

# Beispiel für Überladung des Indexoperators: oper1.cpp

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

class intvec {
    size_t len;          // Anzahl Vektorelemente
    int *vec;            // int-Vektor
public:
    intvec(size_t len=1):len(len), vec(len ? new int[len] : 0){
        for(size_t i=0; i<=len; (*this)[i++]=0); //i<=len Fehler abgefangen
        // bisher: vec[i++]=0 //Indexfehler mit Abbruch
    }

    ~intvec(){ if(this){ delete [] vec, vec=0, len=0; } } //this=0 moeglich

    size_t getl() const { return this ? len : 0; } //this=0 moeglich

    /*const*/ int &operator[](size_t idx) const { //Indexoperator

        static int dummy; //dummy ex. auch ausserhalb
        int *res = &dummy; //res als Adresse von dummy
        if(this){ //this=0 moeglich
            if(idx < len) res = vec+idx; //Adresse von vec[idx]
            else cout<<"Index="<<idx<<" >= "<<len<<"\n";
        }
        else cout<<"Vektor existiert nicht\n";
        return *res; //vec[idx] oder dummy
    }
};

void main(){
    intvec num(10);
    for(size_t i=0; i<num.getl(); num[i]=i++); //num.operator[](i)=i++

    for(size_t i=0; i<= num.getl(); i++){ // <= ist falsch !
        cout<<num[i]<<endl; //Zugriffsfehler bei num.vec[i] weil private
    }

    // num[50]= 50; nicht erlaubt bei const int &operator[]()
    num[50]=50; //Indexfehler, d.h. dummy=50
    cout<<"num[50] = "<<num[50]<<endl; //Indexfehler, d.h. dummy=50

    intvec *pp = 0; // pp als Nullzeiger
    (*pp)[0] = 5; // pp->operator[](0) = 5; --> dummy = 5
    pp->~intvec(); // expliziter Aufruf des Destruktors
    delete pp; pp=0; // impliziter Aufruf des Destruktors
    cin.get();
}

Index=10 >= 10
0
1
2
3
4
5
6
7
8
9
Index=10 >= 10
0
Index=50 >= 10
Index=50 >= 10
num[50] = 50
```