

## Zählen von Objekten einer bestimmten Klasse

### Ziel, Inhalt

- Zur Übung versuchen wir eine Klasse zu schreiben, mit der es möglich ist Objekte einer bestimmten Klasse zu zählen. Wir werden den ++ und den -- Operator überschreiben und das Objekt verwenden, als wäre es ein long-Datentyp

Zählen von Objekten einer bestimmten Klasse	1
Ziel, Inhalt	1
Objekte einer bestimmten Klasse zählen	2
Idee	2
Einführung	2
Das Problem	2
Ausgabe der Memory Leaks	2
Lösungsansatz	3
Beispiel zu Anwendung	3
Übung	4
Erzeugen einer Klasse Count	4
Der Konstruktor	4
Datenelemente für die Anzahl Elemente und die maximale Anzahl Elemente	5
Überschreiben des ++ und des -- Operators	5
Der Destruktor	7

## Objekte einer bestimmten Klasse zählen

### Idee

### Einführung

In der letzten Lektion haben wir eine Klasse geschrieben, die eine einfache Art der Objektzählung ermöglichte. Bei einem unserer Projekte hatten wir ein anderes Problem.

### Das Problem

Wir konnten feststellen, dass wir Speicher nicht wieder freigegeben hatten. Es erschienen verschiedene Warnungen wenn wir das Programm beendeten. Wir konnten jedoch nicht feststellen, welche und wie viele Objekte wir nicht wieder freigegeben hatten.

### Ausgabe der Memory Leaks

Um festzustellen ob aller Speicher wieder freigegeben wird genügt es folgenden Funktion möglichst für im Programm aufzurufen:

```
#include <crtdbg.h>

int main()
{
    // einschalten der Memory-Überwachung
    _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF |
                  _CRTDBG_LEAK_CHECK_DF);

    // einen int Zeiger auf
    // neuen Speicher zeigen
    // lassen
    int* test = new int(5);

    return 0;
}
```

Leider ist diese Funktion spezifisch für die C-Runtime Library von Microsoft. Wenn das Programm beendet wird findet ihr im Debug Fenster der Entwicklungsumgebung folgenden Eintrag:

```
Detected memory leaks!
Dumping objects ->
{41} normal block at 0x00320FF0, 4 bytes long.
  Data: <    > 05 00 00 00
Object dump complete.
```

Wir erhalten also die Information, dass wir Speicher nicht freigegeben haben, wie viel und an welcher Speicheradresse. Es wird sogar der Speicherinhalt angezeigt.

Das gleiche geschieht für Objekte, die man vergisst zu löschen. Die Information reicht aber im Normalfall nicht aus.

### Lösungsansatz

Um von bestimmten Klassen genau zu wissen, wie viele Objekte wir erzeugt haben und nicht wieder freigegeben hatten, brauchten wir einen anderen Mechanismus. Wir schreiben eine einfache Klasse, von der man jeweils ein Objekt erzeugt um alle Objekte einer anderen Klasse zu zählen. Der Konstruktor dieser Klasse nimmt einen string als Namen, und hat praktischerweise den ++ und den -- Operator überschrieben. Bei der Anwendung gilt es nur noch jedes Mal ++ oder -- aufzurufen, je nach dem ob ein Objekt erzeugt wurde, oder eines zerstört wurde.

Wenn unser Count - Objekt am Ende des Programms zerstört wird, soll es ausgegeben, wie viele Objekte sich noch im Speicher befinden, indem es im Konstruktor eine interne Zählvariable und den Namen auswertet.

### Beispiel zu Anwendung

```
// Einen Zähler für Objekte der Klasse
// Person
Count personCount("Personen-Objekte");

// Standardkonstruktor
Person::Person()
{
    ++personCount;
}

// Kopierkonstruktor
Person::Person(const Person& c)
{
    // hier noch Code zum Kopieren
    _nachname = c._nachname;
    _vorname = c._vorname;

    ++personCount;
}

// Destruktor
Person::~~Person()
{
    --personCount;
}
```

## Übung

### Erzeugen einer Klasse Count

Der erste Schritt besteht wie immer im Erzeugen einer Header-Datei für unsere Klasse *Count*. Vergiss die `#include` - Blocker nicht!

```
#ifndef COUNT_H
#define COUNT_H

class Count
{
};

#endif
```

Und wieder einmal wurde im wundersamen Reich der Bits und Bytes eine neue Lebensform definiert!

### Der Konstruktor

Als Konstruktor brauchen wir nur einen, dem wir den Namen der gezählten Klasse mitgeben können.

```
#include <string>

class Count
{
public:
    // Konstruktor
    Count(std::string name)
    {
    }
};
```

Natürlich müssen wir uns diesen Namen merken:

```
class Count
{
public:
    // Konstruktor
    Count(std::string name) :
        _name(name)
    {
    }
private:
    std::string _name;
};
```

## Datenelemente für die Anzahl Elemente und die maximale Anzahl Elemente

Wir wollten am Ende folgende Ausgaben erzielen:

13 von 455 Personen-Objekte wurden nicht freigegeben.

Hierfür müssen wir uns also die Anzahl Objekte und zusätzlich die maximale Anzahl merken.

```
class Count
{
public:
    // Konstruktor
    Count(std::string name) :
        _name(name),
        _count(0),
        _maxCount(0)
    {
    }

private:
    std::string _name;
    unsigned long _count;
    unsigned long _maxCount;
};
```

## Überschreiben des ++ und des -- Operators

Wir wollen ein Objekt der Klasse *Count* ähnlich wie eine Zahl behandeln, indem wir ++ oder -- aufrufen. Damit der Compiler weiss was er machen muss schreiben wir einfach ein wenig Code hierfür.

```
class Count
{
public:
    // Konstruktor
    Count(std::string name) :
        _name(name),
        _count(0),
        _maxCount(0)
    {
    }

    void operator++()
    {
    }

    void operator--()
    {
    }

private:
```

Im ++ Operator erhöhen wir den internen Zähler `_count` und, falls dieser grösser ist als die bisher maximale Anzahl, erhöhen wir auch den `_maxCount`.

Das ist nötig, da Objekte auch wieder entfernt werden und dabei die Variable `_count` erniedrigt wird.

```
void operator++()
{
    ++_count;
    if(_count > _maxCount)
    {
        _maxCount = _count;
    }
}

void operator--()
{
    --_count;
}
```

## Der Destruktor

Beim Zerstören unseres *Count* - Objektes muss dieses noch ausgeben, was es weiss.

```
#ifndef COUNT_H
#define COUNT_H

#include <string>
#include <sstream>
#include <iostream>

class Count
{
public:
    // Konstruktor
    Count(std::string name) :
        _name(name),
        _count(0),
        _maxCount(0)
    {
    }

    // Destruktor
    ~Count()
    {
        std::cout << _count;
        std::cout << " von ";
        std::cout << _maxCount;
        std::cout << " " << _name;
        std::cout << " wurden nicht freigegeben";
        std::cout << std::endl;
    }

    void operator++()
    {
        ++_count;
        if(_count > _maxCount)
        {
            _maxCount = _count;
        }
    }

    void operator--()
    {
        --_count;
    }

private:
    std::string _name;
    unsigned long _count;
    unsigned long _maxCount;
};

#endif
```