CS162 – Programming Techniques

Lab 10
Binary file IO

Cảm ơn thầy Trần Duy Quang đã cung cấp template cho môn học

epartment of Software Engineering-FIT-VNU-HCMUS

1

Notes

Create a single solution/folder to store your source code in a week.

Then, create a project/sub-folder to store your source code of each assignment.

The source code in an assignment should have at least 3 files:

- A header file (.h): struct definition, function prototypes/definition.
- A source file (.cpp): function implementation.
- Another source file (.cpp): named YourID_Ex01.cpp, main function. Replace 01 by id of an assignment.

Make sure your source code was built correctly. Use many test cases to check your code before submitting to Moodle.

Name of your submission, for example: **18125001_W01_07.zip**

2

Content

In this lab, we will review the following topics:

- How to read / write a binary file?

3 Assignments

A: 1 problems / assignments.

H: 5 problems / assignments.

3.1 Integer Array – Load & Save

Write a small program with 2 features (the program should create menu options to allow user to choose one of them).

1. User enters an array of integers from keyboard. The program will save the array to a binary file.
2. User loads an array of integers from a binary file. The program will find the median and output to console.

Format of input.bin:

- 1st number: n, number of elements in the array.
- Next n numbers: elements in the array.

3.2 Date Array – Load & Save

Write a small program with 2 features (the program should create menu options to allow user to choose one of them).

3. User enters an array of dates from keyboard. The program will save the array to a binary file.
4. User loads an array of dates from a binary file. The program will find the newest date and output to console.

Format of input.bin:

- 1st number: n, number of elements in the array.
- Next 3 * n numbers: elements in the array.

3.3 Copy file

Use command prompt argument and binary file IO techniques to create a MYCOPYFILE program.

User will enter:

```
MYCOPYFILE -s path_of_source_file -d path_of_destination
```

The destination file will have same name as the source file.

For example:

```
MYCOPYFILE -s D:/film.mkv -d D:/Level1/Level2/Level3
```

3.4 Split file

Use command prompt argument and binary file IO techniques to create a MYSPLITFILE program.

User will enter:

```
MYSPLITFILE -s path_of_source_file -d path_of_destination -numpart x
```

x is a positive integer number, number of splitted parts.

OR

```
MYSPLITFILE -s path_of_source_file -d path_of_destination -sizeapart x
```

y is a positive integer number (in bytes), size of a splitted part.

For example:

```
MYSPLITFILE -s D:/film.mkv -d D:/Level1/Level2/Level3 -numpart 3
```

OR

```
MYSPLITFILE -s D:/film.mkv -d D:/Level1/Level2/Level3 -sizeapart 1000000
```

Names of splitted parts are: film.mkv.part01, film.mkv.part02, film.mkv.part03…

3.5 Merge file

Use command prompt argument and binary file IO techniques to create a MYMERGEFILE program.

User will enter:

```
MYMERGEFILE -s path_of_part01 -d path_of_destination
```

The program will find all *.part01, *.part02, *.part03… files and merge into a single file.

The program will print errors if there is any part does not exist.

For example:

```
MYMERGEFILE -s D:/Level1/Level2/Level3/film.mkv.part01 -d D:/NewLevel
```