Kelly Williams

12/11/2023

ECE 1310.04

Homework 13

1. 6.10 *(Salesperson Salary Ranges)* Use a one-dimensional array to solve the following problem. A company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9 percent of $5000, or a total of $650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson’s salary is truncated to an integer amount):

1. $200 – 299
2. $300 – 399
3. $400 – 499
4. $500 – 599
5. $600 – 699
6. $700 – 799
7. $800 – 899
8. $900 – 999
9. $1000 and over

A screenshot of a computer program

Description automatically generated

//System Libraries

#include <iostream>

//Function Prototypes

int SalaryRange(int);

void DisplayArray(int[], int);

//Global Constants

//Main Function

using namespace std;

int main(int argc, char\*\* argv)

{

int range[9] = { 0,0,0,0,0,0,0,0,0 }; //array of counters for salary ranges

cout << "=== Welcome to Salesperson Salary Ranges! ===\n";

int salary, sales=0;

while (sales >= 0)

{

cout << "\n(-1 to quit)\nSalesperson's gross sales this week: ";

cin >> sales;

if (sales == -1)

{

cout << "\ngoodbye!\n";

break;

}

salary = 200 + (sales \* 0.09); //200 per week + 9% of sales

cout << endl << "salary: " << salary << endl;

range[SalaryRange(salary)] += 1;

}

cout << "\nSalary Range\tSalespeople\n";

DisplayArray(range, 9);

return 0;

}

int SalaryRange(int salary)

{

if (salary <= 299)

return 0;

if (salary >= 300 && salary <= 399)

return 1;

if (salary >= 400 && salary <= 499)

return 2;

if (salary >= 500 && salary <= 599)

return 3;

if (salary >= 600 && salary <= 699)

return 4;

if (salary >= 700 && salary <= 799)

return 5;

if (salary >= 800 && salary <= 899)

return 6;

if (salary >= 900 && salary <= 999)

return 7;

if (salary >= 1000)

return 8;

}

void DisplayArray(int range[], int size)

{

int min = 200;

int max = 299;

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

{

cout << "$" << min << " - " << max << "\t" << range[i] << endl;

min += 100;

max += 100;

}

cout << "$1000 and over\t" << range[size-1] << endl;

}

2. 6.12 *(Bubble Sort Enhancements)* The bubble sort described in Exercise 6.11 is inefficient for large arrays. Make the following simple modifications to improve the performance of the bubble sort:

1. After the first pass, the largest number is guaranteed to be in the highest-numbered element of the array; after the second pass, the two highest numbers are “in place,” and so on. Instead of making nine comparisons on every pass, modify the bubble sort to make eight comparisons on the second pass, seven on the third pass, and so on.
2. The data in the array may already be in the proper order or near-proper order, so why make nine passes if fewer will suffice? Modify the sort to check at the end of each pass if any swaps have been made. If none have been made, then the data must already be in the proper order, so the program should terminate. If swaps have been made, then at least one more pass is needed.

A screenshot of a computer

Description automatically generated

//System Libraries

#include <iostream>

//Function Prototypes

void DisplayArray(int[], int);

void BubbleSort(int[], int);

//Global Constants

//Main Function

using namespace std;

int main(int argc, char\*\* argv)

{

const int size = 10;

int arr[size];

for (int i = 0; i < size; i++)

{

arr[i] = rand() % 10; //fills the array with random ints from 1-10

}

cout << "The unsorted array:\n";

DisplayArray(arr, size);

//do the modified bubble sort

BubbleSort(arr, size);

cout << "\nThe sorted array:\n";

DisplayArray(arr, size);

return 0;

}

//display 1 dimensional array

void DisplayArray(int x[], int size)

{

cout << "\nIndex\tElement\n";

for (int i = 0; i < size; i++)

cout << " " << i << "\t " << x[i] << endl;

cout << endl;

}

//bubble sort

void BubbleSort(int x[], int size)

{

int temp;

int a = 1;

int b = 0; //checks for swaps

for (int j = 1; j < size; j++)//outer loop keeps looping until all values are in order

{

for (int i = 0; i < size - a; i++)//passes through array

{

if (x[i] > x[i + 1])//compares values

{

b++; //swap count increase

//swap values

temp = x[i];

x[i] = x[i + 1];

x[i + 1] = temp;

}

}

//part a

a++; //reduces the "size" of the array for the next pass since the "highest" relative value is in the correct spot after each pass

//part b

//must check b before resetting to 0

if (b == 0) //if no swap has been made

break; //end loop

b = 0; //reset b after swap

}

}

3. 6.14 Find the error(s) in each of the following statements:

1. Assume that: int a[3];

cout << a[1] << “ “ << a[2] << “ “ << a[3] << endl;

The error is that the index goes up to 3 when it should only go up to two (a[0], a[1], a[2]).

1. double f[3] = {1.1, 10.01, 100.001, 1000.0001};

the size of f is only 3 elements and it is initialized with 4 elements (too many)

1. Assume that: double d[2][10];

d[1, 9] = 2.345;

syntax error – should be d[1][9] =2.345;

3. 6.18 What does the following program do?

A white paper with black text

Description automatically generatedthe whatIsThis function will return the only element of the array if the array size is one, or will return each element of the array summed backwards recursively as seen in line 24.

In this example, the function would return:

b[9]+b[8]+b[7]+b[6]+b[5]+b[4]+b[3]

+b[2]+b[1]+b[0]

= 10+9+8+7+6+5+4+3+2+1 = 55

The function would return 55.