


Örnek Uygulamalar



Örnek Uygulamalar Java-Yığıt

Yığıt (stack) yapısının dinamik dizi ile gerçekleştirimi

- o Stack.java
- o public interface Stack<T>
- o { public boolean empty(); // yığıt boş mu test eder
- o public T top(); // tepedeki elemanı ver (silme)
- o public T pop(); // tepedeki elemanı sil ve ver
- o public void push(T item); // item'i yığıtın tepesine ekle
- o public void clear(); // Yığıtı boşalt }

Yığıt (stack) yapısının dinamik dizi ile gerçekleştirimi

- o `ArrayStack.java`
- o `public class ArrayStack<T> implements Stack<T>`
- o `{`
- o `private static final int DEFAULT_SIZE = 10;`
- o `private T[] array; // yığıt elemanlarını tutan dizi`
- o `private int top; // son eklenen elemanın indisi`
- o `public ArrayStack() // kurucu sınıf`
- o `{ array = (T[]) new Object[DEFAULT_SIZE]; top = -1; }`
- o `public boolean empty() // yığıt boş mu test eder`
- o `{ return (top==-1); }`

Yığıt (stack) yapısının dinamik dizi ile gerçekleştirimi

- `ArrayStack.java`
- `public T top() // tepedeki elemanı ver (silme)`
- `{ if (empty()) return null;`
- `return array[top];`
- `}`
- `public T pop() // tepedeki elemanı sil ve ver`
- `{ if (empty()) return null;`
- `return array[top--];`
- `}`

Yığıt (stack) yapısının dinamik dizi ile gerçekleştirimi

- `ArrayStack.java`
- `public void push(T item) // item'i yığıtın tepesine ekle`
- `{ if (top+1==array.length)`
- `{ T[] newArray = (T[]) new Object[array.length * 2];`
- `for (int i=0; i<array.length; i++)`
- `newArray[i] = array[i];`
- `array = newArray;`
- `}`
- `array[++top] = item;`
- `}`
- `public void clear() // Yığıtı boşalt`
- `{ top = -1; }`
- `}`

Yığıt (stack) yapısının dinamik dizi ile gerçekleştirimi

- TestArrayStack.java
- public class TestArrayStack
- {
- public static void main(String[] args)
- { Stack<String> yigit = new ArrayStack<String>();
- yigit.push("a"); yigit.push("b");
- System.out.println("Yigittaki son eleman: " + yigit.top());
- yigit.push("c");
- String s = yigit.pop();
- System.out.println("Yigittan silinen eleman: " + s);
- yigit.push("d");
- }
- }

Yığıt (stack) yapısının ArrayList ile gerçekleştirimi

- Stack.java // Daha önce verildi
- ArrayListStack.java
- import java.util.*;
- public class ArrayListStack<T> implements Stack<T>
- { private ArrayList<T> array; // yığıt elemanlarını tutan dizi
- public ArrayListStack() // kurucu sınıf
- { array = new ArrayList<T>(); }
- public boolean empty() // yığıt boş mu test eder
- { return array.size()==0; }
-
- public T top() // tepedeki elemanı ver (silme)
- { if (empty()) return null;
- return array.get(array.size()-1);
- }
-

Yığıt (stack) yapısının ArrayList ile gerçekleştirimi

- ArrayListStack.java
- public T pop() // tepedeki elemanı sil ve ver
- {
- if (empty()) return null;
- return array.remove(array.size()-1);
- }
- public void push(T item) // item'i yığıtın tepesine ekle
- { array.add(item); }
- public void clear() // Yığıtı boşalt
- { array.clear(); }
- }

Yığıt (stack) yapısının ArrayList ile gerçekleştirimi

```
○ TestArrayListStack.java
○ public class TestArrayListStack
○ {
○     public static void main(String[] args)
○     {
○         Stack<String> yigit = new ArrayStack<String>();
○         yigit.push("a");
○         yigit.push("b");
○         System.out.println("Yigittaki son eleman: " + yigit.top());
○         yigit.push("c");
○         String s = yigit.pop();
○         System.out.println("Yigittan silinen eleman: " + s);
○         yigit.push("d");
○     }
○ }
```

Yığıt (stack) yapısının bağlantılı liste ile gerçekleştirimi

- Stack.java // Daha önce verildi
- LinkedListStack.java
- `public class LinkedListStack<T> implements Stack<T>`
- `{ private Node<T> top = null; // yığıtın tepesindeki eleman`
- `public boolean empty() // yığıt boş mu test eder`
- `{ return top==null; }`
- `public T top() // tepedeki elemanı ver (silme)`
- `{ if (empty()) return null;`
- `return top.data;`
- `}`
- `public T pop() // tepedeki elemanı sil ve ver`
- `{ if (empty()) return null;`
- `T temp = top.data;`
- `top = top.next;`
- `return temp;`
- `}`

Yığıt (stack) yapısının bağlantılı liste ile gerçekleştirimi

- o LinkedListStack.java
- o public void push(T item) // item'i yığıtın tepesine ekle
 - o { Node<T> newNode = new Node<T>();
 - o newNode.data = item;
 - o newNode.next = top;
 - o top = newNode;
 - o }
- o public void clear() // Yığıtı boşalt
 - o { top = null; }
- o // Yığıt elemanlarını tutan liste elemanı (inner class)
- o class Node<T>
 - o { public T data;
 - o public Node<T> next;
 - o }
- o }

Yığıt (stack) yapısının bağlantılı liste ile gerçekleştirimi


- TestLinkedListStack.java
- public class TestLinkedListStack
- { public static void main(String[] args)
- { Stack<Character> yigit = new LinkedListStack<Character>();
- yigit.push('a');
- yigit.push('b');
- yigit.push('c');
- System.out.println("Yigittaki elemanlar cikariliyor: ");
- while (!yigit.empty())
- {
- System.out.println(yigit.pop());
- }
- }
- }

Hanoi Kuleleri– Özyinelemeli Çözüm- C#

- using System;
- using System.Collections.Generic;
- using System.Text;
- namespace Hanoi
- { class Program
- { static void Main(string[] args)
- { int x; char from='A', to='B', help='C';
- do {
- try
- { Console.Write(" input number of disk: "); x = Int32.Parse(Console.ReadLine()); }
- catch (FormatException e) { x = -10; }
- } while(x== -10 || x>10);

Hanoi Kuleleri– Özyinelemeli Çözüm- C#

- `Console.WriteLine("\n from = A, to = B, help = C\n");`
- `hanoi(x, from, to, help); Console.Read();`
- `}`
- `static void hanoi(int x, char from, char to, char help)`
- `{ if (x > 0)`
- `{ hanoi(x - 1, from, help, to);`
- `move(x, from, to);`
- `hanoi(x - 1, help, to, from);`
- `}`
- `}`
- `static void move(int x, char from, char to)`
- `{ Console.WriteLine(" move disk "+x+" from "+from+" to "+to); }`
- `}`
- `}`



Örnek Uygulamalar JAVA-Kuyruk

Kuyruk (queue) yapısının dinamik dizi ile gerçekleştirimi

- Queue.java
- public interface Queue<T>
- {
- public boolean empty(); // kuyruk boş mu test eder
- public T getFront(); // kuyruğun önündeki elemanı ver (silme)
- public T dequeue(); // kuyruğun önündeki elemanı sil ve ver
- public void enqueue(T item); // item'i kuyruğun arkasına ekle
- public void clear(); // kuyruğu boşalt
- }

Kuyruk (queue) yapısının dinamik dizi ile gerçekleştirimi

- `ArrayQueue.java`
- `public class ArrayQueue<T> implements Queue<T>`
- `{ private static final int DEFAULT_SIZE = 10;`
- `private T[] array; // kuyruk elemanlarını tutan dizi`
- `private int size; // kuyruktaki eleman sayısı`
- `private int front; // en önce eklenen elemanın indisi`
- `private int back; // en son eklenen elemanın indisi`
- `public ArrayQueue() // kurucu sınıf`
- `{ array = (T[]) new Object[DEFAULT_SIZE]; clear(); }`
- `public boolean empty() // kuyruk boş mu test eder`
- `{ return size==0; }`

Kuyruk (queue) yapısının dinamik dizi ile gerçekleştirimi

- `ArrayQueue.java`
- `public T getFront() // kuyruğun önündeki elemanı ver (silme)`
- `{`
- `if (empty()) return null;`
- `return array[front];`
- `}`
- `public T dequeue() // kuyruğun önündeki elemanı sil ve ver`
- `{`
- `if (empty()) return null;`
- `size--;`
- `T temp = array[front];`
- `front = increment(front);`
- `return temp;`
- `}`

Kuyruk (queue) yapısının dinamik dizi ile gerçekleştirimi

- `ArrayQueue.java`
- `public void enqueue(T item) // item'i kuyruğun arkasına ekle`
- `{`
- `if (size==array.length) // kuyruk dolu`
- `{`
- `T[] newArray = (T[]) new Object[array.length * 2];`
- `for (int i=0; i<size; i++, front=increment(front))`
- `newArray[i] = array[front];`
- `array = newArray;`
- `}`
- `back = increment(back);`
- `array[back] = item;`
- `size++;`
- `}`
-

Kuyruk (queue) yapısının dinamik dizi ile gerçekleştirimi

- `ArrayQueue.java`
- `private int increment(int i)`
- `{`
- `i++;`
- `if (i==array.length) i=0;`
- `return i;`
- `}`
- `public void clear() // kuyrukı boşalt`
- `{`
- `size = 0;`
- `front = 0;`
- `back = -1;`
- `}`
- `}`

Kuyruk (queue) yapısının dinamik dizi ile gerçekleştirimi

- TestArrayQueue.java
- public class TestArrayQueue
- { public static void main(String[] args)
- { Queue<Integer> kuyruk = new ArrayQueue<Integer>(); int say=1;
- for (int i=1; i<=15; i++)
- kuyruk.enqueue(i);
- for (int i=1; i<=10; i++)
- System.out.println((say++) + ".kuyruk elemani: " + kuyruk.dequeue());
- for (int i=11; i<=25; i++)
- kuyruk.enqueue(i);
- say=1;
- while (!kuyruk.empty())
- System.out.println((say++) + ".kuyruk elemani: " + kuyruk.dequeue());
- }
- }

Kuyruk (queue) yapısının ArrayList ile gerçekleştirimi

- Queue.java // daha önce verilmişti
- ArrayListQueue.java
- import java.util.ArrayList;
- public class ArrayListQueue<T> implements Queue<T>
- { private ArrayList<T> array; // kuyruk elemanlarını tutan dizi
- public ArrayListQueue() // kurucu sınıf
- { array = new ArrayList<T>(); }
-
- public boolean empty() // kuyruk boş mu test eder
- { return array.size()==0; }
- public T getFront() // kuyruğun önündeki elemanı ver (silme)
- { if (empty()) return null;
- return array.get(0);
- }

Kuyruk (queue) yapısının ArrayList ile gerçekleştirimi

- ArrayListQueue.java
- public T dequeue() // kuyruğun önündeki elemanı sil ve ver
- {
- if (empty()) return null;
- return array.remove(0);
- }
- public void enqueue(T item) // item'i kuyruğun arkasına ekle
- {
- array.add(item);
- }
- public void clear() // kuyruğu boşalt
- {
- array.clear();
- }
- }

Kuyruk (queue) yapısının ArrayList ile gerçekleştirimi

- TestArrayListQueue.java
- public class TestArrayListQueue
- {
- public static void main(String[] args)
- {
- Queue<Integer> kuyruk = new ArrayListQueue<Integer>();
-
- for (int i=1; i<=25; i++)
- kuyruk.enqueue(i);
-
- int say=1;
- while (!kuyruk.empty())
- System.out.println((say++) + ".kuyruk elemani: " + kuyruk.dequeue());
- }
- }

Kuyruk (queue) yapısının bağlantılı liste ile gerçekleştirimi

- Queue.java // daha önce verilmişti
- LinkedListQueue.java
- public class LinkedListQueue<T> implements Queue<T>
- {
- private Node<T> front, back; // kuyruk başı ve sonu
- public LinkedListQueue() // kurucu sınıf
- { clear(); }
-
- public boolean empty() // kuyruk boş mu test eder
- { return front==null; }
-
- public T getFront() // kuyruğun önündeki elemanı ver (silme)
- { if (empty()) return null;
- return front.data;
- }

Kuyruk (queue) yapısının bağlantılı liste ile gerçekleştirimi

- `public T dequeue() // kuyruğun önündeki elemanı sil ve ver`
- `{ if (empty()) return null;`
- `T temp = front.data; front = front.next; return temp;`
- `}`
- `public void enqueue(T item) // item'i kuyruğun arkasına ekle`
- `{ if (empty()) front = back = new Node<T>(item, null);`
- `else back = back.next = new Node<T>(item,null);`
- `}`
- `public void clear() { front = back = null; } // kuyruğu boşalt`
- `// Kuyruk elemanlarını tutan liste elemanı (inner class)`
- `private class Node<T>`
- `{ private T data; private Node<T> next;`
- `public Node(T data, Node next) { this.data = data; this.next = next; }`
- `}`
- `}`

Kuyruk (queue) yapısının bağlantılı liste ile gerçekleştirimi

- TestLinkedListQueue.java
- public class TestLinkedListQueue
- { public static void main(String[] args)
- { Queue<Integer> kuyruk = new ArrayListQueue<Integer>();
- for (int j=1, i=1; i<=20; i++)
- { if ((int)(Math.random()*2) == 1)
- { System.out.println("Kuyruğa ekle: " + j); kuyruk.enqueue(j++); }
- else
- { System.out.println("Kuyruğun önündeki "
- + kuyruk.dequeue() + " çıkarıldı..");
- }
- }
- }
- }

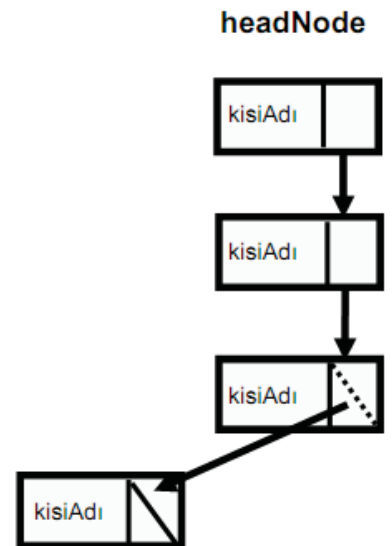
Bağlı Liste ile Kuyruk gerçekleştirimi C#

- **// Queue oluşturma :**
- `class queueNodeC {`
- `public string kisiAdi; public queueNodeC sonraki;`
- `public queueNodeC(string kisiAdi) { this.kisiAdi = kisiAdi; }`
- `}`
- `class queueC`
- `{`
- `public int size; public queueNodeC headNode;`
- `public queueC(string kisiAdi)`
- `{`
- `this.headNode = new queueNodeC(kisiAdi);`
- `this.headNode.sonraki = headNode; this.size = 0;`
- `}`
- `}`
- `queueC kisiKuyruk = new queueC("");`

Bağlı Liste ile Kuyruk gerçekleştirimi C#

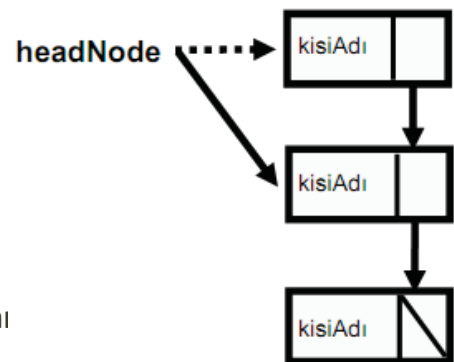
Örnek (C#):

- /* Queue işlemleri :
- boş kuyruk size == 0
- eleman sayısı= size
- eleman ekleme= enqueue(kisiAdi)
- eleman alma= dequeue() */
- public void enqueue(kisiAdi)
- {
- queueNodeC yeniNode = new queueNodeC(kisiAdi);
- queueNodeC aktif = kisiKuyruk.headNode;
- while (aktif.sonraki != aktif)
- { aktif = aktif.sonraki; }
- aktif.sonraki = yeniNode;
- yeniNode.sonraki = yeniNode;
- kisiKuyruk.size++;
- }



Bağlı Liste ile Kuyruk gerçekleştirimi C#

- // Queue işlemleri (eleman alma)
- public void dequeue()
- {
- queueNodeC yeniNode = new queueNodeC(" ");
- yeniNode = kisiKuyruk.headNode;
- kisiKuyruk.headNode = kisiKuyruk.headNode.soni
- kisiKuyruk.size--;
- }



Kuyruk oluşturma, ekleme ve silme C++

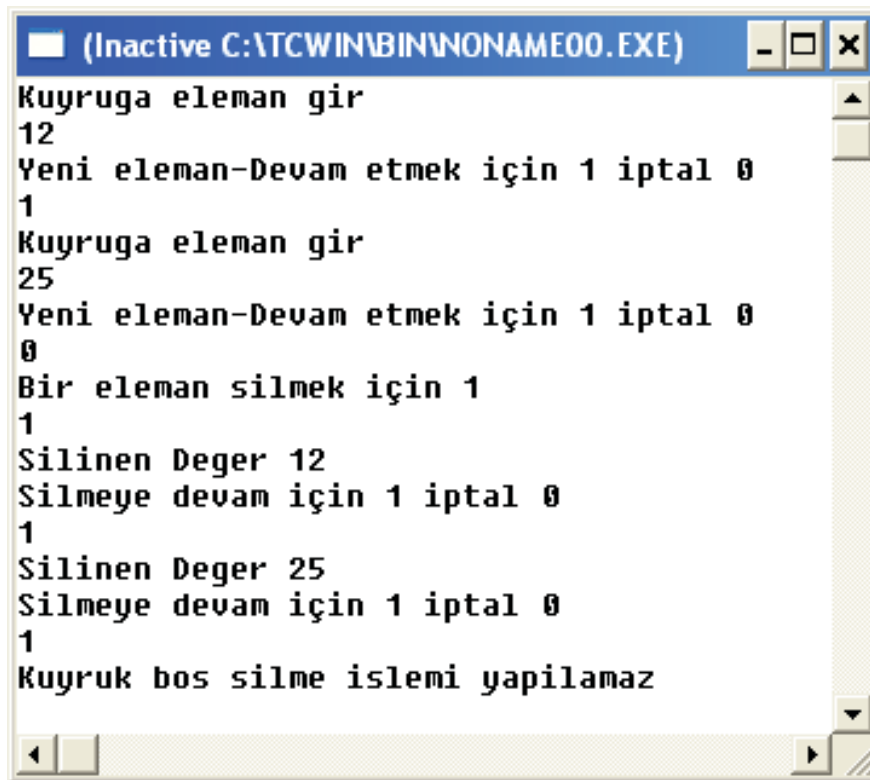
```
○ #include <stdio.h>
○ #include <stdlib.h>
○ #define boyut 10 /*Maksimum kuyruk uzunluğu */

○ void ekle (int kuyruk[], int *arka, int deger)
○ {
○     if(*arka < boyut-1) {
○         *arka= *arka +1;
○         kuyruk[*arka] = deger; }
○     else
○         { printf("Kuyruk doldu daha fazla ilave edilemez\n"); }
○ }

○ void sil (int kuyruk[], int * arka, int * deger)
○ { int i;
○     if(*arka<0) {printf("Kuyruk bos silme islemi yapilamaz\n");}
○     *deger = kuyruk[0];
○     for(i=1;i<=*arka;i++)
○         { kuyruk[i-1]=kuyruk[i]; }
○     *arka=*arka-1;
○ }
```


Kuyruk oluşturma, ekleme ve silme C++

- `main() { int kuyruk[boyut]; int arka, n, deger; arka=(-1);`
- `do {`
- `do { printf("Kuyruğa eleman gir\n"); scanf("%d",°er);`
- `ekle(kuyruk,&arka,deger); //arka değeri arttı`
- `printf("Yeni eleman-Devam etmek için 1 iptal 0 \n"); scanf("%d",&n);`
- `} while(n == 1);`
- `printf("Bir eleman silmek için 1\n"); scanf("%d",&n);`
- `while(n == 1) {`
- `sil(kuyruk,&arka,°er);`
- `printf("Silinen Deger %d\n",deger); printf("Silmeye devam için 1 iptal 0\n");`
- `scanf("%d",&n); }`
- `printf("Eleman girmek için 1 iptal için 0\n"); scanf("%d",&n);`
- `} while(n == 1); }`



```
(Inactive C:\TCWIN\BIN\WONAME00.EXE)
Kuyruğa eleman gir
12
Yeni eleman-Devam etmek için 1 iptal 0
1
Kuyruğa eleman gir
25
Yeni eleman-Devam etmek için 1 iptal 0
0
Bir eleman silmek için 1
1
Silinen Deger 12
Silmeye devam için 1 iptal 0
1
Silinen Deger 25
Silmeye devam için 1 iptal 0
1
Kuyruk bos silme islemi yapilamaz
```

Dizi Üzerinde Çevrimsel Kuyruk- C++

- `#include <stdio.h>`
- `#define N 500 /*Maksimum kuyruk uzunluğu */`
- `#include <stdlib.h>`
- `typedef struct kuyrukyapisi`
- `{ int bas; int son ; int sayac; int D[N]; } KUYRUK ;`
-
- `KUYRUK *M;`
- `void baslangic (KUYRUK *K)`
- `{ K->bas=0;`
- `K->sayac=0;`
- `K->son=0;`
- `}`
-

Dizi Üzerinde Çevrimsel Kuyruk- C++

- void ekle(int veri, KUYRUK *K)
- {
- if(K->sayac>N-1) { printf("Kuyruk dolu"); }
- /*Doğrusal erişimli bir diziye çevrimsel erişim yapılabilmesi için dizi indisi son gözden sonra ilk gözü işaret edebilmesi için artık bölme işlemiyle indis=(indis+1)%N yapılır. */
- K->son=(K->son+1)%N;
- K->D[K->son]=veri;
- K->sayac++;
- }
- void silme(KUYRUK *K,int *deger)
- {
- if((K->sayac)<=0) {printf("Kuyruk bos silme islemi yapilamaz\n");}
- *deger = K->D[K->bas]=0;
- K->bas=(K->bas+1)%N;
- K->sayac--;
- }

Dizi Üzerinde Çevrimsel Kuyruk- C++

```
o main() {
o int veri,n,deger;
o M=(KUYRUK *)malloc(sizeof(KUYRUK));
o baslangic (M);
o do {
o     do { printf("Kuyruğa eleman gir\n");   scanf("%d",&veri); ekle (veri,M);
o         printf("Yeni eleman-Devam etmek için 1 iptal 0 \n"); scanf("%d",&n);
o     } while(n == 1);
o     printf("Bir eleman silmek için 1\n"); scanf("%d",&n);
o     while( n == 1) { silme (M,&deger);
o         printf("Silinen Deger %d\n",deger); printf("Silmeye devam için 1 iptal 0\n");
o         scanf("%d",&n);   }
o     printf("Eleman girmek için 1 iptal için 0\n");   scanf("%d",&n);
o } while(n == 1);
o }
```