Khoi Duong

Prof. Yang

CS360

8/15/2022

HW#6

1.

Source code:

```
#include<iostream>
#include <fstream>
#include <cstdlib>
#define MAX 500 // for 500 address maximum
using namespace std;
//Class addressType definition
class addressType
{
    private:
    //Data member to store street
    string streetAddress;
    //Data member to store city
    string city;
    //Data member to store state
    string state;
    //Data member to store pin
    string zipcode;
    public:
    //Member function to print address
    void print()
    {
    cout<<"\n ----- ADDRESS ----- \n";
    cout<<" Street: "<<streetAddress;
    cout<<"\t City: "<<city;
    cout<<"\t State: "<<state;
    cout<<"\t Zipcode: "<<zipcode;
    }//End of function
```

```
    //Default constructor
    addressType()
    {
    streetAddress = city = state = zipcode = "";
    }//End of constructor

    //Parameterized constructor
    addressType(string sa, string c, string st, string z)
    {
    streetAddress = sa;
    city = c;
    state = st;
    zipcode = z;
    }//End of constructor

    //Getter and setter methods
    void setStreetAddress(string sa) {streetAddress = sa;}
    string getStreetAddress() {return streetAddress;}
    void setCity(string c) {city = c;}
    string getCity() {return city;}
    void setState(string st) {state = st;}
    string getState() {return state;}
    void setZipcode(string z) {zipcode = z;}
    string getZipcode() {return zipcode;}
};//End of class

//Class dateType definition
class dateType
{
    private:
    //Data member to store date, month, and year
    int date, month, year;
    public:
    //Function to print date of birth
    void print()
    {
    cout<<"\n Date of Birth: "<<date<<":"<<month<<":"<<year;
    }//End of function

    //Default constructor
    dateType(): date(0), month(0), year(0) {}

    //Parameterized constructor
    dateType(int d, int m, int y): date(d), month(m), year(y) {}
```

```
    //Getter and setter methods
    void setDate(int d) {date = d;}
    int getDate() {return date;}
    void setMonth(int m) {month = m;}
    int getMonth() {return month;}
    void setYear(int y) {year = y;}
    int getYear() {return year;}
};//End of class

//Class personType definition
class personType
{
    public:
    //Parameterized constructor
    personType(string first = " ", string last = " "): firstName(first),
lastName(last) {}

    //Function to print person name
    void print() const
    {
    cout<<"\n First Name: "<<firstName<<"\t Last Name: "<<lastName;
    }//End of function

    //Getter and setter methods
    void setName(string first, string last) {firstName = first; lastName = last;}
    string getFirstName() {return firstName;}
    string getLastName() {return lastName;}

    private:
    //Data member to store first name and last name
    string firstName;
    string lastName;
};//End of class

//Class extPersonType derived from personType
class extPersonType : public personType
{
    private:
    //Data member to store phone number and person status
    string phoneNumber;
    string personStatus;
    public:
    //Declares an object of addressType class and dateType class
    addressType address;
```

```
    dateType dateOfBirth;

    //Function to print data
    void print() const
    {
    cout<<"\n Phone Number: "<<phoneNumber;
    cout<<"\n Person Type: "<<personStatus;
    }//End of function
    extPersonType(){}
    extPersonType(addressType a, dateType d, string ph, string ps, string fi, string
la):personType(fi, la)
    {
    address = a;
    dateOfBirth = d;
    phoneNumber = ph;
    personStatus = ps;
    }//End of constructor

    //Getter and setter methods
    void setPhoneNumber(string ph) {phoneNumber = ph;}
    string getPhoneNumber() {return phoneNumber;}
    void setPersonStatus(string ps) {personStatus = ps;}
    string getPersonStatus() {return personStatus;}
};//End of class


//Class addressBookType definition
class addressBookType
{
    private:
    //Declares an array of objects of extPersonType class of size MAX
    extPersonType ept[MAX];
    //To store number of records
    int numberOfRecord;
    public:

    //Function to return number of records
    int getNumberOfRecord() {return numberOfRecord;}

    //Prototype of the functions
    void readFile(string);
    void displayFile();
    void sortLastName();
    void searchLastName(string, int);
    void getAddressByMonth(int);
```

```
    void duplicateType(string);
    void saveData();
};//End of class

//Function to display duplicate person status
void addressBookType::duplicateType(string type)
{
    //Loop variable
    int x;
    //Loops till number of records
    for(x = 0; x < getNumberOfRecord(); x++)
    {
    //Checks if current person status is equal to the person status given as
parameter then display the person information
        if(ept[x].getPersonStatus().compare(type) == 0)
        {
            cout<<"\n\n\t\t\t      ******************      Person      Information
****************";
            ept[x].personType::print();
            ept[x].address.print();
            ept[x].dateOfBirth.print();
            ept[x].print();
        }
    }
}

//Function to display the person whose month of the date of birth matches
void addressBookType::getAddressByMonth(int monthNumber)
{
    int x;

    //Loops till number of records
    for(x = 0; x < getNumberOfRecord(); x++)
    {
//Checks if the current month is equal to previous month then display the person
information
        if(ept[x].dateOfBirth.getMonth() == monthNumber)
        {
            cout<<"\n\n\t\t\t      ******************      Person      Information
****************";
            ept[x].personType::print();
            ept[x].address.print();
            ept[x].dateOfBirth.print();
            ept[x].print();
        }
```

```
    }
}

//Function to search a person last name and display person information
void addressBookType::searchLastName(string name, int no)
{
    int x;
    int flag = 0;
    //Loops till number of records
    for(x = 0; x < getNumberOfRecord(); x++)
    {
    //Checks if the current person last name is equal to person name in the parameter
    if(ept[x].personType::getLastName() == name)
    {
        //Check if no is 2 then display complete person information
        if(no == 2)
            {
            cout<<"\n\n\t\t\t      ******************      Person      Information
****************";
            ept[x].personType::print();
            ept[x].address.print();
            ept[x].dateOfBirth.print();
            ept[x].print();
            }//End of inner if
        //If no is 3 display only address phone number and date of birth
    else
    {
        cout<<"\n\n\t\t\t ****************** Person address phone number and date
of birth ****************";
        ept[x].address.print();
        ept[x].dateOfBirth.print();
        ept[x].print();
    }//End of else
    flag = 1;
    }//End of if
    }//End of for
    //Checks if flag value is not one person name not found
    if(flag != 1)
    cout<<"\n Record for "<<name<<" not found.";
}//End of function

//Function to read data from file and store in the class respective data member
void addressBookType::readFile(string fileName)
{
    int co = 0;
```

```
    string f, l;
    int d;

    //Creates an object of ifstream
    ifstream rFile;

    //Opens the file Address.txt for reading
    rFile.open(fileName.c_str());


    // Checks if the file unable to open for reading display's error message and
stop
    if(!rFile)
    {
    cout<<"\n ERROR: Unable to open the file "<<fileName<<" for reading.";
    exit(0);
    }// End of if condition

    //Loops till end of file
    while(!rFile.eof())
    {
    // Checks if current record counter co is equals to maximum record then display
    // error message and come out of the loop
    if(co == MAX)
        {
        cout<<"\n Reached maximum limit. Cannot add more records.";
        break;
        }// End of while loop

    // Read first and last name
    rFile>>f;
    rFile>>l;
    ept[co].setName(f, l);
    // Reads date of birth
    rFile>>d;
    ept[co].dateOfBirth.setDate(d);
    rFile>>d;
    ept[co].dateOfBirth.setMonth(d);
    rFile>>d;
    ept[co].dateOfBirth.setYear(d);
    rFile.ignore();
    // Reads address
    getline(rFile, l);
    ept[co].address.setStreetAddress(l);
    // Reads city
```

```
    getline(rFile, l);
    ept[co].address.setCity(l);
    // Reads state
    getline(rFile, l);
    ept[co].address.setState(l);
    // Reads zip code
    rFile>>l;
    ept[co].address.setZipcode(l);
    // Reads phone number
    rFile>>l;
    ept[co].setPhoneNumber(l);
    // Reads person status
    rFile>>l;
    ept[co].setPersonStatus(l);
    //Increase the counter by one
    co++;
    }
    //Close file
    rFile.close();
    //Assigns the counter value to the number of records
    numberOfRecord = co;
}//End of function

//Function to write data to file
void addressBookType::saveData()
{
    int co = 0;
    //Creates an object of ofstream
    ofstream wFile;
    //Opens the file AddressNew.txt for writing
    wFile.open("AddressNew.txt");

    //Loops till end of the record
    for(int co = 0; co < numberOfRecord; co++)
    {
    // Writes data to file
    wFile<<ept[co].getFirstName()<<" "<<ept[co].getLastName()<<endl;
    wFile<<ept[co].dateOfBirth.getDate()<<" "<<ept[co].dateOfBirth.getMonth()<<" "
    <<ept[co].dateOfBirth.getYear()<<endl;
    wFile<<ept[co].address.getStreetAddress()<<endl;
    wFile<<ept[co].address.getCity()<<endl;
    wFile<<ept[co].address.getState()<<endl;
    wFile<<ept[co].address.getZipcode()<<endl;
    wFile<<ept[co].getPhoneNumber()<<endl;
    // Checks if it is last record then do not write new line character after it
```

```
    if(co == numberOfRecord - 1)
    wFile<<ept[co].getPersonStatus();
    // Otherwise it is not last record write new line character after it
    else
    wFile<<ept[co].getPersonStatus()<<endl;
    }//End of while
    cout<<"\n\n File Saved Successfully.";
    //Close file
    wFile.close();
}//End of function

//Function to display person information
void addressBookType::displayFile()
{
    //Loops till end of the record
    for(int x = 0; x < numberOfRecord; x++)
    {
    cout<<"\n\n\t\t\t  ******************  Person  "<<(x  +  1)<<"  Information
****************";
    ept[x].personType::print();
    ept[x].address.print();
    ept[x].dateOfBirth.print();
    ept[x].print();
    }//End of for
}//End of function

//Function to sort the person information based on last name
void addressBookType::sortLastName()
{
    int x, y;
    //Creates an temporary object
    extPersonType temp;
    //Loops till end of the record
    for(x = 0; x < getNumberOfRecord(); x++)
    {
    //Loops till end of the record minus one and x value
        for(y = 0; y < getNumberOfRecord() - x - 1; y++)
        {
        //Checks if the current person last name is greater than next person last
name
        if(ept[y].personType::getLastName() > ept[y + 1].personType::getLastName())
            {
            //Swapping process
            temp = ept[y];
            ept[y] = ept[y + 1];
```

```
            ept[y + 1] = temp;
            }
        }
    }
}

//Displays menu, accepts user choice and return it
int menu()
{
    int choice;
    //Displays menu
    cout<<"\n\n ********** Address Book Menu **********";
    cout<<"\n\t 1. Sort the address book by last name. ";
    cout<<"\n\t 2. Search for a person by last name. ";
    cout<<"\n\t 3. Print the address phone number and date of birth of a given
person (if exist). ";
    cout<<"\n\t 4. Print names of people whose birthdays are in a given month. ";
    cout<<"\n\t 5. Depending on request, print all family members, friends or
business associates. ";
    cout<<"\n\t 6. Save data.";
    cout<<"\n\t 7. Exit.";
    //Accept user choice
    cout<<"\n\t\t What is your choice? ";
    cin>>choice;
    //Return user choice
    return choice;
}//End of function

//Main function definition
int main()
{
    int choice;
    int month;
    string data;
    string fileName;
    //Creates an object of addressBookType class
    addressBookType ad;
    cout<<"\n Enter the filename: ";
    cin>>fileName;
    //Read the file contents
    ad.readFile(fileName);

    //Loops till user choice is not 6
    do
    {
```

```
    //Displays menu and stores the user choice
        choice = menu();
        switch(choice)
        {
        case 1:
            ad.sortLastName();
            ad.displayFile();
            break;
        case 2:
            cout<<"\n Enter the last name to search record: ";
            cin>>data;
            ad.searchLastName(data, 2);
            break;
        case 3:
            cout<<"\n Enter the last name to print address phone number and date of
birth: ";
            cin>>data;
            ad.searchLastName(data, 3);
            break;
        case 4:
            cout<<"\n Enter the month number print address phone number and date of
birth: ";
            cin>>month;
            ad.getAddressByMonth(month);
            break;
        case 5:
            cout<<"\n Enter the person type (Family / Friends / Business): ";
            cin>>data;
            ad.duplicateType(data);
            break;
        case 6:
            ad.saveData();
            break;
        case 7:
            cout<<"\n\t\t Thanks for using My Address Book.";
            exit(0);
        default:
            cout<<"\n Invalid choice!";
        }
    }while(1);
}//End of main function
```

Supposed that we have a file named “book.txt” with the following data:

Shelly Malik 12 8 2000 Lincoln Drive

Omaha	Kennedy Blvd	2 6 1975
Nebraska	Omaha	Disney Road
68131	Nebraska	Orlando
402-555-1212	68172	Florida
Family	402-777-8888	35672
Donald Duck	Friend	415-782-5555
10 6 1980	Goof Goofy	Business
Disney Street	2 6 1965	Bash Bashfull
Orlando	Disney Street	2 8 1965
Florida	Los Angles	Long Road
11234	California	New York
622-873-8920	91340	New York
Friend	215-782-9000	01101
Chelsea Tomak	Family	212-782-8000
12 8 1999	Brave Balto	Friend

Run program & result:

Enter the filename: book.txt

********** Address Book Menu **********

1. Sort the address book by last name.
2. Search for a person by last name.
3. Print the address phone number and date of birth of a given person (if exist).
4. Print names of people whose birthdays are in a given month.

5. Depending on request, print all family members, friends or business associates.

6. Save data.

7. Exit.

What is your choice? 1

********** Person 1 Information **********

First Name: Brave Last Name: Balto

----- ADDRESS -----

Street: Disney Road City: Orlando State: Florida Zipcode: 35672

Date of Birth: 2:6:1975

Phone Number: 415-782-5555

Person Type: Business

********** Person 2 Information **********

First Name: Bash Last Name: Bashfull

----- ADDRESS -----

Street: Long Road City: New York State: New York Zipcode: 01101

Date of Birth: 2:8:1965

Phone Number: 212-782-8000

Person Type: Friend

********** Person 3 Information **********

First Name: Donald Last Name: Duck

----- ADDRESS -----

Street: Disney Street City: Orlando State: Florida Zipcode: 11234

Date of Birth: 10:6:1980

Phone Number: 622-873-8920

Person Type: Friend

********** Person 4 Information **********

First Name: Goof Last Name: Goofy

----- ADDRESS -----

Street: Disney Street City: Los Angles State: California Zipcode: 91340

Date of Birth: 2:6:1965

Phone Number: 215-782-9000

Person Type: Family

********** Person 5 Information **********

First Name: Shelly Last Name: Malik

----- ADDRESS -----

Street: Lincoln Drive City: Omaha State: Nebraska Zipcode: 68131

Date of Birth: 12:8:2000

Phone Number: 402-555-1212

Person Type: Family

********** Person 6 Information **********

First Name: Chelsea Last Name: Tomak

----- ADDRESS -----

Street: Kennedy Blvd City: Omaha State: Nebraska Zipcode: 68172

Date of Birth: 12:8:1999

Phone Number: 402-777-8888

Person Type: Friend

********** Address Book Menu **********

1. Sort the address book by last name.
2. Search for a person by last name.
3. Print the address phone number and date of birth of a given person (if exist).
4. Print names of people whose birthdays are in a given month.
5. Depending on request, print all family members, friends or business associates.
6. Save data.
7. Exit.

What is your choice? 2

Enter the last name to search record: Tomak

********** Person Information **********

First Name: Chelsea Last Name: Tomak

----- ADDRESS -----

Street: Kennedy Blvd City: Omaha State: Nebraska Zipcode: 68172

Date of Birth: 12:8:1999

Phone Number: 402-777-8888

Person Type: Friend

********** Address Book Menu **********

1. Sort the address book by last name.
2. Search for a person by last name.

3. Print the address phone number and date of birth of a given person (if exist).

4. Print names of people whose birthdays are in a given month.

5. Depending on request, print all family members, friends or business associates.

6. Save data.

7. Exit.

What is your choice? 4

Enter the month number print address phone number and date of birth: 1

********** Address Book Menu **********

1. Sort the address book by last name.

2. Search for a person by last name.

3. Print the address phone number and date of birth of a given person (if exist).

4. Print names of people whose birthdays are in a given month.

5. Depending on request, print all family members, friends or business associates.

6. Save data.

7. Exit.

What is your choice? 5

Enter the person type (Family / Friends / Business): Family

********** Person Information **********

First Name: Goof Last Name: Goofy

----- ADDRESS -----

Street: Disney Street City: Los Angles State: California Zipcode: 91340

Date of Birth: 2:6:1965

Phone Number: 215-782-9000

Person Type: Family

********** Person Information **********

First Name: Shelly Last Name: Malik

----- ADDRESS -----

Street: Lincoln Drive City: Omaha State: Nebraska Zipcode: 68131

Date of Birth: 12:8:2000

Phone Number: 402-555-1212

Person Type: Family

********** Address Book Menu **********

1. Sort the address book by last name.
2. Search for a person by last name.
3. Print the address phone number and date of birth of a given person (if exist).
4. Print names of people whose birthdays are in a given month.
5. Depending on request, print all family members, friends or business associates.
6. Save data.
7. Exit.

What is your choice? 6

File Saved Successfully.

********** Address Book Menu **********

1. Sort the address book by last name.
2. Search for a person by last name.
3. Print the address phone number and date of birth of a given person (if exist).
4. Print names of people whose birthdays are in a given month.
5. Depending on request, print all family members, friends or business associates.
6. Save data.
7. Exit.

What is your choice? 7

Thanks for using My Address Book.

2.

Source code:

```
#include <iostream>
#include <vector>
using namespace std;

class CarbonFootprint {
    protected:
    int miles_driven;
    int miles_per_gallon;
    int no_of_days;
    int no_of_people;
    int gas_con;
    int elec_con;
    public:
    CarbonFootprint(int a, int b): miles_driven(a), miles_per_gallon(b) {}
    CarbonFootprint(int a, int b, int c): no_of_people(a), elec_con(b), gas_con(c)
{}
```

```
    // pure virtual function
    virtual void getCarbonFootprint() = 0;
};

class Building: public CarbonFootprint{
    public:
    Building( int a=0, int b=0, int c=0) : CarbonFootprint(a, b, c) { }
    void getCarbonFootprint()
    {
    cout << "Carbon footprint of building is : " << (1.222*elec_con*0.00045 +
gas_con*12*0.012*0.00045)/no_of_people << " tonnes CO2.";
    }
};

class Car: public CarbonFootprint{
    public:
    Car(int a, int b) : CarbonFootprint(a,b){
        miles_driven = a;
        miles_per_gallon = b;
    }
    void getCarbonFootprint()
    {
    cout << "Carbon footprint of car is : " << (miles_driven/miles_per_gallon) *
19.8 << " lbs. or " << (miles_driven/miles_per_gallon) * 19.8 * 0.00045 <<"
tonnes."<<endl;
    }
};

class Bicycle: public CarbonFootprint{
    public:
    Bicycle( int a=0, int b=0):CarbonFootprint(a, b){
        miles_driven = a;
        no_of_days = b;
    }
    void getCarbonFootprint()
    {
    cout << "By cycling to work you save " << miles_driven * no_of_days << "lbs. of
CO2 emissions" << endl;
    }
};

// Main function for the program
int main(){
    vector <CarbonFootprint*> carbon;
```

```
    int miles1 = 0;
    int miles2 = 0;
    int mpg = 0;
    int days = 0;
    int occ = 0;
    int elec = 0;
    int gas = 0;
    cout << "Enter the miles driven in the car in a year : ";
    cin >> miles1;
    cout << "Enter the average in miles per gallon : " ;
    cin >> mpg;
    Car c(miles1, mpg);
    carbon.push_back(&c);
    cout << endl;
    cout << "Enter the distance to your office (in miles): " ;
    cin >> miles2;
    cout << "How many days do you bike to work? ";
    cin >> days;
    Bicycle b(miles2, days);
    carbon.push_back(&b);
    cout << endl;
    cout << "Enter number of occupants of the building : " ;
    cin >> occ;
    cout << "Enter annual electricity usage (in KilloWatts/hour) : ";
    cin >> elec;
    cout << "Enter annual gas usage (in KilloWatts/hour) : ";
    cin >> gas;
    Building bb(occ, elec, gas);
    carbon.push_back(&bb);
    cout << endl;
    // display carbon footprint of each object
    for ( size_t i = 0; i < carbon.size(); i++ ){
        carbon[i]->getCarbonFootprint();
        cout << endl;
    }

    //release elements of list
    for ( size_t i = 0; i < carbon.size(); i++ ){
        //TODO: release elements in the list
        CarbonFootprint* ptr;
        delete ptr;
    }

    return 0;
}
```

Run program & result:

```
Enter the miles driven in the car in a year : 15000
Enter the average in miles per gallon : 20

Enter the distance to your office (in miles): 15
How many days do you bike to work? 168

Enter number of occupants of the building : 225
Enter annual electricity usage (in KilloWatts/hour) : 1500
Enter annual gas usage (in KilloWatts/hour) : 2350

Carbon footprint of car is : 14850 lbs. or 6.6825 tonnes.

By cycling to work you save 2520lbs. of CO2 emissions

Carbon footprint of building is : 0.0043428 tonnes CO2.
```