Revision of MT262 course

Block 1 Unit 1

- Use of Builder
- Understanding following terms :
  Embedded system, procedural model of programming, event-driven model of programming, source, compile, debug, programming environment, variable, identifier, operating system, application program, code specification, code implementation, input, output, bit, byte, megabyte, RAM, ROM

Block 1 Unit 2

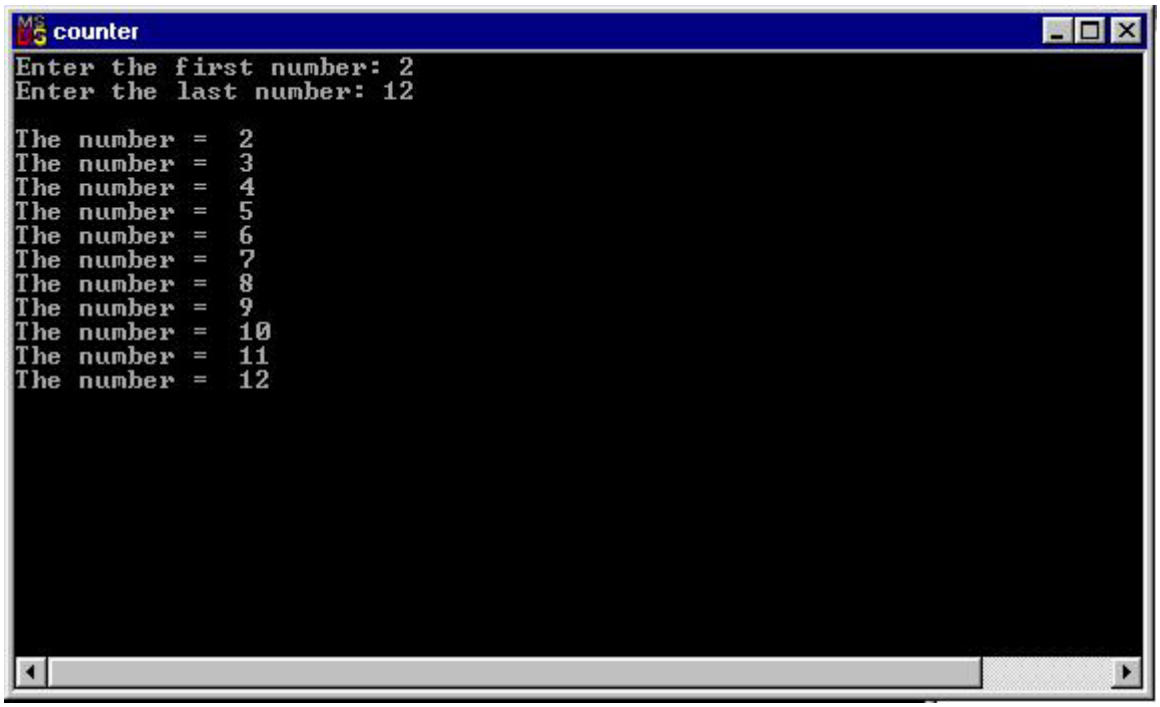
- Anatomy of a program consists of :
  Sequence : i.e. statement 1 is executed before statement2 ..etc
  Selection : if…else, switch…case
  Repetition : while loop, for loop, do…until
- Refine a design step making use of
  if…then…else…ifend
  while loops
- Refine a design until it is ready for coding, in particular by making appropriate variable declarations and initialising relevant variables
- Use the standard numbering system for design steps, through all stages
- Translate a final design into C++ code
- Use recommended spacing, indentation and line breaks when writing designs and code
- Use the standard design and C++ symbols but remember the followings :

Function	**Design**	**C++ code**
Assignment	<-	=
Equality	=	==
While	loop while	while
If	If…then…else	If…else
Numbering	Yes	No

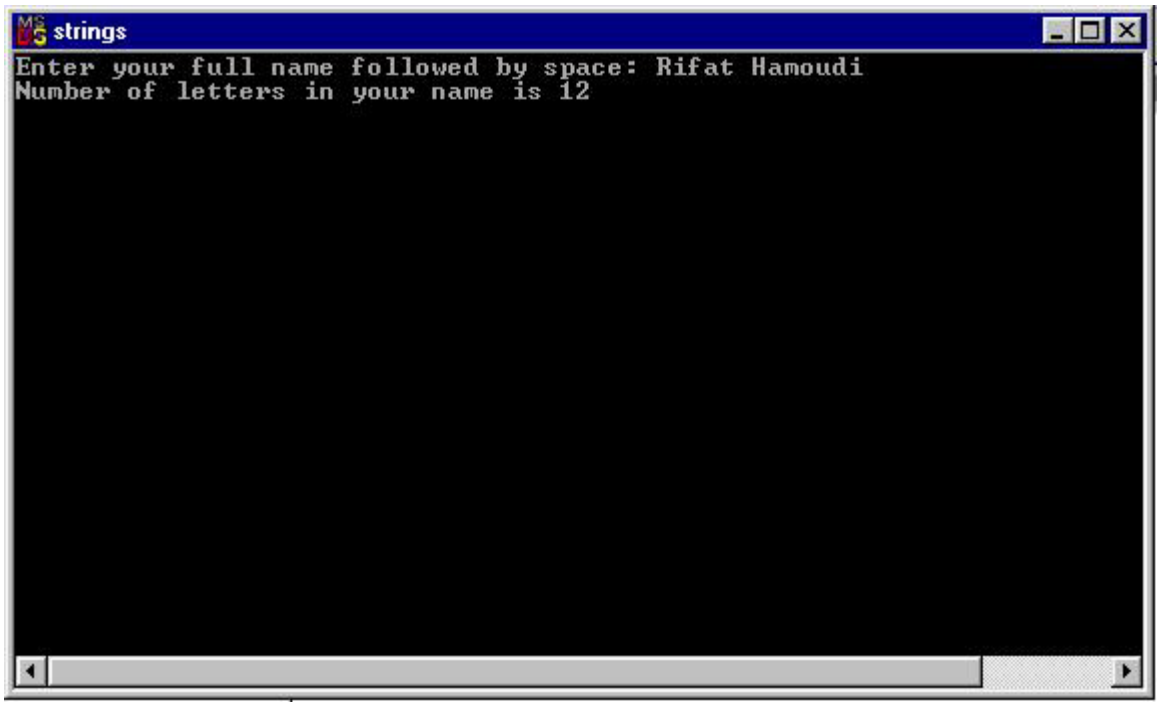
- Use and understand the use of the following terms : top-level design, final design, coding, keyword, variable, assignment of variables, declaration of variables, variable type, data table, identifier, function, if step, if statement, then clause, else clause, branching, while loop, loop body, condition, negation.

Exam questions

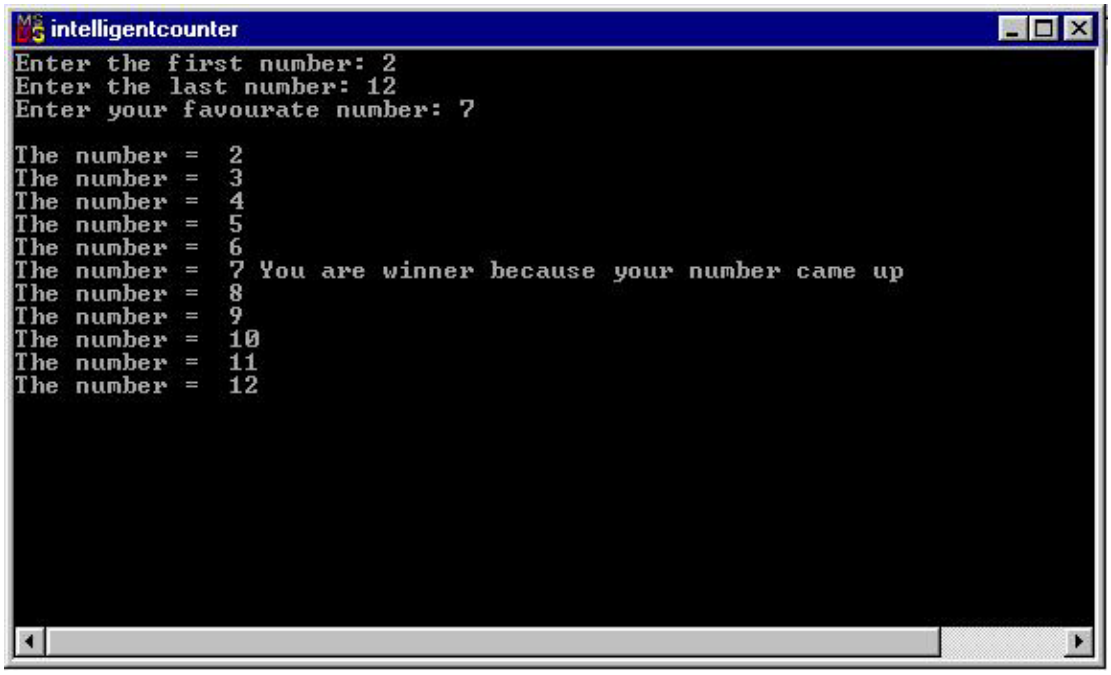
1) Design and implement a C++ program that counts a range of numbers entered by the user.
An example output is as shown below :



2) Design and implement a C++ program that counts the number of letters in a user's name.
An example output is as shown below :



3) Modify question (1) so that the program asks the user to input a favourite number and displays the following text "You are a winner because your number came up" next to the user's favourite number. An example output is as shown below :



Answer to Question 1

The design

Top level design

```
1       read in the user numbers
2       initialise variables
3       loop while the numbers are in the user's range
4               process next number
5       loopend
```

Final design using stepwise refinement

```
1.1.1 write out "Enter the first number: "
1.1.2 read in FirstNumber
1.2.1 write out "Enter the last number: "
1.2.2 read in LastNumber
1.3.2 read in favouriteNumber
2.1   count <- FirstNumber
3     loop while count <= LastNumber
4.1         write out "The number =  "
4.2         write out count
4.3         count <- count + 1
5     loopend
```

The C++ source code

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 1
        Title   :   Counter Program
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project - MT262io.lib
USELIB("MT262io.lib");
//---------------------------------------------------------------------
--------

#pragma argsused // to avoid the prodution of warning messages
relating to argc
                // and **argv

int main(int argc, char **argv)
{
    int count;
    int FirstNumber;
    int LastNumber;

    FirstNumber = ReadIntPr("Enter the first number: ");
    LastNumber = ReadIntPr("Enter the last number: ");
    count = FirstNumber;

    while (count <= LastNumber)
```

```
    {
        WriteIntPrCr("The number = ", count);
          count = count + 1;
    }
    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

Answer to Question 2

The design

Top level design

```
1       read in user name
2       count the letters in the user's name
3       write out the letters in the user's name
```

Final design using stepwise refinement

```
1.1     write out "Enter your full name followed by space: "
1.2     read in Line
2.1.1   Index loop while Index <= Length(Line)
2.3.1           if  PreviousLetter != ' '  then
2.3.2                   LetterCount ifend
2.3.4           Index loopend
3.1     write out "Number of letters in your name is", LetterCount
```

The C++ source code

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 1
        Title   :   Count letters in name program
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project - MT262io.lib
USELIB("MT262io.lib");

//---------------------------------------------------------------------
--------
#pragma argsused // to avoid the prodution of warning messages
                 // relating to argc and **argv

int main(int argc, char **argv)
{
    AnsiString Line;
    int Index;
    int LetterCount;
    char PreviousLetter;
```

```
    Line = ReadStringPr("Enter your full name followed by space: ");
    Index = 1;
    LetterCount = 0;
    PreviousLetter = ' ';

    while (Index <= Length(Line))
    {
        if (PreviousLetter != ' ')
            LetterCount = LetterCount + 1;
        PreviousLetter = Line[Index];
        Index = Index + 1;
    }
    WriteIntPrCr("Number of letters in your name is", LetterCount);
    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

Answer to Question 3

The design

Top level design

```
1       read in the user numbers
2       initialise variables
3       loop while the numbers are in the user's range
4                process next number
5       loopend
```

Final design using stepwise refinement

```
1.1.1   write out "Enter the first number: "
1.1.2   read in FirstNumber
1.2.1   write out "Enter the last number: "
1.2.2   read in LastNumber
1.3.1   write out "Enter your favourite number: "
1.3.2   read in FavouriteNumber
2.1     loop while count <= LastNumber
4.1             write out "The number =  "
4.2             write out count
4.3             if count = FavouriteNumber then
4.4.2            write out "You are a winner because your number came
up"
4.4.3           ifend
5       loopend
```

The C++ source code

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 1
        Title   :   Intelligent Counter Program
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/
```

```
#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project - MT262io.lib
USELIB("MT262io.lib");
//---------------------------------------------------------------------
--------

#pragma argsused // to avoid the prodution of warning messages
relating to argc
                 // and **argv

int main(int argc, char **argv)
{
    int count;
    int FirstNumber;
    int LastNumber;
    int FavouriteNumber;

    FirstNumber = ReadIntPr("Enter the first number: ");
    LastNumber = ReadIntPr("Enter the last number: ");
    FavouriteNumber = ReadIntPr("Enter your favourite number: ");
    Count = FirstNumber;

    while (count <= LastNumber)
    {
        WriteIntPr("\nThe number = ", count);

        if (count == FavouriteNumber)
        {
          WriteString(" You are winner because your number came up");
        }
        count = count + 1;
    }
    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

Block 1 Unit 3

- Design solutions of simple problems that involve pre and post-conditioned while loops;
- Design solutions for simple problems that involve case steps
- Include elementary trapping of user errors in your design
- Code designs that use while and/or case constructs
- Use string variable and operations on them
- Use the notion of an index to access the individual entries that make up a string
- Complete a program from a coded design including ensuring that the course library is included in your project
- Show in problem solving an appreciation of the "specify, design and refine, code, test" process

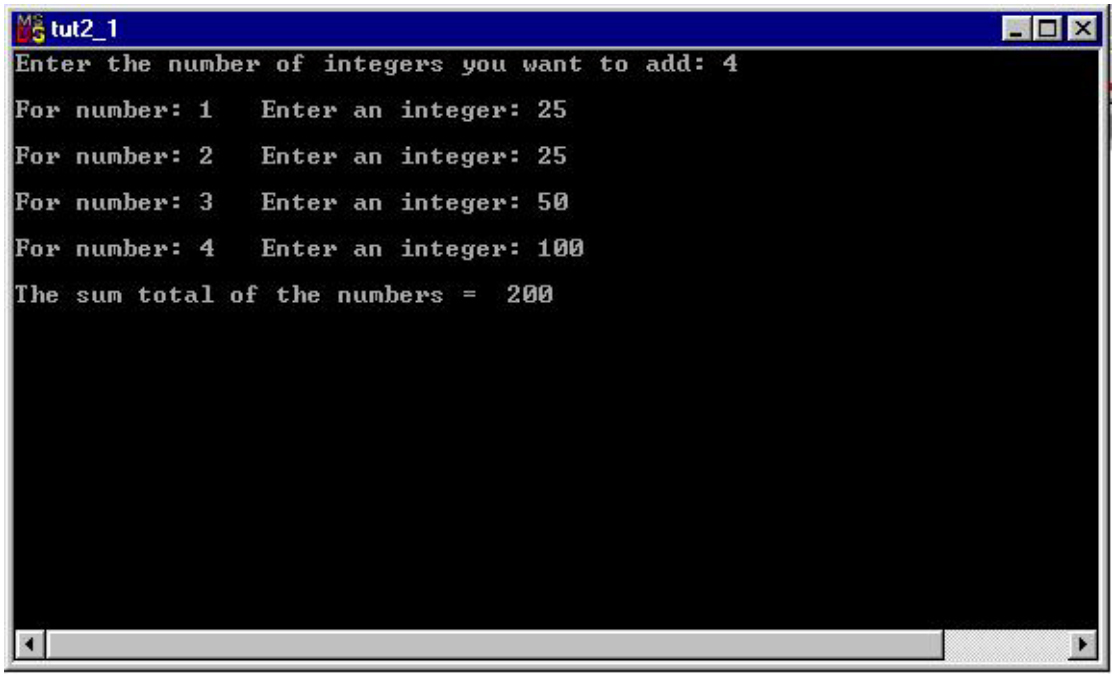
Block 1 Unit 4

- Draw up suitable data sets to test a program
- Trace a design by hand
- Make use of trace to detect errors
- Add comments to design and code as a form of solution documentation

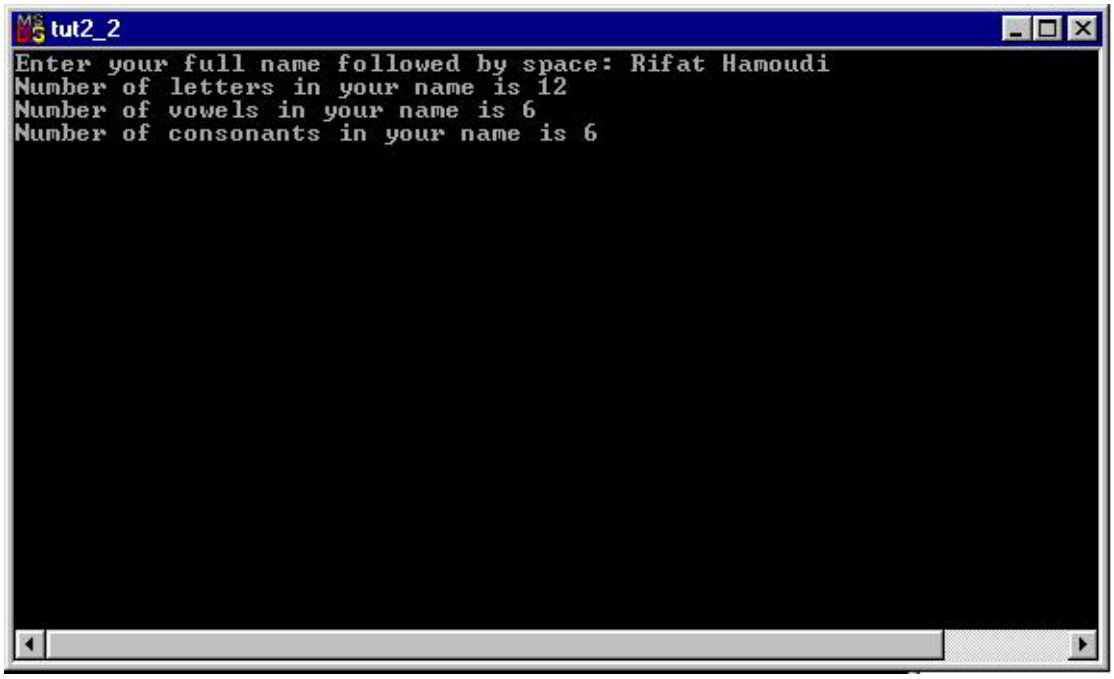
Exam questions

1) Design and implement a C++ program that adds a range of numbers entered by the user.

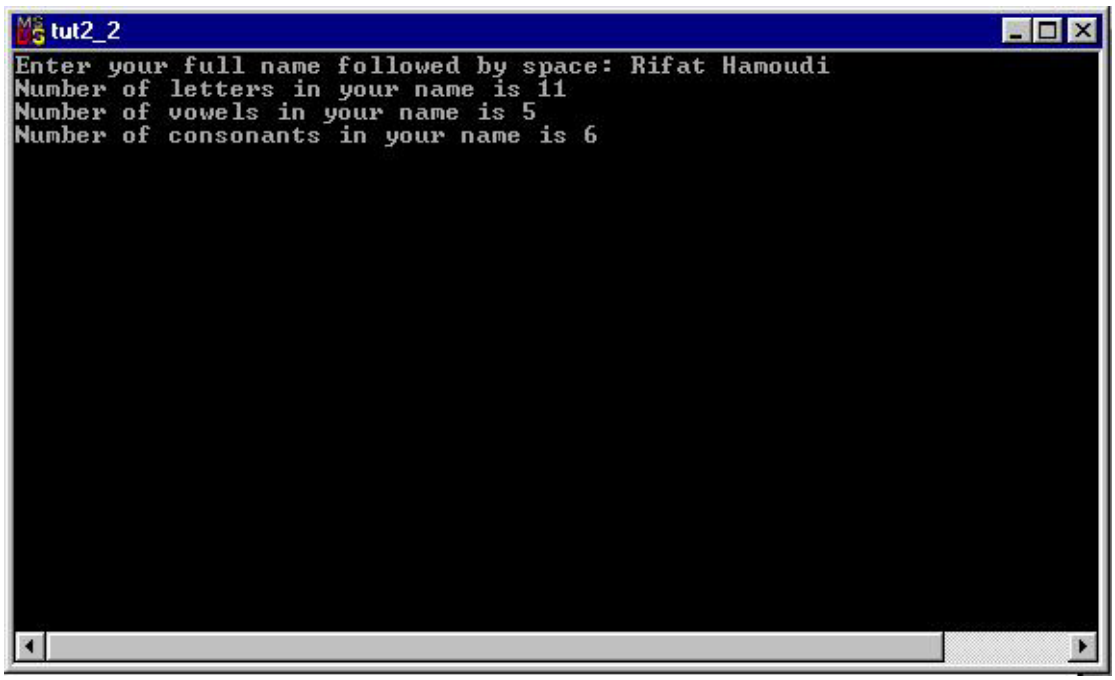
An example output is as shown below :



2) Design and implement a C++ program that counts the total number of letters, number of vowels and number of consonants in a user's name.
An example output is as shown below :



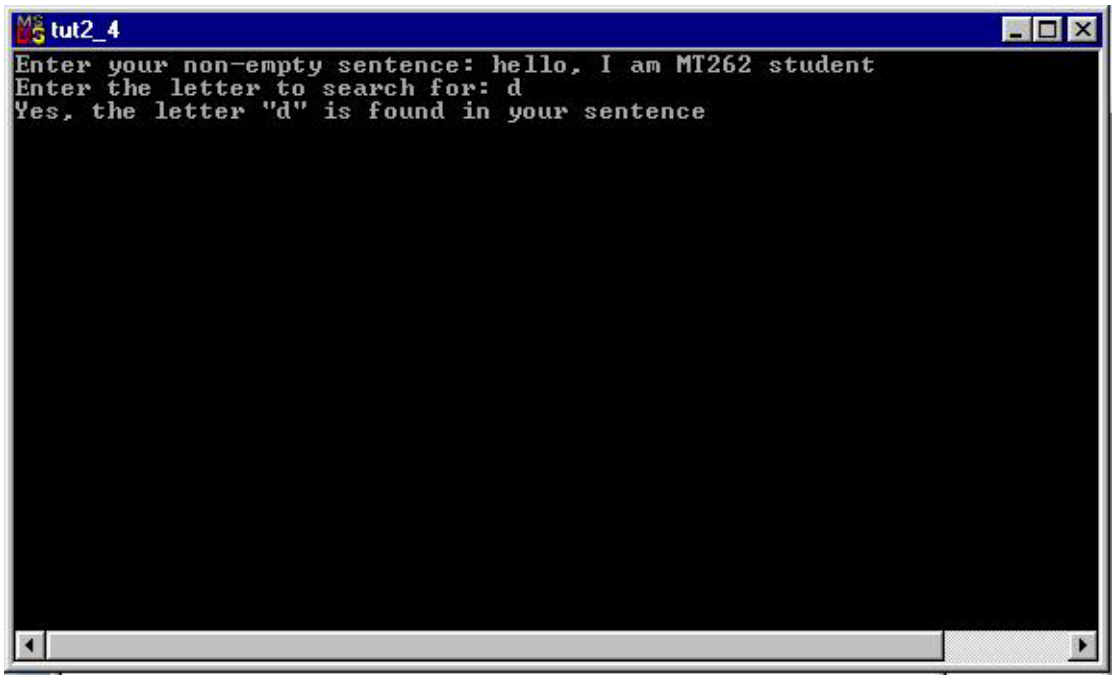
3) In question (2) the program produces the wrong results if the user does not type a space at the end of the name as shown below.



Identify the source of the semantic error in question (2) and correct it so that the program will count the correct number of letters even if the user does not enter a space after the name.

4) Design and implement a program that allows to enter a sentence and search for the existance or otherwise of a letter within the sentence. Include a data table of your design.

An example output is shown below :



Answer to Question 1

The design

Top level design

1 read in the total number of integers to be added
2 initialise variables
3 **loop** while there are still numbers to be added
4 process next number
5 **loopend**

6 Display the total sum of numbers

Final design using stepwise refinement

1.1 write out “Enter the number of integers you want to add: “
1.2 read in Number_of_Integers
2.1 count <- 1
2.2 current_number <- 0
2.3 sum <- 0
3 **loop while** count <= Number_of_Integers
4.1 write out “For number:“
4.2 write out count
4.3.1 write out “ Enter an integer:”
4.3.2 read in current_number
4.4 sum <- sum + current_number
4.5 count <- count + 1
5 **loopend**
6.1 write out “The sum total of the numbers = “
6.2 write out sum

The C++ source code

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 1b
        Title   :   Calculating the sum in a range of numbers
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project -> MT262io.lib
//---------------------------------------------------------------------
--------
USELIB("Mt262io.lib");
//---------------------------------------------------------------------
--------
#pragma argsused // to avoid the prodution of warning messages
relating to argc
                // and **argv

int main(int argc, char **argv)
{
    int count;
    int sum;
    int current_number;
    int Number_of_Integers;
```

```
    Number_of_Integers = ReadIntPr("Enter the number of integers you
want to add: ");
    count = 1;
    current_number = 0;
    sum = 0;

    while (count <= Number_of_Integers)
    {
        WriteIntPrCr("For number:", count);
        current_number = ReadIntPr("   Enter an integer: ");
        sum = sum + current_number;
        count = count + 1;
    }

    WriteIntPrCr("The sum total of the numbers = ", sum);
    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

Answer to Question 2

The design

Top level design

1 read in user name
2 count the letters, vowels and consonants in the user's name
3 write out the letters, vowels and consonants in the user's name

Final design using stepwise refinement

1.1 write out "Enter your full name followed by space: "
1.2 read in Line
2.1.1 Index <- 1
2.1.2 LetterCount <- 0
2.1.3 VowelCount <- 0
2.1.4 ConsonantCount <- 0

2.1.5 Letter <- ‘ ‘
2.2 **loop while** Index <= Length(Line)
2.3.1 **if** Letter != ' ' **then**
2.3.2 LetterCount <- LetterCount + 1
2.3.3 **if** Letter = vowel **then**
2.3.4 VowelCount <- VowelCount + 1
2.3.5 **else**
2.3.6 ConsonantCount <- ConsonantCount + 1
2.3.7 **ifend**
2.3.8 **ifend**
2.3.9 Letter <- Line[Index]
2.3.10 Index <- Index + 1
2.4 **loopend**
3.1 write out "Number of letters in your name is", LetterCount
3.2 write out "Number of vowels in your name is", VowelCount
3.3 write out "Number of consonants in your name is", ConsonantCount

The C++ source code

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 1b
        Title   :   Count letters, vowels and consonants in a name
                    with semantic error
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project -> MT262io.lib
USELIB("MT262io.lib");
//---------------------------------------------------------------------
--------
#pragma argsused // to avoid the prodution of warning messages
relating to argc
                 // and **argv

int main(int argc, char **argv)
{
    AnsiString Line;
    int Index;
    int LetterCount;
    int VowelCount;
    int ConsonantCount;
    char Letter;

    Line = ReadStringPr("Enter your full name followed by space: ");
    Index = 1;
    LetterCount = 0;
    VowelCount = 0;
    ConsonantCount = 0;
    Letter = ' ';
```

```
    while (Index <= Length(Line))
    {
        if (Letter != ' ')
        {
            LetterCount = LetterCount + 1;

            if ((Letter == 'a') || (Letter == 'e') ||
            (Letter == 'i') || (Letter == 'o') ||
            (Letter == 'u'))
                VowelCount = VowelCount + 1;
            else
                ConsonantCount = ConsonantCount + 1;
        }
        Letter = Line[Index];
        Index = Index + 1;
    }
    WriteIntPrCr("Number of letters in your name is", LetterCount);
    WriteIntPrCr("Number of vowels in your name is", VowelCount);
    WriteIntPrCr("Number of consonants in your name is",
ConsonantCount);

    getchar();  // to keep the display on until you press Enter key
    return 0;


}
```

Answer to Question 3

To correct the semantic error, put step 2.3.9 in the final design as step 2.3.1 as follows:

```
…..
2.1.5   Letter <- ‘ ‘
2.2     loop while Index <= Length(Line)
2.3.1           Letter <- Line[Index]
2.3.2           if  Letter != ' '  then
2.3.3                   LetterCount <- LetterCount + 1
2.3.4                   if  Letter is a vowel  then
2.3.5                      VowelCount <- VowelCount + 1
2.3.6                   else
2.3.7                      ConsonantCount <- ConsonantCount + 1
2.3.8                   ifend
2.3.9           ifend
2.3.10  Index <- Index + 1
2.4     loopend
```

……. etc

The C++ source code

```
/*
        Unit    :   MT262
```

```
        Tutorial:   Tutorial 1b
        Title   :   Count letters, vowels and consonants in a name
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project -> MT262io.lib
USELIB("MT262io.lib");
//---------------------------------------------------------------------
--------
#pragma argsused // to avoid the prodution of warning messages
relating to argc
                // and **argv

int main(int argc, char **argv)
{
    AnsiString Line;
    int Index;
    int LetterCount;
    int VowelCount;
    int ConsonantCount;
    char Letter;

    Line = ReadStringPr("Enter your full name: ");
    Index = 1;
    LetterCount = 0;
    VowelCount = 0;
    ConsonantCount = 0;
    Letter = ' ';

    while (Index <= Length(Line))
    {
        Letter = Line[Index];
        if (Letter != ' ')
        {
            LetterCount = LetterCount + 1;

            if ((Letter == 'a') || (Letter == 'e') ||
            (Letter == 'i') || (Letter == 'o') ||
            (Letter == 'u'))
                VowelCount = VowelCount + 1;
            else
                ConsonantCount = ConsonantCount + 1;
        }
        Index = Index + 1;
    }
    WriteIntPrCr("Number of letters in your name is", LetterCount);
    WriteIntPrCr("Number of vowels in your name is", VowelCount);
    WriteIntPrCr("Number of consonants in your name is",
ConsonantCount);

    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

Answer to Question 4

Data table

Type	Identifier	Description
AnsiString	Line	Sentence entered by the user
Char	Letter	Letter entered by the user to be searched for in the sentence
Integer	Begin	Runs over the indexes of characters in *Line*
Integer	End	Used to terminate the loop if the letter is found in the sentence
Boolean	Found	Set true if *Line* contains the letter; otherwise set false

Final design

1.1 write out “Enter your non-empty sentence : “
1.2 read in Line
1.3 read in Letter
1.4 Begin <- 1
1.5 End <- Length(Line)
1.6 Found <- **false**

2.1 **loop while** Begin < End
3.1 **if** Line[Begin] = Letter **then**
3.2 Found <- **true**
3.3 **else**
3.4 Found <- **false**
3.5 **ifend**
4.1 Begin <- Begin + 1
5 **loopend**
6.1 **if** Found = **true then**
6.2 write out “Yes, the letter is found in the sentence.”
6.3 **else**
6.4 write out “No, the letter is not found in the sentence.”
6.5 **ifend**

This consist of semantic error!

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 1b
        Title   :   Program to search for a letter in a sentence
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project -> MT262io.lib
USELIB("MT262io.lib");
//---------------------------------------------------------------------
--------
#pragma argsused // to avoid the prodution of warning messages
relating to argc
                // and **argv

int main(int argc, char **argv)
{
    AnsiString Line;
    char Letter;
    bool Found;
    int Begin;
    int End;

    Line = ReadStringPr("Enter your non-empty sentence: ");
    Letter = ReadCharPr("Enter the letter to search for: ");
    Found = false;
    Begin = 1;
    End = Length(Line);

    while (Begin < End)
    {
        if (Line[Begin] == Letter)
            Found = true;
        else
            Found = false;
```

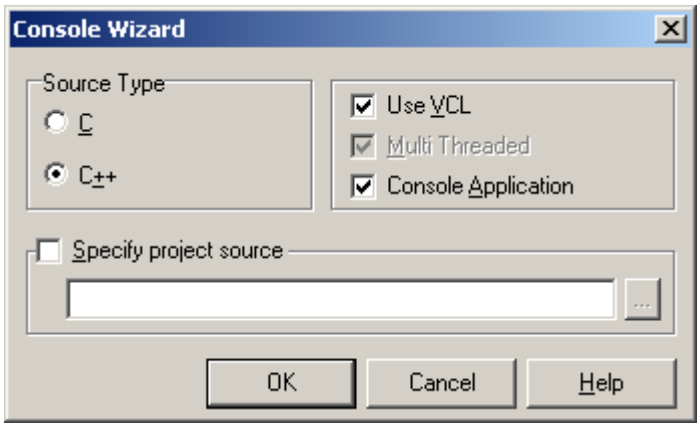
```
        Begin = Begin + 1;
    }

    if (Found == true)
    {
        WriteString("Yes, the letter \"");
        WriteChar(Letter);
        WriteStringCr("\" is found in your sentence");
    }
    else
    {
        WriteString("No, the letter \"");
        WriteChar(Letter);
        WriteStringCr("\" is not found in your sentence");
    }

    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

How to get the code to work and compile on your computer

1) load C++ Builder
2) Click on **File** | **New** then click on **Console Wizard**
3) Make sure the Use VCL and Console Application are checked as below.



4) Click OK
5) Type the C++ code above in the file
6) Click on **File** | **Save Project As** then give it a name e.g. counter click OK
7) Click on **Run** | **Run** (or press F9 key) to run the program

Block 2 Unit 1

- Create data structure definition tables for records and write data structure definition in C++, declaring variables in course style
- Use assignment and equality of records appropriately in designs and code those designs
- Declare in C++ arrays of built-in or programmer defined base type and tables
- Manipulate elements in an array
- Use the dot notation for fields of a record or table
- Use for loops in design, and code such designs
- Use all above ideas in solving problems
- Follow the design and coding of a linear search

Block 2 Unit 2

- Identify and write C++ statements that call functions, including those with more than one parameter
- Interpret and write function specification boxes and C++ function declarations
- Write a suitable program to test a function that you have designed
- Use the course style for declaring and defining functions
- Use void appropriately in function prototypes
- Convert a suitable program to function form for general availability
- Make use of the notion of a string as a sequence of characters to define string handling functions
- Explain the role of functional decomposition in the construction of a modular program, and write such programs, including the use of library header files

Block 2 Unit 3

- Use identifiable subtasks in a design as the basis for modular solution to a problem
- With guidance, group together data types and functions corresponding to subtasks in appropriate modules
- Make use of temporary statements in place of function calls in the early development of a modular solution to enable the main program to be tested, and use a dummy implementation of a function during testing of a module
- Add appropriate #include statements to a module newly added to a project

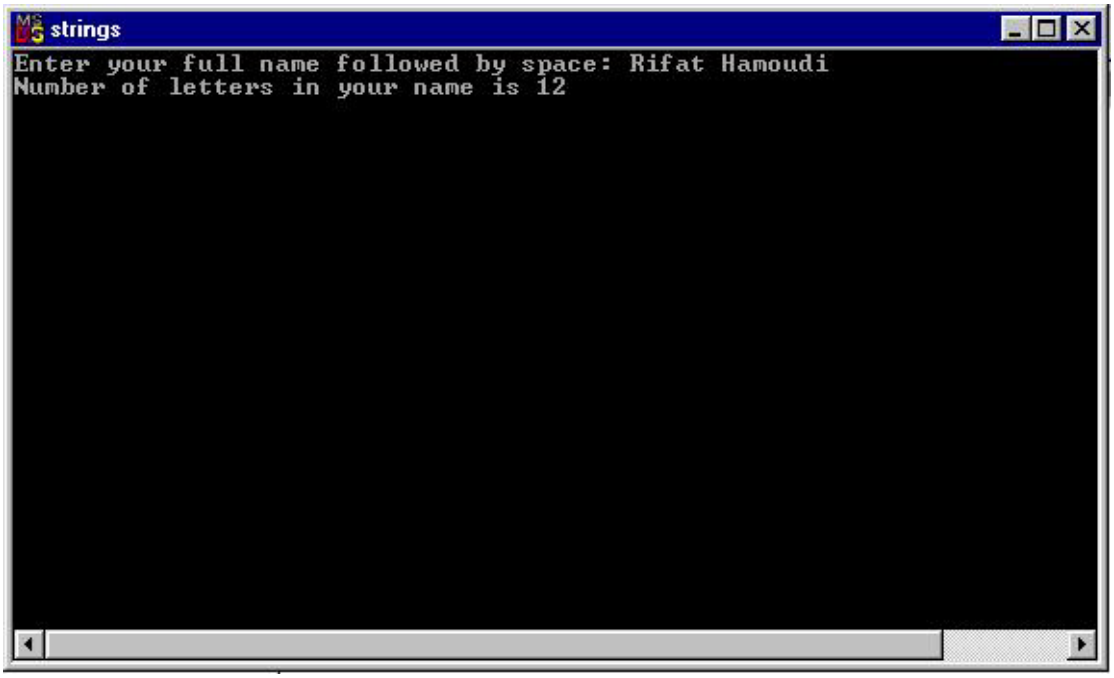
- With guidance, use the concepts of a class and object in the solution of a problem
- Follow the course convention of assigning data to the private section of a class definition and methods to the public section
- Use the dot notation for class or structure members
- Use the double colon notation appropriately when defining methods of a class
- Use statements usch as #define number 6 to cause the compiler to replace number with 6 whenever number is used
- Use and understand the terms : object, class, method of a class

Block 2 Unit 4

- Write designs that require data files to be open for reading and open for writing, and incorporate such files into data table
- Learn to implement your own class the methods

Exam questions

1) Design and implement a C++ program that counts the number of letters in a user's name.
An example output is as shown below :



Trace the program using Borland C++ Builder provided with this course and put a watch on the loop counter and any other variable.

2) This question is to do with preparation for writing a proper Database Application using Object Oriented C++ :

(a) Write a function to convert a character to uppercase
(b) The company want to create a database to keep track of its employees name, address and salary. Write a class with appropriate method definitions any other necessary variables to capture what the company wants. Your methods should be one for initialisation, one for adding an employee and one for displaying the employee details
(c) Sketch design for files needed to write the database application and what should go in each file

3) Write a database application for the company to add and display its employees. For this tutorial assume the maximum number of employees is 2 and just write data to RAM so don't worry about data to file. Add error check to ensure that user choice is always processed. Also use the class definition in Question 2(b) to record and display the employee details. Test your program to make sure that it works.

Answer to Question 1

The design

Top level design

4 read in user name
5 count the letters in the user's name
6 write out the letters in the user's name

Final design using stepwise refinement

1.3 write out "Enter your full name followed by space: "
1.4 read in Line
2.2.1 Index <- 1
2.2.2 LetterCount <- 0
2.2.3 PreviousLetter <- ' '
2.3 **loop while** Index <= Length(Line)
2.3.3 **if** PreviousLetter != ' ' **then**
2.3.2 LetterCount <- LetterCount + 1
2.3.3 **ifend**
2.3.4 PreviousLetter <- Line[Index]
2.3.5 Index <- Index + 1
2.4 **loopend**
3.1 write out "Number of letters in your name is", LetterCount

The C++ source code

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 3
        Title   :   Count letters in name program with semantic error
```

```
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/



#include "MT262io.h"  // modules defined in it e.g. ReadInt
#pragma hdrstop   //tells Builder how to deal with library files

//---------------------------------
// Project|Add to project -> MT262io.lib
USELIB("MT262io.lib");
//---------------------------------------------------------------------------
-
#pragma argsused // to avoid the prodution of warning messages
                 // relating to argc and **argv

int main(int argc, char **argv)
{
    AnsiString Line;
    int Index;
    int LetterCount;
    char PreviousLetter;

    Line = ReadStringPr("Enter your full name followed by space: ");
    Index = 1;
    LetterCount = 0;
    PreviousLetter = ' ';

    while (Index <= Length(Line))
    {
        if (PreviousLetter != ' ')
            LetterCount = LetterCount + 1;
        PreviousLetter = Line[Index];
        Index = Index + 1;
    }
    WriteIntPrCr("Number of letters in your name is", LetterCount);
    getchar();  // to keep the display on until you press Enter key
    return 0;
}
```

The above code has semantic error, detect the semantic error using the trace facility and correct the program?

How to trace a program in C++ Builder

1) Load your code
2) Click on Run then Step Over menu option or Click F8
3) Keep clicking F8 to trace through the program line by line
4) When you want to get into a function implementation click on Run then Trace into or F7, this will take you to the function or method implementation body
5) To add a watch highlight the variable and click on Run then Add Watch or Ctrl+F5, a separate window will appear which will contain the variable you want to watch and its current value, as you trace through the program the value changes accordingly

Answer to Question 2

(a)

```
char Tut3_uppercase(char selection)
{
```

```
    if ((selection >= 'a') && (selection <= 'z'))
        selection = selection - 32;

    return selection;
}
```

(b)

```
#define MaxEmp 2

class CompanyType
{
  private:

  struct EmployeeType
  {
    int Salary;
    AnsiString  Address;
    AnsiString  FullName;
  };
  EmployeeType  Employee[MaxEmp];


  public:

  void Init(void);
  AnsiString AddEmp(AnsiString EmpName, AnsiString EmpAddress, int
EmpSalary);
  void DispEmp(void);
};
```

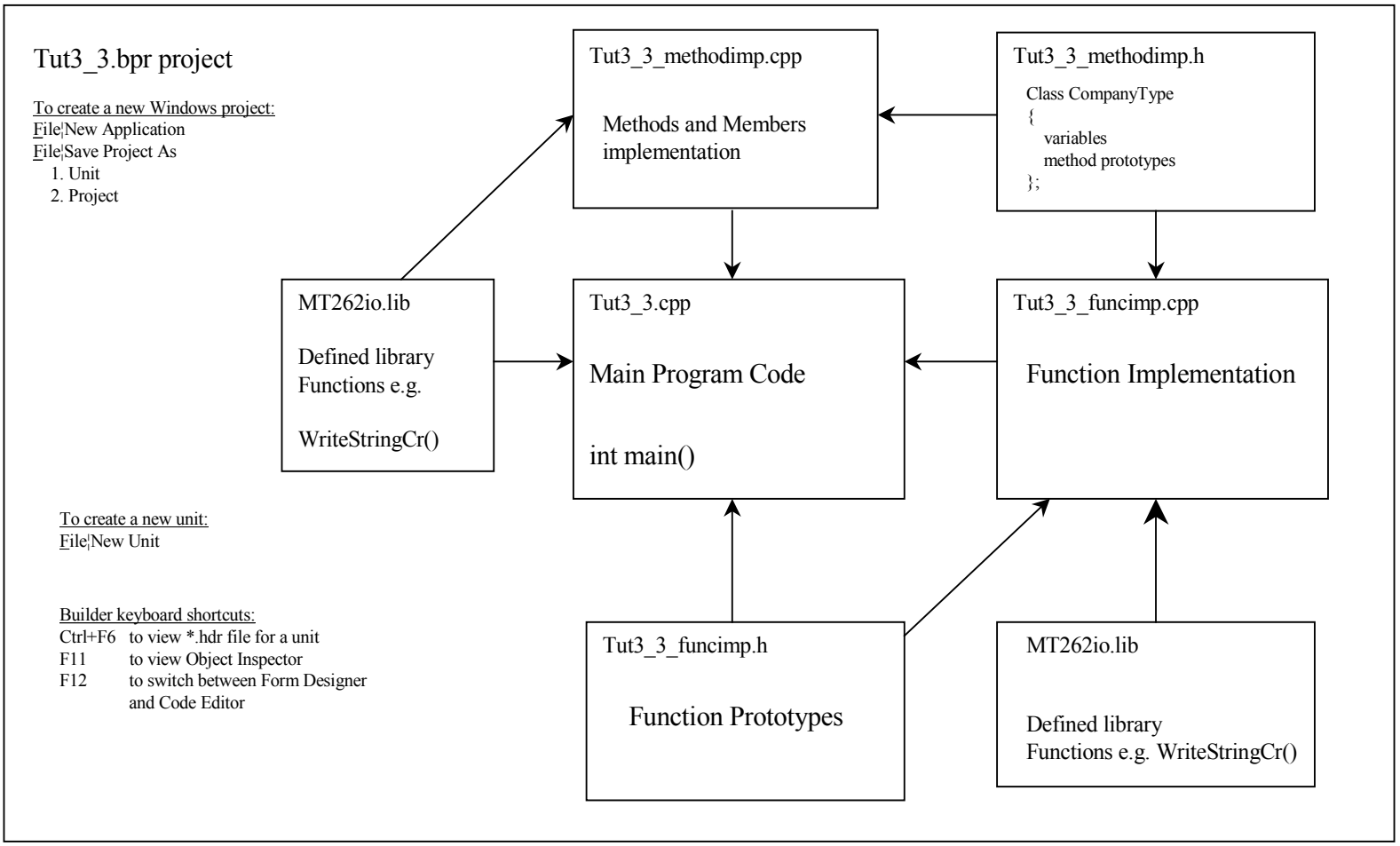
(c)

The files can be divided into 5 files as follows :

Tut3_3.cpp : Main file which contains the main driving code.
Tut3_3_methodimp.cpp : File containing the implementation of the methods in the Company class
Tut3_3_methodimp.h : contains the class definition used by the database
Tut3_3_funcimp.cpp : contains the implementation of functions used by the methods in the Company class and the main driver
Tut3_3_funcimp.h : contains function prototypes (definition)

Answer to Question 3

C++ Builder Project Tut3_3.bpr – Relationships between components



Tut3_3.cpp

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 3_3
        Title   :   Company Database Application Program
        Author  :   Rifat Hamoudi
        Version :   0.1
        Date    :
*/

// Main file for Tutorial 3 database application

#include <condefs.h>
#pragma hdrstop
#include <conio.h>
#include "MT262io.h"
#include "tut3_3_funcimp.h"


//---------------------------------------------------------------------
--------
USELIB("Mt262io.lib");
USEUNIT("tut3_3_funcimp.cpp");
USEUNIT("tut3_3_methodimp.cpp");
//---------------------------------------------------------------------
--------
#pragma argsused

int main(int argc, char **argv)
{
    char Choice;
    bool Done;

     // Insert a line to initialise Done correctly.
     Done = false;

     // Initialise the datastructure for employees
     CompanyInitialise();

     // Insert an appropriate condition for the loop.
     while (!Done) // main loop
     {
          // set up menu of choices
          clrscr();
          WriteStringCr("==== Company Staff Database ====");
          WriteStringCr(" ");
          WriteStringCr("Written by : Rifat for Tutorial 3");
          WriteStringCr(" ");
          WriteStringCr("Select Option");
          WriteStringCr("");// blank line as spacer
          WriteStringCr("A: add new employee");
          WriteStringCr("D: display current employees");
          WriteStringCr("Q: quit");
          // Insert a line to read a character into Choice from the
keyboard.
          Choice = ReadCharPr("Enter A, D or Q : ");
          // Complete the switch statement so that 'A' and 'D'
choices do nothing,
          // but 'Q' does stop the loop.
```

```
          Choice = Tut3_uppercase(Choice);

          switch (Choice)
          {
                case 'A' :  AddNewEmp();
                            break;

                case 'D' :  DisplayEmployees();
                            break;

            // final processing to be added to 'q' case later
                case 'Q' :  Done = true;
                            break;
          }
      } //end of main loop

    return 0;
}
```

tut3_3_methodimp.cpp

```
//---------------------------------------------------------------------------
// Methods implementation

#include <vcl\vcl.h>
#include <fstream.h>
#pragma hdrstop
#include "MT262io.h"
#include "tut3_3_methodimp.h"

//---------------------------------------------------------------------------
void CompanyType::Init(void)
{
    int index;

    // Initialise all Employees data structure

    for (index = 0; index <= MaxEmp; index = index + 1)
    {
        Employee[index].FullName = " ";
        Employee[index].Address = " ";
        Employee[index].Salary = 0;
    }
}

AnsiString CompanyType::AddEmp(AnsiString EmpName, AnsiString
EmpAddress, int EmpSalary)
{
    AnsiString Message;
    Employee[0].FullName = EmpName;
    Employee[0].Address = EmpAddress;
    Employee[0].Salary = EmpSalary;

    Message = "Employee is succesfully added to database.\n";
    return Message;
}


void CompanyType::DispEmp(void)
{
```

```
    int index;
    index = 0;

    while (Employee[index].Salary != 0)
    {
        WriteStringCr("");
        WriteString("Employee Number : ");
        WriteIntCr(index+1);
        WriteString("Full Name : ");
        WriteString(Employee[index].FullName);
        WriteString(" Address : ");
        WriteString(Employee[index].Address);
        WriteString(" Salary : ");
        WriteStringCr(Employee[index].Salary);
        index = index + 1;
    }

    WriteStringCr("Press Enter to continue");
    getchar();
}
```

tut3_3_methodimp.h

```
// Class implementation

#define MaxEmp 2

class CompanyType
{
  private:

  struct EmployeeType
  {
    int Salary;
    AnsiString  Address;
    AnsiString  FullName;
  };
  EmployeeType  Employee[MaxEmp];


  public:

  void Init(void);
  AnsiString AddEmp(AnsiString EmpName, AnsiString EmpAddress, int
EmpSalary);
  void DispEmp(void);
};
```

Tut3_3_funcimp.cpp

```
//---------------------------------------------------------------------
// Function implementation

#include <vcl\vcl.h>
#pragma hdrstop
#include "MT262io.h"
#include "tut3_3_funcimp.h"
#include "tut3_3_methodimp.h"
//---------------------------------------------------------------------
CompanyType Company;

char Tut3_uppercase(char selection)
{
    if ((selection >= 'a') && (selection <= 'z'))
        selection = selection - 32;

    return selection;
}

void CompanyInitialise(void)
{
      Company.Init();
}

void AddNewEmp(void)
{
    AnsiString EmpName;
    AnsiString EmpAddress;
    int EmpSalary;

    EmpName = " ";
    EmpAddress = " ";
    EmpSalary = 0;

    clrscr();
    EmpName = ReadStringPr("Enter Employee Name: ");
    EmpAddress = ReadStringPr("Enter Employee Address: ");
    EmpSalary = ReadIntPr("Enter Employee Salary: ");

    WriteString(Company.AddEmp(EmpName, EmpAddress, EmpSalary));

}

void DisplayEmployees(void)
{
      Company.DispEmp();
}
```

tut3_3_funcimp.h

```
// Function prototypes

char Tut3_uppercase(char option);
void CompanyInitialise(void);
void AddNewEmp(void);
void DisplayEmployees(void);
```

Block 3 Unit 1

- Open and save a new application, naming the unit and the project files appropriately
- Locate and use the main features of Builder's integrated development environment
- Appreciate that when designing a form, a class and one instance of that class are created by Builder in the unit code file
- Drop visual components such as buttons, labels, edit boxes, memos and panels onto a form, resizing and relocating them as required
- Use the Object inspector's property page to set initial values of components
- Use the Object inspector's events page to create event handler templates and to enter appropriate code
- Be aware that double clicking on a component in the Form Designer brings up the event handler for the OnClick event of that component
- Write simple event handlers which cause visual changes at run-time in a program
- Use Builder's help system to access information about classes – in particular, their properties and methods
- Be aware of the hierarchy structure of classes in Builder's visual component library and how properties and methods of a component may be inherited from a class higher in the hierarchy
- Use right arrow notation to refer to properties and events of a class
- With guidance, design and implement an application, using the interface engine approach

Block 3 Unit 2

- Design and code visual interfaces to existing objects, setting initial values using Object inspector
- Design and code visual effects of and processing code for event handlers of button clicks and menu item selections, paying particular attention to set focus to disable and enable appropriate components
- Apply the MT262 ground rules for Windows programming given in the appendix
- Construct a simple main menu with several menus having drop down menus, making use of separators and programmer-defined shortcuts to menu items
- Be aware of the need for proper initialisation and savingof data making use of the event handlers OnCreate and OnDestroy
- Be aware of the roles of inheritance and wrapping in code generated by Builder when objects are adapted

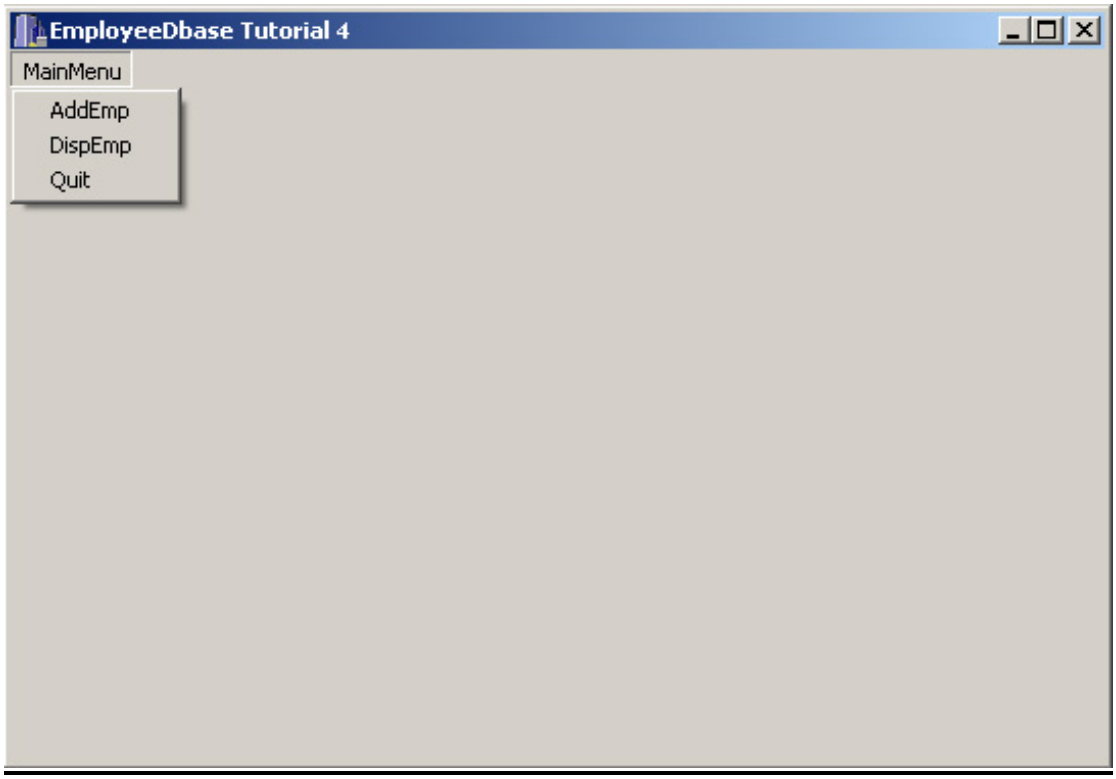
Block 3 Unit 3

- Follow instruction sot install supplied component on palette
- Explain the purpose of a prototype system, and create a simple one from a given specification
- Place appropriate code in the event handlers of Builder's events OnCreate, OnDestroy and OnTimer
- With guidance crate a viewing window with scroll bars where appropriate for a data file used in a program
- With guidance create a graphic displayfor data in a file

Exam questions

Question 1

Write a menu driven software for the employee database and implement an event handler for quitting the software. Save the software as EmpDB and don't worry about implementing event handlers for adding and displaying employees for this tutorial. The graphical output should look similar to the figure below :



Question 2

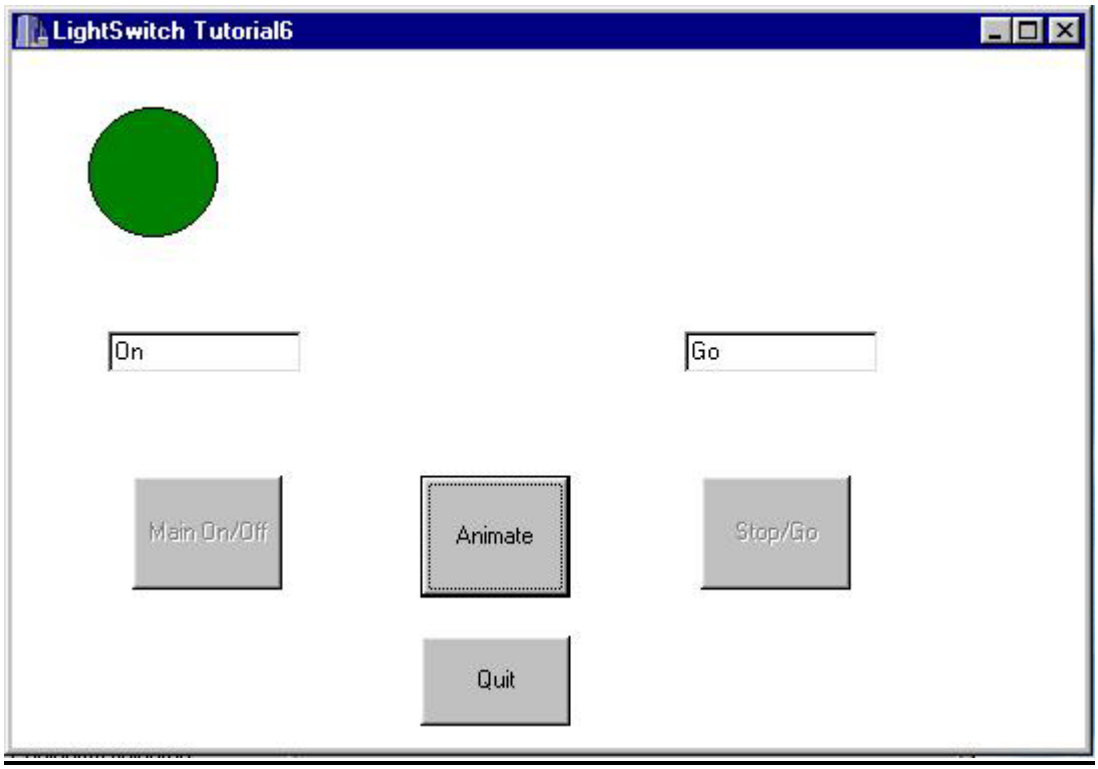
Write an event driven software the has the following specifications :

The software should have four buttons :

(a) Button called Main On/Off which when clicked will change the Form (background colour to black and display the word Off in the Edit box

(b) Button called Stop/Go which when clicked will change the colour of the circle to red and display the Stop in the Edit box
(c) Button called Animate which when clicked will move the Circle to the right of the form when it hits the boundary of the window it moves to the left, then right and so on until the user Quits the software
(d) Button called Quit which when clicked will quit the software

The graphical output should look similar to the figure below :



Answer to Question 1

To solve this do the following with C++ Builder :

1) Click on **File** then **New Application**
2) Save the project by clicking on **File** then **Save Project As**, call the .cpp file as EmpDBU and the project as EmpDB. **DO NOT** use the same name for both as this will crash the program and causes problems.
3) On the Form drag the menu button
4) Click on Form and change the **Caption** to EmployeeDatabase
5) Double click on the menu button on the Form and change the **Caption** to MainMenu
6) Press Enter and then change the Caption to **AddEmp** do the same for the rest
7) Click on Quit then on Events and double click on the right of **OnClick** this will take you to the event handler for what the program should do when the user presses the Quit button. Type Application->Terminate();
8) Save the program and run. This should give you your graphical database menu application.

Code for EmpDB software

EmpDB.cpp (generated by Builder)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("EmpDB.res");
USEFORM("EmpDBU.cpp", MainForm);
//---------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TMainForm), &MainForm);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
//---------------------------------------------------------------------
```

EmpDBU.cpp

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "EmpDBU.h"
//---------------------------------------------------------------------
--------
#pragma package(smart_init)
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------------------------------------------
--------
__fastcall TMainForm::TMainForm(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
--------

void __fastcall TMainForm::QuitMClick(TObject *Sender)
{
    Application->Terminate();
}
//---------------------------------------------------------------------
```

EmpDBU.h

```
//---------------------------------------------------------------------
#ifndef EmpDBUH
#define EmpDBUH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------
class TMainForm : public TForm
{
__published:      // IDE-managed Components
    TMainMenu *MainMenu1;
    TMenuItem *MainMenu2;
    TMenuItem *AddEmpM;
    TMenuItem *DispEmpM;
    TMenuItem *QuitM;
    void __fastcall QuitMClick(TObject *Sender);
private:    // User declarations
public:           // User declarations
    __fastcall TMainForm(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TMainForm *MainForm;
//---------------------------------------------------------------------
#endif
```

Answer to Question 2

To solve this do the following with C++ Builder :

1) Click on **File** then **New Application**
2) Save the project by clicking on **File** then **Save Project As**, call the .cpp file as lightU and the project as light. **DO NOT** use the same name for both as this will crash the program and causes problems.
3) On the Form drag the button labelled OK
4) Click on Form and change the **Caption** according to specification
5) Double click on the relevant button on the Form and change the parameters according to specification
6) Click on relevant button in the component menu in object inspector then on events and double click on the right of **OnClick** this will take you to the event handler for what the program should do when the user presses the particular button. Type your event handler code there
7) Save the program and run. This should give you your graphical light switching and animation application.

Light code

Light.cpp

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("light.res");
USEFORM("lightU.cpp", LightForm);
//---------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TLightForm), &LightForm);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
//---------------------------------------------------------------------
```

LightU.cpp

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
```

```
#include "lightU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TLightForm *LightForm;
//---------------------------------------------------------------------
__fastcall TLightForm::TLightForm(TComponent* Owner)
    : TForm(Owner)
{
      XMove = 5;
}
//---------------------------------------------------------------------
void __fastcall TLightForm::SwitchClick(TObject *Sender)
{
    if (Clock->Enabled)
        Clock->Enabled = false;

    if (LightForm->Color == clWhite)
    {
        LightForm->Color = clBlack;
        Status->Text = "Off";
    }
    else
    {
        LightForm->Color = clWhite;
        Status->Text = "On";
    }
}
//---------------------------------------------------------------------
void __fastcall TLightForm::TrafficLightClick(TObject *Sender)
{
    if (Clock->Enabled)
        Clock->Enabled = false;

    if (Circle->Brush->Color == clGreen)
    {
        Circle->Brush->Color = clRed;
        TrafficSign->Text = "Stop";
    }
    else
    {
        Circle->Brush->Color = clGreen;
        TrafficSign->Text = "Go";
    }
}

//---------------------------------------------------------------------
void __fastcall TLightForm::AnimateBClick(TObject *Sender)
{
    Clock->Enabled = true;
    Switch->Enabled = false;
    TrafficLight->Enabled = false;

    Circle->Brush->Color = clGreen;
    Circle->Left = Circle->Left + XMove;
    Clock->Interval = 50;

    if ((Circle->Left < 0) || (Circle->Left + Circle->Width) >
        (LightForm->ClientWidth - XMove))
        XMove = (-1) * XMove;}
```

```
}
//---------------------------------------------------------------------



void __fastcall TLightForm::QuitBClick(TObject *Sender)
{
     Application->Terminate();
}
//---------------------------------------------------------------------
```

LightU.h

```
//---------------------------------------------------------------------
#ifndef lightUH
#define lightUH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------
class TLightForm : public TForm
{
__published:      // IDE-managed Components
    TButton *Switch;
    TEdit *Status;
    TShape *Circle;
    TButton *TrafficLight;
    TEdit *TrafficSign;
    TTimer *Clock;
    TButton *AnimateB;
    TButton *QuitB;
    void __fastcall SwitchClick(TObject *Sender);
    void __fastcall TrafficLightClick(TObject *Sender);

    void __fastcall AnimateBClick(TObject *Sender);
    void __fastcall QuitBClick(TObject *Sender);
private:    // User declarations
public:           // User declarations
      int XMove;
    __fastcall TLightForm(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TLightForm *LightForm;
//---------------------------------------------------------------------
#endif
```

Block 4 Unit 1

- Write C++ code for drawing linear figures using the LineTo and MoveTo methods, with guidance where non-elementary mathematics is involved
- Use properties of Builder's Canvas to set colours, styles and line thickness of linear figures
- Make appropriate use of the function floor in code to convert a real number to the integer nearest to it
- With guidance, write code to create simple animations based on a linear figure, making use of a timer component in conjunction with the Repaint() method
- Use the Ellipse method in code to create circles and ellipses in particular locations on the canvas
- Explain the role of reflections in lines of symmetry of the circle in algorithms for drawing cirles

Block 4 Unit 2

- For a given display area and ranges of real values, convert given pixel values to real values in those ranges
- Create 'click and drag' zoom rectangle
- Other advanced graphics but I wouldn't worry about it in the exam