Übungen zur Vorlesung
„Grundlagen der Informatik 2“
Studiengang E-Technik

AG Betriebssysteme, FB 3
Jan Bredereke

SoSe 2002

Übung 1 am 18.4.2002

Zahlenrepräsentation

Das Aufgabenblatt 1 ist noch nicht fällig, sondern erst am 2.5.2002. Das Beweispapier dazu im Web wird noch überarbeitet werden.

Zum Vorrechnen kann Aufgabe 2 auf Blatt 1 (der Beweis) gerne auf zwei Leute aufgeteilt werden.

Varianten und Bit-Felder in C

Hiermit kann man die Maschinenrepräsentation einer Zahl inspizieren.

Struktur/Struct

Ist bereits bekannt.

Man kann mehrere Variablen als eine Einheit behandeln. Die Variablen können auch verschiedene Typen haben. Beispiel:

```
/* Komplexe Zahl */
typedef struct {
    int r; /* Real-Anteil */
    int i; /* imaginaerer Anteil */
} complex;
complex c, *pc;
...
    c.r = 1;
    c.i = 2;
    pc = &c;
```

Repräsentation im Speicher:

n	r_{31}	...	r_0
$n+1$	i_{31}	...	i_0

Annahme: Integers sind 32 Bit lang.

Bit-Feld

Man kann auch mehrere Komponenten in einem Speicherwort ablegen lassen, sofern man pro Komponente mit weniger Bits / weniger Wertebereich auskommt:

```
typedef struct {
    unsigned tag : 5;
    unsigned monat : 4;
    unsigned jahr : 12;
} datum;
datum d = {18, 4, 2002};
```

Nur der Typ `unsigned` ist für Komponenten erlaubt.

Repräsentation im Speicher:

n	(unbelegt)	j_{11}	...	j_0	m_3	...	m_0	t_4	...	t_0

Achtung: Auf Big-Endian-Maschinen kann die Belegung andersherum sein. Das ist für uns hier aber nicht relevant.

Variante/Union

Varianten sind Variablen, die Werte verschiedenen Typs enthalten können. Allerdings immer nur abwechselnd. Beispiel:

```
typedef enum {symb, nummer} token_t;
typedef struct {
    token_t sel;
    union {
        char buchst;
        int zahl;
    } val;
} token;
token t;
...
    t.sel = symb;
    t.val.buchst = 'A';
    t.sel = nummer;
    t.val.zahl = 42;
```

Repräsentation im Speicher:

n	(unbelegt)			s_0
$n+1$	(unbelegt)	b_7	...	b_0

oder

n	(unbelegt)		s_0
$n+1$	z_{31}	...	z_0

Es wird genau genug Platz für die größte Variante reserviert.

Typumwandlung mit Varianten

```
...
    t.sel = symb;
    t.val.buchst = 'A';
    printf("%d", t.val.zahl);
```

Was passiert hier?

In C zwingt uns nichts, immer nur auf die aktive Variante zuzugreifen. Wenn wir eine andere Variante wählen, dann wird dasselbe Bitmuster einfach anders interpretiert. Hier wird ein `char` plötzlich als `int` interpretiert.

Ergebnis ist hier 49, da wir den ASCII-Code zur Buchstabendarstellung benutzen. (Falls die unbelegten Bits 0 enthielten.)

Varianten sind gefährlich, weil man leicht Fehler machen kann.

Inspektion der internen Zahlenrepräsentation

Man kann mit Varianten und Bitfeldern die interne Repräsentation von Daten erkunden:

```
/* Inspektion der internen Zahlenrepräsentation */
#include <stdio.h>

typedef struct {
    unsigned b00 : 1;
    unsigned b01 : 1;
    unsigned b02 : 1;
    unsigned b03 : 1;
    unsigned b04 : 1;
    unsigned b05 : 1;
    unsigned b06 : 1;
    unsigned b07 : 1;
    unsigned b08 : 1;
    unsigned b09 : 1;
    unsigned b10 : 1;
    unsigned b11 : 1;
    unsigned b12 : 1;
    unsigned b13 : 1;
    unsigned b14 : 1;
    unsigned b15 : 1;
} bitFeld;

typedef union {
    short zahl;
    bitFeld feld;
} zahlOderFeld;

void druckFeld(zahlOderFeld zf) {
    printf("%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u%u\n",
        zf.feld.b15,
```

```
        zf.feld.b14,
        zf.feld.b13,
        zf.feld.b12,
        zf.feld.b11,
        zf.feld.b10,
        zf.feld.b09,
        zf.feld.b08,
        zf.feld.b07,
        zf.feld.b06,
        zf.feld.b05,
        zf.feld.b04,
        zf.feld.b03,
        zf.feld.b02,
        zf.feld.b01,
        zf.feld.b00);
}

int main() {
    zahlOderFeld zf;

    zf.zahl = 3;
    druckFeld(zf);
    zf.zahl = -2;
    druckFeld(zf);

    return 0;
}
```