Clicker

Bitte den Link

https://vc2.sonia.de/b/har-zq1-o0p-dhs

für WS 2024/25 nutzen.

Die verschiedenen Programmierparadigmen von C++

Termin				Vorlesung	Übungen und Feedback
Vorles	Woche			Freitag ; Block 1+2	
9	10	29. Nov	Fr	STL, Iteratoren; Algorithmus versus Methode	Callsign Extraction simple; Berechnung Raten, correction; Zwischen-Review 29.11
10	11	06. Dez	Fr	lineare und assoziative Container;	Unterstützung/Vorabnahme der Übungen; z.B 11.12 für Abgabe am 18.12 (Zusatztermin in diesem Zeitraum)
11	12	13. Dez	Fr	Verschiebeoperatoren Klasse unique_ptr,, shared_ptr, Lambda-Ausdrücke	finale Abgabe der Übungen; Di 17.12
12	13	20. Dez	Fr	Vorbereitung Klausur	

STL, Iteratoren; Algorithmus versus Methode

lineare und assoziative Container;

Verschiebeoperatoren Klasse unique_ptr,, shared_ptr, Lambda-Ausdrücke

Vorbereitung Klausur

Klausur: Do, 16.1.25, 14-15:30 Uhr
Klausureinsicht: Fr. 24.1.2025 10:00- Uhr

Rückblick auf den 8.11

- Umfangreiche Wiederholung mit Studenten-Interaktion zur Objekterzeugung und –zerstörung
- Minimale Standard-Schnittstelle am Beispiel der Klasse Auto
- Wie kann man die Übungsaufgabe 04 angehen und in Einzelteile unterteilen (Gruppen/Teams?)
- Templates
- Polymorphie / dynamisches Binden
- 15.11 fiel aus
- 22.11 DLR-Besuch

Reihenfolge der Operator-Auswertung

```
int main() {
   fstream datei("datei.txt", ios::out);
   X x(11);   Y y(222);
   datei << x << y << endl;
}
```

Wird erst operator<< von X oder erst von Y aufgerufen?
Wie kann die Hypothese prüfen?

Reihenfolge der Operator-Auswertung

```
class  X {
public:
   X(int a) : si(a) {}
private:
    int si;
    friend ostream& operator<<
       (ostream&, const X&);
};
```

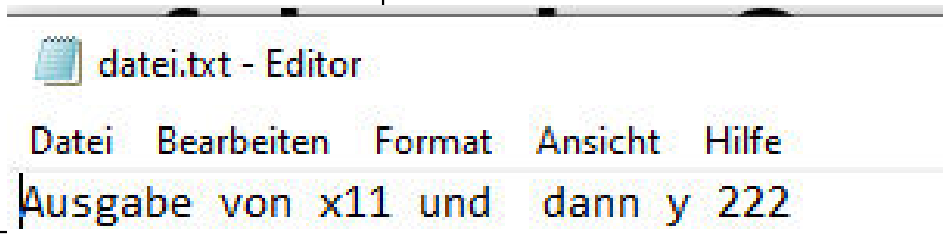
```
class  Y {
public:
   Y(int a) : si(a) {}
private:
   int si;
   friend ostream& operator<<
        (ostream&, const Y&);
};
```

```
ostream& operator<<(ostream&
       ostr, const X& x) {
   cout << " Aufruf von X " <<
       x.si << "; ";
   ostr << x.si;
   return ostr;
}
```

```
ostream& operator<<(ostream&
       ostr, const Y& y) {
   cout << " y wird aufgerufen
       " << y.si << "; ";
   ostr << y.si;
   return ostr;
}
```

Reihenfolge der Operator-Auswertung (2)

```
int main() {
   fstream datei("datei.txt", ios::out);
   X x(11);   Y y(222);
   datei << "Ausgabe von x" << x << " und "
      " dann y " << y << endl;
   cout << endl;
}
```




```
Aufruf von X 11;  y wird aufgerufen 222;
```

Funktionsargumente werden von rechts nach links ausgewertet

```
int main() {
    funkArg(f(3), g(2));
}
void funkArg(int fv, int gv) {
    cout << "\nfunkArg ";
    cout << " fv + gv "
     << fv + gv << endl;
}
```

```
int f(int a) {
   cout << " f(" << a << ") ";
   return a * a;
}
int g(int a) {
    cout << " g(" << a << ") ";
    return a * a * a;
}
```

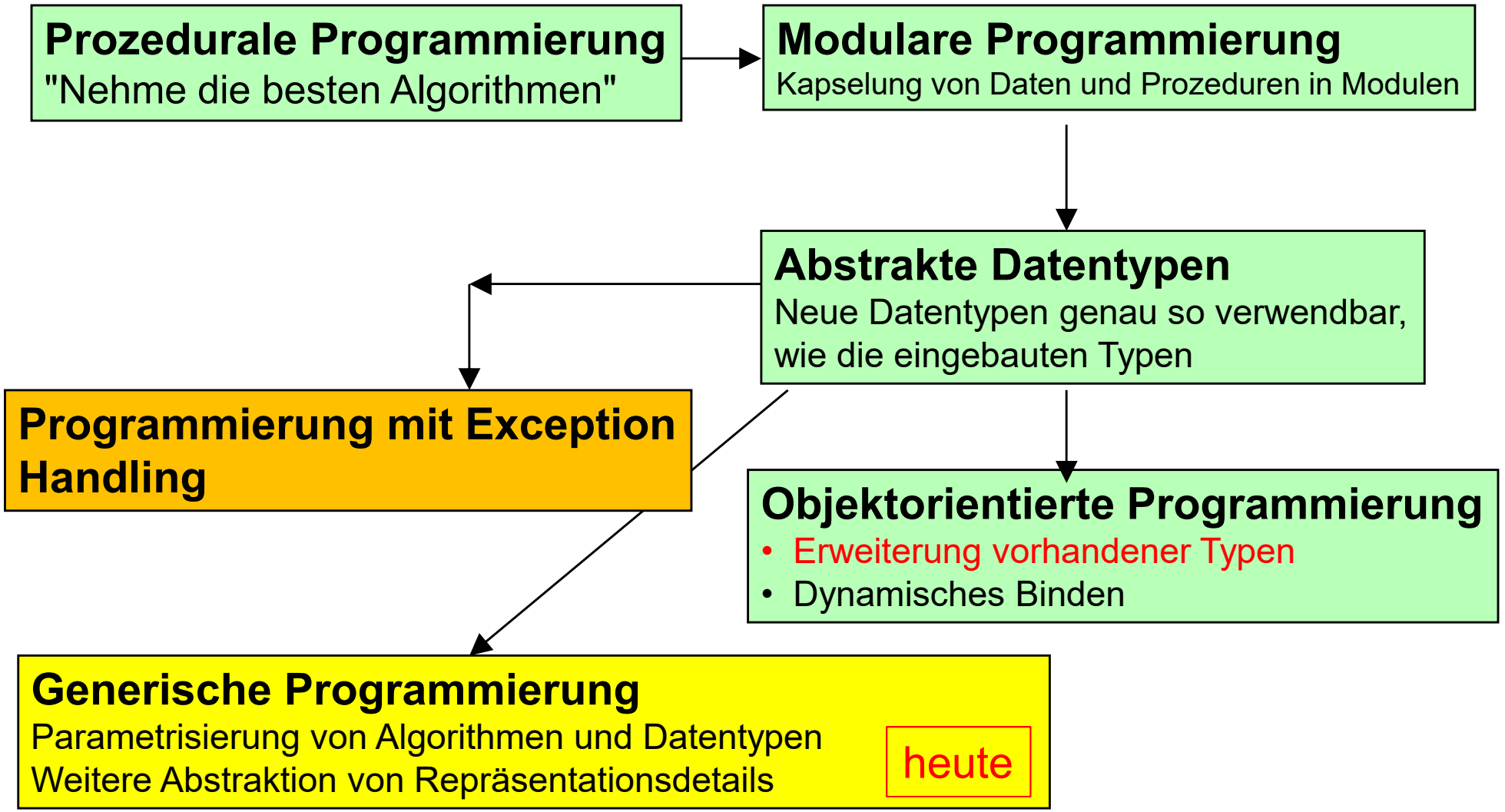
```
 g(2)  f(3)
funkArg  fv + gv 17
```

Und wie sieht die Aufruf-Reihenfolge nun aus?

```
int main() {
    funkArg(f(3), g(2));
}
void funkArg(int fv, int gv) {
    cout << "\nfunkArg ";
    cout << " fv + gv "
     << fv + gv << endl;
}
```

```
int f(int a) {
   cout << " f(" << a << ") ";
   return a * a;
}
int g(int a) {
    cout << " g(" << a << ") ";
    return a * a * a;
}
```

Programmierparadigmen von C++



Ziele der Veranstaltung (Folie aus Vorlesung 1)

- Testgetriebene Software-Entwicklung (Test first)
- „Erst denken, dann hacken“ (zunächst (mit Worten oder formal) beschreiben)
- Die vier Gesichter von C++:
  - Prozedurale Entwicklung in C++ (Zeigermodell, Werte- und Referenzsemantik)
  - Objektorientierte Software-Entwicklung (Vererbung, Polymorphie)
  - Generische Software-Entwicklung (Templates)
  - Programmierung mit der Standard-Template-Library
- Einstieg in die Programmierung im Großen (Bibliotheken, SW-Entwicklung im Team)

Wie wird man ein guter Software-Ingenieur? Mein Anspruch

Kompetenzen können durch Wissen unterfüttert, aber nicht durch Ausbildung produziert werden können.
Vorbilder können hier helfen.
Codierung ist ein wichtiger Bestandteil der Ausbildung, aber steht nicht im Vordergrund. Jeder Informatiker sollte aber das Handwerk beherrschen, da es die Basis seines Berufs bildet.

Sie lernen einfache, gut lesbare Programme zu erstellen, deren Korrektheit nicht dem Zufall überlassen wird.

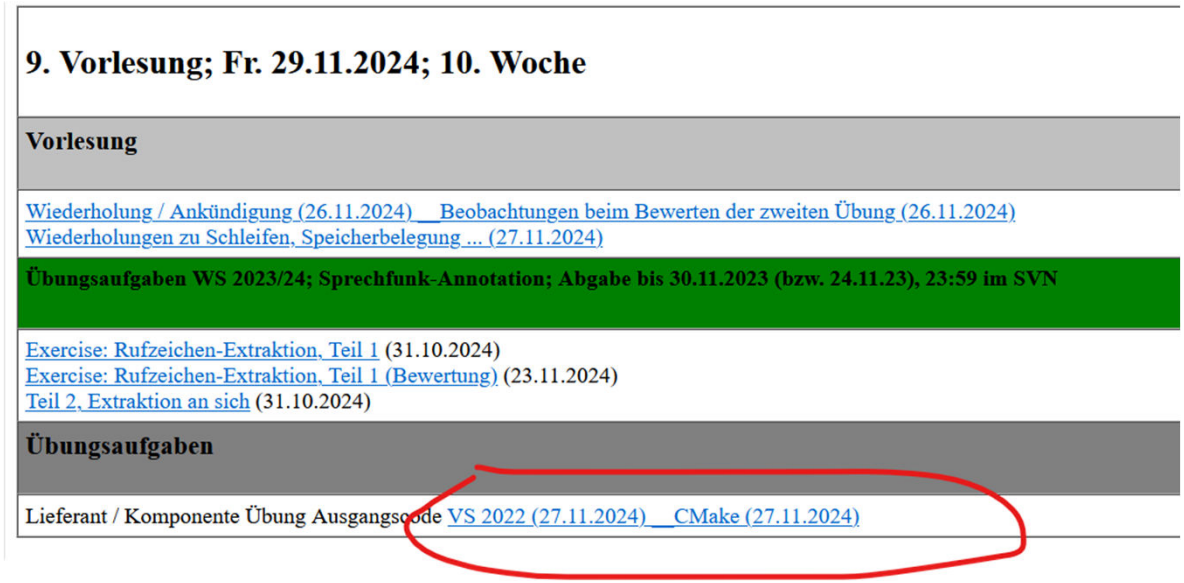
Hauptsache „es läuft“ wird schief gehen.
Standards, Stil und Form der Programmierung sind Gegenstand der Lehre und der Kontrolle im Übungsbetrieb.

Probleme der Praxis

- Erhebung von Anforderungen, die alles andere als klar und vorgegeben sind.
- Koordination vieler Projektmitarbeiter.
- Konflikte zwischen Termin-, Qualitäts- und Funktionalitätszielen
- Arbeit mit neuen keinesfalls fehlerfreien Werkzeugen
- Abstimmung mit schwierigen Partnern (mit mir und ggf. mit Kommilitonen)

Vorlesungsplanung heute, 29.11.2024

- Was so bei Aufgabe 2 und 3 auffiel (Datei „Uebung2.pptx“)
- Systematischer Einstieg in die Programmierung mit Templates (Iterator-Konzept)



9. Vorlesung; Fr. 29.11.2024; 10. Woche

Vorlesung

Wiederholung / Ankündigung (26.11.2024) __Beobachtungen beim Bewerten der zweiten Übung (26.11.2024)
Wiederholungen zu Schleifen, Speicherbelegung ... (27.11.2024)

Übungsaufgaben WS 2023/24; Sprechfunk-Annotation; Abgabe bis 30.11.2023 (bzw. 24.11.23), 23:59 im SVN

Exercise: Rufzeichen-Extraktion, Teil 1 (31.10.2024)
Exercise: Rufzeichen-Extraktion, Teil 1 (Bewertung) (23.11.2024)
Teil 2, Extraktion an sich (31.10.2024)

Übungsaufgaben

Lieferant / Komponente Übung Ausgangscode VS 2022 (27.11.2024) __CMake (27.11.2024)

- Stand bei den Übungen (Zeigen Sie mir (und den anderen) mal, was Sie haben).
- Etwas Wiederholung aus einem Test aus vorherigen Semestern zu Speicherbelegung malen und +Katze/-Katze (Datei Test2-Teil1bis2.pptx)

Bewertungsbogen ausfüllen