C:\Users\bilg\Documents\NetBeansProjects\stack_dizi_tabanli\main.cpp

```
#include <cstdlib>
#include<exception>
#include<iostream>
using namespace std;
#define DEFAULT_KAPASITE 5

class StackEmpty:public exception{
public:
   const char* what() const throw(){
      return "hata: stack bos";
   }
};

template <typename T>
class Stack{
private:
   int enust;//en üstteki elemanın indisi
   int kapasite;//dizi büyüklüğü
   T * dizi;
public:
   Stack(int kapasite = DEFAULT_KAPASITE){
      this->kapasite=kapasite;
      enust=-1;
      dizi=new T[kapasite];
   }

   ~Stack(){
      delete[] dizi;
   }
   void push(const T& data){
      if (length()==kapasite) resize(2*kapasite);
      enust++;
      dizi[enust]=data;
   }
   void pop()throw(StackEmpty){
      if (enust==-1) throw StackEmpty();
      enust--;
   }
   const T& top()throw(StackEmpty){ // peek() en üstteki elemanı al
      if (enust==-1)throw StackEmpty();
      return dizi[enust];
   }
   void clear(){ //ilk duruma getir
      enust=-1;
      //dizi boyutu sabit değilse eskisinin hafıza tutğu yeri serbest bırakıp
      // tekrar oluşturarak boyutunu da default duruma getirebiliriz
      if(dizi != NULL)
         //mevcut durumda dizi hiç NULL olmayacağı için yazmasak da olur
         delete[] dizi;
      kapasite=DEFAULT_KAPASITE;
```

```
        dizi=new T[kapasite];
    }
    bool isEmpty(){ // stack boş mu?
      return enust==-1;
    }
    void resize(int yenikapasite){
        //yer aç. bu fonksiyon olmazsa stack sabit boyutlu olur
       // cout<<"\nyeni kapasite ="<<yenikapasite<<endl;
        T *yeni=new T[yenikapasite];
        for(int i=0;i<=enust;i++)// elemanları yeni diziye gönder
            yeni[i]=dizi[i];
        if(dizi!= NULL)
            //mevcut durumda dizi hiç NULL olmayacağı için yazmasak da olur
            delete[] dizi;
        dizi=yeni;
        kapasite = yenikapasite;
    }
    int length(){
        //eleman sayısı en üstteki indisin 1 fazlası
        return (enust+1);
    }
    /*bool isFull(){ //stack dolu mu?
     * (stack sabit boyutlu dizi ile gerçekleştirilmişse kullanılabilir)
        return enust==kapasite;
    }*/
};

int main(int argc, char** argv) {
    Stack<string> stack1(5);
    try{
        stack1.push("eleman1");
        stack1.push("eleman2");
        stack1.push("eleman3");
        stack1.push("eleman4");
        stack1.push("eleman5");
        stack1.push("eleman6");
        stack1.push("eleman7");

        while(!stack1.isEmpty()){
            cout<<"length :"<<stack1.length()<<endl;
            cout<<"top    :"<<stack1.top()<<endl;
            stack1.pop();
        }

      stack1.pop();//hata verdirir:stack boş olduğu halde pop())
    }
    catch(StackEmpty e){// stack boş hatasını yakala
        cout<<e.what()<<endl;
    }
    catch(exception e){// diğer hatalar
        cout<<e.what()<<endl;
    }
```

```
    return 0;
}
```