Workshop 5 - Exercises

Consider the exercises below. Try to find solutions on your own as far as you can. If you get stuck, you can ask your workshop tutor for advice or use the help sheet. Also, don't worry if you do not manage to complete all tasks within the workshop time. You can finish the sheet on your own time and sample solutions will be made available on Canvas before the next workshop.

1) Advanced File Operations

Go to the Units Section of module Programming for Engineers in Canvas, download the file data.txt and store it in your working directory. DO NOT open the file manually in a text editor – just save it as a file. The file contains three lines of text with no more than 80 characters each. Create a program that takes a file name as argument, then reads the file, and stores the data in a two-dimensional array of characters: Each line of text should be stored along one of the dimensions of the array. The content of each individual line of text should be stored along the other dimension. After storing the contents of the file, write a function that iterates over the array to output the contents on screen in the right order. Have a line break after each line.

2) Dynamic memory allocation and pointers

Create a program that

a) asks the user for a text input of no more than 100 characters and then
b) creates a copy of that input, using pointers. Make sure the copy does not use more memory than necessary
c) prints out onto the screen
   i. how many bytes of memory were allocated for the copy
   ii. the string contained in the copy

Tip: Find out what a null-terminated string is and make sure the copy allows space for the null character.

3) Understanding pointers

Predict the output of the following programs yourself. Then type the code in DevC++, compile and run to check if the output is same as predicted. If the output appears different, think why it is so. Ask your workshop tutor if required.

Program 1:

```
#include <stdio.h>

#define R 10
#define C 20

int main()
{
   int (*p)[R][C];
   printf("%d",  sizeof(*p));
   getchar();
   return 0;
}
```

Program 2:

```
#include <stdio.h>
int main()
{
    int a[5] = {1,2,3,4,5};
    int *ptr = (int*)(&a+1);
    printf("%d %d", *(a+1), *(ptr-1));
    return 0;
}
```

Program 3:

```
#include <stdio.h>
int f(int x, int *py, int **ppz)
{
  int y, z;
  **ppz += 1;
   z  = **ppz;
  *py += 2;
   y = *py;
   x += 3;
   return x + y + z;
}

void main()
{
   int c, *b, **a;
   c = 4;
   b = &c;
   a = &b;
   printf( "%d", f(c,b,a));
   getchar();
}
```