**10. Aufgabenblatt**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vorname** | **Nachname** | **Matrikelnummer** | **Fachrichtung** |
| Philipp | Huber | 63326 | Mabb |
| Marius | Grumer | 63284 | Mabb |

|  |  |
| --- | --- |
| **Punkte** | xx/yy |
| **Bestanden** | ja/nein |

In diesem Dokument sind die Lösungen zur PDF zu finden.

1. Aufgabe

|  |
| --- |
| #ifndef CTESTCLASS\_H\_  #define CTESTCLASS\_H\_  #include < cstdint >  class CTestClass  {  public:  CTestClass();  ~CTestClass();  private:  uint8\_t mID;  };  #endif /\* CTESTCLASS\_H\_ \*/ |

Der Konstruktor muss ohne void deklariert werden.

1. Aufgabe

|  |
| --- |
| #include "Header.h"  CTestClass::CTestClass()  {  mID = 0;  } |

Der Attribut mID kann durch eine Initialisierung in dem Konstruktor der Klasse mit einem beliebigen Wert initialisiert werden. Denkbar ist auch eine Initialisierung über einen Konstruktor mit Parameter. So wird der Default-Ctor von mID verhindert.

1. Aufgabe

|  |
| --- |
| #ifndef CTESTCLASS\_H\_  #define CTESTCLASS\_H\_  #include < cstdint >  class CTestClass  {  public:  CTestClass();  ~CTestClass();  bool getMID(uint8\_t& pID);  private:  uint8\_t mID;  };  #endif /\* CTESTCLASS\_H\_ \*/ |
| #include "Header.h"  CTestClass::CTestClass()  {  mID = 0;  }  bool CTestClass::getMID(uint8\_t& pID)  {  pID = mID;  return true;  } |

1. Aufgabe

* Es wird kein Copy-Konstruktor aufgerufen, somit wird weniger Speicherplatz benötigt.
* Es können mehrere Rückgabeparameter übergeben werden.

1. Aufgabe

|  |
| --- |
| bool CTestClass::setMID(const uint8\_t& pID)  {  //pID = 20; //geht nicht, da schreibgeschützt  this->mID = pID;  return true;  } |