Even Semester (2021)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Group Work****)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Felix**  **Erdiwa**  **Christian Renato** | | **Student ID Number**  **2101693851**  **2101725172** | |
| 1.  2. **Wirengjurit**  3. **Sutandi** | | **Anggara** |  | |
|  |  |
| **Course Code** | **: COMP6048** |  |  | | **Course Name** | | **: Data Structures** | |
| **Class** | **:B2AC**  **L2BC-LEC** |  |  | | **Name of Lecturer(s)** | | **:** 1. Maria Seraphina Astriani | |
|  |  |  |  | |  | | 2. Raymond Kosala  3. Tri Asih Budiono | |
| **Major** | **: CS** |  |  | |  | |  | |
| **Title of Assignment**  (if any) | : Student Archieve Index | |  |  | |  | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  |  | |  | |  | |
| **Due Date** | **: 30-5-2018** |  |  | | **Submission Date** | | **: 30-5-2018** | |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Felix Anggara) (Erdiwa Wirengjurit) (Christian Renato Sutandi)

1. Felix Anggara

2. Erdiwa Wirengjurit

3. Christian Renato Sutandi

**Data Structure**

**Final Project**

**Given Name :** 1. Erdiwa **Surname :** 1. Wirengjurit

2. Felix 2. Anggara

3. Renato 3. Sutandi

**Major :** Computer Science **Student ID :** 1. 2101725172

2. 2101693851

3.

**Batch :** 2021 **Class : B2AC, L2BC-LEC**

**Course Code :** COMP6048 **Lecturers :** Maria Seraphina Astriani

Raymond Kosala Tri Asih Budiono

**Project Name :** Student Archieve Index **Built in :** C++

**Project Description :** A program to manage and store student personal information.

**Purpose :** The purpose of the project is to create a barebone program to manage and store personal information of the students.

**Compiler :** GCC MinGW-w64 (C++17)

**Statement :**

BINUS University strictly prohibit any form of cheating, including direct or indirect cheating, plagiarism, collusion, bribery, etc. as the violation of this prohibition will result in severe punishments, including loss/drop of marks, fail in related course, and expulsion from the university. By signing this assignment at the bottom of this page, I declare that:

1. All the work included in this project is my own work and doesn’t plagiarize from any of other people work
2. All the work included in this project is my own idea, not stealing other people idea.
3. I promise that all the works won’t resubmit this project for another assignment

Signature:

Erdiwa Wirengjurit Felix Anggara Renato Sutandi

1. **Purpose of the program**

The purpose of this program is to create a simple system to manage data of student personal information, either active or no longer active. The program will manage a bunch of .txt data that contains student personal information. Currently the program only supports the basic information except birth date and faculty. But, it shows the basic example of the data management in C++ with the availability of saving the data into the computer that can be re accessed during the program second run.

1. **Specification**

The program using hash table, maps, and queue. Hash table is used to store full data and for single data deep search using keyword (ID Name) due to fast if only search for single data set, maps to store names and id name for search (due to being easier to get data from iteration than using hash table), and queue for storing names of .txt that are scanned inside the Data folder (I suggest NOT to modify .txt file unless one knows what are the uses of these lines). There will be interfaces that using arrays as indicator marks, integers as indicator values. And the interface uses the lowercase w and lowercase s to navigate, also ENTER to select.

1. **Used Library**
2. **Standard C++ Headers**

#include<iostream>

#include<conio.h>

#include<cstdlib>

#include<iomanip>

#include<map>

#include <queue>

#include<utility>

#include <dirent.h>

#include<fstream>

#include<sstream>

#include<cctype>

#include <algorithm>

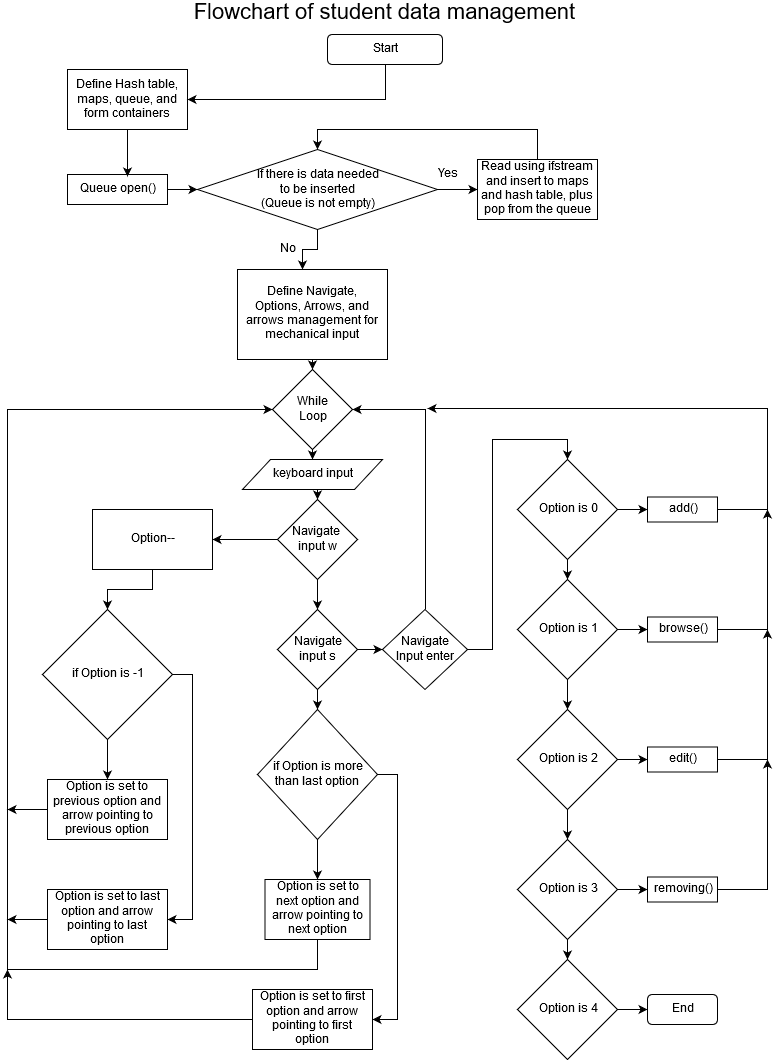
1. **Custom Made Headers**

#include "bstemergency.h"

#include "hashing.h"

#include "form.h"

1. **Flow Chart Diagram**



1. **Custom Classes and Functions Description**
2. **Class Identity**

Identity is the form class that contain the data that will be stored inside the hash table. It is the first foundation of data management, the data itself.

1. **Class AVLnode**

AVLnode is the node of the AVL tree binary tree that will store the form instance and the one of the chains of the tree.

1. **Class AVLtree**

AVLtree is the class of AVL tree binary tree that manage the nodes and the form instances that will be stored or deleted.

1. **Class Hash**

Hash is the hash table that will store the addresses of AVLtree and manage which tree that will store Identity instance using hash function that will pointing to the array address that contains the address that will be the destination that the Identity instance will go.

1. **Function IsFloat()**

This function will check whether the value is float or not.

1. **Function IsInteger()**

This function will check whether the value is integer or not. But, it only supports positive digits.

1. **Function add()**

This function will run the adding interfaces and adding procedural mechanism to add new data.

1. **Function edit()**

This function will run the editing interfaces and editing procedural mechanism to edit data.

1. **Function removing()**

This function will run the removing interfaces and removing procedural mechanism to remove data.

1. **Function browse()**

This function will run the searching interfaces and searching procedural mechanism to search data.

1. **Analysis**

Hash table is indeed fast for data retrieval, storing, and deleting a single portion of data. But, it is inefficient and slow for multiple data retrieval. Even though the complexity of hash table is O((1 + how many times of hashing) \* chaining complexity(in smaller size than n data)), it is inefficient to retrieve multiple data from it. In the other hand, Iterator is slow for retrieving or delete individual data. But for multiple data retrieval, it is the best one for it, because though the complexity is O(n), it doesn’t need to retrieve data from the beginning due to it connected to the next value (A->B->C->D->…->n data).

For the folder scanning, queue is more favourable due to its definition as FIFO (first in, first out). It is more organized if using queue and files in folder is usually have already been sorted by name. So, it can gain the advantages of ‘no need to sort’ from the data that already being sorted. Also, it is more memory wise compared to linked list.

1. **Additional Information**

There are several things that should keep in when using this code. Be careful with the Hash class. ~~It is quite unstable that it really needs readjustment for the hash function output. There are chances that the outputs after modulo processes if being outputted raw without value readjustment can cause crash. To handle this, for the raw hash value that being problematically cause crash needs to be incremented by one or something else~~. It is fixed, but there is still possibility that it may cause crash again for certain value.

For the codes, it can be pulled from this link:

https://github.com/HelixAngler/FelixAnggara\_DS2017\_FinalProject.git

1. **Sources**

<https://rosettacode.org/wiki/AVL_tree>

<http://forum.codecall.net/topic/60157-read-all-files-in-a-folder/>

<https://stackoverflow.com/questions/447206/c-isfloat-function>