Fachbereich Medienproduktion

- Herzlich willkommen zur Vorlesung im Studienfach:
  - Grundlagen der Informatik

Themenübersicht

- Grundlagen der Informatik
  - Grundlagen der Rechnertechnik
  - Algorithmen und Datenstrukturen
    - Primitive Datentypen
    - Abstrakte Datentypen
    - Algorithmen
  - Rechnernetze und das Internet
  - IT Sicherheit

Programmiersprachen

- Assembler
  - Maschinensprache
  - Individuell für jede Prozessorarchitektur
- Hochsprachen
  - C, C++, C#, Basic, Pascal
  - Übersetzung (Compiler) notwendig
- Skriptsprachen
  - Python, Java, PHP
  - Laufzeitumgebung (Runtime) notwendig

Variablen

- Kleiner Speicher für Daten wie
  - Ganze Zahlen
  - Reale Zahlen
  - Zeichen
- Meist Deklaration erforderlich:
  - Festlegen eines Variablennamens und des Typs
  - Dient auch zur Reservierung von Speicher
  - Initialer Wert kann festgelegt werden
- Gültigkeitsbereich der Variable
  - Kann lokal begrenzt sein
  - Kann global im gesamten Programm zur Verfügung stehen

Grundlegende Operationen

- Zuweisungsoperation

  =
- Arithmetische Operationen

  + - * / %
- Boolsche Operationen

  ! NOT & AND | OR
- Vergleichsoperationen

  == != < > && ||

Beispiele zu Operationen

```
// Variablen Deklaration und teilweiser Initialisierung
int a = 5;
int b = 3;
int c;

// Arithmetische Operationen und Zuweisungen
c = a + b;    // Inhalt von c ist 8
c = a – b;    // Inhalt von c ist 2
c = a * b;  // Inhalt von c ist 15

// Weitere Nutzung von Variable c
a = c * b;    // Überschreibt Inhalt von a mit 45 (15 * 3)
b = a + c;   // Überschreibt Inhalt von b mit 60 (45 + 15)
```

Abkürzende Schreibweisen

```
// Empfehlung
a = a + 1;                  a++;    //Inkrementieren
a = a - 1;                  a--;    //Dekrementieren

// eher vermeiden, da Lesbarkeit schlechter
a = a + b;                   a+=b;
a = a * b;                   a*=b;
```

Boolsche Operationen

NOT

Wahrheitstabelle

A	NOT A
0	1
1	0

AND

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

Boolsche Operationen

- Anwendung bitweise auf Variable

z.B. 8-Bit

x	0	1	0	1	1	0	0	1
y	0	0	1	0	1	1	0	1
x & y	0	0	0	0	1	0	0	1

Bedingte Ausführung

```
if (Bedingung)
{ Auszuführender Code falls Bedingung zutrifft }
else
{ Auszuführender Code falls Bedingung nicht zutrifft }

Beispiel
int a = 5;
int b = 3;
int c;

if ( a > b)
  c = a – b;     // Ausführung: 5 - 3
else
  c = b – a;
```

Schleifen

```
while(Bedingung)              // Bedingung wird vorher geprüft
{ Code zur Wiederholten Ausführung }

do
{ Code zur Wiederholten Ausführung
} while(Bedingung);           // Bedingung wird nachher geprüft

for (Iterator)                // Iteration über eine Menge an Elementen
{ Code zur Wiederholten Ausführung }

break;                        // Anweisung innerhalb der Schleife, um Schleife zu verlassen
continue;                     // Anweisung innerhalb der Schleife, um nächsten Durchlauf  zu starten
```

Beispiele zu Schleifen

```
int a = 50;
int b = 3;
int c = 0;

while ( a >= b)
{
   a = a - b;
   c++;
}
```

Funktionen

- Kapselung von Operationen -> Modularisierung
- Wiederverwendbarkeit
- main() – als Programmeinstiegsroutine

```
<Rückgabewert> Funktionsname(Argumente)
{
Funktionsrumpf
return <Rückgabewert>;
}
```

Aufruf mit

```
c = Funktionsname(Argumente)
```

Beispiel zu Funktionen

```
int  kleinerGauss(int n)
{    // Funktionsrumpf
   int summe=0;       //lokale Variable
   while(n>0)
   {
       summe = summe+n;
       n--;
   }
   return summe;
}

main()
{
  int test;
   test = kleinerGauss(5);          // 15
   test = kleinerGauss(10);        // 55
   test = kleinerGauss(1000);     //500.500
}
```

- Vielen Dank für Ihre Aufmerksamkeit!