/\*

Michael Dobachesky

SE 124.12

Program 5

PURPOSE:

You have been asked by your payroll manager to produce a program to search payroll data to determine whether an employee has earned overtime pay.

The manager would like a program which will search data which consists of the employee ID, last name, first name, rate of pay and hours worked.

VARIABLE DICTIONARY:

REPRESENTS TYPE VARIABLE

Create disk? char create\_disk

Enter employees to file char enter\_employee

First run? char first\_run

Match for algorithm char id\_match

Run reply char run\_reply

Totals switch char totals\_switch

Employee ID int employee\_id

ID to search for int employee\_search

Number of overtime employees int overtime\_employees

Max row for algorithm int row\_max

Min row for algorithm int row\_min

Standard work week int std\_work\_week

Subscript number int subscript\_number

First name string first\_name

Last name string last\_name

Hours worked double hours\_worked

Counter for number of rows double max\_row

Overtime hours double overtime\_hours

Overtime pay double overtime\_pay

Overtime rate double overtime\_rate

Rate of pay double rate\_of\_pay

Total overtime paid double total\_overtime\_paid

\*/

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

char create\_disk;

char enter\_employee;

char first\_run;

char id\_match;

char run\_reply;

char totals\_switch;

int employee\_id[100];

int employee\_search;

int overtime\_employees;

int row\_max;

int row\_min;

int std\_work\_week;

int subscript\_number;

string first\_name[100];

string last\_name[100];

double hours\_worked[100];

double max\_row;

double overtime\_hours;

double overtime\_pay;

double overtime\_rate;

double rate\_of\_pay[100];

double total\_overtime\_paid;

void setup();

void build\_file();

void load\_arrays();

void enter\_target();

void search();

void successful();

void unsuccessful();

void totals();

ofstream fout;

ifstream fin;

int main()

{

totals\_switch = 'N';

first\_run = 'Y';

cout << "Do you want to run the Overtime Pay Program? (Y/N) ";

cin >> run\_reply;

run\_reply = toupper(run\_reply);

while (run\_reply != 'Y' && run\_reply != 'N')

{

cout << "Invalid response " << endl;

cout << "Please enter either a Y or an N: ";

cin >> run\_reply;

run\_reply = toupper(run\_reply);

}

system("cls");

while (run\_reply == 'Y')

{

if (first\_run == 'Y')

{

first\_run = 'N';

cout << "Would you like to enter a new list of employees to process? (Y/N) ";

cin >> create\_disk;

create\_disk = toupper(create\_disk);

while (create\_disk != 'Y' && create\_disk != 'N')

{

cout << "Invalid response " << endl;

cout << "Please enter either a Y or an N: ";

cin >> create\_disk;

create\_disk = toupper(create\_disk);

}

system ("cls");

if (create\_disk == 'Y')

{

build\_file();

}

setup();

load\_arrays();

}

enter\_target();

search();

cout << "Do you want to do another search? (Y/N) ";

cin >> run\_reply;

run\_reply = toupper(run\_reply);

while(run\_reply != 'Y' && run\_reply != 'N')

{

cout << "Invalid response " << endl;

cout << "Please enter either a Y or an N: ";

cin >> run\_reply;

run\_reply = toupper(run\_reply);

}

system ("cls");

}

if (totals\_switch == 'Y')

{

totals();

}

return 0;

}

void setup()

{

std\_work\_week = 40;

overtime\_rate = 1.5;

total\_overtime\_paid = 0;

max\_row = 0;

cout.setf(ios::fixed, ios::floatfield);

cout.setf(ios::showpoint);

cout.precision(2);

}

void build\_file()

{

fout.open("employee\_data.txt");

subscript\_number = 0;

enter\_employee = 'Y';

while (enter\_employee == 'Y')

{

cout << "Please enter employee ID: ";

cin >> employee\_id[subscript\_number];

cout << "Please enter employee last name: ";

cin >> last\_name[subscript\_number];

cout << "Please enter employee first name: ";

cin >> first\_name[subscript\_number];

cout << "Please enter employee rate of pay: ";

cin >> rate\_of\_pay[subscript\_number];

cout << "Please enter hours employee worked: ";

cin >> hours\_worked[subscript\_number];

fout << employee\_id[subscript\_number] << " " << last\_name[subscript\_number] << " " << first\_name[subscript\_number] << " " << rate\_of\_pay[subscript\_number] << " " << hours\_worked[subscript\_number] << " " << endl;

subscript\_number = subscript\_number + 1;

cout << "Would you like to enter another employee? (Y/N) ";

cin >> enter\_employee;

enter\_employee = toupper(enter\_employee);

while (enter\_employee != 'Y' && enter\_employee != 'N')

{

cout << "Invalid response " << endl;

cout << "Please enter either a Y or an N: ";

cin >> enter\_employee;

enter\_employee = toupper(enter\_employee);

}

system ("cls");

}

fout.close();

}

void load\_arrays()

{

fin.open("employee\_data.txt");

subscript\_number = 0;

max\_row = max\_row + 1;

fin >> employee\_id[subscript\_number] >> last\_name[subscript\_number] >> first\_name[subscript\_number] >> rate\_of\_pay[subscript\_number] >> hours\_worked[subscript\_number];

while(!fin.eof())

{

subscript\_number = subscript\_number + 1;

max\_row = max\_row + 1;

fin >> employee\_id[subscript\_number] >> last\_name[subscript\_number] >> first\_name[subscript\_number] >> rate\_of\_pay[subscript\_number] >> hours\_worked[subscript\_number];

}

fin.close();

}

void enter\_target()

{

cout << "Please enter an employee ID to look up: ";

cin >> employee\_search;

}

void search()

{

id\_match = 'N';

row\_min = 0;

row\_max = max\_row;

subscript\_number = 0;

while (id\_match == 'N' && row\_min <= row\_max)

{

subscript\_number = int((row\_min + row\_max) / 2);

if (employee\_search < employee\_id[subscript\_number])

{

row\_max = subscript\_number - 1;

}

else

{

if (employee\_search > employee\_id[subscript\_number])

{

row\_min = subscript\_number + 1;

}

else

{

id\_match = 'Y';

}

}

}

if (id\_match == 'Y')

{

successful();

}

else

{

unsuccessful();

}

}

void successful()

{

if (hours\_worked[subscript\_number] > std\_work\_week)

{

totals\_switch = 'Y';

overtime\_hours = hours\_worked[subscript\_number] - std\_work\_week;

overtime\_pay = overtime\_hours \* rate\_of\_pay[subscript\_number] \* overtime\_rate;

overtime\_employees = overtime\_employees + 1;

total\_overtime\_paid = total\_overtime\_paid + overtime\_pay;

cout << first\_name[subscript\_number] << " " << last\_name[subscript\_number] << " " << "has worked " << overtime\_hours << " hours of overtime and is eligible for $" << overtime\_pay << " of overtime pay" << endl;

}

else

{

cout << first\_name[subscript\_number] << " " << last\_name[subscript\_number] << " " << "is not eligible for overtime pay" << endl;

}

}

void unsuccessful()

{

cout << "The employee ID was not located " << endl;

}

void totals()

{

cout << "Total number of searched employees with overtime pay: " << overtime\_employees << endl;

cout << "Total amount of overtime paid to those employees: $" << total\_overtime\_paid << endl;

system("pause");

}