Michelle Zulli

Dr. Alireza Ebrahimi

Introduction to C++ and OOP

August 28, 2015

# Module 7: Payroll

The program was modified to use a new sort that is a combination of an exchange sort and a selection sort (exsel sort). In addition, I modified the program so that it works with only one input file instead of separate files for hourly and salary employees. This is the way the program was working prior to Module 6. I also went through the program and updated the names of classes, variables, and functions to be more consistent within the program.

## Code

#include<iostream>

#include<fstream>

#include<iomanip>

using namespace std;

class Employee {

public:

ifstream fin;

string firstName, lastName;

int id;

float hours, baseHours, otHours, baseRate, otRate, gross, net;

float taxAmt, taxRate;

char status;

// set base and ot hours

void findHours() {

baseHours = hours;

otHours = 0;

if (baseHours > 40) {

otHours = baseHours - 40;

baseHours = 40;

}

} // findHours

// set ot rate

virtual void findOtRate() {}

// calculate gross pay

virtual void findGross() {}

// calculate tax amount

void findTaxAmt() {

taxAmt = gross \* taxRate;

} // findTaxAmt

// calculate net pay

void findNet() {

net = gross - taxAmt;

} // findNet

// print table headers

void printHeader() {

cout << setiosflags(ios::left)

<< " ZULLI PAYROLL"

<< endl

<< setw(16) << "FIRST NAME" << setw(16) << "LAST NAME"

<< setw(8) << "STATUS" << setw(6) << "ID" << setw(8) << "HOURS"

<< setw(10) << "OT HOURS" << setw(10) << "RATE"

<< setw(10) << "OT RATE" << setw(10) << "GROSS"

<< setw(10) << "TAX" << setw(10) << "NET"

<< endl

<< "=============== =============== ======= ===== ======= "

<< "========= ========= ========= ========= ========= ========="

<< endl;

} // printHeader

// print data for Employee

void printData() {

cout << setprecision(2)

<< setiosflags(ios::showpoint | ios::fixed | ios::left)

<< endl

<< setw(16) << firstName << setw(16) << lastName

<< setw(8) << status << setw(6) << id

<< setw(8) << hours << setw(10) << otHours << setw(10) << baseRate

<< setw(10) << otRate << setw(10) << gross

<< setw(10) << taxRate << setw(10) << net

<< endl;

} // printData

Employee() {

taxRate = .3;

fin.open("employees.in");

}

~Employee() {

fin.close();

}

};

class Hourly: public Employee {

public:

// set ot rate

void findOtRate() {

otRate = baseRate \* 1.5;

} // findOtRate

// calculate gross pay

void findGross() {

gross = (baseHours \* baseRate) + (otHours \* otRate);

} // findGross

Hourly() {

// fin.open("hourly.in");

}

~Hourly() {

// fin.close();

}

};

class Salary: public Employee {

public:

// set ot rate

void findOtRate() {

otRate = baseRate / 52 / 40 \* 1.5;

} // findOtRate

// calculate gross pay

void findGross() {

gross = baseRate / 52 + (otHours \* otRate);

} // findGross

Salary() {

// fin.open("salary.in");

}

~Salary() {

// fin.close();

}

};

int main() {

Employee \*empl = new Employee;

Employee \*staff[6];

for (int i = 0; i < 6; i++) {

empl->fin >> empl->status;

if (empl->status == 'h' || empl->status == 'H') {

staff[i] = new Hourly();

}

else {

staff[i] = new Salary();

}

staff[i]->status = empl->status;

empl->fin >> staff[i]->firstName >> staff[i]->lastName >> staff[i]->id

>> staff[i]->hours >> staff[i]->baseRate;

staff[i]->findHours();

staff[i]->findOtRate();

staff[i]->findGross();

staff[i]->findTaxAmt();

staff[i]->findNet();

}

// print table header

empl->printHeader();

// sort pointer array with ExSel sort

int lower, upper, sml = 0, scan = 0;

lower = 0;

upper = 5;

while (lower < upper) {

sml = lower;

scan = lower;

while (scan < upper) {

if (staff[scan]->net > staff[scan + 1]->net) {

Employee \*temp = staff[scan];

staff[scan] = staff[scan + 1];

staff[scan + 1] = temp;

if (staff[scan]->net < staff[sml]->net) sml = scan;

}

scan++;

}

Employee \*temp = staff[lower];

staff[lower] = staff[sml];

staff[sml] = temp;

upper = upper - 1;

lower = lower + 1;

}

// print Employee data

for (int i = 0; i < 6; i++) {

staff[i]->printData();

}

// print highest and lowest net

cout << endl;

cout << "LOWEST NET : $ " << staff[0]->net << " (" << staff[0]->firstName

<< " " << staff[0]->lastName << ")" << endl;

cout << "HIGHEST NET: $" << staff[5]->net << " (" << staff[5]->firstName

<< " " << staff[5]->lastName << ")" << endl;

return 0;

} // MAIN

## Output

