COMP 30400 – Programming I & II

Lab 2

Practical Overview

This practical covers the introduction to variables, user input and user output. By the end of this session you should have your programming portfolio looking like this:

ProgrammingPortfolio/1/helloworld.c
ProgrammingPortfolio/2/helloworldnewline.c
ProgrammingPortfolio/3/add.c
ProgrammingPortfolio/4/sizes.c
ProgrammingPortfolio/5/colour.c
ProgrammingPortfolio/6/colourfirstletter.c
ProgrammingPortfolio/7/colouroverflow.c

For each of the programs in this lab, add comments to your code to help explain what each statement and declaration is doing. Recall that the comment characters are:

/* like this */

where the order is reversed at the end.

1) Portfolio Program 3

Write a program that creates two variables of type int. Assign a value of 2 to one and 3 to the other. Add the two variables, and using the printf() function, output the result to the screen.

Hints: int one = 1;
int three = one + two; /* add variables named one and two*/
printf("%d", variable_name);

2) Portfolio Program 4

Using the sizeof(<variable or type>) function and the printf function, write a program that displays the size of several variable types:

int
char
float
double
An integer array with 3 elements
A character array with 4 elements
A float array with 5 elements
A double array with 6 elements

Hints: int size_of_int = sizeof(int);
int numbers[10];
int size_of_array = sizeof(numbers);
int age = 10;
printf("%d\n", age);

3) Portfolio Program 5

Write a program that asks the user what's their favourite colour.
Store the user's input in a char array.
Print the user's input back into the terminal window using printf.

Use the example earlier in these notes to help.
Test that your code works with long colour names (e.g. "turquoise")

Hints:

Example copied from notes:

```
/* number.c: Input a number from the user and display it.
 * Author: Peter Cahill
 ***/

#include <stdio.h>

int main(void)
{
   char buffer[ 5 ]; /* 4 digits + 1 for the to indicate the end of the character array*/
   printf("What's your favourite number?\n");
   scanf("%4s", buffer );
   printf("You said: %s\n", buffer);
   return 0;
}
```

The example in the notes is limited to 4 characters of input. Many colour names require more characters than this.

4) Portfolio Program 6

Building on your solution to program 4:
Ask the user what's their favourite colour. Then output the first letter of the colour they input.

Hints:

The colour variable will be an array of characters.

There are 2 solutions.

1. Print out the first character of the array (%c), remembering to access the first

element in the array.

2. Modify the second character in the array to be '\0'. This will make printf think the array is of length 1 - as the '\0' character indicates the end of the string.

In a real application, solution 1 is best as we may want to use the name of the colour afterwards. In solution 2, we have modified the array by replacing what was in the second character with '\0'. For proof-of-concept, either is fine.

5) Portfolio Program 7

Building on your solution to program 4, try this as an experiment:

1. Modify your scanf command to use "%s" instead of "%4s" - i.e. where we do not tell scanf how many characters to input.
2. Modify your variable declaration to use a small array (e.g. char colour[3];)
3. Compile and run the program4. When the program is waiting for user input, hold down any key (e.g. "a") for about 1 minute.

5. As soon as you press the return key, the program will probably crash, as the user input has been written to memory locations that were not supposed to be written into.

6. Different systems will report the crash differently. On windows, it will open a window to alert the user that the program crashed. On linux or mac it will report a "segmentation fault".

7. Run the program again, and enter 5 characters – (assuming that you have a character array of length 3). This program will probably work without crashing, as although some of the memory is corrupted, the memory required for the program to finish may be ok. In larger programs, these type of errors may cause the program to crash at any random time in the future as it will only crash when it needs to use the memory that was incorrectly written into.

6) Extras

Remember that programming is a PRACTICAL topic. If you finish these programs within the 2 hours, use the extra time to experiment. Try making other programs, add errors to one of your working programs to see if they are detected by the compiler - what do the error messages say? (e.g. remove a ; from the end of a statement, have a look at the error message.) Try running some of your programs and when they're waiting for your input, just press the return/enter key without any characters first – what happens?

7) Remember to backup your portfolio.