THE UNIVERSITY COLLEGE
OF THE CARIBOO

Computing Science – **COMP213**

Lab Exercises #7 – C – *Structures*

Solutions

```
/* File: lab7_1.c
   Date: Nov 2, 2004
   Author: Yanni Giftakis
   Purpose: Display the memory addresses of variables declarations, as well as
       a structure variables; the goal being to determine the order of members
       of the structure.

       For the specific order, see the assignment sheet.

       The order of memory allocation will be:
           - reverse of variables in main(), but the structure is considered
             a single entity, so it's members are allocation in advancing
             order.
*/

// the simple structure that stores a double, int, and char
typedef struct
{
    double d;   // 8 bytes
    int    i;   // 4 bytes
    char   c;   // 1 byte
} struct_type;

int main()
{
       // local variables
    double      d;
    int         i;
    char        c;
    struct_type st;
    int         i2;

       // show the addresses of each variable, including structure and members
    printf ("d:  %p\n",&d);
    printf ("i:  %p\n",&i);
    printf ("c:  %p\n",&c);
    printf ("st: %p\n",&st);
    printf ("st.d: %p (should be same as st)\n",&st.d);
    printf ("st.i: %p\n",&st.i);
    printf ("st.c: %p\n",&st.c);
    printf ("i2: %p\n",&i2);

    getchar();             // pause
    return (0);            // end function
}// end of main()
```

```
/* File: lab7_2.c
   Date: Nov 2, 2004
   Author: Yanni Giftakis
   Purpose:  The program uses a structure to store three values (a,b,c) and an
      average.  Asking the user for the 3 values, it calculates and stores the
      average before displaying all values.

      The program repeats until the user enters the three values 0,0,0.

      (the program is just to experiment with structures)
*/

#include <stdio.h>

// the simple structure to holds the a,b,c, and average
typedef struct
{
    double a,b,c;    // the three values
    double average;  // the average
} newdata_type;

int main()
{
    newdata_type data;    // structure var. to hold values

      // initial entry
    printf ("Enter 3 double values together (0.0 0.0 0.0 to stop)): ");
    scanf ("%lf %lf %lf",&data.a, &data.b, &data.c);

      // loop until user enters all zeroes.
    while (!((data.a == 0.0) && (data.b == 0.0) && (data.c == 0.0)))
    {
        data.average = (data.a+data.b+data.c)/3;  // calculate average
        printf("For the values %lf, %lf, %lf\n   avrg = %lf\n\n",
            data.a, data.b, data.c, data.average);

          // initial entry
        printf ("Enter 3 double values together (0.0 0.0 0.0 to stop)): ");
        scanf ("%lf %lf %lf",&data.a, &data.b, &data.c);
    }
    printf ("\n Thank you. \n");

    getchar();       // pause
    return (0);      // end function
}// end of main()
```

```
/* File: lab7_3.c
   Date: Nov 2, 2004
   Author: Yanni Giftakis
   Purpose:  The program uses a structure to store three values (a,b,c) and an
      average.  Asking the user for the 3 values, it calculates and stores the
      average before displaying all values.

      The program repeats until the user enters the three values 0,0,0.

      (the program is just to experiment with structures)

      This program is the same as lab7_2.c, but uses a pointer to a structure
      variable.
*/

#include <stdio.h>

// the simple structure to holds the a,b,c, and average
typedef struct
{
    double a,b,c;    // the three values
    double average;  // the average
} newdata_type;

int main()
{
      // pointer to structure var. to hold values
    newdata_type *data = (newdata_type*)calloc(1, sizeof(newdata_type));

      // initial entry
    printf ("Enter 3 double values together (0.0 0.0 0.0 to stop)): ");
    scanf ("%lf %lf %lf",&(data->a), &(data->b), &(data->c));

      // loop until user enters all zeroes.
    while (!((data->a == 0.0) && (data->b == 0.0) && (data->c == 0.0)))
    {
        data->average = (data->a+data->b+data->c)/3;  // calculate average
        printf("For the values %lf, %lf, %lf\n   avrg = %lf\n\n",
            data->a, data->b, data->c, data->average);

          // initial entry
        printf ("Enter 3 double values together (0.0 0.0 0.0 to stop)): ");
        scanf ("%lf %lf %lf",&(data->a), &(data->b), &(data->c));
    }
    printf ("\n Thank you. \n");

    getchar();       // pause
    return (0);      // end function
}// end of main()
```

```
/* File: lab7_4.c
   Date: Nov 2, 2004
   Author: Yanni Giftakis
   Purpose:  The program uses an union to overlap a short int and a 2-element
      array of char.

      The goal is to place a number on one member (the short or the array),
      and extract a value from the other.  Since both members share the same
      memory location.

      The user is asked to choose from the options,
        1.  enter 2 characters and obtain the integer
        2.  type in the integer and 2 characters, and verify the input
*/

#include <stdio.h>

// the union that overlaps a short and a 2-element array of char
typedef union
{
    short v;        // variable to hold numeric value
    char  c[2];     // character array at same memory location
} verifytype;

int main()
{
    verifytype translate,  // used to translate chars to ints
               check;       // used in checking (verifying) chars and an int
    char option = ' ';  // which option the user selected

    //-----------------------------------------------
    while (option != '3')   // while the user has not decided to quit
    {
        option = ' ';
        printf ("\n\nWhich option:\n  1-Chars to Int; 2-Check; 3-quit: ");
        scanf ("%c",&option);
        getchar();

        switch (option)
        {
           case '1':    // character to int
              printf ("\nEnter 2 characters (c1 c2): ");
              scanf ("%c %c",&translate.c[0],&translate.c[1]);
              getchar();

              printf ("The related 'password' int is: %d",translate.v);
              break;

           case '2':    // check
              printf ("\nEnter the integer: ");
              scanf ("%d",&translate.v);
              getchar();

              printf ("\nEnter the two characters (c1 c2): ");
              scanf ("%c %c",&check.c[0],&check.c[1]);
              getchar();

              if (translate.v == check.v)   // if both ints are the same
                 printf ("*** Verified ***\n");
              else   // ints are not the same
                 printf (" Not verified. \n");
              break;
        }
    }
    return (0);      // end function
}// end of main()
```

```
/* File: lab7_5.c
   Date: Nov 2, 2004
   Author: Yanni Giftakis
   Purpose:  The program displays a bar-graph based on the input provided by
      the user: a series of input sets: symbols and size.

      See assignment sheet for example.
*/

#include <stdio.h>

// the set type that holds the character and the sequence size
typedef struct
{
    char  symbol;     // character to display
    int   size;       // length size, # of symbols to show
} settype;

//prototype for:
void showset(settype item);

/* the main() function to perform everything */
int main()
{
    int barsize = 0;   // bar graph size
    settype *ray;      // array of sets for the bargraph
    int i=0;           // loop control

    //-------------------------------------------------------------
    printf ("\nPlease enter the number of sets: ");
    scanf ("%d",&barsize);
    getchar();
    ray = (settype*)calloc(barsize, sizeof (settype));   // allocate array

    for (i=0; i<barsize; i++)       // loop to input the sets
    {
        printf ("Set %d: ",(i+1));
        scanf ("%d %c",&(ray[i].size), &(ray[i].symbol));
        getchar();
    }
    printf("\n");
    for (i=0; i<barsize; i++)       // loop for all sets
    {
       showset (ray[i]);                // display bar line
       printf("\n");                    // new line
    }
    printf("\n");

    getchar();
    return (0);      // end function
}// end of main()


/* showset() - displays a line of character a specific size (length) */
void showset(settype item)
{
    int i=0;    // loop control var.

    for (i=1; i<=item.size; i++)  // for for 0 to (size-1) times
       printf ("%c",item.symbol);
}// end of showset()
```

```
/* File: lab7_6.c
   Date: Nov 2, 2004
   Author: Yanni Giftakis
   Purpose:  The program uses the union to associate a character with a
       structure variable that identifies each bit.
*/

#include <stdio.h>

// the "byte" data type
typedef struct
{
   unsigned int b0 : 1;   // lsb
   unsigned int b1 : 1;
   unsigned int b2 : 1;
   unsigned int b3 : 1;
   unsigned int b4 : 1;
   unsigned int b5 : 1;
   unsigned int b6 : 1;
   unsigned int b7 : 1;   // msb
} bytetype;

//the union that relates a character with the bytetype
typedef union
{
   char c;
   bytetype b;
} controltype;

//prototype for:
void display_byte (bytetype word);

/* the main() function to perform everything */
int main()
{
    controltype relate1,relate2;    // three controltype variables

    //-------------------------------------------------------------
    printf ("\nEnter a character, to see the binary form: ");
    scanf ("%c",&relate1.c);               // store character
    getchar();
    printf ("For character %c (%d), the byte is: ",relate1.c,(int)relate1.c);
    display_byte (relate1.b);                // show the binary
    printf("\n\n");

    //-------------------------------------------------------------
       // store the binary for character @ (64): 01000000
    relate2.c = 0;       // set all to zero
    relate2.b.b6 = 1;    // set bit 6 to 1
    printf ("For character %c (%d), the byte is: ",relate2.c,(int)relate2.c);
    display_byte (relate2.b);
    printf("\n\n");

    //-------------------------------------------------------------
       // store the binary for character & (38): 00100110
    relate2.c = 0;       // set all to zero
    relate2.b.b1 = 1;
    relate2.b.b2 = 1;
    relate2.b.b5 = 1;
    printf ("For character %c (%d), the byte is: ",relate2.c,(int)relate2.c);
    display_byte (relate2.b);
    printf("\n\n");

    getchar();
    return (0);      // end function
}// end of main()
```

```
// display_byte() - displays a bytetype as a string of
//                  bits: msb -> lsb
void display_byte (bytetype word)
{
    printf("%d%d%d%d%d%d%d%d",
          word.b7,
          word.b6,
          word.b5,
          word.b4,
          word.b3,
          word.b2,
          word.b1,
          word.b0);
}// end of display_byte()
```