string0_true_1.cpp

```
#include <iostream>  // string0_true_1.cpp
using namespace std;

class zk { // Beispiel fuer Klasse mit Zeichenkette, wobei s nur lesbar ist

            char const * const s;// Zeiger auf Zeichen (Zeichenkette), OK

        public:
            zk(char const * const z = 0):s(z?strcpy(new char[strlen(z)+1],z):0){

//          s = z;         // Compile Error

            cout<<"Konstruktor zk, s = "<<(this->s ? this->s : "0")<<endl;
            }

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

            char const * const get_s(){ return s; }

 // nicht moeglich, Compile-Error:
 //         void set_s(char *z = 0){ this->s = z; }

 //         void set_s(char *z = 0){
 //             delete [] s;
 //             this->s = z?strcpy(new char[strlen(z)+1],z):0;
 //         }
};
```

string0_true_1.cpp

```
void main(){
            char *u = "HTW Dresden";

            char *z = strcpy(new char[strlen(u)+1], u);

            zk *s1 = new zk(z);

            delete [] z; z = 0;

//          char *tt = s1->get_s();     // Compile Error

            char const * const t = s1->get_s();

            cout<<"t = "<<(t?t:"0")<<endl;

//          delete [] t; t=0;          // Compile Error
//           t = "HTW";               // Compile Error

            delete s1; s1 = new zk("HTW");

            zk *s2 = new zk(*s1);   // Kopie von *s1, s1->s und s2->s
                                    // verweisen auf identischen Speicher
                                    // Ausweg: eigener Kopierkonstruktor

            delete s1; s1 = 0;

//  s2->s ex. nicht:
            cout<<"s2->s = "<<(s2->get_s()?s2->get_s():0)<<endl;

//          delete s2; s2=0;   // Abbruch

}
```