| 10 Programmierung II | |
|---|---|
| Semester | 2 |
| Dauer (Semester) | einsemestrig |
| Credit Points | 5 |
| Pflicht/ Wahlpflicht | Pflicht |
| Häufigkeit des Angebotes/ Verwendbarkeit | Jedes Sommersemester |
| Modulverantwortliche(r) | Prof. Dr. Carsten Link |
| Lerngebiet | Grundlagen der Informationstechnik |
| Teilnahmevoraussetzungen | Erfolgreicher Abschluss des Moduls Programmierung I wird empfohlen |
| Lernergebnisse | Die Studierenden demonstrieren, dass sie über ein mentales Modell von Informationsdarstellung und Programmablauf verfügen, indem sie einen Rechner (mit rekursiv absteigendem Parser) für Bool'sche Ausdrücke konstruieren.<br>Die Studierenden können<br>• mit den Besonderheiten des Programmablaufs bei der Mikrocontrollerprogrammierung umgehen, indem sie in einem Programm asynchrone Ereignisse mit den adäquaten Sprachmitteln behandeln und die geeigneten Datenstrukturen auswählen.<br>• Klassen entwerfen, um mit angepassten Typen Aufgabenstellungen besser (abstrakter) umsetzen zu können, indem sie eine oder mehrere Klassen deklarieren und definieren, welche in einem vorgegebenen Programm verwendet werden.<br>• mittels der wesentlichen Konzepte der objektorientierten Programmierung Programme entwerfen, indem sie diese Konzepte bei der Implementierung einer Aufgabenstellung verwenden.<br>• die verschiedenen Arten des Polymorphismus differenzieren, um mit dem jeweils passenden Code von konkreten Typen zu abstrahieren.<br>• ein vorgegebenes Programm mit den passenden Arten polymorpher C++-Sprachmittel ausstatten und bezüglich der Lesbarkeit und Wartbarkeit verbessern.<br>• wesentliche Teile der C++-Standardbibliothek anwenden; hierbei können sie für kleine Problemstellungen die richtigen Bibliotheksteile benennen.<br>• ausgewählte Bibliotheksteilen bei der Implementierung einer Aufgabenstellung benutzen und einschätzen, welche die dazu besser geeigneten sind.<br>• (einige wenige) Idiome und Patterns auf eigenen Code adaptieren. |

| | |
|---|---|
| | • eine gegebene Problemstellung analysieren, in kleinere Teile zerlegen und mit den jeweils angemessenen C/C++-Sprachmitteln implementieren. |
| Prüfungsvorleistung | keine |
| Medien-/ Lernform | Multimedial aufbereitetes Online-Studienmodul zum Selbststudium mit zeitlich parallel laufender Online-Betreuung (E-Mail, Foren, Chat, Webkonferenzen, Einsendeaufgaben u. a.) sowie Online-Programmierübungen |
| Arbeitsaufwand | Selbststudium: ca. 138 h<br>Webkonferenzteilnahme: ca. 10 h<br>Prüfung: 120 Minuten |
| Präsenzart | In Online-Konferenz möglich |
| Prüfungsform | Klausur (120 min.) |
| Voraussetzung für die Vergabe von Leistungspunkten | Studienleistung (Programmierübung): Erfolgreiche Bearbeitung von 3 Einsendeaufgaben (1 CP).<br>Bewertet mit "Bestanden"<br><br>Prüfungsleistung (4 CP): Bestehen der Prüfung (Klausur oder mündliche Prüfung) |
| Literatur | Stroustrup, Bjarne (2014): The C++ programming language. [C++ 11]. 4. ed., 2. print. Upper Saddle River, NJ: Addison-Wesley.<br>Kirch, Ulla; Prinz, Peter (2013): C++ - das Übungsbuch. 4., überarb. Aufl. Heidelberg, München, Landsberg, Frechen, Hamburg: mitp. |
| weitere Hinweise | Dieses Modul wird auf Deutsch angeboten |

| Studieninhalte |
|---|
| **Berechnungen auf wichtigen Datentypen**<br>Elementare Datentypen, char-ASCII-Glyph, I/O, Bitschieberei; call stack, Rekursion; free store<br>**Nichtlinearer Programmablauf**<br>Reentranz; event-based programming vs. thread-based; interrupt service routines; locking, lock-free data structures<br>**Benutzerdefinierte Datentypen**<br>Komposition neuer benutzerdefinierter Datentypen, um Ausdrücksmächtigkeit zu erhöhen; operator overloading; einfache Klasse ohne Vererbung; Typumwandlung (implizit/explizit)<br>**Objektorientierte Programmierung**<br>Information hiding; subtyping; Interface vs. Implementation; aggregation vs. composition; Identität vs. Äquivalenz; Delphi-style OO (i.e. only free store objects, no assignment operator, only explicit ctors )<br>**Polymorphismus**<br>Subtyping polymorphism mit virtuellen Methoden; Einordnung des bekannten Polymorhismustypen ad-hoc polymorphism; Generic Polymprhism (templates) |

**C++-Standardbibliothek**
Std Container; Std Algorithmen; moderner C++-Progammierstil
**Idiome und Muster**
Constructional: virtual ctor, factory, …; Resource Acquisition Is Initialization (RAII); Design-by-contract (DBC)
**Eleganz und diesbezügliche Hindernisse**
Klassendesign; Schwierigkeiten bei der Kombination von Sprach-Features; Objekte und Pointer (copy ctor, operator=, ...); Vererbung und operator overloading; Vererbung und container