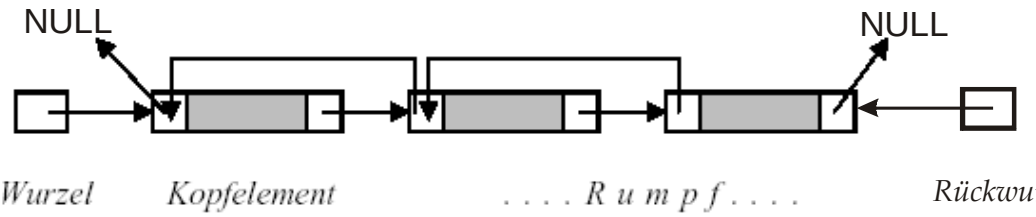
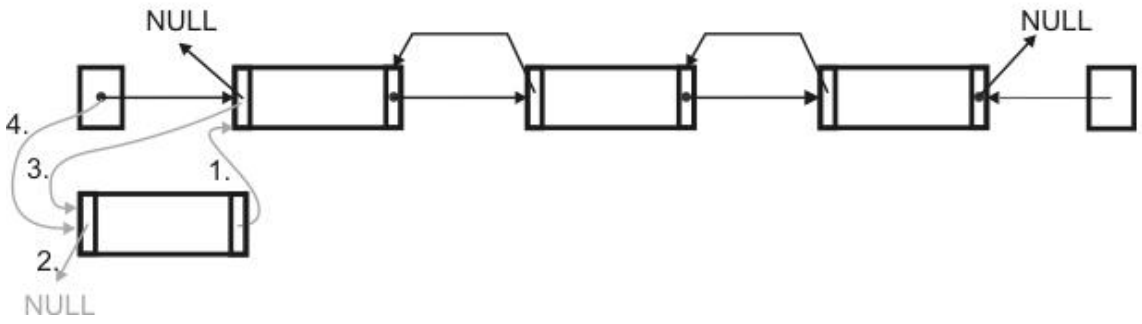
Aufbau einer doppelt verketteten Liste



Struktogramm:

Variable u .Speicher für neu anmelden	
neu->inh=wert	
neu->next=wurzel	
neu->prev=NULL	
wurzel=NULL	
rwurzel=neu	wurzel->prev=neu
wurzel=neu	

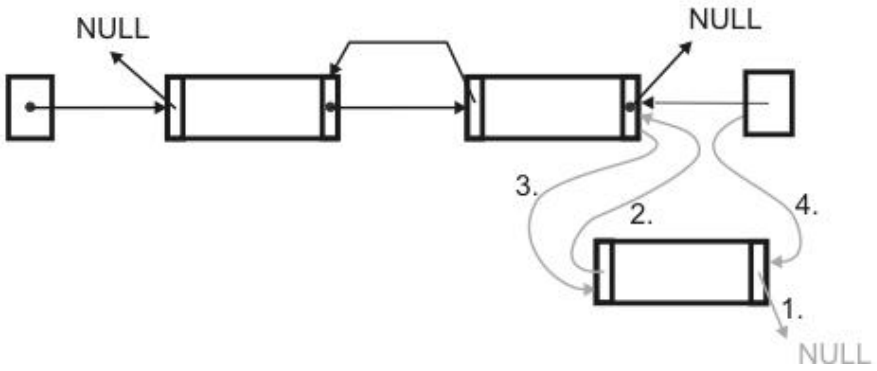
Prinzip (grafisch):



Struktogramm:

Variable u .Speicher für neu anmelden
neu->inh=wert
neu->next=NULL
neu->prev=rwurzel
wurzel=NULL
wurzel=neu / rwurzel->next=neu
rwurzel=neu

Prinzip (grafisch):



```
//Eingabe eines vollständig geklammerten Infix-Ausdrucks
//aeußere Klammern koennen entfallen, UPN wird erzeugt
//nur + und * implementiert

#include "stacky.h"
#include <string.h>
using namespace std;

int main()
{
 init(s);
 char term[20];
 cin>>term;
 for (int posi=0; posi<strlen(term) ;posi++)
 {
   if (term[posi]>='0' && term[posi]<='9')
     cout<<term[posi];
      else
      if (term[posi]=='+' || term[posi]=='*')
        push(s,term[posi]);
        else
        if (term[posi]==')')
        {
         cout<<char (top(s));
         pop(s);
        }
 }
 if (!empty(s)) cout<<char (top(s));
 system("pause");
 return 0;
}
```

Levelorder (als Funktion)

```
void levelorder(Kasten * where)
{
 cout<<"LevelOrder-Traversierung\n";
 init(q);
 enqueue(q,where);
 while (!empty(q))
 {
  Kasten * hilf=head(q);
  cout<<hilf->inh<<"_ ";
  dequeue(q);
  if (hilf->lnf!=NULL) enqueue(q,hilf->lnf);
  if (hilf->rnf!=NULL) enqueue(q,hilf->rnf);
 }
}
```