ÇİFT Bağlı Listeler

Çift bağlı listeler tıpkı tek bağlı listeler gibi her düğümün 2 pointeri vardır. Pointer in biri sonraki düğümü, diğer pointer bir önceki düğümü gösterir.Bunun bize kazandırdıkları;

- Listede ileri ve geri yönde hareket edilebilir.
- Kolayca listenin her yerine eleman eklenebilir.
- Düğümleri kolaylıkla silebilirsin

API çift bağlı listedeki düğümleri tanımlar.

```
typedef struct dllist {
  struct dllist *flink;
  struct dllist *blink;
  Jval val;
} *Dllist;
```

dllist.o tarafından desteklenen işlemler;

- **Dllist new_dllist()**: Yeni bir çift bağlı liste için yer ayırır ve değer döndürür.
- **free_dllist(Dllist l)**: Listede ayrılan tüm bellek alanlarını arar ve yok eder.Listesi boş olmak zorunda değildir.
- **dll_prepend(Dllist l, Jval val)**: Listenin başına yeni bir düğüm ekler.Bu düğümün değeri val dir.dll_prepend() geri dönüş değeri yoktur.
- **dll_append(Dllist l, Jval val)**: Listenin sonuna yeni bir düğüm ekler.Bu düğümün değeri val dir.Geri dönüş değeri yoktur.
- **dll_insert_b(Dllist n, Jval val)**: Belirtilen düğümün soluna ekleme yapar.Değeri val dir.
- **dll_insert_a(Dllist n, Jval val)**: Belirtilen düğümün sağına ekleme yapar.Değeri val dir.
- **Dllist dll_nil(Dllist l)**: Listesi için bir başlangıç düğümü pointerla döner.dll.first(l) çağırmaya gerek yok.l ->flink i kullanabilirsin.
- **Dllist dll_first(Dllist l)**: Listedeki ilk düğüm bir pointerla döner.Eğer liste boşsa başlangıç düğümüne döner.dll_nil() çağrısına gerek yok, l->flink şeklinde kullanılabilir.
- **Dllist dll_last(Dllist l)**: Listedeki son düğüm pointerla döner.Eğer liste boşşsa başlangıç düğümüne döner.l->blink şeklinde kullanılabilir.
- **Dllist dll_next(Dllist n)**: Listedeki n'den sonraki düğüm pointerla döner.n listedeki son düğüm ise dll_next() başlangıç düğümüne döner.n->flink şeklinde kullanılabilir.
- **Dllist dll_prev(Dllist n)**: Listedeki n'den önceki düğüm pointerla döner.Eğer listedeki ilk eleman n ise dll_prev() başlangıç düğümüne döner.n->blink şeklinde kullanılabilir.
- **int dll_empty(Dllist l)**: l var mı yok mu kontrol eder, yoksa boş döndürür.
- **Jval dll_val(Dllist n):** n düğümünün val alanını döndürür.Genellikle kullanılmaz ama bazen kullanışlı bir yöntemdir.
- **int dll_delete_node(Dllist n)**: Listeden n düğümünü siler ve bellek alanını serbest bırakır.

  Son olarak ileri ve geri hareket etmek için 2 makromuz var.Dllist olan bir ptr, Dllist olan bir list.

```
#define dll_traverse(ptr, list) \
  for (ptr = (list)->flink; ptr != (list); ptr = ptr->flink)
#define dll_rtraverse(ptr, list) \
  for (ptr = (list)->blink; ptr != (list); ptr = ptr->blink)
```

İMPLEMENTASYON

Her dllist uygulaması dairesel çift bağlı listedir.Dllist.c içerisindeki kod

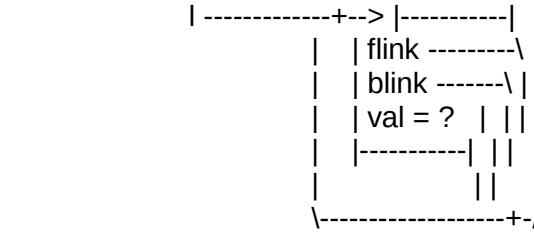
dllist düğümü için bir typedef;

```
typedef struct dllist {
  struct dllist *flink;
  struct dllist *blink;
  Jval val;
} *Dllist;
```

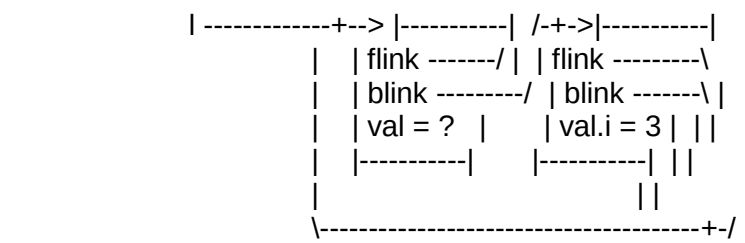
Her düğümün 2 pointeri var.flink listedeki bir sonraki düğümü, blink bir önceki düğümü tutmak içindir.Bir Dllist başlangıç veya bitiş düğümünde bir pointerdır.

Listede her 2 yönde hareket edilebilir.flink listenin ilk düğümünü, blink listenin son düğümünü tutar.Başlangıç düğümünde ilk düğümün blink noktası son düğümün flink noktası ile aynı yerdir.

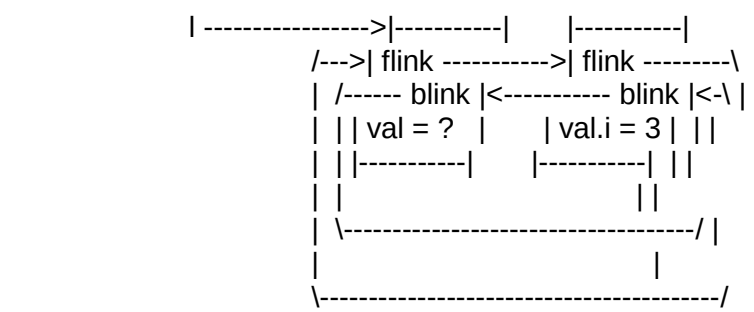
l listesinin boş hali:



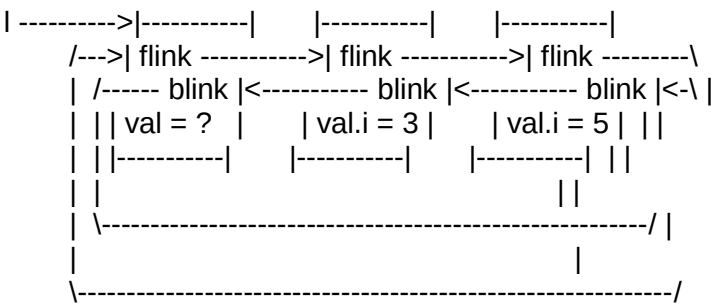
dll_append(l, new_jval_i(3)); çağrıldıktan sonraki hali(veya **dll_prepend(l, new_jval_i(3)** da çağırsak aynısı):



Listenin yeni hali:



dll_append(l, new_jval_i(5)) çağrıldıktan sonraki hali:



Örneği daha fazla ilerletmek istemiyorum.Şimdilik kullanış biçimi anlaşılmış olması gerekir.

Prosedür implementasyonların çoğu gereksiz prosedür ya da makrolardır.

```
Dllist new_dllist()
{
  Dllist d;

  d = (Dllist) malloc (sizeof(struct dllist));
  d->flink = d;
  d->blink = d;
  return d;
}
```

```
dll_empty(Dllist l)
{
  return (l->flink == l);
}
```

```
free_dllist(Dllist l)
{
  while (!dll_empty(l)) {
    dll_delete_node(dll_first(l));
  }
  free(l);
}
```

```
#define dll_first(d) ((d)->flink)
#define dll_next(d) ((d)->flink)
#define dll_last(d) ((d)->blink)
#define dll_prev(d) ((d)->blink)
#define dll_nil(d) (d)
```

Kod parçasının önemli kısmı **dll_insert_b()** ve **dll_delete_node** dur. **dll_insert_b(n, v)** ile biz **malloc()** la yeni bir düğüm alırız.değer olarak v atanır ve n düğümünün soluna bağlanır.n için yeni düğümün flink alanını oluştururuz ve blink kullanımı n->blink şeklindedir.Sonra yeni bir düğüm için n->blink i oluştururuz ve eski n->blink in flink alanı düğüm içindir.Aşağıda kodları mevcuttur:

```
dll_insert_b(Dllist node, Jval v)       /* Inserts before a given node */
{
  Dllist new;

  new = (Dllist) malloc (sizeof(struct dllist));
  new->val = v;

  new->flink = node;
  new->blink = node->blink;
  new->flink->blink = new;
  new->blink->flink = new;
}
```

Dll_insert_b() ile yapabileceğimiz basit fonksiyon çağrıları:

```
dll_insert_a(Dllist n, Jval val)        /* Inserts after a given node */
{
  dll_insert_b(n->flink, val);
}

dll_append(Dllist l, Jval val)     /* Inserts at the end of the list */
{
  dll_insert_b(l, val);
}

dll_prepend(Dllist l, Jval val)    /* Inserts at the beginning of the list */
{
  dll_insert_b(l->flink, val);
}
```

Silmek oldukça kolay. Listeden n->blink için **n->flink->blink i, n->flink** için **n->blink->flink i** kurarak n'den düğümü kaldırmalıyız.Sonra n i serbest bırakırsın.

```
dll_delete_node(Dllist node)           /* Deletes an arbitrary iterm */
{
  node->flink->blink = node->blink;
  node->blink->flink = node->flink;
  free(node);
}
```

Kullanışlı örnekler

Bizim standartlarımızdan biri ilk örnek:Standart giriş tersleme.Açıklama gerektirmeyen basit bir örnek.dllreverse.c içinde mevcut.

```
#include < stdio.h >
#include < string.h >
#include "fields.h"
#include "dllist.h"

main()
{
  IS is;
  Dllist l;
  Dllist tmp;

  is = new_inputstruct(NULL);
  l = new_dllist();

  while (get_line(is) >= 0) {
    dll_append(l, new_jval_s(strdup(is->text1)));
  }

  dll_rtraverse(tmp, l) printf("%s", jval_s(tmp->val));
}
```

Diğer örnek ise bir başka standarttır:Dlist içinde standart giriş okurken bunu yaparız ve emin olun Dllist in en çok n tane düğümü vardır.Sonra yazdırırız ve dlltail.c ile derleriz:

```
#include < stdio.h >
#include < string.h >
#include "fields.h"
#include "dllist.h"

main(int argc, char **argv)
{
  IS is;
  int n;
  Dllist l;
  Dllist tmp;

  if (argc != 2) {
    fprintf(stderr, "usage: dlltail n\n");
    exit(1);
  }
  n = atoi(argv[1]);
  if (n < 0) {
    fprintf(stderr, "usage: dlltail n  -- n must be >= 0\n");
    exit(1);
  }

  is = new_inputstruct(NULL);
  l = new_dllist();

  while (get_line(is) >= 0) {
    dll_append(l, new_jval_s(strdup(is->text1)));
```

```
    if (is->line > n) {
      tmp = dll_first(l);
      free(jval_s(dll_val(tmp)));
      dll_delete_node(tmp);
    }
  }

  dll_traverse(tmp, l) printf("%s", jval_s(tmp->val));
}
```