0. Inhaltsverzeichnis

1. Algorithmen

1.1. Suche Namen im Telefonbuch

```
1. Sei N die Anzahl der Einträge
2. Beginne mit I = N/2
3. Prüfe den I-ten Eintrag
   3.1   Wenn der gesuchte Name kleiner ist
            Dann betrachte nur noch die Einträge 1 bis I-1 und mache bei 1. weiter
   3.2   Wenn der gesuchte Name größer ist
            Dann betrachte nur noch die Einträge I+1 bis N und mache bei 1. weiter
   3.3   Wenn der gesuchte Name der gleiche ist
            Dann beende den Algorithmus
```

1.2. Suche Telefonnummer im Telefonbuch

```
1. Sei N die Anzahl der Einträge
2. Beginne mit I = 1
3. Prüfe den I-ten Eintrag
   3.1   Wenn die gesuchte Telefonnummer nicht mit der gefundenen Nummer übereinstimmt
            Dann erhöhe I um 1 und mache mit 3. weiter
            Andernfalls beende den Algorithmus
```

1.3. Münzautomat

```
1. Sei X=0 der bisher eingeworfene Betrag
2. Warte auf Münzeinwurf
3. Wenn eine Münze eingeworfen wurde, dann ermittle den Wert M der Münze
   3.1   Wenn die Münze ungültig ist
            Dann gib die Münze mit einer Fehlermeldung zurück und mache mit 2. weiter
   3.2   Erhöhe den eingeworfenen Betrag um M und überprüfe X
      3.2.1 Wenn X < 3,- EUR ist
               Dann mache mit 2. weiter
      3.2.2 Wenn x >= 3,- EUR ist
               Dann berechne das Wechselgeld und beende den Algorithmus
```

1.4. Datumsvergleich

```
1. Sei T1.M1.J1 das erste und T2.M2.J2 das zweite Datum
2. Vergleiche die beiden Jahre J1 und J2
   2.1   Wenn J1 < J2
            Dann ist Datum 1 kleiner als Datum 2
   2.2   Wenn J1 > J2
            Dann ist Datum1 größer als Datum 2
   2.3   Andernfalls vergleiche M1 und M2
      2.3.1 Wenn M1 < M2
               Dann ist Datum 1 kleiner als Datum 2
      2.3.2 Wenn M1 > M2
               Dann ist Datum 1 größer als Datum 2
      2.3.3 Andernfalls vergleiche T1 und T2
         2.3.3.1  Wenn T1 < T2
                     Dann ist Datum 1 kleiner als Datum 2
         2.3.3.2  Wenn T1 = T2
                     Dann ist Datum 1 größer als Datum 2
         2.3.3.3  Andernfalls ist Datum 1 gleich Datum 2
```

1.5. Datumsdifferenz

1.5.1. Anzahl der Jahre zwischen zwei Datumsangaben

```
 1. Benutze Aufgabe 1.4 um festzustellen, welches das kleinere Datum D1 und welches das größere Datum D2 ist
 2. Setze Differenz = J2 - J1
 3. Wenn (M2 < M1) oder ((M2 = M1) und (T2 < T1)) ist, dann verringere Differenz um 1
 4. Differenz liefert das gewünschte Eregbnis
```

1.5.2. Anzahl der Monate zwischen zwei Datumsangaben

```
 1. Benutze Aufgabe 1.4 um festzustellen, welches das kleinere Datum D1 und welches das größere Datum D2 ist
 2. Benutze Aufgabe 1.5.1, um die Anzahl Diff_Jahre der Jahre zwischen den beiden Datumsangaben zu bestimmen
 3. Wenn (M2 < M1) ist
      Dann setze Differenz = 12 - M1 + M2
      Sonst setze Differenz = M2 - M1
 4. Wenn (T2 < T1) ist
      Dann verringere Differenz um 1
 5. Das gewünschte Ergebnis ist Differenz + Diff_Jahre *12
```

1.5.3. Anzahl der Tage zwischen zwei Datumsangaben

```
 1. Benutze Aufgabe 1.4 um festzustellen, welches das kleinere Datum D1 und welches das größere Datum D2 ist
 2. Berechne die Anzahl A1 der Tage zwischen Datum 01.M1.J1 und T1.M1.J1
 3. Addiere zu A1 die Anzahl der Tage zwischen Datum 01.01.J1 und 01.M1.J1 ohne Schalttagberücksichtigung
 4. Wenn J1 Schaltjahr und M1 > 2 ist, dann erhöhe A1 um 1
 5. Addiere zu A1 die Anzahl der Tage zwischen Datum 01.01.01 und 01.M1.J1 ohne Schalttagberücksichtigung
 6. Erhöhe A1 um (J1 Div 4) (alle durch 4 teilbaren Jahre sind Schaltjahre)
 7. Ziehe von A1 (J1 Div 100) ab (alle durch 100 teilbaren Jahre sind in Nr. 6 auszunehmen)
 8. Erhöhe A1 um (J1 Div 400) (alle durch 400 teilbaren Jahre sind doch in Nr. 6 zu berücksichtigen)
 9. Berechne mit den Schritten 2-8 die Anzahl A2 der Tage zwischen 01.01.01 und T2.M2.J2
10. Die Differenz zwischen A2 und A1 ergibt das gesuchte Ergebnis
```

1.6. Wochentagsbestimmung

```
1. Berechne mit Aufgabe 1.5.3 die Tagesdifferenz D zwischen heutigem Datum und gesuchtem Datum
2. Zähle vom heutigen Wochentag ausgehend D Tage rückwärts
3. Das Ergebnis ist der gesuchte Wochentag
```

2. Ein-/Ausgabe

2.1. Begrüßungsmeldung:

```
#include <stdio.h>

void main (void)
{
   printf("hello world\n\n");
}
```

2.2. Operation

```
#include <stdio.h>

void main(void)
{
   float zahl1;
   float zahl2;

   printf ("Bitte geben Sie zwei Zahlen ein:\n");
   scanf ("%f%f", &zahl1, &zahl2);
   printf ("%f+%f=%f\n", zahl1, zahl2, zahl1+zahl2);
   printf ("%f-%f=%f\n", zahl1, zahl2, zahl1-zahl2);
   printf ("%f*%f=%f\n", zahl1, zahl2, zahl1*zahl2);
   printf ("%f/%f=%f\n", zahl1, zahl2, zahl1/zahl2);
}
```

2.3. ASCII-Wert

```
#include <stdio.h>

void main(void)
{
   char c;

   printf ("Bitte geben Sie ein Zeichen ein:");
   c=getchar();
   printf ("\nDas Zeichen %c hat den ASCII-Code %d\n", c, c);
}
```

2.4. Datum

```
#include <stdio.h>

void main (void)
{
   int tag;
   int monat;
   int jahr;

   printf ("Bitte geben Sie das gewünschte Datum ein.\n");
   printf ("  Tag = ");
   scanf ("%d", &tag);
   printf ("Monat = ");
   scanf ("%d", &monat);
   printf (" Jahr = ");
   scanf ("%d", &jahr);
   printf ("\nSie haben folgendes Datum eingegeben: %d.%d.%d\n", tag, monat, jahr);
}
```

2.5. Beträge

```
#include <stdio.h>

void main(void)
{
   float betrag1;
   float betrag2;

   printf ("Bitte geben Sie zwei EUR-Betraege ein:\n");
   printf ("Betrag 1 = ");
   scanf ("%f", &betrag1);
   printf ("Betrag 2 = ");
   scanf ("%f", &betrag2);
   printf ("\nSie haben folgende Betraege erfasst:\n");
   printf ("    EUR %10.2f\n", betrag1);
   printf ("    EUR %10.2f\n", betrag2);
}
```

3. Selektionen

3.1. Minimum aus 2

```
#include <stdio.h>

void main (void)
{
   int zahl1;
   int zahl2;

   printf ("Bitte geben Sie zwei Zahlen ein:\n");
   printf ("Zahl 1 = ");
   scanf ("%d", &zahl1);
   printf ("Zahl 2 = ");
   scanf ("%d", &zahl2);
   printf ("Das Minimum von %d und %d ist %d\n", zahl1, zahl2, (zahl1 < zahl2? zahl1 : zahl2));
}
```

3.2. Minimum aus 3

```
#include <stdio.h>

void main (void)
{
   int zahl1;
   int zahl2;
   int zahl3;
   int minimum;

   printf ("Bitte geben Sie drei Zahlen ein:\n");
   printf ("Zahl 1 = ");
   scanf ("%d", &zahl1);
   printf ("Zahl 2 = ");
   scanf ("%d", &zahl2);
   printf ("Zahl 3 = ");
   scanf ("%d", &zahl3);
   if (zahl1 < zahl2)
   {
      if (zahl1 < zahl3)
         minimum = zahl1;
      else
         minimum = zahl3;
   }
   else
   {
      if (zahl2 < zahl3)
         minimum = zahl2;
      else
         minimum = zahl3;
   }
   printf ("Das Minimum von %d, %d und %d ist %d\n", zahl1, zahl2, zahl3, minimum);
}
```

3.3. Eingabe Zahl

```
#include <stdio.h>

void main (void)
{
   int zahl;
   int min=50;
   int max=150;

   printf ("Bitte geben Sie eine Zahl zwischen %d und %d ein:\n", min, max);
   printf ("Zahl = ");
   scanf ("%d", &zahl);
   if ((zahl < min) || (zahl > max))
      printf ("Die Zahl %d liegt nicht zwischen %d und %d!\n", zahl, min, max);
   else
      printf ("Sie haben die Zahl %d eingegeben\n", zahl);
}
```

3.4. Operation

```
#include <stdio.h>

void main (void)
{
   float operand1;
   float operand2;
   char Operator;
   char dummy;

   printf ("Bitte geben Sie zwei Fliesskommazahlen und einen Operator ein:\n");
   printf ("  Zahl 1 = ");
   scanf ("%f", &operand1);
   printf ("  Zahl 2 = ");
   scanf ("%f", &operand2);
   dummy = getchar();
   printf ("Operator = ");
   Operator = getchar();
   switch (Operator)
   {
      case '+': printf ("%.2f+%.2f=%.2f\n", operand1, operand2, operand1 + operand2); break;
      case '-': printf ("%.2f-%.2f=%.2f\n", operand1, operand2, operand1 - operand2); break;
      case '*': printf ("%.2f*%.2f=%.2f\n", operand1, operand2, operand1 * operand2); break;
      case '/': printf ("%.2f/%.2f=%.2f\n", operand1, operand2, operand1 / operand2); break;
      default : printf ("Der Operator \"%c\" ist unzulaessig\n", Operator);
   }
}
```

3.5. Tagesdifferenz

```
#include <stdio.h>

void main (void)

{
   int tag;
   int monat;
   int jahr;
   int diff;

   printf ("Bitte geben Sie ein Datum ein:\n");
   printf ("  Tag = ");
   scanf ("%d", &tag);
   printf ("Monat = ");
   scanf ("%d", &monat);
   printf (" Jahr = ");
   scanf ("%d", &jahr);
   diff = tag;
   switch (monat)
   {
      case 12: diff += 30; /* November   vollständig vorbei */
      case 11: diff += 31; /* Oktober    vollständig vorbei */
      case 10: diff += 30; /* September  vollständig vorbei */
      case  9: diff += 31; /* August     vollständig vorbei */
      case  8: diff += 31; /* Juli       vollständig vorbei */
      case  7: diff += 30; /* Juni       vollständig vorbei */
      case  6: diff += 31; /* Mai        vollständig vorbei */
      case  5: diff += 30; /* April      vollständig vorbei */
      case  4: diff += 31; /* März       vollständig vorbei */
      case  3: diff += 28; /* Februar    vollständig vorbei */
      case  2: diff += 31; /* Januar     vollständig vorbei */
      case  1: diff +=  0; /* kein Monat vollständig vorbei */
   }
   /* Ab März: Schaltjahr berücksichtigen */
   if (monat > 2)
      if (jahr % 4 == 0)
         if (jahr % 100 == 0)
         {
            if (jahr % 400 == 0)
               diff += 1;
         }
         else
            diff += 1;
   printf ("Seit dem 1.1.%d sind %d Tage vergangen\n", jahr, diff);
}
```

3.6. Resttage

```
#include <stdio.h>

void main (void)
{
   int tag;
   int monat;
   int jahr;
   int diff;
   int anzahltage = 365;

   printf ("Bitte geben Sie ein Datum ein:\n");
   printf ("  Tag = ");
   scanf ("%d", &tag);
   printf ("Monat = ");
   scanf ("%d", &monat);
   printf (" Jahr = ");
   scanf ("%d", &jahr);
   /* Berechne Tage seit 01.01. */
   diff = tag;
   switch (monat)
   {
      case 12: diff += 30; /* November   vollständig vorbei */
      case 11: diff += 31; /* Oktober    vollständig vorbei */
      case 10: diff += 30; /* September  vollständig vorbei */
      case  9: diff += 31; /* August     vollständig vorbei */
      case  8: diff += 31; /* Juli       vollständig vorbei */
      case  7: diff += 30; /* Juni       vollständig vorbei */
      case  6: diff += 31; /* Mai        vollständig vorbei */
      case  5: diff += 30; /* April      vollständig vorbei */
      case  4: diff += 31; /* März       vollständig vorbei */
      case  3: diff += 28; /* Februar    vollständig vorbei */
      case  2: diff += 31; /* Januar     vollständig vorbei */
      case  1: diff +=  0; /* kein Monat vollständig vorbei */
   }
   /* Schaltjahr berücksichtigen */
   if (jahr % 4 == 0)
      if (jahr % 100 == 0)
      {
         if (jahr % 400 == 0)
         {
            anzahltage += 1;
            if (monat > 2)
               diff += 1;
         }
      }
      else
      {
         anzahltage += 1;
         if (monat > 2)
            diff += 1;
      }

   /* Ziehe von der Gesamtzahl der Tage die bereits vergangenen Tage ab */
   printf ("Bis zum 31.12.%d muessen noch %d Tage vergehen\n", jahr, anzahltage +1 - diff);
}
```

3.7. Datumsvergleich

```
#include <stdio.h>

void main (void)
{
   int tag1, monat1, jahr1;
   int tag2, monat2, jahr2;

   printf ("Bitte geben Sie zwei unterschiedliche Datumswerte ein:\n");
   printf ("Datum 1\n  Tag = ");
   scanf ("%d", &tag1);
   printf ("Monat = ");
   scanf ("%d", &monat1);
   printf (" Jahr = ");
   scanf ("%d", &jahr1);
   printf ("Datum 2\n  Tag = ");
   scanf ("%d", &tag2);
   printf ("Monat = ");
   scanf ("%d", &monat2);
   printf (" Jahr = ");
   scanf ("%d", &jahr2);
   /* Vergleich der beiden Datumswerte */
   if (tag1 + monat1 *32 + jahr1 *512 < tag2 + monat2 *32 + jahr2 *512)
```

```
      printf ("Der %d.%d.%d ist frueher als der %d.%d.%d\n", tag1, monat1, jahr1, tag2, monat2, jahr2);
   else
      printf ("Der %d.%d.%d ist spaeter als der %d.%d.%d\n", tag1, monat1, jahr1, tag2, monat2, jahr2);
   /*
   Alternative:
   if ((jahr1 < jahr2) ||
       ((jahr1 == jahr2) && (monat1 < monat2)) ||
      ((jahr1 == jahr2) && (monat1 == monat2) && (tag1 < tag2)))
      -> Datum1 früher als Datum2
   else
      -> Datum1 später als Datum2
   */
}
```

3.8. Münzautomat

```
#include <stdio.h>

void main (void)
{
   int muenze;
   int betrag=0;

   printf ("Bitte geben Sie den bisher eingegebenen Betrag in Cent ein:");
   scanf ("%d", &betrag);
   printf ("Welche Muenze wollen Sie einwerfen (Gesamtbetrag 5,- EUR)?");
   scanf ("%d", &muenze);
   switch (muenze)
   {
      case  10:
      case  20:
      case  50:
      case 100:
      case 200: betrag += muenze; break;
      default : muenze = 0;
   }
   if (muenze == 0)
      printf ("ungueltige Muenze! Rueckgabe\n");
   else if (betrag > 500)
      printf ("Ueberzahlung! Wechselgeld: %d\n", betrag - 500);
   else
      printf ("eingeworfener Betrag: %d\n", betrag);
}
```

3.9. Menü

```
#include <stdio.h>

void main (void)
{
   int menue;

   printf ("Ihre Auswahlmoeglichkeiten:\n");
   printf ("---------------------------\n");
   printf ("<1> Menuepunkt 1\n");
   printf ("<2> Menuepunkt 2\n");
   printf ("<3> Menuepunkt 3\n");
   printf ("<4> Menuepunkt 4\n");
   printf ("<5> Menuepunkt 5\n\n");
   printf ("<0> Ende\n");
   printf ("---------------------------\n");
   printf ("Ihre Wahl :");
   scanf ("%d", &menue);
   switch (menue)
   {
      case 0: break;
      case 1: printf ("Menuepunkt 1\n"); break;
      case 2: printf ("Menuepunkt 2\n"); break;
      case 3: printf ("Menuepunkt 3\n"); break;
      case 4: printf ("Menuepunkt 4\n"); break;
      case 5: printf ("Menuepunkt 5\n"); break;
      default: printf ("ungueltige Eingabe!\n");
   }
   if (menue == 0)
      printf ("ENDE!\n");
}
```

4. Iterationen

4.1. Quadratzahlen

```
#include <stdio.h>
#include <math.h>

void main (void)
{
   int quadrat=1;

   do
   {
      printf ("%d^2=%d\n", int(sqrt(quadrat)), quadrat);
      /* (a+1)² = a² + 2a +1 */
      quadrat += 2 * int (sqrt (quadrat)) +1;
   }
   while (quadrat <= 100);

   /* Alternative:

   for (quadrat=1;quadrat<=10;quadrat++)
      printf ("%d^2=%d\n", quadrat, quadrat * quadrat);

   */
}
```

4.2. Zeichen

```
#include <stdio.h>

void main (void)
{
   int c;
   /* ACHTUNG! Funktioniert nicht mit unsigned char, da beim Hochzählen von c bei  */
   /* einem Wert c=255 nicht der Wert c=256, sondern der Wert c=0 zugewiesen wird! */

   for (c=32;c<=255;c++)
      printf ("#%d=%c\n", c, c);
}
```

4.3. Teiler

```
#include <stdio.h>

void main (void)
{
   int zahl;
   int teiler;

   printf ("Bitte geben Sie eine positive Zahl ein:");
   scanf ("%d", &zahl);
   /* Alle Zahlen zwischen 1 und zahl auf Teilbarkeit prüfen */
   for (teiler=1;teiler<=zahl;teiler++)
      if (zahl % teiler == 0)
         printf ("%d ist Teiler von %d\n", teiler, zahl);
}
```

4.4. Teileranzahl

```
#include <stdio.h>
#include <math.h>

void main (void)
{
   int zahl;
   int teiler;
   int teilerzahl=0;

   printf ("Bitte geben Sie eine positive Zahl ein:");
   scanf ("%d", &zahl);
   /* Zu jedem Teiler, der kleiner als die Wurzel der Zahl ist,   */
   /* existiert ein weiterer Teiler, der größer als die Zahl ist. */
   for (teiler=1;teiler<sqrt(zahl);teiler++)
      if (zahl % teiler == 0)
         teilerzahl += 2;
   /* Bei Quadratzahlen ist die Wurzel ebenfalls Teiler */
   if (int (sqrt(zahl)) == sqrt(zahl))
      teilerzahl++;
   printf ("%d besitzt %d Teiler\n", zahl, teilerzahl);
}
```

4.5. Primzahl

```
#include <stdio.h>
#include <math.h>

void main (void)
{
   int zahl;
   int teiler;
   int prim;

   printf ("Bitte geben Sie eine positive Zahl ein :");
   scanf ("%d", &zahl);
   /* zahl kann nur Primzahl sein, wenn zahl > 1 */
   prim = (zahl>1);
   /* Alle Zahlen zwischen 2 und der Wurzel auf Teilbarkeit prüfen. */
   /* Wenn ein Teiler vorliegt, kann zahl keine Primzahl sein.      */
   for (teiler=2; teiler<=int(sqrt(zahl)); teiler++)
      if (zahl % teiler == 0)
         prim = 0;
   if (prim)
      printf ("Die Zahl %d ist Primzahl\n", zahl);
   else
      printf ("Die Zahl %d ist keine Primzahl\n", zahl);
}
```

4.6. Primzahlliste

```
#include <stdio.h>
#include <math.h>

void main (void)
{
   int zahl;
   int teiler;
   int prim;

   for (zahl=1;zahl<=100;zahl++)
   {
      /* zahl kann nur Primzahl sein, wenn zahl > 1 */
      prim = (zahl>1);
      /* Alle Zahlen zwischen 2 und der Wurzel auf Teilbarkeit prüfen. */
      /* Wenn ein Teiler vorliegt, kann zahl keine Primzahl sein.      */
      for (teiler=2; teiler<=int(sqrt(zahl)); teiler++)
         if (zahl % teiler == 0)
            prim = 0;
      if (prim)
         printf ("Die Zahl %d ist Primzahl\n", zahl);
      else
         printf ("Die Zahl %d ist keine Primzahl\n", zahl);
   }
}
```

4.7. Münzeinwurf

```
#include <stdio.h>

void main(void)
{
   int einwurf=0;
   int muenze;
   int betrag=500;

   do
   {
      printf ("Bitte geben Sie den Wert der eingegebenen Münze in Cent ein :");
      scanf ("%d", &muenze);
      switch (muenze)
      {
         case  10:
         case  20:
         case  50:
         case 100:
         case 200: if (einwurf + muenze <= betrag)
                     einwurf += muenze;
                   else
                      printf ("zuviel bezahlt! Rueckgabe: %d\n", muenze);
                   break;
         default: printf ("ungueltige Muenze! Rueckgabe: %d\n", muenze);
      }
      printf ("eingeworfener Betrag: %d\n", einwurf);
   }
   while (einwurf < betrag);
```

```
}
```

4.8. Eingabe Zahl

```
#include <stdio.h>

void main (void)
{
   int zahl;
   int min=10, max=100;
   int gueltig;

   do
   {
      printf ("Bitte geben Sie eine Zahl zwischen %d und %d ein :", min, max);
      scanf ("%d", &zahl);
      gueltig = (zahl >= min) && (zahl <= max);
      if (!gueltig)
         printf ("Das war wohl nix! Bitte versuchen Sie es nochmal.\n");
   }
   while (!gueltig);
   printf ("eingegebene Zahl: %d\n", zahl);
}
```

4.9. Eingabe Zeichen

```
#include <stdio.h>

void main (void)
{
   int c;
   int gueltig;

   do
   {
      printf ("Bitte geben Sie eines der folgenden Zeichen ein : 'A','B','C','D','1','2','3'\n");
      printf ("Ihre Eingabe :");
      /* Die Eingabe eines einfachen <RETURN> abfangen */
      do
         c = getchar();
      while (c == '\n');
      /* Die verschiedenen Eingaben überprüfen */
      switch (c)
      {
         case 'A':
         case 'B':
         case 'C':
         case 'D':
         case '1':
         case '2':
         case '3': gueltig = 1; break;
         default: gueltig = 0;
      }
      if (!gueltig)
         printf ("Das war wohl nix! Bitte versuchen Sie es nochmal.\n");
   }
   while (!gueltig);
   printf ("eingegebenes Zeichen: %c\n", c);
}
```

4.10. Summieren

```
#include <stdio.h>

void main (void)
{
   int teilerg;
   int zahl;
   int op;
   int ende;

   /* Operator abfragen */
   printf ("Bitte geben Sie den Operator (+ oder *) ein :");
   do
      op = getchar ();
   while ((op != '+') && (op != '*'));
   /* Neutralelement der Operation ermitteln */
   teilerg = (op = '+') ? 0 : 1;
   ende = teilerg;
   /* Zahlen einlesen und Operation ausführen */
   do
   {
      printf ("Bitte naechste Zahl eingeben (%d = Ende): ", ende);
      scanf ("%d", &zahl);
```

```
      teilerg = (op = '+') ? teilerg + zahl : teilerg * zahl;
      printf ("Teilergebnis: %d\n", teilerg);
   }
   while (zahl != ende);
   printf ("Endergebnis: %d\n", teilerg);
}
```

4.11. Taschenrechner

```
#include <stdio.h>
#include <ctype.h>

void main (void)
{
   int teilerg;
   int zahl;
   char op=0;

   printf ("Taschenrechner:\n");
   do
   {
      /* Zahl und Operator einlesen */
      printf ("Zahl und Operator eingeben ('=' = Ende)\n");
      scanf ("%d", &zahl);
      /* Je nach gespeichertem Operator Operation ausführen */
      switch (op)
      {
         case '+': teilerg += zahl; break;
         case '-': teilerg -= zahl; break;
         case '*': teilerg *= zahl; break;
         case '/': teilerg /= zahl; break;
         case 0  : teilerg = zahl;
      }
      /* nächsten Operator einlesen */
      do
         op = getchar();
      while (isspace (op));
      printf ("Teilergebnis: %d\n", teilerg);
   }
   while (op != '=');
   printf ("Endergebnis: %d\n", teilerg);
}
```

4.12. ggT und kgV

```
#include <stdio.h>

void main (void)
{
   int zahl1, zahl2;
   int ggT, kgV;

   printf ("Bitte geben Sie zwei positive Zahlen ein:\n");
   printf (" erste Zahl = ");
   scanf ("%d", &zahl1);
   printf ("zweite Zahl = ");
   scanf ("%d", &zahl2);
   /* Alle Zahlen von zahl1 aus absteigend auf Teilbarkeit prüfen */
   ggT = zahl1;
   while ((zahl2 % ggT != 0) || (zahl1 % ggT != 0))
      ggT--;
   /* kgV aus ggT berechnen */
   kgV = (zahl1 * zahl2) / ggT;
   printf ("ggT(%d,%d)=%d\n", zahl1, zahl2, ggT);
   printf ("kgV(%d,%d)=%d\n", zahl1, zahl2, kgV);
}
```

5. Listen

5.1. Suche das größte/kleinste Element in einer Int-Liste mit maximal 100 Elementen

```
#include <stdio.h>

int maximum (int *liste, int n)
/* Liefert das Maximum aus einer Liste mit n Elementen */
{
   int max;

   /* Zunächst wird angenommen, das letzte Element sei das größte */
   n--;
   max = liste[n];
   for (n--;n>=0;n--)
   {
      /* Prüfe alle anderen Elemente bis zum ersten, ob es nicht */
      /* größer ist, als das bisher größte gefundene Element     */
      if (liste[n] > max)
         max = liste[n];
   }
   return max;
}

int minimum (int *liste, int n)
/* Liefert das Minimum aus einer Liste mit n Elementen */
{
   int min;

   /* Zunächst wird angenommen, das letzte Element sei das kleinste */
   n--;
   min = liste[n];
   for (n--;n>=0;n--)
   {
      /* Prüfe alle anderen Elemente bis zum ersten, ob es nicht */
      /* kleiner ist, als das bisher kleinste gefundene Element  */
      if (liste[n] < min)
         min = liste[n];
   }
   return min;
}

void main (void)
{
   int liste[100];
   int i;

   for (i=0;i<100;i++)
      liste[i] = -i*i +20*i +15;
   printf ("Das Maximum ist %d\n", maximum (liste, 100));
   printf ("Das Minimum ist %d\n", minimum (liste, 100));
}
```

5.2. Vertauschen zweier Listen-Elemente

```
#include <stdio.h>

void tausche (int &element1, int &element2)
/* Vertauscht die beiden Variablenwerte */
/* Rückgabe über Referenz-Parameter     */
{
   int temp;

   temp     = element1;
   element1 = element2;
   element2 = temp;
}

void main (void)
{
   int liste [20];
   int i;

   /* Liste mit Werten füllen */
   for (i=0;i<20;i++)
      liste [i] = i;
   /* Elemente vertauschen */
   for (i=0;i<20;i++)
      tausche (liste [i], liste [(2*i)%20]);
   /* Liste ausgeben */
   for (i=0;i<20;i++)
      printf ("%d ", liste [i]);
```

```
   printf ("\n");
}
```

5.3. Vertauschen des größten und des kleinsten Elements einer Liste

```
#include <stdio.h>

int minimum (int *liste, int n, int &position)
/* Liefert das kleinste Element einer Liste mit n Einträgen. */
/* Die Position des Minimums wird über den Referenzparameter */
/* "position" zurückgeliefert.                               */
{
   int min;

   n--;
   min = liste [n];
   position = n;
   for (n--;n>=0;n--)
      if (liste[n] < min)
      {
         min = liste [n];
         position = n;
      }
   return min;
}

int maximum (int *liste, int n, int &position)
/* Liefert das größte Element einer Liste mit n Einträgen.   */
/* Die Position des Maximums wird über den Referenzparameter */
/* "position" zurückgeliefert.                               */
{
   int max;

   n--;
   max = liste [n];
   position = n;
   for (n--;n>=0;n--)
      if (liste[n] > max)
      {
         max = liste [n];
         position = n;
      }
   return max;
}

void tausche (int &element1, int &element2)
{
   int temp;

   temp     = element1;
   element1 = element2;
   element2 = temp;
}

void tausche_minmax (int *liste, int n)
/* Vertauscht Minimum und Maximum einer Liste mit n Einträgen */
{
   int pos_min, pos_max;

   /* Die Funktionen "mininum" und "maximum" aufrufen, */
   /* um die Position des größten und des kleinsten    */
   /* Elements zu erhalten. Das Funktionsergebnis wird */
   /* jeweils ignoriert.                               */
   minimum (liste, n, pos_min);
   maximum (liste, n, pos_max);
   tausche (liste [pos_min], liste [pos_max]);
}

void ausgabe_liste (int *liste, int n)
{
   int i;

   for (i=0;i<n;i++)
      printf ("%d ", liste [i]);
   printf ("\n");
}

void main (void)
{
   int liste [20];
   int i;
```

```
   for (i=0;i<20;i++)
      liste [i] = i;
   /* Minimum und Maximum in der gesamten Liste tauschen */
   tausche_minmax (liste, 20);
   /* Minimum und Maximum in der Teilliste vom 2ten bis zum */
   /* 18ten Element tauschen                          */
   tausche_minmax (&(liste [1]), 18);
   ausgabe_liste (liste, 20);
}
```

5.4. Sortieren einer Liste

```
#include <stdio.h>

void tausche (int &element1, int &element2)
{
   int temp;

   temp     = element1;
   element1 = element2;
   element2 = temp;
}

void sortieren (int *liste, int n)
/* Verwendet Bubble-Sort zum Sortieren */
{
   int nochmal=0;
   int i;
   int durchgaenge=0;

   /* Prüfe, ob in der Liste zwei Elemente hintereinanderstehen, bei denen  */
   /* das Sortierkriterium verletzt ist. Wenn ja, vertausche die beiden und */
   /* mache mit dem nächsten Paar weiter. Sollte beim Durchlaufen eine Ver- */
   /* tauschung nötig gewesen sein, muß die Liste anschließend erneut über- */
   /* prüft werden.                                                         */
   do
   {
      durchgaenge++;
      i = 0;
      /* Liste muß hoffentlich nicht nochmal durchlaufen werden */
      nochmal = 0;
      for (i=0;i<n-1;i++)
         /* Wenn ein Paar nicht richtig sortiert ist ... */
         if (liste [i] > liste [i+1])
         {
            /* ... muß die Sortierung korrigiert und die Liste */
            /* nochmal durchlaufen werden.                     */
            tausche (liste [i], liste [i+1]);
            nochmal = 1;
         }
   }
   while (nochmal);
   printf ("Anzahl der Durchgaenge beim Sortieren: %d\n", durchgaenge);
}

void ausgabe_liste (int *liste, int n)
{
   int i;

   for (i=0;i<n;i++)
      printf ("%d ", liste [i]);
   printf ("\n");
}

void main (void)
{
   int liste[20];
   int i;

   for (i=0;i<20;i++)
      liste [i] = 20 -i;
   ausgabe_liste (liste, 20);
   sortieren (liste, 20);
   ausgabe_liste (liste, 20);
}
```

5.5. Umkehren der Reihenfolge der Elemente

```
#include <stdio.h>

void invers (int *liste, int n)
{
   int i, temp;

   /* Vertausche die Elemente der Liste paarweise um */
    /* die Reihenfolge der Elemente umzudrehen        */
   for (i=0;i<(n /2);i++)
   {
      temp = liste [i];
      liste [i] = liste [n -i -1];
      liste [n -i -1] = temp;
   }
}

void ausgabe (int *liste, int n)
{
   int i;

   /* Ausgabe der Liste */
   for (i=0;i<n;i++)
      printf ("%d ", *(liste++));
   printf ("\n");
}

void main(void)
{
   int liste [10] = {12, 5, 6, 13, 18, 3, 10, 7, 2, 11};

   ausgabe (liste, 10);
   invers (liste, 10);
   ausgabe (liste, 10);
}
```

5.6. Ausgabe aller Indizes, an denen ein gewünschtes Element steht.

```
#include <stdio.h>

void suche (int *liste, int n, int element)
{
   int i;

   printf ("Element %d steht an den Stellen ", element);
   /* Durchsuche die Liste nach dem gewünschten Element ... */
   for (i=0;i<n;i++)
   {
      /* ... und gib bei jeder Übereinstimmung die Position aus */
      if (liste [i] == element)
         printf ("%d ", i+1);
   }
   printf ("\n");
}

void ausgabe (int *liste, int n)
{
   int i;

   /* Ausgabe der Liste */
   for (i=0;i<n;i++)
      printf ("%d ", liste [i]);
   printf ("\n");
}

void main (void)
{
   int liste[10] = {3, 4, 3, 1, 2, 4, 3, 5, 2, 3};

   ausgabe (liste, 10);
   suche (liste, 10, 3);
   suche (liste, 10, 4);
}
```

5.7. Löschen eines Elements und Verschieben aller nachfolgenden Elemente nach vorn.

```
#include <stdio.h>

void del_element (int *liste, int n, int del_pos)
/* Löscht aus der angegebenen Liste das Element mit dem Index     */
/* "del_pos" und verschiebt alle nachfolgenden Elemente nach vorn */
{
   int i;

   /* Ersetze ab der Stelle des zu löschenden Elements jedes */
   /* Element durch seinen Nachfolger                  */
   for (i=del_pos; i<n; i++)
      liste [i] = liste [i+1];
}

void ausgabe (int *liste, int n)
{
   int i;

   /* Ausgabe der Liste */
   for (i=0;i<n;i++)
      printf ("%d ", liste [i]);
   printf ("\n");
}

void main (void)
{
   int liste [10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

   ausgabe (liste, 10);
   del_element (liste, 10, 3);
   ausgabe (liste, 9);
   del_element (liste, 10, 6);
   ausgabe (liste, 8);
}
```

6. Datensätze

6.1. Ein Datensatz für Name, Vorname und Kundennummer

```
#include <stdio.h>

struct kundendaten
{
   int kundennr;
   char name [40];
   char vorname [40];
};

struct kundendaten input (void)
/* Liest einen Kundendatensatz ein und liefert */
/* die Eingabe als Funktionsergebnis zurück    */
{
   struct kundendaten kunde;

   printf ("Bitte geben Sie die Kundendaten ein:\n");
   printf ("(Kundennr=0 -> abbrechen)\n");
   /* Kundendaten einlesen */
   printf ("Kundennr = ");
   scanf ("%d", &kunde.kundennr);
   if (kunde.kundennr != 0)
   {
      printf ("Name     = ");
      scanf ("%39s", &kunde.name);
      printf ("Vorname  = ");
      scanf ("%39s", &kunde.vorname);
   }
   /* Eingabe zurückliefern */
   return kunde;
}

void output (struct kundendaten kunde)
/* Gibt die gewünschten Kundendaten auf dem Bildschirm aus */
{
   printf ("Ihre Eingabe:\n");
   printf ("Kundennr: %d\n", kunde.kundennr);
   printf ("Name:     %s\n", kunde.name);
   printf ("Vorname:  %s\n", kunde.vorname);
}

void main (void)
{
   output (input ());
}
```

6.2. Ein Datensatz für Datumsangaben

```
#include <stdio.h>

struct datumsangabe
{
   int tag;
   int monat;
   int jahr;
};

struct datumsangabe input (void)
/* Liest eine Datumsangabe ein und gibt das Datum */
/* als Funktionswert zurück.                      */
{
   struct datumsangabe datum;
   int gelesen;

   printf ("Bitte geben Sie das gewünschte Datum in der Form tt.mm.yyyy ein:\n");
   printf ("Datum = ");
   gelesen = scanf ("%2d.%2d.%4d", &datum.tag, &datum.monat, &datum.jahr);
   /* Wenn die Eingabe nicht korrekt war, wird das Datum auf 0 gesetzt */
   if (gelesen != 3)
   {
      printf ("Fehlerhafte Eingabe!!!\n");
      datum.tag = 0;
      datum.monat = 0;
      datum.jahr = 0;
   }
   return datum;
}

void output (struct datumsangabe datum)
```

```
/* Gibt das gewünschte Datum aus */
{
   printf ("Ihre Eingabe: \n");
   printf ("%02d.%02d.%02d\n", datum.tag, datum.monat, datum.jahr);
}

void main (void)
{
   output (input ());
}
```

6.3. Konvertieren: Datumsangabe in 10-stellige Zeichenkette

```
#include <stdio.h>

struct datumsangabe
{
   int tag;
   int monat;
   int jahr;
};

struct datumsangabe input (void)
/* Liest eine Datumsangabe ein und gibt das Datum */
/* als Funktionswert zurück.                      */
{
   struct datumsangabe datum;
   int gelesen;

   printf ("Bitte geben Sie das gewünschte Datum in der Form tt.mm.yyyy ein:\n");
   printf ("Datum = ");
   gelesen = scanf ("%2d.%2d.%4d", &datum.tag, &datum.monat, &datum.jahr);
   /* Jahrhundert ggfs. hinzufügen: Zahlen > 30 -> 19xx, Zahlen <= 30 -> 20xx */
   if (datum.jahr < 1000)
      if (datum.jahr > 30)
         datum.jahr = 1900 + datum.jahr;
      else
         datum.jahr = 2000 + datum.jahr;
   /* Wenn die Eingabe nicht korrekt war, wird das Datum auf 0 gesetzt */
   if (gelesen != 3)
   {
      printf ("Fehlerhafte Eingabe!!!\n");
      datum.tag = 0;
      datum.monat = 0;
      datum.jahr = 0;
   }
   return datum;
}

void int_to_string (int zahl, char *string, int laenge)
/* Konvertiert die angegebene Zahl in eine Zeichenkette der Länge "laenge"    */
/* Falls "zahl" zu groß ist, werden die führenden Stellen abgeschnitten!      */
/* ACHTUNG! "string" muß ausreichend viel Speicherplatz zur Verfügung stellen */
/* Andernfalls wird der nachfolgende Speicherbereich überschrieben!!!         */
{
   /* Solange die Zeichenkette die Mindestlänge noch nicht erreicht hat, ... */
   while (laenge > 0)
   {
      /* Ermittle das nächste Zeichen (Einer-Stelle) und schreibe dieses */
      /* Zeichen an die nächste Position von rechts                      */
      *(string+laenge-1) = char ((zahl % 10) +48);
      /* Danach wird "zahl" ganzzahlig durch 10 geteilt, um die nächste */
      /* Stelle zu ermitteln, und laenge um eins reduziert              */
      zahl = zahl / 10;
      laenge--;
   }
}

void datum_in_string (struct datumsangabe datum, char *datstring)
/* Konvertiert das angegebene Datum in eine Zeichenkette.  */
/* ACHTUNG! Es muß sichergestellt sein, daß Tag, Monat und */
/* Jahr zwei- bzw. vierstellig sind, und daß "datstring"   */
/* mindestens 11 Zeichen zur Verfügung stehen!             */
{
   /* Tag in die Zeichenkette eintragen */
   int_to_string (datum.tag, datstring, 2);
   /* In der Zeichenkette zwei Zeichen weiter gehen und */
   /* dort einen Punkt eintragen.                       */
   datstring += 2;
   *(datstring++) = '.';
   /* Monat in die Zeichenkette eintragen */
```

```
   int_to_string (datum.monat, datstring, 2);
   /* In der Zeichenkette wieder zwei Zeichen weiter gehen */
   /* und dort ebenfalls einen Punkt eintragen             */
   datstring += 2;
   *(datstring++) = '.';
   /* Zum Schluß das Jahr eintragen und die Zeichenkette */
   /* mit der Endkennung '\0' abschließen.               */
   int_to_string (datum.jahr, datstring, 4);
   datstring += 4;
   *(datstring) = '\0';
}

void main (void)
{
   struct datumsangabe datum;
   char datstr[11];

   datum = input();
   datum_in_string (datum, datstr);
   printf ("Ihre Eingabe: %s\n", datstr);
}
```

6.4. Konvertieren: 10-stellige Zeichenkette in Datumsangabe

```
#include <stdio.h>
#include <stdlib.h>

struct datumsangabe
{
   int tag;
   int monat;
   int jahr;
};

void output (struct datumsangabe datum)
/* Gibt das gewünschte Datum aus */
{
   printf ("Ihre Eingabe: \n");
   printf ("%02d/%02d/%02d\n", datum.tag, datum.monat, datum.jahr);
}

struct datumsangabe str_in_datum (char *datstring)
/* Konvertiert den angegebenen String in ein Datums-Datensatz   */
/* Achtung! Es muß sichergestellt sein, daß der String die Form */
/* tt.mm.jjjj besitzt!                                          */
{
   struct datumsangabe datum;

   /* Die einzelnen Teile des Strings durch eine Endkennung trennen */
   *(datstring +2) = '\0';
   *(datstring +5) = '\0';
   /* Die einzelnen Teile des Strings in Zahlen konvertieren */
   datum.tag   = atoi (datstring);
   datum.monat = atoi (datstring +3);
   datum.jahr  = atoi (datstring +6);
   /* Statt der Endkennung wieder die Punkte eintragen, da sich */
   /* diese Änderung nach außen auswirkt !!!                    */
   *(datstring +2) = '.';
   *(datstring +5) = '.';
   /* Ergebnis zurückliefern */
   return datum;
}

void main (void)
{
   char datstr [11];
   struct datumsangabe datum;

   printf ("Bitte geben Sie ein Datum im Format 'tt.mm.jjjj' ein: ");
   scanf ("%10s", &datstr);
   datum = str_in_datum (datstr);
   output (datum);
}
```

6.5. Aufgabe 6.4 mit Gültigkeitsprüfung für die Zeichenkette

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

struct datumsangabe
{
   int tag;
```

```
   int monat;
   int jahr;
};

void output (struct datumsangabe datum)
/* Gibt das gewünschte Datum aus */
{
   printf ("Ihre Eingabe: \n");
   printf ("%02d/%02d/%02d\n", datum.tag, datum.monat, datum.jahr);
}

int datum_check (char *datstring)
{
   return ((isdigit (datstring[0])) && (isdigit (datstring[1])) &&
         (datstring[2] == '.')    && (isdigit (datstring[3])) &&
         (isdigit (datstring[4])) && (datstring[5] == '.')    &&
         (isdigit (datstring[6])) && (isdigit (datstring[7])) &&
         (isdigit (datstring[8])) && (isdigit (datstring[9])));
}

struct datumsangabe str_in_datum (char *datstring)
/* Konvertiert den angegebenen String in ein Datums-Datensatz       */
/* Zuvor wird überprüft, ob der String die Form tt.mm.jjjj besitzt! */
{
   struct datumsangabe datum={0,0,0};

   if (!datum_check (datstring))
   {
      printf ("ungueltiges Datumsformat: %s\n", datstring);
      return datum;
   }
   /* Die einzelnen Teile des Strings durch eine Endkennung trennen */
   *(datstring +2) = '\0';
   *(datstring +5) = '\0';
   /* Die einzelnen Teile des Strings in Zahlen konvertieren */
   datum.tag   = atoi (datstring);
   datum.monat = atoi (datstring +3);
   datum.jahr  = atoi (datstring +6);
   /* Statt der Endkennung wieder die Punkte eintragen, da sich */
   /* diese Änderung nach außen auswirkt !!!                    */
   *(datstring +2) = '.';
   *(datstring +5) = '.';
   /* Ergebnis zurückliefern */
   return datum;
}

void main (void)
{
   char datstr [11];
   struct datumsangabe datum;

   printf ("Bitte geben Sie ein Datum im Format 'tt.mm.jjjj' ein: ");
   scanf ("%10s", &datstr);
   datum = str_in_datum (datstr);
   output (datum);
}
```

6.6. Bruchrechnen

```
#include <stdio.h>

struct tbruch
{
   int zaehler;
   int nenner;
};

void ausgabe (struct tbruch bruch)
/* Gibt den Bruch auf dem Bildschirm aus */
{
   printf ("%d/%d\n", bruch.zaehler, bruch.nenner);
}

int ggT (int x, int y)
/* Berechnet rekursiv den ggT von x und y */
{
   if (y == 0)
      return x;
   else
      return ggT (y, x % y);
}
```

```
void kuerzen (struct tbruch &bruch)
/* Kürzt den Bruch gegen den ggT von Zähler und Nenner */
{
   int faktor;

   faktor = ggT (bruch.zaehler, bruch.nenner);
   bruch.zaehler /= faktor;
   bruch.nenner  /= faktor;
}

struct tbruch addition (struct tbruch summand1, struct tbruch summand2)
/* Berechnet die Summe zweier Brüche und kürzt das */
/* Ergebnis so weit möglich                    */
{
   struct tbruch summe;

   summe.zaehler = summand1.zaehler * summand2.nenner +
               summand2.zaehler * summand1.nenner;
   summe.nenner = summand1.nenner * summand2.nenner;
   kuerzen (summe);
   return summe;
}

struct tbruch subtraktion (struct tbruch minuend, struct tbruch subtrahend)
/* Berechnet die Differenz zweier Brüche und kürzt */
/* das Ergebnis so weit möglich                    */
{
   struct tbruch differenz;

   differenz.zaehler = minuend.zaehler * subtrahend.nenner -
                  subtrahend.zaehler * minuend.nenner;
   differenz.nenner = minuend.nenner * subtrahend.nenner;
   kuerzen (differenz);
   return differenz;
}

struct tbruch multiplikation (struct tbruch faktor1, struct tbruch faktor2)
/* Berechnet das Produkt zweier Brüche und kürzt das */
/* Ergebnis so weit möglich                      */
{
   struct tbruch produkt;

   produkt.zaehler = faktor1.zaehler * faktor2.zaehler;
   produkt.nenner  = faktor1.nenner  * faktor2.nenner;
   kuerzen (produkt);
   return produkt;
}

struct tbruch division (struct tbruch dividend, struct tbruch divisor)
/* Berechnet den Quotienten zweier Brüche und kürzt das */
/* Ergebnis so weit möglich                             */
{
   struct tbruch quotient;

   quotient.zaehler = dividend.zaehler * divisor.nenner;
   quotient.nenner  = dividend.nenner * divisor.zaehler;
   kuerzen (quotient);
   return quotient;
}

void main (void)
{
   struct tbruch b1 = {5, 12};
   struct tbruch b2 = {8, 45};

   ausgabe (addition (b1, b2));
   ausgabe (subtraktion (b1, b2));
   ausgabe (multiplikation (b1, b2));
   ausgabe (division (b1, b2));
}
```

6.7. komplexe Zahlen

```
#include <stdio.h>

struct komplex
{
   int x;
   int y;
};
```

```
void ausgabe (struct komplex zahl)
/* Gibt die komplexe Zahl aus */
{
   printf ("%d+%di\n", zahl.x, zahl.y);
}

struct komplex addition (struct komplex summand1, struct komplex summand2)
/* Addiert zwei komplexe Zahlen und liefert das Ergebnis zurück */
{
   struct komplex summe;

   summe.x = summand1.x + summand2.x;
   summe.y = summand1.y + summand2.y;
   return summe;
}

struct komplex subtraktion (struct komplex minuend, struct komplex subtrahend)
/* Berechnet die Differenz zwischen zwei komplexen Zahlen und */
/* liefert das Ergebnis zurück                                */
{
    struct komplex differenz;

    differenz.x = minuend.x - subtrahend.x;
    differenz.y = minuend.y - subtrahend.y;
    return differenz;
}

struct komplex multiplikation (struct komplex faktor1, struct komplex faktor2)
/* Berechnet das Produkt zweier komplexer Zahlen und liefert */
/* das Ergebnis zurück                                       */
{
   struct komplex produkt;

   produkt.x = faktor1.x * faktor2.x - faktor1.y * faktor2.y;
   produkt.y = faktor1.x * faktor2.y + faktor1.y * faktor2.x;
   return produkt;
}

void main (void)
{
   struct komplex z1={1,1}, z2={2,0};

   ausgabe (addition (z1, z2));
   ausgabe (subtraktion (z1, z2));
   ausgabe (multiplikation (z1, z2));
}
```

7. Modularität

7.1. Zahlvergleich

```
#include <stdio.h>

int vergleich (int zahl1, int zahl2)
{
   /* Die Funktion liefert   */
   /* -1, wenn zahl1 < zahl2 */
   /*  0, wenn zahl1 = zahl2 */
   /* +1, wenn zahl1 > zahl2 */
   if (zahl1 < zahl2)
      return -1;
   if (zahl1 > zahl2)
      return 1;
   return 0;
}

void main (void)
{
   int zahl1, zahl2;

   printf ("Bitte geben Sie zwei Zahlen ein:\n");
   printf ("Zahl 1 = ");
   scanf ("%d", &zahl1);
   printf ("Zahl 2 = ");
   scanf ("%d", &zahl2);
   /* Je nach Ergebnis des Funktionsaufrufs Ausgabetext ausgeben */
   switch (vergleich (zahl1, zahl2))
   {
      case -1: printf ("%d ist kleiner als %d\n", zahl1, zahl2); break;
      case  0: printf ("%d ist gleich %d\n", zahl1, zahl2); break;
      case  1: printf ("%d ist groesser als %d\n", zahl1, zahl2); break;
   }
}
```

7.2. Eingabe Zahl

```
#include <stdio.h>

int eingabe_int (int min, int max)
{
   int zahl;
   int gueltig;

   do
   {
      scanf ("%d", &zahl);
      /* Eingabe prüfen */
      gueltig = (zahl >= min) && (zahl <= max);
      /* Bei Fehleingabe Piepston = '\a' und Hinweis ausgeben */
      if (!gueltig)
         printf ("%cBitte nochmal: ", '\a');
   }
   while (!gueltig);
   return zahl;
}

void main(void)
{
   int min=10, max=100;

   printf ("Bitte geben Sie eine Zahl zwischen %d und %d ein: ", min, max);
   printf ("Ihre Eingabe: %d\n", eingabe_int (min, max));
}
```

7.3. Eingabe String

```
#include <stdio.h>

void eingabe_str (char *hinweis, char *eingabe)
{
   printf ("%s", hinweis);
   scanf ("%s", eingabe);
}


void main (void)
{
   char name[40];

   eingabe_str ("Bitte geben Sie Ihren Namen ein: ", name);
   printf ("Hallo %s\n", name);
}
```

7.4. Minimum aus 3

```
#include <stdio.h>

int minimum (int zahl1, int zahl2, int zahl3, int &min_nr)
{
   /* Suche, welche der drei Zahlen die kleinste ist. */
   if (zahl1 <= zahl2)
   {
      if (zahl1 <= zahl3)
         min_nr = 1;
      else
         min_nr = 3;
   }
   else
   {
      if (zahl2 <= zahl3)
         min_nr = 2;
      else
         min_nr = 3;
   }
   /* Liefere das entsprechende Minimum zurück */
   switch (min_nr)
   {
      case 1: return zahl1;
      case 2: return zahl2;
      case 3: return zahl3;
      default: min_nr = 0; return 0;
   }
}

void main (void)
{
   int z1, z2, z3;
   int min, min_nr;

   printf ("Bitte geben Sie drei Zahlen ein:\n");
   printf ("Zahl 1 = ");
   scanf ("%d", &z1);
   printf ("Zahl 2 = ");
   scanf ("%d", &z2);
   printf ("Zahl 3 = ");
   scanf ("%d", &z3);
   min = minimum (z1, z2, z3, min_nr);
   printf ("Das Minimum der drei Eingaben ist %d und wurde als %d. Zahl eingegeben\n",
         min, min_nr);
}
```

7.5. Potenz

```
#include <stdio.h>
#include <math.h>

float potenz (float basis, float exponent)
{
   /* a^b = e^(b * log (a))                     */
   /* Berechnung nur für positive Basen möglich */
   if (basis > 0)
      return float (exp (exponent * log (basis)));
   else
      return 0;
}

void main (void)
{
   float basis, exp;

   printf ("Bitte geben Sie zwei reelle Zahlen ein:\n");
   printf ("Basis (>0) = ");
   scanf ("%f", &basis);
   printf ("Exponent = ");
   scanf ("%f", &exp);
   printf ("Potent = %f\n", potenz (basis, exp));
}
```

7.6. Vertauschen

```
#include <stdio.h>

void tausch (int &zahl1, int &zahl2)
{
   int save;

   /* zyklische Vertauschung durchführen */
   save  = zahl1;
   zahl1 = zahl2;
   zahl2 = save;
}

void main (void)
{
   int z1, z2;

   printf ("Bitte geben Sie zwei Zahlen ein:\n");
   printf ("Zahl 1 = ");
   scanf ("%d", &z1);
   printf ("Zahl 2 = ");
   scanf ("%d", &z2);
   tausch (z1, z2);
   printf ("getauscht: Zahl 1 = %d, Zahl 2 = %d\n", z1, z2);
}
```

7.7. Ziffern umkehren

```
#include <stdio.h>

int reverse (int zahl)
{
   int r=0;

   /* Schiebe das bisher in r gespeicherte Zwischenergebnis um eine Stelle */
   /* nach links und füge die Einerstelle von zahl hinzu. Danach schiebe   */
   /* den Wert von zahl um eine Stelle nach rechts                         */
   while (zahl != 0)
   {
      r = r*10 + (zahl % 10);
      zahl = zahl /10;
   }
   return r;
}

void main (void)
{
   int zahl;

   printf ("Bitte geben Sie eine Zahl ein: ");
   scanf ("%d", &zahl);
   printf ("Rueckwaerts gelesen: %d\n", reverse (zahl));
}
```

7.8. Konvertieren

```
#include <stdio.h>
#include <string.h>

void konvert (int zahl, int to_binaer, char *ergebnis)
{
   int basis;
   char teilerg[256];

   strcpy (ergebnis, "");
   basis = (to_binaer) ? 2 : 16;
   do
   {
      /* bisheriges Ergebnis merken */
      strcpy (teilerg, ergebnis);
      /* nächstes Zeichen ermitteln, das rechts angefügt werden soll */
      switch (zahl %basis)
      {
         case  0: strcpy (ergebnis, "0"); break;
         case  1: strcpy (ergebnis, "1"); break;
         case  2: strcpy (ergebnis, "2"); break;
         case  3: strcpy (ergebnis, "3"); break;
         case  4: strcpy (ergebnis, "4"); break;
         case  5: strcpy (ergebnis, "5"); break;
         case  6: strcpy (ergebnis, "6"); break;
         case  7: strcpy (ergebnis, "7"); break;
         case  8: strcpy (ergebnis, "8"); break;
         case  9: strcpy (ergebnis, "9"); break;
         case 10: strcpy (ergebnis, "a"); break;
         case 11: strcpy (ergebnis, "b"); break;
         case 12: strcpy (ergebnis, "c"); break;
         case 13: strcpy (ergebnis, "d"); break;
         case 14: strcpy (ergebnis, "e"); break;
         case 15: strcpy (ergebnis, "f"); break;
      }
      /* neues Zwischenergebnis erstellen */
      ergebnis = strcat (ergebnis, teilerg);
      zahl = zahl /basis;
   }
   while (zahl != 0);
}

void main (void)
{
   int zahl;
   char zahlkonv[256];

   printf ("Bitte geben Sie eine Zahl ein: ");
   scanf ("%d", &zahl);
   konvert (zahl, 1, zahlkonv);
   printf ("%d ist binaer = %s\n", zahl, zahlkonv);
   konvert (zahl, 0, zahlkonv);
   printf ("%d ist hexadezimal = %s\n", zahl, zahlkonv);
   printf ("Kontrolle: %x\n", zahl);
}
```

8. Rekursion

8.1. Summe

```
#include <stdio.h>

int summe (int bis)
/* berechnet rekursiv die Summe 1+...+bis */
{
   /* Abbruchbedingung testen */
   if (bis == 0)
      /* Beim Erreichen der Abbruchbedingung 0 zurückgeben */
      return 0;
   else
      /* Andernfalls Rekursionsformel verwenden */
      return bis + summe (bis -1);
}

void main (void)
{
   int zahl;

   printf ("Bitte geben Sie eine positive Zahl ein: ");
   scanf ("%d", &zahl);
   printf ("1+...+%d = %d\n", zahl, summe (zahl));
}
```

8.2. Fakultät

```
#include <stdio.h>

int fakultaet (int n)
/* berechnet rekursiv die Fakultät n! */
{
   /* Abbruchbedingung testen */
   if (n==0)
      /* Wenn Abbruchbedingung erreicht, 1 zurückgeben */
      return 1;
   else
      /* Ansonsten Rekursionsformel anwenden */
      return n * fakultaet (n-1);
}

void main (void)
{
   int zahl;

   printf ("Bitte geben Sie eine positive Zahl ein: ");
   scanf ("%d", &zahl);
   printf ("%d! = %d\n", zahl, fakultaet (zahl));
}
```

8.3. Fibonacci-Zahlen

```
#include <stdio.h>

int aufrufe=0;

int fibonacci (int n)
/* Berechnet rekursiv die n-te Fibonacci-Zahl */
/* Erhöht die globale Variable "Aufrufe" pro  */
/* Aufruf um eins.                            */
{
   /* Anzahl der Aufrufe zählen */
   aufrufe++;
   /* Abbruchbedingung testen */
   if (n < 1)
      /* Beim Erreichen der Abbruchbedingung 1 zurückgeben */
      return 1;
   else
      /* Ansonsten Rekursionsformel verwenden */
      return fibonacci (n-1) + fibonacci (n-2);
}

void main (void)
{
   int zahl;

   printf ("Bitte geben Sie ein positive Zahl ein: ");
   scanf ("%d", &zahl);
   printf ("Die %d. Fibonacci-Zahl lautet %d\n", zahl, fibonacci (zahl));
   printf ("Zur Berechnung wurde die Funktion %d mal aufgerufen\n", aufrufe);
}
```

8.4. ggT

```
#include <stdio.h>

int ggT (int x, int y)
/* berechnet rekursiv den ggT von x und y */
{
   /* Wenn y Teiler von x ist, dann muß y der ggT sein */
   if (x % y == 0)
      return y;
   else
      /* Andernfalls ist ggT (x,y) = ggT (x % y, y) = ggT (y, x % y) */
      return ggT (y, x % y);
}

void main (void)
{
   int zahl1, zahl2;

   printf ("Bitte geben Sie zwei positive Zahlen ein:\n");
   printf ("Zahl 1 = ");
   scanf ("%d", &zahl1);
   printf ("Zahl 2 = ");
   scanf ("%d", &zahl2);
   printf ("ggT(%d,%d)=%d\n", zahl1, zahl2, ggT(zahl1, zahl2));
}
```

8.5. Türme von Hanoi

```
#include <stdio.h>

void verschiebe_scheibe (int scheiben_nr, int von, int nach)
{
   printf ("Scheibe %d von Stange %d nach Stange %d verschieben\n", scheiben_nr, von, nach);
}

void verschiebe_turm (int scheiben_anzahl, int von, int nach)
/* Rekursiver Algortihmus der Türme von Hanoi */
{
   int frei = 6 - (von + nach);

   /* Die obere Spitze des Turms (alle Scheiben bis auf */
   /* die unterste) zur Seite legen */
   if (scheiben_anzahl > 1)
      verschiebe_turm (scheiben_anzahl -1, von, frei);
   /* Unterste Scheibe auf die Ziel-Stange verschieben */
   verschiebe_scheibe (scheiben_anzahl, von, nach);
   /* Die obere Spitze ebenfalls auf die Zielstange bewegen */
   if (scheiben_anzahl > 1)
      verschiebe_turm (scheiben_anzahl -1, frei, nach);
}

void main (void)
{
   int anzahl;

   printf ("Bitte geben Sie an, wieviele Scheiben den Turm bilden: ");
   scanf ("%d", &anzahl);
   verschiebe_turm (anzahl, 1, 3);
}
```

9. Dateien

9.1. Zeichenketten in einer Datei speichern

```
#include <stdio.h>
#include <string.h>

int getline (char *line)
/* Die Funktion liest eine Zeile ein und liefert */
/* die Länge der Eingabe als Fkt-Wert zurück.    */
{
   if (gets (line) == NULL)
      return 0;
   else
      return strlen (line);
}

int schreiben (char *dateiname)
/* Schreibt die Eingabe über die Tastatur in eine Datei  */
/* Die Eingabe einer Leerzeile beendet die Schleife      */
/* Die Funktion liefert die Anzahl der eingegeben Zeilen */
{
   FILE *datei;
   char zeile[256];
   int zeilen = 0;

   /* Datei zum Schreiben öffnen */
   datei = fopen (dateiname, "w");
   if (datei == NULL)
   {
      printf ("FEHLER beim Öffnen der Datei %s\n", dateiname);
      return 0;
   }
   /* Solange Zeilen einlesen, bis eine Leerzeile eingegeben wurde */
   while (getline (zeile) > 0)
   {
      /* und die eingegebene Zeile in die Datei speichern */
      fputs (zeile, datei);
      fputs ("\n", datei);
      /* Anzahl der Zeilen erhöhen */
      zeilen++;
   }
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
   return zeilen;
}

void main (void)
{
   /* Backslash im Dateiname wird als '\\' eingegeben */
   char name[40] = "\\temp\\test.txt";

   schreiben (name);
}
```

9.2. Zeichenketten aus einer Datei auslesen

```
#include <stdio.h>

int lesen (char *dateiname)
/* Liest die angegebene Datei aus und gibt den Inhalt auf */
/* dem Bildschirm aus.                                    */
/* Die Funktion liefert die Anzahl der gelesenen Zeilen   */
{
   FILE *datei;
   char zeile[256];
   int zeilen = 0;

   /* Datei zum Lesen öffnen */
   datei = fopen (dateiname, "r");
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return 0;
   }
   /* Die nächste Zeile oder maximal 256 Zeichen einlesen */
   while (fgets (zeile, 256, datei) != NULL)
   {
      /* Wenn Zeichen eingelesen wurden, auf dem Bildschirm ausgeben */
      /* und Anzahl der gelesenen Zeilen erhöhen                     */
      printf ("%s", zeile);
      zeilen++;
```

```
   }
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
   return zeilen;
}

void main (void)
{
   /* Backslash im Dateiname wird als '\\' eingegeben */
   char name[40] = "\\temp\\test.txt";

   lesen (name);
}
```

9.3. Kopieren einer Datei

```
#include <stdio.h>
#include <string.h>

int kopieren (char *quelldatei, char *zieldatei)
/* Kopiert die angegebene Datei                                    */
/* Die Funktion liefert die Anzahl der kopierten Zeichen zurück */
{
   FILE *quelle;
   FILE *ziel;
   char zeichen;
   int anzahlch = 0;

   quelle = fopen (quelldatei, "r");
   if (quelle == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", quelldatei);
      return 0;
   }
   ziel = fopen (zieldatei, "w");
   if (ziel == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", zieldatei);
      return 0;
   }
   /* Quelldatei zeichenweise auslesen */
   do
   {
      zeichen = fgetc (quelle);
      /* Wenn noch ein Zeichen gelesen wurde, in die Zieldatei kopieren */
      if (zeichen != EOF)
      {
         fputc (zeichen, ziel);
         anzahlch++;
      }
   }
   while (zeichen != EOF);
   /* Dateien wieder schließen */
   if (fclose (ziel) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", zieldatei);
   if (fclose (quelle) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", quelldatei);
   return anzahlch;
}

void main (void)
{
   char dateivon[40]  = "\\temp\\test.txt";
   char dateinach[40] = "\\temp\\kopie.txt";

   printf ("%d Zeichen kopiert...\n", kopieren (dateivon, dateinach));
}
```

9.4. Zeichen und Zahlen in eine Datei speichern

```
#include <stdio.h>
#include <ctype.h>

char inputch (void)
/* Fragt solange die Tastatur ab, bis ein "echtes" */
/* Zeichen eingegeben wurde.                       */
/* Die Funktion liefert dieses Zeichen zurück      */
{
   char ch;

   /* Warte ... */
   do
   {
```

```
      ch = getchar();
   }
   /* ... bis ein "echtes" Zeichen eingegeben wurde */
   while (isspace (ch));
   return ch;
}

int inputint (void)
/* Fragt solange die Tastatur ab, bis eine Zahl */
/* eingegeben wurde.                            */
/* Die Funktion liefert diese Zahl zurück       */
{
   int zahl,c;

   /* Warte ... */
   do
   {
      c = scanf ("%d", &zahl);
   }
   /* ... bis wirklich eine Zahl eingegeben wurde */
   while (c==0);
   return zahl;
}

void input (char *dateiname)
{
   FILE *datei;
   char zeichen;
   int zahl;

   /* Datei zum Schreiben öffnen */
   datei = fopen (dateiname, "w");
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return;
   }
   do
   {
      /* Zeichen und Zahl abfragen ... */
      printf ("Bitte geben Sie ein Zeichen ein: ");
      zeichen = inputch();
      printf ("Bitte geben Sie eine Zahl ein (0=Ende): ");
      zahl = inputint();
      /* ... und in die Datei schreiben */
      fprintf (datei, "%c %d ", zeichen, zahl);
   }
   /* Bei Eingabe der Zahl "0" beenden */
   while (zahl != 0);
   /* Datei schließen */
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
}

void main (void)
{
   char datei[40] = "\\temp\\gemischt.txt";

   input (datei);
}
```

9.5. Zeichen und Zahlen aus Nr. 9.4 auslesen

```
#include <stdio.h>

void output (char *dateiname)
{
   FILE *datei;
   char zeichen;
   int zahl;
   int gelesen;

   /* Datei zum Lesen öffnen */
   datei = fopen (dateiname, "r");
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return;
   }
   do
   {
```

```
      /* Zeichen und Zahl einlesen und ausgeben */
      gelesen = fscanf (datei, "%c %d ", &zeichen, &zahl);
      if ((gelesen != EOF) && (gelesen > 0))
         printf ("%c %d ", zeichen, zahl);
   }
   while ((gelesen != EOF) && (gelesen > 0));
   /* Datei schließen */
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
}

void main (void)
{
   char dateiname[40] = "\\temp\\gemischt.txt";

   output (dateiname);
}
```

9.6. Datenstruktur in eine Datei speichern

```
#include <stdio.h>

struct kundendaten
{
   int kundennr;
   char name[20];
   double zuzahlen;
};

void schreiben (char *dateiname, struct kundendaten kunde)
{
   FILE *datei;

   /* Datei zum Anfügen neuer Datensätze öffnen */
   datei = fopen (dateiname, "a+");
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return;
   }
   /* Daten schreiben */
   fprintf (datei, "%d %s %f\n", kunde.kundennr, kunde.name, kunde.zuzahlen);
   /* Datei schließen */
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
}

void main (void)
{
   char dateiname [40] = "\\temp\\kunden.txt";
   struct kundendaten kunde1 = {123, "Meier", 499.95};
   struct kundendaten kunde2 = {421, "Huber", 129.50};

   schreiben (dateiname, kunde1);
   schreiben (dateiname, kunde2);
}
```

9.7. Datenstruktur aus einer Datei auslesen

```
#include <stdio.h>

struct kundendaten
{
   int kundennr;
   char name[20];
   double zuzahlen;
};

struct kundendaten kunde_lesen (FILE *datei, int &gueltig)
/* Liest den nächsten Datensatz ein. Über den Referenzparameter */
/* "gueltig" wird dabei zurückgeliefert, ob noch ein Datensatz  */
/* gelesen werden konnte.                                       */
{
   struct kundendaten kunde;
   int gelesen;

   gelesen = fscanf(datei, "%d ", &kunde.kundennr);
   gelesen += fscanf(datei, "%s ", &kunde.name);
   gelesen += fscanf(datei, "%lf\n", &kunde.zuzahlen); // %lf liefert bei scanf double
   gueltig = (gelesen > 0);
   /* Wenn kein Datensatz gelesen werden konnte, wird der Kundendatensatz geleert */
   if (gelesen == 0)
   {
```

```
      kunde.kundennr = 0;
      *kunde.name = '\0';
      kunde.zuzahlen = 0;
   }
   return kunde;
}

void lesen (char *dateiname)
{
   FILE *datei;
   int ok;
   struct kundendaten kunde;

   /* Datei zum Lesen öffnen */
   datei = fopen (dateiname, "r");
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return;
   }
   /* Daten einlesen */
   do
   {
      kunde = kunde_lesen (datei, ok);
      if (ok)
         printf ("%d %s %f\n", kunde.kundennr, kunde.name, kunde.zuzahlen);
   }
   while (ok);
   /* Datei schließen */
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
}

void main (void)
{
   char dateiname [40] = "\\temp\\kunden.txt";

   lesen (dateiname);
}
```

9.8. Dateiaktionen und Int-Pointern

```
#include <stdio.h>

void schreiben (char *dateiname)
/* Schreibt solange Zahlen in die angegebene Datei */
/* bis eine "0" eingegeben wird.                   */
{
   FILE *datei;
   int zahl;
   int *pzahl=&zahl;

   /* Datei zum Schreiben öffnen */
   datei = fopen (dateiname, "w");
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return;
   }
   printf ("Bitte geben Sie Zahlen ein (0=Ende)\n");
   /* Zahlen einlesen und in die Datei schreiben */
   do
   {
      scanf ("%d", pzahl);
      if (*pzahl != 0)
         fprintf (datei, "%d ", *pzahl);
   }
   while (*pzahl != 0);
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
}

void lesen (char *dateiname)
{
   FILE *datei;
   int zahl;
   int *pzahl=&zahl;
   int gelesen;

   /* Datei zum Lesen öffnen */
   datei = fopen (dateiname, "r");
```

```
   if (datei == NULL)
   {
      printf ("Fehler beim Öffnen der Datei %s\n", dateiname);
      return;
   }
   printf ("Ihre Eingabe war:\n");
   /* Zahlen wieder auslesen und ausgeben */
   do
   {
      gelesen = fscanf (datei, "%d", pzahl);
      if ((gelesen != EOF) && (gelesen > 0))
         printf ("%d, ", *pzahl);
   }
   while ((gelesen != EOF) && (gelesen > 0));
   if (fclose (datei) != 0)
      printf ("Fehler beim Schließen der Datei %s\n", dateiname);
}

void main (void)
{
   char dateiname [40] = "\\temp\\zahlen.txt";

   schreiben (dateiname);
   lesen (dateiname);
}
```