/\*

Author:Hamza awad

file:jobslab.cpp

program: takes data from input file of jobs and their  and puts into a struct array and prompts user to search for data either with jobtitle or skill and pri$

\*/

#include <iostream>

#include <iomanip>

#include <cstring>

#include <string>

#include <fstream>

#include <cctype>

#include <stdlib.h>

using namespace std;

//define our constents

const int MAX\_SKILLS = 10;

const int MAX\_STRING\_LENGTH = 256;

const int MAX\_JOBS=10;

struct Job {

   char jobTitle[MAX\_STRING\_LENGTH];

   int numSkills;

   char skills[MAX\_SKILLS][ MAX\_STRING\_LENGTH];

   double salary;

   char company[MAX\_STRING\_LENGTH];

};

void read(ifstream &input,Job jobs[],int &numbersRead);

void searchByTitle(Job jobs[],char keyword[20],int numbersRead);

void searchBySkill(Job jobs[],char keyword[20],int numbersRead);

void sortJobs(Job jobs[],int numbersRead);

void printheader();

void printjob(Job results);

int main(){

    Job jobs[MAX\_STRING\_LENGTH];  //declare our variables and our stuct

    char c;

    char keyword[20];

    char filename[20];

    char option[10];

    int numbersRead;

    ifstream input;

    cout<<"Welcome to the Jobs.com"<<endl;

    cout<<"Jobs file: ";

    cin>>filename;  //gets input file from user

    input.open(filename);

  //checks if file exits

    if (input.fail ()){

        // when file could  not open an error message prints out

        cout << "Input file "<<filename<<" does not exist. Please contact the administrator. ";

        cout<<endl;  //error message for no files

        cout<<"Thank you for using jobSearch."<<endl;

      return 1;

    }

//checks if file is empty

    input.get (c);

    if (input.eof ()){

        cout << "Input file "<<filename<<"is empty. Please contact the administrator. ";

        cout<<endl;

        //error message for empty file

        cout<<"Thank you for using jobSearch."<<endl;

        return 1;

    }

      input.putback(c);

      read(input,jobs,numbersRead);

      sortJobs(jobs, numbersRead);

//options for user to search by job title or skill

     int optionAsInt;

     do{

       cout<<"1 Search for job by title"<<endl;

       cout<<"2 Search for a job by skill"<<endl;

       cout<<"3 Quit"<<endl;

       cout<<"Option> ";

       cin>>option;

       optionAsInt=atoi(option);

       if(optionAsInt==1){

         cout<<"search jobs: ";

         cin>>keyword;  //search word

         int length=strlen(keyword);

         for(int i=0;i<length;i++){  //lowers all letters of word from user

              keyword[i]=tolower(keyword[i]);

         }

         searchByTitle(jobs,keyword,numbersRead);

       }

      else if(optionAsInt==2){

             cout<<"search Skill: ";

             cin>>keyword;

             int length=strlen(keyword);

             for(int i=0;i<length;i++){ //lowers all letters of word from user

                keyword[i]=tolower(keyword[i]);

             }

             searchBySkill(jobs,keyword,numbersRead);

      }

      else if(optionAsInt!=3){

             cout<<"please enter valid option"<<endl;

      }

      }while(optionAsInt!=3); //keep prompting the user for options until they want to quit

     cout<<"Thank you for using jobSearch."<<endl;

return (0);

}

/\* void read: reads file into struct.

   Parameters:ifstream &input,Job jobs[],int &numbersRead.

Post-condition: parameter has been filled with jobs from file.

   Returns: nothing

\*/

void read(ifstream &input,Job jobs[],int &numbersRead){

    //read file data into struct array up to 10 jobs or less

           int i=0;

           for(i=0;i<MAX\_JOBS;i++){

              input.getline(jobs[i].jobTitle,50);  //reads title

              input>>jobs[i].numSkills; //number of skills

              input.get(); //skips enter key

              for(int j=0;j<jobs[i].numSkills;j++){

//uses number of skills for the current job to read in  skills

                 input.getline(jobs[i].skills[j],50);

              }

              input>>jobs[i].salary;

              input.get();

              input.getline(jobs[i].company,50);

              if(input.eof()){

                 break;

              }

         }

numbersRead=i;

}

/\* void searchByTitle:searches for jobs by title using input of keyword from user

   Parameters:Job jobs[],char keyword[20],int numbersRead

   Post-condition: jobs have been searched and result printed.

   Returns: nothing

\*/

void searchByTitle(Job jobs[],char keyword[20],int numbersRead){

    bool isFound=false;

    bool isFirst=true;

    for(int i=0; i<numbersRead;i++){

        char\* look;

        int length=strlen(jobs[i].jobTitle);

        char temp[length];

        strcpy(temp,jobs[i].jobTitle);

        for(int j=0;j<length;j++){

           temp[j]=tolower(temp[j]);

        }

        look=strstr(temp,keyword);

        if(look){

           isFound=true;

//if search was found we creat output table only one time and print the rest of search results

           if(isFirst){

           printheader();

           isFirst=false;

           }

           printjob(jobs[i]);

        }

    }

    if(!isFound){  //error message for when no result is found for search

       cout<<"not found"<<endl;

    }

}

/\* void searchBySkill:searches for jobs by skill using input of keyword from user

   Parameters:Job jobs[],char keyword[20],int numbersRead

   Post-condition: jobs have been searched and result printed.

   Returns: nothing

\*/

void searchBySkill(Job jobs[],char keyword[20],int numbersRead){

    bool isFound=false;

    bool isFirst=true;

    for(int i=0; i<numbersRead;i++){

        for(int skillCtr=0;skillCtr<jobs[i].numSkills;skillCtr++){

           int length=strlen(jobs[i].skills[skillCtr]);

           char temp[length];

           strcpy(temp,jobs[i].skills[skillCtr]);

           for(int k=0;k<length;k++){

               temp[k]=tolower(temp[k]);

           }

    char\* result;

          result=strstr(temp,keyword);

          if(result){

            isFound=true;

            if(isFirst){

              printheader();

              isFirst=false;

            }

            printjob(jobs[i]);

            break;

          }

       }

   }

  if(!isFound){

     cout<<"not found"<<endl; //error message for when no result is found for search

   }

}

/\* void sortjobs: sorts data in alphabatical order.

   Parameters:Job jobs[],int numbersRead

   Post-condition: jobs have been sorted alphabeticlly by last title

   Returns: nothing

\*/

void sortJobs(Job jobs[],int numbersRead){

        int mini;

        Job temp;

        int i;

        int j;

       for(int i=0;i<numbersRead-1;i++){

          mini=i;

            for(int j=i+1;j<numbersRead;j++){

                if(strcmp(jobs[j].jobTitle,jobs[mini].jobTitle) < 0){

                   mini=j;

                }

            }

        temp=jobs[i];

       jobs[i]=jobs[mini];

       jobs[mini]=temp;

    }

}

/\* void printheader: creates table for output.

   Parameters:none.

   Post-condition: jobs organized and printed in a table formate

   Returns nothing

\*/

void printheader(){

     cout<<setw(40)<<left<<"Job Title";

     cout<< setw(6) <<left<<"Salary";

     cout<<setw(12)<<" ";

     cout<< setw(20) <<right<<"company"<<endl;

     cout<<"----------------------------------------------------------------------------"<<endl;

}

/\* void printjob: prints all data sorted and organized.

   Parameters:Job result

   Post-condition: data is sorted and organized.

   Returns: nothing

\*/

void printjob(Job result){

     int i=0;

     cout<<setw(40)<<left<<result.jobTitle;

     cout<<setw(5)<<"$";

     cout<<setw(7)<<right<<result.salary;

     cout<<setw(9)<<" ";

     cout<<setw(20)<<left<<result.company<<endl;

     for(i=0;i<result.numSkills;i++){

         cout<<setw(5)<<" "<<left<<result.skills[i]<<endl;

     }

}