Şafak Akıncı 18.10.2016
152120141003

Experiment 1
Introduction to C++ Programming – I

Objectives

To write simple computer programs in C++.

Prelab Activities

Programming Output

1-63

2-39

3-Please enter an integer:
```
Hello
```

4- Please enter an integer:
```
Goodbye
```

5- Please enter an integer:
```
Hello
Goodbye
```

6- Goodbye

7- Hello!

8- The value of x is: 2
```
The value of x is: 3
The value of x is: 4
The value of x is: 5
The value of x is: 6
The final value of x is: 6
```

9- The student with a grade of 85 Passed

The student with a grade of 55 Failed

10- 2 1 -8 -1

11- none one two three four

12- AC

Correct The Code

13- 1 : it isn't a program statement, it is preprocessor (#) **syntax error** : `#include <iostream>`
2 : missing ";" **syntax error** : `using namespace std;`

14- 2 : input should be assigned to different from -1 **logic error** : `int input = 0;`

15- : **no error** :

Lab Exercises

Lab Exercise 1 – Random Data Generation, Mean, Variance and Standart Deviation Calculation

```
/*****************************************
 * OOP1_Lab_1.cpp                          *
 *****************************************
 * IDE : Xcode                             *
 * Author : Şafak AKINCI                   *
 * Experiment 1: Introduction to C++ - I   *
 *****************************************/
#include <iostream>
#include <time.h>                  //to call rand() function.
#include <iomanip>                 //to call setw() function.
#include <string>                  //to declare string variables.
#include <math.h>                  //to call sqrt() function.
using namespace std;               //to don't write for each code std:: (e.x. std::cout )

bool TakeUserInput(int& data_size, int& min_number, int& max_number);
int* CreateDataSet(int dataSize, int min_number, int max_number);
double CalculateMean(int* data_array, int data_size);
double CalculateVariance(int* data_array, int data_size, double mean);
double CalculateStandartDeviation(int* data_array, int data_size, double mean);
double* CalculateZScore(int* data_array, int data_size, double mean, double std_deviation);
void PrintMessage(const string& message);
void PrintDataArray(int* data_array, int data_size);
void PrintMean(double mean);
void PrintVariance(double variance);
void PrintStandartDeviation(double std_deviation);
void PrintZScoreArray(double* zscore, int data_size);
double* CalculateTScore(int* data_array, int data_size, double mean, double std_deviation);
void PrintTScoreArray(double* tscore, int data_size);

int main() {
    //To don’t get the same numbers when the rand() function is called.
    srand( (unsigned) time(NULL));

    int dataSize, minNumber, maxNumber;

    //To check whether user enters an input or not.
    if (!TakeUserInput(dataSize, minNumber, maxNumber)){
        PrintMessage("TERMINATED BY USER!");
        return 1;
    }

    int* DataArray = CreateDataSet(dataSize,minNumber,maxNumber);
    //The dynamic array's address that is returned from CreateDataSet function is assigned to pointer called DataArray.
    PrintDataArray(DataArray, dataSize);
    //All elements of the DataArray is printed to console.
    double mean = CalculateMean(DataArray,dataSize);
    //Mean is calculated by the CalculateMean function and it is assigned the double variable called mean.
    PrintMean(mean);
    //mean is printed to console.

    double variance = CalculateVariance(DataArray, dataSize, mean);
    //Variance is calculated by the CalculateVariance function and it is assigned the double variable called variance.
    PrintVariance(variance);
    //variance is printed to console.

    double stdDeviation = CalculateStandartDeviation(DataArray,dataSize, mean);
    //Standart deviation is calculated by the CalculateStandartDeviation function and it is assigned the double variable called stdDeviation.
    PrintStandartDeviation(stdDeviation);
    //stdDeviation is printed to console.

    double* zscore = CalculateZScore(DataArray, dataSize, mean,stdDeviation);
    //The dynamic array's address that is returned from CalculateZScore function is assigned to pointer called zscore.
    PrintZScoreArray(zscore, dataSize);
    //All elements of zscore is printed to console.

    double* tscore = CalculateTScore(DataArray, dataSize, mean, stdDeviation);
    //The dynamic array's address that is returned from CalculateTScore function is assigned to pointer called tscore.
    PrintTScoreArray(tscore, dataSize);
    //All elements of tscore is printed to console.

        return 0;
}//end main


//TakeUserInput will take three input from the user, data_size, min_number, max_number
//and return true to the main function if the data_size is different from 0.
bool TakeUserInput(int& data_size, int& min_number, int& max_number)
{
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

//  There is 2 "|" and "USER INPUT" 10.                 10+2=12;
//  26-12=14;   14/2=7;     7 + 10(USER INPUT) = 17;   To center "USER INPUT", setw(17)
    cout<<"|"<<setw(17)<<"USER INPUT"<<setw(9)<<"|\n";
    //  |       USER INPUT       |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
    cout<<"Please, enter the data_size! Enter 0 to terminate:\t\t";
    cin>>data_size;

    //If data size is equal to zero, program will terminate and function will return zero.
    if(data_size==0)
        return false;

    cout<<"Please, enter the min_number:\t";
    cin>>min_number;

    cout<<"Please, enter the max_number:\t";
    cin>>max_number;
    return true;
}    //end TakeUserInput ( )


//PrintMessage will print the const string variable called "message" to console.
void PrintMessage(const string& message){
    cout<<endl<<message<<endl;
}     //end PrintMessage ( )


//CreateDataSet will create an array that generated by random numbers between the min_number and max_number.
int* CreateDataSet(int dataSize, int min_number, int max_number){

    //      If we declare an array it will create in STACK, we will lost its elements after the CreateDataSet function ended.
    //      We should create a dynamic array to reach its elements and address from another functions even if CreateDataSet function ends.
    //      Because the array will be created in HEAP, so we won't lost its elements.
    int* array = new int [dataSize];

    for(int i=0; i<dataSize; i++)
        array[i] = min_number + rand() % (max_number - min_number +1);
    //Random numbers between min_number and max_number is assigned to dynamic array.

    //The first element's address of the dynamic array is returned to the main function.
    return array;
}    //end CreateDataSet ( )


//PrintDataArray will print the whole elements of the array.
//It takes one pointer that holds the first element's address of the array and one integer value called data_size.
void PrintDataArray(int* data_array, int data_size){
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
            else if(i==25)  cout<<"+\n";
                else    cout<<"-";
    }// +------------------------+
    //  There is 2 "|" and the size of the "DATA ARRAY" 10.           10+2=12;
    //  26-12=14;   14/2=7;     7 + 10(DATA ARRAY) = 17;    To center "DATA ARRAY", setw(17)
    cout<<"|"<<setw(17)<<"DATA ARRAY"<<setw(9)<<"|\n";
    //  |       DATA ARRAY       |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

    for(int i=0; i<data_size; i++)
        cout<<"|"<<setw(24)<<data_array[i]<<"|"<<endl;
// | i|

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

}    //end PrintDataArray ( )


//CalculateMean calculates the mean of the array and returns it to main function.
//It takes one pointer that holds the first element's address of the array and one integer value called data_size.
double CalculateMean(int* data_array, int data_size){

    double total=0;

    //Whole elements of the array will be totalled.
    for(int i=0; i<data_size; i++)
        total+=data_array[i];

    // (total / data_size) is equal to mean and it will return to main function.
    return ( total / data_size );
}    //end CalculateMean ( )

//PrintMean just prints the variable called mean that is sent from main function.
void PrintMean(double mean){

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
//  There is 2 "|" and "MEAN" 4.        4+2=6;
//  26-6=20;   20/2=10;     10 + 4(MEAN) = 14;   To center "MEAN", setw(14)
    cout<<"|"<<setw(14)<<"MEAN"<<setw(12)<<"|\n";
    //  |          MEAN          |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

    cout<<"|"<<setw(24)<<mean<<"|\n";
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
}     //PrintMean ( )


//CalculateVariance calculates the variance of the array and returns it to main function.
//It takes one pointer that holds the first element's address of the array, one integer value called data_size and one double called mean.
double CalculateVariance(int* data_array, int data_size, double mean){

    double variance=0;

    for(int i=0; i<data_size; i++)
        variance+=pow( (data_array[i] - mean) , 2);
    //  The variable "variance" is now the total of the formula. (Variance / data_size) gives us the variance.

    //  It is done that to get rid of more variable (e.x. total).
    return ( variance / data_size );
}    //end CalculateVariance ( )

//PrintVariance just prints the variable called variance that is sent from main function.
void PrintVariance(double variance){

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

//  There is 2 "|" and "VARIANCE" 8.                     8+2=10;
//  26-10=16;   16/2=8;     8 + 8(VARIANCE) = 16;    To center "VARIANCE", setw(16)
    cout<<"|"<<setw(16)<<"VARIANCE"<<setw(10)<<"|\n";
    //  |        VARIANCE        |
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

    cout<<"|"<<setw(24)<<variance<<"|\n";

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
}    //end PrintVariance ( )

//CalculateStandartDeviation calculates the standart deviation of the array and returns it to main function.
//It takes one pointer that holds the first element's address of the array, one integer variable called data_size and one double called mean.
double CalculateStandartDeviation(int* data_array, int data_size, double mean){

    double variance=0;

    for(int i=0; i<data_size; i++)
        variance+=pow( (data_array[i] - mean) , 2);

    //  The variable "variance" is now the total of the formula. (Variance / data_size) gives us the variance.
    //  It is done that to get rid of more variable (e.x. total).

    //  Square root of the variance gives us the standart deviation of the array.
    return sqrt(variance / data_size);
}    //end CalculateStandartDeviation ( )

//PrintStandartDeviation just prints the variable called st_deviation that is sent from main function.
void PrintStandartDeviation(double std_deviation){
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

//  There is 2 "|" and "STANDART DEVIATION" 18.                    18+2=20;
//  26-20=6;   6/2=3;     3 + 18(STANDART DEVIATION) = 21;    To center "STANDART DEVIATION", setw(21)
    cout<<"|"<<setw(21)<<"STANDART DEVIATION"<<setw(5)<<"|\n";
    //  |   STANDART DEVIATION   |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
    cout<<"|"<<setw(24)<<std_deviation<<"|"<<endl;

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
}    //end PrintStandartDeviation ( )

//CalculateZScore calculates the Z score of each elements of the array and returns this zScoreArray to main function.
//         It takes one pointer that holds the first element's address of the array,
//         one integer variable called data_size, and two double variables called mean and std_deviation.
double* CalculateZScore(int* data_array, int data_size, double mean, double std_deviation){

//Function returns the first element's address of the zScoreArray, to do that it should be declared as dynamic array.
    double* zScoreArray = new double [data_size];

    for(int i=0; i<data_size; i++)
        zScoreArray[i] = (data_array[i]-mean) / std_deviation;

    //The first element's address of the zScoreArray is returned to the main function.
    return zScoreArray;
}    //end CalculateZScore ( )

//PrintZScoreArray prints the whole elements of the array.
//It takes one pointer that holds the first element's address of the zScoreArray and one integer value called data_size.
void PrintZScoreArray(double* zscore, int data_size){

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

//  There is 2 "|" and the size of the "Z__SCORE" 8.                        8+2=10;
//  26-10=16;   16/2=8;     8 + 8(Z__SCORE) = 16;    To center "Z__SCORE", setw(16)
    cout<<"|"<<setw(16)<<"Z__SCORE"<<setw(10)<<"|\n";
    //  |        Z__SCORE        |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

    for(int i=0; i<data_size; i++)
        cout<<"|"<<setw(24)<<zscore[i]<<"|"<<endl;
// | i|
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
}    //end PrintZScoreArray ( )

//CalculateTScore calculates the T score of each elements of the array and returns this tScoreArray to main function.
//         It takes one pointer that holds the first element's address of the array,
//         one integer variable called data_size, and three double variables called data_size, mean and std_deviation.
double* CalculateTScore(int* data_array, int data_size, double mean, double std_deviation){
//Function returns the first element's address of the tScoreArray, to do that tScoreArray should be declared as dynamic array.
    double* tScoreArray = new double [data_size];

    for(int i=0; i<data_size; i++)
        tScoreArray[i] = 10 * ( (data_array[i] - mean) / std_deviation ) + 50;

    //The first element's address of the tScoreArray is returned to the main function.
    return tScoreArray;
}    //end CalculateTScore ( )

//PrintTScoreArray prints the whole elements of the array.
//It takes one pointer that holds the first element's address of the tScoreArray and one integer value called data_size.
void PrintTScoreArray(double* tscore, int data_size){
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

//  There is 2 "|" and the size of the "Z__SCORE" 8.                        8+2=10;
//  26-10=16;   16/2=8;     8 + 8(T__SCORE) = 16;    To center "T__SCORE", setw(16)
    cout<<"|"<<setw(16)<<"T__SCORE"<<setw(10)<<"|\n";
    //  |        T__SCORE        |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
    for(int i=0; i<data_size; i++)
        cout<<"|"<<setw(24)<<tscore[i]<<"|"<<endl;
    //  |                       i|
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
}    //end PrintTScore ( )
```

Quiz

```
double* CalculateDecimalScaledData (int* data_array, int data_size);
int FindMinJValue (int* data_array, int data_size);
double FindAbsoluteMax (int* data_array, int data_size);
void PrintDecimalScaledData (double* decimal_scaled_data, int data_size);

//      CalculatedDecimalScaledData will divide each element of the array to the minimimJ'th power of ten, and return the whole array to main function.
double* CalculateDecimalScaledData (int* data_array, int data_size){

//      To get minimumJ FindMinJValue is called.
    int minimumJ = FindMinJValue(data_array, data_size);

//      Declared dynamic array called minJArray to return its address to main function.
    double* minJArray = new double [data_size];

//      Calculation of the minimum J.
    for(int i=0; i<data_size; i++)
        minJArray[i] = data_array[i] / pow(10,minimumJ);

//      minJArray’s address is returned to main function.
    return minJArray;
}//end  CalculatedDecimalScaledData ( )

//         FindMinJValue will divide each element of the array to the nth power of ten, and return n as MinJValue.
int FindMinJValue (int* data_array, int data_size){

    double absoluteMaximum = FindAbsoluteMax(data_array, data_size);

    //If absoluteMaximum is lower than one, function will return the power of ten.
    int power=0;
    for( ; !(absoluteMaximum<1); power++)
        absoluteMaximum /= pow(10,power);

//         power is increased by the for loop, when the loop is terminated power once more increased.
//         That’s why FindMinJValue ( ) returned “ power – 1 “
    return power-1;
} //end FindMinJValue
```

```
//         FindAbsoluteMax will find the absolute maximum value of the array and return it.
double FindAbsoluteMax (int* data_array, int data_size){
//         "temp" is declared to make any number of data_array positive. "maximumNumber" is declared to find which one is bigger.
    int temp, maximumNumber = data_array [0];

            for(int i=0; i<data_size; i++){
                        temp = data_array[i];                              //          data_array [ i ] is assigned to temp to don't change the real value of data_array.

                        if(temp<0)                                         //         If the number is negative
                                    temp = temp*-1;                        //         it will be positive.

                        if(temp>maximumNumber)                             //         Comparing which one is bigger.
                                     maximumNumber = temp;                 //         The biggest one is assigned to variable called maximumNumber
            }//end for


            //          maximumNumber will be returned.
    return maximumNumber;
}//end FindAbsoluteMax ( )

//PrintDecimalScaledData will print the whole elements of the array.
void PrintDecimalScaledData (double* DecimalScaledData, int data_size){

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

    //  There is 2 "|" and the size of the "DecimalScaltedData" 18.
    //  26-20=6;   6/2=3;     3 + 18(DecimalScaledData) = 21;    To center "DecimalScaledData", setw(21)           18+2=20;
cout<<"|"<<setw(21)<<"DecimalScaledData"<<setw(5)<<"|\n";
    // |    DecimalScaledData   |

    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+

    for(int i=0; i<data_size; i++)
        cout<<"|"<<setw(24)<<DecimalScaledData[i]<<"|"<<endl;
// | i|
    for(int i=0; i<=25; i++){
        if(i==0)    cout<<"+";
        else if(i==25)  cout<<"+\n";
        else    cout<<"-";
    }// +------------------------+
}//end  PrintDecimalScaledData ( )
```

Conclusion

*	I’ve learned the steps to take when using **setw** function which is defined in **iomanip** header.

*	Pointers and arrays are created in STACK MEMORY so, we will lose their address and values when a function terminated.

*	To reach a pointer or an array in every function and don’t lose its values, and if we want to return its address to another function, we have to declare them as DYNAMIC.

*	To declare something as dynamic, we use **new** operator in C++.

Note: The report prepared by Şafak Akıncı.