Hassan Harajly

10/12/17

Cis200

Assignment 2

1. **Problem Statement**

**Make a program that will will ask the user for employee information in order to display it onto the screen**

1. **Requirements**
   1. **Assumptions**

The user will enter an integer for the age of the employee and the id number.

The user will enter a float for the salary

Will use command line input

* 1. **Specifications**

The user will enter employee data.

The user will input age of the employee an integer of 13-110 will only be valid

For invalid data : display “error only numbers beteeen 13-110 are acceptable”

The user will input an id number only positive integers are allowed

For invalid input : display “error please input positive integers only”

The user will enter a salary of type float must be equal to or greater than 0

For bad input: display”error please input positive numbers”

Ask the user if they would like to display data

1. **Decomposition Diagram** (Used to break program down into components visually. Diagram can have as many components as needed. Defines functionality that will solve the problem – does NOT define a flow of actions)

Employee management

Input

process

output

Call appropriate class methods

Display when input is incorrect

Ask user if they would like to go again

Print the data of the employees

User inputs whether they want to display the data enetered

Load information inside of the classes using gets and sets

Check if input is Valid and within limits

User enters employee information by command line

1. **Test Strategy**

* Valid Data
* Invalid Data

1. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User enters age<0 |  |  |  |  |
| Invalid Data | 2 | User enters invalid idnumber |  |  |  |  |
| Valid Data | 3 | User enters correct salary number |  |  |  |  |

1. **Initial Algorithm**

class employee

{

Private

Int age,id

Float salary

employee( )

{ ..... }

void setAge(int ageValue)

{ age = 0

Id =0

Salary=0

}

void setId(int idValue)

{

Id=idValue

}

void setSalary(float salaryValue)

{

Salary=salaryValue

}

};

int employee::getAge( )

{

Return age

}

int employee::getId( )

{

Return id

}

5

float employee::getSalary( )

{

Return salary

}

void main( )

{

// create an array of Employee with two dimensions

Employee emp[2][3]

For int j=0 to 2

For int i=0 to 3

Emp[j][i].setage()

Emp[j][i].setid()

Emp[j][i].setsalary

printEmployee( )

printEmplyee( arg1, arg2, arg3);

end for

end for

end main

void printEmployee(arg1, arg2, arg3)

{

// for loop

....

// print out an average age and an average salary

....

}

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User enters age<0 | -2 | “Error enter positive integers please” |  |  |
| Invalid Data | 2 | User enters invalid idnumber | Jdjie3 | “error reenter only positive integers” |  |  |
| Valid Data | 3 | User enters correct salary number | 30000 | “would you like to display or enter more information?” |  |  |
| Invalid data | 4 | User inputs char for age | J | “reenter a positive integer please” |  |  |
| Invalid data | 5 | User inputs wrong character for the program restart | A | “please enter y or no if you would like to reenter new values” |  |  |

1. **Code**

**///hassan harajly**

**//description::program stores employee information**

**//cis 200 10/21/17**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**//description class that stores employee information**

**class employee {**

**private:int age;**

**int id;**

**float salary;**

**string jobTitle;**

**float timeOnJob;**

**public:**

**employee() { age = 0; id = 0; salary = 0; } // default constructor: age=0, id=0, salary=0**

**//precondition:age integer**

**//post condition private variable new value assigned**

**//description: enters new age for age variable**

**void setAge(int ageValue) {**

**if (age >= 0)//as long as age isnt 0 it will enter new values for age**

**{**

**age = ageValue;**

**}**

**else**

**{**

**age = 0;**

**}**

**employee();**

**} //precondition:sets integer for id val**

**//postcondition:new id value**

**//description sets new id value**

**void setId(int idValue) { id = idValue; employee(); } // let id = idValue**

**//precondition: a float of a yearly salary**

**//postcondition: new private variable data entry**

**//enters salary of given employee**

**void setSalary(float salaryValue) {**

**salary = salaryValue; employee();**

**} // salary = salaryValueint**

**int getAge(); // return age**

**int getId(); // return id**

**float getSalary(); // return salary};**

**void setjobTitle(string title) { jobTitle.assign(title); }//sets the job title given by the user//precond:a string //post condition none**

**string getJobTitle() { return jobTitle; }//returns jobtitle in a string format**

**void settimeOnJob(float timer) { timeOnJob = timer; }//sets time spent on job to private variableprecondition:float postcondition:float denotiing time spent**

**float gettimeOnJob() { return timeOnJob; }//returns time spent on job**

**};**

**int employee::getAge()//implementation of above functions**

**{**

**return age;**

**}**

**int employee::getId() { return id; }//implementation of above functions**

**float employee::getSalary() {**

**return salary;//implementation of above functions**

**}**

**const int arg2 = 2;**

**const int arg3 = 3;//arg2 and arg3 are size of the object array**

**//precondition:object array of type employee integers for size of array**

**//postcondition:printed information**

**//description:/prints all the employees information given by the user**

**{**

**void printemployee(employee \*objarr[arg2][arg3], int ar2, int ar3)**

**{**

**static int z = 0;**

**for (int j = 0; j < arg2; j++)**

**for (int i = 0; i < arg3; i++)**

**{**

**z++;**

**cout << "employee" << z << " age:" << objarr[j][i]->getAge();**

**cout << " employee" << z << " id:" << objarr[j][i]->getId();**

**cout << " employee" << z << " salary:" << objarr[j][i]->getSalary() << endl;**

**}**

**}**

**}**

**//subclass of type employee with diff data**

**class supervisor :public employee**

**{**

**private:**

**int numberOfTeamsSupervised;**

**int numberOfEmployeesSupervised;**

**public:**

**void setteamsupervised(int supervised) { numberOfTeamsSupervised = supervised; }**

**int getsupervised() { return numberOfTeamsSupervised; }**

**void setnumberofemployees(int emp) { numberOfEmployeesSupervised = emp; }**

**int getnumberofemployees() { return numberOfEmployeesSupervised; }**

**};//subclass of type employee with different data**

**class staff :public employee**

**{**

**private:**

**bool teamLeader;**

**string applicationSupported[3];**

**string jobSkill[3];**

**public:**

**void setapplication(string app[3]) {**

**for (int i = 0; i < 3; i++)**

**{**

**applicationSupported[i].assign(app[i]);**

**}**

**}**

**void setjobskill(string jobskil[3]) {**

**for (int i = 0; i < 3; i++)**

**{**

**jobSkill[i].assign(jobskil[i]);**

**}**

**}**

**};**

**void main()**

**{**

**employee companyEmployees[2][3], \*arg1[2][3];//initialize array**

**companyEmployees[0][0].setAge(30); companyEmployees[0][0].setId(111); companyEmployees[0][0].setSalary(30000);**

**companyEmployees[0][1].setAge(31); companyEmployees[0][1].setId(112); companyEmployees[0][1].setSalary(31000);**

**companyEmployees[0][2].setAge(32); companyEmployees[0][2].setId(113); companyEmployees[0][2].setSalary(32000);**

**companyEmployees[1][0].setAge(33); companyEmployees[1][0].setId(114); companyEmployees[1][0].setSalary(33000);**

**companyEmployees[1][1].setAge(34); companyEmployees[1][1].setId(115); companyEmployees[1][1].setSalary(34000);**

**companyEmployees[1][2].setAge(35); companyEmployees[1][2].setId(116); companyEmployees[1][2].setSalary(35000);**

**for (int j = 0; j < arg2; j++)//sets arg1=companyemployees with pointer for easier loading**

**{**

**for (int i = 0; i < arg3; i++)**

**{**

**arg1[j][i] = &companyEmployees[j][i];**

**}**

**}**

**printemployee(arg1, arg2, arg3);//prints data given by the array**

**system("pause");**

**}**

1. **Updated Algorithm**

create class employee which have following attributes  
age of int type, id of int type, salary of float type, jobTitle of string type and timeOnJob of float type.  
these attributes must be private.  
employee class have default constructor which initialize the age, id and salary to its default value.  
Getters and setters for each atttribute.  
the printemployee() takes argument of 2d array of employee type, which print the employee information

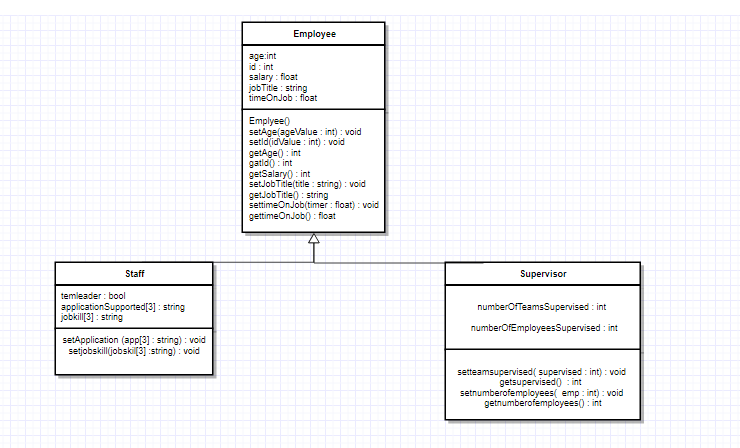
Now create supervisor class which inherits employee  
it has following attribute  
numberOfTeamsSupervised of int type and numberOfEmployeesSupervised of int type  
create getter and setters for each attribute

create class staff which inherit employee  
it has following attributes  
teamLeader of bool type, applicationSupported of string type array and jobskill array of string type

create two methods  
void setapplication(string app[3]) which set the application support  
void setjobskill(string jobskil[3]) which assign the jobskill

create main() to check the functionality of classes created

**Uml:**



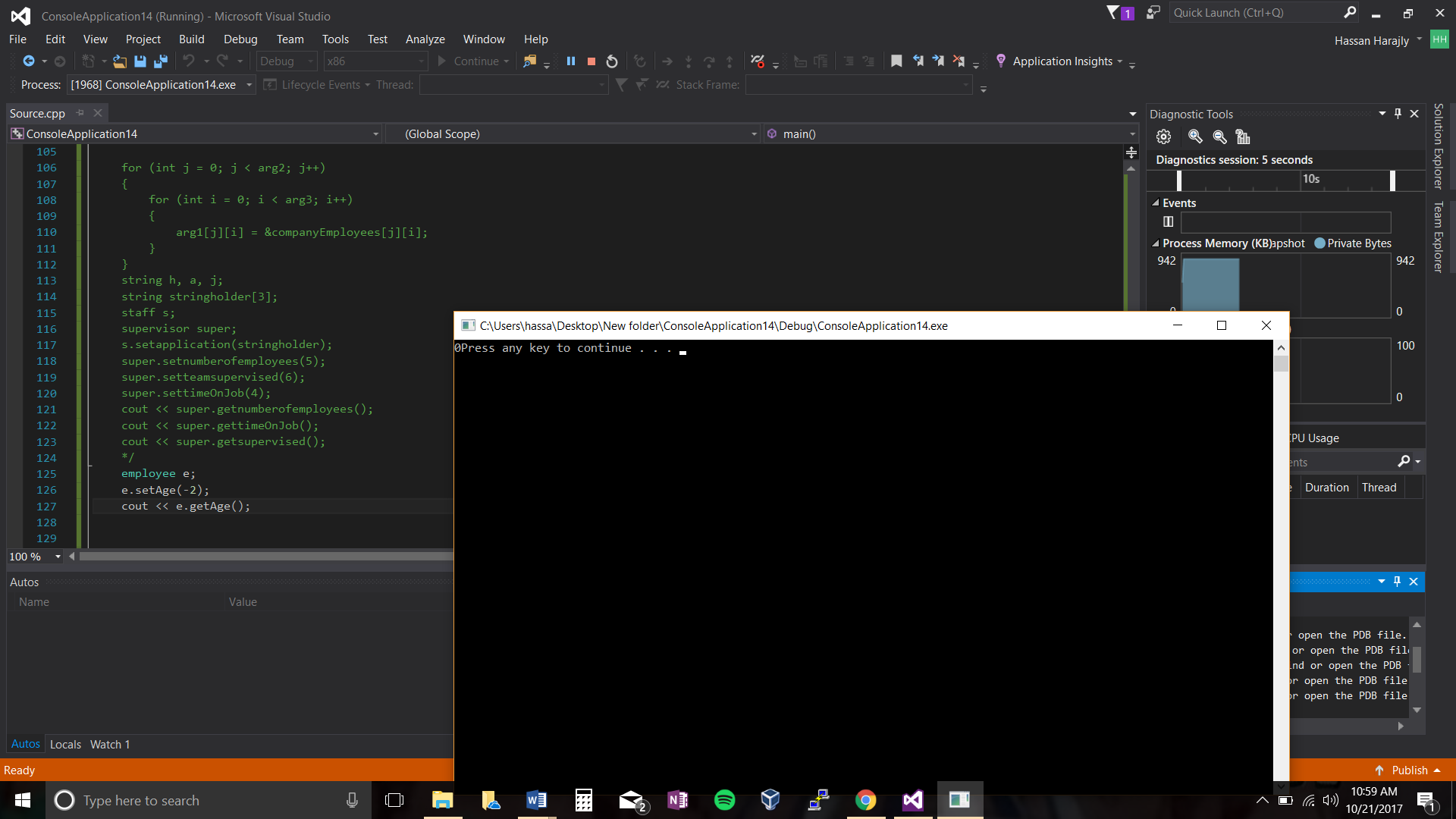
1. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User enters age<0 | -2 | “Error enter positive integers please” | Age is reset to 0(input should be validated before sent to function inside of the class, we only designed stub mains, it depends on how the class is implemented) | pass |
| Invalid Data | 2 | User enters invalid idnumber | Jdjie3 | “error reenter only positive integers” | Compiler error since the id number is only integers thus input should be validated during implementation | pass |
| Valid Data | 3 | User enters correct salary number | 30000 | “would you like to display or enter more information?” | Data is entered without problem | pass |
| Invalid data | 4 | User inputs char for age | J | “reenter a positive integer please” | Compiler error | pass |
| Invalid data | 5 | User inputs wrong character for the program restart | A | “please enter y or no if you would like to reenter new values” | Program will not prompt new user input(program was not designed to do so) | fail |
| Valid data | 6(stub main testing) | Test every method inside of the class to verify it works correctly | See in screen shot section |  |  |  |
| Valid data | 7 | Print function works correctly | None | All employee data | All employee data | pass |

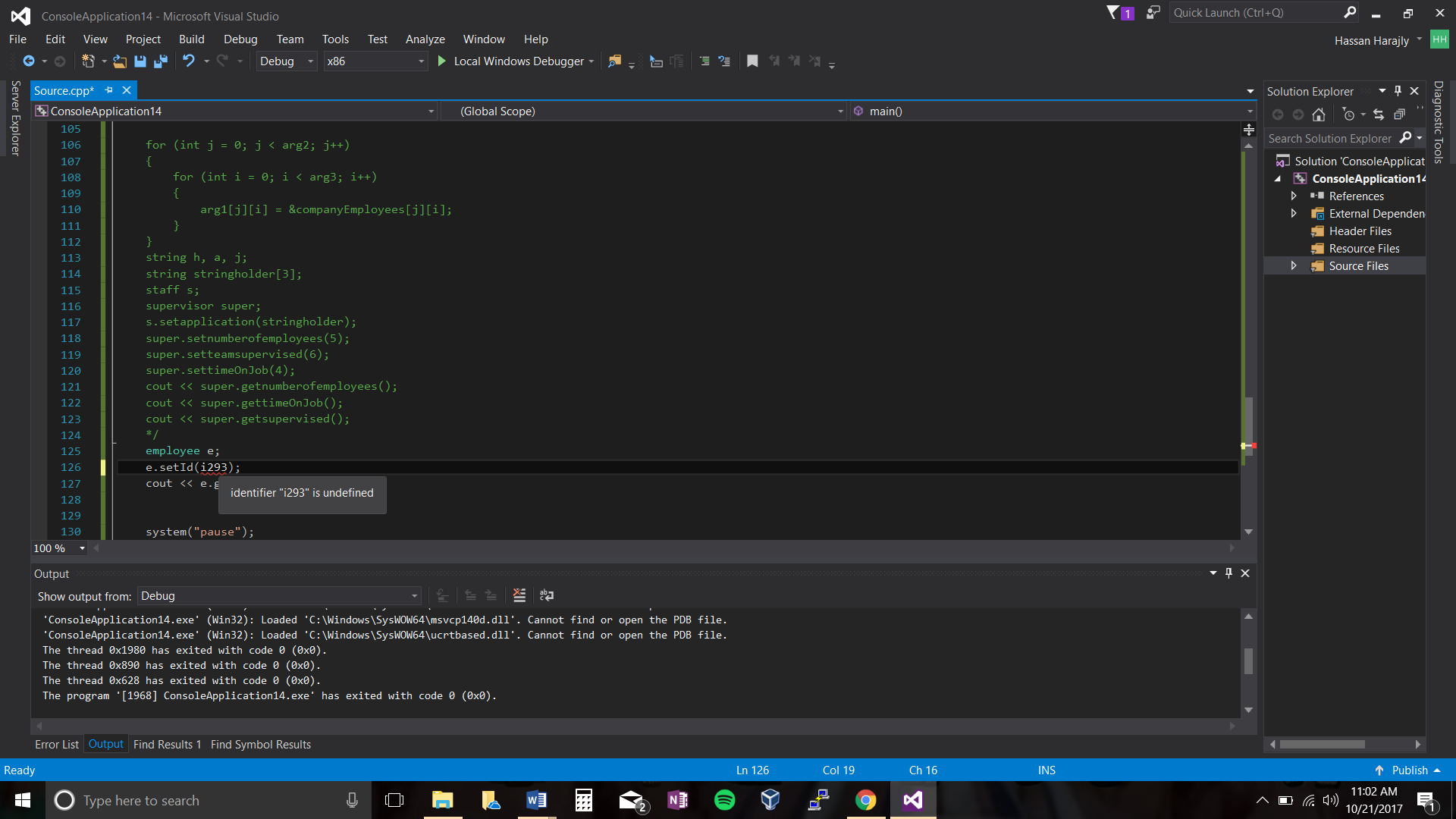
1. **Screenshots**

Test Cases 1-9

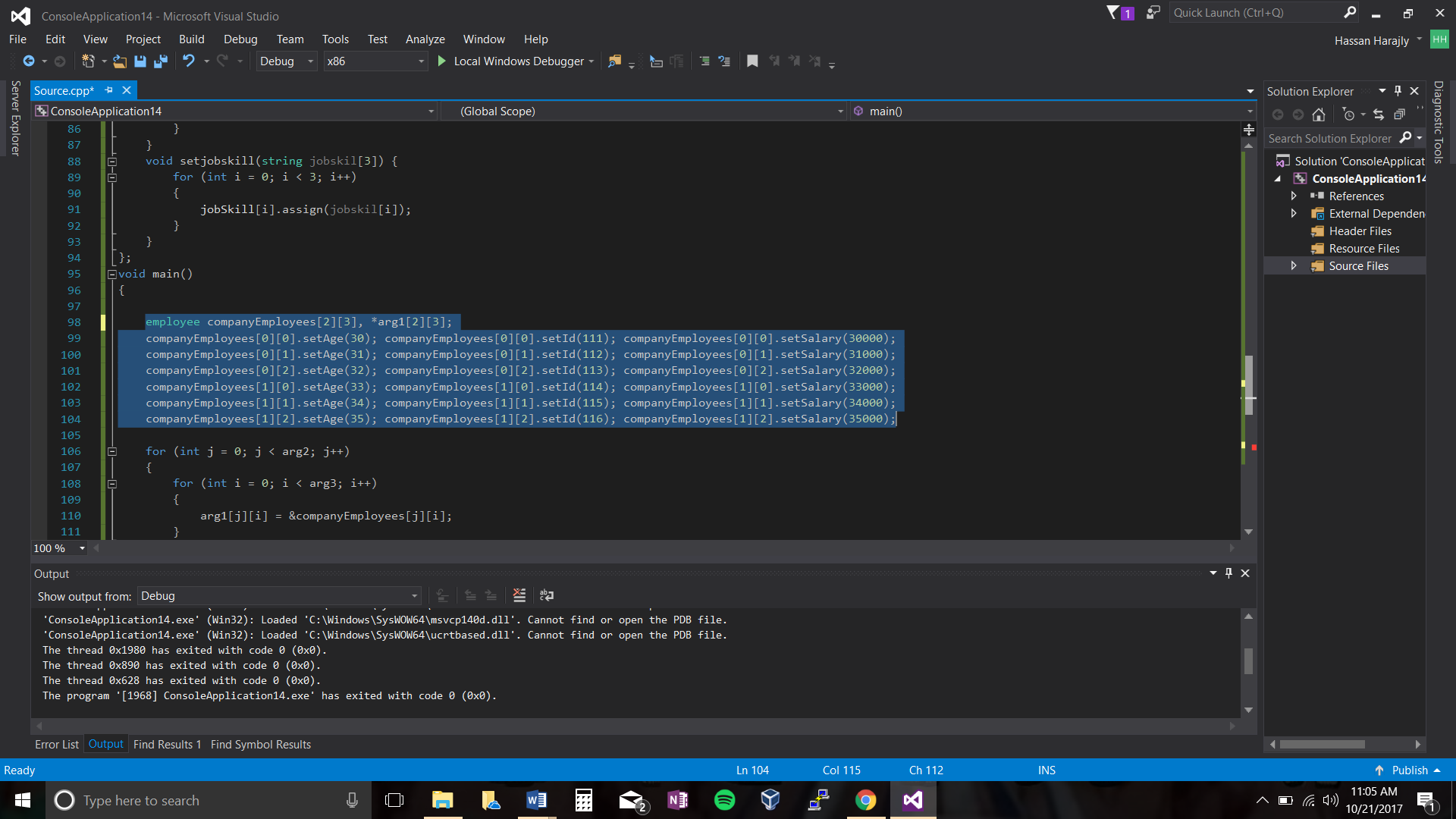
Test case 1:



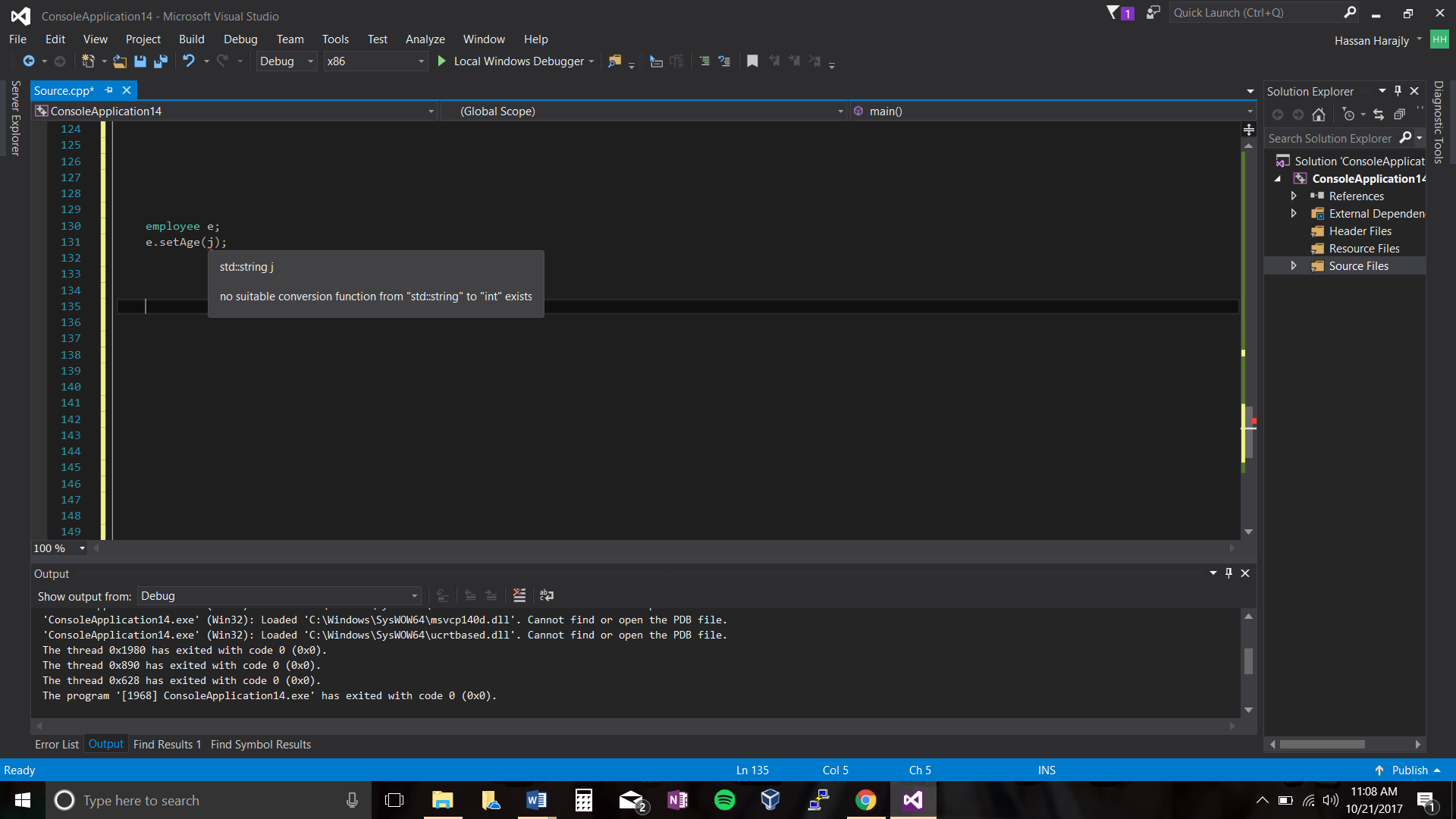
Test case 2:



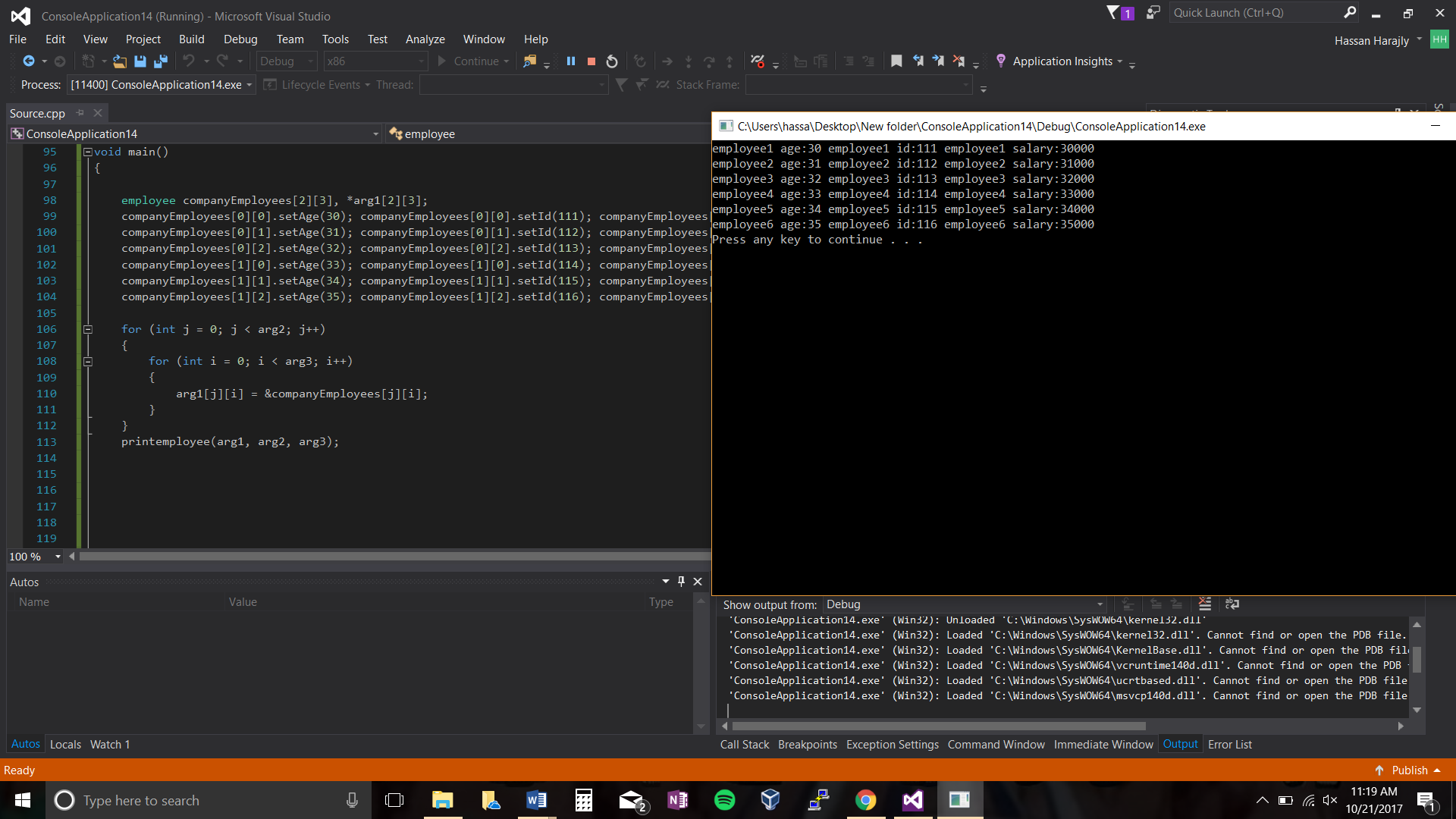
Test case 3:



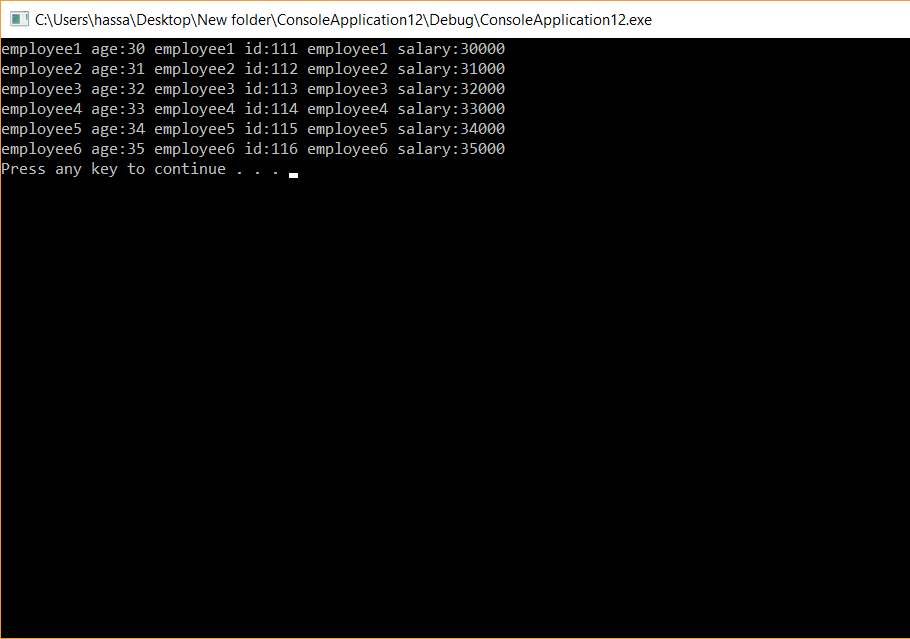
Test case4:

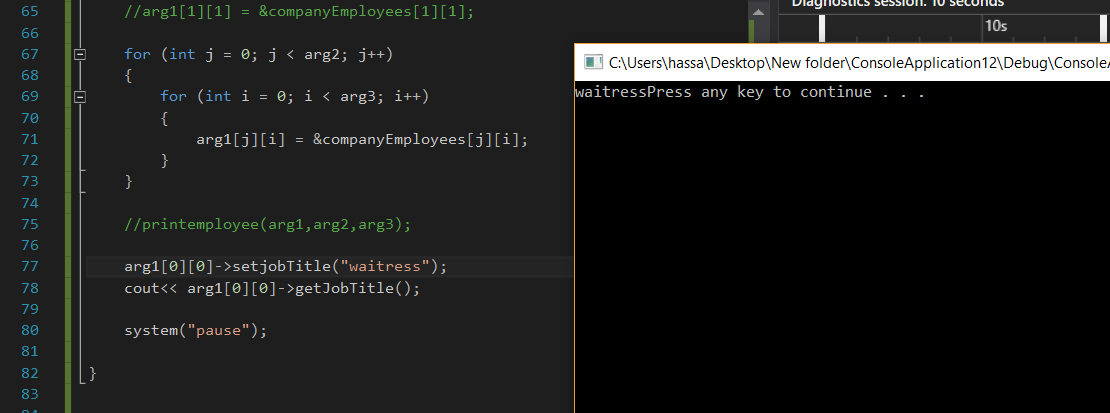


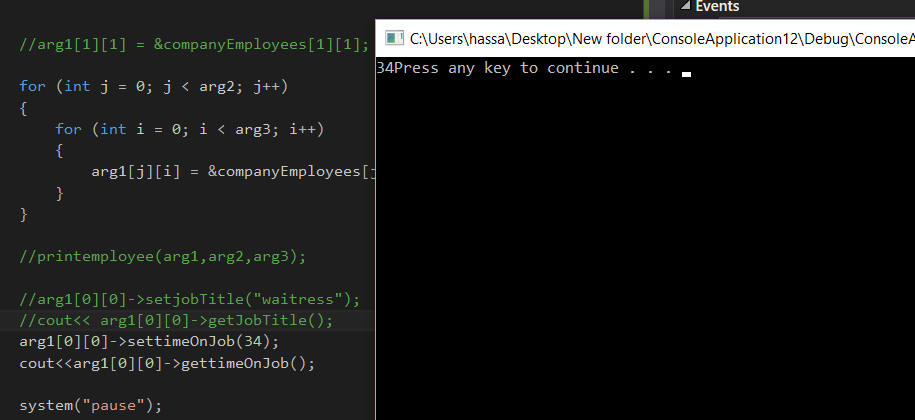
Test case 7:

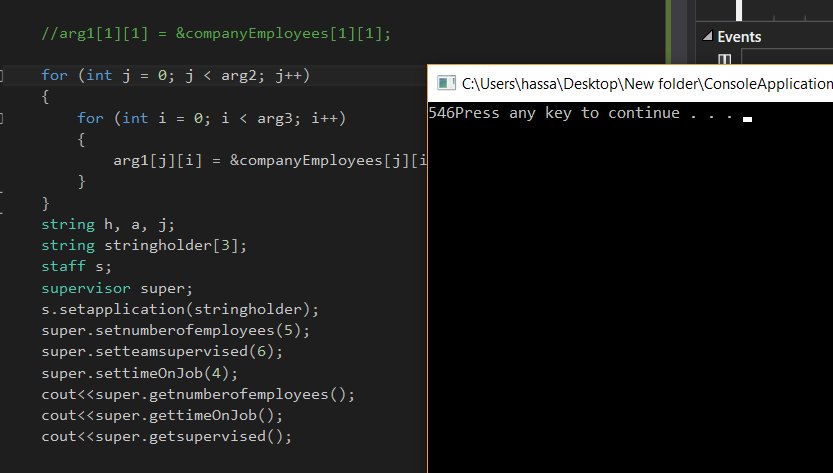


Stub main testing:/test case 6









1. **Error Log**

|  |  |  |
| --- | --- | --- |
| Error Type | Cause of Error | Solution to Error |
|  |  |  |

1. **Status**

The program works 100% with assumptions in place