**Assignment: C++ Programming**

**Source Code**

#include <iostream.h>

#include <fstream.h>

#include <iomanip.h>

#include <math.h>

#include <conio.h>

#include <string.h>

using namespace std;

template <class T>

class BST

{

struct node{

T value;

struct node\* right;

struct node\* left;

};

public:

BST();

~BST();

void Insert(T data);

bool Search(T data);

void InOrder();

void PreOrder();

void PostOrder();

int size();

private:

struct node\* root;

int treeSize;

void Insert(struct node\*\* node, T data);

bool Search(struct node\* node, T data);

void InOrder(struct node\* node);

void PreOrder(struct node\* node);

void PostOrder(struct node\* node);

void deleteTree(struct node\* node);

};

template <class T>

BST<T>::BST(){

this->root = NULL;

this->treeSize = 0;

}

template <class T>

BST<T>::~BST(){

deleteTree(this->root);

}

template <class T>

int BST<T>::size(){

return this->treeSize;

}

template <class T>

void BST<T>::Insert(T data){

Insert(&(this->root), data);

}

template <class T>

void BST<T>::Insert(struct node\*\* node, T data){

if(\*node == NULL) {

struct node\* tmp = new struct node;

tmp->value = data;

tmp->left=NULL;

tmp->right = NULL;

\*node = tmp;

this->treeSize++;

}else{

if(data > (\*node)->value){

Insert(&(\*node)->right, data);

}else{

Insert(&(\*node)->left, data);

}

}

}

template <class T>

void BST<T>::InOrder(){

InOrder(this->root);

cout << endl;

}

template <class T>

void BST<T>::InOrder(struct node\* node){

if(node != NULL){

InOrder(node->left);

cout << node->value<<endl;

InOrder(node->right);

}

}

template <class T>

void BST<T>::PreOrder(){

PreOrder(this->root);

cout << endl;

}

template <class T>

void BST<T>::PreOrder(struct node\* node){

if(node != NULL) {

cout << node->value;

InOrder(node->left);

InOrder(node->right);

}

}

template <class T>

void BST<T>::PostOrder(){

PostOrder(this->root);

cout << endl;

}

template <class T>

void BST<T>::PostOrder(struct node\* node){

if(node != NULL){

InOrder(node->left);

InOrder(node->right);

cout << node->value;

}

}

template <class T>

void BST<T>::deleteTree(struct node\* node){

if(node != NULL){

deleteTree(node->left);

deleteTree(node->right);

delete node;

}

}

template <class T>

bool BST<T>::Search(T data){

return Search(this->root, data);

}

template <class T>

bool BST<T>::Search(struct node\* node, T data){

if(node == NULL){

return false;

}else{

if(data == node->value){

return true;

}

if(data > node->value){

return Search(node->right, data);

}else{

return Search(node->left, data);

}

}

}

#endif // BST\_H

fstream file;

char year[5]="";

//Non-Member function to find month, to avoid user data-entry

char \*returnMonth(int m){

switch(m){

case 1 : return("January");

case 2 : return("February");

case 3 : return("March");

case 4 : return("April");

case 5 : return("May");

case 6 : return("June");

case 7 : return("July");

case 8 : return("August");

case 9 : return("September");

case 10 : return("October");

case 11 : return("November");

case 12 : return("December");

}

return("Invalid");

}

//Non-Member functions, checks for whether file is loaded or not

//it returns 1 if not loaded and 0 otherwise.

int checkFile(){

if(file==NULL){

cout<<"\nThere is No Data to Display\n";

getch();

return(1);

}

return(0);

}

class weather

{

public:

double avgtemp[12];

char month[12][20];

weather();

void getdata(int m){ //simple getdata...

strcpy(month[m],returnMonth(m)); //avoiding user to input

cout<<"\nEnter temperature for "<<month[m]<<" : ?\b";

cin>>avgtemp[m];

}

void displaydata(int m){ //simple displaydata

cout<<setw(20)<<setprecision(2)<<setiosflags(ios::left)

<<setiosflags(ios::showpoint)

<<month[m]<<setw(10)<<setiosflags(ios::right)<<avgtemp[m];

}

void displaytemp(int i){ //Display only Temperature data

cout<<setw(10)<<setprecision(2)<<setiosflags(ios::right)

<<setiosflags(ios::showpoint)<<avgtemp[i];

}

double returntemp(int i){

return(avgtemp[i]);

MyNotepad::~MyNotepad()

{

delete ui;

}

void MyNotepad::newDocument()

{

currentFile.clear();

ui->textEdit->setText(QString());

}

void MyNotepad::open()

{

QString fileName = QFileDialog::getOpenFileName(this, "Open the file");

QFile file(fileName);

currentFile = fileName;

if (!file.open(QIODevice::ReadOnly | QFile::Text)) {

QMessageBox::warning(this, "Warning", "Cannot open file: " + file.errorString());

return;

}

setWindowTitle(fileName);

QTextStream in(&file);

QString text = in.readAll();

ui->textEdit->setText(text);

file.close();

}

void MyNotepad::save()

{

QString fileName;

// If we don't have a filename from before, get one.

if (currentFile.isEmpty()) {

fileName = QFileDialog::getSaveFileName(this, "Save");

currentFile = fileName;

} else {

fileName = currentFile;

}

QFile file(fileName);

if (!file.open(QIODevice::WriteOnly | QFile::Text)) {

QMessageBox::warning(this, "Warning", "Cannot save file: " + file.errorString());

return;

}

setWindowTitle(fileName);

QTextStream out(&file);

QString text = ui->textEdit->toPlainText();

out << text;

file.close();

}

void MyNotepad::saveAs()

{

QString fileName = QFileDialog::getSaveFileName(this, "Save as");

QFile file(fileName);

if (!file.open(QFile::WriteOnly | QFile::Text)) {

QMessageBox::warning(this, "Warning", "Cannot save file: " + file.errorString());

return;

}

currentFile = fileName;

setWindowTitle(fileName);

QTextStream out(&file);

QString text = ui->textEdit->toPlainText();

out << text;

file.close();

}

void MyNotepad::print()

{

#if QT\_CONFIG(printer)

QPrinter printDev;

#if QT\_CONFIG(printdialog)

QPrintDialog dialog(&printDev, this);

if (dialog.exec() == QDialog::Rejected)

return;

#endif // QT\_CONFIG(printdialog)

ui->textEdit->print(&printDev);

#endif // QT\_CONFIG(printer)

}

void MyNotepad::exit()

{

QCoreApplication::quit();

}

void MyNotepad::copy()

{

#if QT\_CONFIG(clipboard)

ui->textEdit->copy();

#endif

}

void MyNotepad::cut()

{

#if QT\_CONFIG(clipboard)

ui->textEdit->cut();

#endif

}

void MyNotepad::paste()

{

#if QT\_CONFIG(clipboard)

ui->textEdit->paste();

#endif

}

void MyNotepad::undo()

{

ui->textEdit->undo();

}

void MyNotepad::redo()

{

ui->textEdit->redo();

}

void MyNotepad::selectFont()

{

bool fontSelected;

QFont font = QFontDialog::getFont(&fontSelected, this);

if (fontSelected)

ui->textEdit->setFont(font);

}

void MyNotepad::setFontUnderline(bool underline)

{

ui->textEdit->setFontUnderline(underline);

}

void MyNotepad::setFontItalic(bool italic)

{

ui->textEdit->setFontItalic(italic);

}

void MyNotepad::setFontBold(bool bold)

{

bold ? ui->textEdit->setFontWeight(QFont::Bold) :

ui->textEdit->setFontWeight(QFont::Normal);

}

void MyNotepad::about()

{

QMessageBox::about(this, tr("About MDI"),

tr("Notepad in QT C++ By CppBuzz.com. Users are allowed to download & modify it."

"text editor using QtWidgets"));

}

void loadfile();

void displaytempscale(); //for displaying temperature scale

void updatedata(int m,double t)

{

strcpy(month[m],returnMonth(m));

avgtemp[m]=t;

}

int validate(double t){ //Validate entered temperature (change to suit ur requirement).

if(t>55 || t < (-20))

return(1); //states that invalid entry

else

return(0); //valid entry

}

};

weather :: weather(){ //Constructor to intialize object

for(int i=0;i<12;i++)

{

avgtemp[i]=0;

month[i][0]=NULL;

}

}

void weather :: loadfile(){ //for Selecting Year or creation of new year file.

clrscr();

file.close(); //It is required when trying to open different year files,

//while there is already loaded any year file.

char filename[20]="";

cout<<"\n\*\*\*\*\*Select a Year to Work With\*\*\*\*\*\n\n";

cout<<"Enter Year (eg: 2001) : ?\b";

cin>>year;

strcat(filename,"c:\\"); //Here i am assuming that path must be c:\

strcat(filename,"data"); //if u wan't to skip that criteria just remove this lines.

strcat(filename,year);

strcat(filename,".dat");

file.open(filename,ios::ate|ios::nocreate|ios::in|ios::out|ios::binary);

if(file==NULL){

char ans;

cout<<"\nFile Dose Not Exist\n"

<<"Do you wan't to create it! (y or n) N : ?\b"; //By default No

ans=getche();

if(ans=='y'|| ans=='Y'){

file.open(filename,ios::ate|ios::in|ios::out|ios::binary);

cout<<"\n\nFile Created Successfully";

}

else{

file;

cout<<"\n\nFail to load data file";

}

}

else if(file!=NULL){

cout<<"\nData File is successfully loaded";

}

getch();

}

void weather :: displaytempscale(){

int i,c=0;

cout<<"\n\n";

for(i=1;i<=70;i++)

{

if(i<=10)

cout<<"0";

else if(i<=20)

cout<<"1";

else if(i<=30)

cout<<"2";

else if(i<=40)

cout<<"3";

else if(i<=50)

cout<<"4";

else if(i<=60)

cout<<"5";

else if(i<=70)

cout<<"7";

}

cout<<"\n";

for(i=1;i<=7;i++)

{

for(c=0;c<10;c++)

cout<<c;

}

cout<<"\n";

}

void intro()

{

clrscr();

cout<<"\n\n\n\n\n\n\n\n\t\t\t";

cout<<"\t\t\tEmail : admin@syntax-example.com\n"

<<"\t\t\tWebsite : http://www.syntax-example.com";

cout<<"\n\n\t\t\t";

cout<<"Thanks for using.";

getch();

}

void main(){

char choice='1';

int cnt,i,j,iNum,totstars=0,location,m;

double decNum,dNum,high=0,low=99999,avg=0,t;

char c;

weather obj;

file.close();

do{

clrscr();

cout<<"\n\*\*\*\*\*Main Menu\*\*\*\*\*\n";

cout<<"1> Select a year to work with\n"

<<"2> Display data as Table\n"

<<"3> Display data as horizontal histogram\n"

<<"4> Display Yearly Statistics to date\n"

<<"5> Record Data\n"

<<"6> Change Data\n"

<<"7> Save Data\n"

<<"0> Exit\n"

<<"Please enter a number (0...7) => ";

choice=getche();

//again for sake of simplicity i have directly return code

//in case brace rather than going for member-function.

switch(choice){

case '0' : intro();

goto out;

case '1' : obj.loadfile();

break;

case '2' : clrscr();

cout<<"\n\*\*\*\*\*Display Data as a Table\*\*\*\*\*\n\n";

if(checkFile()){

goto endtable;

}

cout<<"\nTable of Temperature data for "<<year;

cout<<"\n\n";

file.seekg(0,ios::end);

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

file.seekg(0,ios::beg);

for(i=1;i<=cnt;i++)

{

if(i==1)

cout<<"\nQuater 1 : ";

else if(i==4)

cout<<"\nQuater 2 : ";

else if(i==7)

cout<<"\nQuater 3 : ";

else if(i==10)

cout<<"\nQuater 4 : ";

file.read((char \*)&obj, sizeof(obj));

if(!file.eof()){

obj.displaytemp(i);

}

file.clear();

}

getch();

endtable:

break;

case '3' : clrscr();

cout<<"\n\*\*\*\*\*Display Data as a Horizontal Histogram\*\*\*\*\*\n\n";

if(checkFile()){

goto endhistogram;

}

cout<<"Histogram of Temperature data for "<<year;

obj.displaytempscale();

file.seekg(0,ios::end);

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

file.seekg(0,ios::beg);

for(i=1;i<=cnt;i++)

{

cout<<"\n"<<setw(15)<<setiosflags(ios::left)<<returnMonth(i);

file.read((char \*)&obj, sizeof(obj));

if(!file.eof()){

decNum=obj.returntemp(i);

iNum=floor(decNum);

dNum=decNum-iNum; //for finding decimal part.

totstars=iNum;

if(dNum >= 0.5)

totstars++;

for(j=1;j<=totstars;j++)

cout<<"\*";

cout<<" "<<totstars;

}

file.clear();

}

obj.displaytempscale();

getch();

endhistogram:

break;

case '4' : clrscr();

cout<<"\n\*\*\*\*\*Display Yearly Statistics to Date\*\*\*\*\*\n\n";

if(checkFile()){

goto endstatus;

}

cout<<"Temperature Statistics data for "<<year;

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

file.seekg(0,ios::beg);

high=avg=0;

low=99999;

for(i=1;i<=cnt;i++)

{

file.read((char \*)&obj, sizeof(obj));

double tmp=obj.returntemp(i);

if(!file.eof()){

if(high < tmp)

high=tmp;

if(low > tmp)

low=tmp;

avg=avg+tmp;

}

file.clear();

}

avg=avg/double(cnt);

cout<<"\n\nHighest Monthly Average : "<<high;

cout<<"\nLowest Monthly Average : "<<low;

cout<<"\nAverage Yearly Temperature : "<<avg;

getch();

endstatus:

break;

case '5' : clrscr();

cout<<"\n\*\*\*\*\*Record Data\*\*\*\*\*\n\n";

if(checkFile()){

goto endRecord;

}

else{

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

if(cnt==12)

cout<<"\n\nData-Entry of "<<year<<" is already been done\n";

for(i=cnt+1;i<=12;i++)

{

enteragain:

cout<<"\nDo u wan't to enter data for "<<returnMonth(i)<<" (Y or N) Y : ?\b";

c=getche();

if(c=='n' || c=='N')

goto endRecord;

obj.getdata(i);

if(obj.validate(obj.returntemp(i)))

{

cout<<"\nInvalid Data Entry\n";

goto enteragain;

}

cin.get(c);

file.write((char \*) &obj, sizeof(obj));

}

}

getch();

endRecord:

break;

case '6' : clrscr();

cout<<"\n\*\*\*\*\*Change Data\*\*\*\*\*\n\n";

if(checkFile()){

goto endchange;

}

else{

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

tryagain:

cout<<"\nEnter Month (in digit) whose temperature is to be changed : ?\b";

cin>>m;

if(m <= 0 || m > cnt){

cout<<"\n\nInvalid Month\n";

getch();

goto tryagain;

}

tempagain:

cout<<"\nEnter Temperature : ?\b";

cin>>t;

if(obj.validate(t))

{

cout<<"\nInvalid Data Entry\n";

goto tempagain;

}

file.seekg(0,ios::beg);

location= (m-1) \* sizeof(obj);

file.seekp(location);

obj.updatedata(m,t);

cin.get(c);

file.write((char \*) &obj, sizeof(obj))<<flush;

}

endchange:

break;

case '7' : clrscr();

cout<<"\n\*\*\*\*\*Store the Current Data\*\*\*\*\*\n\n";

if(checkFile()){

goto endsave;

}

flush(file);

cout<<"\nData in Memory is Saved successfully\n";

getch();

endsave:

break;

default : cout<<"\n\nInvalid Input\n";

getch();

}

}while(choice!='0');

flush(file);

file.close();

clrscr();

out:

using namespace std;

void moveTemperaturesToRight(double temperatures[],

double windSpeed[],

string windDirection[])

{

for (int i = 3; i > 0; i--)

{

temperatures[i] = temperatures[i - 1];

windSpeed[i] = windSpeed[i - 1];

windDirection[i] = windDirection[i - 1];

}

}

int main()

{

string name;

int choice;

int numOfReadings = 0;

int size;

double temperatures[4], windSpeeds[4];

string windDirections[4];

bool initialized = false;

char str;

//Have the user provide a name for the weather station upon entry.

cout << "Enter the name of weather station: ";

getline(cin, name);

//Control loop to perform various actions.

while (true)

{

cout << "I. Input a complete weather reading." << "\n";

cout << "P. Print the current weather." << "\n";

cout << "H. Print the weather history (from most recent to oldest)." << "\n";

cout << "E. Exit the program." << "\n";

cout << "Enter your choice: ";

cin >> str;

if (str != 'I'&& str != 'P'&& str != 'H'&& str != 'E')

choice = 0;

else

choice = str;

//Switch based on choice.

switch (choice)

{

case 'I':

moveTemperaturesToRight(temperatures,

windSpeeds,

windDirections);

cout << "Enter the temperature:";

cin >> temperatures[0];

//get correct wind speed

do

{

cout << "Enter the wind speed (a value >=0):";

cin >> windSpeeds[0];

while (cin.fail() || (cin.peek() != '\r' && cin.peek() != '\n'))

{

cout << "Invalid Input! Re Enter the wind speed" << endl;

cin.clear();

while (cin.get() != '\n');

cin >> windSpeeds[0];

}

} while (windSpeeds[0] < 0);

//get correct wind direction

do

{

cout << "Enter the wind direction (North,South,East or West):";

cin >> windDirections[0];

} while (windDirections[0] != "North" && windDirections[0] != "South" && windDirections[0] != "East" && windDirections[0] != "West");

initialized = true;

if (initialized)

numOfReadings++;

if (numOfReadings > 4)

numOfReadings = 4;

break;

case 'H': //Print the current weather, if valid weather is entered.

cout << "Enter size of the history wants:";

cin >> size;

if (numOfReadings<size)

{

cout << "History size is high";

break;

}

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

{

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

cout << "Temperature: " << temperatures[i] << "\n";

cout << "Wind speed: " << windSpeeds[i] << "\n";

cout << "Wind direction: " << windDirections[i] << "\n" << "\n";

}

if (numOfReadings == 0)

cout << "Please enter the details before asking to print." << "\n";

break;

case 'P':

if (numOfReadings == 0)

{

cout << "Please enter the details before asking to print." << "\n";

break;

}

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

cout << "Temperature: " << temperatures[0] << "\n";

cout << "Wind speed: " << windSpeeds[0] << "\n";

cout << "Wind direction: " << windDirections[0] << "\n" << "\n";

break;

case 'E':

return 0; //Stops execution.

default:

cout << "Invalid choice. Please follow the menu." << "\n";

}

}

}

if (option == '1')

{

cout << "Enter the temperature in Celsius: ";

cin >> cel;

fahr = (1.8 \* cel) + 32.0; //temperature conversion formula

cout << "\nTemperature in degree Fahrenheit: " << fahr << " F" << endl;

}

//option for converting Fahrenheit into Celsius

else if (option == '2')

{

cout << "Enter the temperature in Fahrenheit: ";

cin >> fahr;

cel = (fahr - 32) / 1.8; //temperature conversion formula

cout << "\nTemperature in degree Celsius: " << cel << " C" << endl;

}

else

cout << "Error Wrong Input." << endl;

return 0;

}

cout<<"\n\*\*\*\*\*Record Data\*\*\*\*\*\n\n";

if(checkFile()){

goto endRecord;

}

else{

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

if(cnt==12)

cout<<"\n\nData-Entry of "<<year<<" is already been done\n";

for(i=cnt+1;i<=12;i++)

{

enteragain:

cout<<"\nDo u wan't to enter data for"<<returnMonth(i)<<" (Y or N) Y : ?\b";

c=getche();

if(c=='n' || c=='N')

goto endRecord;

obj.getdata(i);

if(obj.validate(obj.returntemp(i)))

{

cout<<"\nInvalid Data Entry\n";

goto enteragain;

}

cin.get(c);

file.write((char \*) &obj, sizeof(obj));

}

}

getch();

endRecord:

break;

case '6' : clrscr();

cout<<"\n\*\*\*\*\*Change Data\*\*\*\*\*\n\n";

if(checkFile()){

goto endchange;

}

else{

cnt=file.tellg();

cnt=cnt/sizeof(obj); //cnt contains how many records previously recorded

tryagain:

cout<<"\nEnter Month (in digit) whose temperature is to be changed : ?\b";

cin>>m;

if(m <= 0 || m > cnt){

cout<<"\n\nInvalid Month\n";

getch();

goto tryagain;

}

tempagain:

cout<<"\nEnter Temperature : ?\b";

cin>>t;

if(obj.validate(t))

{

cout<<"\nInvalid Data Entry\n";

goto tempagain;

}

file.seekg(0,ios::beg);

location= (m-1) \* sizeof(obj);

file.seekp(location);

obj.updatedata(m,t);

cin.get(c);

file.write((char \*) &obj, sizeof(obj))<<flush;

}

endchange:

break;

case '7' : clrscr();

cout<<"\n\*\*\*\*\*Store the Current Data\*\*\*\*\*\n\n";

if(checkFile()){

goto endsave;

}

flush(file);

cout<<"\nData in Memory is Saved successfully\n";

getch();

endsave:

break;

default : cout<<"\n\nInvalid Input\n";

getch();