# School of Computer Science and Engineering - B.Tech (H)

**Course Name:** Fundamentals of Programming with C
**Course Code:** CS1000

Author: Abhijit Das, Dept. of SOCSE

| | |
|---|---|
| Author(s) | Abhijit Das |
| Authorized by | |

# Document Revision History

| Version | Date | Author(s) | Reviewer(s) | Comments |
|---|---|---|---|---|
| 1.0 | 17/09/2023 | Abhijit Das | | Draft Version |

# Contents

## Context

This document contains assignments to be completed as part of the hands on for the subject Fundamentals of Programming with C (**Course Code:** CS1000)

## Guidelines

- The lab guide has been designed to give a hands on experience to map the concepts learnt in the theory session with practical assignments
- The assignments have been categorized into solved assignments to hand hold a beginner, partially solved assignments to begin trying out on their own and unsolved assignments to let learners write code completely on their own
- These assignments contain coding exercises, debugging exercises, coding standards exercises and self review assignments
- There is a threaded assignment which can be built incrementally every day. This will help understanding the concepts and building a complete application.
- Solving these exercises methodically would provide confidence to the learner to attempt the module and comprehensive exams
- The estimated time would help a learner to solve problems given a deadline
- The assignments need to be completed on day basis as instructed by the facilitators and submitted appropriately
- In order to complete the course, assignments in this document must be completed in the sequence mentioned
- A practice guide is also a part of the Lab assignments containing many code snippets for practice and logic building exercises.

## ACTIVITY/ASSIGNMENT-1 Assignments

Note: The pseudo code can be written in notepad and saved as .txt files and submitted. For the trace tables, excel sheets or tables embedded in word documents can be submitted.

### Assignment 1: Pseudo code- Sequential pattern

Objective: Writing pseudo code for sequence pattern problems

Background: Pseudo code addresses Sequential, Selectional or Iterational patterns of writing code

Problem Description: The finance department of a company wants to calculate the gross pay of an employee in the company. The number of hours worked by the employee and the pay rate should be accepted from the user and the gross pay should be calculated as the below formula.

*Gross Pay = Number of hours worked * Pay rate*

Estimated time: 10 minutes

Step 1: Analyze the problem

Step 2: Identify the data to be accepted from the user

Step 3: Identify calculation/computation statements

Step 4: Identify the outputs

Step 5: Write the following Pseudo code using NotePad

```
CALCULATE-GROSS-PAY
     1. input No_of_Hours, Pay_Rate
     2. Gross_Pay = No_of_Hours * Pay_Rate
     3. display "Gross Pay is: ", Gross_Pay
```

Step 6: Identify the name of the procedure, variables used and the keywords in the above pseudo code
Step 7: Save the file as CalculateGrossPay.txt

Summary of this assignment:
In this assignment, you have learnt:

- To analyze a problem and understand for the logic by writing pseudo code

## Assignment 2: Test a Sequential pattern Pseudo code

Objective: Test pseudo code using dry run technique and trace table

Background: Pseudo code can be tested without computers by dry running

Problem Description: Dry Run the pseudo code written for Assignment 1 using Trace Table

Estimated time: 10 minutes

Step 1: Identify the missing data in the Trace table and fill the Trace table

```
Trace Table when inputs are as given below:
No_of_Hours = 10, Pay_Rate = 40 Rs
```

| Line Number | No_of_Hours | Pay_Rate | Gross_Pay | Output |
|---|---|---|---|---|
| 1 | 10 | ? | | |
| 2 | | | ? | |
| 3 | | | | ? |

Summary of this assignment:
In this assignment, you have learnt:
- To Dry run a pseudo code using trace table

## Assignment 3: Pseudo code - Selectional pattern(1)

Objective: Write pseudo code for selectional pattern problems

Background: Selectional patterns are used for making decisions. They are useful while validating data. Validation is needed to check the correctness of the data being entered which if not done, leads to incorrect results

Problem Description: The problem given in Assignment 1 should be extended to check whether the Number of Hours accepted from the user is less than or equal to 0. If so an appropriate error message should be displayed.

Estimated time: 10 minutes

Step 1: Identify the validations to be done

Step 2: Write the Pseudo code

Summary of this assignment:
In this assignment, you have learnt:
- To analyze a selectional problem and write the pseudo code using selectional statements

## Assignment 4: Test a Selectional pattern Pseudo code(1)

Objective: Test pseudo code using dry run technique and trace table

Background: Pseudo code can be tested without computers by dry running

Problem Description: Dry Run the pseudo code written for Assignment 3 using Trace Table

Estimated time: 20 minutes

Case 1: Dry run the pseudo code using Trace table when the inputs are as given below:
No_of_Hours = 40, Pay_Rate = 60 Rs

Case 2: Dry run the pseudo code using Trace table when the inputs are as given below:
No_of_Hours = 0, Pay_Rate = 60 Rs

Case 3: Dry run the pseudo code using Trace table when the inputs are as given below:
No_of_Hours = -20, Pay_Rate = 60 Rs

Summary of this assignment:
In this assignment, you have learnt:

- To Dry run a selectional pattern pseudo code in such a way that all the logical paths are covered

## Assignment 5: Pseudo code - Iterational pattern(1)

Objective: Write pseudo code for iterational pattern problems

Background: Iterational pattern is used to repeat a certain activity for a certain number of times. An iteration has a counter, a stop condition for the iteration and an activity to be done

Problem Description: The problem given in Assignment 1 is extended to check whether the Number of Hours accepted from the user is less than or equal to 0. If so display an error message. Reenter the data the Number of Hours until a valid data is entered.

Estimated time: 15 minutes

Step 1: Write the following Pseudo code

```
CALCULATE-GROSS-PAY
      1. input No_of_Hours, Pay_Rate
      2. while (--------) do
      3.    display " -------------------- "
      4.    ------------------------------------------------
      5. end-while
      6. Gross_Pay = No_of_Hours * Pay_Rate
      7. display "Gross Pay is: ", Gross_Pay
```

Step 2: Fill in the missing (------ ) parts in the Pseudo code

Summary of this assignment:
In this assignment, you have learnt:

- To analyze an iterational problem and write the pseudo code using iterational statement (while..do)

## Assignment 6: Test an Iterational pattern Pseudo code (1)

Objective: To understand how to test a pseudo code using dry run technique and trace table

Background: Pseudo code can be tested without computers by dry running

Problem Description: Dry Run the pseudo code written for Assignment 5 using Trace Table

Estimated time: 20 minutes

Case 1: Dry run the pseudo code using Trace table when the inputs are as given below:
No_of_Hours = 10, Pay_Rate = 30 Rs

Case 2: Dry run the pseudo code using Trace table when the inputs are as given below:
1st iteration: No_of_Hours = 0, Pay_Rate = 30 Rs
2nd iteration: No_of_Hours = 20

Fill in the condition and the missing parts (?) in the trace table

```
Trace Table when inputs are as given below:
 No_of_Hours = 0, 20, Pay_Rate = 30 Rs
```

| Line Number | No_of_Hours | Pay_Rate | Condition | Gross_Pay | Output |
|---|---|---|---|---|---|
| 1 | 0 | 30 | | | |
| 2 | | | ? | | |
| 3 | | | | | ? |
| 4 | 20 | | | | |
| 2 | | | ? | | |
| ? | | | | | |
| ? | | | | | |

Case 3: Dry run the pseudo code using Trace table when the inputs are as given below:
1st iteration: No_of_Hours = 0, Pay_Rate = 30 Rs
2nd iteration: No_of_Hours = -1
3rd iteration: No_of_Hours = 1

Summary of this assignment:
In this assignment, you have learnt:

- To dry run an iterational pattern pseudo code to test all the logical paths being covered

## Assignment 7: Exercises for Self Review:

1. Write the Pseudo code for the following problems and dry run it using Trace table

    a. Accept a number and check whether a number is Positive or Negative. If the number is POSITIVE display the message “POSITIVE NUMBER” otherwise display “NEGATIVE NUMBER”.

    b. Accept an year and check whether the year is Leap year or not. The year accepted can lie between 1600 and 2010 only.
    Rules for determining leap year are given below:
        1. Most years that can be divided evenly by 4 are leap years. (For example, 2012 divided by 4 = 503: Leap year!)
        2. Exception: Century years are NOT leap years UNLESS they can be evenly divided by 400. (For example, 1700, 1800, and 1900 were not leap years, but 1600 and 2000, which are divisible by 400, were leap years.)

## Activity / Assignments-2

## Assignment 8: Using Visual Studio IDE to create the first C program

Objective: To learn how to write a C program, to compile and execute it.

**Note:** Demonstration will be shown in the classroom.

Summary of this assignment:

In this assignment, you have learnt:

- How to create a project workspace in Visual Studio
- How to type code in the Visual C++ IDE and to save it as part of a workspace/project.
- How to compile and link a C program
- How to execute your program from the Visual Studio IDE
- How to close the workspace

## Assignment 3: Understanding the variables and conversion specifiers- Debugging Assignment

Objective: To define variables and to print the numbers in decimal, hex decimal and octal number systems

Background: C supports different basic data types (`int`, `float`, `double`, `char` etc).

Note: A text file VariablesDemo.c is provided to you in Supplied Source Code folder within the Lab Guide folder.

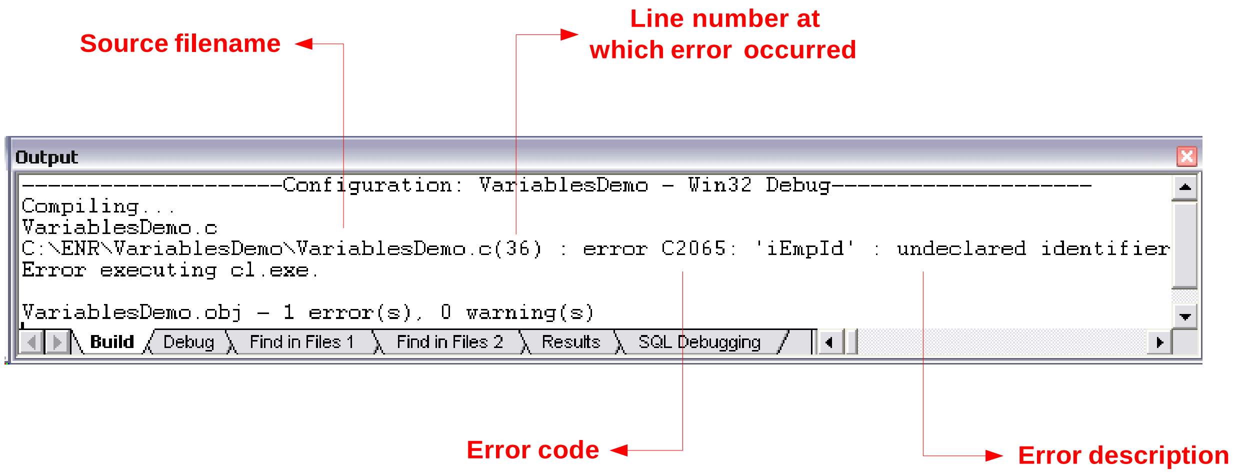
Problem Description: Declare variables to store employee id, basic salary and allowances. Display all the details. Also display employee id in hexadecimal format.

Estimated time: 20 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'VariablesDemo'

Step 2: Create a Source file in the 'VaribalesDemo' project with the name VariablesDemo.c. Now copy the source code, given in VaribalesDemo.c to the one created in the project

Step 3: Compile the program and the compilation errors and warnings will be displayed as given below:



```
--------------------Configuration: VariablesDemo - Win32 Debug--------------------
Compiling...
VariablesDemo.c
C:\ENR\VariablesDemo\VariablesDemo.c(36) : error C2065: 'iEmpId' : undeclared identifier
Error executing cl.exe.

VariablesDemo.obj - 1 error(s), 0 warning(s)
```

Step4: Error message says, C:\...\VariablesDemo.c (36) iEmpId : undeclared identifier. The number 36 in brackets indicate the line number in which the error occurred. The error description says that 'iEmpId' is an undeclared identifier. It appears that the variable 'iEmpId' has not been declared earlier.

Note 1: To go to the line in which error occurred, double click on that particular error message in the Output Window.

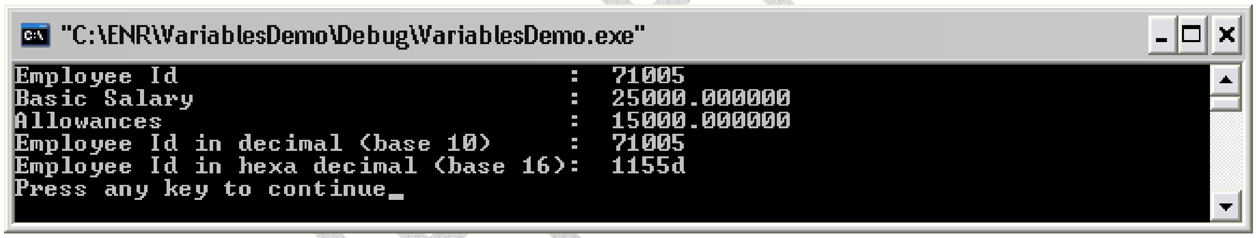
Note 2: Alternately you can use Ctrl + G in visual studio and type in the line number yourself.

Go to the line indicated by compiler and inspect the error.

Step 5: Include the statement `int iEmpId` under variable declarations and compile again.

Step 6: If there are no typographical errors, program should compile successfully.

Step 7: Execute the program and the output screen should be as follows:


"C:\ENR\VariablesDemo\Debug\VariablesDemo.exe"

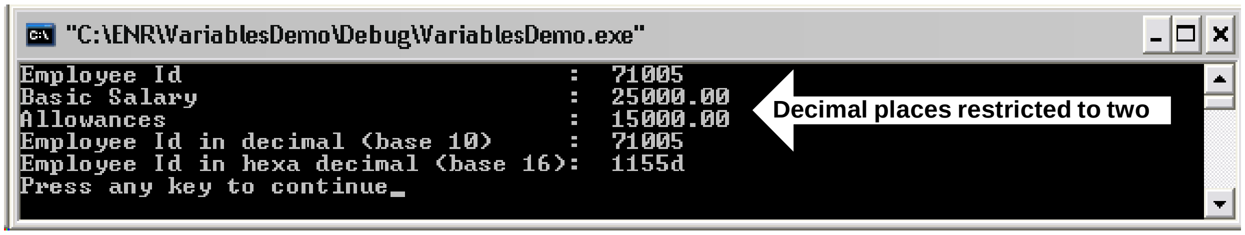
```
Employee Id                       :  71005
Basic Salary                      :  25000.000000
Allowances                        :  15000.000000
Employee Id in decimal (base 10)  :  71005
Employee Id in hexa decimal (base 16):  1155d
Press any key to continue_
```

The floating point values are displayed as "25000.000000" and "15000.000000". This can be restricted to two decimal places (precision) by modifying the printf() statements as given below:

```
printf("Basic Salary                         :  %.2lf\n",
                                                       dBasicSalary);
printf("Allowances                           : %.2f\n",
                                                    fAllowances);
```

Here .2f or .2lf indicates that only two decimal places are allowed after the decimal point.

Step 8: Modify the printf() statements, compile again and execute. The output should be as given below:



```
Employee Id                      :   71005
Basic Salary                     :   25000.00
Allowances                       :   15000.00
Employee Id in decimal (base 10)     :   71005
Employee Id in hexa decimal (base 16):   1155d
Press any key to continue_
```

"C:\ENR\VariablesDemo\Debug\VariablesDemo.exe"
Decimal places restricted to two

Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Summary of this assignment:
In this assignment, you have learnt:
- Variable declarations
- Conversion specifiers

## Assignment 4: Understanding the range of data types

Objective: To define variables and to print the values based on the range permitted by the datatype and observe the results

Background: C supports different basic data types (`int`, `float`, `double`, `char` etc) with different ranges .

Note: A text file Range.c is provided to you in Supplied Source Code folder within the Lab Guide folder.

Problem Description: Declare variables to store different values and observe the results if the values are within and out of range in case of signed and unsigned integers

Estimated time: 10 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'Range'

Step 2: Add the source code, Range.c provided into the project

Step 3: Compile the program

Step 4: Execute the program and observe the output.

Summary of this assignment:
In this assignment, you have learnt:
- Integer datatype and its range
- If the number is within the range of the datatype , then same number will be assigned

- In case of signed integer, if the number is outside the range, then –ve values will be assigned in a circular sequence
  For example, if iNumber = 32768, then - 32768 will be assigned
  if iNumber = 32769, then -32767 will be assigned and so on
- In case of unsigned integer, if the number is outside the range, then starting from 0, positive numbers will be assigned in a circular sequence
  For example, if iValue = 65536, then 0 will be assigned
  if iValue = 65537, then 1 will be assigned and so on

## Assignment 5: Using Character Variables-Debugging Assignment

Objective: To understand the use of character variables.

Background: C supports char data type for character variables.

Note: A text file JobBand.c is provided to you in Supplied Source Code folder within the Lab Guide folder.

Problem Description:
Extending the code in Assignment No. 9, include a job band for the employee. Job band can be ‘A’ or ‘B’ or ‘C’ or ‘D’ or ‘E’.

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name ‘JobBand’

Step 2: Add the source code, JobBand.c provided into the project

Step 3: Compile the program. There will be compilation errors. Debug the errors.

Step 4: Recompile the program and execute the program.

Step 5: Include a printf() statement to print the ASCII value of job band as given below:

```
printf("ASCII value of %c Job band                : %d\n",
                                                  cJobBand,cJobBand);
```

Step 6: Recompile the program and execute the program and observe the output containing the ASCII value of ‘B’.

Summary of this assignment:
In this assignment, you have learnt:
- Usage of character variables
- Initializing char variables
- Printing characters using %c conversion specifier

## Assignment 6: ASCII usage

Objective: To learn ASCII code usage and the conversion between the character and integer datatype.

Problem Description: Declare a character variable and print the ASCII value of the corresponding character

Estimated time: 20 minutes

Step 1: Consider the program written in the previous assignment (the corrected JobBand.c) related to character variables. Declare a char variable to store the confirmation status of the employee. The confirmation status can store either ‘Y’ or ‘N’.

Step 2: Assign the ASCII value of ‘Y’ to confirmation status variable.

Step 3: Print the character ‘Y’ from the ASCII value stored

Note: ASCII table gives the mapping of the character variable to a number called the ASCII value. For example, if a character variable is containing a character ‘A’, the ASCII value of ‘A’ is 65.

Summary of this assignment:
In this assignment, you have learnt:

- Usage of char variables and ASCII values

## Assignment 7: Coding standards

Objective: To understand the importance of coding standards and use them

Background: Every organization follows coding standards for the programs written in order to improvethe maintainability aspect.

Note: A text file CodingStandards.c is provided to you in Supplied Source Code folder within the Lab Guide folder.

Problem Description: Identify the missing coding standards in the given source code, correct them and execute the code. (The coding standards include documentation, indentation, variable naming standards, proper comments, file header, footer etc)

Estimated time: 10 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name ‘CodingStandards’

Step 2: Add the source code, CodingStandards.c provided into the project

Step 3: Identify the missing coding standards and include them in the given source code

Step 4: Compile the program

Step 5: Execute the program and observe the output

Summary of this assignment:

In this assignment, you have learnt:
- To use the various coding standards to write well documented code

## Assignment 8: Using preprocessor directives and arithmetic operators

Objective: To learn the usage of preprocessor directives and arithmetic operators

Note: A text file SalaryComputation.c provided to you in Supplied Source Code folder within the Lab Guide folder

Problem Description: Write a program to define a constant named "INCOMETAXPERCENT" and display the gross salary, income tax percentage and net salary

Note: The gross salary is the sum of basic salary and the allowances .The income tax is calculated as 20% of the gross salary. The net salary is the difference of the gross salary and the income tax.

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'SalaryComputation'

Step 2: Add the source code, SalaryComputation.c provided into the project

Step 3: Write the missing code in the program. The missing code is given as comments.

Step 4: Compile the program

Step 5: Execute the program and observe the output

Summary of this assignment:
In this assignment, you have learnt:
- Declaration of constants using #define
- Arithmetic operations

## Assignment 9: Understanding Typecasting-Debugging Assignment

Objective: To understand typecasting

Note: A text file AverageCustomerFeedback.c provided to you in Supplied Source Code folder in the Lab Guide folder

Problem Description: Write a program to find the average customer feedback (In a scale of 10) for the employee. The customer received feedbacks from three customers.
The program is given to you.

Estimated time: 15 minutes

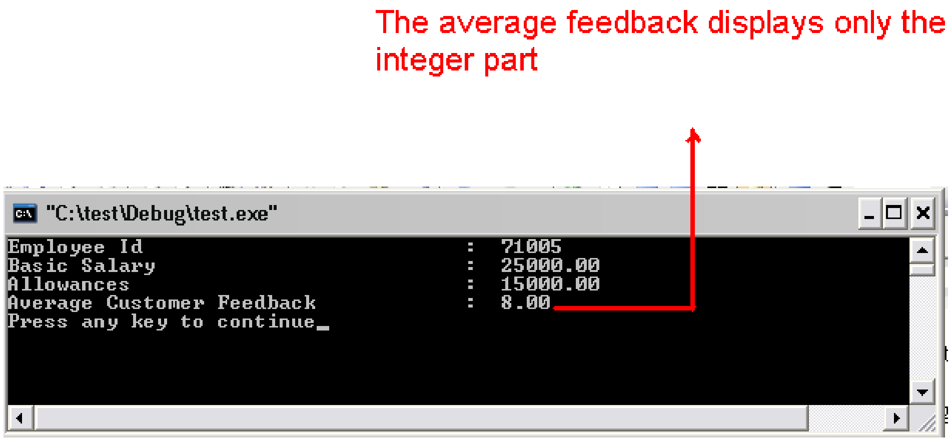
Step 1: Create an empty project in Visual C++ (Console Application) with name 'AverageCustomerFeedback'

Step 2: Add the source code, AverageCustomerFeedback.c provided into the project

Step 3: Compile the program and fix the compilation errors.

Step 4: Ignore the warnings and execute the program.

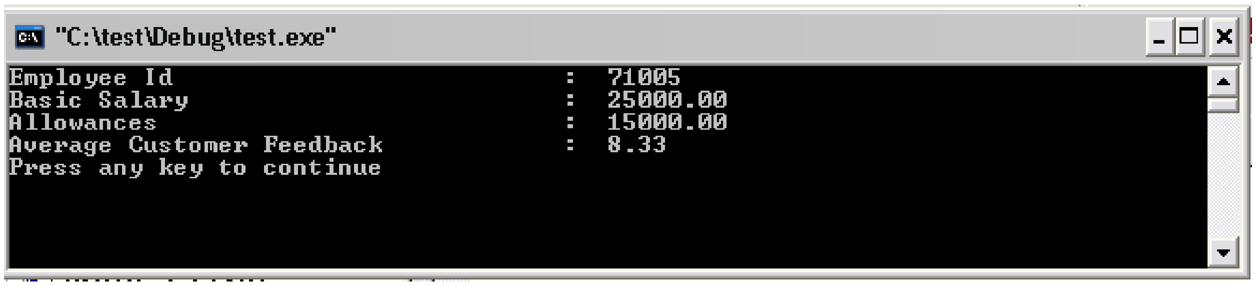
Step 5: The output will be as given below:



```
Employee Id                      :  71005
Basic Salary                     :  25000.00
Allowances                       :  15000.00
Average Customer Feedback        :  8.00
Press any key to continue_
```

Step 6: Identify and correct the error related integer division and type casting.

Step 7: Compile the program. There should not be any warnings. Execute the program and the output should be as given below:



```
Employee Id                          :  71005
Basic Salary                         :  25000.00
Allowances                           :  15000.00
Average Customer Feedback            :  8.33
Press any key to continue
```

Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Summary of this assignment:
In this assignment, you have learnt:

- Type casting

## Assignment 10: Exercises for Self Review

2. Debug the following code snippet?

```
int main(int argc,char **argv) {
        int iEmpCode = 1001;
        printf("\n-------------------------------------------------------------- \n");
        printf("              Employee Information                   \n");
        printf("-------------------------------------------------------------- \n\n");
        float fSalary;
        printf("Enter the salary of an employee\n");
        scanf("%f", &fSalary);
        printf("Enter the employee department code\n");
        scanf("%d", &iDeptCode);
        printf("Employee code = %d\n",iEmpCode);
        printf("Salary = %f\n",fSalary);
        printf("Department code = %d\n",iDeptCode);
        return 0;
 }
```

3. The following source code does not implement the coding standards. Identify them and implement the required coding standards.

```
/******************************************************************************
* Filename    : CodingStandards2.c
* Author      : RV University, SOCSE
* Date        : 17-09-2023
* Description : Helps to understand the coding standards for a program
******************************************************************************/

/* Include files */
#include<stdio.h>

/***********************************************************
```

```
* Description: main()
* Input Parameters:
*    argc – Command line arguments count
*    argv - Command line arguments
* Returns : 0 on successful execution to the operating system
*          and 1 if is unsuccessful.
************************************************************/
void main(int argc,char** argv){

    /*declaring and initializing variables */
    int iNumber =10,iNumber_2;
    double fValue;

    printf("Enter the values\n");
    scanf("%d %lf", &iNumber_2,&fValue); /* reading from the user */

    /*displaying values */
    printf("%d %d %0.2lf",iNumber,iNumber_2,fValue);

}
/******************************************************************************
*End of CodingStandards2.c
******************************************************************************/
```

4. What is the output of the following code snippet?

```
#include<stdio.h>
#define NUMBER 10
int  main(int argc,char **argv) {
        int iValue = 20,iResult;
        iResult = iValue + NUMBER ;
        printf("Sum = %d", iResult);
        return 0;
 }
```

5. Debug the following code snippet:

```
#include<stdio.h>
int main (int argc, char** argv) {
    int iEmpId;
    printf("Enter the employee id ");
    scanf("%d",iEmpId);
    printf("\nEmployee Id  : %d\n",iEmpId);
    return 0;
}
```

6. Find the output of the following program:

```
int main(int argc, char **argv) {
```

```
                int iValue1=10, iValue2=3;
                float iResult;
                iResult = iValue1 / iValue2;
                printf("%f",iResult);
                return 0;
}
```

7. Find the output of the following program:

```
int main(int argc, char **argv) {
                double iValue=100.5;
                printf("%d",sizeof(iValue));
                return 0;
}
```

8. Debug the following program and fix the error:

```
/*****************************************************************************
* Filename    : Operators.c
* Author : RV University, SOCSE
* Date   : 17-09-2023
* Description : To understand the use modulus operator
*****************************************************************************/
/* Include files */
#include<stdio.h>
#define NUMBER 2
/*****************************************************************************
* Function              : main()
* Description  : main fuction to understand the use modulus operator
* Input Parameters:
*        int argc - Number of command line arguments
*   char **argv  The command line arguments passed
* Returns: 0 on success to the operating system
*****************************************************************************/
void main(int argc,char **argv) {

         /*Variable declaration*/
         float fValue = 5.2,iResult;

         /* Calculating remainder */
         iResult = fValue % NUMBER ;

          /* Displaying the Result */
         printf("Result = %d", iResult);
 }
/*****************************************************************************


* End of Operators.c
 *****************************************************************************/
```

## Activity/Assignments-3

### Assignment 11: Using 'if', 'if-else ' and 'else-if' statement

Objective: To understand usage of 'if', 'if-else' and 'else-if' statement.

Problem Description:

a) Write a program to accept employee id, basic salary and allowances from the user. The program should find the income tax to be paid (in Indian Rupees) and the net salary after the income tax deduction. An employee should pay income tax if his monthly gross salary is more than Rs. 10,000 (Indian Rupees) and the percentage of income tax is 20%. Display the employee id, basic salary, allowances, gross pay, income tax and net pay.

   Estimated time: 15 minutes

b) Extend the source code of Previous Assignment No. 11 a) by including the following: An employee should pay 10% income tax if his monthly gross salary is less than 10,000 rupees (Indian Rupees). Display the employee id, basic salary, allowances, gross pay, income tax and net pay.

   Estimated time: 10 minutes

c) Extend the Assignment No. 11 b) to find the income tax to be paid (In Indian Rupees) and the total salary after the income tax deduction as per the details given in below table.

   An employee should pay income tax as per the following:

| Gross Salary (In Indian Rupees) | Income Tax percentage |
|---|---|
| Below 5,000 | Nil |
| 5,001 to 10,000 | 10 % |
| 10,001 to 20,000 | 20% |
| More than 20,000 | 30% |

   Display the employee id, basic salary, allowances, gross pay, income tax and net pay.

Estimated time: 15 minutes

Note: The modified SalaryComputation.c used for one of the previous assignments can be reused in this assignment

Summary of this assignment:
In this assignment, you have learnt:

- if construct
- if-else construct
- else-if construct

## Assignment 12: Handling Compiler and Linker Errors-Debugging Assignment

Objective: To understand how to handle compiler and linker errors and writing error handling code.

Background: This assignment helps you to understand how to handle compiler and linker errors.

Note: A text file compileerrors.c provided to you in Supplied Source Code folder within the Lab Guide folder.

Problem Description:

Estimated time: 20 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'compileerrors'

Step 2: Add the source code, compileerrors.c provided into the project.

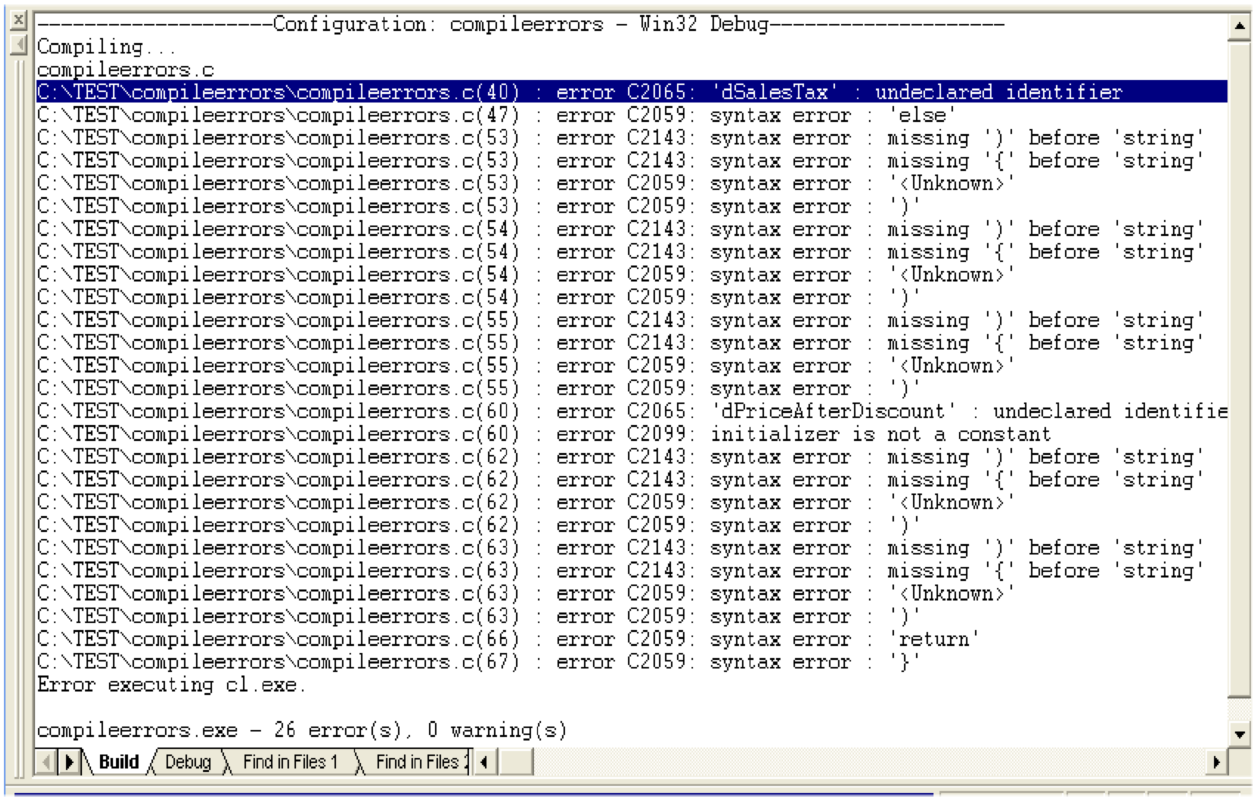
Step 3: Compile the program.

Step 4: The program will have many errors and warnings. (26 errors and 0 warnings at least). Do not get overwhelmed by the number of errors.

When compiling, as a thumb rule, always start by tackling the first error the compiler shows and move next.

Note: The compiler goes in a sequential order from the top of the file to the bottom of the file. (i.e. The compiler starts compiling from line 1, 2, 3… till the end of file).

In most cases where there are so many compiler errors, the errors occurring later may be related to one of the earlier errors. Therefore, as a thumb Thumb rule, always start by handling the first error the compiler shows.



```
--------------------Configuration: compileerrors - Win32 Debug--------------------
Compiling...
compileerrors.c
C:\TEST\compileerrors\compileerrors.c(40) : error C2065: 'dSalesTax' : undeclared identifier
C:\TEST\compileerrors\compileerrors.c(47) : error C2059: syntax error : 'else'
C:\TEST\compileerrors\compileerrors.c(53) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(53) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(53) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(53) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(54) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(54) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(54) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(54) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(55) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(55) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(55) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(55) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(60) : error C2065: 'dPriceAfterDiscount' : undeclared identifie
C:\TEST\compileerrors\compileerrors.c(60) : error C2099: initializer is not a constant
C:\TEST\compileerrors\compileerrors.c(62) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(62) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(62) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(62) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(63) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(63) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(63) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(63) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(66) : error C2059: syntax error : 'return'
C:\TEST\compileerrors\compileerrors.c(67) : error C2059: syntax error : '}'
Error executing cl.exe.

compileerrors.exe - 26 error(s), 0 warning(s)
```

Step 5: The first error here, is in Line 40 and it says ‘dSalesTax’ undeclared identifier. Go to Line 40 by either double clicking on that error message or by using Ctrl + G option.

Inspect the code in line 40.

```
scanf ("%lf", &dSalesTax);
```

A variable dSalesTax is being used. Look in the code for declaration of this variable and see if there are any typos there.

Step 6: When you look in the code, in the declaration section, it appears that the variable dSalesTax is not declared.

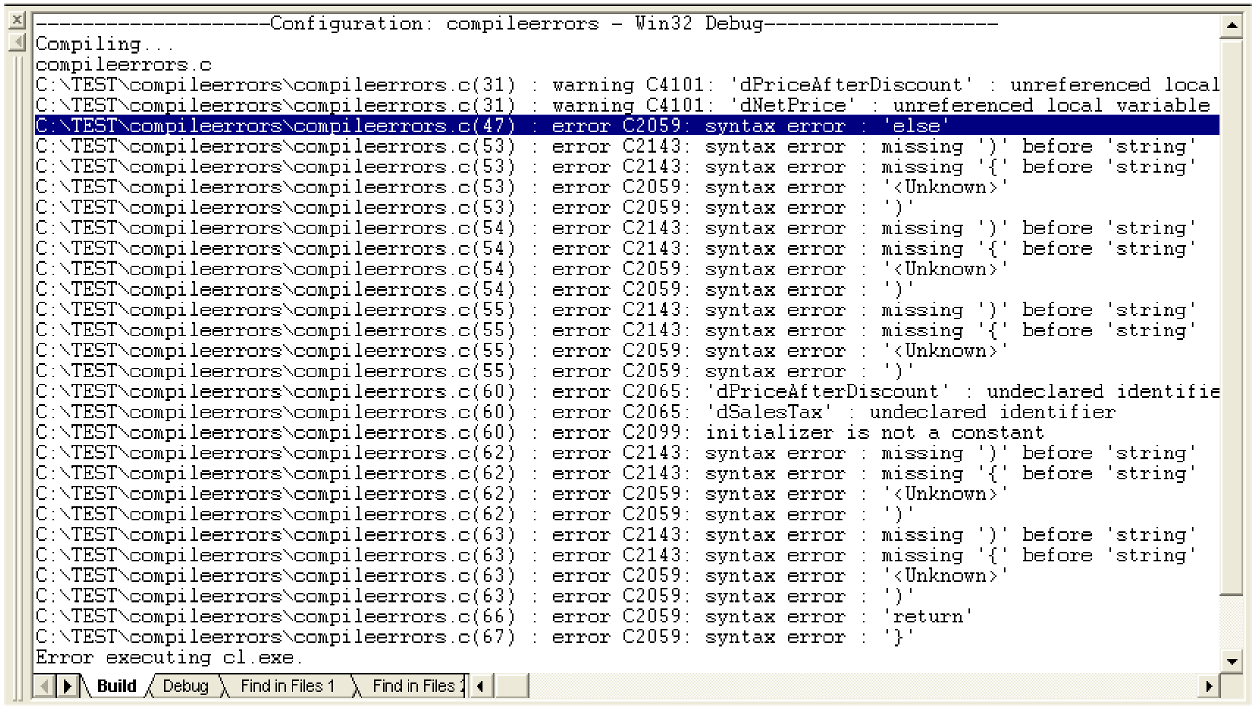
```
/* declaration of variables */
double dPrice, dDiscount;
double dPriceAfterDiscount, dNetPrice;
```

Step 7: Declare the variable dSalesTax and then recompile.

```
double dPrice, dDiscount, dSalesTax;
```

Step 8: Now you will notice that the previous error message will not appear anymore because it has been fixed.

You will see a few warnings. For now, until all compiler errors are resolved, ignore the warnings. Handle the first error you see now.



```
--------------------Configuration: compileerrors - Win32 Debug--------------------
Compiling...
compileerrors.c
C:\TEST\compileerrors\compileerrors.c(31) : warning C4101: 'dPriceAfterDiscount' : unreferenced local
C:\TEST\compileerrors\compileerrors.c(31) : warning C4101: 'dNetPrice' : unreferenced local variable
C:\TEST\compileerrors\compileerrors.c(47) : error C2059: syntax error : 'else'
C:\TEST\compileerrors\compileerrors.c(53) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(53) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(53) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(53) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(54) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(54) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(54) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(54) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(55) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(55) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(55) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(55) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(60) : error C2065: 'dPriceAfterDiscount' : undeclared identifie
C:\TEST\compileerrors\compileerrors.c(60) : error C2065: 'dSalesTax' : undeclared identifier
C:\TEST\compileerrors\compileerrors.c(60) : error C2099: initializer is not a constant
C:\TEST\compileerrors\compileerrors.c(62) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(62) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(62) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(62) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(63) : error C2143: syntax error : missing ')' before 'string'
C:\TEST\compileerrors\compileerrors.c(63) : error C2143: syntax error : missing '{' before 'string'
C:\TEST\compileerrors\compileerrors.c(63) : error C2059: syntax error : '<Unknown>'
C:\TEST\compileerrors\compileerrors.c(63) : error C2059: syntax error : ')'
C:\TEST\compileerrors\compileerrors.c(66) : error C2059: syntax error : 'return'
C:\TEST\compileerrors\compileerrors.c(67) : error C2059: syntax error : '}'
Error executing cl.exe.
```

Note: Warnings should be ignored only till all the compiler errors are handled. If after your compiler errors drop to Zero and there are many warnings in code, it is very likely that you will face runtime problems in the code.

As a thumb rule, your code should always compile with Zero errors and Zero Warnings. You must address all the warnings the compiler raises as well.

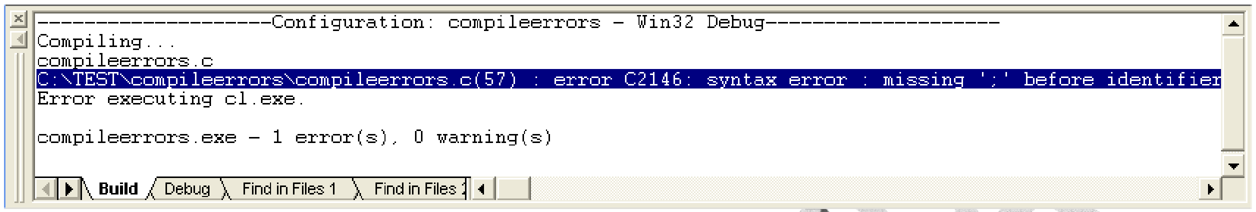
Step 9: The error is in line 47 and says "syntax error: else". Go to line 47 by double clicking on the error message. Inspect the code in Line 47.

```
/* For any product costing more than INR 500, Discount = 25% */
if (dPrice > 500.0)
        dDiscount = 25.0;
}
else {
```

The line else seems to be OK in Syntax. So, let us inspect a few lines earlier. The line 44 which contains if (dPrice > 500.0) is missing a curly brace! This has resulted in the compiler being thrown off. Fix the error.

```
    /* For any product costing more than INR 500, Discount = 25% */
    if (dPrice > 500.0) {
           dDiscount = 25.0;
    }
    else {
```

Step 10: Recompile again. Miraculously, there is only ONE compiler error!



```
--------------------Configuration: compileerrors - Win32 Debug--------------------
Compiling...
compileerrors.c
C:\TEST\compileerrors\compileerrors.c(57) : error C2146: syntax error : missing ';' before identifier
Error executing cl.exe.

compileerrors.exe - 1 error(s), 0 warning(s)
```

Note: Let us analyze what exactly happened here! The missing curly brace resulted in the compiler's parse tree being broken from that point onwards and everything else after that missing curly brace appeared as an error to the compiler. Fixing that curly brace alone resolved close to 24 errors!

It is a common tendency of programmers to spend a lot of time in such compiler errors. The following best practice or thumb rule always helps in speeding up compilation process.

Best Practice:

- Always handle the FIRST compiler error and then move on to the next.
- Whenever you see too many compiler errors which don't make sense, suspect that there might be a missing brace. Many compilers (including VC++ 6.0) are not effective at diagnosing such errors in code clearly.

Step 11: The error now is in line 57 and says "Syntax error: missing ; before identifier dPriceAfterDiscount". Go to line 57. Inspect the code in line 56 and 57.

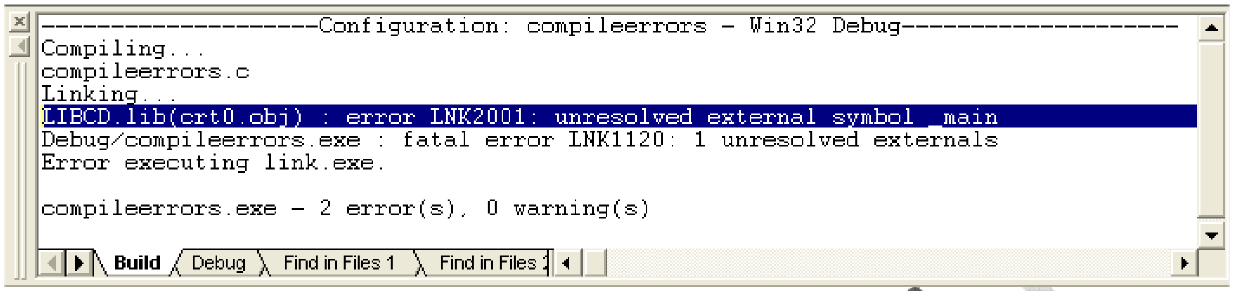
```
printf ("Sales Tax: %.2lf %%\n", dSalesTax)
dPriceAfterDiscount = dPrice - ((dPrice * dDiscount) / 100.0);
```

The line 57 does have a semicolon. But line 56 that comes just before the variable dPriceAfterDiscount does not contain a semicolon. The error description "Syntax error : missing ; before identifier dPriceAfterDiscount" indicates the same.

Include semicolon in line 56.

```
printf ("Sales Tax: %.2lf %%\n", dSalesTax);
dPriceAfterDiscount = dPrice - ((dPrice * dDiscount) / 100.0);
```

Step 12: The Compiler completes the compilation without any errors. Then the linker takes over. However, the linker will throw an error now “unresolved external symbol _main”.



```
--------------------Configuration: compileerrors - Win32 Debug--------------------
Compiling...
compileerrors.c
Linking...
LIBCD.lib(crt0.obj) : error LNK2001: unresolved external symbol _main
Debug/compileerrors.exe : fatal error LNK1120: 1 unresolved externals
Error executing link.exe.

compileerrors.exe - 2 error(s), 0 warning(s)
```

Note: Unlike compiler errors, linker errors don’t come with a line number! When the linker does not find a required method or variable that is being used in source code, it will throw up an error.

The best way to address linker errors is to see if the symbol (function name or variable name) whose name the linker shows exists in code or not.

In this case, the linker is saying the symbol _main is unresolved. So, the most likely place to find this problem is near the declaration of function main.

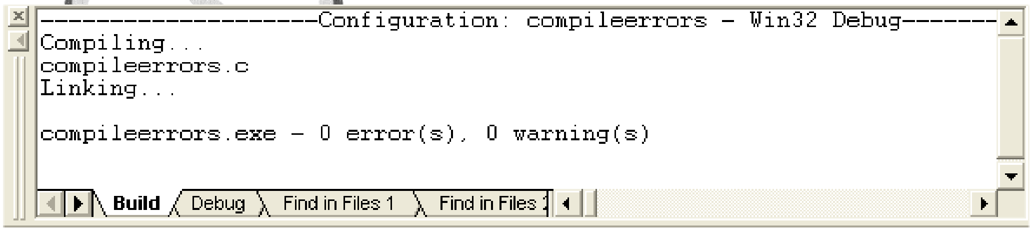
Step 13: Go to the declaration of function ‘main’.

```
int mains (int argc, char** argv)
```

The name of function main has a typo! It is spelt as ‘mains’ and therefore the linker is not able to find that symbol! Correct the typo.

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

Step 14: Recompile the program. The program should compile with Zero errors and Zero warnings.

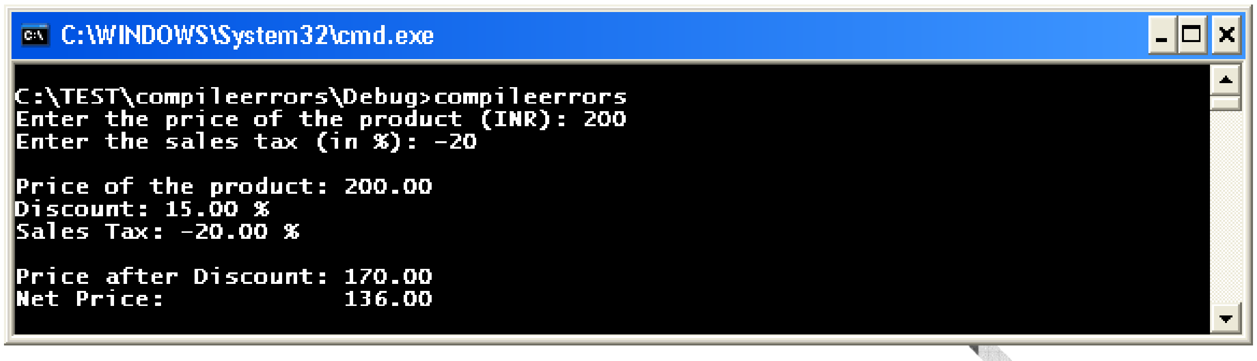


```
--------------------Configuration: compileerrors - Win32 Debug-------
Compiling...
compileerrors.c
Linking...

compileerrors.exe - 0 error(s), 0 warning(s)
```

Step 15: Run the program again. Supply legal values when the program prompts. The program should run successfully.

Step 16: Now, run the program again. This time, Supply the Sales tax as -20.


C:\WINDOWS\System32\cmd.exe

```
C:\TEST\compileerrors\Debug>compileerrors
Enter the price of the product (INR): 200
Enter the sales tax (in %): -20

Price of the product: 200.00
Discount: 15.00 %
Sales Tax: -20.00 %

Price after Discount: 170.00
Net Price:            136.00
```

The program accepted a negative value and the sales tax was computed as negative! The program compiled and executed successfully, but has a problem because error conditions are not being handled. So, the price after tax has become less than the price after discount!

Step 17: Let us add some error handling code to prevent this scenario.

The Sales Tax should be in the range of 0-100% only. Any other values should be treated as illegal.

Go to the place in code where Sales Tax in percentage is being accepted from the user.

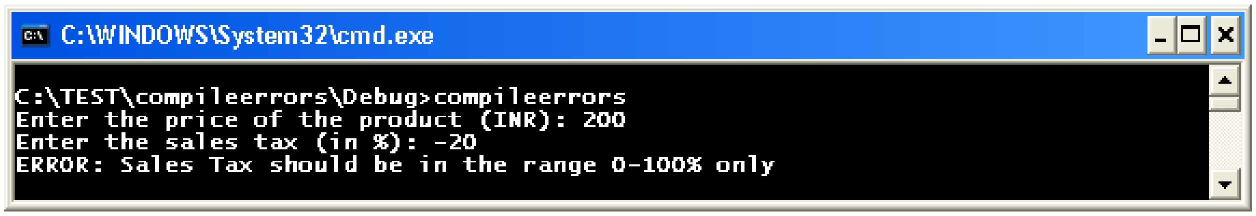
```
/* Read the Sales Tax in percent */
printf ("Enter the sales tax (in %%): ");
scanf ("%lf", &dSalesTax);
fflush (stdin);
```

Step 18: Add the following lines of code (in bold) after the fflush (stdin) statement.

```
/* Read the Sales Tax in percent */
printf ("Enter the sales tax (in %%): ");
scanf ("%lf", &dSalesTax);
fflush (stdin);

/* Error Handling code - if sales tax has illegal values */
if ((dSalesTax < 0.0) || (dSalesTax > 100.0)) {
     printf ("   ERROR: Sales Tax should be in the
                 range 0-100%% only\n");
     exit (1);
}
```

Step 19: Recompile the program and run it again. Again, give a legal value for the price of the product, but -20 for Sales Tax in percentage. Your program now handles error conditions effectively.



```
C:\TEST\compileerrors\Debug>compileerrors
Enter the price of the product (INR): 200
Enter the sales tax (in %): -20
ERROR: Sales Tax should be in the range 0-100% only
```

Step 20: Similarly add error handling code required for the price of the product as well. (Hint: Price of the product cannot be Zero or less). Recompile and test it.

Summary of this assignment: In this assignment, you have learnt:

- How to correct compilation and linker errors

## Assignment 13: Using ‘switch’ statement

Objective: To understand the usage of ‘switch’ statement.

Problem Description: Extend the supplied source code VariablesDemo.c which is debugged and error free to compute the Basic Salary according to the Job Band. The details are as follows:

| Job Band | Basic Salary(In Indian Rupees) |
|---|---|
| A | 10,000 |
| B | 15,000 |
| C | 35,000 |
| D | 50,000 |

- Declare Job Band as  char variable
- Accept the Job Band
- Use the switch statement to compute the basic salary
- Print the Basic Salary

Estimated time: 20 minutes

Summary of this assignment: In this assignment, you have learnt:

- switch construct usage

## Assignment 14: Understanding the working of while loop –(1)

Objective: To understand the working of while loop.

Note: A text file whileLoopWorking1.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Problem Description: Write a program to find the product of first 5 numbers (1 to 5)

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'whileLoopWorking1'

Step 2: Add the source code, whileLoopWorking1.c provided into the project

Step 3: Go through the code and fill the missing parts (?) in the Trace table given below

| Iteration Number | `iIndex` | `iMultiple` |
|---|---|---|
| 1 | 1 | 1 |
| 2 | ? | ? |
| 3 | 3 | 6 |
| 4 | ? | ? |
| 5 | ? | ? |

Step 5: Identify the output of code and verify that by executing the program using Microsoft Visual Studio

Summary of this assignment:
In this assignment, you have learnt:

- Working of while loop

## Assignment 15: Understanding the working of while loop – (2)

Objective: To understand the working of while loop.

Note: A text file whileLoopWorking3.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Problem Description: Go through the code and create Trace table

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'whileLoopWorking3'

Step 2: Add the source code, whileLoopWorking3.c provided into the project

Step 3: Understand the code and fill the blanks in the following Trace table according to the above code

| Iteration Number | iIndex | iSum |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Step 4: Identify the output of code and verify that by executing the code using Microsoft Visual Studio

Summary of this assignment:
In this assignment, you have learnt:
Working of while loop

## Assignment 16: Understanding the working of for loop – (1)

Objective: To understand the working of for loop

Note: A text file forLoopWorking1.c is provided to you in Supplied Source Code folder in the Lab Guide folder

Problem Description: Write a program to find the sum of first 5 numbers (1 to 5)

Estimated time: 10minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'forLoopWorking1'

Step 2: Add the source code, forLoopWorking1.c provided into the project

Step 3: Go through the code and understand the following Trace table according to the above code

| Iteration Number | iIndex | iSum |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 3 |
| 3 | 3 | 6 |
| 4 | 4 | 10 |
| 5 | 5 | 15 |

Step 4: Identify the output of code and verify it by executing the code using Microsoft Visual Studio

Summary of this assignment:

In this assignment, you have learnt:

- Working of for loop

## Assignment 17: Understanding the working of for loop – (2)

Objective: To understand the working of for loop.

Note: A text file forLoopWorking2.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Problem Description: Write a program to find the odd multiples among the first 10 multiples of first 10 numbers.

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'forLoopWorking2'

Step 2: Add the source code, forLoopWorking2.c provided into the project

Step 3: Go through the code and generate the Trace table for the same

Step 4: Identify the output of code and verify that by executing the code using Microsoft Visual Studio

Summary of this assignment:
In this assignment, you have learnt:

- Working of for loop

## Assignment 18: Understanding the working of for loop – (3)

Objective: To understand working of for loop and its usage

Problem Description: Write a program to generate the multiplication table for a given number

Estimated time: 20 minutes

Summary of this assignment:
In this assignment, you have learnt:
for loop usage

## Assignment 19: Understanding the working of do while loop –(1)

Objective: To understand the working of do while loop.

Note: A text file doWhileLoopWorking1.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Problem Description: Write a program to calculate the square of first 5 numbers

Estimated time: 10 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name ‘doWhileLoopWorking1’

Step 2: Add the source code, doWhileLoopWorking1.c provided into the project

Step 3: Go through and fill the following Trace table according to the above code

| Iteration Number | iIndex | iResult |
|---|---|---|
| 1 | 1 | 1 |
| 2 | ? | ? |
| 3 | 3 | 9 |
| 4 | ? | ? |
| 5 | 5 | 25 |

Step 4: Identify the output of code and verify that by executing the code using Microsoft Visual Studio

Summary of this assignment:
In this assignment, you have learnt:

- Working of do while loop

## Assignment 20: Understanding the working of do while loop – (2)

Objective: To understand the working of do while loop.

Note: A text file dowhileLoopWorking3.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Problem Description: Go through the code and create traceability table for loop

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name ‘dowhileLoopWorking3’

Step 2: Add the source code, dowhileLoopWorking3.c provided into the project

Step 3: Understand the code and Fill the following traceability table according to the above code

| Iteration Number | iIndex | iSum |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Step 4: Identify the output of code and verify that by executing the code using Microsoft Visual Studio

Summary of this assignment:
In this assignment, you have learnt:

- Working of `do while loop`

## Assignment 21: Understanding the working of do while loop – (3)

Objective: To understand working of do while loop and its usage

Problem Description: Write a program to count the number of positive, negative and zero values entered by the user.

Estimated time: 20 minutes

Summary of this assignment:
In this assignment, you have learnt:

- do while loop usage

## Assignment 22: Exercises for Self Review

1. Find the output of the following program:

```
int main(int argc, char **argv) {
        int iValue=156;
        if (iValue = 0) {
                printf("%d",iValue + 10);
        }
        else {
                printf("%d",iValue – 10);
        }
        return 0;
}
```

2. Find the output of the following code snippet:

```
int iValue = 10;
while (iValue <= 100) {
        printf("%d\n",iValue);
}
```

3. Find the output of the following code snippet:

```
int iValue;
for (iValue=10; iValue >= 1; iValue--) {
        printf("%d\n",iValue);
}
```

4. What is the output of the following code snippet?

```
int main(int argc,char **argv) {
   int iNumber = 5;
   while(1 == iNumber){
          iNumber = iNumber - 1;
          printf("%d ",iNumber);
          --iNumber;
  }
  return 0;
 }
```

5. What is the output of the following code snippet?

```
int main(int argc,char **argv) {
    int iIndex;
    for(iIndex = 20; iIndex > 0;iIndex--){
          if(0 != iIndex % 2){
                   printf("%d ",iIndex);
          }
     }
   return 0;
 }
```

6. Find the output of the following program:

```
int main(int argc, char **argv) {
        int iValue=2;
        switch(iValue) {
                default: printf("Not Valid ");
                case 1: printf("ONE ");
                        break;
```

```
            case 2: printf("TWO");
                    break;
        }
        return 0;
}
```

7. Find the output of the following program:

```
int main(int argc, char **argv) {
        int iValue=1;
        switch(iValue) {
                default: printf("Not Valid ");
                         break;
                case 1: printf("ONE ");
                        break;
                case 2: printf("TWO");
                        break;
        }
        return 0;
}
```

8. Debug the following program and fix the error:

```
/*****************************************************************************************
* Filename    : LeapYear.c
*  Author : RV University, SOCSE
* Date    : 17-09-2023
* Description : Program to check whether the given year is leap year or not
*****************************************************************************************/

/* Include files */
#include<stdio.h>

/*****************************************************************************************
* Function              : main()
* Description  : main fuction to check whether the given year is leap year
*              or not
* Input Parameters:
*       int argc - Number of command line arguments
*   char **argv  The command line arguments passed
* Returns: 0 on success to the operating system
***************************************************************************************/

void main(int argc,char **argv) {
        /*Variable declaration*/
        int iYear;
        printf("enter the year\n");
```

```
            scanf("%d",&iYear);

            /*The year is Leap if it is century year and is divisible by 400)
            The year is Leap if it is no-century year and is divisible by 4)*/
            if(0 == iYear%100){

                    if(0 == iYear%400){
                            printf("Leap Year\n");
                    }
                    else {
                            printf("Not a Leap Year\n");
                    }
             }
            else {
                            if(0 == iYear%4){
                                    printf("Leap Year\n");
                            }
                            else {
                                    printf("Not a Leap Year\n");

                             }
             return 0;
     }


    /***********************************************************************************
    * End of LeapYear.c
    ***********************************************************************************/
```

9. Find the output of the following code snippet:

```
int main(int argc,char **argv)  {
    int iValue=1;
    do {
          printf("%d ",iValue);
          if ((iValue % 2) !=0 ) {
                  continue;
          }
         iValue++;
   } while (iValue < 5);
   return 0;
}
```

10. Find the output of the following code snippet:

```
int main(int argc,char **argv)  {
        int iValue1,iValue2;
        for (iValue1=1; iValue1<=5; iValue1++) {
                for (iValue2=1;iValue2<=3; iValue2++) {
                        if (iValue1 % iValue2 == 0) {
                                break;
                        }
                }
                printf("%d ",iValue1);
        }
 return 0;
}
```

11. What is the output of the following code snippet?

```
int main(int argc,char **argv) {
        int iCount1,iCount2 ;
        for(iCount1 = 1;iCount1<=4; iCount1++){

                for(iCount2=1;iCount2<=4;iCount2++){

                        if(iCount1 == iCount2){
                                continue;
                        }
                         printf("%d ",iCount2);
                  }
                printf("\n\n");
                printf("%d ",iCount1);
        }
         return 0;
 }
```

12. . What is the output of the following code snippet?

```
 int main(int argc,char **argv) {
        int iIndex = 1;
        for(;;){
                  if(iIndex > 20){
                      break;
                  }
                  else {
                       printf("%d ",iIndex);
                       iIndex = iIndex * 2;
                  }

          }
          return 0;
}
```

## Ac Assignments

### Assignment 23: Declaring and using 1-D arrays

Objective: To declare and initialize arrays

Problem Description: Write a program to declare an array to store the employee ids for 4 employees. Initialize the array elements with following values (71005, 71006, 71007, 71008) and print the same

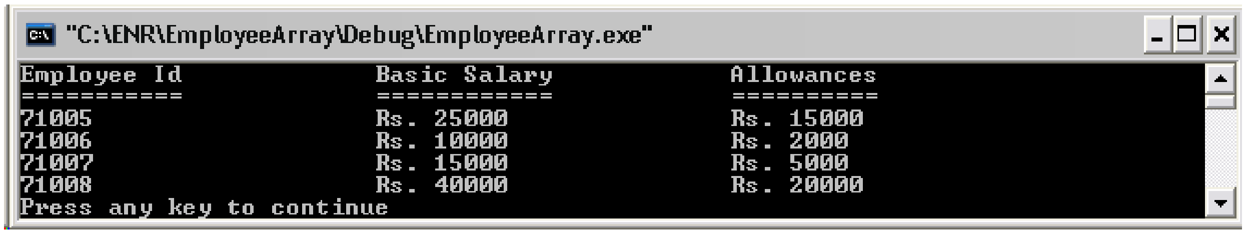
Estimated time: 30 minutes

Step 1: Extend the program to declare and initialize arrays to store the basic salary and allowances.

Initialize the array for storing basic salary with the following values :-
25000,10000,15000,40000

Initialize the array for storing allowances with the following values :-
15000,2000,5000,20000

Step 2: Display all the details in a tabular format (Refer to the sample screen given below).



```
Employee Id             Basic Salary            Allowances
===========             ============            ==========
71005                   Rs. 25000               Rs. 15000
71006                   Rs. 10000               Rs. 2000
71007                   Rs. 15000               Rs. 5000
71008                   Rs. 40000               Rs. 20000
Press any key to continue
```

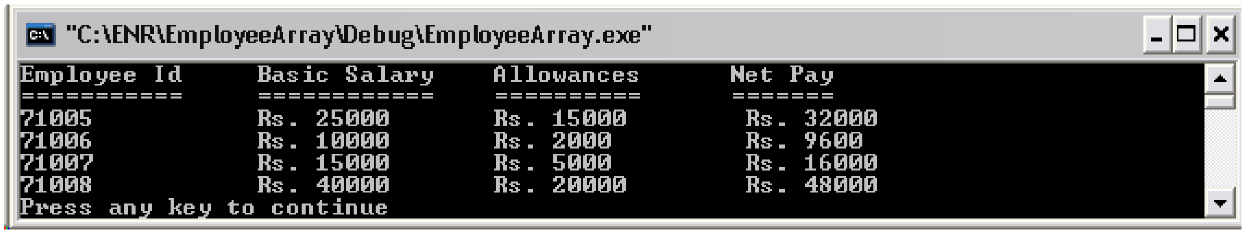
Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Step 3: Extend the program to declare an array to store the net pay.

Note: Reuse the formulas used in supplied source code SalaryComputation.c to calculate the income tax, gross, net salary.

Step 4: Compute the net pay and store the net pay into this array.

Step 5: Display all the details in a tabular format (Refer to the sample screen given below).



```
"C:\ENR\EmployeeArray\Debug\EmployeeArray.exe"
Employee Id      Basic Salary     Allowances       Net Pay
===========      ============     ==========       =======
71005            Rs. 25000        Rs. 15000         Rs. 32000
71006            Rs. 10000        Rs. 2000          Rs. 9600
71007            Rs. 15000        Rs. 5000          Rs. 16000
71008            Rs. 40000        Rs. 20000         Rs. 48000
Press any key to continue
```

Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Summary of this assignment:
In this assignment, you have learnt:

- Declaring an array and initializing it during declaration
- Accessing the individual array elements

## Assignment 24: Referencing Arrays – Debugging Assignment

Objective: To understand the valid array references

Problem Description: To declare an array and understand the valid array references

Note: A text file EmployeeArray.c is provided to you in Supplied Source Code folder within the Lab Guide folder

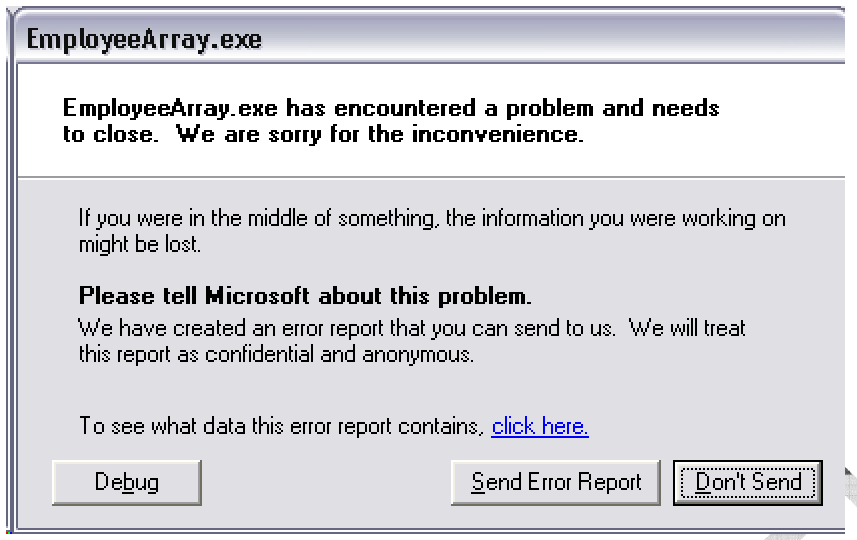
Estimated time: 10 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'EmployeeArray'

Step 2: Add the source code, EmployeeArray.c provided into the project

Step 3: Compile the program. There will not be compilation errors or warnings if there are no typographical errors.

Step 4: Execute the program. There will be a run time error as given below:


EmployeeArray.exe

EmployeeArray.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

**Please tell Microsoft about this problem.**
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

Step 5: Identify and fix the bug.

Note: The C compiler does not check the array bounds and they do not generate compilation errors for invalid array references. But they may lead to run time errors. It is the responsibility of the programmer to take care of array bounds.

Summary of this assignment:
In this exercise, you have learnt:

- Valid and Invalid array references
- Array bounds checking

## Assignment 25: Using 'while' loop with an Array

Objective: To understand using 'while' loops.

Problem Description: Write a program to display the employee details one at a time using a while loop.

Note: A text file whileloops.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 15 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'whileloops'

Step 2: Add the source code, whileloops.c provided into the project

Step 3: Write the missing code given as comments in the supplied source code

Step 4: Compile the program and execute the program

Note: fflush(stdin) is used to clear the input buffer. This is required before any character input is accepted.

Summary of this assignment:
In this assignment, you have learnt
- The use of ‘while’ loop with an array

## Assignment 26: Using ‘for’ loop with an Array

Objective: To understand the usage of ‘for’ loop.

Problem Description: Write a program to accept a job band from the user and to count the number of confirmed employees with the entered job band.

Estimated time: 35 minutes

Data Structures:
Declare an array to store the employee ids and initialize the array with the employee ids 71005, 71006, 71007, 71008, 71009, 71010.

Declare an array to store the job band of employees and initialize the array with the job bands ‘D’, ‘B’, ‘C’, ‘A’, ‘B’,’C’.

Declare another array to store the confirmation status of employees and initialize them with the following: ‘Y’, ‘N’, ‘Y’,’Y’,’Y’,’N’

There is a one to one relationship between these arrays. i.e, employee 71005 has job band ‘D’ and the confirmation status is ‘Y’. Employee 71006 has job band ‘B’ and the confirmation status is ‘N’ and so on.

The program should accept a job band and calculate the number of confirmed (those with confirmation status as ‘Y’) employees with the accepted job band.

For example, if the accepted Job Band is ‘C’, then the Number of Confirmed employees will be 1 since there are two employees – 71007 & 71010 with Job Band ‘C’ and only 71007 is having confirmation status as ‘Y’.

Summary of this assignment:
In this assignment, you have learnt
- Usage of for loops with arrays

## Assignment 27: Design of menus

Objective: To understand how to design menus and navigate between menu options.

Problem Description: Write a menu driven program to change job band, confirm an employee and generate reports.

Note: A text file MenuDesign.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 30 mins

Step 1: Create an empty project in Visual C++ (Console Application) with name 'MenuDesign'

Step 2: Add the source code, MenuDesign.c provided into the project

Step 3: Write a menu driven program with the following options:

1. Change Job Band
2. Confirm Employee
3. Report
4. Exit

Step 4: Data Structures: Reuse the data structures and initial values used for Assignment No 25

Step 5: Implement the following functionalities:

1. Change Job Band: Accept the employee id and the new job band. Update the job band of the employee. If an invalid employee id is entered (employee id not found in the array), display a suitable error message.
2. Confirm Employee: Accept the employee id. Update the confirmation status to 'Y'. If an invalid employee id is entered (employee id not found in the array), display a suitable error message.
3. Report: Display the employee id, JobBand and the confirmation status of all employees in a tabular format.
4. Exit: This option will enable the user to quit the program.

Note: The program written for Assignment no. 25 can be reused for this assignment.

Summary of this assignment:
In this assignment, you have learnt

- Design of menus
- Data validation

## Assignment 28: Declaring and Using 2-D arrays

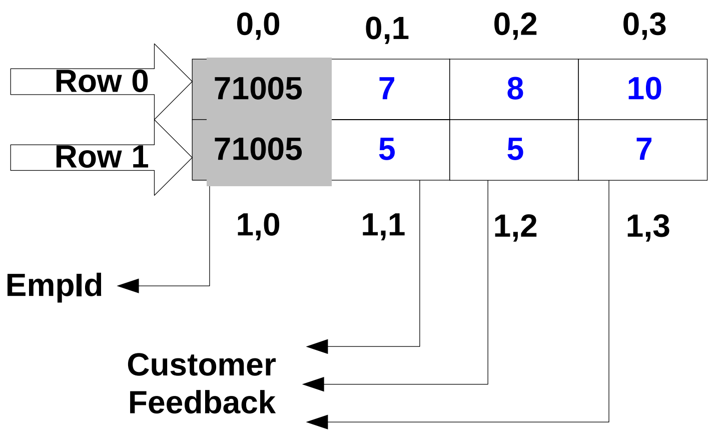
Objective: To understand the need for 2-D arrays to store a tabular data.

Problem Description:
Write a program to find the average feedback given by the customers for the employees working for them.

The employee id and the customer feedback (Both employee id and the feedback are integers) can be stored into a 2-D array. When there are 2 employees and 3 customers, the total size of the array should be minimum 6 (2 rows and 3 columns).

Note: Here a new array should be written and should not reuse the 1-D arrays written for previous assignments.

This can be represented as follows:



Note: A text file AverageCustomerFeedback2DArrays.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 20 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'AverageCustomerFeedback2DArrays'

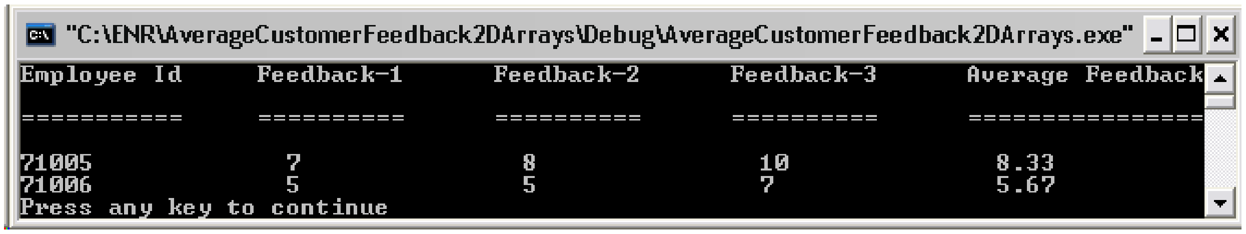
Step 2: Add the source code, AverageCustomerFeedback2DArrays.c provided into the project

Step 3: Compile and execute the program

Step 4: Declare a 1-D array to store the average customer feedback for two employees.

Step 5: Compute the average customer feedback for each employee and store it into the 1-D array.

Step 6: Display the details as given below:



```
"C:\ENR\AverageCustomerFeedback2DArrays\Debug\AverageCustomerFeedback2DArrays.exe"
Employee Id     Feedback-1      Feedback-2      Feedback-3      Average Feedback

===========     ==========      ==========      ==========      ================

71005             7               8               10              8.33
71006             5               5               7               5.67
Press any key to continue
```

Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Summary of this assignment:
In this assignment, you have learnt:

- Declaring 2-D arrays
- Referencing 2-D arrays

## Assignment 29: Exercises for Self Review

1. Find the output of the following program:
```
int main(int argc, char **argv) {
        int aiValues[]={10,20,30,40};
        printf("%d",aiValues[1]);
        return 0;
}
```

2. Find the output of the following program:
```
int main(int argc, char **argv) {
        int aiValues[10]={10,20,30,40};
        printf("%d",sizeof(aiValues));
        return 0;
}
```

3. Find the output of the following program:
```
int main(int argc, char **argv) {
        int aiArray[2][2]={10,20,30,40};
        printf("%d",aiArray[1][1]);
        return 0;
}
```

## Activity / Assignments-5

## Assignment 30: Linear Search

Objective: To understand how to parse arrays and search for an element. Based on the search index, operations can be performed on the related arrays

Problem Description: Write a program to search for an Employee Id and update the salary of the corresponding employee by 20 percent.

Note: A text file SearchArray.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 15 min

Step 1: Create an empty project in Visual C++ (Console Application) with name ‘SearchArray’

Step 2: Add the source code, SearchArray.c provided into the project

Step 3: Compile and execute the program and observe the output

Summary of this assignment:
In this assignment, you have learnt

- To parse an array to find an element
- Using the position of occurrence of the element, make modification to related array at that position

## Assignment 31: Declaring and Using Pointers

Objective: To Declare and use the pointer to access the values of the variables.

Problem Description: Declare the pointer variables and display their size.

Note: A text file EmployeePointer.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 20 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name ‘EmployeePointer’

Step 2: Add the source code, EmployeePointer.c provided into the project

Step 3: Compile the program and execute the program

Step 4: Declare and initialize variables to store basic salary (double data type) and allowances (float data type).

Step 5: Declare two pointers that can point to basic salary and allowances.

Step 6: Initialize the pointers to the address of the respective variables.

Step 7: Display basic salary and allowances through the pointer.

Step 8: Display the size of basic salary and allowances variables.

Step 9: Display the size of basic salary pointer and allowances pointer

Summary of this assignment:
In this assignment, you have learnt:
- Declaring and initializing pointers
- Accessing the value using the pointer

## Assignment 32: Un initialized Pointers – Debugging Assignment

Objective: To understand the need for initialization of pointers.

Problem Description: Declare a pointer and dereference (Reference the value) without initializing it. Correct the error encountered.

Note: A text file PointerDebug.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 10 minutes

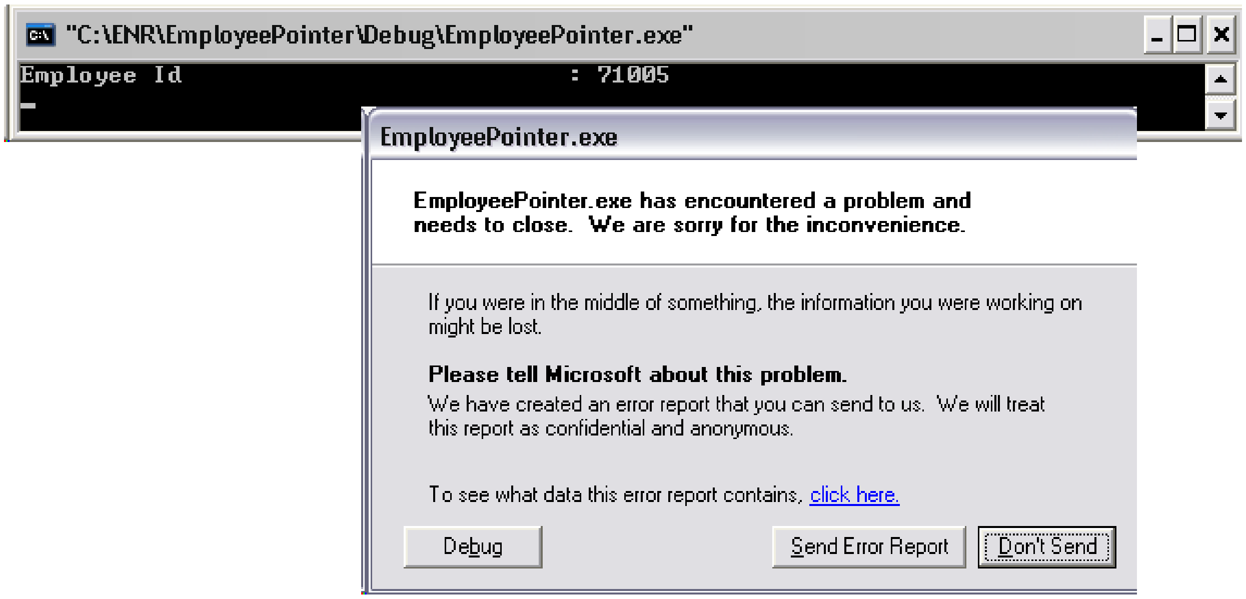
Step 1: Create an empty project in Visual C++ (Console Application) with name 'PointerDebug'

Step 2: Add the source code, PointerDebug.c provided into the project

Step 3: Compile the program

Step 4: Ignore the warnings and execute the program.

Step 5: The program will raise a run time error as shown below:



"C:\ENR\EmployeePointer\Debug\EmployeePointer.exe"

```
Employee Id                         : 71005
_
```

EmployeePointer.exe

**EmployeePointer.exe has encountered a problem and needs to close. We are sorry for the inconvenience.**

If you were in the middle of something, the information you were working on might be lost.

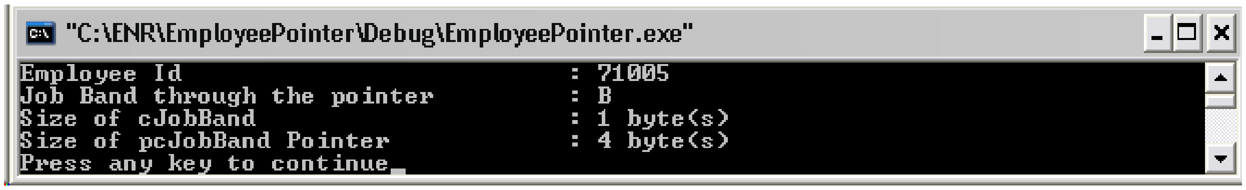
**Please tell Microsoft about this problem.**

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

Step 6: Fix the bug. (Hint: Check the pointer initialization)

Step 7: Recompile the program and execute it. If there are no typographical errors, the program should execute successfully and the output should be as shown below:



"C:\ENR\EmployeePointer\Debug\EmployeePointer.exe"

```
Employee Id                         : 71005
Job Band through the pointer        : B
Size of cJobBand                    : 1 byte(s)
Size of pcJobBand Pointer           : 4 byte(s)
Press any key to continue_
```

Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Summary of this assignment:
In this assignment, you have learnt:

- Debugging run time errors related to pointers
- The need for initializing the pointer variables

## Assignment 33: Declaring and Using Strings

Objective: To understand the usage of strings.

Background: Strings are stored using 1-D char arrays. Only one string can be stored using a 1-D char array. To print the strings, %s conversion specifier is used. Strings have a terminator character at the end of the string.

Problem Description: Extend the program EmployeePointer.c to declare the name of the employee (1-D char array). Initialize the employee name. Display employee id and name.

Estimated time: 20 minutes

Note: Since scanf () function expects pointers, the address of the variable should be supplied. Example: `scanf("%d",&iEmpId);`
To read a string, only the array name is supplied. Since the name of an array is a pointer (address by itself), it is not preceded by & symbol.

Example:
```
char acEmpName[20];
scanf("%s",acEmpName);
```

Summary of this assignment:
In this assignment, you have learnt:

- Declare and Initialize Strings

## Assignment 34: Strings and char pointer

Objective: To understand the usage of char pointer to store strings.

Background: Strings can be stored using a char pointer. Only restriction is that such strings cannot be accepted using scanf or gets function but can be initialized while declaration. To print the strings, %s conversion specifier is used.

Problem Description: Modify the EmployeePointer.c program to use char pointer to declare the employee name. Initialize employee name. Display employee id and name (Use the char pointer declared for the name).

Estimated time: 15 minutes

Summary of this assignment:
In this assignment, you have learnt:

- Usage of char pointer

## Assignment 35: Automation of Credit Point calculation for a Mobile company

Objective: To understand the design of menus and data validation.

Problem Description:
“Airtel” wants to automate the process of giving credit points to its customers based on their monthly bill to encourage more usage by the customers

To implement this, write a menu driven program with the following options.

1. Update Credit Points
2. Reset Credit Points
3. Customer Report
4. Exit

Estimated time: 45 min

Step 1: Data Structures:
Declare an array to store the custid (for 5 customers) and initialize it with the following customer ids: 71005, 71006, 71007, 71008, 71009

Declare an array to store the name of the customers (for 5 customers) and initialize it with the following customer names: “Lara”, “Lin”, “Lara”, “Krishna”, “Sophia”

Declare an array to store the credit points secured by the customers (for 5 customers) and initialize it with the following: 120, 115, 25, 0, 20

There is a one to one relationship between these arrays. That is, customer id is 71005, customer name is “Lara” and the points are 120. Customer id is 71006, customer name is “Lin” and the points are 115and so on.

Step 2: Implement the following functionalities:

1. Update Credit Points
   Accept the customer id and the monthly bill amount in Dollars. Add the points scored for the customer based on the monthly bill amount

   | Monthly Bill Amount | Points |
   |---|---|
   | < $50 | 0 |
   | >= $50 and < $100 | 5 |
   | >= $100 and < $200 | 10 |
   | >= $200 and < $ 500 | 15 |
   | >= $500 | 20 |

   Display the total credit points secured by the customer.

   Validation: The accepted customer id should be an existing customer id and the monthly bill amount should be greater than zero. If an invalid customer id or invalid monthly bill is entered, then display a suitable error message.

2. Reset Credit Points
   Reset (Make it to zero) the credit points for all the customers. Display a success message after resetting the credit points for all the customers.

3. Customer Report
   Display the customer id, customer name and credit points for all the customers who have more than 100 credit points. If there is no customer with more than 100 points then display a suitable message. Refer to the report layout given below:

```
                              Airtel
                              =====

        Customer Id          Customer Name       Credit Points
        ==========           =============       ===========
        -------------        -----------------   ---------------
        -------------        -----------------   ---------------
```

   Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

4. Exit
   This enables the user to quit the program.

Summary of this assignment:
In this assignment, you have learnt:

- The usage of loops
- Data validation
- Menu design

## Assignment 36: Exercises for Self Review

1. Debug the following code for the logical error:

```
int main(int argc, char **argv) {
        int aiEmployeeId[4]={1001,1002,1003,1004};
        int iEmployeeId,iIndex,iFlag=0;

        /* Input from the user */
        printf("Enter the Employee Id to be searched\n");
        scanf("%d",&iEmployeeId);

        /* searching the accepted Employee Id in the given list */

        for(iIndex=0;iIndex<4;iIndex++){
                if(aiEmployeeId[iIndex] == iEmployeeId){
                        iFlag = 1;
                }
                else{
                        break;
```

```
                    }
            }

            /* checking the flag value to display the appropriate message */
            if(1 == iFlag ){
                    printf("Employee Id found");
            }
            else{
                    printf("Employee Id not found");
            }
        return 0;
    }
```

2. Find the output of the following program:

```
int main(int argc, char **argv) {
        int iValue=100;
        int *piValue = &iValue;
        *piValue = *piValue + 5;
        printf("%d",iValue);
        return 0;
}
```

3. Find the output of the following program:

```
int main(int argc, char **argv) {
            char acString[20] = ”Programming”;
            printf(“%d”,sizeof(acString));
            return 0;
}
```

4. What is the difference between scanf() and gets()?

5. Debug the following code snippet:

```
#include <stdio.h>
int main (int argc, char** argv) {
        int iEmpId;
        char acEmpName[20];
        printf("Enter the employee id ");
        scanf("%d",&iEmpId);
        printf("Enter the employee name ");
        scanf("%s",acEmpName[0]);
        printf("\nEmployee Id : %d\n",iEmpId);
        printf("Employee Name: %s\n",acEmpName);
        return 0;
}
```

6. Debug the following code snippet:

```
int main(int argc, char **argv) {
        char acString[15];
        acString = "Assignment";
        printf("%s",acString);
        return 0;
}
```

## Activity / Assignments -6

### Assignment 37: Usage of String Handling Functions - strlen

Objective: To understand the usage of string handling functions.

Background: The length of a string can be found out using strlen() function

Problem Description: Extend the program EmployeePointer.c to check if the length of the employee name is more than 25 chars, if so display an appropriate error message.

Estimated time: 15 minutes

Summary of this assignment:
In this assignment, you have learnt:
Usage of strlen()

### Assignment 38: Usage of String Handling Functions - strcat

Objective: To understand the usage of string handling functions.

Background: Two strings can be combined using strcat() function

Problem Description: Modify the program EmployeePointer.c to accept the Employee Name as First name, Middle name and Last name, and Department Name. After accepting, First name, Middle name and Last name must be combined to give Employee name.

Display Employee name, Employee id and Department Name

Use scanf for accepting First name, Middle Name and Last Name
Use gets for accepting Department Name

Note: scanf() reads till a space, tab space or enter key is pressed . It can read only a single string.
gets() reads until the enter key is pressed. It can read multiple strings with blank spaces.

Estimated time: 15 minutes

Summary of this assignment:
In this assignment, you have learnt:
- Usage of strcat()
- Difference between accepting strings using scanf and gets

## Assignment 39: Usage of String Handling Functions - strcmpi

Objective: To understand the usage of string handling functions.

Background: Two strings can be compared using strcmpi() (case insensitive) function

Problem Description: Modify the program EmployeePointer.c to accept the details of 2 employees and check whether their Department names are same, if so display a message as "Both the employees are working in the same department".

The employee details should include Employee Id, Employee Name and Department Name.

A good user interface design should display first the list of all valid Department names and then ask the user to input the details for 2 employees. In this way the user need not memorize all the valid Department names.

Estimated time: 20 minutes

Summary of this assignment:
In this assignment, you have learnt:
Usage of strcmpi()

## Assignment 40: Usage of String Handling Functions – strcmp and strcmpi

Objective: To understand the usage of string handling functions.

Background: Two strings can be compared using strcmp() (case sensitive) and strcmpi ( case insensitive) functions

Problem Description: Write a program to accept the username and password for your system login. Perform a case sensitive comparison for password and a case insensitive comparison for username. If a match is found, display a success message.

Note: The username and password can be hard-coded in the program.

Estimated time: 20 minutes

Summary of this assignment:
In this assignment, you have learnt:

- Usage of strcmpi()
- Usage of strcmp()

## Assignment 41: Array of Strings

Objective: To understand the usage of array of strings (2-D array of char).

Background: 1-D array can store only one string. To store multiple strings 2-D array of char (array of strings) is used. For example, to store employee names for more than one employee array of strings is used. To access a string in an array of strings, only the row index is supplied. When a row and a column index is supplied, it references a single character from the string. So if acEmpName[0] contains the string "Ram", `printf("%s",acEmpName[0]);` will print the string "Ram" and `printf("%c",acEmpName[0][2]);` will print the character 'm'.

Problem Description: Write a program to store the employee id and employee name for 3 employees.

Note: A text file ArrayofStrings.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 20 minutes

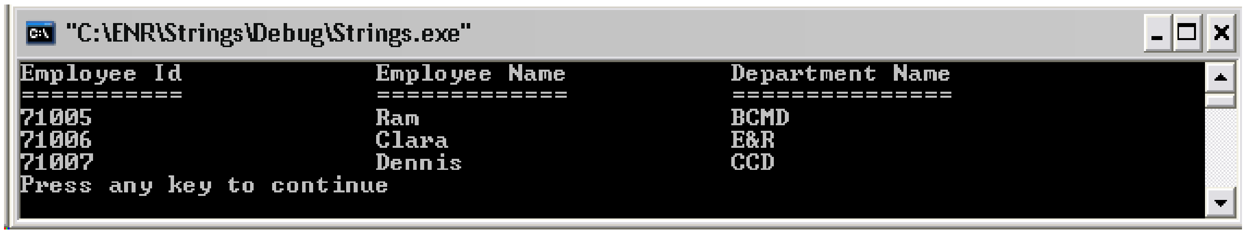
Step 1: Create an empty project in Visual C++ (Console Application) with name 'ArrayofStrings'

Step 2: Add the source code, ArrayofStrings.c provided into the project

Step 3: Compile and execute the program.

Step 4: Declare a 2-D array to store the department name of the employee (Example: E&R, CCD, BCMD, IVS etc.)

Step 5: Display the details as given below:


"C:\ENR\Strings\Debug\Strings.exe"

```
Employee Id             Employee Name           Department Name
===========             =============           ===============
71005                   Ram                     BCMD
71006                   Clara                   E&R
71007                   Dennis                  CCD
Press any key to continue
```

Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

Summary of this assignment:
In this assignment, you have learnt:

- Declaration of 2-D char arrays
- Referencing individual strings in a 2-D array of char

## Assignment 42: Usage of  sscanf() and sprintf()

Objective: To understand the usage of sscanf() and sprintf().

Problem Description:
Write a program to print employee id, employee name and residential address.

Note: A text file Usageofsscanfandsprintf.c is provided to you in Supplied Source Code folder within the Lab Guide folder

Estimated time: 30 minutes

Step 1: Create an empty project in Visual C++ (Console Application) with name 'Usageofsscanfandsprintf'

Step 2: Add the source code, Usageofsscanfandsprintf.c provided into the project

Step 3: Insert relevant code wherever necessary

Step 4: Compile and execute the program.

Summary of this assignment:
In this assignment, you have learnt:

- Usage of sscanf()
- Usage of sprintf()
- Usage of strcat()

## Assignment 43: Exercises for Self Review

1. Find the output of the following program:

```
int main(int argc, char **argv) {
                    char acString[]=”Assignment”;
                    printf(“%d %d”,strlen(acString),sizeof(acString));
                    return 0;
}
```

2. Find the output of the following program:

```
  int main(int argc, char **argv) {
                    char acString1[] = ”Exam”;
                    char acString2[] = ”exam”;
                    printf(“%d”,strcmpi(acString1,acString2));
                    return 0;
 }
```

## Activity / Assignments -7

### Assignment 44: Usage of Code Tuning Techniques – (1)

Objective: To understand the usage of code tuning techniques.

Problem Description:
Analyze the following code snippet and tune the code snippet wherever necessary.

Note: For all the assignments on code tuning techniques, the tuned code can be submitted in a .txt file.

```
iNum=1;
while(iNum<= iX/2)
{
        afSal[iNum] = 1.5* afSal[iNum];
        iNum=iNum+1;
}
```

Estimated time: 15 minutes
Summary of this assignment:
In this assignment, you have learnt:
    Usage of code tuning techniques - Minimize work performed inside Loops

### Assignment 45: Usage of Code Tuning Techniques – (2)

Objective: To understand the usage of code tuning techniques.

Problem Description:

Analyze the following code snippet and tune the code snippet wherever necessary.

Note: For all the assignments on code tuning techniques, the tuned code can be submitted in a .txt file.

```
for(iNum=1; iNum <=iTotal; iNum++) {
        afCommission[iNum] = iNum* fRevenue* fBaseCommission * fDiscount;
}
```

Estimated time: 15 minutes

Summary of this assignment:
In this assignment, you have learnt:
Usage of code tuning techniques - Minimize work performed inside Loops

## Assignment 46: Usage of Code Tuning Techniques – (3)

Objective: To understand the usage of code tuning techniques.

Problem Description:
Analyze the following code snippet and tune the code snippet wherever necessary.

Note: For all the assignments on code tuning techniques, the tuned code can be submitted in a .txt file.

```
for( iNum=2;iNum<10;iNum++){
        aiArr[1]=10;
        aiArr[iNum]=iNum*aiArr[iNum-1];
}
```

Estimated time: 15 minutes

Summary of this assignment:
In this assignment, you have learnt:
Usage of code tuning techniques - Minimize array references

## Assignment 47: Usage of Code Tuning Techniques – (4)

Objective: To understand the usage of code tuning techniques.

Problem Description:
Analyze the following code snippet and tune the code snippet wherever necessary.

Note: For all the assignments on code tuning techniques, the tuned code can be submitted in a .txt file.

```
iNum = 10;
for(iIndex=0;iIndex<10;iIndex++){
        if(iNum>20)
                printf("%d is greater than 20",iNum);
        else
                printf"%d is less than or equal to 20", iNum);
        --------------------
        --------------------
}
```

Estimated time: 15 minutes

Summary of this assignment:
In this assignment, you have learnt:
Usage of code tuning techniques - Unswitching of loops