THE UNIVERSITY COLLEGE
OF THE CARIBOO
Computing Science

Lab Exercises #2 – C

Marks: 5 marks | Due: Wednesday, Sept. 29, 2004 (in lecture)

Introduction

The following exercises are indented to help you experiment with the C programming language. The assumption is that you are already familiar with the core syntax concepts of Java (or C++), as well as experienced in creating software solutions to problems.

As reference material, consider a C programming book from the library, so accessing the following links:

- "C programming notes" http://www.eskimo.com/~scs/cclass/notes/top.html ← **very useful**
- "Cprogramming.com" http://www.cprogramming.com/
- the K & R text (which is also available in online form; see website)

Compilers and Tools

As part of the course, you are expected to become familiar with the UNIX/Linux operating system environment, as well as with a standard K&R C/C++ compiler, such as the GNU C++ compiler found in UNIX/Linux.

To satisfy both criteria above, you have been provided an account on the Computing Science Linux server (cs1.cariboo.bc.ca), which is accessible from the labs, and off-campus over the Internet. Further, you can use the Knoppix Linux distribution at home (saving to floppy is required because the Knoppix ramdisk is temporary; ask you instructor how to do this).

> ***Note**: If using UNIX/Linux, all standard C functions, such as* **printf** *and* **strlen** *are available through* ***man*** *pages. For example, looking up help on the* **printf** *function:* [user@server]$ **man printf**

Exercises / Programming Problems

1. C does not initialise variables with default values (as does Java).

   To examine this, write a program that declares, but does not initalise, four (4) variables, one of each type: **int, double, char,** and a **boolean**. Use **printf()** and the special format symbols to display the values of the variables: %d – integers, %f – floating-point, %c – characters.

   For Booleans, use the following "short cut" **if**-statement to display if it is *true* or *false*,

   ```
   boolean x;
   ( x ) ? printf("true") : printf("false");
   ```

   > ***Note**: The ANSI-C standard does not define a* **boolean** *datatype (also a few define a* **bool** *type).. Add the following before the* main() *function definition:* **`typedef short boolean;`** *, and use as expected. The values for this new* **boolean** *type:* 0=*false; all other values=true*

   Do the values change if you run the program repeatedly? Declare a new variable before the ones above (such as: **int x=10**); how does this change the output of the program?

2. The memory addresses of variables are easy to display in C, using the "address of" operator (**&**) and the **printf()** function with the format symbol %p – pointer.

   For example, the following shows the value of a variable as well as the memory address (in hex notation),

   ```
   int val = 20;
   printf ("%d", val);    // display value stored at val
   printf ("%p", &val);   // display memory address of variable val
   ```

Write a program that declares 3 variables of type **int**, followed by 3 variables of type **double** (it does not matter if the variables are initialised). Have the program display the memory addresses of the variables in the exact order they were declared.

With memory addresses based on bytes (8 bits), can you determine how many bytes each type (**int, double**) occupies? Also, by examining the output, what observation can you make on variable order in memory?

Add 3 more variables of type **char**, and display these addresses. Who many bytes for the **char** type?

3. Write a program that displays an isosceles triangle (Christmas-tree shape, with 2 sides being the same length). The size is obtained by asking the user.

Examples (user input is underlined):

```
How large is the triangle: 3
```

```
  *
 ***
*****
```

```
How large is the triangle: 4
```

```
   *
  ***
 *****
*******
```

Suggestions for the program,

- use the **printf()** or **putchar()** functions to output characters
- use the **scanf()** function to capture the user's size input as an integer

```
int size = 0;         // declare variable size
scanf ("%d",&size);   // store input to address of var. size
```

- write, and use, a method called **repeat()** that has two parameters: **length** and **character**; the method displays a line of **character**s of a specific **length** *(consider calling this method to display the spaces* (' ') *and stars* ('*')
- finally, before beginning to program, examine the program mathematically. (*Hint*: always an odd number of stars ('*') on a row and preceding spaces form an inverted triangle (the O's).)

```
OOO*
OO***
O*****
*******
```

4. The compiler directive statement **#define** allows special identifications to be defined for a program (such as constant values and header file (.h) names). An interesting use is to remap the C keywords with your own.

Use the simple "Hello Word" program below, and include the **#define** statements below before **main()**, then rewrite **main()** to use the new keyword defines,

```
#include <stdio.h>

int main (void)
{
    printf ("Hello World");
    return 0;
}
```

```
#define wholeNum int
#define display  printf
#define sendBack return
#define end      ;
```

Conclusion

Submit the well-documented source code for the programs. Provide types answers to all questions.

Attach a cover page, with: *course number, exercises/assignment title, your name, date of submission*.

> ***Note**: Well-documented describes: complete* file documentation *(file name, author, date of last update, and purpose of program); proper comments for variable declarations; proper comments for major segments and statements (such as input / output sections, loops, and function definitions).*