

Lab Guide for Programming Practices



Author(s)	S.Meenakshi VijayKumar Dani
Authorized by	Srikantan Moorthy
Creation/Revision Date	Jun - 2009
Version	1.1

COPYRIGHT NOTICE

All ideas and information contained in this document are the intellectual property of Education and Research Department, Infosys Technologies Limited. This document is not for general distribution and is meant for use only for the person they are specifically issued to. This document shall not be loaned to anyone, within or outside Infosys, including its customers. Copying or unauthorized distribution of this document, in any form or means including electronic, mechanical, photocopying or otherwise is illegal.

Education and Research Department
Infosys Technologies Limited
Electronics City
Hosur Road
Bangalore - 561 229, India.

Tel: 91 80 852 0261-270
Fax: 91 80 852 0362
www.infy.com
<mailto:E&R@infy.com>

Document Revision History

Version	Date	Author(s)	Reviewer(s)	Comments
1.0	March 09	S.Meenakshi	Vijay Kumar Dani	Lab Guide for Bridge course
1.1	Jun 09	VijayKumar Dani Anil Kumar	S.Meenakshi	Changes incorporated after pilot roll out feedback
1.2	Jun 10	Anooja M Jacob Anil P	Vijaykumar Dani	Changes incorporated after changes in duration and scope of course

Contents

COPYRIGHT NOTICE	II
DOCUMENT REVISION HISTORY	III
CONTENTS.....	IV
CONTEXT	1
GUIDELINES	1
DAY-1 ASSIGNMENTS	1
ASSIGNMENT 1: PSEUDO CODE- SEQUENTIAL PATTERN	1
ASSIGNMENT 2: TEST A SEQUENTIAL PATTERN PSEUDO CODE	2
ASSIGNMENT 3: PSEUDO CODE - SELECTIONAL PATTERN(1).....	3
ASSIGNMENT 4: TEST A SELECTIONAL PATTERN PSEUDO CODE(1).....	3
ASSIGNMENT 5: PSEUDO CODE - ITERATIONAL PATTERN(1)	4
ASSIGNMENT 6: TEST AN ITERATIONAL PATTERN PSEUDO CODE (1).....	4
ASSIGNMENT 7: EXERCISES FOR SELF REVIEW:	5
DAY-2 ASSIGNMENTS	6
ASSIGNMENT 8: USING VISUAL STUDIO IDE TO CREATE THE FIRST C PROGRAM.....	6
ASSIGNMENT 9: UNDERSTANDING THE VARIABLES AND CONVERSION SPECIFIERS- DEBUGGING ASSIGNMENT	16
ASSIGNMENT 10: UNDERSTANDING THE RANGE OF DATA TYPES	18
ASSIGNMENT 11: USING CHARACTER VARIABLES-DEBUGGING ASSIGNMENT	19
ASSIGNMENT 12: ASCII USAGE	20
ASSIGNMENT 13: CODING STANDARDS	20
ASSIGNMENT 14: USING PREPROCESSOR DIRECTIVES AND ARITHMETIC OPERATORS	21
ASSIGNMENT 15: UNDERSTANDING TYPECASTING-DEBUGGING ASSIGNMENT	22
ASSIGNMENT 16: USING 'IF', 'IF-ELSE ' AND 'ELSE-IF' STATEMENT.....	23
ASSIGNMENT 17: HANDLING COMPILER AND LINKER ERRORS-DEBUGGING ASSIGNMENT	24
ASSIGNMENT 18: USING 'SWITCH' STATEMENT	30
ASSIGNMENT 19: UNDERSTANDING THE WORKING OF WHILE LOOP	30
ASSIGNMENT 20: UNDERSTANDING THE WORKING OF FOR LOOP.....	31
ASSIGNMENT 21: UNDERSTANDING THE WORKING OF DO WHILE LOOP.....	32
ASSIGNMENT 22: EXERCISES FOR SELF REVIEW	33
DAY-3 ASSIGNMENTS	37
ASSIGNMENT 23: DECLARING AND USING 1-D ARRAYS	37
ASSIGNMENT 24: REFERENCING ARRAYS - DEBUGGING ASSIGNMENT	38
ASSIGNMENT 25: USING 'FOR' LOOP WITH AN ARRAY	39
ASSIGNMENT 26: DESIGN OF MENUS	40
ASSIGNMENT 27: DECLARING AND USING 2-D ARRAYS	41
ASSIGNMENT 28: DECLARING AND USING POINTERS	42
ASSIGNMENT 29: UN INITIALIZED POINTERS - DEBUGGING ASSIGNMENT.....	43

ASSIGNMENT 30: DECLARING AND USING STRINGS	44
ASSIGNMENT 31: STRINGS AND CHAR POINTER	45
ASSIGNMENT 32: USAGE OF STRING HANDLING FUNCTIONS - STRLEN	45
ASSIGNMENT 33: USAGE OF STRING HANDLING FUNCTIONS - STRCAT	46
ASSIGNMENT 34: USAGE OF STRING HANDLING FUNCTIONS - STRCMP AND STRCMP1	46
ASSIGNMENT 35: ARRAY OF STRINGS	47
ASSIGNMENT 36: EXERCISES FOR SELF REVIEW	48
DAY-4 ASSIGNMENTS	50
ASSIGNMENT 37: DEVELOPING FUNCTIONS	50
ASSIGNMENT 38: RETURNING VALUES	53
ASSIGNMENT 39: PASS BY VALUE	53
ASSIGNMENT 40: PASS BY REFERENCE	54
ASSIGNMENT 41: PASSING 1-D ARRAYS AS ARGUMENTS TO FUNCTIONS	55
ASSIGNMENT 42: PASSING 2-D ARRAYS AS ARGUMENTS TO FUNCTIONS	55
ASSIGNMENT 43: AUTOMATION OF CREDIT POINT CALCULATION FOR A MOBILE COMPANY	56
ASSIGNMENT 44: RECURSIVE FUNCTION	58
ASSIGNMENT 45: CREATING STRUCTURES	58
ASSIGNMENT 46: EMPLOYEE NAME VALIDATION	60
ASSIGNMENT 47: PASSING STRUCTURES TO FUNCTIONS USING PASS BY VALUE	60
ASSIGNMENT 48: PASSING STRUCTURES TO FUNCTIONS USING PASS BY REFERENCE	61
ASSIGNMENT 49: IMPLEMENTING THE ARRAY OF STRUCTURES - PART I	61
ASSIGNMENT 18: USING THE ARRAY OF STRUCTURES - PART II	63
ASSIGNMENT 50: EXERCISES FOR SELF REVIEW	64
DAY-5 ASSIGNMENTS	71
DAY-6 ASSIGNMENTS	71
ASSIGNMENT 51: IDENTIFYING TEST CASES USING BOUNDARY VALUE ANALYSIS	72
ASSIGNMENT 52: TESTING A PROGRAM USING THE TEST CASES AND LOGGING TEST RESULTS AND DEFECTS	73
ASSIGNMENT 53: DEBUGGING USING IDE (STEP-BY-STEP EXECUTION)	74
ASSIGNMENT 54: EXERCISES FOR SELF REVIEW	75
APPENDIX: ADDITIONAL ASSIGNMENTS	75

Context

This document contains assignments to be completed as part of the hands on for the subject Programming Practices (Course code: LA1027).

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
- These assignments contain coding exercises, debugging exercises, coding standards exercises and self review assignments
- Solving these exercises methodically would provide confidence to the learner to attempt the assessments
- 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
- Additional assignments can be used for more hands-on practice

Day-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 iterative pattern pseudo code to test all the logical paths being covered

Assignment 7: Exercises for Self Review:

- 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.)

Day-2 Assignments

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.

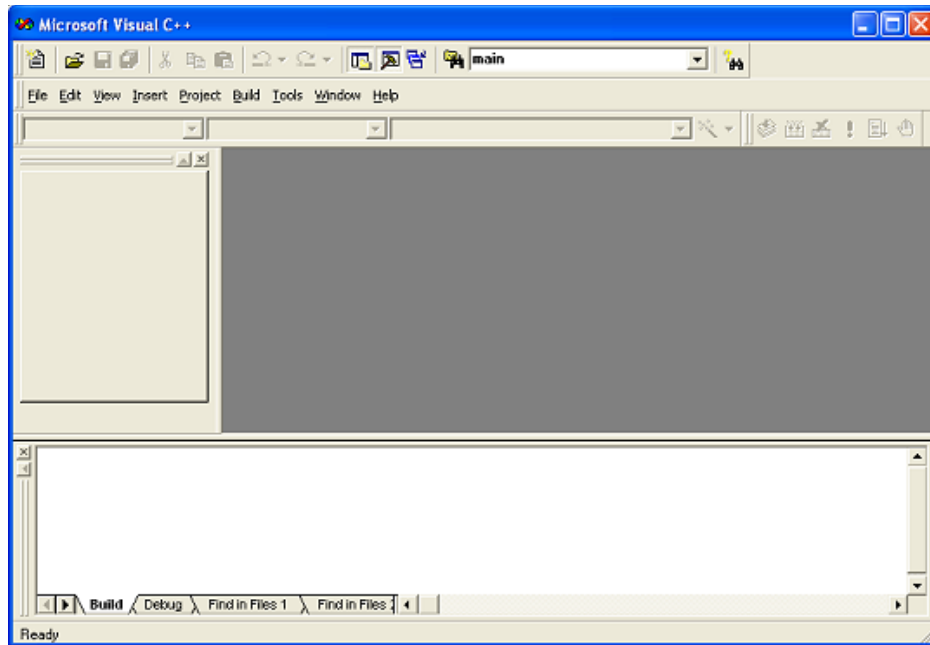
Background: The Microsoft Visual C++ version 6.0 includes a rich variety of professional tools to help write code in C, C++, and many other technologies such as MFC, OLE, ODBC, DAO, ActiveX, and COM. The IDE (Integrated Development Environment) is made up of several windows that help in editing, compiling, executing and debugging the programs. In this module we will be working only with C programs.

Problem Description:

Estimated time: 20 minutes

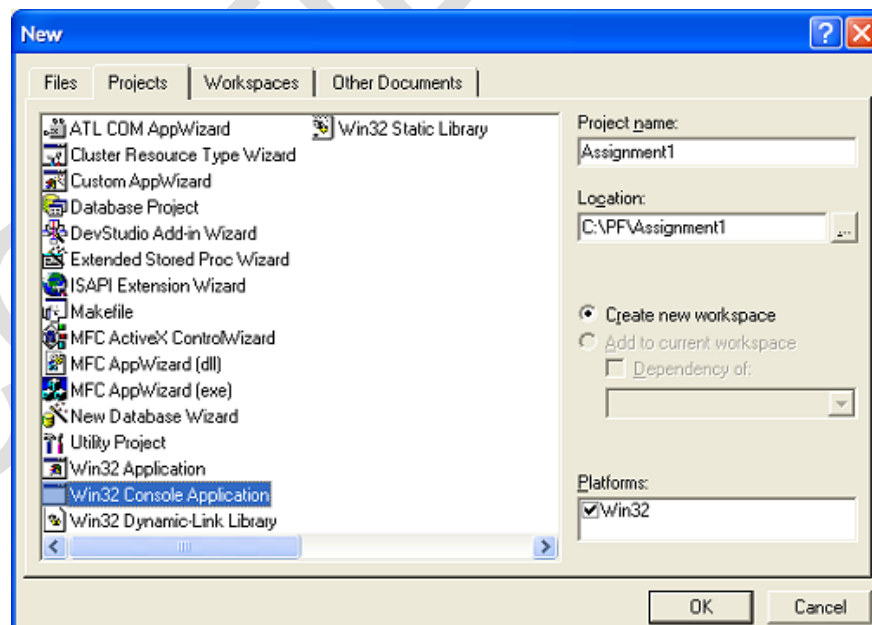
Step 1: Click on start -> Programs -> Microsoft Visual Studio 6.0 -> Microsoft Visual C++ 6.0

Step 2: The Microsoft VC++ IDE opens.



Step 3: To create the first C program, you have to first create a project workspace. A workspace is a container for all the files that make up a project.

Click on the menu option File -> New. You get the following screen:



In this window, Projects tab would be selected by default, if not click on the Projects tab. Select the project type as “Win32 Console Application”.



Note: A console application is one that interacts with a text screen. Console applications are one-window, text-based programs.

The project workspace you create will be stored as a folder in the hard disk. Provide the path, where the project should be stored in the “Location field”.

Example: C:\PF

Type the name of the project in the “Project Name” field.

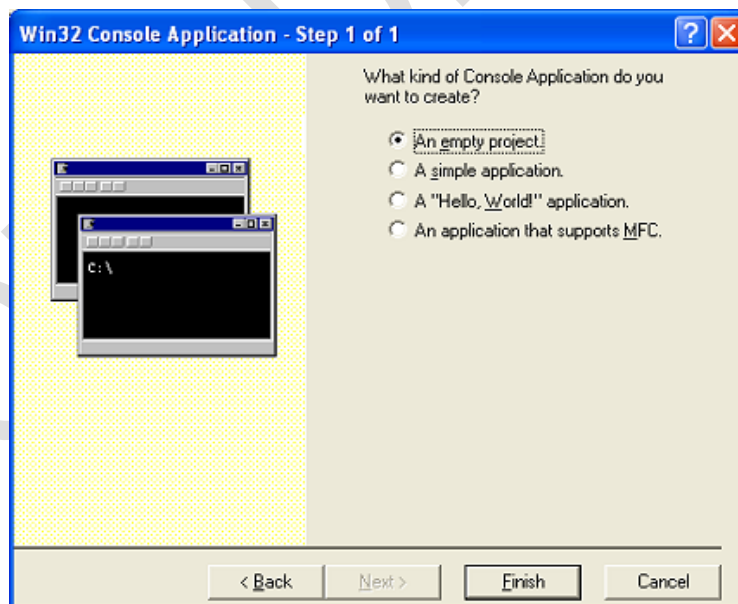
Example: Assignment1



Note: The name of the project that you had typed in the Project Name field automatically appends to the path specified in the “Location” field.

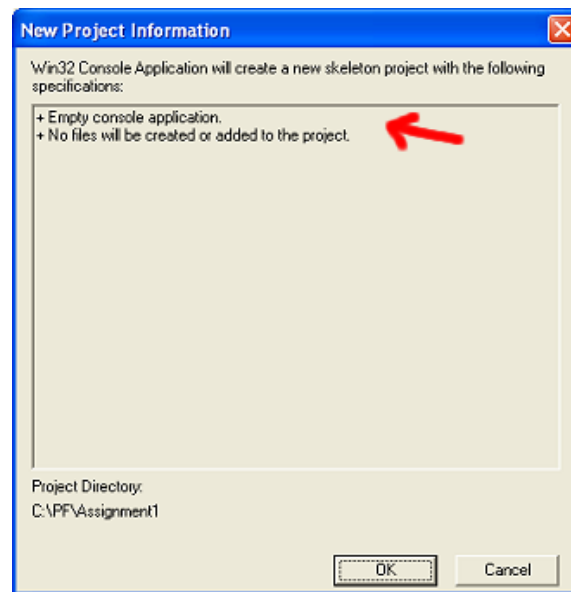
Click on OK.

Step 4: You get the following screen wherein you can select the kind of application to create.



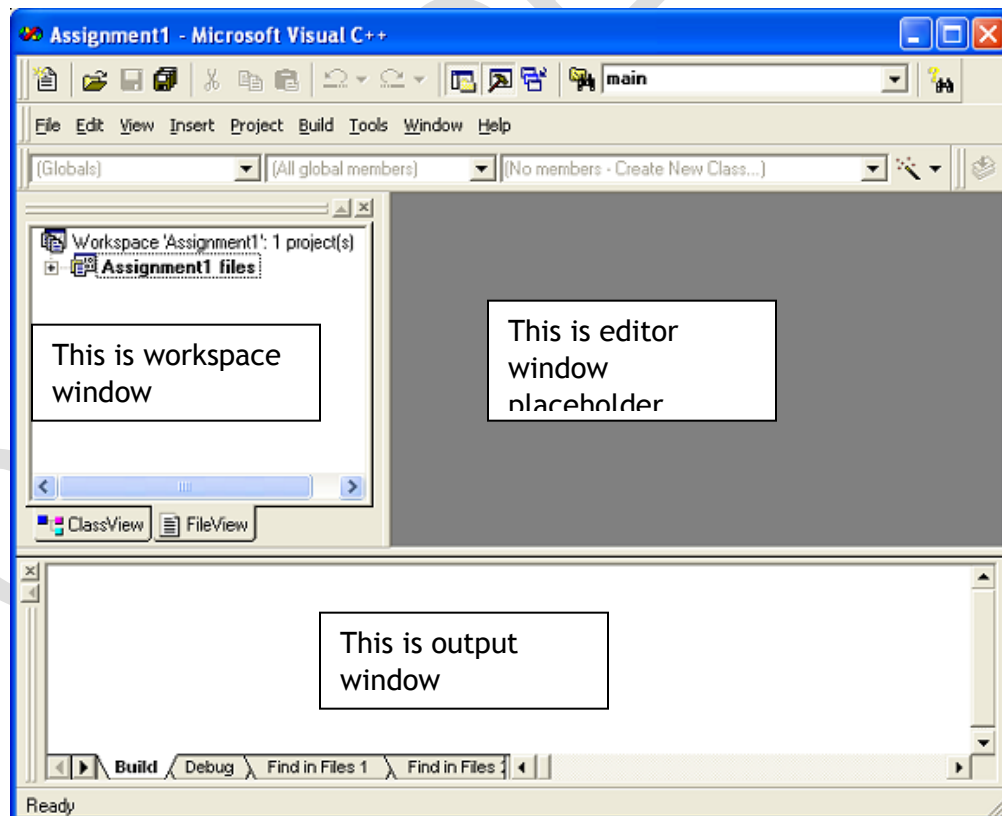
Since we are creating a new application, select “An empty project” option and then click on the Finish button.

Step 5: The following screen appears that gives the summary of the project workspace you have created.



Read the summary provided by the Visual C++ IDE.

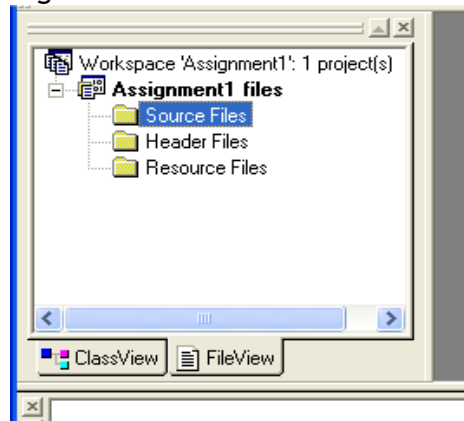
Step 6: Click on OK and you see the following screen.



The project workspace has been created now. Check to make sure that a folder by name Assignment1 has been created.

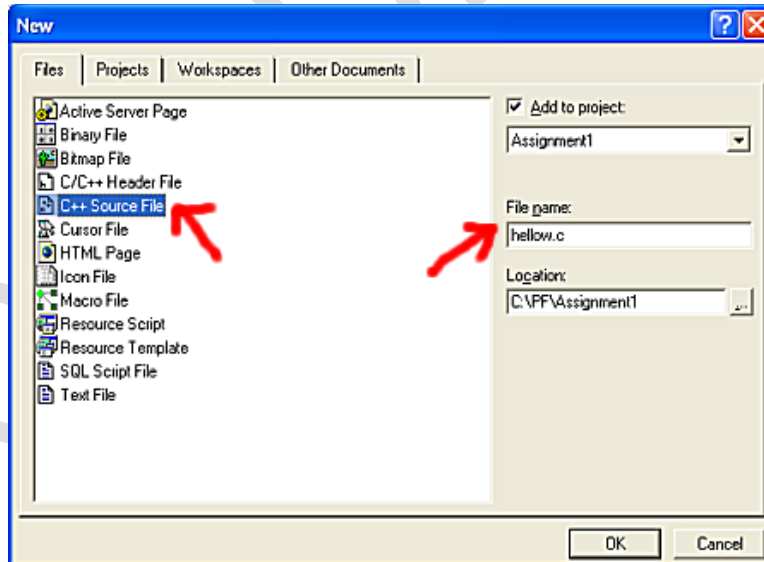
Observe that inside this folder Assignment1 many other supporting files will be created. A folder by the name Debug will also be created with some files in it. The executables created, after compilation and linking will be stored in this Debug folder.

Step 7: Once the workspace has been created, the source code has to be written. For this a .c file has to be created. Click on the “File View” in the workspace window. Click on the [+] symbol you see before the “Assignment1 files” node.



Click on the Source Files folder that you see in blue in the above screen.

Step 8: Click on the menu option File -> New. You see the following screen.



Click on the “Files” tab and select the file type, C++ Source File. Type the file name as hellow.c in the File name field. Click on OK button.

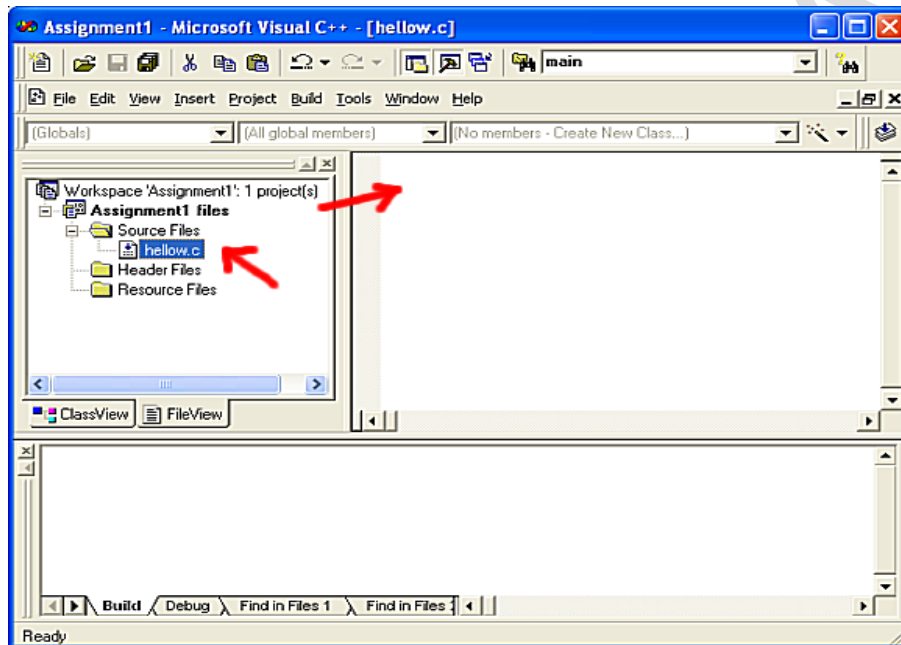


Note 1: By default Visual Studio is designed to work with C++. However, C++ is backward compatible with C, therefore we can create a C file.

Note 2: If the extension for the file is not specified, the default extension .cpp will be given. Since we are working on C programs, do not forget to give the extension as .c

Click on the OK button.

Step 9: You will see the following screen and the editor appears wherein you can type your C source code.



Step 10: Type the following c program.

```

/*****
* Filename: hellow.c
* Description: First C Program. Prints "Hello world"
* Author:    E&R Department, Infosys Technologies Ltd.
* Date:      18-Jan-2005
*****/

#include <stdio.h>

/*****
* Function: main()
* Description: Displays the caption "Hello World". This is
*              the entry point of any C program
*****/
```



```

* Input Parameters:
*   int argc - Number of command line arguments
*   char **argv - The command line arguments passed
* Returns: 0 on success to the operating system
*****/

int main (int argc, char **argv) {
    /* Display the caption on the screen */
    printf ("Hello World\n");

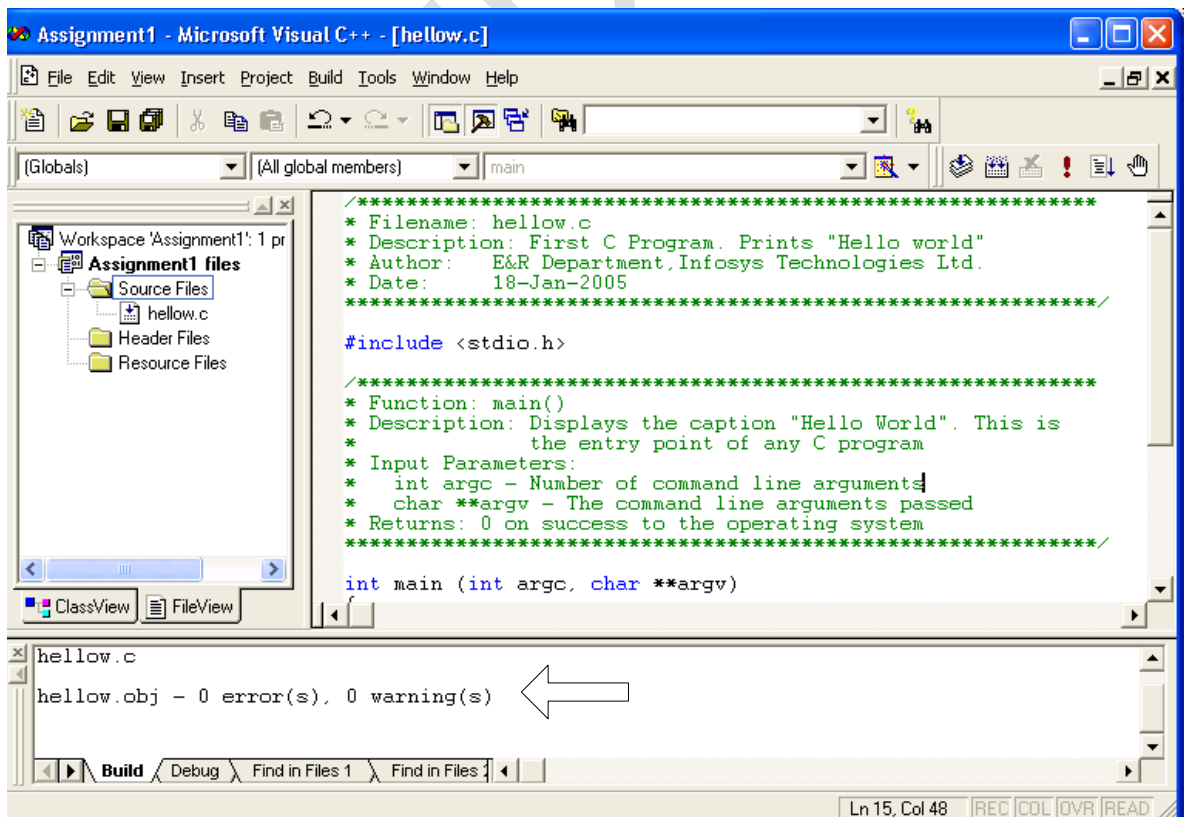
    /* Return a success code to the Operating System */
    return 0;
}

/*****
* End of hellow.c
*****/

```

Step 11: Save the program by clicking on the menu option File->Save. Click on the menu option Build -> Compile hellow.c.

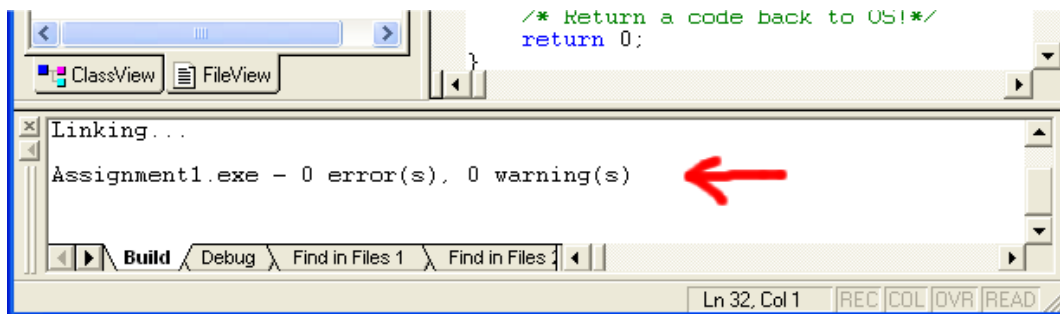
The program starts to compile. The compilation summary messages appear in the output window. If there are no errors in the source code, you can see the screen as below.



Observe the message in the output window.

Step 12: Click on the menu option Build -> Assignment1.exe.

The linking phase starts and if there are no linking errors, you will get the message “0 errors, 0 warnings” as shown below:

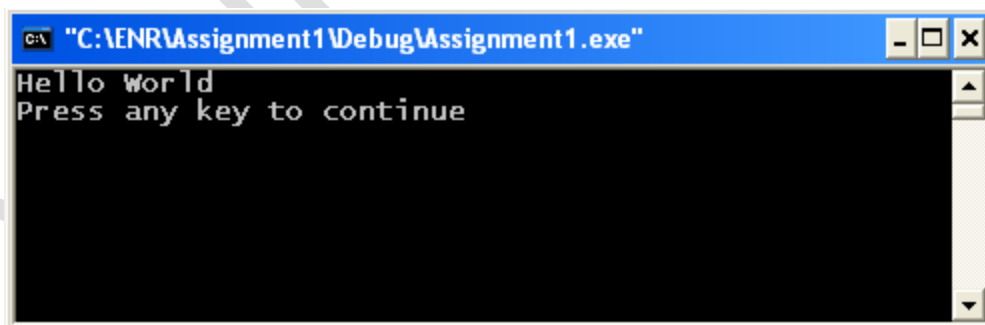


Read the messages by the linker in the output window. Also look under the C:\PF\Assignment1\Debug folder. The executable file would have been created.



Note: All the programs should compile with **zero errors and warnings**. If there are warnings, the code has to be fixed before executing the program.

Step 13: To execute the program, click on the menu option Build -> Assignment1.exe option. Your program executes now and you see the following output:



Step 14: To close the project workspace, use the menu option File -> Close Workspace.



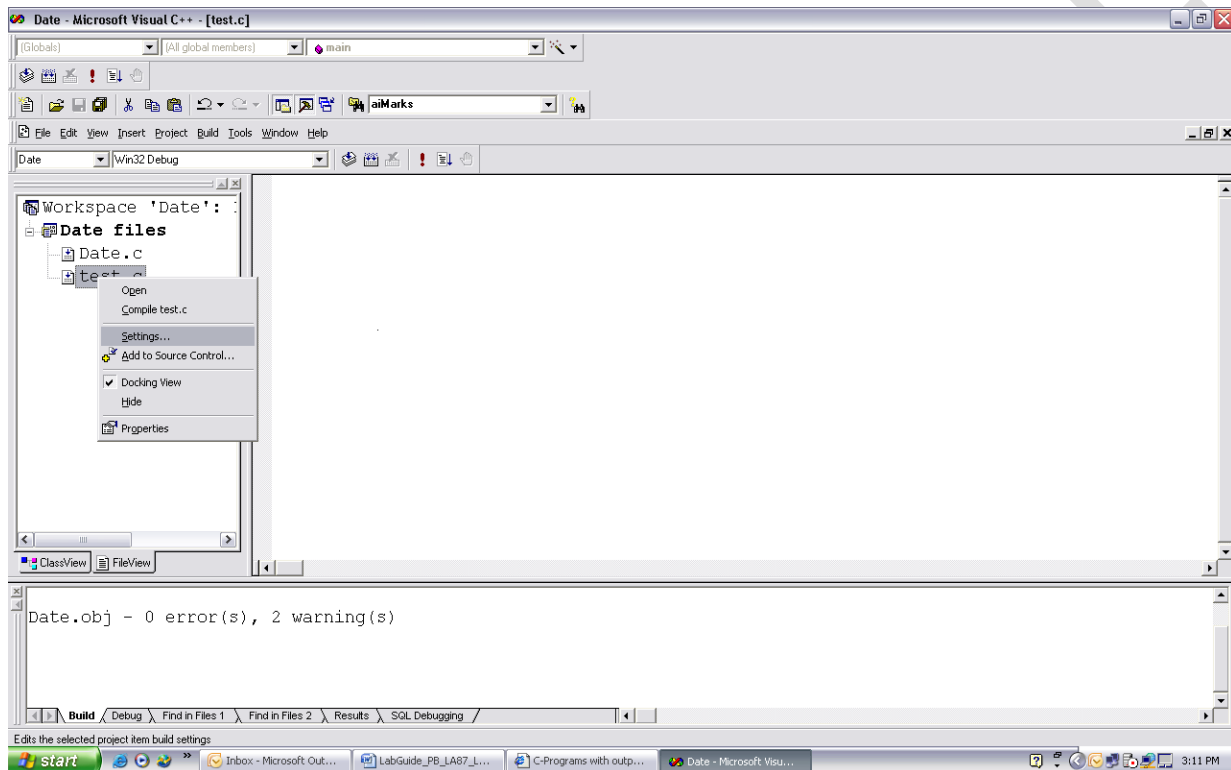
Note: The best practice would be to create one workspace for each day and include all the assignment files for that day in the same workspace.

To add a new file to a workspace, go to File->New->C++ source file.

Whenever a particular file (.c file) needs to be compiled and executed, exclude all the remaining files in the workspace from build by right clicking on the file, choose settings option in that general tab and then click on exclude file.

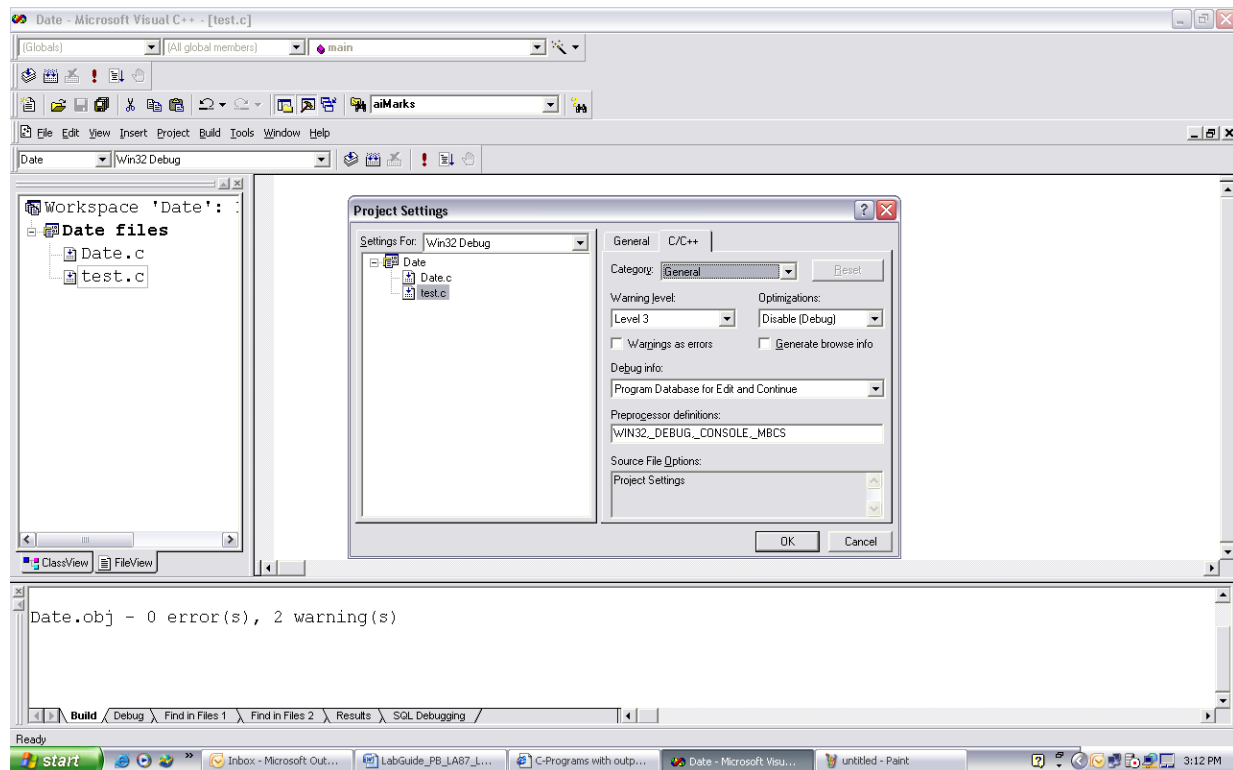
Follow the below steps to exclude a file from build.

Step a: Right Click on the .c file to be excluded and choose Settings option

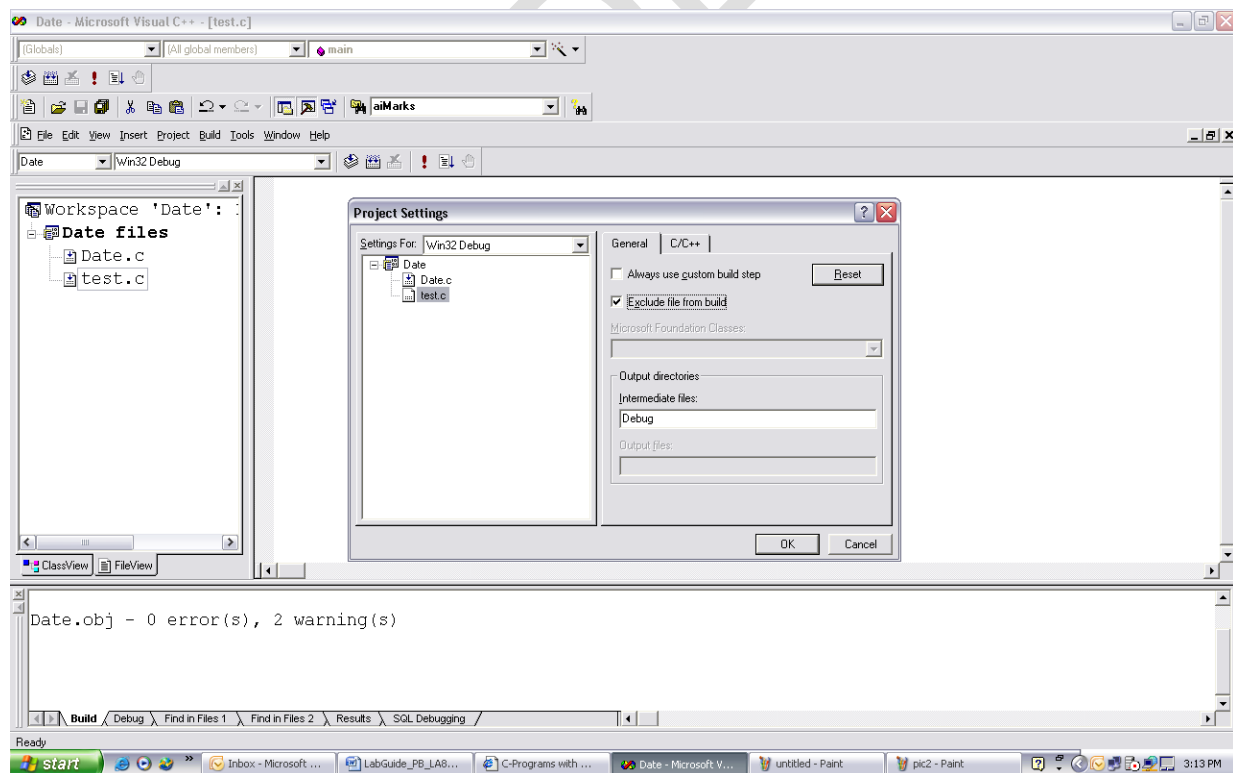


Step b: Choose General tab in Settings option

CONFIDENTIAL



Step c: Click on Exclude from build check box and then click Ok to apply the settings



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 9: 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.

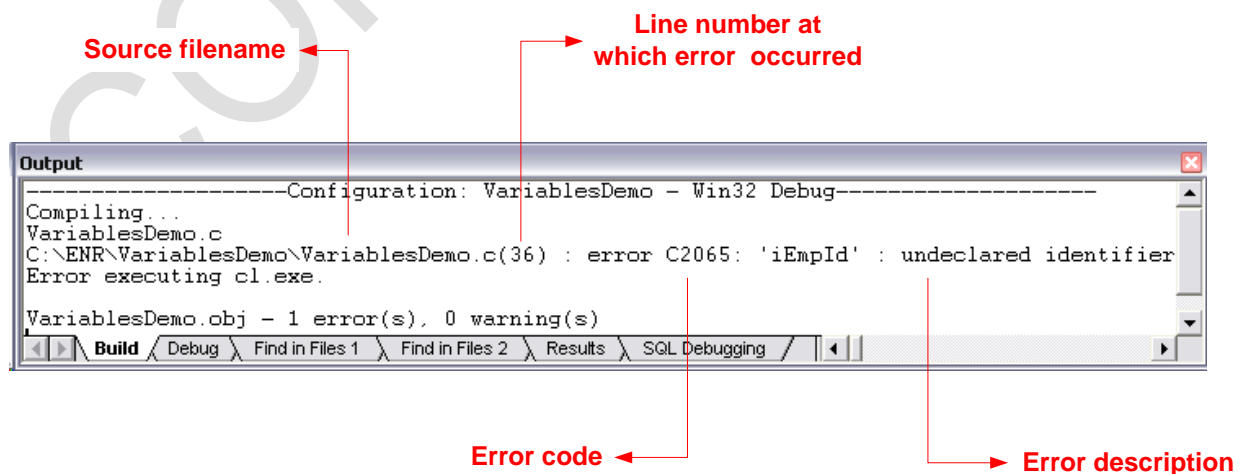
Problem Description: Declare variables to store employee id, basic salary and allowances. Display all the details. Also display employee id in hexadecimal format.

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 'VariablesDemo' project with the name **VariablesDemo.c**. Now copy the source code, given in VariablesDemo.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:



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:

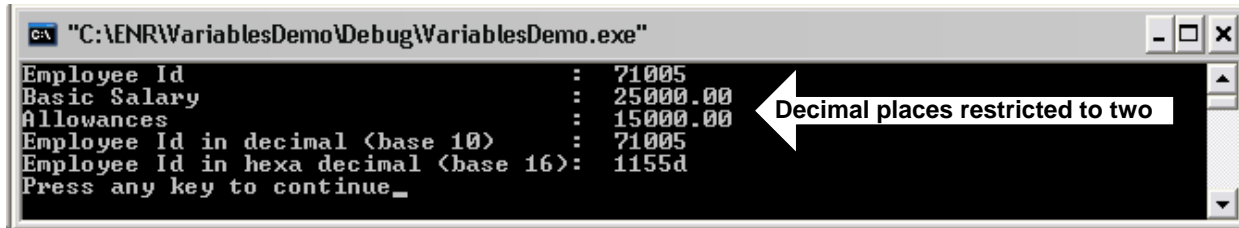
```
"C:\ENR\VariablesDemo\Debug\VariablesDemo.exe"
Employee Id           : 71005
Basic Salary          : 25000.000000
Allowances            : 15000.000000
Employee Id in decimal (base 10) : 71005
Employee Id in hexa decimal (base 16): 1155d
Press any key to continue_
```

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:



```
C:\ENR\VariablesDemo\Debug\VariablesDemo.exe
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_
```

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 10: 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 11: 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:

Include a job band for the employee. Job band can be 'A' or 'B' or 'C' or 'D' or 'E'.

Estimated time: 20 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 12: 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 13: Coding standards

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

Background: Infosys follows coding standards for the programs written in order to improve the 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 14: 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 15: 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'

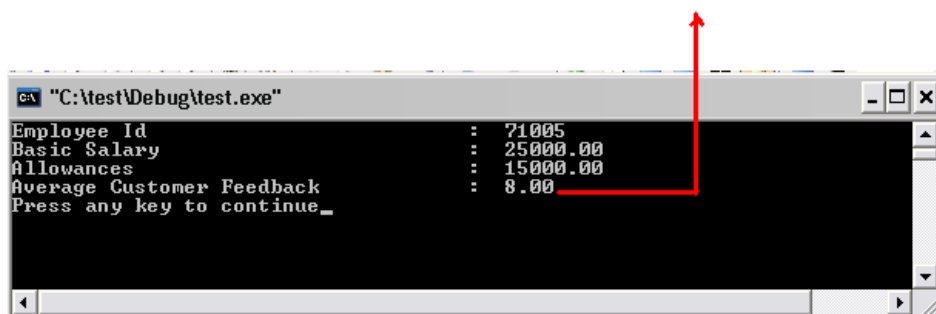
Step 2: Add the source code, `AverageCustomerFeedback.c` provided into the project

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:

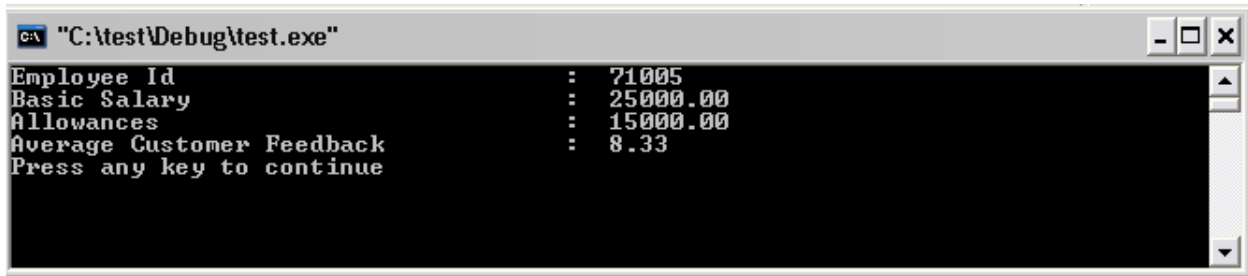
The average feedback displays only the integer part



```
"C:\test\Debug\test.exe"
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:



```
C:\Test\Debug\test.exe
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 16: 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 above assignment 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: Use `SalaryComputation.c` provided in the supplied source code folder for this assignment

Summary of this assignment:

In this assignment, you have learnt:

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

Assignment 17: 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 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 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.

compileerrors.exe - 26 error(s), 0 warning(s)
Build Debug Find in Files 1 Find in Files

```

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 identifier
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.
Build Debug Find in Files Find in Files

```



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.

```

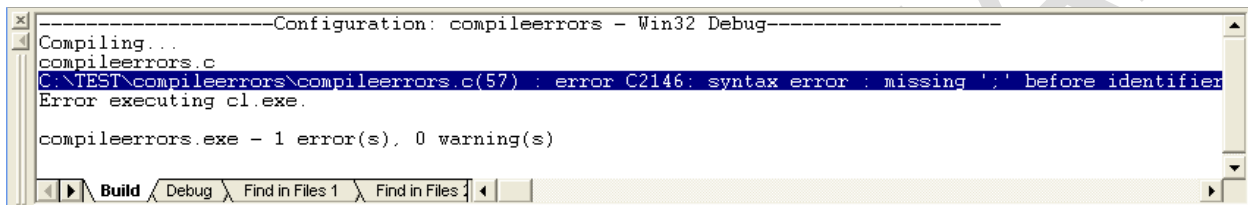
/* For any product costing more than INR 500, Discount = 25% */
if (dPrice > 500.0)
    dDiscount = 25.0;
}
else {

```

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!



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.

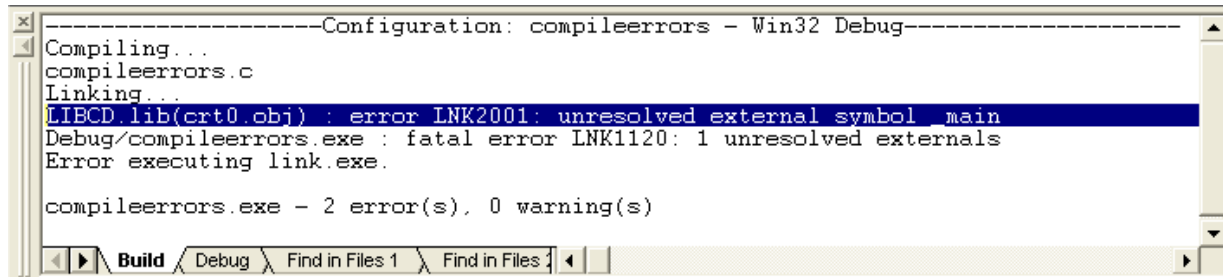
Include semicolon in line 56.

```
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”.



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`.

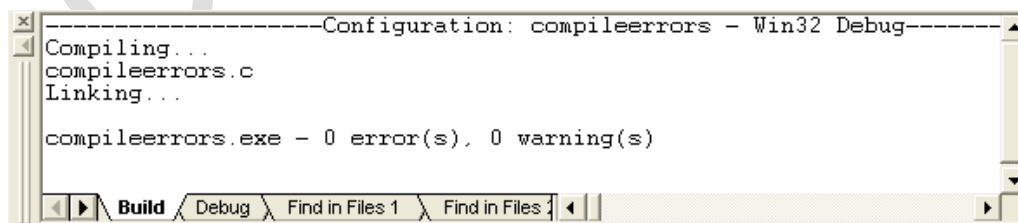
Step 13: Go to the declaration of function 'main'.

```
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.

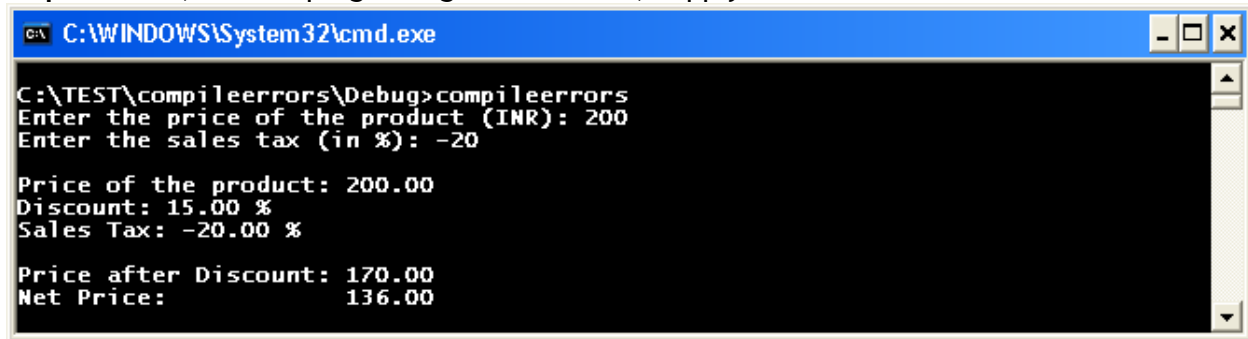
```
int main (int argc, char** argv)
```

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



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.

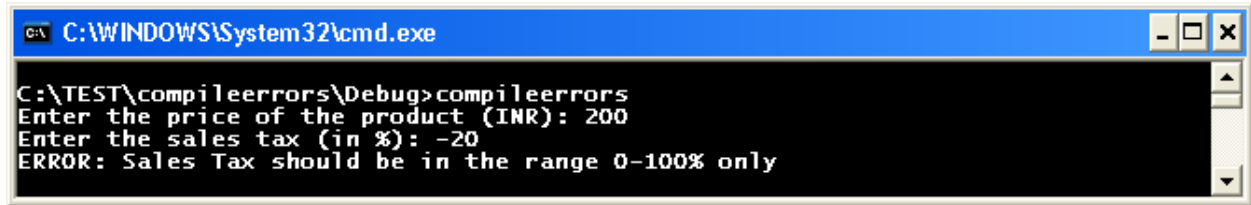
```
/* Read the Sales Tax in percent */
printf ("Enter the sales tax (in %%): ");
scanf ("%lf", &dSalesTax);
fflush (stdin);
```

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:\WINDOWS\System32\cmd.exe
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 18: 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 19: Understanding the working of while loop

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 20: Understanding the working of for loop

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 21: Understanding the working of do while loop

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 22: Exercises for Self Review

1. 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;
}
```

2. Debug the following code snippet:

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

3. Find the output of the following program:

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

4. Debug the following program and fix the error:

```
/* *****
 * Filename   : Operators.c
 * Author    : Education & Research Dept, Infosys Technologies Limited
 * Date      : 04-Apr-2008
 * 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
*****/

```

5. 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;
}

```

6. 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;
}

```

```
}
```

7. 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;  
}
```

8. 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;  
}
```

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 ilIndex = 1;
    for(;;){
        if(ilIndex > 20){
            break;
        }
        else {
            printf("%d ",ilIndex);
            ilIndex = ilIndex * 2;
        }
    }
    return 0;
}
```

Day-3 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: 20 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).



```
C:\ENR\EmployeeArray\Debug\EmployeeArray.exe
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
```

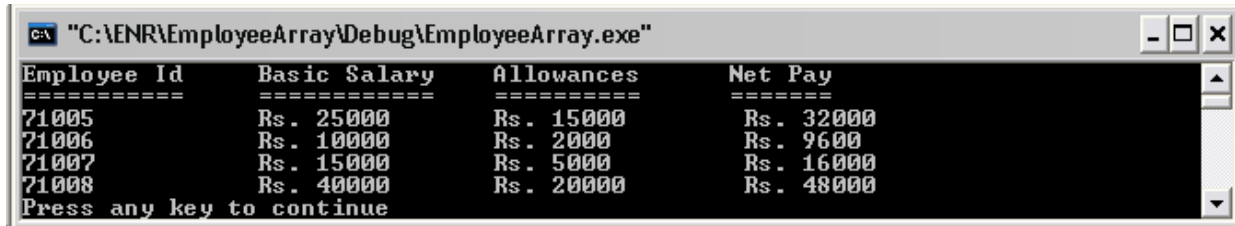
Note: Use tab (\t) and new line constant (\n) for getting the output in the above format

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).



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

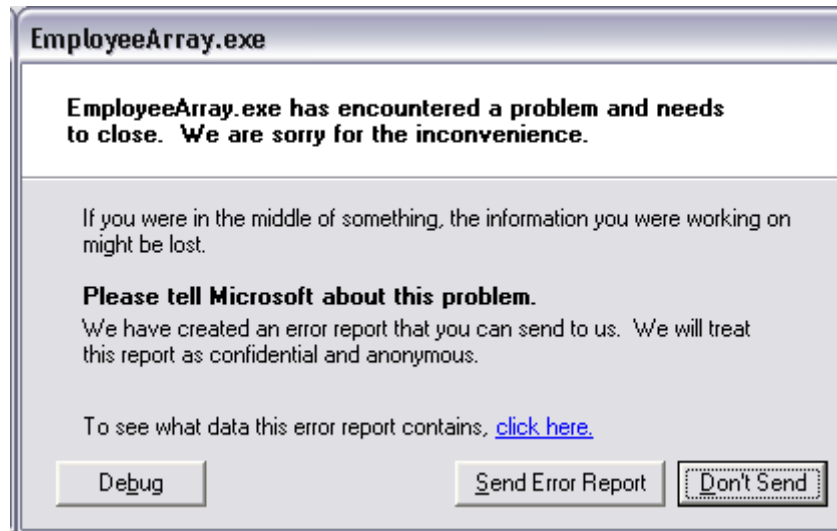
Estimated time: 15 minutes

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:



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 '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 26: 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: 20 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: 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 Job Bands and Confirmation status used in earlier assignments can be reused

Summary of this assignment:

In this assignment, you have learnt

- Design of menus
- Data validation

Assignment 27: 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:

	0,0	0,1	0,2	0,3
Row 0	71005	7	8	10
Row 1	71005	5	5	7
Empld	1,0	1,1	1,2	1,3
Customer Feedback				

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'

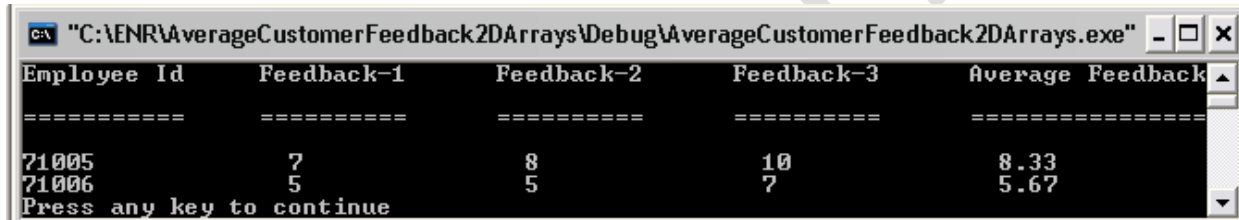
Step 2: Add the source code, **AverageCustomerFeedback2DArrays.c** provided into the project

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:



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 28: 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 29: 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

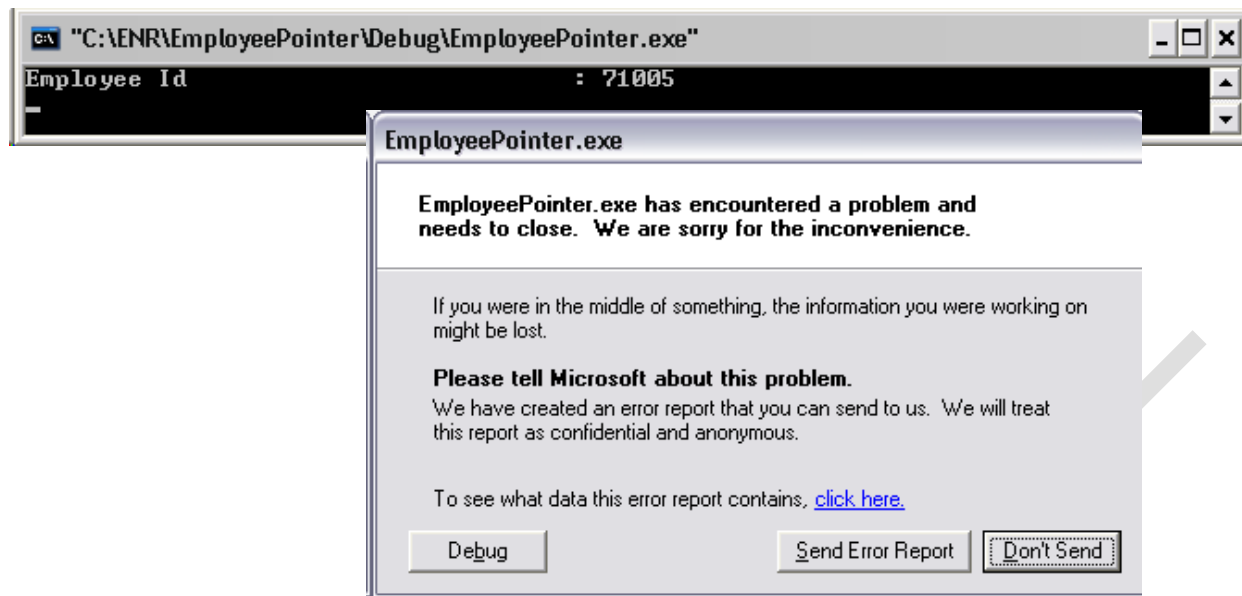
Step 1: Create an empty project in Visual C++ (Console Application) with name 'PointerDebug'

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

Step 3: Compile the program

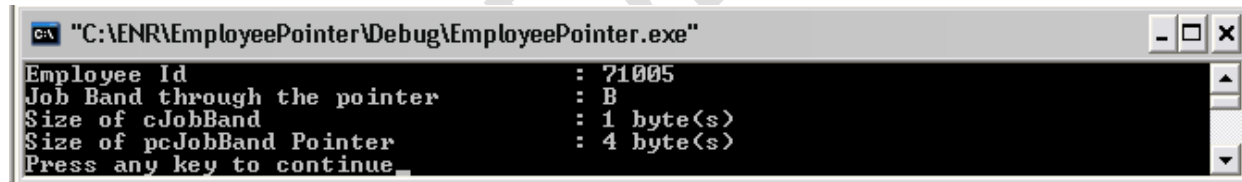
Step 4: Ignore the warnings and execute the program.

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



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:



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 30: 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 31: 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 32: 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 33: 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 34: 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 35: 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

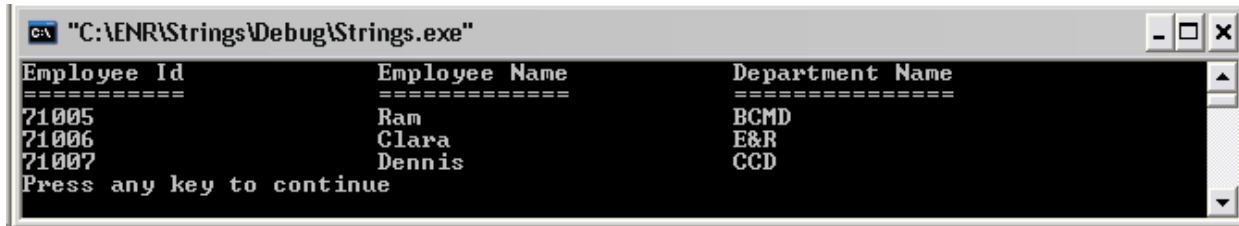
Step 1: Create an empty project in Visual C++ (Console Application) with name 'ArrayofStrings'

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:



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 36: Exercises for Self Review

1. 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;
}
```

2. 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;
    }
}
```

3. 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;
}
```

4. Find the output of the following program:

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

5. Debug the following code snippet:

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

6. 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;
}
```

7. 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;
    }
```

Day-4 Assignments

Assignment 37: Developing functions

Objective: Learn and appreciate modular programming.

Problem Description: Write a program to accept employee id, job band and department code and to validate the job band and department code using functions. The valid job bands are 'A', 'B', 'C' and 'D'. The department code must be in the range 110 - 125.

Estimated time: 30 minutes

Step 1: Consider the following program:

```
/* *****
 * Filename: EmployeeJobBandDeptCodeValidation.c
 * Description: To understand modular programming and designing
 *              functions.
 * Author:     E&R Department, Infosys Technologies Ltd.
 * Date:      16-Mar-2007
 * ***** */

#include <stdio.h>
#include <stdlib.h>

#define VALID 1
#define INVALID 0

/* Declare the prototype of the functions */
int fnValidateJobBand(char cJobBandTemp);
int fnValidateDeptCode(int iDeptCodeTemp);

/* *****
 * Function: main()
 * Description: To accept details of an employee and to validate
 *              the job band and the department code
 *              using a function.
 * Input Parameters:
 *   int argc - Number of command line arguments
 * ***** */
```

```
*   char **argv   The command line arguments passed
* Returns: 0 on success to the operating system
*****/

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

    /* Declare the variables */
    int iEmpId;
    char cJobBand;
    int iDeptCode;
    int iRet;

    /* Accept the employee details from the user*/
    printf("Enter the employee id ");
    scanf("%d",&iEmpId);
    printf("Enter the job band ");
    scanf("%c",&cJobBand);
    printf("Enter the department code ");
    scanf("%d",&iDeptCode);

    /* Print the employee id */
    printf("Employee Id:%d ",iEmpId);

    /* Call the function to validate the job band */
    iRet = fnValidateJobBand(cJobBand);

    /* Display the message based on the return value */
    if (iRet == VALID)
    {
        printf("%c is a valid job band\n", cJobBand);
    }
    else
    {
        printf("%c is an invalid job band\n",cJobBand);
    }

    /* Call the function to validate the deptcode */
    /* Replace this comment and write the code here to invoke
       the function to validate the department code */

    /* Display the message based on the return value */
    /* Replace this comment and write the code here to display
       valid or invalid message */

    /* Return a success code to the Operating System */
```



```
        return 0;
    }

/*****
* Function: fnValidateJobBand()
* Description: To accept the job band of an employee
*              and to validate the job band. The valid job bands are
*              'A', 'B', 'C' and 'D'
* Input Parameters:
*   char cJobBandTemp - The job band of an employee
* Returns: VALID if the job band is valid and INVALID if the job band
*          is invalid
*****/

int fnValidateJobBand(char cJobBandTemp)
{
    /* Validate the job band */
    switch (cJobBandTemp)
    {
        case 'A':
        case 'a': return VALID;

        case 'B':
        case 'b': return VALID;

        case 'C':
        case 'c': return VALID;

        case 'D':
        case 'd': return VALID;

        default : return INVALID;
    }
}

/*****
* Function: fnValidateDeptCode()
* Description: To accept the dept code of an employee
*              and to validate the deptcode. The deptcode is in the
*              range 110 to 125.
* Input Parameters:
*   int iDeptCodeTemp - The department code of an employee
* Returns: VALID if the dept code is valid and INVALID if the
*          dept code is invalid
*****/
```

```
int fnValidateDeptCode(int iDeptCodeTemp)
{

    /* Validate the department code */
    /* Replace this comment and write the code here to validate
       the department code */

}

/*****
* End of EmployeeJobBandDeptCodeValidation.c
*****/
```

Summary of this assignment:

In this assignment, you have learnt

- How to write function prototypes
- How to write function header and function body
- How to call user-defined functions

Assignment 38: Returning Values

Objective: To understand how to update the value of a variable using a function

Problem Description: Write a program to accept the employee id and the basic salary. The basic salary should be increased by 20% using a function. Display the employee id and the updated basic salary in the function 'main'.

Estimated time: 20 minutes

Assignment 39: Pass by Value

Objective: To understand the limitations of pass by value mechanism

Problem Description: Write a program to accept the employee id job band and the basic salary (In Indian Rupees) of an employee. Write a function called fnEmployeeJobBandChange to change the job band and the basic salary of employees. Refer to the following table:

Existing Job Band	New Job Band	Basic Salary hike
A	B	50%
B	C	40%
C	D	30%
D	D (No change in the band)	20%

Validations: The job band should be 'A' or 'B' or 'C' or 'D'. The basic salary should be more than Rs. 4000

Estimated time: 20 minutes

Step 1: Use `Pass_By_Value.c` file containing the partial code available in the supplied source code folder

Step 2: Write the necessary code in above given program and compile it.

Step 3: After successful compilation of the program, execute it. Supply the necessary values and observe the output.

Step 4: There is no change in the Job band and the basic salary. This is because the values are passed using “pass by value”.



Note: A function can return only one value using the ‘return’ statement. Since “pass by value” copies the values of variables from one memory location to another memory location, the changes are not only to the copy of the variables. The changes are not reflected back into the original contents of the variables.

Summary of this assignment:

In this assignment, you have learnt

- Pass by value
- A function cannot return multiple values using the ‘return’ statement

Assignment 40: Pass by Reference

Objective: To understand pass by reference.

Problem Description:

Modify the program written in “Assignment 39” Pass by Value” which implements the functionality of function “fnEmployeeJobBandChange” using “pass by reference”.

The prototype of “fnEmployeeJobBandChange” is given below:

```
void fnEmployeeJobBandChange(char *pcJobBandTemp,  
                             double *pdBasicSalaryTemp);
```

The function call from main() will be as given below:

```
fnEmployeeJobBandChange (&cJobBand, &dBasicSalary);
```

Estimated time: 20 minutes

Assignment 41: Passing 1-D Arrays as Arguments to Functions

Objective: Passing 1-D arrays to functions

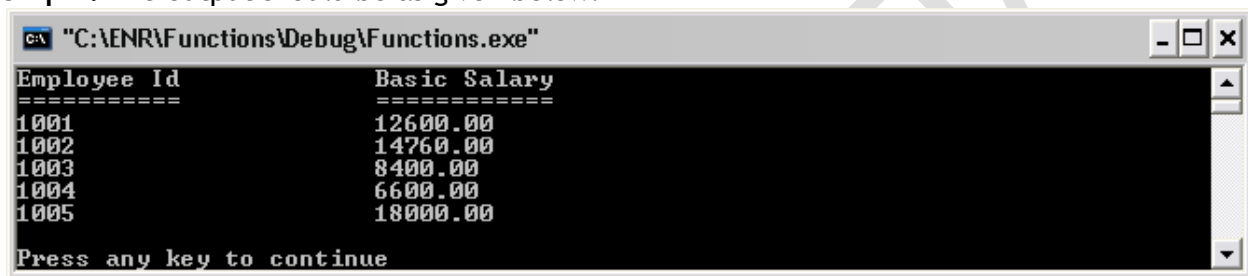
Problem Description: Write a program to increase the basic salary by 20% using a function.

Estimated time: 20 minutes

Step 1: Use `1D_Array_As_Arguments.c` file containing the partial program code available in the supplied source code folder

Step 2: Write the necessary code in the above given program and compile it.

Step 3: The output should be as given below:



Employee Id	Basic Salary
1001	12600.00
1002	14760.00
1003	8400.00
1004	6600.00
1005	18000.00

Press any key to continue



Note: When an entire 1-D array is passed as an argument to a function, the array is passed by reference. So the changes that are done to the array in the functions alter the values in the original array.

Summary of this assignment:

In this assignment, you have learnt

- How to pass 1-D arrays as arguments to functions

Assignment 42: Passing 2-D Arrays as Arguments to Functions

Objective: Passing 2-D arrays to functions

Problem Description:

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

Estimated time: 20 minutes

Step 1: Use `2D_Array_As_Arguments.c` file containing the partial program code available in the supplied source code folder

Step 2: Write the necessary code in the above given program and compile it.

Step 3: Compile and execute the program. The output should be as given below:

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: When an entire 2-D array is passed as an argument to a function, the function header should declare the parameter with the column size specified. The row size could be ignored. There will be a compilation error if the column size is ignored.

Summary of this assignment:

In this assignment, you have learnt

- How to pass 2-D arrays as arguments to functions

Assignment 43: Automation of Credit Point calculation for a Mobile company

Objective: To understand the use of functions.

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

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 customer (for 5 customers) and initialize it with the following customer names: “Lara”, “Lin”, “Lara”, “Krishna”, “Sophia”

Declare an array to store the points scored 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 114 and so on.

Implement the following functionalities by writing functions:

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.

Estimated time: 30 minutes

Assignment 44: Recursive function

Objective: To understand the concept of Recursion

Problem Description: Write a program using recursion to check whether the given number is prime or not

Estimated time: 15 minutes

Summary of this assignment:

In this assignment, you have learnt:

- Implementation of the concept of recursion

Assignment 45: Creating structures

Objective: Creating and using structures.

Problem Description: Write a program to create a structure to store the employee id, name and basic salary. Display the employee details.

Estimated time: 25 minutes

Step 1: Type the following program:

```
/* *****  
* Filename: EmpDetails.c  
* Description: To understand the usage of structures  
* Author: E&R Department, Infosys Technologies Ltd.  
* Date: 16-Mar-2007  
* ***** */  
  
#include <stdio.h>  
#include <string.h>  
  
/* Declare the structure to store employee details */  
struct _employee  
{  
    int iEmpId;  
    char acEmpName[20];  
    double dBasicSalary;  
};
```

```
/* Define a new data type named "Employee" */
typedef struct _employee employee;

/*****
* Function: main()
* Description: To initialize the employee details and to display them.
* Input Parameters:
*   int argc - Number of command line arguments
*   char **argv The command line arguments passed
* Returns: 0 on success to the operating system
*****/

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

    /* Declare the structure variable */
    employee sEmp;

    /* Initialize the members of the structure */
    sEmp.iEmpId = 1001;
    strcpy(sEmp.acEmpName, "Lara");
    sEmp.dBasicSalary = 7500.00;

    /* Print the employee details */
    printf("Employee Id :%d\n", sEmp.iEmpId);
    printf("EmpName      :%s\n", sEmp.acEmpName);
    printf("Basic Salary: %.2lf\n", sEmp.dBasicSalary);

    /* Return a success code to the Operating System */
    return 0;
}

/*****
* End of EmpDetails.c
*****/
```

Step 2: Compile and execute the program.

Summary of this assignment:

In this assignment, you have learnt:

- The declaration of structures
- Accessing structure members

Assignment 46: Employee name validation

Objective: Passing individual structure members as arguments to functions.

Problem Description: Write a program to accept the details of an employee and to validate the employee name. The allowed characters in an employee name are: lower case and upper case alphabets, blank spaces, dot (.) and apostrophe (').

Estimated time: 20 minutes

Step 1: Use `Employee_Name_Validation.c` file containing the partial program code available in the supplied source code folder

Step 2: Write the necessary code in the above given program and compile it

Step 3: Compile and execute the program

Summary of this assignment:

In this assignment, you have learnt:

- Passing individual structure members as arguments to functions
- Validation of name

Assignment 47: Passing Structures to functions using pass by Value

Objective: Passing the whole structure as arguments to functions using pass by value.

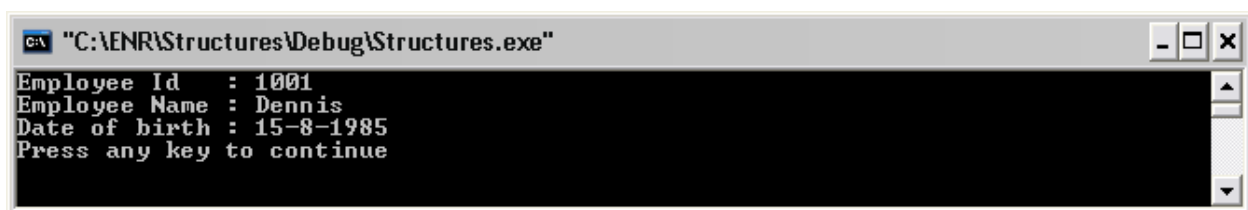
Problem Description: Write a program to store the details of an employee (employee id, name, date of birth and basic salary) into a structure. Use a function to increase the basic salary by 20%.

Estimated time: 30 minutes

Step 1: Use `Structures_Pass_By_Value.c` file containing the partial program code available in the supplied source code folder

Step 2: Write the necessary code in the above given program and compile it

Step 3: Compile and execute the program. The output should be as given below:



```
C:\ENR\Structures\Debug\Structures.exe
Employee Id : 1001
Employee Name : Dennis
Date of birth : 15-8-1985
Press any key to continue
```

Summary of this assignment:

In this assignment, you have learnt:

- Passing whole structure as arguments to functions
- Returning structure variables from functions

Assignment 48: Passing Structures to functions using Pass by Reference

Objective: Passing the whole structure as arguments to functions.

Problem Description: Write a program to store the details of an employee (employee id, name, date of birth, basic salary and allowances) into a structure. Use a function to compute the allowances as 40% of the basic salary. Display all the details and the gross pay.

Estimated time: 30 minutes

Step 1: Use `Structures_Pass_By_Reference.c` file containing the partial program code available in the supplied source code folder

Step 2: Write the necessary code in the above given program and compile it

Step 3: Compile and execute the program

Summary of this assignment:

In this assignment, you have learnt:

- Passing structures using pass by reference

Assignment 49: Implementing the array of structures - Part I

Objective: Including multiple source files in a workspace, complete the coding of functions, compile and execute as a whole project.

Background: The partial code for a Retail Application has been developed. The implementation of `fnCheckSupplierCode` function is not done. The source files need to be integrated in one workspace, compiled and executed.

Problem Description:

The Retail Application involves maintenance of items. For this purpose the partial source code has been developed. Following are the given files:

- `RetailAppln.c`
- `ItemMaint.c`
- `Initialization.c`
- `Structures.h`

These files need to be integrated into one application, compiled and executed. These files are present in the within **Supplied_Source_Code** folder. The details are as follows:

RetailAppln.c

The menu driven program to maintain the information about items such as item code, item name and item price has been coded in **RetailAppln.c**. The menu has the following options:

1. Add item
2. Update item price
3. Item Report
4. Exit

ItemMaint.c

Read the specifications for the functionalities which have been implemented in **ItemMaint.c**.

Add Item Module: The item code should be automatically generated starting from 1000 and the next item added will get the code as 1001 and so on. The program should accept the item name, price and the supplier code from the user. After the following validations, the details should be stored into array of **structure item**

Validations in Add Item Module:

Item Name:

- Write the code to validate whether entered Item name has only alphabets, digits or blank spaces. If not, display an appropriate error message.
- Code is provided to validate whether the entered item name is not already present in the array of structure, **item**(No duplications)

Item Price:

- Write the code to validate whether entered item price is greater than zero. If not, display an appropriate error message

Supplier Code:

- Write the code in function **fnCheckSupplierCode** in file **ItemMaint.c** to validate whether entered supplier code is one of the codes that exists in the array of structure, **supplier**. If not, display an appropriate error message

Update Item Price: The program should accept the item code and the price. After validations, the corresponding item in the array should be updated.

Validations in Update Item Price Module:

- Code is provided to validate whether entered Item code is an existing item code, present in array of structure **item**.
- Write the code to validate whether entered item price is greater than zero. If not, display an appropriate error message.

Item Report: Code is provided to display the details of the items as given below:

Item Code	Item Name	Price	Supplier Code
=====	=====	=====	=====

You need to integrate the given files into one application using the steps mentioned below and write the required code in `ItemMaint.c`

Step 1: Create a project named “RetailAppIn”.

Step 2: Copy all the supplied files into the project folder created in step 1.

Step 3: Right click on “Header Files” and Select “Add Files to Folder” option. Add the supplied file “Structures.h”.

Step 4: Click on “FileView” option and right click on “Source Files” and select "Add Files to Folder ..." option. Add the supplied file "Initialization.c"

Step 5: Click on “FileView” option and right click on “Source Files” and select "Add Files to Folder ..." option. Add the supplied file "RetailAppIn.c"

Step 6: Click on “FileView” option and right click on “Source Files” and select "Add Files to Folder ..." option. Add the supplied file "ItemMaint.c". Write the required code in this file.

Step 7: Compile and execute the program.

Estimated time: 45 min

Summary of this assignment:

In this exercise, you have learnt:

- Integrating multiple files in one workspace
- Usage of array of structures

Assignment 18: Using the array of structures - Part II

Objective: Including multiple source files in a workspace, complete coding of the functions, compile and execute as a whole project.

Background: The code for a Retail Application has been developed. The source files need to be integrated in one workspace, compiled and executed.

Problem Description:

Extend the Retail Application written in the previous assignment by adding the supplier module. The following functionality needs to be implemented:

Add supplier: The program must accept supplier code (4 digits), supplier name (25 characters) and phone number (10 characters) from the user. After validations, the details should be stored into the array of **structure supplier**

Note: Due to range limitation of int data type, char array for storing phone number is used.

Validations:

- Supplier code should not be existing in the supplier array
- The Supplier name can contain only alphabets, digits and blank spaces
- The Phone number can contain only digits.

Step 1: Create a file “**supplier.c**” in the already created workspace i.e. “RetailAppln” of previous Assignment.

Step 2: Write the code for implementing the functionalities for supplier module in the file created in step 1.

Step 3: Modify the main menu in **RetailAppln.c** to include an option “Add supplier” and appropriately calling the function

Step 4: Compile and execute the program

Estimated time: 40 min

Summary of this assignment:

In this exercise, you have learnt:

- Extending existing code
- Usage of array of structures

Assignment 50: Exercises for Self Review

1. Find the output of the following program:

```
int giValue = 5;
void fnUpdate() {
    giValue = giValue - 2;
}
int main(int argc, char **argv) {
    giValue = giValue + 5;
    fnUpdate();
    printf("%d", giValue);
}
```

```
}
```

2. Find the output of the following program:

```
int fnUpdate(int iTemp) {  
    iTemp = iTemp + 5;  
    return iTemp;  
}  
int main(int argc, char ** argv) {  
    int iValue = 25;  
    iValue = fnUpdate(iValue);  
    printf("%d",iValue);  
}
```

3. Find the output of the following program:

```
void fnUpdate(int *piTemp1, int *piTemp2) {  
    *piTemp1 = *piTemp1 + 5;  
    *piTemp2 = *piTemp2 + 2;  
}  
int main(int argc, char **argv) {  
    int iValue1= 100, iValue2 = 50;  
    fnUpdate(iValue1, iValue2);  
    printf("%d %d",iValue1, iValue2);  
}
```

6. Find the output of the following code snippets:

I.

```
void fnUpdate(int *piPointer)  
{  
    *piPointer = *piPointer + 100;  
}  
int main (int argc, char **argv)  
{  
    int iNumber = 10;  
    printf("%d ",iNumber);  
    fnUpdate(&iNumber);  
    printf("%d",iNumber);  
    return 0;  
}
```

II.

```
void fnSwap(int *piPointer1 , int *piPointer2)  
{  
    int *piTemp;  
    piTemp = piPointer1;  
    piPointer1 = piPointer2;  
    piPointer2 = piTemp;  
}  
int main (int argc, char **argv)
```

```
{
    int iNumber1 = 10, iNumber2 = 20;
    printf("Before Swap, %d %d\n",iNumber1,iNumber2);
    fnSwap(&iNumber1,&iNumber2);
    printf("After Swap, Swap%d %d\n",iNumber1,iNumber2);
    return 0;
}
```

III. void fnSwap(int *piPointer1 , int *piPointer2)

```
{
    int iTemp;
    iTemp = *piPointer1;
    *piPointer1 = *piPointer2;
    *piPointer2 = iTemp;
}
int main (int argc, char **argv)
{
    int iNumber1 = 10, iNumber2 = 20;
    printf("Before Swap, %d %d\n",iNumber1,iNumber2);
    fnSwap(&iNumber1,&iNumber2);
    printf("After Swap, Swap%d %d\n",iNumber1,iNumber2);
    return 0;
}
```

IV. void fnUpdate(int aiTemp[])

```
{
    int iIndex;
    for (iIndex=0;iIndex < 5; iIndex++)
    {
        aiTemp[iIndex] = aiTemp[iIndex]+5;
    }
}
int main (int argc, char **argv)
{
    int aiArray[]={10,20,30,40,50};
    int iIndex;
    fnUpdate(aiArray);
    for (iIndex=0;iIndex < 5; iIndex++)
    {
        printf("%d\n",aiArray[iIndex]) ;
    }
    return 0;
}
```

```
V. void fnUpdate(int iTemp)
{

    iTemp = iTemp+5;

}
int main (int argc, char **argv)
{
    int aiArray[]={10,20,30,40,50};
    int ilIndex;
    for (ilIndex=0;ilIndex < 5; ilIndex++)
    {
        fnUpdate(aiArray[ilIndex]);
    }
    for (ilIndex=0;ilIndex < 5; ilIndex++)
    {
        printf("%d\n",aiArray[ilIndex]) ;
    }
    return 0;
}
```

```
VI. int fnUpdate(int iTemp)
{

    iTemp = iTemp+5;
    return iTemp;

}
int main (int argc, char **argv)
{
    int aiArray[]={10,20,30,40,50};
    int ilIndex;
    for (ilIndex=0;ilIndex < 5; ilIndex++)
    {
        aiArray[ilIndex] = fnUpdate(aiArray[ilIndex]);
    }
    for (ilIndex=0;ilIndex < 5; ilIndex++)
    {
        printf("%d\n",aiArray[ilIndex]) ;
    }
    return 0;
}
```



```
VII. void fnUpdate(int *piPointer)
{
    (*piPointer)++;
}

int main (int argc, char **argv)
{
    int aiArray[]={10,20,30,40,50};
    fnUpdate(aiArray);
    printf("%d",*aiArray);
    return 0;
}

VIII. void fnUpdate(int *piPointer)
{
    int iLoopIndex;
    for (iLoopIndex=0;iLoopIndex<5;iLoopIndex++)
    {
        *piPointer = *piPointer + 1;
        piPointer++;
    }
}

int main (int argc, char **argv)
{
    int aiArray[]={10,20,30,40,50};
    int iLoopIndex;
    fnUpdate(aiArray);
    for (iLoopIndex=0;iLoopIndex<5;iLoopIndex++)
    {
        printf("%d",aiArray[iLoopIndex]);
    }
    return 0;
}
```

4. Consider the following program:

```
#include<stdio.h>
int fnAddSub(int,int);
int main(int argc,char **argv) {
    int iNumber1 = 5,iNumber2 = 8,iRet1;
    iRet1 = fnAddSub(iNumber1,iNumber2);
    printf("%d", iRet1);
    return 0;
}
```

```
int fnAddSub(int iNumber1,int iNumber2){
    int iSum,iDiff;
    iSum = iNumber1 + iNumber2;
    iDiff = iNumber1 - iNumber2;
    return (iSum);
    return (iDiff);
}
```

What would be returned by fnAddSub to main()? Identify the output in the main() and correct the statements if any.

5. Debug the following program:

```
#include<stdio.h>
int fnAdd(int,int);
int main(int argc,char **argv) {
    int iNumber1 = 5,iNumber2 = 8,iRet1;
    iRet1 = fnAdd(iNumber1,iNumber2);
    iRet2 = fnMultiply(iNumber1,iNumber2);
    printf("%d", iRet1);
    return 0;
}
```

```
int fnAdd(int iNumber1,int iNumber2){
    int iSum;
    iSum = iNumber1 + iNumber2;
    return (iSum);
}
```

6. Find the output of the following program:

```
struct _items {
    int iltemId;
    char acItemName[15];
};
typedef struct _items items;
int main(int argc,char **argv) {
    items sltems;
    sltems.iltemId = 100;
    strcpy(sltems.acItemName,"Soap");
    printf("%d\n",sltems.iltemId);
    printf("%s", sltems.acItemName);
}
```

7. Consider the following structure declaration:

```
typedef struct _Employee{
    int iEmpld;
    char acName[50];
    double dBasic;
```

```
} Employee;
```

- a) What is _Employee?
- b) What is Employee?
- c) What is the use of typedef statement here?

8. Consider the following structure declaration:

```
struct _MyStructure{  
    int iVar;  
    int aiArray[5];  
};  
typedef struct _MyStructure MyStructure;
```

```
MyStructure sVar={10,{1,2,3,4,5}};
```

How will you access the second element of the array aiArray?

9. Identify whether the given statements are correct or wrong and why?

- a.

```
typedef struct _Employee{  
    a. int iEmpId=100;  
    b. char acName[50];  
    c. double dBasic;  
} Employee;
```
- b.

```
typedef struct _Employee{  
    a. int iEmpId;  
    b. char acName[50];  
    c. double dBasic;  
} Employee={1001,"Meena",10000};
```
- c.

```
struct Employee{  
    a. int iEmpId;  
    b. char acName[50];  
    c. double dBasic;  
} sEmployee={1001,"Meena",10000};
```

10. Consider the following structure declaration:

```
typedef struct _Employee{  
    int iEmpId;  
    char acName[50];  
    double dBasic;
```

```
} Employee;
```

```
Employee sEmp;
```

```
Employee *psEmp=&sEmp;
```

- a) How will you accept values into the acName using the sEmp variable? (Write only the scanf statement)
- b) How will you access the iEmpId variable using the psEmp variable ?

11. Identify the valid structure declarations from the following:

- a)

```
typedef struct _Date {  
    int iDay;  
    int iMonth;
```

```
        int iYear;
    } Date;
    typedef struct _Exam {
        int iExamId;
        Date sDateOfExam;
    } Exam;
    Exam sExam;
b) struct _Date {
    int iDay;
    int iMonth;
    int iYear;
} Date;

struct _Exam {
    int iExamId;
    Date sDateOfExam;
};
typedef struct _Exam Exam

int main(int argc, char** argv){
    Exam sExam;
    return 0;
}
c) typedef struct _Exam {
    int iExamId;
    struct _Date {
        int iDay;
        int iMonth;
        int iYear;
    };
} Exam;
Exam sExam;
```

Day-5 Assignments

Use the Practice Guide for Hands-On

Day-6 Assignments

Use the Practice Guide for Hands-On (contd..)

Assignment 51: Identifying Test cases using Boundary value analysis

Objective: To identify and document test cases for a given functionality, using Boundary value Analysis.

Problem Description: Consider the function below whose functionality is described in its documentation.

```

/*****
* Function: fnComputeInterestRate
* Description: Given the term of the deposit, computes and returns
*   the interest rate. Minimum duration for the deposit is 1 month and
*   maximum duration is 48 months
*
* Criteria for Interest:
*   1 month to 2 months  1%
*   3 months to 5 months  3.5%
*   6 months to 11 months  4.5%
*   12 months to 24 months  5.0%
*   25 months to 48 months 5.5%
* PARAMETERS:
* int iNumberOfMonths  Number of months
*
* RETURNS: Appropriate Rate of Interest. -1.0 in case of error.
*****/

float fnComputeInterestRate (int iNumberOfMonths) {
    ...
    ...
}

```

Estimated time: 20 minutes

Note: The range to be considered for the test cases involving duration in months is :
Lower limit 1 and Upper limit 48

Step 1: Create an Excel sheet with the columns shown below.

Sl No	Test case name	Test Procedure	Pre-condition	Expected Result	Reference to Detailed Design / Spec Document

Step 2: Identify the test cases based on boundary value analysis.

Step 3: Document them in the Excel sheet created earlier.

Summary of this assignment:

In this assignment, you have learnt:

- Creating a spreadsheet to document test cases
- Documenting test cases
- Identifying test cases using Boundary value analysis

Assignment 52: Testing a program using the test cases and logging Test Results and Defects

Objective: To test code using test cases documented in earlier assignments and to log defects into the Defect Tracking Excel sheet created earlier.

Problem Statement: Ask your trainer to provide you with the code required (intrate.c).

Estimated time: 30 minutes

Step 1: Create a defect tracking excel sheet as shown in the diagram below. (Also Refer to “6.2.1 Defect Tracking System” in the course material).

	A	B	C	D	E	F	G	H	I
1	#	Submitted by	Description	Detected Stage	Assigned To	Type of Defect	Injected Stage	Action Taken	Notes
2	101	Nagendra	When 100 is passed for iPercentScore, fnFindGrade returns 'Z'	Unit Testing	John Doe	Logical Error	Coding	To be fixed	(Additional notes here)
3	102								
4	103								

Step 2: Create an empty project in Visual C++ (Console Application) with name 'intrate' And add the source code provided into the project.

Step 3: Compile the program. The program should compile with Zero errors and warnings.

Step 4: Open the Unit Test Plan Excel sheet containing the test cases identified using Boundary value analysis and Equivalence partitioning. Add an additional column to the Excel Sheet “**Results**”.

Step 5: Execute test cases one by one and compare if the expected result is the same as what is documented in the Test Plan.

If the expected result is the same, mark the test case as “**PASSED**” in the “**Results**” column.

If the expected result in the Test Plan is different from the result returned by the program:

- Mark the test case as “**FAILED**” in the “**Results**” column
- Add an entry into the Defect Tracking Excel sheet
- The description of the defect in the Defect Tracking Excel sheet should contain full details of the preconditions if any, inputs, output, the expected output and any other information that will help a developer to debug and fix the defect.

Step 6: Execute all the test cases. Mark the test result for all the tests.

Summary of this assignment:

In this assignment, you have learnt:

- Creating a Defect Tracking Excel sheet
- Unit Testing according to the Test Plan
- Recording Test Results
- Logging defects

Assignment 53: Debugging using IDE (Step-by-Step execution)

Objective: To debug code using the IDE by using step-by-step execution feature.

Problem Statement: Ask your trainer to provide you with the code required (kiosk.c).

Estimated time: 30 minutes

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

Step 2: Add the source code provided into the project.

Step 3: Compile the program. The program should compile with Zero errors and warnings.

Step 4: Follow the steps in sections “6.4 Debugging using the IDE” and “6.4.1 Debugging using step by step execution”. Identify the bug using Step-by-Step execution approach as described in the course material.

Step 5: Fix the bugs identified.

Step 6: Recompile the program.

Step 7: Test the program.**Summary of this assignment:**

In this exercise, you have learnt:

- Debugging using the IDE
- Able to use judgment when to use “Step-into” and when to use “Step-over” functionality while debugging
- Identifying and fixing bugs

Assignment 54: Exercises for Self Review

12. Assume that an employee information system accepts the job code of employees in the range 1 to 5. Identify the test cases using boundary value analysis.
13. For the above exercise, identify the test cases using equivalence partitioning.
14. When a visual debugger is not available, which are the debugging techniques that could be used?
15. What are the prerequisites for the code review?

Appendix: Additional Assignments**Simple Level Assignments**

1. Display as follows using selectional and/or iterational constructs:

(a) abcdefgh z

(b) AaBbCcDdEe ... Zz

(c) AbCdEfGhIj

(d) *
* *
* * *
* * * *

(e) 1
1 2
1 2 3

(f) 1

2 3
4 5 6

Medium Level Assignments

2. Accept a string and check whether it contains only alphabets (a-z or A-Z)
3. Create an array that stores the marks scored by students in a subject. You can assume the size of the student array to be 10 . The range of the marks must be 0 to 25. Write a program to accept the marks of the students and count the number of students who have scored each marks in the range, i.e you need to count how many students have scored 0, how many students scored 1, how many students scored 2 ... how many students scored 25
4. Accept a string from the user and reverse the string using pointers
5. Accept a number from user. Find the reverse of that number.
6. Create an array of 10 numbers. Find the sum of all the elements of that array.

Advanced Level Assignments

7. Accept a string of 'n' characters length. 'n' can be fixed using #define. Replace characters in the string as follows:
 - vowels: a by 1 , e by 2 , i by 3 , o by 4, u by 5
 - all consonants to be replaced by 9
 - all other characters including special characters to be replaced by 8
 -
8. Create an array to hold the employee numbers. Create another array to hold the employee names. Create another array to hold the employee phone numbers. Assume appropriate data types for the array. The user inputs the employee number and your program should display all the details of the employee in a tabular format. Also take care that all employee numbers are unique