**OOP345**

**Assignment Solutions**

**Part A**

**Program # 1**

You are to create a database of books that are stored using a vector. Keep track of the author, title and publication date of each book. Your program should have a main menu that allows the user to select from the following:

1. Add a book’s author, title, and date
2. Print an alphabetical list of the books sorted by author
3. Quit

You must use a class to hold the data for each book. This class must hold three string fields, one to hold the author’s name, one for the publication date, and another to hold the book’s title. Store the entire database of books in a vector where each vector element is a book class object.

To sort the data, use the generic sort function from the <algorithm> library. Note that this requires you to define the <operator to compare two objects of type book so that the author field from the two books are compared.

***Note: you are required to create three files, Book.h, Book.cpp and TestProgram.cpp.***

A sample of the input/output behavior might look as follows. Your I/O need not look identical, this is just to give you an idea of the functionality.

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 1**

Enter title:

**Problem solving with c++**

Enter author:

**Savitch, walter**

Enter date:

**2006**

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 2**

The books entered so far, sorted alphabetically by author are:

Savitch, walter. Problem Solving with C++. 2006.

Sturgeon, Theodore, More Than Human. 1953.

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 1**

Enter title:

**At Home in the Universe**

Enter author:

**Kauffman**

Enter date:

**1996**

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 2**

The books entered so far, sorted alphatetically by artist are:

Kauffman, At Home in the Universe, 1996.

Savitch, walter. Problem Solving with C++. 2006.

Sturgeon, Theodore, More Than Human. 1953.

**Book.h**

#ifndef BOOK\_H

#define BOOK\_H

#include <string>

using namespace std;

class Book

{

public:

Book();

Book(string new\_author, string new\_title, string new\_date);

void setData(string new\_author, string new\_title, string new\_date);

string getAuthor() const;

string getTitle() const;

string getDate() const;

friend bool operator< (const Book &book1, const Book &book2);

private:

string author, title, date;

};

#endif

**Book.cpp**

Add the definitions of data members of class Book. The definition of friend function should be (in Book.cpp):

bool operator< (const Book &book1, const Book &book2)

{

return (book1.author < book2.author);

}

**TestProgram.cpp**

Don’t forget to add header files, vector and algorithm along with other required header files.

using namespace std;

void AddNewBook(vector<Book> \*bookdata);

void PrintBooks(vector<Book> &bookdata);

void SortBooks(vector<Book> &bookdata);

void PrintMenu();

int main()

{

vector<Book> bookdata;

//Add your logic to test your class, using functions mentioned above

return 0;

}

**Code For Program #1**

**File – Book.h**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 30-10-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#ifndef BOOK\_H

#define BOOK\_H

#include <string>

#include<vector>

using namespace std;

class Book

{

private:

string author, title, date;

public:

Book();

Book(string new\_author, string new\_title, string new\_date);

void setData(string new\_author, string new\_title, string new\_date);

string getAuthor() const;

string getTitle() const;

string getDate() const;

friend bool operator < (const Book& book1, const Book& book2);

};

#endif

**File – Book.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 30-10-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include "Book.h"

#include <iostream>

#include<algorithm>

#include <vector>

using namespace std;

Book::Book()

{

author = "Not yet entered";

title = "Not yet entered";

date = "Not yet entered";

}

void Book::setData(string new\_author, string new\_title, string new\_date)

{

author = new\_author;

title = new\_title;

date = new\_date;

}

string Book::getAuthor() const

{

return author;

}

string Book::getTitle() const

{

return title;

}

string Book::getDate() const

{

return date;

}

Book::Book(string new\_author, string new\_title, string new\_date)

{

this->author = new\_author;

this->title = new\_title;

this->date = new\_date;

}

bool operator<(const Book& book1, const Book& book2)

{

return (book1.getAuthor() < book2.getAuthor());

}

**File – TestProgram.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 30-10-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include <iostream>

#include <algorithm>

#include "Book.h"

#include <vector>

#include<string>

using namespace std;

void AddNewBook(vector<Book>\* bookdata)

{

string title, author, date;

cout << "Enter title:\n";

cin.ignore();

getline(cin, title);

cout << "Enter author:\n";

getline(cin, author);

cout << "Enter date:\n";

getline(cin, date);

Book newBook(author, title, date);

(\*bookdata).push\_back(newBook);

}

void PrintBooks(vector<Book>& bookdata)

{

cout << "The books entered so far, sorted alphabetically by author are:\n";

for (auto itr = bookdata.begin(); itr != bookdata.end(); itr++)

{

cout << "\t" << itr->getAuthor() << ". " << itr->getTitle() << ". " << itr->getDate() << ".\n";

}

}

void SortBooks(vector<Book>& bookdata)

{

sort(bookdata.begin(), bookdata.end());

}

int PrintMenu()

{

int ch;

cout << "Select from the following choices : \n";

cout << "\t1. Add new book\n"

<< "\t2. Print listing sorted by author\n"

<< "\t3. Quit\n";

cout << "Choice : ";

cin >> ch;

return ch;

}

int main()

{

vector<Book> bookdata;

int choice = PrintMenu();

while (choice != 3)

{

switch (choice)

{

case 1:

AddNewBook(&bookdata);

break;

case 2:

SortBooks(bookdata);

PrintBooks(bookdata);

break;

case 3:

break;

default:

cout << "Make a valid choice!!!!! " << endl;

}

choice = PrintMenu();

}

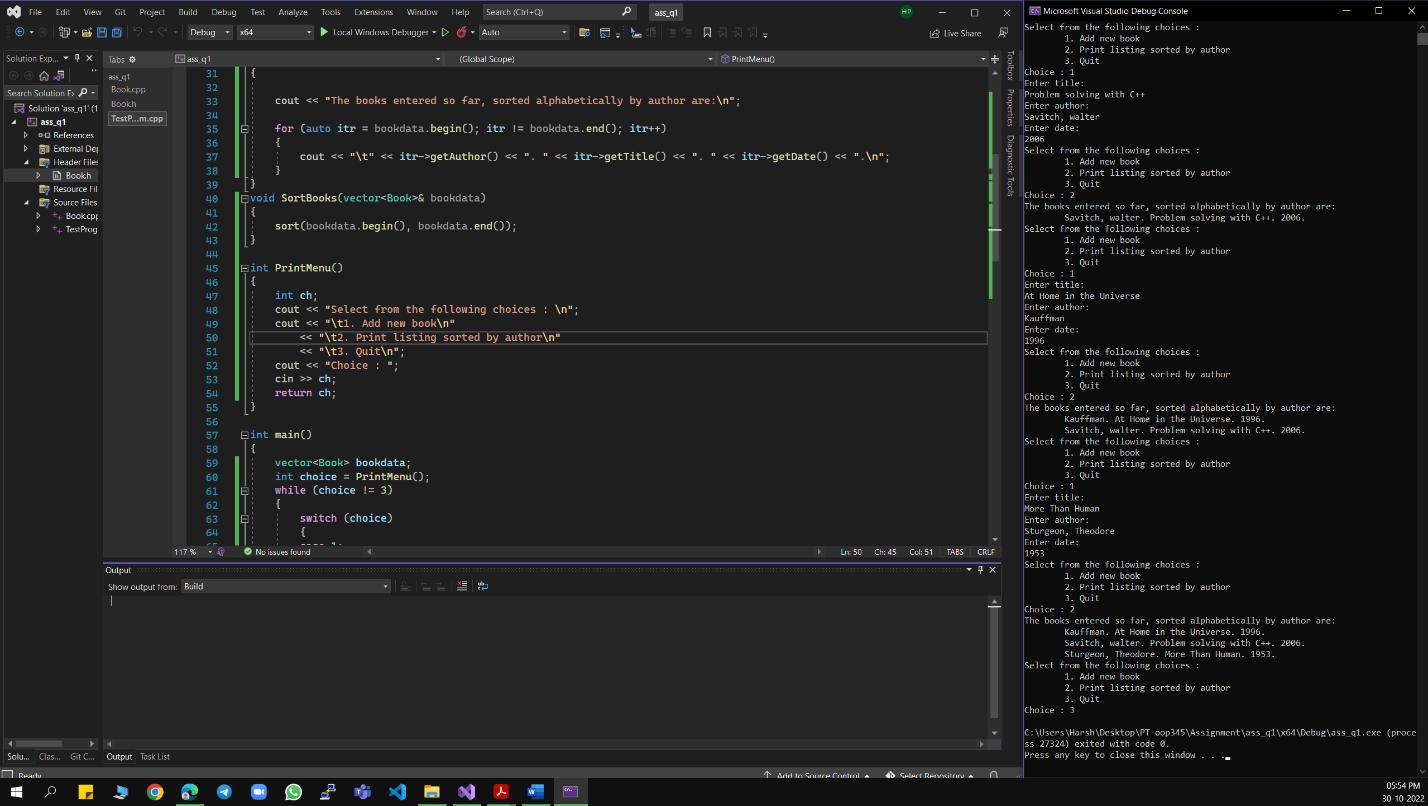
return 0;

}

**Screenshots for program #1 Output:**

**Graphical user interface, text

Description automatically generated**



**Program # 2**

Write a program to implement English Dictionary that contains data members as word, meaning, synonym, antonym and example\_sentence. Use an appropriate STL container and apply the relevant algorithms to maintain the sense of Dictionary.

**Code For Program #2**

**File – Dictionary.h**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 30-10-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#ifndef Dictionary\_H

#define Dictionary\_H

#include <iostream>

#include<string>

#include <iterator>

#include <map>

using namespace std;

class Dictionary

{

private:

string word, meaning, synonym, antonym, example;

public:

Dictionary();

Dictionary(string wrd, string mean, string synm, string atnm, string exm);

string getWord();

string getMeaning();

string getSynonym();

string getAntonym();

string getExample();

void setWord(string wrd);

void setMeaning(string mean);

void setSynonym(string synm);

void setAntonym(string atnm);

void setExample(string exm);

void print();

};

int menu();

void Insert(map<string, Dictionary>& dictionary);

void Delete(map<string, Dictionary>& dictionary);

void Search(map<string, Dictionary> dictionary);

void Update(map<string, Dictionary>& dictionary);

void Display(map<string, Dictionary> dictionary);

#endif

**File – Dictionary.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 30-10-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include "Dictionary.h"

Dictionary::Dictionary()

{

word = "Not yet entered";

meaning = "Not yet entered";

synonym = "Not yet entered";

antonym = "Not yet entered";

example = "Not yet entered";

}

Dictionary::Dictionary(string wrd, string mean, string synm, string atnm, string exm)

{

word = wrd;

meaning = mean;

synonym = synm;

antonym = atnm;

example = exm;

}

//getters

string Dictionary::getWord(){return word;}

string Dictionary::getMeaning(){return meaning;}

string Dictionary::getSynonym(){return synonym;}

string Dictionary::getAntonym(){return antonym;}

string Dictionary::getExample(){return example;}

//setters

void Dictionary::setWord(string wrd){word = wrd;}

void Dictionary::setMeaning(string mean){meaning = mean;}

void Dictionary::setSynonym(string synm){synonym = synm;}

void Dictionary::setAntonym(string atnm){antonym = atnm;}

void Dictionary::setExample(string exm){example = exm;}

//to print the word

void Dictionary::print()

{

cout << getWord() << endl << "Meaning\n " << getMeaning() << "\nSimilar: "<<getSynonym()

<< "\nAntonym: " << getAntonym() << "\nExample\n " << getExample()

<< endl;

}

//menu for insert, delete, search, update and to display a word from the dictionary

int menu()

{

int ch;

cout << "Menu for English Dictionary of words \n";

cout<< "1. To add a word \n"

<<"2. To Delete/erase a word\n"

<<"3. Search a word \n"

<<"4. To Update a word's description\n"

<<"5. To Display a word\n"

<<"6. Exit\n";

cout << "Choice: ";

cin >> ch;

return ch;

}

// insert function asks for word to add a word using insert and takes the description of word and insert into our dictionary

void Insert(map<string, Dictionary>& dictionary)

{

string word, meaning, synonym, antonym, example;

cout << "\nEnter word to add: ";

cin.ignore();

getline(cin, word);

if (dictionary.count(word) == 0)

{

cout << "\nEnter meaning of the word \"" << word << "\" : ";

getline(cin, meaning);

cout << "\nEnter synonym of the word \"" << word << "\" : ";

getline(cin, synonym);

cout << "\nEnter antonym of the word \"" << word << "\" : ";

getline(cin, antonym);

cout << "\nEnter example for the word \"" << word << "\" : ";

getline(cin, example);

Dictionary data(word, meaning, synonym, antonym, example);

dictionary.insert(pair<string, Dictionary>(word, data));

cout << endl << word << " added to dictionary!!!" << endl;

}

else

{

cout << endl << word << " word already exist in the dictionary!!!" << endl;

}

}

//Delete function searched the word we want to delete and then if it founds then it deletes using erase the word otherwise print that no such word found

void Delete(map<string, Dictionary>& dictionary)

{

string word;

cin.ignore();

cout << "\nEnter word to delete: ";

getline(cin, word);

if (dictionary.count(word) == 1)

{

dictionary.erase(word);

cout << "\n" << word << " deleted from dictionary" << endl;

}

else

{

cout << "\n" << word << " not found in dictionary" << endl;

}

}

// Search function searches the word using find method of stl and if not prints not found

void Search(map<string, Dictionary> dictionary)

{

string word;

cin.ignore();

cout << "\nEnter word to search: ";

getline(cin, word);

if (dictionary.count(word) == 1)

{

auto itr = dictionary.find(word);

Dictionary data = itr->second;

cout << "\n";

data.print();

}

else

{

cout << "\n" << word << " not found in dictionary" << endl;

}

}

// update function first searches for the word and then make changes to it and if not found prints not in dictionary

void Update(map<string, Dictionary>& dictionary)

{

string word, meaning, synonym, antonym, example;

cin.ignore();

cout << "\nEnter word for update: ";

getline(cin, word);

if (dictionary.count(word) == 1)

{

auto itr = dictionary.find(word);

Dictionary data = itr->second;

cout << "\nEnter meaning of the word \"" << word << "\" to update: ";

getline(cin, meaning);

cout << "\nEnter synonym of the word \"" << word << "\" to update:";

getline(cin, synonym);

cout << "\nEnter antonym of the word \"" << word << "\" to update: ";

getline(cin, antonym);

cout << "\nEnter example for the word \"" << word << "\" to update:";

getline(cin, example);

itr->second.setMeaning(meaning);

itr->second.setSynonym(synonym);

itr->second.setAntonym(antonym);

itr->second.setExample(example);

cout << endl << word << " successfully updated the dictionary!!!" << endl;

}

else

{

cout << "\n" << word << " not found in dictionary" << endl;

}

}

// display function shows all words in dictionary

void Display(map<string, Dictionary> dictionary)

{

if (dictionary.size() > 0)

{

cout << "\n---Dictionary data---\n";

for (auto itr = dictionary.begin(); itr != dictionary.end(); itr++)

{

Dictionary data = itr->second;

data.print();

cout << endl;

}

}

else

{

cout << "\nDictionary is empty!!" << endl;

}

}

**File – main.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 30-10-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include<iostream>

#include<map>

#include"Dictionary.h"

using namespace std;

int main()

{

map<string, Dictionary> dictionary;

int choice = menu();

while (choice != 6)

{

switch (choice)

{

case 1:

Insert(dictionary);

break;

case 2:

Delete(dictionary);

break;

case 3:

Search(dictionary);

break;

case 4:

Update(dictionary);

break;

case 5:

Display(dictionary);

break;

case 6:

break;

default:

cout << "Not a valid choice!!!!!! " << endl;

}

choice = menu();

}

return 0;

}

**Text

Description automatically generated** **Text

Description automatically generated** Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated

**Part B**

**Program #3**

A frequency table lists words and the number of times each word appears in a text file.  
Write a program that creates a frequency table for a file whose name is entered by the  
user. You can use a map of string-int pairs. You may want to use the C library function  
ispunct() (in header file CTYPE.H) to check for punctuation so you can strip it off the  
end of a word, using the string member function substr(). Also, the tolower() function  
may prove handy for uncapitalizing words.

**File – text.cpp**

This is a testing file.

this file will be used to test the frequency of words written in this file.

**File – FreqMap.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 13-11-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include <iostream>

#include<iomanip>

#include <string.h>

#include <map>

#include <algorithm>

#include <fstream>

using namespace std;

typedef map<string, int> FreqMap;

FreqMap Map;

bool punctChecker(string word)

{

int l = word.length();

if (ispunct(word[l - 1]))

return true;

return false;

}

void freqTable() {

cout << left << setw(10) << "Word" << "frequency" << endl;

for (auto itr : Map)

{

cout << left << setw(10) << itr.first << itr.second << endl;

}

}

int main()

{

string filename, word;

fstream file;

cout << "Enter file name: ";

cin >> filename;

file.open(filename.c\_str());

while (file >> word)

{

if (punctChecker(word))

{

word = word.substr(0, word.length() - 1);

}

transform(word.begin(), word.end(), word.begin(), [](unsigned char c)

{ return tolower(c); });

Map[word]++;

}

cout << "Frequency Table of words in file named: " << filename << "\n"

<< endl;

freqTable();

return 0;

}

Text

Description automatically generated

**Program #4**

Create a single linked list that contains the data (age) of your friends. Perform basic operations including insertion, deletion, searching and display. The insertion operation should only allow a friend’s data to be inserted in sorted order only.

**File – Age.cpp**

#include <iostream>

using namespace std;

struct node

{

int data;

node \*next;

};

class List

{

private:

struct node \*head;

public:

List()

{

head = nullptr;

}

void insertAge()

{

struct node \*ptr, \*temp;

int age;

ptr = (struct node \*)malloc(sizeof(struct node));

cout << "Enter Age to insert: ";

cin >> age;

if (head == NULL || head->data >= ptr->data)

{

ptr->data = age;

ptr->next = NULL;

head = ptr;

cout << "\*\*\*Age inserted\*\*\*\n";

return;

}

ptr->data = age;

temp = head;

while (temp->next != NULL && temp->next->data < ptr->data)

{

temp = temp->next;

}

ptr->next = temp->next;

temp->next = ptr;

cout << "\*\*\*Age inserted\*\*\*" << endl;

}

void deleteAge()

{

struct node \*ptr, \*temp;

if (head == NULL)

{

cout << "list is empty" << endl;

}

else if (head->next == NULL)

{

head = NULL;

free(head);

cout << "\*\*\*Friend's Age Deleted Successfully!\*\*\*" << endl;

}

else

{

ptr = head;

while (ptr->next != NULL)

{

temp = ptr;

ptr = ptr->next;

}

temp->next = NULL;

free(ptr);

cout << "\*\*\*Friend's Age Deleted Successfully!\*\*\*" << endl;

}

}

node \*searchAge(int value)

{

node \*temp = head;

while (temp != NULL)

{

if (temp->data == value)

{

return temp;

}

temp = temp->next;

}

return NULL;

}

void displayAge()

{

struct node \*ptr;

ptr = head;

if (ptr == NULL)

{

cout << "List is Empty" << endl;

}

else

{

cout << "Entered ages: " << endl;

while (ptr != NULL)

{

cout << ptr->data << " ";

ptr = ptr->next;

}

cout << endl;

}

}

};

int main()

{

List L;

int choice = 0, value;

while (choice != 5)

{

cout << "\nProgram to store age of friends" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1. Insert Age\n"

<< "2. Delete Age\n"

<< "3. Display Age list\n"

<< "4. Search Age\n"

<<"5. To Exit" <<endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Enter your Choice: ";

cin >> choice;

switch (choice)

{

case 1:

L.insertAge();

break;

case 2:

L.deleteAge();

break;

case 3:

L.displayAge();

break;

case 4:

cout << "Enter an Integer to search in the list: ";

cin >> value;

if (L.searchAge(value))

{

cout << value << " exist in the list" << endl;

}

else

{

cout << value << " doesn't exist in the list" << endl;

}

break;

case 5:

exit(0);

break;

default:

cout << "\nPlease enter a valid choice..\n\n";

}

}

return (0);

}

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

**Part C**

**Program #5**

Create a student hash table that contains information, studentID (int), name (string), marks\_oop345 (float). The size of hash table is equal to the number of students in the class. Use linear probing in case of collisions. Perform insertion, deletion, display and search operations.

**File – HashTable.h**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 22-11-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include <iostream>

#include <cassert>

#include <string>

#define MAX 55

struct StudentData

{

int StudentId;

std::string StudentName;

float marks\_oop345;

};

class hashtable

{

private:

StudentData hashdata[MAX];

public:

int used;

hashtable();

int getused() const;

void insert(const StudentData &entry);

void find(int key, bool &found, StudentData &entry);

void findindex(int key, bool &found, int &i);

int hash(int key) const;

void display();

bool deleteRecord(int key);

};

**File – HashTable.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 22-11-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include <iostream>

#include <iomanip>

#include <cassert>

#include "HashTable.h"

using namespace std;

int idx = 0;

hashtable::hashtable()

{

used = 0;

for (int i = 0; i < MAX; i++)

{

hashdata[i].StudentId = -1;

}

}

int hashtable::getused() const

{

return used;

}

void hashtable::find(int key, bool &found, StudentData &entry)

{

findindex(key, found, idx);

if (found)

{

entry = hashdata[idx];

}

}

void hashtable::findindex(int key, bool &found, int &i)

{

int count = 0; // counter

i = hash(key);

// linear probing

while (count < MAX && hashdata[i].StudentId != -1 && hashdata[i].StudentId != key && hashdata[i].StudentId != -1000)

{

count++;

i = (i + 1) % MAX;

}

found = hashdata[i].StudentId == key;

cout << "Found = " << found << endl;

}

void hashtable::insert(const StudentData &entry)

{

bool alreadyhere;

int index = 0;

findindex(entry.StudentId, alreadyhere, index);

if (!alreadyhere)

{

assert(getused() < MAX);

used++;

}

hashdata[index] = entry;

}

int hashtable::hash(int key) const

{

int hashnum = key % MAX;

return hashnum;

}

void hashtable::display()

{

cout << "\nDisplaying Hash Table\n";

if (hashdata->StudentId == -1 && hashdata->StudentId == 0 && hashdata->StudentId == 1000)

{

cout << "\n------------>Empty Hash Table<------------" << endl;

}

for (int i = 0; i < MAX; i++)

{

if (hashdata[i].StudentId != -1 && hashdata[i].StudentId != 0 && hashdata[i].StudentId != -1000)

{

cout << "\n\n";

cout << " Student Id: " << hashdata[i].StudentId << endl;

cout << " Student Name: " << hashdata[i].StudentName << endl;

cout << " Subject Marks: " << hashdata[i].marks\_oop345 << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

}

}

bool hashtable::deleteRecord(int key)

{

bool alreadyhere;

int index = 0;

findindex(key, alreadyhere, index);

if (alreadyhere)

{

hashdata[index].StudentId = -1000;

return true;

}

else

{

return false;

}

}

**File – main.cpp**

// Name: Harsh Patel

// Seneca Student ID: 146315205

// Seneca email: hpatel262@myseneca.ca

// Date of completion: 22-11-2022

//

// I confirm that I am the only author of this file

// and the content was created entirely by me.

#include <iostream>

#include "HashTable.h"

using namespace std;

#include <string>

void print\_menu()

{

cout << "\nThe following choices are available: " << endl;

cout << "I insert a new record or update existing record " << endl;

cout << "F Find a record " << endl;

cout << "S Get the number of records " << endl;

cout << "D Delete" << endl;

cout << "Z display " << endl;

cout << "Q Quit this program " << endl;

}

int main()

{

char choice;

hashtable dataTable;

StudentData rec;

int key, size = 0;

bool found = false;

do

{

print\_menu();

cout << "Enter choice: ";

cin >> choice;

choice = toupper(choice);

switch (choice)

{

case 'I':

cout << "Enter Student Id as key (int >0) for record: ";

// cin >> rec.StudentId

while (!(cin >> rec.StudentId) || rec.StudentId < 1)

{

cout << "-----------> Enter Valid Id(int): ";

cin.clear();

cin.ignore(INT32\_MAX, '\n');

}

cout << "Entered Student Id in record: " << rec.StudentId << endl;

cout << "Enter Student Name for record: ";

cin.ignore();

getline(cin, rec.StudentName);

for (int i = 0; i < rec.StudentName.length(); i++)

{

if (isdigit(rec.StudentName[i]))

{

cout << "-----------> Enter Valid Student Name(Characters Only): ";

cin.clear();

//cin.ignore();

getline(cin, rec.StudentName);

}

}

cout << "Entered Student Name for record: " << rec.StudentName << endl;

cout << "Enter marks for Subject OOP345 for record: ";

while (!(cin >> rec.marks\_oop345) || rec.marks\_oop345 < 0)

{

cout << "-----------> Enter Valid Marks: ";

cin.clear();

cin.ignore(INT32\_MAX, '\n');

}

// cin >> rec.marks\_oop345;

cout << "Entered marks for Subject OOP345 for record: " << rec.marks\_oop345 << endl;

dataTable.insert(rec);

cout << "Record was inserted in hash table" << endl;

break;

case 'F':

cout << "Enter Student Id (int >=0) to search for record: ";

// cin >> key;

while (!(cin >> key) || key < 1)

{

cout << "-----------> Enter Valid Id(int): ";

cin.clear();

cin.ignore(INT32\_MAX, '\n');

}

cout << "Entered Student Id to search in record: " << key << endl;

dataTable.find(key, found, rec);

if (found)

{

cout << "\nRecord was found. " << endl;

cout << "Student Id => " << rec.StudentId << endl;

cout << "Student Name => " << rec.StudentName << endl;

cout << "Subject Marks => " << rec.marks\_oop345 << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

else

cout << "Record with Student Id " << key << " not found" << endl;

break;

case 'S':

size = dataTable.getused();

cout << "There are " << size << " records in the hash table" << endl;

cout << "There are " << MAX - size << " empty slots left in the table" << endl;

break;

case 'D':

cout << "Enter Student Id to Delete record: ";

// cin >> key;

while (!(cin >> key) || key < 1)

{

cout << "-----------> Enter Valid Id(int): ";

cin.clear();

cin.ignore(INT32\_MAX, '\n');

}

cout << "Entered Student Id to Delete record: " << key << endl;

if (dataTable.deleteRecord(key))

{

cout << "successful deletion" << endl;

dataTable.used--;

}

else

{

cout << "Student Id not found to delete" << endl;

}

break;

case 'Q':

cout << "Thank you for using my Hash table" << endl;

break;

case 'Z':

dataTable.display();

break;

default:

cout << choice << " is invalid" << endl;

}

} while ((choice != 'Q'));

return 0;

}

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with low confidence