COS 2021 – Fundamental Data Structures – Spring 2024

Course Project – Library Management System

Documentation

**Task.cpp**

* #include <iostream> - calling exceptions and console output
* #include <string> - using all properties of the type string
* #include <sstream> - allows us to read and write on files
* #include <vector> - using vectors
* #include <algorithm> - using the sort function
* #include <fstream> - includes the standard Input/Output File Stream Library
* #include <iomanip> - let’s convert double to string with precision to the second post-decimal number
* #include <regex> - let us check the format of the dates. Whether they are correct or not.
* #include <chrono> - access to time
* #include <thread> - let us use time functions
* #include “KMP.cpp” – allows us to use the KMP algorithm class in the class file “KMP.cpp”

Data collections

using namespace std; - used for calling exceptions, checking whether there is a given ID in the IDs vector, console output and to save time to not write everywhere ” std::” where it is required

enum Condition { New, Good, Average, Bad, OnTheVergeOfCollapsing, Stolen}; - creating a enum for physical condition of physical items

enum Subscription { Free, Basic, Standard, Premium } - creating an enum for the subscription plan of a software bought by the library to be used by people

vector<string> IDs; - used to check whether there is already an item with a specific ID before creating a new one with the same ID

regex dateFormat – it checks whether the date

* is a leap year
* is in the format MM/DD/YYYY
* does not contain any letters or other characters

Functions used for calculations in the classes’ methods

bool isValidDateFormat(string date)

* checks if the format of the date is correct (MM/DD/YYYY) by using regex

bool isValidDateFormatForItem(string date, string dateType, string id)

* checks if the format of the date is correct (MM/DD/YYYY) by using regex
* if it is not, it displays personal message for the item (type and ID) and the specific date.

void separateDate(string date, int& month, int& day, int& year) – separates a string input of date (MM/DD/YYY) into month (MM), day (DD) and year (YYYY) as those variables are pointers and are changed in correspondence to the date

void checkYear(string y, string date, string itemType, string ID) – checks whether a given year is correct through regex

* A correct year is one between the years 100 and 2024.
  + It cannot be a year in the future (2025-etc.).
  + There can be books even from the year 100. Even if it is unlikely.
* If it is not in the right format, it gives error message on the command window.
* It also checks the year with the items’ release date’s year.
  + That way we avoid data confusion.
* If any error message is show, it will be personal for the item (type and ID).

int daysInMonth(int month, int year) – returns how many days are in the given month (January - December)

* If the year is a leap one, February shall return 29. If not, 28.

int daysBetweenTwoDates(string oldDate, string laterDate) – calculating how many days are there between two dates

Condition ConditionFromString(string conditionStr) – returning a Condition based on a string input of the requested condition.

* If the requested type of condition does not exist, we throw an invalid argument exception: *"There is no such type of condition for the physical item!"*
* Used for generating physical items in main function – Book, Magazine and DVD

Classes

class Item – an abstract class

* **protected** members – the data slots that will be inherited by the descendants of Item – Physical, Electronical, Book, Ebook, Magazine, DVD and Software
  + string name – the name/title of the item
  + string type – the type of item (like the class) – Book, Ebook, Magazine, DVD and Software
  + string ID – the unique ID of the item so that it can be separated from other items (with the same name)
  + long string description – what it contains
  + string releaseDate – the date it became available to the public
  + string dateOfReservation – when a person got it from the library
  + string dateOfReturn – when the customer should return it
  + bool physical – whether the item is physical
    - For Physical items (Book, Magazine and DVD) it will be true.
    - For Electronic items (Ebook and Software) it will be false.
    - Later during the inheritance it will be changed in correspondence to the item type.
  + void checkID(string id) – checks whether there is already an item with a specific ID in the vector IDs before creating this new one with the same ID
    - If there is, it throws an invalid argument exception “There is already an item with this ID.”.
    - If there is not, it proceeds to create the new item.

It is called in every constructor of the class Item.

* **public** members
  + Constructors (using the const keyword to prevent the new data from modification)
    - Standard (empty) constructor with default data for all data slots
      * string name - empty
      * string type – “Item”
      * string ID - empty
      * long string description - empty
      * string releaseDate - empty
      * string dateOfReservation - empty
      * string dateOfReturn - empty
      * bool physical - true
    - Argument constructor – used to create an item with previously set data (except type and physical variables).
    - Copy constructor – used to create a copy of another item in the currently generated new item
      * All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
  + Methods (using the const keyword to prevent data modification)
    - Getters for the following data:
      * string name - GetName()
      * string ID – GetID()
      * string type – GetItemType()
      * string description – GetDescription()
      * string releaseDate – GetReleaseDate()
      * string dateOfReservation – GetDateOfReservation()
      * string dateOfReturn – GetDateOfReturn()
      * bool physical – IsItPhysicalOrElectronic()
      * There are no setters as we do not want the data to be modified.
      * All of them have **const** so that the data cannot be modified.
    - bool isLate(string currentDate) – checks whether the item is late for the given current date
    - **virtual** string CalculateLateFee(string) = 0;
      * virtual – allows us to get an access to Item’s method in the driver program
      * The **= 0** at the end of the function declaration signifies that the functions are pure virtual, meaning they have no implementation in the Item class itself. Pure virtual functions must be overridden by any concrete derived class before that class can be instantiated.
      * No const

class Physical – a child class of *Item* and parental class of *Book*, *Magazine* and *DVD*

* **protected** members – the data slots that will be inherited by *Book, Magazine* and *DVD*
  + Condition condition – the condition state of the physical item
* **public** members
  + Constructors (using the const keyword to prevent the new data from modification)
    - Standard (empty) constructor with default data for all data slots
      * It also inherits the Item empty constructor as we don’t need to present default vlues of the other data slots.
      * string type – “Physical”
      * Condition condition – New
      * bool physical - true
    - Argument constructor – used to create a physical item with previously set data.
      * It also inherits the Item argument constructor as we don’t need to present values of the other data slots.
    - Copy constructor – used to create a copy of another physical item in the currently generated new physical item
      * It also inherits the Item copy constructor as we don’t need to present values of the other data slots.
      * All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
    - It calls upon the parent’s assignment operator= so that we don’t have to repeat every data slot from the parent.
  + Methods
    - Getter for the string version of condition - GetPhysicalCondition()
      * using the const keyword to prevent data modification
    - void DegradeCondition
      * without Condition value - degrades the current physical condition by one
      * with Condition value – degrades the current physical condition to a specific
        + If a value for a better condition is given, there will be an invalid\_argument error.
        + You can’t upgrade the condition.
    - void GetsStolen() – whether the item is already stolen or not, it is classified as such
    - bool isStolen() – shows whether the item is stolen or still in the library
      * using the const keyword to prevent data modification
    - string CalculateLateFee(string currentDate)
      * The virtual method from the abstract class.
      * In this case, the fee is calculated by the number of days for the reservation. The more time it was in a person, the bigger the fee will get if they are late.
      * Depending on how long the item has not been return for, the closer the tax rate to the initial will be.
      * In the end, it returns the fee in a string with percentages.
        + If the item is not late, it will show a message.

class Electronical – a child class of *Item* and parental class of *Ebook* and *Software*

* **protected** members – the data slots that will be inherited by *Ebook* and *Software*
  + string expirationDate – the date after which the electronic product will no longer work on devices.
* **public** members
  + Constructors (using the const keyword to prevent the new data from modification)
    - Standard (empty) constructor with default data for all data slots
      * It also inherits the Item empty constructor as we don’t need to present default values of the other data slots.
      * string type – “Electronic”
      * string expirationDate – “”
      * bool physical - false
    - Argument constructor – used to create an electronic item with previously set data.
      * It also inherits the Item argument constructor as we don’t need to present values of the other data slots.
    - Copy constructor – used to create a copy of another electronic item in the currently generated new electronic item
      * It also inherits the Item copy constructor as we don’t need to present values of the other data slots.
      * All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
    - It calls upon the parent’s assignment operator= so that we don’t have to repeat every data slot from the parent.
  + Methods
    - Getter for the expirationDate – GetCodeExpirationDate()
      * using the const keyword to prevent data modification
    - string CalculateLateFee(string currentDate)
      * The virtual method from the abstract class.
      * In this case, the fee is calculated by the number of days for the reservation. The more time it was in a person, the bigger the fee will get if they are late.
      * If the person releases the item within two days, the fee will be nullified.
      * In the end, it returns the fee in a string with percentages.
        + If the item is not late, it will show a message.

class Book – a child of Physical

* **private** members – the data slots
  + string author – the person who wrote it
  + string genre – the type/genre of book
  + int year – the year the book was published
* **public** members
  + Standard (empty) constructor with default data for all data slots
    - It also inherits the Physical empty constructor as we don’t need to present default values of the other data slots.
    - string type – “Book”
  + Argument constructor – used to create a with previously set data.
    - It also inherits the Physical argument constructor as we don’t need to present values of the other data slots.
  + Copy constructor – used to create a copy of another in the currently generated new book
    - It also inherits the copy constructor as we don’t need to present values of the other data slots.
    - All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
    - It calls upon the parent’s assignment operator= so that we don’t have to repeat every data slot from the parent.
  + Methods (using the const keyword to prevent data modification)
    - Getters for the following data:
      * string author – GetAuthor()
      * string genre – GetGenre()
      * int year – GetYear()

class Ebook

* **private** members – the data slots
  + string author – the person who wrote it
  + string genre – the type/genre of book
  + int year – the year the book was published
* **public** members
  + Standard (empty) constructor with default data for all data slots
    - It also inherits the Electronical empty constructor as we don’t need to present default values of the other data slots.
    - string type – “Ebook”
  + Argument constructor – used to create a with previously set data.
    - It also inherits the Electronical argument constructor as we don’t need to present values of the other data slots.
  + Copy constructor – used to create a copy of another in the currently generated new ebook
    - It also inherits the copy constructor as we don’t need to present values of the other data slots.
    - All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
    - It calls upon the parent’s assignment operator= so that we don’t have to repeat every data slot from the parent.
  + Methods (using the const keyword to prevent data modification)
    - Getters for the following data:
      * string author – GetAuthor()
      * string genre – GetGenre()
      * int year – GetYear()

class Magazine

* **private** members – the data slots
  + string publisher – the company that published the magazine
  + string issueNumber – the number of the current issue of the magazine
* **public** members
  + Standard (empty) constructor with default data for all data slots
    - It also inherits the Physical empty constructor as we don’t need to present default values of the other data slots.
    - string type – “Magazine”
  + Argument constructor – used to create a with previously set data.
    - It also inherits the Physical argument constructor as we don’t need to present values of the other data slots.
  + Copy constructor – used to create a copy of another in the currently generated new magazine
    - It also inherits the copy constructor as we don’t need to present values of the other data slots.
    - All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
    - It calls upon the parent’s assignment operator= so that we don’t have to repeat every data slot from the parent.
  + Methods (using the const keyword to prevent data modification)
    - Getters for the following data:
      * string publisher – GetPublisher()
      * string issueNumber – GetIssueNumber()

class DVD

* **private** members – the data slots
  + string director – the person who directed the movie
  + string genre the genre/type of the movie
  + int year – the year the movie was released
  + double duration – the time in minutes
* **public** members
  + Standard (empty) constructor with default data for all data slots
    - It also inherits the Physical empty constructor as we don’t need to present default values of the other data slots.
    - string type – “DVD”
    - bool physical - false
  + Argument constructor – used to create a with previously set data.
    - It also inherits the Physical argument constructor as we don’t need to present values of the other data slots.
  + Copy constructor – used to create a copy of another in the currently generated new
    - It also inherits the copy constructor as we don’t need to present values of the other data slots.
    - All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
  + Methods (using the const keyword to prevent data modification)
    - Getters for the following data:
      * string director – GetDirector()
      * string genre – GetYear()
      * int year – GetYear()
      * double duration – GetDuration()

class Software

* **private** members – the data slots
  + string username – the name under which the profile for the software works on
  + Subscription subscriptionPlan – the type of plan that it is paid for
  + double usageTime – the time spent on the program for the user
* **public** members
  + Standard (empty) constructor with default data for all data slots
    - It also inherits the Electronical empty constructor as we don’t need to present default values of the other data slots.
    - string type – “Software”
    - bool physical - false
  + Argument constructor – used to create a with previously set data.
    - It also inherits the Electronical argument constructor as we don’t need to present values of the other data slots.
  + Copy constructor – used to create a copy of another in the currently generated new
    - It also inherits the copy constructor as we don’t need to present values of the other data slots.
    - All of the data is being replicated, including the ID.
  + Assignment operator= - it does the same operation just like the copy constructor
  + Methods (using the const keyword to prevent data modification)
    - Getters for the following data:
      * string username – GetUsername()
      * string version of subscriptionPlan – GetSubscriptionPlan()
      * double usageTime – GetUsageTime()

Main functions for the program which are the core of the functionalities in the library management system

void KMPcommand(vector<Item\*>& Items, string searchString)

* The function calls the KMP class.
* It uses the KMP sorting algorithm to find specific Items, no matter the type, with a specific search string in their names.

void display(vector<Item\*>& Items, string type)

* The function displays a vector of items.
* Depending on the type of items that is required, it will display only those items.
* It can also display a specific item if there is such item with an ID.

void degradingCondition(vector<Item\*>& Items, string target, string newCondition)

* The program that uses DegradeCondition() in all physical items and changes them in correspondence to the new condition.
* If newCondition is “.”, then it degrades it by 1 version worse.

void LateFeeCalculating(vector<Item\*>& Items, string target, string date)

* Calculating the fee for later returned item.
* It can look for all items or a specific one through its ID.

void ItemExpirationDate(vector<Item\*>& ElectronicItems, string target)

* It shows the expiration date of a specific item (through ID target) or all of them.

void Stealing(vector<Item\*>& StolenItems, string target)

* Changes the status of either a specific item (through ID target) or all of them to “Stolen”.

Main program

int main()

After 3. and 4. the command window will be cleaned so that not too much data is display at once and the user has a better experience and avoid confusion.

1. When the user enters our program, they will be given a message about how to operate with it.
2. They will be given a choice what file to use for a data input in the specific format that is written in “items.txt”.
   1. If they decide to use another file, they will be given a prompt to enter its name.
   2. They will be given another chance to change the file until they finally decide for real.
   3. If the new chosen file doesn’t work, the program will throw a message that such file does not exist and will be terminated.
3. The program will start reading the input file with the data for all items.
   1. Each row will present a new item from a specific type (book, ebook, magazine, software and DVD).
   2. They will be put in the correct type of class and then put in the vector collection.
      1. If the type given for the new item does not exist, nothing will be created.
4. After the user will again be given two choices. Whether they want to input commands through prematurely prepared file named “commands.txt” or through the command window.
   1. In either case, the commands will be read in the same way.
   2. There the user will have the ability to access the several command that we have implemented for data modification in our items in the vector.
      1. CheckIfLate
      2. CalculateLateFee
      3. KMP
      4. GetCodeExpirationDate
      5. GetsStolen
      6. DegradeCondition
      7. Show commands (only for command window)
      8. Show/Display
      9. End program
5. In the end, the user will be asked for one last command – whether or not they want the data for all of the items in the library to be exported in another file named “OutputData.txt”.
   1. The format of data export will be similar to the one of the display function for the command window.

**KMP.cpp**

* #include <iostream> - calling exceptions and console output
* #include <string> - using all properties of the type string
* #include <vector> - using vectors

using namespace std; - used for calling exceptions, checking whether there is a given GUID in the GUIDs vector, console output and to save time to not write everywhere ”std::” where it is required

class KMP

* **private** members
  + vector<int> computeLPS(const string& pattern)
    - This function computes the Longest Prefix Suffix (LPS) array for a given pattern string. The LPS array is used in the Knuth-Morris-Pratt (KMP) algorithm to efficiently search for occurrences of the pattern within a text. The LPS array stores the length of the longest proper prefix which is also a suffix for each position in the pattern.
* **public** members
  + int SearchString(const string& text, const string& pattern)
    - The function that is called in the main method.
    - This function implements the Knuth-Morris-Pratt (KMP) algorithm for string pattern matching. Given a text string and a pattern string, it searches for all occurrences of the pattern within the text. It returns a vector of pairs, where each pair contains the starting index of an occurrence of the pattern in the text and the pattern itself. This algorithm utilizes the LPS array computed for the pattern string to efficiently skip comparisons while searching for matches in the text.

*The program is tested with different values of different types of value. With the presented one, it should not give any exceptions. If it is changed so that exceptions are required, it will do so.*