Clicker

Bitte den Link

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

für WS 2024/25 nutzen.

	Termin			Vorlesung	Übungen und Feedback
Vorles	Woche			Freitag ; Block 1+2	
5	5	25. Okt	Fr	Werte-, Zeigerparameter etc., Klassen string, vector<T> ADT; Klassen, ti	main; keine Bewertung, aber zusammen mit
6	7	06. Nov	Mi	tiefe und flache	
7	7	08. Nov	Fr	Operatoren; Templates	
	8	15. Nov	Fr	fällt sehr, sehr wahrscheinlich aus	tiefe, flache Kopie, minimale Std-Schnittstelle; dyn Array Abgabe bis Fr 15.11
8	9	22. Nov	Fr	DLR-Besuch	Unterstützung/Vorabnahme der Übungen
9	10	29. Nov	Fr	STL, Iteratoren; Algorithmus versus	
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 18.12
12	13	20. Dez	Fr	Vorbereitung Klausur	

Freitag, 15.11 ist Abgabe der 2. und 3. Aufgabe,
Unterstützung per BBB gewünscht?

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

Unterstützung per BBB? (Maximal 20 Minuten)

- Do. 24.10; 15-18 Uhr
- Di 29.10 15-16 Uhr
- MI 30.10 15-18 Uhr
- Di 5.11 15-18 Uhr
- Do 7.11 15-18 Uhr

- Spätere Termine nach dem 7.11 natürlich auch

Rückblick

Nächste Übungsaufgabe zur Abgabe vorstellen

Aufgabe Mit30_40_Belegen (60 Minuten als Einzelübung)
- Zeiger auf Zeiger int** ptr
- Zeiger als Referenzen f(int* & apr_zeiger) { }

Andere nützliche Dinge über Zeiger
- const_cast, reinterpret_cast
- const int* const ptr; etc.

Clicker-“Abstimmung“

```
int  j  =  61;  /* 1 */
const  int  ka  =  36;  /* 2 */
if  (ka  =  144)  /* 3 */
      const double PI = 2.0 * j;  /* 4 */
```

Welche Codezeile ist SYNTAKTISCH falsch?

1. 1
2. 2
3. 3
4. 4

Clicker-“Abstimmung“

```
int  j  =  61;  /* 1 */
const  int  ka  =  36;  /* 2 */
if  (ka  =  144)  /* 3 */
      const double PI = 2.0 * j;  /* 4 */
```

Welche Codezeile ist SYNTAKTISCH falsch?

1. 1
2. 2
3. 3
4. 4

Ergebnis:
__ 1,2, 3, 4; ka ist eine Konstante

Clicker-“Abstimmung“

```
void tester1(int& p1) {
    p1 = 42;
}
void aufgFunk1() {
    int j = 122;
    tester1(j);
    datei << "j= " << j;
}
```

Was wird in „datei“ ausgegeben?

1. j= 42
2. j= 122
3. 42
4. 122

Clicker-“Abstimmung“

```
void tester1(int& p1) {
    p1 = 42;
}
void aufgFunk1() {
    int j = 122;
    tester1(j);
    datei << "j= " << j;
}
```

Was wird in „datei“ ausgegeben?

1. j= 42
2. j= 122
3. 42
4. 122

Ergebnis:
__ j = 42

Clicker-“Abstimmung“

```
void tester3(int fld[]) {
    fld[2] = 126;
}
void aufgFunk3() {
    int arr[] = { 11, 2, 3 };
    tester3(arr);
    datei << "[2] " << arr[2];
}
```

Was wird in „datei“ ausgegeben?

1. [2] 11
2. [2] 2
3. [2] 3
4. [2] 126

Clicker-“Abstimmung“

```
void tester3(int fld[]) {
    fld[2] = 126;
}
void aufgFunk3() {
    int arr[] = { 11, 2, 3 };
    tester3(arr);
    datei << "[2] " << arr[2];
}
```

Was wird in „datei“ ausgegeben?

1. [2] 11
2. [2] 2
3. [2] 3
4. [2] 126

Ergebnis:
__ [2] 126

Clicker-“Abstimmung“

```
void mapTest() {
   map<string, double> gewicht;
   gewicht["Torr"] = 16.39;
   gewicht["Torr"] = 81.60;
   datei << gewicht["Torr"]
      << " " << gewicht["Ida"];
}
```

Was wird in „datei“ ausgegeben (Blanks nicht beachten)?

1. Absturz
2. 81.60 Absturz
3. 81.6 0
4. 81.60 16.39

Clicker-“Abstimmung“

```
void mapTest() {
   map<string, double> gewicht;
   gewicht["Torr"] = 16.39;
   gewicht["Torr"] = 81.60;
   datei << gewicht["Torr"]
      << " " << gewicht["Ida"];
}
```

Was wird in „datei“ ausgegeben (Blanks nicht beachten)?

1. Absturz
2. 81.60 Absturz
3. 81.6 0
4. 81.60 16.39

Ergebnis:
81.6 0

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

996	anzahl	144
1000	zahl[0]	21
1004	zahl[1]	31
1008	zahl[2]	41
1012	zahl[3]	17
1020	zahl[4]	22
1024	zahl[5]	19
1028	p	???

Was steht in den Adressen ab 1028 nach Ausführung des Blauen?

1. 1008
2. 1012,
3. 41,
4. 17

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

.

996	anzahl	144
1000	zahl[0]	21
1004	zahl[1]	31
1008	zahl[2]	41
1012	zahl[3]	17
1020	zahl[4]	22
1024	zahl[5]	19
1028	p	1008

.

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

.

996	anzahl	144
1000	zahl[0]	21
1004	zahl[1]	31
1008	zahl[2]	41
1012	zahl[3]	17
1020	zahl[4]	22
1024	zahl[5]	19
1028	p	1008

Was steht in den Adressen ab 1008-1011 und 1028-1031 nach Ausführung des Blauen?

1. 41 / 1008
2. 42 / 1008
3. 41 / 1009
4. 41 / 1012

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

.

996	anzahl	144
1000	zahl[0]	21
1004	zahl[1]	31
1008	zahl[2]	42
1012	zahl[3]	17
1020	zahl[4]	22
1024	zahl[5]	19
1028	p	1008

.

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

.

996	anzahl	144
1000	zahl[0]	21
1004	zahl[1]	31
1008	zahl[2]	42
1012	zahl[3]	17
1020	zahl[4]	22
1024	zahl[5]	19
1028	p	1008

Was steht in den Adressen ab 1008-1011 und 1028-1031 nach Ausführung des Blauen?

1. 43 / 1008
2. 42 / 1008
3. 42 / 1009
4. 42 / 1012

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

996	anzahl	144
1000	zahl[0]	21
1004	zahl[1]	31
1008	zahl[2]	42
1012	zahl[3]	17
1020	zahl[4]	22
1024	zahl[5]	19
1028	p	1012

Was steht in den Adressen ab 1008 und 1028 nach Ausführung des Blauen?

1. 41 / 1008
2. 42 / 1008
3. 41 / 1009
4. 41 / 1012

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

Was wird in „datei“ ausgegeben?

1. 31, 41, 17,
2. 41, 42, 17,
3. 41, 17, 18,
4. 41, 17, 22,

Clicker-“Abstimmung“

```
void zeigerTest () {
   int anzahl = 144;
   int zahl[] = { 21, 31, 41, 17, 22, 19 };
   int* p = &(zahl[2]);
   datei << *p << ",";
   (*p)++;
   datei << *p << ",";
   ++p;
   datei << *p << ",";
}
```

Was wird in „datei“ ausgegeben?

1. 31, 41, 17,
2. 41, 42, 17,
3. 41, 17, 18,
4. 41, 17, 22,

Ergebnis:
41, 42, 17,

Clicker-“Abstimmung“

Welche Aussage trifft zu?

1. Header-Dateien haben üblicherweise die Endung „.h“?
2. Quellcode-Dateien haben üblicherweise die Endung „.h“?

Clicker-“Abstimmung“

Welche Aussage trifft zu?

1. Header-Dateien haben üblicherweise die Endung „.h“?
2. Quellcode-Dateien haben üblicherweise die Endung „.h“?

_-_ Header __Source

Clicker-“Abstimmung“

Welche Aussage trifft zu?

1. Funktionen werden in einer Header-Datei deklariert (Schnittstelle angegeben)
2. Funktionen werden in der Quellcode-Datei deklariert

Clicker-“Abstimmung“

Welche Aussage trifft zu?

1. Funktionen werden in einer Header-Datei deklariert (Schnittstelle angegeben)
2. Funktionen werden in der Quellcode-Datei deklariert

_-_ Header __Source

Clicker-“Abstimmung“

Welche Aussage trifft zu?

1. Größere Funktionen werden in der Header-Datei implementiert /definiert.
2. Größere Funktionen werden in der Quellcode-Datei implementiert /definiert.

Clicker-“Abstimmung“

Welche Aussage trifft zu?

1. Größere Funktionen werden in der Header-Datei implementiert /definiert.
2. Größere Funktionen werden in der Quellcode-Datei implementiert /definiert.

__ Header _-_ Source

Clicker-“Abstimmung“

Vorgang.h

```
#ifndef VORGANG_HEADER
#define VORGANG_HEADER

struct Vorgang {
   double dauer;
   double fruehanf;
   double spaetend;
};
void initVorgang(Vorgang* v);
void ausgabeVorgang(const Vorgang* v);
bool istDurchfuehrbar(Vorgang *v);
void passeFruehanfAn(
        Vorgang* v, const Vorgang* vv);
void passeSpaetEndAn(
        Vorgang* v, const Vorgang* vn);

#endif /* VORGANG_HEADER */
```

Wie nennt man die roten Codeteile?

1. Dateikopf
2. Include-Wächter
3. Datei-Deklaration
4. Funktionsdefinition

Clicker-“Abstimmung“

Vorgang.h

```
#ifndef VORGANG_HEADER
#define VORGANG_HEADER

struct Vorgang {
   double dauer;
   double fruehanf;
   double spaetend;
};
void initVorgang(Vorgang* v);
void ausgabeVorgang(const Vorgang* v);
bool istDurchfuehrbar(Vorgang *v);
void passeFruehanfAn(
        Vorgang* v, const Vorgang* vv);
void passeSpaetEndAn(
        Vorgang* v, const Vorgang* vn);

#endif /* VORGANG_HEADER */
```

Wie nennt man die roten Codeteile?

1. Dateikopf
2. Include-Wächter
3. Datei-Deklaration
4. Funktionsdefinition

__ Kopf _-_ Wächter
__ Deklaration __ Funk-Def

Clicker-“Abstimmung“

Welche Aufgabe hat ein Include-Wächter?

1. Kurzbeschreibung des Datei-Inhalts
2. Mehrfaches Einbinden / Includen der gleichen Datei in der gleichen Übersetzungseinheit (Source-Datei *.cxx) verhindern
3. Erleichterung der Funktionstests
4. Hinweise auf externe Symbole für den Linker

Clicker-“Abstimmung“

Welche Aufgabe hat ein Include-Wächter?

1. Kurzbeschreibung des Datei-Inhalts
2. Mehrfaches Einbinden / Includen der gleichen Datei in der gleichen Übersetzungseinheit (Source-Datei *.cxx) verhindern
3. Erleichterung der Funktionstests
4. Hinweise auf externe Symbole für den Linker

__ Beschreibung _-_Nur einmal includen
__ Testen __Linker-Symbole

Vorlesungsplanung

Wiederholung, insbesondere Aufgaben aus dem einem Test für vorherige Semester

Klasse string
Klasse vector<T>, Klasse array<T,size>

Referenz oder Zeiger oder Werte oder doch was anderes?

Klassen beginnen

- Am Beispiel von Vektor, Konstruktor, Destruktor
- Konstruktoren
- +Z, -Z
- LogTrace / FunctionLog