Notes on Files

1. We typically use files when
   - we have to keep data for a longer period than the time the program is being executed, and
   - do not know how many values will have to be processed.

   For example, a class list to keep all the students' student numbers and assignment marks will be typically kept in a file. The number of students in a class will vary from year to year or semester to semester. We also want to keep record of these student numbers and the assignment marks each student obtained, for a considerable time, certainly longer than the time the program is running. Using a different file for each class will allow us to keep record of the student numbers and assignment marks for these classes, while using the same program to e.g. calculate the year mark for each student, provided we save the data in different files.
   Therefore, most programs are written to be use multiple times to process similar data, and the ability to specify which data file to use when executing the program should be built into the program.

2. We can create an input file by using the Code::Blocks editor and creating a new source file, enter the data, and save it as a file with an extension of .dat. You can also use Notepad. Save your input file in the same directory where you save your program.
   The contents of the input file (and any output file should the program have produced one or more) can be viewed with Code::Blocks. This file is opened with `File | Open Project or File` and by selecting the correct file in the correct directory.

3. One gets input from a file into your program, or send output to a file from your program by using streams, a special kind of variable called an object in C++. The object type (class) for input-variable streams is named `ifstream`, and for output-variable streams, `ofstream`. To use streams, the header file for the `fstream` library has to be added to the code with the `#include <fstream>` directive for file processing.
   ```
   #include <fstream> //directive to use for file processing
   ```

4. Every input and output file used by your program has two names, an external file name and an internal file name. The external file name is the real file name, i.e. the name you will see in a directory list, but this file name is only used when the file is opened with the function `open()`. One connects the stream object (used in the program code) to the actual file by opening the file with the `open()` member function. Once the file has been opened you always use the stream name. Note that the `open()` function expects a C string parameter, which is why the external file name (`inName` in the code below) obtained from the user is converted to a C string with `inName.c_str()`.

   Note that if you create the input file in a directory different from the one where you save your program, you need to specify the path as well, when specifying the filename, e.g.
   ```
   C:\unisa\cos112\datafiles\orders.dat
   ```

   When using a file, it is essential to test whether the file is available. The following code segment tests whether the first input file is available before attempting to extract from it:
   ```
   if (!infile)
   {
   ```

```
    cout << "Cannot open file " << inName << " Aborting!" <<
endl;
    exit(1);
}
```

In the above code segment the program is terminated immediately when `exit(1)` is invoked. We need to add

```
#include <cstdlib>
```

to our program i.e. the header file for the library that contains the `exit()` function.

The initial code in a program where we use files, will usually look like this:

```
#include <cstdlib> //header file for the library that contains the
                   // exit() function
#include <fstream> //directive to use for file processing
   ifstream infile; //internal file name
   string inName; //external file name to be obtained from user
   cout << endl << "Enter the input file name. " << endl;
   cin >> inName;
   infile.open(inName.c_str()); //associate external file name with
                                //internal file name
   if (infile.fail()) //test whether file is available
   {
       cout << "Cannot open file "
            << inName << " Aborting!" << endl;
       exit(1);
   }
```

4. In C++ files can be handled either as a data file, i.e. containing data values, or as a text file. A text file is processed character by character, while the values in a data file is processed value by value.
   A text file is typically processed in the following way:

```
char ch;
infile.get(ch);
while (!infile.eof())
{
      //process ch
      infile.get(ch);
}
```

   Compare this to the typical way to process a data file (i.e. containing values that should be processed as e.g. `int`, `string` or `float`), e.g. a file containing one `int` followed by a `string` on each line:

```
int value;
string name;
while (infile >> value >> name)
{
//process value and name
}
```

NB: There is no need to read all the data values on one line into a string variable and then extract the data values one by one from the string.

5. The final step is to close the output file:

```
infile.close();
```

6. Note that when file streams are passed as arguments to a function, all the input- and output-file streams need to be reference parameters for the function, since they will all be modified, as can be seen from the prototype given below. The file streams should be passed by reference rather than by value since the internal state of a file stream object may change with an open operation even if the contents of the file have not changed. When you declare a reference parameter, the function call will pass the *memory address* of the actual parameter, instead of copying the parameter value into the formal parameter.

```
void process(ifstream& infile, ofstream& outfile)
```

7. Sample program outline:

```
#include <iostream> // for screen/keyboard i/o
#include <cstdlib> //header file for the library that contains the
                   // exit() function
#include <fstream> //directive to use for file processing
int main()
{
   ifstream infile; //internal file name
   string inName; //external file name to be obtained from user
   cout << endl << "Enter the input file name. " << endl;
   cin >> inName;
   infile.open(inName.c_str()); //associate external file name with
                                //internal file name
   if (infile.fail()) //test whether file is available
   {
       cout << "Cannot open file "
            << inName << " Aborting!" << endl;
       exit(1);
   }
   int value;
   string name;
   while (infile >> value >> name) //processing a data file
   {
   //process value and name
   }
   infile.close();
   return 0;
}
```

8. Please see section 6.1 in Savitch for more info.