# Übungen OOSWE/Progr. 2 (C++) – KE 4

**Der C/C++-Coding Styleguide ist einzuhalten.**

Folgende Einstellungen sind für Debug und Release (All Configurations) vorzunehmen:

| **Einstellung** | **Wert** |
|---|---|
| Solution Platform | x86 |
| Properties->Conf. Properties->C/C++->General->Warning Level | Level4 (/W4) |
| Properties->Conf. Properties->C/C++->General->Treat Warnings As Errors | Yes (/WX) |
| Properties->Conf. Properties->C/C++->General->SDL checks | Yes (/sdl) |
| Properties->Conf. Properties->C/C++->Code Generation->Basic Runtime Checks | Default |
| Properties->Conf. Properties->C/C++->Code Generation->Security Checks | Enable Security Checks (/GS) |
| Properties->Conf. Properties->C/C++->Language->C++ Language Standard | ISO C++ 20 Standard |

Legen Sie sich eine Solution an, die alle Aufgaben als Projekte enthält.

Die geforderten Kommentare sind in Englisch zu hinterlegen.

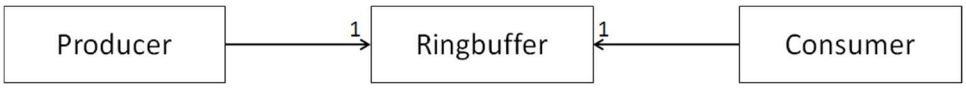
**Aufgabe 1:**

**1.1** Implementieren Sie ein Programm, welches die Klasse Ringbuffer aus KE03_AG2 wiederverwendet. Legen Sie sich dazu das Projekt KE04_AG1 an.

- Main.cpp
- Ringbuffer.h (Kopieren Sie diese Datei aus KE03_AG2 in das neue Projektverzeichnis)
- Ringbuffer.cpp (Kopieren Sie diese Datei aus KE03_AG2 in das neue Projektverzeichnis)

Fügen Sie die beiden Ringbuffer-Dateien mittels *Add* und *Existing Item..* dem Projekt hinzu.

Implementieren Sie zusätzlich die beiden Klassen **Producer** (Producer.h und Producer.cpp) sowie **Consumer** (Consumer.h und Consumer.cpp), welche eine Assoziation zu der Ringbuffer-Klasse haben. Die einfache Assoziation wird in beiden Klassen mit einem privaten Zeiger auf ein Ringbuffer-Objekt realisiert. Producer und Consumer sollen sich im namespace „Client“ befinden.




Die Klasse Producer soll eine Methode s32WriteToRingbuffer enthalten, während die Klasse Consumer eine Methode s32ReadFromRingbuffer enthalten soll. S32WriteToRingbuffer bekommt ein int32_t-Wert als Referenz übergeben, dieser int32_t-Wert wird dann im assoziierten Ringbuffer abgespeichert. s32ReadFromRingbuffers soll jeweils ein int32_t-Wert aus dem assoziierten Ringbuffer per Referenz einlesen.

Implementieren Sie den Konstruktor und den Destruktor von Producer und Consumer (inklusive Testausgaben in Debug). Da Producer und Consumer immer genau ein Ringbuffer-Objekt kennen, ist dieses per „Constructor Injection“ als Zeiger bei der Instanziierung (Konstruktor) zu übergeben.
Die Elemente des Ringbuffers sollen im Konstruktor von Ringbuffer mit 0 initialisiert werden. Tipp: Durch welchen „kleinen Trick“ können Sie dies bei new im Konstruktor von Ringbuffer erzwingen?

**1.2** Deklarieren und implementieren Sie eine Funktion void vTestCreateObjects (void), welche von main aufgerufen wird.

```
void vTestCreateObjects (void);

int main(void)
{
   vTestCreateObjects();

   return 0;
}
```

In dieser Funktion sollen dynamisch ein Ringbuffer mit zehn Elementen sowie ein assoziierter Producer (statisch) und Consumer (dynamisch) instanziiert werden.

Testen Sie Ihr Programm indem Sie abwechselnd s32WriteToRingbuffer und s32ReadFromRingbuffer und die Testausgaben von Ringbuffer aufrufen. Überprüfen Sie Ihr Programm auf Memory Leaks. Beseitigen Sie diese ggf.

**Aufgabe 2:**
**2.1** Es soll eine Klasse IOChannel mit **drei** I/O Kanälen (Ringbuffer) als Klasse realisiert werden. Diese Klasse soll eine Komposition mit drei Ringbuffer haben. IOChannel soll sich im gleichen Namespace befinden wie Ringbuffer befinden. Verwenden Sie die Implementierung von Ringbuffer aus KE04_AG1.


Über diese drei Kanäle (Ringbuffer) können Elemente (int32_t) gelesen und gespeichert werden. Jeder Kanal (Ringbuffer) kann maximal zehn Elemente puffern. Fügen Sie dem Ringbuffer noch einen leeren Konstruktor hinzu (Standardgröße sei 10U).
Die Komposition können Sie einfach als Array von privaten Objektvariablen des Typs Ringbuffer realisieren (`Ringbuffer Channels[3];`).

IOChannel bietet die folgenden Methoden an (public):
```
int32_t s32WriteElementToChannel(uint32_t u32Channel, const int32_t& rs32Element);
int32_t s32ReadElementFromChannel(uint32_t u32Channel, int32_t& rs32Element);
```

Als Rückgabewerte sollen die Rückgabewerte des Ringbuffers verwendet werden. U32Channel muss dabei <3U sein. Verwenden Sie zur Überprüfung von u32Channel ein assert (http://www.gidf.de/).

Fügen Sie der Klasse IOChannel noch einen Konstruktor und einen Destruktor (inklusive Testausgaben in Debug) hinzu.

**2.2** Deklarieren und implementieren Sie eine Funktion void vTestCreateObjects (void), welche von main aufgerufen wird.

```
void vTestCreateObjects (void);

int main(void)
{
   vTestCreateObjects();

   return 0;
}
```

In dieser Funktion sollen zwei IOChannel Objekte (1x dynamisch und 1x statisch) instanziiert werden. Testen Sie beide IOChannels unter Verwendung von Ausgaben auf der Konsole. Überprüfen Sie Ihr Programm auf Memory Leaks. Beheben Sie ggf. die Memory Leaks.

Welche Aussage können Sie bezüglich der Aufrufreihenfolge des Konstruktors und des Destruktors von Objektvariablen im Vergleich zum übergeordneten Objekt treffen? Schreiben Sie dies als Kommentar über main.