OOP Lab Assignment 5

Jaynam Modi. PG-43. G3. Sep 09, 2020

- Problem Statement :

A shop maintains an inventory of items. It stores information of items like Item_Code, Item_Name, Quantity and Cost in a data file. Whenever Customer wants to buy an item, sales person inputs the Item_Code and/or Item_Name and the system searches in a file and displays whether it is available or not otherwise an appropriate message is displayed. If it is, then the system displays the item details and request for the quantity of items required. If the requested quantity of items are available, the total cost of items is displayed; otherwise the message is displayed as required items not in stock. After purchasing an item, system updates the file.

Design a system using a class called Items with suitable data members and member functions. Implement C++ program for the inventory system that will create a data file containing the Record of Items in the following form:

Item_Code	Item_name	Quantity	Cost in Rs.
3	Pens	24	10
17	Notebooks	46	14.99

- Data Members :

1. Item_Code
2. Item_Name
3. Quantity
4. Cost

- Member Function :

1. Create file and store Record of Items
2. Search an Item in the file by Item_Code or Item_Name
3. Arrange the Items by Item_Code or Item_Name
4. Update the file

- Objectives :

1. To learn the concepts of file handling in C++. To know to read and write text in the file.

2. To learn the file streams, file pointers and input and output operations on file.

3. To learn update operations on file.

- Theory:

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

- ofstream: Stream class to write on files
- ifstream: Stream class to read from files
- fstream: Stream class to both read and write from/to files.

All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files.

C++ provides us with the following operations in File Handling:

- Creating a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()

Every file maintains two pointers called get_pointer (in input mode file) and put_pointer (in output mode file) which tells the current position in the file where reading or writing will takes place.

These pointers help attain random access in file. That means moving directly to any location in the file instead of moving through it sequentially.

- Algorithm / Class Diagram / Implementation :

1. START.
2. Create the object of ofstream class.

3. Open the data file containing item information using ofstream class object.

4. Write the item information (Item_Code, Item_Name, Quantity and Cost) in a data file using output functions.

5. Close the file.

6. Create the object of ifstream class.

7. Open the data file using that object and read the records in a buffer from the file till eof().

8. Ask the user to search a record from a data file by Item_Code or Item_Name.

9. Input Item_Code or Item_Name from user to search Record.

10. Compare User's Item_Code or Item_Name with Records in a data file which is read in a buffer.

11. If Matching found then display the complete records of Item (Item_Code, Item_Name, Quantity and Cost).

12. Else Display Message that Record not found.

13. Ask the user to input quantity of required items.

14. Compare User's quantity with available quantity.

15. If the requested quantity of items are available, the total cost of items is displayed and update the records of items after purchase.

16. Else display the message as required items not in stock.

17. Arrange the Records in a file by Item_Code or Item_Name

18. Close the file.

19. STOP

- Platform :

  - 64-bit Open source Linux or its dderivatives.

  - Open Source C++ Programming tool like G++/Eclipse Editor.

- Input :

1. Item information (Item_Code, Item_Name, Quantity and Cost) to store in a data file.
2. Item_Code/ Item_Name to search a record in the file.
3. Quantity of required items to be purchased.

- Output :

1. Item information (Item_Code, Item_Name, Quantity and Cost)
2. Cost of Item to be purchased.
3. Updated Records in the File.
4. Sorted Records in the File.

- Conclusion :

Thus, implemented the Inventory System using File Handling concepts. This System is able to search the record by an Item_code and Item_Name. Records in the system are updated after purchase. Also All Records are arranged by Item_code or Item_Name.

- FAQs :

1. Which are the classes are used in file handling?

> In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

2. Explain the file pointers and seekg() and tellg() functions?

> Every file maintains two pointers called get_pointer (in input mode file) and put_pointer (in output mode file) which tells the current position in the file where reading or writing will takes place.

seekg() is a function in the iostream library that allows us to seek an arbitrary position in a file. It is mainly used to set the position of the next character to be extracted from the input stream from a given file in C++ file handling.

The tellg() function is used with input streams, and returns the current "get" position of the pointer in the stream. It has no parameters and returns a value of the member type pos_type, which is an integer data type representing the current position of the get

stream pointer.

3. Which are the functions used in read operation?

> open( ), read( ) and close( ) functions are used in read operation.

4. Write the File Opening Modes.

> There are the following File Opening modes :

ios_base::app - Seek to end-of-file before each write.

ios_base::ate - Seek to end-of-file immediately after opening the file, if it exists.

ios_base::binary - Open file in binary mode (alternative is text mode).

ios_base::in - Open file for input (implied for istream).

ios_base::out - Open file for output (implied for ostream).

ios_base::trunc - Truncate file, if it exists (default for ostream).

5. What is Eof()?

> C++ provides a special function, eof(), that returns nonzero (meaning TRUE) when there are no more data to be read from an input file stream, and zero (meaning FALSE) otherwise.