

Doubly Linked List Örneği

Bu projede doubly linked listde add, delete, createlist, display list, search, _delete,_search,_insert fonksiyonları oluşturulmuştur. Her düğüm kendisinden önceki düğümün adresini ve sonraki düğümün adresini tutmaktadır. Ek olarak bir de öğrenci bilgilerini içeren ogrb struct pointerini tutmaktadır.

Öğrenci bilgilerini tutmak ogrb adında bir struct tanımlanmıştır ve node struct ının içerisine eklenmiştir.

```
struct ogrbilgi {  
    int numara;  
    char ad[20];  
    char soyad[20];  
    char bolum[30];  
    int sinif;  
};  
  
typedef struct ogrbilgi ogrb;  
  
struct node {  
    struct node* onceki;  
    struct node* sonraki;  
    ogrb* ogrbilgi;  
};  
  
typedef struct node NODE;  
  
typedef struct {  
    int count;  
    NODE* head;
```

```
    NODE* rear;
    NODE* konum;
    int (*compare)(void* veri1, void* veri2);
} LIST;
```

Fonksiyonların tanımı şöyledir:

```
LIST* CreateLinkList(int (*compare)(void* veri1, void* veri2));
LIST* DestroyLinkedList(LIST* liste);
int addNode(LIST* liste, void* ogr);
bool deleteNode(LIST* liste, void* ogrkey, void** out);
bool searchNode(LIST* liste, void* veri, void** out);
bool displayLinkList(LIST* liste, int nereden, void** out);

static bool _insert(LIST* liste, NODE* onceki, void* ogr);
static void _delete(LIST* liste, NODE* konum, void** out);
static bool _search(LIST* liste, NODE** onceki, NODE** konum, void* veri);
```

Program kullanıcıya hangi işlevi yapmakla istediğini seçtirmekle başlıyor.

Seçilen işleme göre ilgili fonksiyonlar çalışıyor ve bu işlem kullanıcı programdan çıkmak isteyene kadar devam ediyor. Bununla ilgili main kodu:

```
int main()
{
    int secim;
    void* key;
    void** out;
    LIST* liste = NULL;
```

```
while(1){
    printf("1-Create List\n2-Add Node\n3-Delete Node\n4-Search Node\n5-
Display List\n\n");
    scanf("%d", &secim);
    switch (secim) {
        case 0: printf("Programdan cikiliyor");
                exit(0);
        case 1:
                liste = CreateLinkList(compare);
                printf("Liste olusturuldu\n");
                break;
        case 2: {
                char gecici[20];
                ogrb* bilgi = malloc(sizeof(ogrb)); // Yeni bir öğrenci bilgisi için
bellek ayırın

                printf("AD:");
                scanf("%s", gecici);
                strcpy(bilgi->ad, gecici);

                printf("SOYAD:");
                scanf("%s", gecici);
                strcpy(bilgi->soyad, gecici);

                printf("BOLUM:");
                scanf("%s", gecici);
                strcpy(bilgi->bolum, gecici);

                printf("SINIF:");
                int gecici1;
```

```

scanf(" %d",&gecici1);
bilgi->sinif = (rand() % 4) + 1;

printf("NUMARA: 213608590__");
bilgi->numara = scanf(" %d",&gecici1);
addNode(liste, bilgi);
printf("Node eklendi\n");
break;
}

case 3:
    printf("Silme istediginiz ogrenci numarasini girin: ");
    int* silinecekNumara;
    scanf("%d", silinecekNumara);
    if (deleteNode(liste, (void*)silinecekNumara, &out)) {
        printf("%d numaralı öğrenci silindi.\n", silinecekNumara);
    } else {
        printf("%d numaralı öğrenci bulunamadı.\n", silinecekNumara);
    }
    break;

case 4:
    printf("Aranacak ogrenci numarasini girin: ");
    int aranacakNumara;
    scanf("%d", &aranacakNumara);
    if (searchNode(liste, &aranacakNumara, &out)) {
        printf("%d numaralı öğrenci bulundu.\n", aranacakNumara);
    } else {
        printf("%d numaralı öğrenci bulunamadı.\n", aranacakNumara);
    }
    searchNode(liste, 21360859001, out);

```

```

        break;
    case 5:
        printf("Liste:\n");
        while (displayLinkList(liste, 0, &out)) {
            ogrb* bilgi = (ogrb*)out;
            printf("Numara: %d, Ad: %s, Soyad: %s, Bolum: %s, Sinif: %d\n",
                bilgi->numara, bilgi->ad, bilgi->soyad, bilgi->bolum, bilgi->sinif);
        }
        displayLinkList(liste, 5, out);
        break;
    default:
        printf("Geçersiz seçim.\n");
        break;
    }
}

// Liste üzerinde işlem yapıldıktan sonra gerekirse belleği serbest bırakın
//liste = DestroyLinkedList(liste);
}

```

Eğer programda kullanıcı CreateLinkedList i seçerse oluşan çıktı:

```

C:\Users\zehra\Desktop\Veri_ x + v
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
257
258 int main
259 {
260
261     int 1-Create List
262     void 2-Add Node
263     void 3-Delete Node
264     LIST 4-Search Node
265     prin 5-Display List
266     scan
267     1
268     switch
269         Liste olusturuldu
270
271     Process returned 0 (0x0) execution time : 1.511 s
272     Press any key to continue.
273
274
275
276
277
278

```

Add Node fonksiyonu seçilirse kullanılan kodlar :

```

int addNode(LIST* liste, void* ogr) {
    bool bulundu;

```

```
bool bul;
```

```
NODE* onceki;
```

```
NODE* konum;
```

```
bul = _search(liste, &onceki, &konum, ogr);
```

```
if (bulundu) {
```

```
    return 1;
```

```
}
```

```
bulundu = _insert(liste, onceki, ogr);
```

```
if (!bulundu) {
```

```
    return (-1);
```

```
}
```

```
return (0);
```

```
}
```

```
static bool _insert(LIST* liste, NODE* onceki, void* ogr) {
```

```
    NODE* newnode;
```

```
    newnode = (NODE*)malloc(sizeof(NODE));
```

```
    if (!newnode) {
```

```
        return false;
```

```
}
```

```
newnode->ogrbilgi = ogr;
```

```
newnode->sonraki = NULL;
```

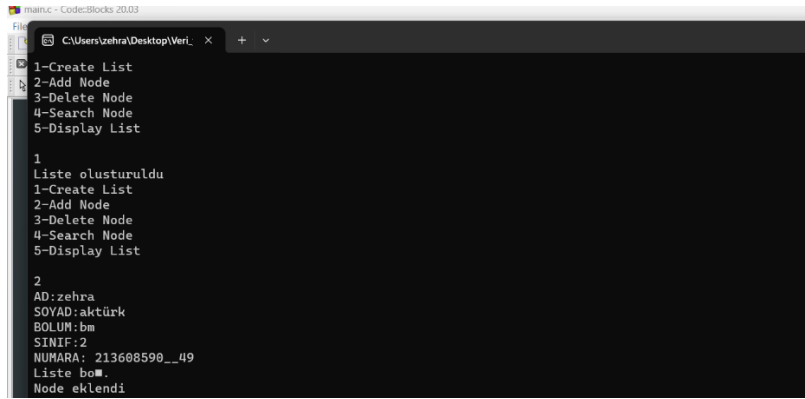
```
newnode->onceki = NULL;
```

```
if (onceki == NULL) {
```

```
    newnode->sonraki = liste->head;
```

```
if (liste->head) {
    liste->head->onceki = newnode;
}
liste->head = newnode;
if (liste->count == 0) {
    liste->rear = newnode;
}
} else {
    newnode->sonraki = onceki->sonraki;
    newnode->onceki = onceki;
    if (newnode->sonraki != NULL) {
        newnode->sonraki->onceki = newnode;
    } else {
        liste->rear = newnode;
    }
    onceki->sonraki = newnode;
}
(liste->count)++;
return true;
}
```

Add Node fonksiyonu çalıştırıldığında oluşan çıktı:

A screenshot of a CodeBlocks IDE window. The title bar shows 'main.c - CodeBlocks 20.03'. The menu bar includes 'File', 'Edit', 'View', 'Tools', 'Settings', 'Help'. The 'Tools' menu is open, showing options: '1-Create List', '2-Add Node', '3-Delete Node', '4-Search Node', and '5-Display List'. The main editor area shows the following output:

```
1
Liste olusturuldu
1-Create List
2-Add Node
3-Delete Node
4-Search Node
5-Display List

2
AD:zehra
SOYAD:aktürk
BOLUM:bm
SINIF:2
NUMARA: 213608590__49
Liste boş.
Node eklendi
```

Ve diğer fonksiyonlara ait kodlar:

```
bool deleteNode(LIST* liste, void* ogrkey, void** out) {
```

```
    bool bul;
```

```
    NODE* onceki;
```

```
    NODE* konum;
```

```
    bul = _search(liste, &onceki, &konum, ogrkey);
```

```
    if (bul) {
```

```
        printf("Aranan öğrenci numarası bulundu.\n");
```

```
        _delete(liste, konum, out);
```

```
        printf("Öğrenci başarıyla silindi.\n");
```

```
    } else {
```

```
        printf("Aranan öğrenci numarası bulunamadı.\n");
```

```
    }
```

```
    return bul;
```

```
}
```

```
static bool _search(LIST* liste, NODE** onceki, NODE** konum, void*  
veri) {
```

```
    int result;
```


***onceki = NULL;**

***konum = liste->head;**

```
if (liste->count == 0) {  
    printf("Liste boş.\n");  
    return false;  
}
```

```
if ((*liste->compare)(veri, liste->head->ogrbilgi->numara) < 0) {  
    *konum = NULL;  
    printf("Aranan öğrenci numarası listede bulunamadı.\n");  
    return false;  
}
```

```
if ((*liste->compare)(veri, liste->rear->ogrbilgi->numara) >= 0) {  
    *onceki = liste->rear;  
    *konum = NULL;  
    printf("Aranan öğrenci numarası listede bulunamadı.\n");  
    return false;  
}
```

```
while ((result = (*liste->compare)(veri, (*konum)->ogrbilgi->numara))  
>= 0) {  
    *onceki = *konum;  
    *konum = (*konum)->sonraki;  
}
```

```
if (result == 0) {
```

```
    printf("Aranan öğrenci numarası bulundu.\n");
    return true;
} else {
    printf("Aranan öğrenci numarası listede bulunamadı.\n");
    return false;
}
}
```

```
void _delete(LIST* liste, NODE* konum, void** out) {
    *out = konum->ogrbilgi;

    if (konum->onceki == NULL) {
        liste->head = konum->sonraki;
        if (liste->head) {
            liste->head->onceki = NULL;
        }
    } else {
        konum->onceki->sonraki = konum->sonraki;
    }

    if (konum->sonraki == NULL) {
        liste->rear = konum->onceki;
    } else {
        konum->sonraki->onceki = konum->onceki;
    }

    (liste->count)--;
    free(konum);
}
```

```
    return;  
}
```

```
bool searchNode(LIST* liste, void* veri, void** out) {  
    bool bul;  
  
    NODE* onceki;  
    NODE* konum;  
  
    bul = _search(liste, &onceki, &konum, veri);  
    if (bul) {  
        *out = konum->ogrbilgi;  
    } else {  
        *out = NULL;  
    }  
  
    return bul;  
}
```

```
bool displayLinkList(LIST* liste, int nereden, void** out) {  
    if (liste->count == 0) {  
        return false;  
    }  
}
```

```

if (nereden == 0) {
    liste->konum = liste->head;
    *out = liste->konum->ogrbilgi;
    return true;
} else {
    if (liste->konum->sonraki == NULL) {
        return false;
    } else {
        liste->konum = liste->konum->sonraki;
        *out = liste->konum->ogrbilgi;
        return true;
    }
}
}

```

```

LIST* DestroyLinkedList(LIST* liste) {
    NODE* deletePtr;

    if (liste) {
        while (liste->count > 0) {
            free(liste->head->ogrbilgi);
            deletePtr = liste->head;
            liste->head = liste->head->sonraki;

            if (liste->head) {
                liste->head->onceki = NULL; // Yeni eklenen satır
            }
            (liste->count)--;
        }
    }
}

```

```
    free(deletePtr->ogrbilgi); // Önce öğrenci bilgilerini serbest bırak
    free(deletePtr); // Sonra düğümü serbest bırak
}
}
free(liste); // Listenin kendisini serbest bırak
return NULL;

}
```

Zehra Aktürk

21360859049