Seminarblatt 1

1. Datentypen für Variable

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 auch auf die Komponenten zugreifen

```
#include <iostream>

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

int main() {
 date d;

 d.day = 28;
 d.month = 02;
 d.year = 2023;

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

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

complex add( complex x, complex y) {
 complex result;
 result.re = x.re + y.re;
 result.im = x.im + y.im;
 return result;
}

int main() {
 complex a = {1,2};
 complex b = {5,7};

 complex r = 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 entweder auf dem Heap oder dem Stack gespeichert
- Heap = Speicherbereich, in dem dynamisch angeforderte Variablen angelegt werden.
- Stack = Speicherbereich für lokale Variablen und Funktionsparameter. Der Stack hat eine feste und begrenzte 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

```
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 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 (werden wir nicht weiter verwenden)

- Zeichenketten werden als ein eindimensionales Character-Array behandelt
- Wie bei allen Arrays in C/C++, die Länge wird **nicht** mitgespeichert
- Ende der gültigen Zeichenkette ist durch ein 0 Byte gekennzeichnet
- Vereinfachte Initialisierung erlaubt
  ```
  char s[] = "Das ist ein Test."; // String-Schreibweise
  ```
- sizeof(s) gibt den Speicherbedarf des Arrays zurück
  ```
  void foo(char s[]) {
      char s1[] = "ABC";
      s[0] = 'a';
  }
  ```

4.5 C++ Strings

- std::string ist eine Klasse für Zeichenketten mit Operatoren und Methoden zur Manipulation.
- Vereinfachte Initialisierung erlaubt
  ```
  std::string str1 = "hello";
  ```
- Beispiele: Funktion size() für die Länge der Zeichenkette.
- https://en.cppreference.com/w/cpp/string/basic_string

5. Übungen

1. Schreibe 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.

2. Definiere eine Struktur für die Eingabe von Kundendaten mit folgenden Informationen: Eine Nummer, ein Name, eine Postleitzahl, und eine Ortsname.
   Erzeugen Sie ein Array vom Typ der Struktur.

3. Schreibe eine Funktion die zwei Matrizen von Ganzzahlen multipliziert.
   Matrizen zweidimensionalen Arrays. Die erste ist eine MxN-Matrix und die zweite eine NxP-Matrix. Die Variable M, N und P können als Konstante definiert sein.

Jede Aufgabe muss mit mindestens einer Funktion gelöst werden!