Clicker für HHe

Einloggen vom Handy auf https://vc2.sonia.de/b/har-zq1-o0p-dhs
für WS 2024/25.
Konferenz dort ohne Audio starten.
Auch von Laptop mit anderen Namen dort einloggen und testen, ob man antworten kann
Nun Umfrage erstellen, indem auf Plus geklickt wird.

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	
7	7	08. Nov Fr	Operatoren (Teil 2); Templates	Was ist bei Aufgabe 4 zu tun und wie?
	8	15. Nov Fr	fällt aus	tiefe, flache Kopie, minimale Std-Schnittstelle; dyn Array Abgabe bis Fr 15.11
8	9	22. Nov Fr	DLR-Besuch Treffen am Tor bis 08:55	Unterstützung/Vorabnahme der Übungen zusätzlicher Termin in diesemZeitraum; für Zwischenfeedback finale Abgabe 17.12
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 (Polymorphie)	finale Abgabe der Übungen; Di 17.12
12	13	20. Dez Fr	Vorbereitung Klausur	

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

Rückblick

- Wiederholung mit Referenz oder Zeiger oder Werte oder doch was anderes?
- FunktionLog, LogTrace
- MacheNix als Übung für Sie (Kopierkonstruktor und Zuweisungsoperator, selber „Hand anlegen“ für DynVorgangsArray)
- Operatoren
- Übung Plus, bei der Vektor-Klasse bzw. bei DynVorgangsArray

Clicker-“Abstimmung“

```
ofstream datei("ausgabe.txt", ios::out);
class Hund
{
public:
   Hund(string f) {farbe=f;
                   datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
private:
  string farbe;
};
```

```
void func1() {
   Hund bello("g ");
   datei << " Ende ";
}
```

Welche Dateiausgabe?

1. +H g Ende
2. Ende
3. Clicker ist anstrengend
4. +H g Ende -H g

Clicker-“Abstimmung“

```
ofstream datei("ausgabe.txt", ios::out);
class Hund
{
public:
   Hund(string f) {farbe=f;
                   datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
private:
  string farbe;
};
```

```
void func1() {
   Hund bello("g ");
   datei << " Ende ";
}
```

Welche Dateiausgabe?

1. +H g Ende
2. Ende
3. Clicker ist anstrengend
4. +H g Ende -H g

Ergebnis:
1_ 2 __ 3 __ 4__ +H g Ende -H g

Clicker-“Abstimmung“

```
ofstream datei("ausgabe.txt", ios::out);
class Hund
{
public:
   Hund(string f) {farbe=f;
                   datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
private:
  string farbe;
};
```

```
void func2() {
   Hund bello("g ");
   Hund anton("s ");
   datei << " Ende ";
}
```

Welche Dateiausgabe?

1. +H g +H s Ende
2. +H g +H s Ende -H g -H s
3. +H g +H s Ende -H s -H g
4. +H g +H s -H g -H s Ende

Clicker-“Abstimmung“

```
ofstream datei("ausgabe.txt", ios::out);
class Hund
{
public:
   Hund(string f) {farbe=f;
                   datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
private:
  string farbe;
};
```

```
void func2() {
   Hund bello("g ");
   Hund anton("s ");
   datei << " Ende ";
}
```

Welche Dateiausgabe?

1. +H g +H s Ende
2. +H g +H s Ende -H g -H s
3. +H g +H s Ende -H s -H g
4. +H g +H s -H g -H s Ende

Ergebnis 1:

1__ 2 _ 3 _ +H g +H s Ende -H s -H g 4__

„Noch“ sehr komplexe Fragestellung

```
ofstream datei("ausgabe.txt", ios::out);
class Hund   {
public:
   Hund(string f="?") {farbe=f;
              datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
   string GetFarbe()const  {return farbe;}
private:
  string farbe;
};
```

```
void func4_hlp(Hund h) {
     datei << " hlp ";   }

void func4() {
   Hund bello("g");
   func4_hlp(bello);
   datei << " E ";  }
```

Welche Dateiausgabe?

1. +H g hlp -H g E -H g
2. +H g +H g hlp -H g E -H g
3. +H g hlp E -H g
4. +H g hlp E

„Noch“ sehr komplexe Fragestellung

```
ofstream datei("ausgabe.txt", ios::out);
class Hund   {
public:
   Hund(string f="?") {farbe=f;
              datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
   string GetFarbe()const  {return farbe;}
private:
  string farbe;
};
```

```
void func4_hlp(Hund h) {
      datei << " hlp ";   }

void func4() {
   Hund bello("g");
   func4_hlp(bello);
   datei << " E ";  }
```

Welche Dateiausgabe?

1. +H g hlp -H g E -H g
2. +H g +H g hlp -H g E -H g
3. +H g hlp E -H g
4. +H g hlp E

Ergebnis:

1_ +H g hlp -H g E -H g 2 _ 3 _ 4__

„Achtung“ Fragestellung

```
ofstream datei("ausgabe.txt", ios::out);
class Hund   {
public:
   Hund(string f="?") {farbe=f;
              datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
   string GetFarbe()const  {return farbe;}
private:
  string farbe;
};
```

```
void func5() {
   Hund bello();
   Hund wau;
   datei << " Ende ";  }
```

Welche Dateiausgabe?

1. +H ? +H ? Ende –H? –H?
2. +H ? Ende –H?
3. Syntaxfehler
4. +H ? +H ? –H? Ende –H?

„Achtung“ Fragestellung

```
double sin(double);
```

Eine Funktion, die „sin“ heißt, ein double akzeptiert und ein double zurückliefert!

```
Hund fritz(double);
```

Eine Funktion, die „fritz“ heißt, ein double akzeptiert und einen Hund zurückliefert!

```
void func5() {
   Hund bello();
   Hund wau;
   datei << " Ende ";  }
```

Eine Funktion, die „**bello**“ heißt, kein Argument besitzt und einen Hund zurückliefert!

Die gleiche Frage noch einmal

```
ofstream datei("ausgabe.txt", ios::out);
class Hund   {
public:
   Hund(string f="?") {farbe=f;
              datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
   string GetFarbe()const  {return farbe;}
private:
  string farbe;
};
```

```
void func5() {
   Hund bello();
   Hund wau;
   datei << " Ende ";  }
```

Welche Dateiausgabe?

1. +H ? +H ? Ende –H? –H?
2. +H ? Ende –H?
3. Syntaxfehler
4. +H ? +H ? –H? Ende –H?

„Achtung“ Fragestellung

```
ofstream datei("ausgabe.txt", ios::out);
class Hund   {
public:
   Hund(string f="?") {farbe=f;
              datei << "+H " << farbe;};
   ~Hund() {datei << " -H " << farbe;};
   string GetFarbe()const  {return farbe;}
private:
  string farbe;
};
```

```
void func5() {
   Hund bello();
   Hund wau;
   datei << " Ende ";  }
```

Welche Dateiausgabe?

1. +H ? +H ? Ende –H? –H?
2. +H ? Ende –H?
3. Syntaxfehler
4. +H ? +H ? –H? Ende –H?

Ergebnis:

1_ 2 +H ? Ende –H? 3 _ 4__

Clicker: Schnittstelle

```
class Baum  {
public:
   Baum(Baum* eltern, int blaetter);
   int GetAnzahlDerBlaetter() const;
   const Baum* GetVater();
   Baum* GetKind();
   Baum* elternTeil;
private:
   void SetAnzahlDerBlaetter(int b);
   int anzahlDerBlaetter;
   Baum* kindTeil;
   Baum* GetKindBaum();
};
void baumTest() {
   Baum b(nullptr, 1008);
   b.elternTeil = nullptr; }
```

Die Codezeile: **b.elternTeil = nullptr;**

1. Ist syntaktisch korrekt und guter Programmierstil
2. Ist syntaktisch falsch
3. Attribute gehören nicht in die Klassen-Schnittstelle.
4. Eine Klasse sollte nicht „Baum“ heißen.

Schnittstelle

```
class Baum  {
public:
   Baum(Baum* eltern, int blaetter);
   int GetAnzahlDerBlaetter() const;
   const Baum* GetVater();
   Baum* GetKind();
   Baum* elternTeil;
private:
   void SetAnzahlDerBlaetter(int b);
   int anzahlDerBlaetter;
   Baum* kindTeil;
   Baum* GetKindBaum();
};
```

```
void baumTest() {
   Baum b(NULL, 1008);
   b.elternTeil = NULL;  // korrekt
```

Das ist aber keine Programmierung mit abstrakten Datentypen.

Attribute sollten nicht direkt zur Schnittstellen gehören.

Clicker: Schnittstelle

```
class Baum  {
public:
   Baum(Baum* eltern, int blaetter);
   int GetAnzahlDerBlaetter() const;
   const Baum* GetVater();
   Baum* GetKind();
   Baum* elternTeil;
private:
   void SetAnzahlDerBlaetter(int b);
   int anzahlDerBlaetter;
   Baum* kindTeil;
   Baum* GetKindBaum();
};
void baumTest() {
   Baum b(nullptr, 1008);
   b.elternTeil = nullptr; }
```

Die Codezeile: **b.elternTeil = nullptr;**

1. Ist syntaktisch korrekt und guter Programmierstil
2. Ist syntaktisch falsch
3. Attribute gehören nicht in die Klassen-Schnittstelle.
4. Eine Klasse sollte nicht „Baum“ heißen.

Die Codezeile: **b.elternTeil = NULL;**

1. Ist syntaktisch korrekt und guter Programmierstil
2. Ist syntaktisch falsch
3. **Attribute gehören nicht in die Klassen-Schnittstelle.**
4. Eine Klasse sollte nicht „Baum“ heißen.

Minimale Standardschnittstelle

Folgende Elemente sind für viele Klassen eine notwendige Grundausstattung:

- Standardkonstruktor: **X()**
- Kopierkonstruktor: **X(const X& x)**
- Destruktor: **~X()**
- Zuweisungsoperator: **X& operator= (const X& x)**

Die Meinungen zum Thema "Minimale Standardschnittstelle" sind allerdings uneinheitlich.

Empfehlung: Beim Entwurf einer Klasse für jedes dieser vier Elemente prüfen, ob es definiert sein soll. Wenn ja, entscheiden, ob das jeweils vom System erzeugte Default-Element in Ordnung ist, oder ob das Element explizit selbst definiert werden soll. Wenn nein, dann sollte das Element explizit unterbunden werden, siehe dazu die folgende Erläuterung.

Clicker-“Abstimmung“

```
class Auto {
   public:
      Auto(string, float, float, int);
      void SetHubraum(int hub);
      int WieWeitKommeIch();
   private:
      string hersteller;
      float verbrauch;
      float tankinhalt;
      int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Nein, Destruktor fehlt
2. Nein, Kopierkonstruktor fehlt
3. Nein, Standardkonstruktor fehlt
4. Nein, Zuweisungs-Operator fehlt

Clicker-“Abstimmung“

```
class Auto {
   public:
      Auto(string, float, float, int);
      void SetHubraum(int hub);
      int WieWeitKommeIch();
   private:
      string hersteller;
      float verbrauch;
      float tankinhalt;
      int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Nein, Destruktor fehlt
2. Nein, Kopierkonstruktor fehlt
3. Nein, Standardkonstruktor fehlt
4. Nein, Zuweisungs-Operator fehlt

Alles fehlt → jede Antwort korrekt

Clicker-“Abstimmung“

```
class Auto {
   public:
      Auto(int hubraum=14);
     ~Auto();
private:
   int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Nein, Destruktor fehlt
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

Clicker-“Abstimmung“

```
class Auto {
   public:
      Auto(int hubraum=14);
     ~Auto();
private:
   int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Nein, Destruktor fehlt
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

1. Nein, Destruktor fehlt
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

Clicker-“Abstimmung“

```
class Auto {
   public:
      Auto(int hubraum=14);
     ~Auto();
      Auto(const Auto& auto);
      bool operator==(const Auto& a);
private:
   int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Ja
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

Clicker-“Abstimmung“

```
class Auto {
   public:
      Auto(int hubraum=14);
     ~Auto();
      Auto(const Auto& auto);
      bool operator==(const Auto& a);
private:
   int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Ja
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

1. Ja
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

Clicker-“Abstimmung“

```
class Auto {
   public:
 Auto(int hubraum=14);
~Auto();
 Auto(const Auto& auto);
 Auto& operator=(const Auto& auto);
private:
   int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Ja
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

Clicker-“Abstimmung“

```
class Auto {
   public:
 Auto(int hubraum=14);
~Auto();
 Auto(const Auto& auto);
 Auto& operator=(const Auto& auto);
private:
   int hubraum;
};
```

Hat Auto eine minimale Standardschnittstelle?

1. Ja
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

1. Ja
2. Nein, Kopierkonstruktor fehlt
3. Nein, Zuweisungs-Operator fehlt
4. Nein, Standardkonstruktor fehlt

Vorlesungsplanung

Umfangreiche Wiederholung mit Studenten-Interaktion
  zur Objekterzeugung und –zerstörung
Minimale Standard-Schnittstelle am Beispiel der Klasse Auto

Operatoren am Beispiel der Klasse von Vektor zum Selbermachen

Wie kann man die Übungsaufgabe 04 angehen und in Einzelteile unterteilen (Gruppen/Teams?)

Beginn Templates

Code für Übung heute

7. Vorlesung; Fr. 08.11.2024
Vorlesung
Wiederholung / Ankündigung (30.10.2023) Operatoren (20.10.2023) Sonstiges zu Klassen (zum Selbststudium) (20.10.2023)
Übungsaufgaben WS 2023/24; Sprechfunk-Annotation
siehe vorherige Woche; mit Abgabe bis Sonntag, den 5.11.2023 23:59 Uhr
Übungsaufgaben
Klasse Vektor mit Operatoren Ausgangscode (26.10.2024) VS2022 Aufgabe Klasse Matrix mit Operatoren Ausgangscode (26.10.2024) VS2022

Bitte schon mal herunterladen