3 Integer Zahlen

3.1 Definition

Um ganze Zahlen in Variablen zu speichern, muß spezieller Speicherplatz reserviert werden. Das geschieht durch eine Vereinbarung zu Beginn des Programms, wobei auch eine Initialisierung stattfinden kann.
So werden mit

```
int xylos, holz = 18;
unsigned short bois;
int lec;
```

die Variablen *xylos, holz, lec* und *bois* als ganzzahlig vom Typ `int` bzw. `unsigned short` vereinbart. Gleichzeitig wird *holz* mit dem Wert 18 initialisiert. An den Suns sind die ganzzahligen Datentypen wie folgt implementiert.

Integer Datentypen				
Variablentyp	bytes	Minimum	Maximum	Darst.
`short`	2	-32768	32767	7
`unsigned short`	2	0	65535	7U
`int`	4	-2147483648	2147483647	7
`unsigned int`	4	0	4294967295	7U
`long`	4	-2147483648	2147483647	7L
`unsigned long`	4	0	4294967295	7UL
`long long`	8	-9223372036854775808	9223372036854775807	7LL
`unsigned long long`	8	0	18446744073709551615	7ULL

Wir sehen also, daß auf unseren Rechnern Variablen vom Typ `int` als Datentyp `long` vereinbart werden. Das muß aber nicht auf jedem System so sein.
Neben der dezimalen Schreibweise kann noch die duale und hexadezimale verwendet werden. Oktale ganze Zahlen beginnen mit einer führenden Null oder einem *backslash*, gefolgt von drei Ziffern, hexadezimale Zahlen beginnen mit `\x`, gefolgt von zwei Ziffern, wobei die fehlenden Ziffern als `a, b, ..., f` vereinbart werden.
Mit dem header-file `limits.h` stehen die an dem jeweiligen Compiler festgelegten Grenzen in Variablen zur Verfügung, sie müssen nicht gesondert vereinbart werden.

Variablen aus `limits.h`	
`SHRT_MIN`	kleinste darstellbare `short int` Zahl
`SHRT_MAX`	größte darstellbare `short int` Zahl
`INT_MIN`	kleinste darstellbare `int` Zahl
`INT_MAX`	größte darstellbare `int` Zahl
`UINT_MAX`	größte darstellbare `unsigned int` Zahl
`LONG_MIN`	kleinste darstellbare `long` Zahl
`LONG_MAX`	größte darstellbare `long` Zahl
`ULONG_MAX`	größte darstellbare `unsigned long` Zahl
`LONG_LONG_MIN`	kleinste darstellbare `long long` Zahl
`LONG_LONG_MAX`	größte darstellbare `long long` Zahl
`ULONG_LONG_MAX`	größte darstellbare `unsigned long long` Zahl

Listing 10: Programm zum Testen der integers (noansi) (sintegers.c)

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

volatile int char_min = CHAR_MIN;

int main(void) {
printf("Size of Boolean type is %d byte(s)\n\n",
       (int) sizeof(_Bool));

printf("Number of bits in a character: %d\n", CHAR_BIT);
printf("Size of character types is %d byte\n",
       (int) sizeof(char));
printf("Signed char min: %d max: %d\n",
      SCHAR_MIN, SCHAR_MAX);
printf("Unsigned char min: 0 max: %u\n",
       (unsigned int) UCHAR_MAX);
printf("Default char is ");
if (char_min < 0)
        printf("signed\n\n");
else if (char_min == 0)
        printf("unsigned\n\n");
else
        printf("non-standard\n\n");
```

```
printf("Size of short int types is %d bytes\n",
       (int) sizeof(short));
printf("Signed short min: %d max: %d\n",
       SHRT_MIN, SHRT_MAX);
printf("Unsigned short min: 0 max: %u\n\n\n",
       (unsigned int) USHRT_MAX);

printf("Size of int types is %d bytes\n",
       (int) sizeof(int));
printf("Signed int min: %d max: %d\n", INT_MIN, INT_MAX);
printf("Unsigned int min: 0 max: %u\n\n\n",
       (unsigned int) UINT_MAX);

printf("Size of long int types is %d bytes\n",
       (int) sizeof(long));
printf("Signed long min: %ld max: %ld\n",
       LONG_MIN, LONG_MAX);
printf("Unsigned long min: 0 max: %lu\n\n\n", ULONG_MAX);

printf("Size of long long types is %d bytes\n",
       (int) sizeof(long long));
printf("Signed long long min: %lld max: %lld\n",
       LLONG_MIN, LLONG_MAX);
printf("Unsigned long long min: 0 max: %llu\n",
       ULLONG_MAX);

return 0;
}
```

3.2 Arithmetische Operatoren

Für integer-Zahlen des gleichen Datentyps gibt es die arithmetischen Operatoren `+, -, *`, die nicht näher beschrieben werden müssen. Mit / wird eine ganzzahlige Division durchgeführt, bei der etwaige Nachkommastellen abgeschnitten werden. Mit `%` wird der Rest der ganzzahligen Division ermittelt. Arithmetische Ausdrücke werden von links nach rechts abgearbeitet, wobei

die üblichen Klammervorschriften gelten.
`1949/3+23` ergibt `672`,
`1949/(3+23)` ergibt `74`, `49+5*2` ergibt `59`.
`7/4 != 7 * 3/4`,
`1949%3` ergibt `2`.
Werden Variablen unterschiedlichen Typs verknüpft, findet eine automatische Umwandlung statt. Die wird nicht immer das ausführen, was sich der Programmierer wünscht. Deshalb sollte man die Umwandlung durch das sogenannte *casting* erzwingen. Das geschieht dadurch, daß man den gewünschten Datentyp in runden Klammern vor die Variable setzt, die umgewandelt werden soll.
Problematisch wird es, wenn Variablen unterschiedlichen Datentyps durch Operatoren verbunden werden, aber auch wenn ein Overflow auftritt. Das Verhalten hierbei ist systemabhängig.

Listing 11: Verschiedene Datentypen (diffdata.c)

```c
#include <stdio.h>

int main(void)
{
        short x = 17;
        int z= 17, y = 40000;
        x += y;
        printf("%hd\n", x);
        z += y;
        printf("%d\n", z);
        return 0;
}
```

Das Programm liefert:
```
-25519
40017
```

3.3 Ein- und Ausgabe von Integer-Zahlen

Ein- und Ausgabefunktionen werden in `stdio.h` definiert, der mit `#include <stdio.h>` vereinbart wird.

Ausgabe:

int printf("*text1* %*speztyp text2* ", *var*); *var* wird gemäß der Angaben *spez* und *typ* gedruckt, *text1* und *text2* erscheinen vor und nach der Zahlenausgabe. Es können auch mehrere Variablen ausgegeben werden, zu jeder Variablen muß eine `%` Vereinbarung vorhanden sein. Die Funktion liefert die Zahl der gedruckten Zeichen zurück.

Dabei wird *spez* wie folgt festgelegt:

spez	
`-`	es wird nach links ausgerichtet
`Zahl`	Angabe der reservierten Spalten
`h`	für short Ausgabe
`l`	für long Ausgabe
`ll`	für long long Ausgabe

Die Typen sind wie folgt festgelegt:

typ		
Format	Variablentyp	Darstellung
`c`	int	im ASCII-Code
`d`	int	dezimal
`i`	int	dezimal
`u`	unsigned int	dezimal
`o`	unsigned int	oktal
`x`	unsigned int	hexadezimal
`X`	unsigned int	HEXADEZIMAL

Punkte können ausgelassen werden, die Reihenfolge muß aber eingehalten werden.

Die mit “ eingegrenztein Zeichenketten knnen auch Zeichen enthalten, die nicht zur Sprache C selbst gehören, etwa Umlaute. Ihre Darstellung hängt aber vom jeweils verwendeten Rechner ab. Einige spezielle Zeichen müssen maskiert werden, so `%` durch `%%`; die folgenden Kontrollsequenzen sind ebenso erlaubt.

Kontrollsequenzen	
`\n`	Zeilenvorschub
`\t`	Tabulator
`\b`	backspace
`\a`	Klingelzeichen
`\\`	\
`\?`	?
`\“`	“
`\’`	’

Regeln, ähnlich der Ausgabe, gelten für das Einlesen von *var*, wobei das vorgestellte `&` zwingend ist.

Eingabe:

int scanf("%*spez typ*", &*var*); Es wird gemäß der Vereinbarung in *typ* eingelesen, die Variable *var* ist damit belegt. Es können auch mehrere Zahlen eingelesen werden, für jede Variable muß eine `%` Vereinbarung gesetzt werden. Die Funktion liefert folgende Werte zurück.

Rückgabewerte	
`0`	Formatangabe paßt nicht zum Datentyp,
`-1`	Dateiende erreicht, symbolische Konstante `EOF`,
`n > 0`	Zahl der erfolgreich eingelesenen Variablen.

spez	
`h`	für short Eingabe
`l`	für long Eingabe
`ll`	für long long Eingabe

typ		
Format	Variablentyp	verlangte Darstellung
`d`	int	dezimal
`i`	int	dezimal, oktal oder hexadezimal
`u`	unsigned int	dezimal ohne Vorzeichen
`o`	unsigned int	oktal ohne Vorzeichen
`x`	unsigned int	hexadezimal ohne Vorzeichen
`X`	unsigned int	HEXADEZIMAL ohne Vorzeichen

Hierbei wird die oktale Darstellung an der führenden Null erkannt, die hexadezimale Darstellung an dem führenden `ox` bzw. `0X`.

Das Programm zum $3n + 1$-Problem können wir jetzt so fassen.

Listing 12: 3 n + 1 Problem (3n+1c.c)

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

/* 3 n + 1 Problem, dritter Versuch */
/* input: 134217727 ?? */

static unsigned int check (unsigned int);

unsigned int check (unsigned int m)
{
        if ( m % 2u == 1u) {
                if ( m > UINT_MAX / 3u )
                        return 0u;
                else
                        return 3u * m + 1u;
        } else
                return m/2u;
}

int main(void) {
        unsigned int n, counter = 0u, maxcount = UINT_MAX;
        printf("Eingabe einer natuerlichen Zahl: ");
        (void) scanf("%u", &n);
        while ( n > 1u && counter < maxcount ) {
                counter++;
                n = check(n);
                if (n == 0u)
                        printf("Ueberlauf\n");
                printf("%u\n", n);
        }
        printf("%u Iterationen\n", counter);
        return 0; }
```