Parcurgerea in latime a grafurilor memorate prin liste este similara cu diferenta ca vecinii unui nod adaugat in coada se cauta in lista corespunzatoare lui:

```
#include<fstream>
#include<iostream>

using namespace std;

struct nod
{
	int info;
	nod *urm;
};

nod *L[20];

int viz[100];  //marchez cu 1 nodurile vizitate
int m,n;

int prim, ultim, C[100];

void bfi_lis()
{
	int varf, nr;
	nod *p;
	while(prim <= ultim)
	{
		varf=C[prim];
		p=L[varf]; // se parcurge lista elementelor din varful cozii
		while(p)
		{
			nr=p->info;
			if(viz[nr]==0) //numai daca nu a fost vizitat
			{
				ultim++; //maresc coada
				C[ultim]=nr; //il adaug in coada
				viz[nr]=1;
			};  //il marchez ca fiind vizitat
			p=p->urm;
		}
		prim++; //avansez la urmatorul varf  din coada
	}
}


void afisare(int nr_nod)
{
	nod *p=L[nr_nod];
	cout<<"Lista vecinilor lui "<<nr_nod<<" este: ";
```

```
	nod *q=p;
	if(q!=NULL)
		while(q)
		{
			cout<<q->info<<" ";
			q=q->urm;
		}
	cout<<endl;
}

int main()
{
	int i,j;
	nod *p;
	ifstream f("muchii.txt"); //memorare graf in listele de adiacenta
	f>>n>>m;
	while(f>>i>>j)
	{
		p=new nod;
		p->info=j;
		p->urm=NULL;
		if(L[i]!=NULL)
		{
			nod *q=L[i];
			while (q->urm)
				q=q->urm;
			q->urm=p;
		}
		else
			L[i]=p;
	}
	f.close();
	cout<<"\nListele de adiacenta "<<endl;
	for(i=1; i<=n; i++)
		afisare(i);

	int ndr;
	cout<<endl<<"Nodul de inceput ";
	cin>>ndr;
	viz[ndr]=1;
	prim=ultim=1;
	C[prim]=ndr;
	cout<<endl<<"\nParcurgere in latime "<<endl;
	bfi_lis();
	for(i=1;i<=ultim;i++)
		cout<<C[i]<<" ";
	return 0;
}
```

Utilizand functia recursiva:

```
#include<fstream>
#include<iostream>

using namespace std;

struct nod
{
	int info;
	nod *urm;
};

nod *L[20];

int viz[100];  //marchez cu 1 nodurile vizitate
int m,n;

int prim, ultim, C[100];

void bfr_lis()
{
	int varf, nr;
	nod *p;
	if(prim<=ultim)
		{
			varf=C[prim];
			p=L[varf];// se parcurge lista elementelor din varful cozii
			while(p)
			{
				nr=p->info;
				if(viz[nr]==0)//numai daca nu a fost vizitat
				{
					ultim++;//maresc coada
					C[ultim]=nr;//il adaug in coada
					viz[nr]=1; //il marchez ca fiind vizitat
				}
				p=p->urm;
			}
			prim++; //avansez la urmatorul nod din coada
			bfr_lis();
		}
}

void afisare(int nr_nod)
{
	nod *p=L[nr_nod];
	cout<<"Lista vecinilor lui "<<nr_nod<<" este: ";
	nod *q=p;
```

```
	if(q!=NULL)
		while(q)
		{
			cout<<q->info<<" ";
			q=q->urm;
		}
	cout<<endl;
}

int main()
{
	int i,j;
	nod *p;
	ifstream f("muchii.txt"); //memorare graf in listele de adiacenta
	f>>n>>m;
	while(f>>i>>j)
	{
		p=new nod;
		p->info=j;
		p->urm=NULL;
		if(L[i]!=NULL)
		{
			nod *q=L[i];
			while (q->urm)
				q=q->urm;
			q->urm=p;
		}
		else
			L[i]=p;
	}
	f.close();
	cout<<"\nListele de adiacenta "<<endl;
	for(i=1; i<=n; i++)
		afisare(i);

	int ndr;
	cout<<endl<<"Nodul de inceput ";
	cin>>ndr;
	viz[ndr]=1;
	prim=ultim=1;
	C[prim]=ndr;
	cout<<endl<<"\nParcurgere in latime "<<endl;
	bfr_lis();
	for(i=1;i<=ultim;i++)
		cout<<C[i]<<" ";
	return 0;
}
```