  }
  }
  void exit()                    //Exit function
  {
  exit();
}
void main()
{
cout<<"This is linear list program!\n"<<endl;
cout<<"Enter the values of the nodes!\n";
nodeptr h,p,plist;
int i;
p=new node();
cin>>p->info;
p->next=p;
plist=p;
h=p;
for(i=2;i<=6;++i){
p=new node();
cin>>p->info;
p->next=nill;
h->next=p;
h=p;}
int x;
do
{
cout<<endl;
cout<<"1-Add from begin\n";
cout<<"2-Add from end\n";
cout<<"3-Add from between\n";
cout<<"4-Delete from begin\n";
cout<<"5-Delete from end\n";
cout<<"6-Delete from between\n";
```

```
cout<<"7-Print\n";
cout<<"8-Exit";
cout<<endl<<"Enter your choice! \n";
cin>>x;
switch(x)
{
case 1:insbeg(plist);break;
case 2:insend(plist);break;
case 3:insbet(plist);break;
case 4:delbeg(plist);break;
case 5:delend(plist);break;
case 6:delbet(plist);break;
case 7:printpl(plist);break;
case 8:exit();break;
default: cout<<"Erorr! , Please try again "<<endl;
}
}
while(x=8);
}
```

Circular Linked List –

is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.



Basic Operations

Following are the important operations supported by a circular list.

- **Insertion** − Adds an element in the list.
- **Deletion** − Deletes an element from the list.
- **Display** − Displays the complete list.
- **Search** − Searches an element using the given key.
- **Delete** − Deletes an element using the given key.

insert, delete, and print in the Circular linked list

```
#include<iostream.h>
#include<stdlib.h>
struct node{
int info;
struct node *next;
};
typedef node *nodeptr;
void printplistc(nodeptr plist){
nodeptr h;
h=plist;
cout<<"The elemnts of cyrcle list are : "<<endl;
while(h->next !=plist){
cout<<h->info<<"    ";
h=h->next;}
cout<<"   "<<h->info;}
void insbeginc(nodeptr &plist){
nodeptr p,h;
h=plist;
while(h->next !=plist)
h=h->next;
p=new node();
cin>>p->info;
p->next=plist;
plist=p;
h->next=plist;
}
void insbetweenc(nodeptr &plist){
nodeptr a,b,p;
int i=0,j;
```

```
cout<<"enter the location of the new node "<<endl;
cin>>j;
a=plist;
while(i!=j){
b=a;
a=a->next;
++i;
}
p=new node();
cin>>p->info;
b->next=p;
p->next=a;
}
void insendc(nodeptr &plist){
nodeptr p,h;
h=plist;
while(h->next !=plist)
h=h->next;
p=new node();
cin>>p->info;
h->next=p;
p->next=plist;
}
void delbeginc(nodeptr &plist){
nodeptr h,p;
h=plist;
while(h->next !=plist)
h=h->next;
p=plist;
plist=plist->next;
h->next=plist;
delete(p);
}
```

```
void delbetweenc(nodeptr &plist){
nodeptr a,b;
int i=0,j;
cout<<"enter the location of the node that you want to delete it "<<endl;
cin>>j;
a=plist;
while(i !=j){
b=a;
a=a->next;
++i;
}
b->next=a->next;
delete(a);
}
void delendc(nodeptr &plist){
nodeptr a,b;
a=plist;
while(a->next !=plist){
b=a;
a=a->next;
}
b->next=plist;
delete(a);
}
void exit(){}
void main(){
cout<<"Enter the elements! "<<endl;
nodeptr h,p,plist;
int i;
p=new node();
cin>>p->info;    //Enter value of first node
p->next=p;
plist=p;
```

```
h=p;
for(i=2;i<=6;++i){     //Enter values of other nodes
p=new node();
cin>>p->info;
p->next=plist;
h->next=p;
h=p; }
int x;
do{
cout<<endl;
cout<<"1-insbeginc\n";
cout<<"2-insbetweenc\n";
cout<<"3-insendc\n";
cout<<"4-delbeginc\n";
cout<<"5-delbetweenc\n";
cout<<"6-delendc\n";
cout<<"7-printplistc\n";
cout<<"8-exit\n";
cin>>x;
switch(x){
case 1:insbeginc(plist);break;
case 2:insbetweenc(plist);break;
case 3:insendc(plist);break;
case 4:delbeginc(plist);break;
case 5:delbetweenc(plist);break;
case 6:delendc(plist);break;
case 7:printplistc(plist);break;
case 8:exit();break;
default: cout<<"error"<<endl;
}                   //end of switch
}                   //end of do
while(x=8);
}
```