Exercises for

Introduction to Programming in C

Michael Griffiths
IT-Services
The University of Sheffield
Email: m.griffiths@sheffield.ac.uk

January 2021

Exercises for Introduction to Programming in C

Example Problems

1. The course examples are available in a number of places including the course website, Blackboard and as a git repository.

- The website link is http://rcg.group.shef.ac.uk/courses/cic6006/introtoc.zip . Copy the course material into the directory you just created and unzip into that directory.
- HPC users (e.g. traininghpc), start a worker node using the command:
  ```
  srun --pty bash -i
  ```
  - Use cd to change directory to a working directory of your choice then use the git clone command to obtain the examples. The required command is:

```
git clone http://github.com/rcgsheffield/introc
```

2. Open a new file hello.c in an editor or development environment and write the hello world program. Compile and run the application.
3. Modify program arith.c to test the effect of the decrement, increment and assignment operations.
4. Write a program that uses a for loop to display numbers for three different cases
   a. Display the values from 1 to 20
   b. Display the values from 2 to 20
   c. Display the values from 10 to 1

(Hint: Use the example for1.c)

5. Modify whileif.c so that the user provides a value for the number of files and set the while loop control variable to this value.
   a. Introduce a new integer variable iMaxFiles and modify the program so that if the user enters a value for the number of files greater than iMaxFiles then the user is warned and requested for a new value for the number of files.
6. Modify the root finding examples for the newton and bisection method to call a function defined by a c- function (rather than inline as performed in the code example)
7. Compile and run functions.c. Run the program several times and observe that it always provides. The same output.
   a. Seed the random number generator using the statement srand(time(NULL));
   b. Run the program several times and observe the output
8. Modify the integration examples to compute the error function, defined by

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-x^2}\, dx$$

   Modify the program so that erf(x) is computed for a range of values

9. Compile and run the following programs
   a. array.c initialising and using arrays with pointers

    b. bubblesort.c Bubble sort example, using call by reference to manipulate data passed into a function
    c. arrayref.c Using pointer notation to manipulate arrays

10. Modify the routine bubblesort.c to use the qsort function
    a. Use the time function to compare the speed of the bubblesort and quicksort routine.
11. Use scanf to write a routine which reads a string that is input by the user.
12. Compile and run the String Manipulation Examples, program16.

13. With reference to stringarray.c Pointer Arrays, Arrays of strings. Writing a string array to a file, Reading a string array from a file.
    a. Write a function, KeybInputStrArray(char ***pStrArray) that creates an array of strings from user key board input.
    b. Write a function FileWriteStrArray(char ***pStrArray, char *sFilename) that writes an array of strings to the file whose name is held in the character array sFilename.
    c. Write a function, FileInputStrArray(char ***pStrArray) that reads from a file and creates an array of strings.

14. Simple make examples

    The examples for use with these exercises may be found on iceberg at /usr/local/courses/appdev/appdev.tar.gz . Change to your working directory and using:

    ```
    cp  /usr/local/courses/appdev/appdev.tar.gz .
    ```

    The archive can be unpacked using;   tar –zxvf appdev.tar.gz

    Run the make command and build each of the applications. Run and test the applications.

Extra Problems for Introduction to Programming in C

1. **Read and write text to a file**

There is more than one method to do this. This is the one I quickly did in the session.

```
FILE* examplefile;
examplefile=fopen("U:\\Example.txt","w");
fprintf(examplefile,"Example of some text to write to file");
fclose(examplefile);

char szStringFromFile[100];
int iCounter = 0;
examplefile=fopen("U:\\Example.txt","r");
while(!feof(examplefile) && iCounter < 99)
{
szStringFromFile[iCounter++]=(char)fgetc(examplefile);
}
szStringFromFile[iCounter] = '\0';
fclose(examplefile);
puts(szStringFromFile);
```

C++ sequential file access example

```
// Create a structure to hold the data we want
struct Item
{
char szName[20];
char szType[20];
char szPrice[5];
};

// Create some variables & data
Item one={"Apple","Fruit","0.20"};
Item two={"Bannana","Fruit","0.15"};
Item three={"Peas","Veg","0.10"};
Item four={"Digestive","Biscuit","0.30"};
Item five={"Cornflakes","Breakfast","0.05"};

// Empty variables that we will use to read data back in
Item ToRead1={};
Item ToRead4={};
Item ToRead3={};

// Open a file
FILE* file;
file = fopen("U:\\Items.txt","w+");

// Write the data to the file
fwrite(&one,sizeof(Item),1,file);
fwrite(&two,sizeof(Item),1,file);
fwrite(&three,sizeof(Item),1,file);
fwrite(&four,sizeof(Item),1,file);
fwrite(&five,sizeof(Item),1,file);

// Move to the start of the file and read the first item
fseek(file,0,SEEK_SET);
fread(&ToRead1,sizeof(Item),1,file);

// Read the 4th item
fseek(file,(sizeof(Item)*3),SEEK_SET);
fread(&ToRead4,sizeof(Item),1,file);

// Read the 3rd item
fseek(file,(sizeof(Item)*2),SEEK_SET);
fread(&ToRead3,sizeof(Item),1,file);

// Output to the screen
printf("Item 1 is: %s\n",one.szName);
printf("Item 4 is: %s\n",four.szName);
printf("Item 3 is: %s\n",three.szName);

// Close the file
fclose(file);
```

2. **String Copy and Comparsion**

```
// Create some strings
char szStringX[20] = "Hello, World";
char szStringY[20];
char szStringZ[20] = "Hi, World";

// Copy szStringX to szStringY
strcpy(szStringY,szStringX);
printf("szStringX is %s\n",szStringX);
printf("szStringY is %s\n",szStringY);

// Compare szStringX and szStringY to see if they are the same
if(!strcmp(szStringX,szStringY))
printf("Strings are the same\n");
else
printf("Strings are NOT the same\n");

// Compare szStringX and szStringY to see if they are the same
if(!strcmp(szStringX,szStringZ))
printf("Strings are the same\n");
else
printf("Strings are NOT the same\n");
```

3. **Exercise: Adapt the example program structures.c to provide a structure addition function.**

   Add a function to the program structures.c which adds together two structures.

   The function should create and return a pointer to the new structure as follows

   ```
   newstruct.name="addedcontents";
   newstruct.processor ="added processor";
   newstruct.num_procs = n1.num_procs+n2.num_procs;
   ```

   In the example above n1 and n2 refer to the two input values for the structures.

   After the next session the example should be updated again so that the string values may be concatenated.

   HINTS
   1. Simplify the program structure by making use of the typedef statement.
   2. In the addnode function, use malloc to create a new structure (to store the result)

3. The addnode function should return the pointer created using malloc.
4. Don't forget to use free to release the memory!

4. Write a program which.

   1. Requests the user to input a matrix.
   (the user will enter each matrix value in turn)

   2. Display the matrix entered by the user.

   3. Transpose the matrix and display it. (The transpose of a matrix is obtained by writing the matrix rows as columns.

   How could we use pointers to generate a transposed matrix without explicitly moving the data values?