**Двойно свързан списък**

В компютърните науки, двойно свързаният списък е свързана структура от данни, състояща се от множество последователно свързани елементи наречени върхове (nodes на английски). Всеки един връх съдържа две полета наричани връзки, които са референции (указатели) към предишния и следващия връх в поредицата от върхове. Връзките на началните и крайните елементи в двойно свързания списък имат по един специален вид разклонение, служещо за прекратяване обхождането на списъка. Този специален вид разклонение обичайно е празен/кух връх (sentinel node) или “null”. Ако списъкът има само един празен връх, то той е кръгообразно свързан чрез него. Двойно свързаният списък може да бъде представен и като два отделни   
единично свързани списъка, съставящи се от едни и същи елементи, но в противоположен ред.

//коментар под снимката//Двойно свързан списък, чиито върхове съдържат три полета: елемент със стойност от целочислен тип (integer), връзката към следващия елемент (node) и такава към предишното.

Това, което позволява обхождането на списъка в двете посоки, са двете връзки на всеки елемент, към върховете (референциите към съседните елементи). Въпреки че добавянето и премахването на елемент от двойно свързания списък изисква промяната на повече връзки, отколкото същата операция в единично свързания, операциите са по - опростени и потенциално по - ефикасен (за елементите различни от крайните), защото по време на обхождане няма нужда да се взима под внимание връзката към предишното разклонение и няма нужда да обхождаме списъка, за да намерим връзката, която искаме да променим.

Тази концепция е и в основата на техниката за запаметяване в мнемоничната (статия уики) свързаща система (наричана още “свързан метод“–) /\*ПРЕПРАТКИ/\*

**Номенклатура и имплементация**  
Първият и последният елемент на двойно свързания списък, наричани съответно “head” (глава) и “tail” (опашка), могат да бъдат достигнати незабавно (т.е. без обхождане на списъка). Те позволяват обхождането на списъка от началото или края. Например обхождане на листа от началото към края, или от края към началото, за намиране на елемент с конкретна стойност. След като намерим конкретен елемент с дадена стойност, той може да се използва като начало за ново обхождане на списъка в двете посоки - към началото на списъка или към неговия край.

Полетата представляващи връзките към съседните елементи в двойно свързаният списък често се наричат “next” и “previous” или “forward” и “backward” съответно предходен и следващ. Указателите, които държат съответните полета най- често са представени като “pointer” (референция), но могат също така да бъдат и адресни отклонения или индекси в масив, в който съществуват елементите, към които се сочи.

В [[компютърните науки]], '''двойно свързаният списък ''' е [[en:Linked\_data\_structure|свързана структура от данни]], състояща се от множество последователно свързани [[en: nodes(computer science)| елементи]]. Всеки един елемент съдържа две [[en:field (computer science)|полета]] наречени ''връзки'', които са [[en: reference (computer science)|референции (указатели)]] към предишния и следващия връх в поредицата от елементи. Връзките на началните и крайните елементи в двойно свързания списък имат по един специален вид разклонение, служещо за прекратяване обхождането на списъка. Този специален вид разклонение обичайно е празен елемент ([[en:sentinel node|sentinel node]]) или [[en:null pointer|null]]. Ако списъкът има само един празен връх, то той е кръгообразно свързан чрез него. Двойно свързаният списък може да бъде представен и като два отделни [[en: linked list|единично свързани списъка]], съставящи се от едни и същи елементи, но в противоположен ред.

В [[компютърните науки]], '''двойно свързаният списък ''' е [[en:Linked\_data\_structure|свързана структура от данни]], състояща се от множество последователно свързани [[en: nodes(computer science)| елементи]]. Всеки един елемент съдържа две [[en:field (computer science)|полета]] наречени ''връзки'', които са [[en: reference (computer science)|референции (указатели)]] към предишния и следващия елемент в поредицата от елементи. Връзките на '''началните''' и '''крайните''' елементи в двойно свързания списък имат по един специален вид разклонение, служещо за прекратяване обхождането на списъка. Този специален вид разклонение обичайно е празен елемент ([[en:sentinel node|sentinel node]]) или [[en:null pointer|null]]. Ако списъкът има само един празен елемент, то той е кръгообразно свързан чрез него. Двойно свързаният списък може да бъде представен и като два отделни [[en: linked list|единично свързани списъка]], съставящи се от едни и същи елементи, но в противоположен ред.

[[File:doubly-linked-list.svg|frame|none|alt=A doubly linked list whose nodes contain three fields: an integer value, the link to the next node, and the link to the previous node.|Двойно свързан списък, чиито върхове съдържат три полета: елемент със стойност от целочислен тип (integer), връзката към следващия елемент (node) и такава към предишното.]]<!--THIS IMAGE SHOULD BE MODIFIED TO USE THE STANDARD NOTATION FOR NULL POINTERS. (Actually, the concept of "null pointer" should be left out and be replaced by "terminator" because sentinel links are quite commonly used instead of null pointers -tony).-->

Това, което позволява обхождането на списъка в двете посоки, са двете връзки на всеки елемент. Въпреки че добавянето и премахването на елемент от двойно свързания списък изисква промяната на повече връзки, отколкото същата операция в единично свързания, операциите са по - опростени и потенциално по - ефикасни (за елементите различни от крайните), защото по време на обхождане няма нужда да се взима под внимание връзката към предишното разклонение и няма нужда да обхождаме списъка, за да намерим връзката, която искаме да променим.

Тази концепция е и в основата на техниката за [[en:memorization|запаметяване]] в [[en:mnemonic link system|мнемоничната свързваща система]] (наричана още “свързващ метод“).

**Basic algorithms  
Open doubly linked lists**  
/\*code example/\*  
/\*code example/\*

**Обхождане на списъка.**

Обхождането на двойно свързаният списък може да се направи и в двете посоки. Дори посоката на обхождане може да бъде променяна многократно. Обхождането често е наричано итериране, но използването на тази терминология е неподходящо, тъй като итерацията има добре дефинирана семантика (например в математиката), която не е аналог на обхождането.

Next node (следващи)  
/\*code example/\*

Previous node (следващи)  
/\*code example/\*

**Вмъкване на елемент**  
Следните симетрични функции вмъкват елемент преди или след конкретен елемент  
/\*code example/\*

За вмъкването на елемент в началото на евентуално празен списък ни трябва функция:  
/\*code example/\*

Вмъкването на елемент в края на списъка става чрез симетричната функцията:

**Премахване на елемент**Премахването на елемент става по- лесно отколкото вмъкването, но изисква по- специално внимание, когато елементът, който искаме да премахнем, е първият или последният от списъка.  
/\*code example/\*

**Вальо:  
Circular doubly linked list  
......**

Advanced concepts – дотук е стигнал Вальо