Edgardo Richard Ventura (Eddie)
Frank Alvino
CIS121 061 - Introduction to Programming
3/31/2024

CIS 121 Introduction to Programming
Problem Set 11 – Reading Data from a File

1. Read in auto make, model, MSRP (manufacturer suggested resale price) and sales price from a file. See auto.txt below. For each auto (line of data) compute the savings to be MSRP – sales price and sales tax in a single function.

   Pass to the function MSRP, sales price by value, sales price and sales tax by reference. The function will:
   1. Compute savings = MSRP – Sales Price
   2. Compute sales tax = Sales Price * 0.07

   For each auto (line of data) display make, model, MSRP, sales price, savings and sales tax. Sum the savings. After all autos (lines of data processed) display sum of savings.

| Auto.txt | | |
|---|---|---|
| Honda Accord | 25000.00 | 22000.00 |
| Honda CRV | 30000.00 | 28000.00 |
| Toyota Corolla | 28000.00 | 25000.00 |
| Ford Fusion | 23000.00 | 20000.00 |

| ***Input*** | ***Process*** | ***Output*** |
|---|---|---|
| File: **auto.txt** | • Read each line from the file.<br>• Compute savings as MSRP - Sales Price.<br>• Compute sales tax as Sales Price * 0.07.<br>• Sum the savings across all entries. | Make<br>Model<br>MSRP<br>Sales Price<br>Savings<br>Sales Tax |
| Make<br>Model<br>MSRP<br>Sales Price | | Sum of savings processed after all lines. |

| ***Name*** | ***Etymology*** |
|---|---|
| msp | Manufacturer Suggested Retail Price (MSRP) |
| sp | Sales Price |
| sv | Savings from MSRP |
| st | Sales Tax on sales price |
| ts | Total Savings across all entries |

2. Read in the grocery item, quantity and cost per item from a file. See grocery.txt below. For each item determine the extended price (quantity * cost per item) within a function.

   Pass to the function quantity and price. The function will:
   1. Compute extended price = quantity * price
   2. Return extended price

   For each item (line of data) display item, quantity, cost per item and extended price.
   Also, sum the extended price.
   After all lines of data are processed, compute tax of 7% (sum of extended price * 0.07). Compute total receipt to be sum of extended price + tax. Display sum of extended price, tax and total receipt.

| Grocery.txt | | |
|---|---|---:|
| Banana | 10 | 0.50 |
| Oranges | 5 | 0.10 |
| Cookies | 15 | 0.30 |
| Hamburgers | 4 | 1.00 |
| Buns | 4 | 0.50 |
| Soda | 10 | 0.99 |

| ***Input*** | ***Process*** | ***Output*** |
|---|---|---|
| File: **grocery.txt** | • Read each line from the file.<br>• Compute extended price as Quantity * Cost per Item.<br>• Sum the extended prices to get total extended price.<br>• Calculate sales tax as the total extended price * 0.07.<br>• Calculate total receipt as total extended price + sales tax. | Item<br>Quantity<br>Cost per Item<br>Extended Price |
| Item<br>Quantity<br>Cost per Item | | At the end;<br>total extended price<br>sales tax<br>total receipt. |

| ***Name*** | ***Etymology*** |
|---|---|
| pty | Quantity of grocery items |
| cpi | Cost Per Item |
| ep | Extended Price |
| totalEP | Total Extended Price |
| tax | Sales Tax on total extended price |
| totalrec | Total Receipt including sales tax |

3. Read gallons of gas used and miles travelled from a file. See trips.txt below. For each trip compute miles per gallon within a function.

   Pass to the function gallons and miles. Within the function:
   1. Compute mpg = miles / gallons.
   2. Return mpg.

   For each trip (line of data) display gallons, miles and mpg.
   Sum miles travelled and gallons of gas used.
   After all data is processed (end of file) display sum of gallons and sum of miles travelled.

| Trips.txt | |
|---|---|
| 20 | 500 |
| 10 | 600 |
| 5 | 50 |
| 10 | 350 |

| ***Input*** | ***Process*** | ***Output*** |
|---|---|---|
| File: **trips.txt** | • Read each line from the file.<br>• Compute miles per gallon as Miles / Gallons. | Gallons Used<br>Miles Traveled<br>Miles Per Gallon |
| Gallons Used<br>Miles Traveled | • Sum total miles and total gallons over all trips. | At the end:<br>total gallons<br>total miles<br>total miles/gallon |

| ***Name*** | ***Etymology*** |
|---|---|
| gal | Gallons used on the trip |
| mi | Miles traveled on the trip |
| mpg | Miles Per Gallon |
| totalGal | Total Gallons used |
| totalMi | Total Miles traveled |
| totalMpg | Total Miles Per Gallon |

4 Read in last name and annual salary from a file. See empl.txt below. For each employee (line of data) compute their bi-weekly pay within a function.

Pass to the function the annual salary. The function will:

1. Compute bi-weekly salary = annual salary / 26
2. Return bi-weekly salary

For each line display last name, annual salary and bi-weekly salary. Sum the annual salaries. Count the number of employees (each line of data).
After all employees processed compute average annual salary (total annual salary / count of employees). Display sum of annual salary, count of employees and average annual salary.

| Empl.txt | |
|---|---|
| Jones | 50000.00 |
| Adams | 65000.00 |
| Baker | 45000.00 |
| Smith | 75000.00 |

| *Input* | *Process* | *Output* |
|---|---|---|
| File:<br>**empl.txt** | • Read each line from the file.<br>• Compute monthly, bi-weekly, weekly, and hourly wages from the annual salary.<br>• Sum all annual salaries.<br>• Count number of employees. | Last Name<br>Annual Salary<br>Monthly Pay<br>Bi-weekly Pay<br>Weekly Pay<br>Hourly Wage |
| Last Name<br>Annual Salary | | At the end:<br>Total all annual salaries<br>count of employees<br>average annual salary |

| *Name* | *Etymology* |
|---|---|
| ln | Last name of the employee |
| 1pay | Annual salary |
| 12pay | Monthly salary estimate |
| 21pay | Bi-weekly salary |
| 52pay | Weekly salary estimate |
| per/hr | Hourly wage estimate |
| total | Total of all annual salaries |
| employeeCount | Number of employees processed |
| averageSalary | Average annual salary |

5 The input to this program is last name, student code and credits taken. See file student.txt below.

You are to read each line of data into your program (one line at a time of course). Write a function to calculate tuition owed and course fees in a single function. You must pass to the function student code and credits taken by value and tuition owed and course fees by reference.
Within the function compute tuition owed and course fees:

1. Determine cost per credit hour. In district students (student code == 'I') pay $250.00 per credit hour. Out of district students (student code == 'O') pay $500.00 per credit hour.
2. Calculate tuition owed to be credit hours * cost per credit hour
3. Compute course fees as 10% of the total tuition.

Also, within the program sum the tuition owed for all students. Count the number of students. For each student (line of data), display last name, student code, cost per credit hour, credits taken, tuition owed and course fees. Additionally, sum the tuition owed.
After all students are processed (end of loop), calculate the average tuition cost per student (sum of tuition owed / number of students). Display sum of tuition owed, number of students and average tuition cost per student.

| Student.txt | | |
|---|---|---|
| Jones | I | 15.00 |
| Baker | O | 20.00 |
| Davis | I | 10.00 |
| Michaels | O | 12.00 |
| Baez | I | 12.00 |

| Input | Process | Output |
|---|---|---|
| File:<br>**student.txt** | • Read each line from the file.<br>• Compute tuition owed based on credits and in-district or out-district rates.<br>• Compute course fees as 10% of tuition.<br>• Sum tuition owed and count students. | Last Name<br>Student Code<br>Credits Taken<br>Tuition Owed<br>Course Fees |
| Last Name<br>Student Code<br>Credits Taken | | At the end:<br>total tuition owed<br>number of students<br>avg tuition/student |

| *Name* | *Etymology* |
|---|---|
| ln | Last name of the student |
| sc | Student code ('I' or 'O') |
| ch | Credits taken |
| tu | Tuition owed |
| cf | Course fees |
| total | Total tuition owed |
| studentCount | Number of students processed |
| averageTuition | Average tuition per student |