```
#include <iostream>

using namespace std;

struct nod
{ //Definirea stucturii nodului listei dublu inlantuite
    nod *ant;//Adresa nodului anterior
    double inf;//Informatia memorata in nod
    nod *urm;//Adresa nodului urmator
};
typedef struct nod NOD; //Definirea tipului NOD
NOD *prim,*ultim; //Adresa primului si ultimului nod al listei
void creare(); //Prototipul functiei pentru crearea listei dublu
inlantuite
void parcurg_p_u(NOD *); //Prototipul functiei pentru parcurgerea
listei
void parcurg_u_p(NOD *); //Prototipul functiei pentru parcurgerea
in sens invers a listei
int main()
{
    creare(); //Apelul functiei pentru crearea listei
    cout<<"Parcurgera listei de la primul catre ultimul element:\n";
    parcurg_p_u(prim); //Apelul functiei pentru parcurgerea listei
    cout<<"\nParcurgerea listei in sens invers:\n";
    parcurg_u_p(ultim); //Apelul functiei pentru parcurgerea in
sens invers
    return 0; //Functia main() trebuie sa intoarca un intreg
}
void creare() //Definirea functiei pentru crearea listei
{   //Crearea primului nod al listei
    NOD *p,*noul_nod; //p=un nod oarecare; noul_nod este nodul care
se adauga
    p=new NOD; //Rezervarea memoriei pentru un nod
    prim=p; //Aceste va fi primul nod
    ultim=p;//Deocamdata acesta e si ultimul nod
    p->urm=NULL;//Nodul nu are predecesor
    cout<<"Informatia (0 pentru terminare): "; cin>>p->inf;
    p->ant=NULL;//Nodul nu are nici succesor
    while(p->inf) //Adaugarea de noi noduri pana informatia e 0
    {
        noul_nod=new NOD;//Rezervarea memoriei pentru un nou nod ce
se adauga
        noul_nod->ant=p;//Se face lgatura cu nodul anterior
        p->urm=noul_nod;//Se face legatura cu nodul urmator
        cout<<"Informatia (0 pentru terminare): ";
cin>>noul_nod->inf;//Se memoreaza informatia pentru nodul adaugat
        p=noul_nod;//p va indica acum nodul adaugat
```

```
    }
    ultim=p; //Marcam ultimul nod
    ultim->urm=NULL;//Ultimul nod nu are succesor
}
void parcurg_p_u(NOD *p)//Definirea functiei pentru parcurgerea
listei
{
    while (p)//Parcurgerea listei cat timp p are succesor
    {
        cout<<p->inf<<" ";//Se prelucreaza (afiseaza) informatia
din nodul p
        p=p->urm; //Se trece la nodul urmator
    }
}
void parcurg_u_p(NOD *p)//Definirea functiei pentru parcurgerea
listei in sens invers
{
    while (p)//Parcurgerea listi cat timp p are predecesor
    {
        cout<<p->inf<<" ";//Se prelucreaza (afiseaza) informatia
din nodul p
        p=p->ant;//Se trece la nodul anterior
    }
}
```