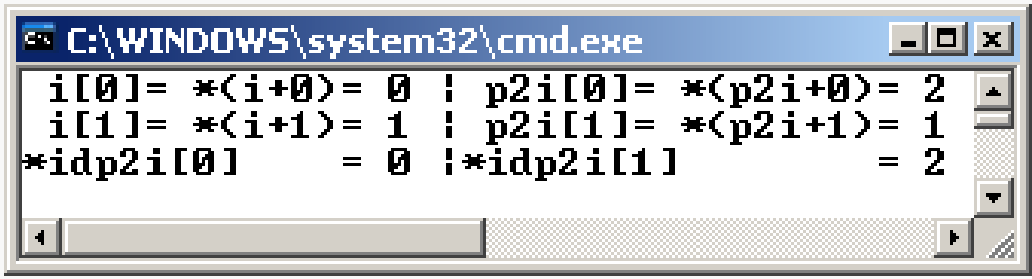
Übersicht: Programmiersprache C

Beispiel: Zeiger, Felder, Zeiger-Felder (*)

```
#define NUM 2
int main()
{ int *p2i, *idp2i[NUM], i[NUM], j1;
  p2i = (int *) calloc (NUM, sizeof(int));
  if (!p2i) exit(0);
  idp2i[0] = &i  [0];
  idp2i[1] = &p2i[0];
for(j1=0;j1<NUM;j1++)
{ i  [j1] = j1;
  p2i[j1] = NUM-j1;
  printf(" i[%d]= *(i+%d)= %d |",j1,j1,*(i+j1));
  printf(" p2i[%d]= *(p2i+%d)= %d\n\r",
                j1,        j1, p2i[j1]); }
printf ("*idp2i[0] = %d |", **idp2i);
printf ("*idp2i[1] = %d\n\r",**(idp2i+1));
free (p2i);  return (0); }
```

```
*idp2i[0] = 0;
```

Nicht initialisiert: Absturz!



(Pointer)

(*) *engl.: pointers, arrays*

Übersicht: Programmiersprache C

Im obigen Beispiel intuitiv beansprucht:

- Felder von Adreß-Variablen (=Zeiger-Variablen) enthalten hintereinander gespeicherte Adressen; der Feldname ist die Adresse der ersten gespeicherten Adresse(-Variable):

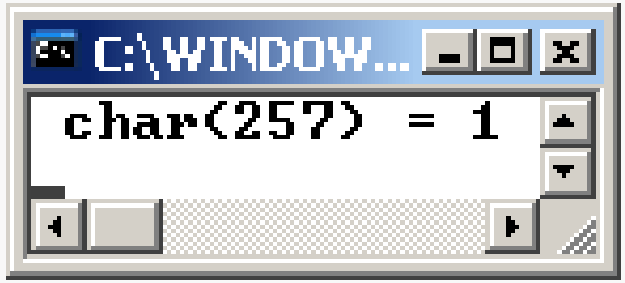
  „Mehrfache Indirektion“ (=„Zeiger auf Zeiger“, „Adresse von Adresse“ – Grad beliebig, meist <3)

- Zeiger-Casten zulässig. Vorsicht: Datenverlust möglich!

Beispiel:

```
int main(void)
{ int   i   = 257;
  char *p2c = (char *) &i;
  i = *p2c;
  printf (" char(257) = %d\n\r", i);
  return (0);
}
```

Überträgt nur ein Byte: Fehler!



(Pointer)

Übersicht: Programmiersprache C

Zeichenkette (String) in C: nullterminiertes **`char`**-Feld.

- Wichtige Unterscheidung:

  `'A'` ist der ganzzahlige ASCII-Wert (65) zum Zeichen „A“

  `"A"` ist die (Startadresse der) Zeichenkette aus dem Zeichen „A“ und dem sog. Nullzeichen **`'\0'`**

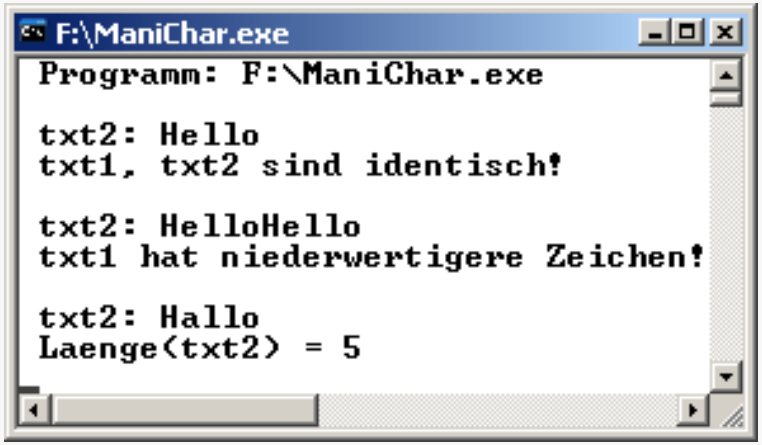
- Neben Wertzuweisung (**`txt[0]='A';`**), nützliche Fktn zur Manipulation v. Zeichenketten ab Adresse (**`char*`**) **`txt1`**, **`txt2`**:

```
strcpy(txt1,txt2);//kopiert Inh. txt2 nach txt1
strcmp(txt1,txt2);//liefert 0, falls identisch,
// <0 falls ASCII(txt1)<ASCII(txt2),>0 sonst
strcat(txt1,txt2);//Anhaengen von txt2 an txt1
                  //(inkl. '\0'-Management!)
strlen (txt1));//liefert txt1-Laenge (ohne'\0')
```

- Hinweis: In **`int main(int argc, char *argv[])`** zeigt **`*argv[0]`** auf den Namen des laufenden Prozesses

Übersicht: Programmiersprache C

Beispiel: /*Manipulation von Zeichenketten:*/



(Pointer)

```
int main(int argc, char *argv[])
{ int   j=0;
  char *txt1="Hello",txt2[80];
  printf(" Programm: %s\n\n\r",
  argv[0]); strcpy(txt2,txt1);
  printf(" txt2: %s\n\r",txt2);
  if (!strcmp (txt1, txt2))
  printf(" txt1, txt2 sind identisch!\n\n\r");
  strcat(txt2,txt1); printf(" txt2: %s\n\r",txt2);
  j = strcmp(txt1, txt2);
  if(!j)printf(" txt1 u. txt2 sind identisch!\n\r");
  else {if(j<0)printf(" txt1");else printf(" txt2");
     printf(" hat niederwertigere Zeichen!\n\n\r");}
  txt2[1] = 'a'; txt2[5] = '\0';
  printf (" txt2: %s\n\r", txt2);
  printf(" Laenge(txt2) = %d\n\r", strlen(txt2));
  _getch(); return (0); }
```

Prüft auf 0!

Übersicht: Programmiersprache C

- Weitere Funktionen zur Manipulation von Zeichenketten ab Adresse (**char***) **txt1**, **txt2**:

```
sprintf(txt1, const char* format[,var1,..]);
//schreibt Zeichenkette aus format[] in txt1
strchr(txt1, int ch ); // liefert Zeiger auf das
//erste Zeichen ch in der Zeichenkette txt1
strstr(txt1, txt2); // liefert Zeiger auf die
//erste Kopie von txt2 in der Zeichenkette txt1
```

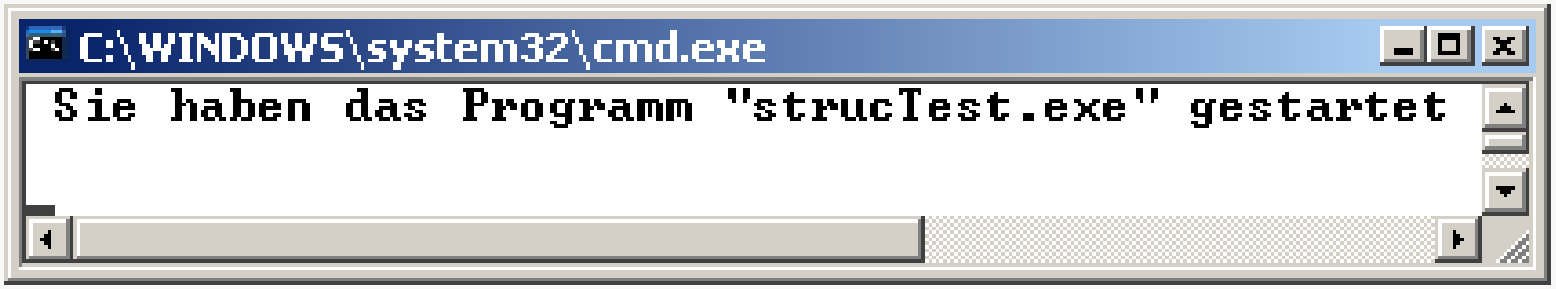
Beispiel:

```
j=1000;
sprintf(txt2, "Testtext Nr. %d", j);
printf (txt2);  printf ("\n\r");
printf ("\'T\' ist der %d. von %d Buchstaben",
        strchr(txt2,'T')-txt2+1, strlen (txt2));
```

Übung:

- Lassen Sie ein gestartetes Programm sich mit seinem Namen melden, ohne den ganzen Pfad zu seiner Speicherung offenzulegen!

- Weitere Hinweise und Hilfen: s. Übungsblatt.



(strucTest.exe)

Übersicht: Programmiersprache C

- Struktur: Sammlung von Variablen („Strukturelementen“) unter einem Namen; Deklaration durch **`struct`** – z.B.:

```
struct Fach
{ char FachName[20];
  int  FachSWS;
};
```

  Anweisung deklariert neuen Datentyp (**`Fach`**); Variablen-Deklaration (z.B.): **`struct Fach kurs1;`**

  Eleganter: Definition neuen Namens für neuen Datentyp:

```
typedef struct Fach Vorlesg;
Vorlesg kurs2; //gleicher Typ wie kurs1
```

- Strukturzuweisung (*engl. structure assignment*) überträgt mit einer Anweisung (*Statement*) alle Daten einer Struktur auf eine andere: **`kurs2 = kurs1;`**
- Struktur-Felder, -Zeiger wie bei Standard-Datentypen

Übung (Forts.):

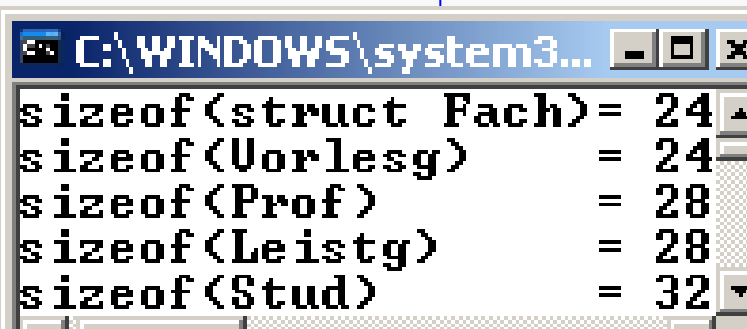
- Struktur-Entwurf für Bestandteile eines Studiengangs:

```
#define LANG    20
typedef struct Fach
{ char FachName[LANG];
  int  FachSWS;
} Vorlesg;

typedef struct
{ char     Name[LANG];
  int      nVorl;
  Vorlesg *Vorl;
} Prof;
```

Struktur name

Typ name

```
typedef struct
{ struct Fach;
  float  FachNote;
} Leistg;

typedef struct
{ char    Name[LANG];
  int     Matrikel;
  int     nLeist;
  Leistg *Leist;
} Stud;
```



Übersicht: Programmiersprache C

- Zugriff auf Strukturelemente: direkt mit Punktoperator (.):

```
#define PROF 3
#define VORL 5
int main(int argc, char *argv[])
{ Prof    BS_P[PROF];
  Vorlesg BS_V[VORL]; int j1,j2; /*...*/
  for (j1=0; j1<PROF; j1++)
  { BS_P[j1].nVorl=3; /*...*/
    BS_P[j1].Vorl = (Vorlesg *)
    calloc(BS_P[j1].nVorl,sizeof(Vorlesg));
    for (j2=0; j2<BS_P[j1].nVorl; j2++)
   { BS_P[j1].Vorl[j2]=BS_V[(j1+j2)%VORL];
                          //Strukturzuweisung } }
```

- ... indirekt (über Zeiger) mit Pfeiloperator (->):

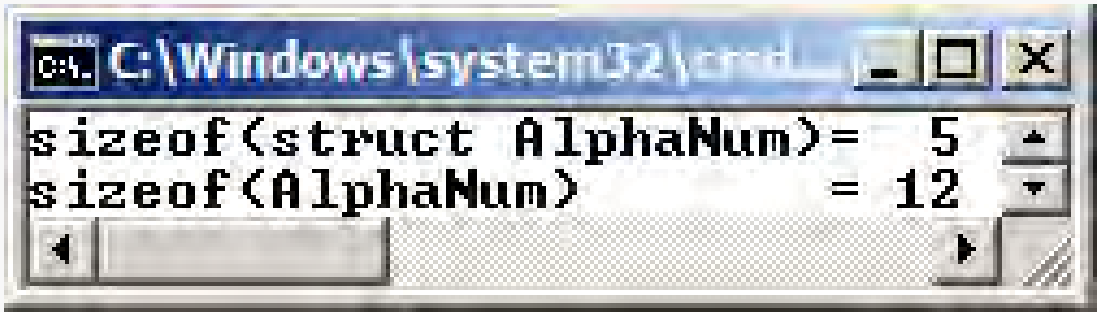
```
int setStud (Stud *defSt, Vorlesg *BS_V)
{ /*...*/  defSt->nLeist = 3;  /*...*/ }
```

- Vorsicht – Data Alignment:

  Der Speicherbedarf von Strukturen ist ein ganzzahliges Vielfaches des größten darin enthaltenen Datentyps.

  Beispiel:

```
struct AlphaNum
{ char Alpha[5]; };
typedef struct
{ char Alpha[5]; int  Arith; } AlphaNum;
printf ("sizeof(struct AlphaNum)= %2d\n\r",
sizeof(struct AlphaNum));
printf ("sizeof(AlphaNum)        = %2d\n\r",
sizeof(AlphaNum));
```



Übersicht: Programmiersprache C

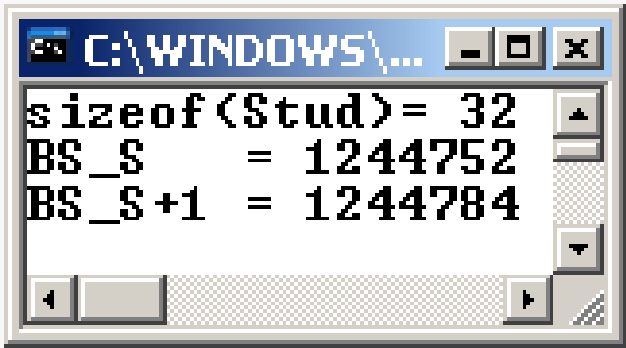
- Wichtiges Instrument: Übergabe v. Struktur-Zeiger an Fkt:
  - Kopie nur einer Adresse auf den Stack (Zeit!)
  - Möglichkeit der Änderung des Struktur-Inhalts

- Achtung: Zeigerarithmetik immer element- / typ-bezogen!

  Beispiel:

```
Stud BS_S[STUD];
printf("sizeof(Stud)= %d\n\r",sizeof(Stud));
printf("BS_S    =%8d\n\r",BS_S);//=&BS_S[0]
printf("BS_S+1 =%8d\n\r",BS_S+1);
```

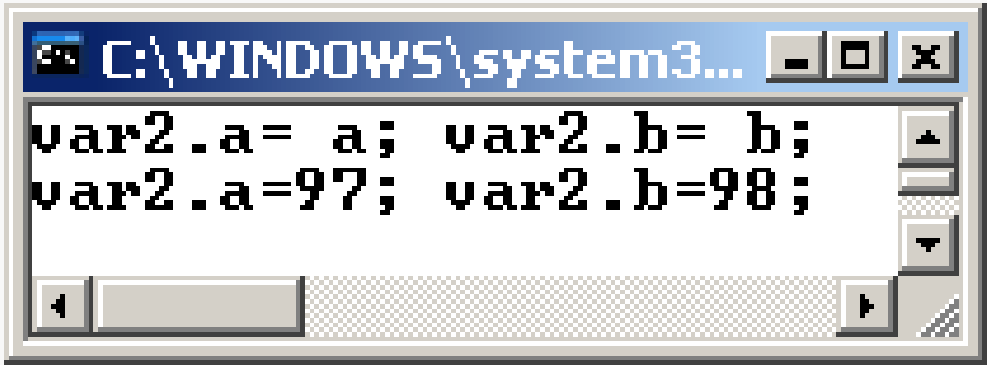
Bei Bedarf Interpretation der Adressen als Zahlen: Casten bzw. Variablen zuweisen (**`long`** oder **`int`**)



Übersicht: Programmiersprache C

- Vereinfachte Einrichtung von Struktur-Variablen ohne Struktur-Namen und ohne Typ-Definition – z.B.:

```
struct //structName //(optional)
{ char a, b;
} var1, var2;
var1.a='a'; var1.b=*"b"; var2=var1;
printf ("var2.a= %c; var2.b= %c;\n\r",
          var2.a,      var2.b);
printf ("var2.a=%2d; var2.b=%2d;\n\r",
          var2.a,      var2.b);
```

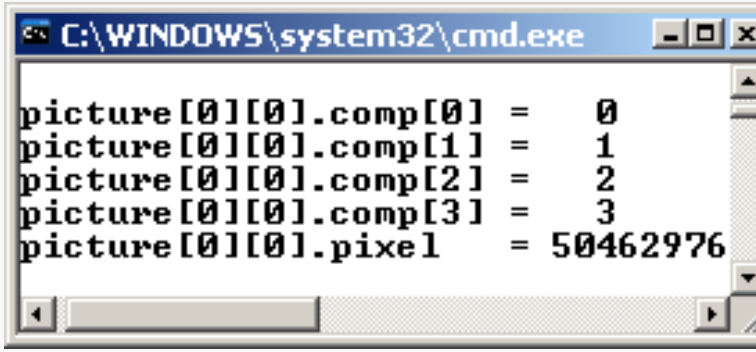


(unionStruc.exe)

Übersicht: Programmiersprache C

- In der Notation verwandt: **union**, d.h. Speicheradresse, die von mehreren Variablen unterschiedlicher Datentypen genutzt wird. **Beispiel:**

```
union//unionName//optional
{           int  pixel;
  unsigned char comp[4];
} picture[HIGH][WIDE];int j1;
for (j1=0; j1<=3; ++j1)
{ picture[0][0].comp[j1]=j1;
  printf(" picture[0][0].comp[%d] = %3d\n\r",
        j1, picture[0][0].comp[j1]);
}//binaer: 00000011 00000010 00000001 00000000
printf (" picture[0][0].pixel   = %d\n\r",
          picture[0][0].pixel);//dez.:50462976
```



(unionStruc.exe)

Typische Anwendungen: Setzen v.RGBA-Farbkomponenten; div. Formatkonvertierungen, spezielle Rundungsverfahren

- In der Notation wie Strukturen, in der Wirkung wie Präprozessor-Konstanten: Aufzählungen (Enumerationen)

  `enum` *`Name`* `{`*`Bezeichner1`*`,` *`Bezeichner2`* `[,…]};`

  vereinbart eine Folge von Namen, die mit ganzzahligen Werten in steigender Folge belegt werden. Variablen vom Typ ***`Name`*** können (nur) solche Werte annehmen.

  Beispiel:

```
enum boolean{niete, geilo, noidea=-1, off, on};
                          //niete==off==0, geilo==on==1
enum boolean flag=off; /*(…)*/
flag = niete;
```

  Vorteil gegenüber **`#define`**: Wertzuweisung automatisch (beginnend bei 0, sofern nicht explizit); Gewähr für unterschiedliche Werte, auch bei Code-Änderung.

Übersicht: Programmiersprache C

- Wichtige Iterationsanweisungen („Schleifen“):

  `for(`*`Initialisierung; Bedingung; Inkrement`*`)`

  `{` *`/*Anweisungen des Schleifenkoerpers*/`* **`}`**

  (seit C99 zulässig: Variablen-Deklaration in der Schleife)

  Beispiel:

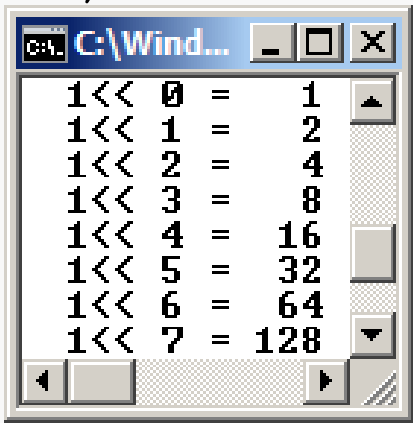
```
for (int bit=0; bit<8; bit++)
  printf("1<<%2d =%4u\n\r  ",bit, 1<<bit);
```

Auch ohne Klammern { }, wenn nur eine Anweisung

Spezialfall: Endlosschleife, z.B.:

```
for ( ; ; ) if (/*(…Abbruch…)*/) break;
```

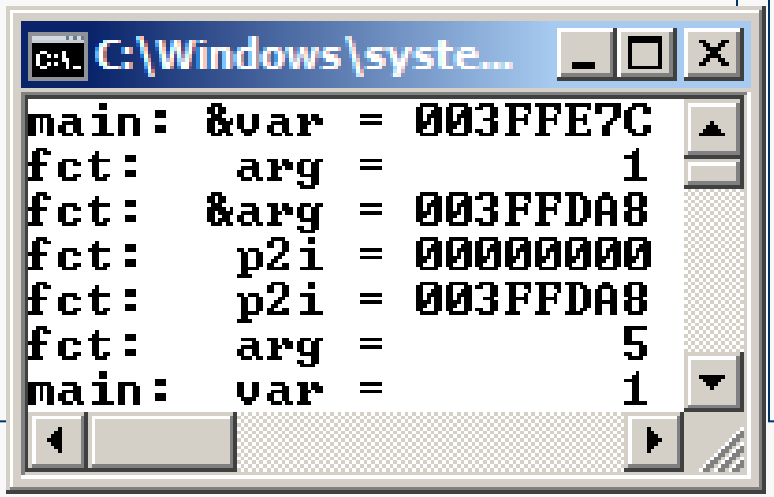
[„Shift-Operatoren“ **<<** und **>>** verschieben die Bits einer Zahl um so viele Stellen, wie der zweite Operand (Zahl oder Variable) angibt. Jede Verschiebung bedeutet Multiplikation oder Division mit 2.]



Übersicht: Programmiersprache C

Beispiel:

```
int main(void)
{ int var=1;
  printf("main: &var =
         %p\n\r", &var);
  fct (var);
  printf("main:  var =
         %8d\n\r", var);
  _getch();
  return 0;
}
```




```
int fct (int arg)
{ int *p2i=NULL;
  printf("fct:   arg =
         %8d\n\r", arg);
  printf("fct:  &arg =
         %p\n\r", &arg);
  printf("fct:   p2i =
         %p\n\r",  p2i);
  p2i = &arg;
 *p2i = 5;
  printf("fct:   p2i =
         %p\n\r",  p2i);
  printf("fct:   arg =
         %8d\n\r", arg);
  return (1);
}
```

Initialisierung!
Sonst: Absturz!

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

- while-Schleife: Ausführung, solange Bedingung erfüllt

```
while (/*Bedingungen*/)
{/*Anweisungen*/
}
```

- do-while-Schleife: mindestens eine Ausführung

```
do { /*Anweisungen*/
   } while ( /*Bedingungen*/ );
```

Für do- / while- / do-while-Schleifen gilt:

- **`break;`**
  Sprung aus der Schleife führt zur ersten Anweisung nach dem Ende der Schleife
- **`continue;`**
  springt ans Ende der Schleife / an den Beginn der nächsten Iteration

- Selektionsanweisung switch:

```
switch (Ausdruck)
{ case Konstante1: /*Anweisungen;*/
  case Konstante2: /*Anweisungen;*/
  default:         /*Anweisungen;*/
}
```

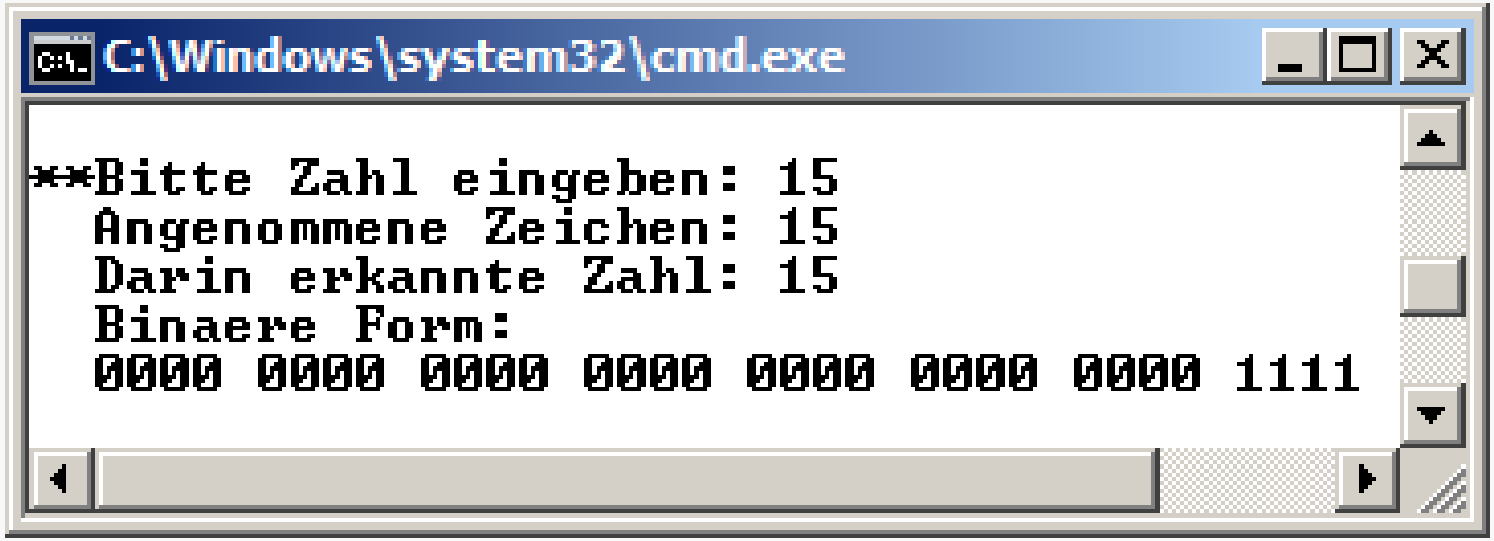
Beispiel:

```
key = _getch();
switch (key)
{ case ESC: exit(0);
  case 'h': helpfunc(); break;
  default:  printf ("%c", key);
}
```

Übung:

- Implementieren Sie ein Programm, das zu einer beliebigen **`int`**- oder **`float`**-Zahl die dazugehörige rechnerinterne Bitbelegung als Folge von Nullen und Einsen ausgibt!

- Weitere Hinweise und Hilfen: s. Übungsblatt.



(BitTest.exe)