
Aufgaben zur Klausur C im WS 96/97 (II21)

Zeit: 75 Minuten

erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 7 Seiten

Aufgabe 1:

Gegeben sei das folgende C-Programm zur Verarbeitung von Mengen als Bitstrings.

```
#include <stdio.h>

typedef unsigned char Menge;
#define MengeMax 8

void printMenge(Menge s) {
    unsigned int i = MengeMax;
    while ( i-- != 0 )
        printf("%1u", (unsigned int)((s >> i) & 1));
}

static unsigned int linecnt = 0;

#define PRINT(s) { printf("%2u) ", ++linecnt); printMenge(s); printf("\n"); }

#define einStueck(n,m) (dieErsten(m+1) ^ dieErsten(n))
#define dieErsten(n) (einElement(n) - 1)
#define einElement(i) ( (Menge)(1 << (i)) )

int main(void) {
    Menge s1;

    PRINT( einElement(1) );
    PRINT( einElement(MengeMax) );

    PRINT( (Menge)0 );
    PRINT( (Menge)42 );

    PRINT( einStueck(3,5) );
    PRINT( einStueck(5,3) );
    PRINT( einStueck(0,MengeMax) );

    PRINT( 42 & 41 );
    PRINT( 42 && 41 );

    s1 = ~einStueck(0,4) | einStueck(2,6); PRINT(s1);
    s1 = einStueck(1,5) ^ ((32 - 1) * 2); PRINT(s1);

    s1 = 6 + 48;
    s1 = s1 ^ (s1 & (~s1 + 1)); PRINT(s1);

    return 0;
}
```

Die Mengen sind in diesem Beispiel 8 Bits lang, können also die Elemente $0, 1, \dots, 7$ enthalten. *printSet* gibt eine Menge im Binärformat aus. Die Menge, die nur die 1 enthält würde als 00000010 ausgegeben werden. Das *PRINT* Makro gibt jeweils eine Menge pro Zeile aus und numeriert die Zeilen durch.

Welche 12 Ausgabezeilen erzeugt dieses Programm?

1)

2)

3)

4)

5)

6)

7)

8)

9)

10)

11)

12)

Aufgabe 2:

Gegeben sei das folgende C-Programm:

```
#include <stdio.h>

#define minm(x,y) ((x) < (y) ? (x) : (y))

long minf(long x, long y) { return x < y ? x : y; }

unsigned ausgewertet = 0;

#define e(x) ( ausgewertet++, (x) )

int main(void) {
    long f[] = { 8, -6, -2, 42, 13 };
    long r0;

    double g[] = { 3.14, 0.25, 2.11, 6.66 };
    double r1;

    ausgewertet = 0;
    r1 = minm( e(g[2]), e(g[3]) );

    printf(" r1 = %4.2f, ausgewertet = %u\n", r1, ausgewertet);

    ausgewertet = 0;
    r1 = minf( e(g[2]), e(g[3]) );

    printf(" r1 = %4.2f, ausgewertet = %u\n", r1, ausgewertet);

    ausgewertet = 0;
    r0 = minm( minm( e(f[3]), e(f[2]) ),
               e(f[4]) );

    printf(" r0 = %ld, ausgewertet = %u\n", r0, ausgewertet);

    ausgewertet = 0;
    r0 = minf( minf( e(f[3]), e(f[2]) ),
               e(f[4]) );

    printf(" r0 = %ld, ausgewertet = %u\n", r0, ausgewertet);

    return 0;
}
```

Welche vier Ausgabezeilen erzeugt dieses Programm:

1)

2)

3)

4)



Aufgabe 3:

Gegeben sei das folgende C-Programmstück für die Verarbeitung von binären Bäumen:

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>

typedef char * String;

typedef struct node * Bintree;
struct node {
    String info;
    Bintree l;
    Bintree r;
};

#define empty ((Bintree)0)

Bintree remove_root(Bintree t) {
    assert(t != empty);

    /* löschen der Wurzel des Baumes t */
    /* ... */

    return t;
}

int compare(String e1, String e2) {
    int c = strcmp(e1,e2);
    return
        ( c > 0) ? 1 : (c == 0) ? 0 : -1;
}

extern Bintree remove(String e1, Bintree t);
```

Implementieren Sie die fehlende *remove* Routine. Diese löscht ein Element, hier eine Zeichenreihe, aus einem binären Baum, falls das Element in dem Baum vorhanden ist.

Nutzen Sie die in dem Programmstück vorgegebenen Datentypen, Makros und Funktionen. (Lösung auf der nächsten Seite).

Die *remove* Routine:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....