Veri Yapıları 2

1 Eklenen eleman sona eklenir.
Çıkan eleman önden çıkar

Ön ve arka elemanlar

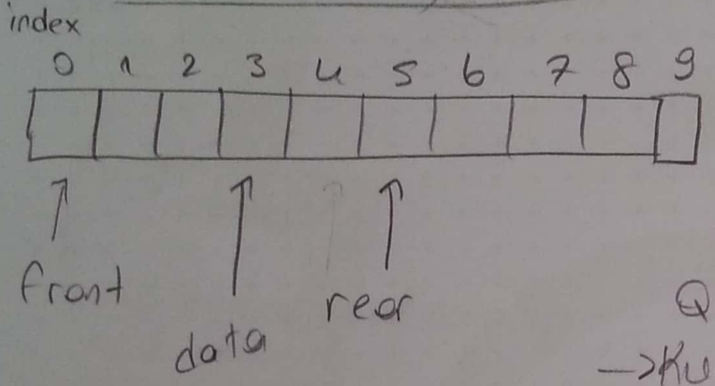
QUEUE → KUYRUK
DE QUEUE

Kuyruğun sonunu gösteren } rear.

KUYRUKLARIN MPLEMENTASYONU
DZ
(ARRAY)

rear → çalışması için NULL gerekir.

peek → sondaki düğümü bulmak için



cnt = 6 ← eleman sayısı

QUEUE_SIZE = 10 Dizi boyutu

→ Kuyruğa ekleme işlemine enqueue
→ Çıkarma işlemine dequeue

DAİRESEL KUYRUK

Yığına veri ekleme işlemine ekleme, arama, silme, sıralama,
Kuyruk yapısı: ilk giren ilk çıkar first in first out
Yığında erişim top elemanına bağlı onu görebiliyoruz
Kuyrukta önemli elemanlardan biri reardır.

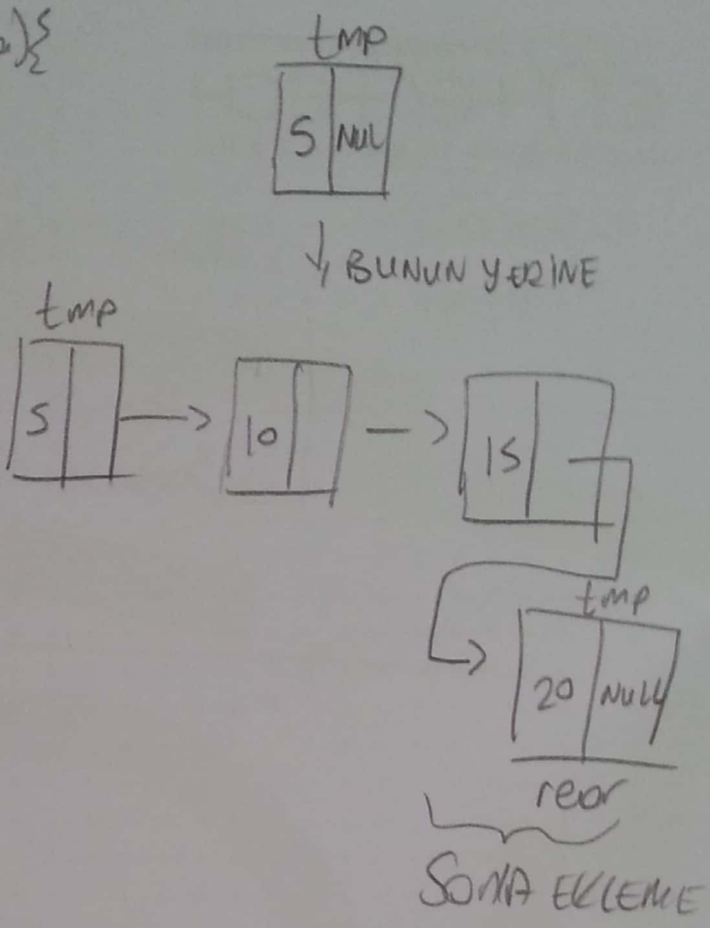
```
#include <iostream>
#include <cstdlib>
# include <ctime>

using namespace std;
 struct node{
   int info;
   struct node *next;
 }*front, *rear;

 class queue{
   public:
   Queue(){
       rear=NULL;
       front=NULL;
  }
  void enqueue(int data);
   void dequeue(        );
  void display(        );
  int search(node*,int aro);
  void peek(        );
 };

 int main(){
 Queue queue;
 int data,aranan,pos=-1, N;
 int choice,i;
 while(true){
 cout <<"\n..."<<endl;
 cout<<"operation on queue "<<endl;
 cout<<"\n ..."<<endl;
cout<<"1.Enqueue(Add)Element into the queue<<endl;
cout<<"2.Dequeue(Remove) Element from the queue<<endl;
cout<<"3.Display the Queue"<<endl;
cout<<"4.Peek the Queue"<<endl;

cout<<"5 Quit"<<endl;
cout<<"Enter your choice:"<<endl;
cin>>choice;
cout<<"~~**************~~"<<endl;
```

①

```
switch(choice){
   case 1:
     cout<<"Eklenecek deger:"<<endl;
     cin>>data;
     queue.enqueue(data);
     break;
   case 2:
     queue.dequeue();
     break;
   case 3:
     queue.display();
     break;
   case 4:
     queue.peek();
     break;
   case 5:
     exit(0);
     break;
   default:
     cout<<"Hatali giris"<<endl;
     break;
  }
 }
 return 0;
}
```

DEVAMI
VAR
(ARKADA)

```
void Queue::enqueue(int data){
    node* tmp;
    tmp=new(struct node);
    tmp->info=data;
    tmp->next=NULL;
    if(front==NULL){
        front=tmp;
    }else{}
        rear->next=tmp;
    }
    rear=tmp;
}
Void Queue::dequeue(){
    Node * tmp;
    tmp=new (struct node);
    if(front==NULL){
        cout<<"Kuyruk Bos"<<endl;
    }else{
        cout<<"Cikacak eleman:"<<front->info<<endl;
        tmp=front;
        front=tmp->next;
        free(tmp);
    }
}
void Queue::display()
    node *p=new node;
    p=front;
    if(front==NULL){
        cout<<"\nNothing to Display\n";
    }else{
        cout<<"Queue Elements:"<<endl;
        while(p!=NULL){
            cout<<endl<<dec<<p->info;
            p=p->next;
        }
    }
}
```



Devamı var arkada

```
void Queue::peek(){
    if(front == NULL)
          cout<<"Queue is empty"<<endl;
    else{
         cout<<"Peek elements: "<<endl;
         cout<<dec<<front->info<<endl;
    }
}
```

peek

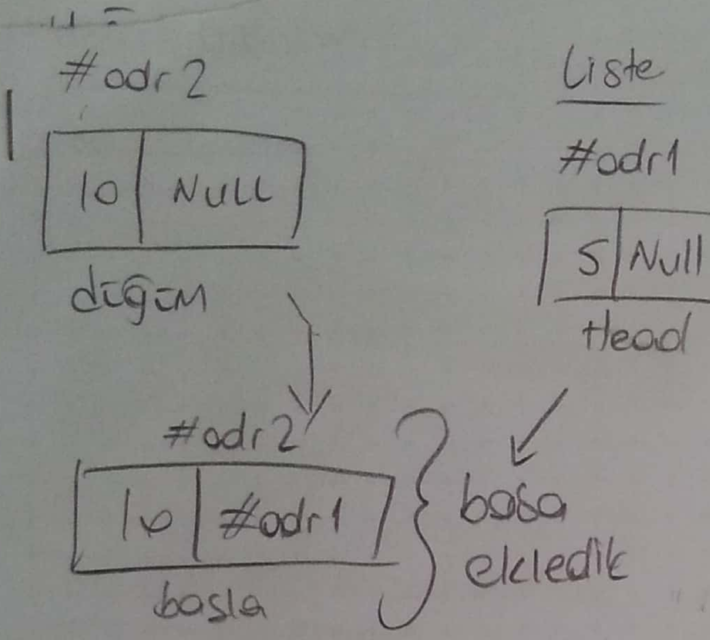
hangi elemanın

çıkacağını

gösterir.

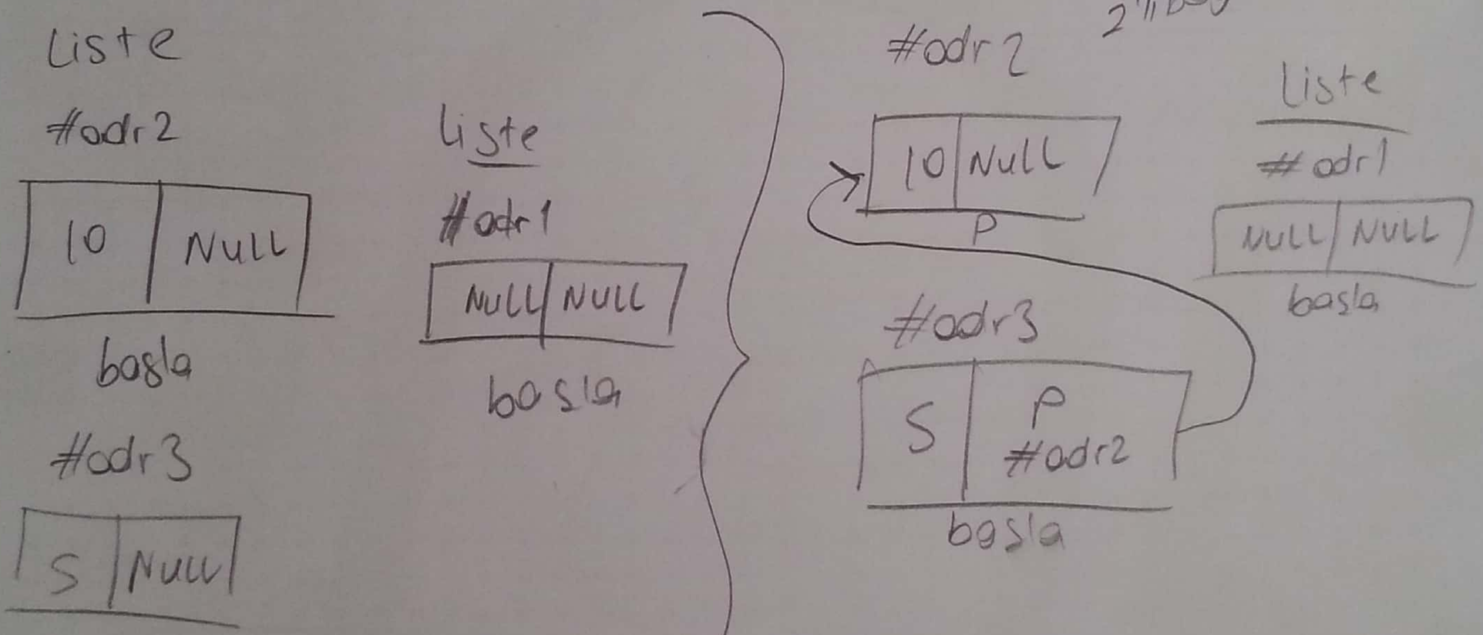
Scanned by CamScanner

Veri Yapilari 1

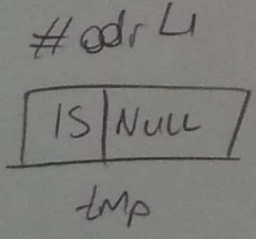


15 → son eleman
5 → head node

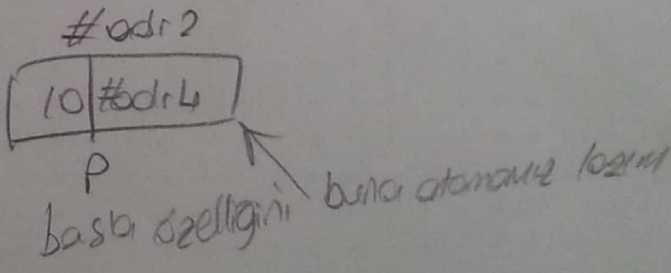
~~B~~ Bağlı liste
araya ekleme
arama
silme
başlama
2'li bağlı liste



Sona eklemek için



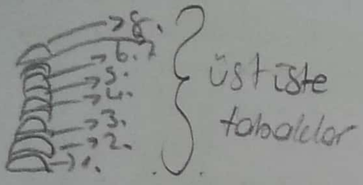
Silmek için
Liste



IĞINLAR (STACKS) ya da yığıt

…st üste veri

tabak örneği



if (stk -> top == -1) //stack boş mu diye kontrol ediyor.

push -> stacklara ekleme işlemi yapar.

pop -> Stack'lerden bir düğümü silmek

top düğümü => Stack'in en üstteki verisini bulmak

initialize -> Bir stack'e başlangıç değerini vermek

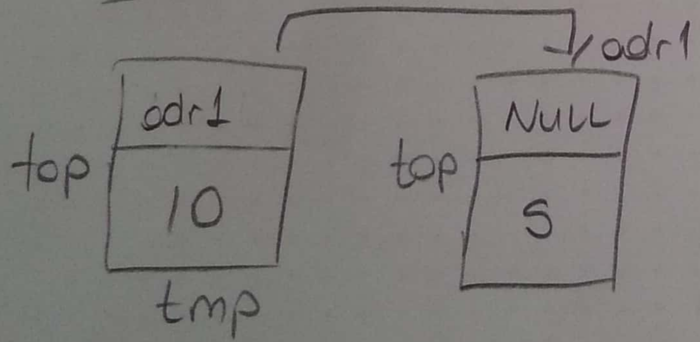
tic tac toe oyununun programı

Push Element info the Stack

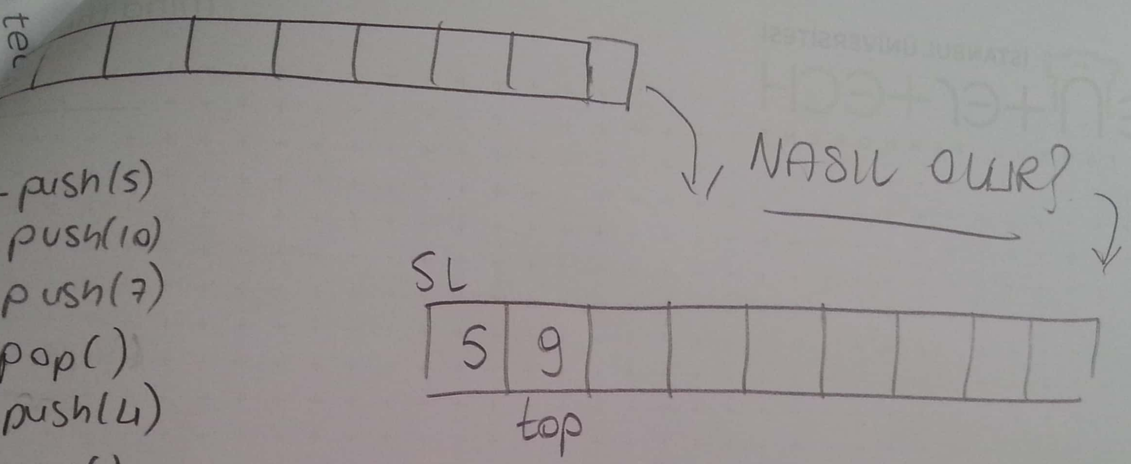
Düğüm oluşturmak

```
node *stack_list::push(node *top,
                            int item)
{
node *tmp;
tmp=new (struct node);
tmp->info =item;
tmp->Link =item;
top =tmp;
return top;
```

Head Node



sl.push(5)
sl.push(10)
sl.push(7)
sl.pop()
sl.push(4)
sl.pop()
sl.pop()
sl.push(9)


NASIL OLUR?
SL
5
9
top