C'de Yapılar

struct: Birbirleri ile ilgili birçok veriyi tek bir isim altında toplamak için bir yol.

Örnek: Bir z karmaşık sayısını ele alalım

z = x + iy
real imaginary

```
struct complex{
    int real;
    int im;
};
struct complex a,b;
a.real= 4;
a.im =7;
struct complex obj;
struct complex *p=&obj;
p->real=2;
p->im = 3;
```

. operatörü
-> "

iki karmaşık sayısı toplayan bir fonksiyon

```
struct complex add(struct complex a,
                   struct complex b){
    struct complex result;
    result.real = a.real+b.real;
    result.im = a.im +b.im;
    return result;
}
```

Alternatif struct tanımı

```
typedef struct{
    ....
    ....
    ....
}rec;
rec x,y;
```

http://web.karabuk.edu.tr/hakankutucu/BLM227notes.htm

Veri Yapılarına Giriş

Tanım: Veri yapısı bilgisayarda veriyi saklamak ve düzenlemek için bir yol.

1.) Mantıksal Modelleri
2.) Gerçekleştirilmesi (Implementation)

1.) Soyut veri tipleri (ADT, Abstract Data Type)

Örneğin listeler

- Herhangi tipte birçok elemanı saklar.
- Elemanlar okunur
- Elemanlar güncellenir
- Ekleme / Silme

2. Gerçekleştirilmesi

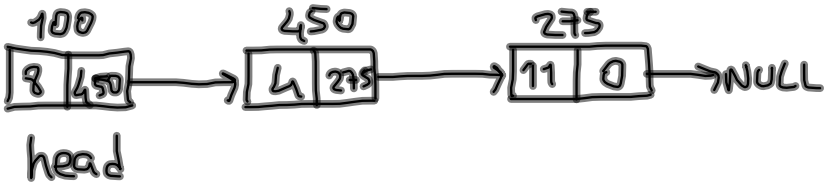
Diziler

Dizilerin kısıtları:

1) Derleme aşamasında dizinin boyutu bilinmeli
2) Ekleme işlemi: dizinin diğer elemanlarını kaydırma gerektirir.

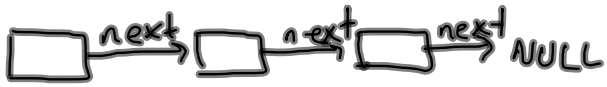
Bağlı Listeler (Linked-Lists)

```
struct node{
    int data;
    struct node *next;
};
```



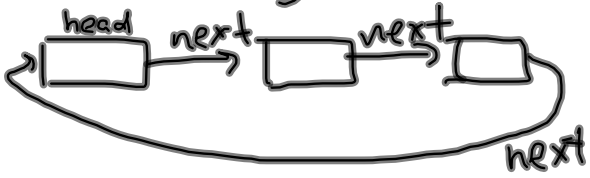
Bağlı Liste Türleri

1. Tek bağlı listeler (next)



2. Çift bağlı listeler (next+previous)

3. Dairesel bağlı listeler (next)

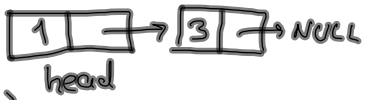


Bağlı Listeler ile İşlemler

1. Liste oluşturma
2. Listeye eleman ekleme
3. Listeden " silme
4. Arama
5. Listenin elemanları yazma
6. " " sayma

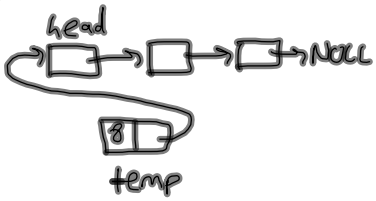
Liste oluşturma

```
main (){
    struct node *head;
    head=(struct node *)malloc(sizeof(struct node));
    head->data=1;
    head->next=NULL;
    head->next=new node();
    head->next->data=3;
    head->next->next=NULL;
}
```



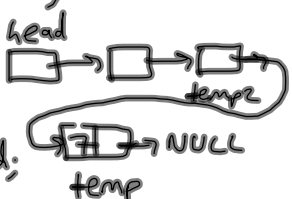
Listenin Başına Ekleme

```
struct node*addhead(struct node *head,int key){
    struct node *temp=new node();
    temp->data=key;
    temp->next=head;
    head=temp;
    return head;
}
```



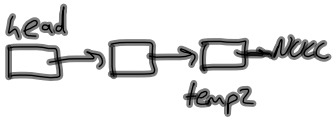
Listenin Sonuna Ekleme

```
struct node*addlast(struct node *head){
    struct node *temp=new node();
    temp->data=7;
    temp->next=NULL;
    struct node *temp2=head;
    while(temp2->next!=NULL)
        temp2=temp2->next;
    temp2->next=temp;
    return head;
}
```



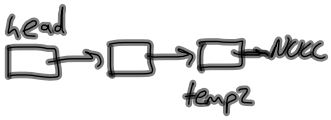
Listenin Elemanlarını Yazma

```
void print(struct node *head){
  if(head==NULL){
     printf("Liste bos");
     return;
  }struct node *temp2=head;
  while(temp2 != NULL){
       printf("%d",temp2->data);
       temp2=temp2->next;
  }
}
```



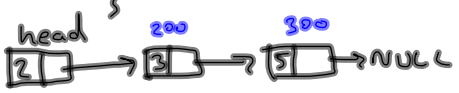
Listenin Elemanlarını Sayma

```
int count(struct node *head){
  if(head==NULL){
     printf("Liste bos");
     return;
  }struct node *temp2=head;
  int counter=0;
  while(temp2 != NULL){
       counter++;
       temp2=temp2->next;
  }
  return counter;
}
```



Soru: Listenin elemanlarını yazan özyinelemeli bir fonksiyon yazınız.

```
void print_rec(struct node *head){
    if(head==NULL)
        return;
    else{
        printf("%d",head->data);
        print_rec(head->next);
    }
}
```



print_rec(head) → print_rec(200) → print_rec(300) → print_rec(NULL)

2 3 5