Seminar 1

1. Data Types

Die fünf wichtigsten Typen sind:

```
bool        done          { false    }; // Boolesche Werte (false, true)
int         answer        { 42       }; // ganze Zahlen
double      avg_grade     { 4.2      }; // Gleitkommazahlen
char        decimal_point { '.'      }; // (einzelne) Zeichen
std::string nick          { "Kant"   }; // Zeichenketten
```

Typen | Namen | Literale

Zuweisung mit Operator =

```
01 int main () {
02     int a { 2 };     // Initialisierung: a mit Anfangswert 2
03     int b { a };     // Initialisierung: b mit Anfangswert 2, aus a ermittelt
04     a = 4;           // Zuweisung: a erhaelt den Wert 4
05     b = 6 + a;       // Zuweisung: b erhaelt den Wert 10
06     a = a + 8;       // Zuweisung: a erhaelt den Wert 12
07     //c = 5;         // Fehler, c ist unbekannt
08     //…
09   }
```

2. Ein- und Ausgabe

```
01  /*
02     nickname.cpp
03     yymmdd-OSk
04     Liest einen Nickname ("Spitznamen") und das Geburtsjahr ein,
05     gibt die beiden eingelesenen Werte wieder aus.
06  */
07
08  #include <iostream>      // E/A-Stroeme aus der StdLib
09  using std::cin;          // using-Deklaration fuer cin aus std
10  using std::cout;         // using-Deklaration fuer cout aus std
11  #include <string>        // Strings aus der StdLib
12  using std::string;       // using-Deklaration fuer string aus std
13
14  int main( )
15  {
16    string nick { "capitalQ" }; // nick ist eine Variable vom Typ std::string
17    int    yob  { 1997 };       // yob (year of birth) ist eine Variable von Typ int
18
19    cout << "Bitte Nickname eingeben (gefolgt von \'Enter\'): ";
20    cin  >> nick; // lies die Zeichen in nick ein
21
22    cout << "Bitte Geburtsjahr eingeben (gefolgt von \'Enter\'): ";
23    cin  >> yob;  // lies die Zeichen in yob ein
24
25    std::clog << "\n\n\t" << "nick ist: " << nick
26              << "\n\tyob ist: " << yob << "\n\n";
27    return 0;
28  }
```

3. Strukturen

- neue Datentypen, die über die eingebauten Typen wie int, float hinausgehen
- mehrere Datenwerte zu einer Einheit
- die Elemente einer Struktur können unterschiedlichen Typ haben
- die einzelnen Elemente werden über Namen identifiziert
- man behandelt die Struktur dann als Ganzes
- bei Bedarf kann man immer noch auf die Komponenten zugreifen

```
#include <iostream>

struct date {
 int day;
 int month;
 int year;
};

int main() {
 date d;

 d.day = 12;
 d.month = 12;
 d.year = 2000;

 std::cout << d.day << "." << d.month <<
"." << d.year;
  return 0;
}
```

```
#include <iostream>
struct komplex {
 float re;
 float im;
};

komplex komp_add( komplex x, komplex y) {
 komplex ergebnis;
 ergebnis.re = x.re + y.re;
 ergebnis.im = x.im + y.im;
 return ergebnis;
}

int main() {
 komplex a = {1,2};
 komplex b = {2,2};

 komplex r = komp_add(a,b);
 std::cout << r.re << " " << r.im;

 return 0;
}
```

4. Arrays

- Arrays sind eine Zusammenfassung von Variablen gleichen Typs, wobei die einzelne Variable über eine Zahl, den Index, identifiziert wird
- die Länge des Arrays wird nicht mitgespeichert
- die Länge ist nur im Sichtbarkeitsbereich der Definition des Arrays bekannt
- sehr große Arrays sollen auf dem Heap (dynamisch) angelegt werden

4.1 Speicherorganisation

Der Speicher eines Programms ist in verschiedene Speicherbereiche untergliedert

- einen für den Code
- einen für globale und statische Variablen
- einen Heap
- einen Stack

- Variablen werden üblicherweise auf dem Heap oder dem Stack gespeichert
- Heap = Speicherbereich, in dem dynamisch angeforderte Variablen angelegt werden
- auf dem Stack erfolgt die Erzeugung auf zusammenhängenden Speicherblöcken. Wir nennen es “Stack Allocation”, da die Erzeugung im Funktionsaufrufstapel erfolgt.
- lokale Variablen werden auf dem Stack angelegt
- der Stack hat eine feste und manchmal kleine Größe

4.2 statische Erzeugung

- Länge ist konstant und zur Kompilation bekannt
- Array auf dem Stack angelegt

```
char text[100];
```

4.3 dynamische Erzeugung

- Array wird zur Laufzeit auf dem Heap angelegt

```
const int len = 100; // len kann, aber muss nicht konstant sein
char *text = new char[len];
char *const t = new char[len]; // t ist ein konstanter Zeiger!
delete[] text; // Speicherplatz des Arrays wird freigegeben
delete[] t;
```

- Funktion liegt ausserhalb des Sichtbarkeitsbereichs der Definition des Arrays
- Länge des Arrays ist nicht bekannt und sollte als Parameter mitgegeben werden
- sizeof kann nur die Anzahl Bytes des Zeigers ermitteln

```
void print(char *s) { … }
void print(char s[]) { … } // ist zu bevorzugen, weil Array ersichtlich
ist
```

- mehrdimensionale C-Arrays werden im Speicher als Kette von eindimensionalen Arrays abgespeichert

- die Länge der ersten Dimensionen ist nur im Sichtbarkeitsbereich der Definition des Arrays bekannt
- die Längen der weiteren Dimensionen gehen mit in den Typ ein

```
const int rows = 2, columns = 3;
int m[rows][columns];
int mm[rows][columns] = {{ 1,2,3}, {4,5,6}};
int mmm[][columns] = {1,2,3,4,5,6};
int v = mm[0][1]; // v=2
m[1][1] = v;
```

4.4 C Strings

- Zeichenketten werden als ein eindimensionales Character-Array behandelt
- Ende der gültigen Zeichenkette ist durch ein 0-Character gekennzeichnet
- wie bei allen Arrays in C/C++: die Länge wird nicht mitgespeichert
- vereinfachte Initialisierung erlaubt

```
char s[] = "Das ist ein Test."; // String-Schreibweise
```

- sizeof(s) gibt den Speicherbedarf der Kopie zurück

```
void foo(char s[]) { char s1[] = "ABC"; s[0] = 'a'; }
```

5. Übungen

1. Schreiben Sie ein Programm, das die Summe zweier eingegebener ganzzahliger Zahlen ausgibt.
2. Schreiben Sie ein Programm, das Sie nach Ihrem Vor- und Nachnamen fragt. Die Anwendung gibt eine Begrüßung aus. Die Begrüßung enthält den gesamten Namen sowie die Anzahl der Zeichen, die der gesamte Name enthält.
3. Schreiben Sie eine Konsolenanwendung mit Menu, die:
   a. ein Folge von Ganzzahlen bis 0 einliest und die Summe aller gelesenen Zahlen ausgibt.
   b. findet bei einem gegebenen Zahlen Vektor die längste zusammenhängende Folge, so dass alle Elemente sind gleich.
4. Definieren Sie eine Struktur für die Eingabe von Kundendaten mit folgenden Informationen: Eine Nummer, ein Name, eine Postleitzahl, und eine Ortsname: `int nr`, `char name[20]`, `int plz`, `char ort[20]`. Erzeugen Sie ein Array vom Typ der Struktur und sortieren Sie (mit Insertion-Sort) es nach `name`.
5. Schreiben Sie eine Funktion `rem_repet(char s[])`, die alle Wiederholungen von Zeichen aus einer Zeichenkette entfernt.
6. Schreiben Sie eine Funktion für zweidimensionalen Arrays, die eine mxn-Matrix von Ganzzahlen mit einer nxp-Matrix von Ganzzahlen multipliziert. Die Variable m, n und p können als Konstanten definiert werden.

Jede Aufgabe muss mit mindestens einer Funktion gelöst werden. Für jede Funktion ist es **erforderlich** aussagekräftigen Kommentar zu schreiben und zu Testen!