string0_true.cpp - korrekte Klasse mit dynamischen Zeichenketten

```
#include <iostream>
using namespace std;

class zk {//Beispiel fuer korrekte Klasse mit dynamischen Zeichenketten
          //Nachteil: Jede Methode, die get_s() ruft, erhaelt eine Kopie
          //des C-strings und muss diesen eigenstaendig freigeben !!
          //
          //Ein Kopierkonstruktor und Zuweisungsoperator fehlt hier noch,
          //vgl. string0_better und string0_best etwas spaeter
    char *s;
  public:
    //Der Konstruktor reserviert neuen Speicher fuer s und kopiert z
    //nach s, damit sind der Parameter z und das Member s vollstaendig
    //unabhaengig, jedoch mit den identischen Werten (Zeichen) belegt:
    zk(char *z = 0):s(z ? strcpy(new char[strlen(z)+1], z) : 0){
       cout<<"Konstruktor zk, s = "<<(this->s ? this->s : "0")<<endl;
       cout<<"        Adresse s = "<<(int *)s<<endl;
    }

    ~zk(){ cout<<"Destruktor zk,  s = "<<(s?s:"0")<<endl; delete [] s; s=0; }

    //Vom Member s wird eine Kopie mit neuem Speicher auf dem heap erzeugt,
    //d.h. die Umgebung, die get_s() aufruft erhaelt die Adresse der Kopie
    //und muss diesen Speicher irgendwann wieder mit delete [] freigeben !!
    //Das muss beim gesamten Einsatz von get_s() beachtet werden !
    char *get_s(){ return s ? strcpy(new char[strlen(s)+1], s) : 0; }
   const char * const cget_s(){ return s; }                  // alternativ mit const

   //Wie der Konstruktor, zusaetzlich muss vor der Neubelegung von s das bisherige s
    //mit delete [] s freigegeben werden:
   void set_s(char *z=0){delete [] s;this->s =z?strcpy(new char[strlen(z)+1], z):0;}
};
```

string0_true.cpp - korrekte Klasse mit dynamischen Zeichenketten

```
void main(){
     char *temp = "HTW Dresden";
     char *z = strcpy(new char[strlen(temp)+1],temp);
     cout<<"        Adresse z = "<<(int *)z<<endl;
     zk *s1 = new zk(z);
     delete [] z; z=0;          // s1->s wird nicht beeinflusst
     char *s01 = s1->get_s();  // Kopie von s1->s ueber s1->get_s()
     cout<<"s1->s = "<<(s01?s01:"0")<<endl;
     delete [] s01;             // Freigabe der Kopie von s1->S

     const char * const s02 = s1->cget_s();
     cout<<"s02 = "<<(s1->cget_s() ? s1->cget_s() : "0")<<endl;
  // delete [] s02;             // nicht zu verhindern, bringt Programm zum Absturz !
  // s02 = 0;                   // Compile-Error wegen const

     s1->set_s("TU Dresden");
     s01 = s1->get_s();         // Neue Kopie von s1->s
     cout<<"s1->s = "<<(s01?s01:"0")<<endl;

     delete [] s01; s01 = 0;   // Freigabe der Kopie von s1->S
     delete s1; s1=0;
}
```

```
        Adresse z = 00322AB0
Konstruktor zk, s = HTW Dresden
        Adresse s = 00322D40
s1->s = HTW Dresden
s02 = HTW Dresden
s1->s = TU Dresden
Destruktor zk,  s = TU Dresden
```