Tutorial 2 MT262

Tutor : Rifat Hamoudi
Staff No. : 00567451
Pager No. : 07669-801 509

I have put this tutorial on the web. This tutorial can be viewed and downloaded from http://www.users.totalise.co.uk/~rifat then selecting MT262 Tutorials then Tutorial 3.

1) In the first part of the question you are asked to write a C++ program from a given final design. The purpose of the program is as follows. A line of English text is to be entered from the keyboard. Part of the line will be contained in brackets () and only one pair of brackets is present. The program is to write out the content of the brackets. For example if the input line is

This is a sentence (with some brackets) for testing.

The output will be

with some brackets

A final design for the program is as follows :

```
1.1     write out “Enter your line of text containing one pair of brackets: “
1.2     read in Sentence
1.3     Index <- 1
2.1     loop while Sentence[Index] ≠ ‘(‘
2.2             Index <- Index + 1
2.3     loopend
2.4     Index <- Index + 1
3.1     loop while Sentence[Index] ≠ ‘)’
3.2             write out Sentence[Index]
3.3             Index <- Index + 1
3.4     loopend
```

a) Write a C++ program as a console application to implement this design exactly as it is. What would data table be like?

b) Design and implement a program complimentary to part (a). This time the brackets and their contents are to be removed.

2) This question concerns a program to detect whether a sentence entered by the user contains a specific letter.

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
5.1  if Found = true then
5.2    write out "Yes, the letter is found in the sentence."
5.3  else
5.4    write out "No, the letter is not found in the sentence."
6    ifend
```

```
/*
        Unit     :   MT262
        Tutorial:   Tutorial 2
        Title    :   Program to search for a letter in a sentence
        Author   :   Rifat Hamoudi
        Version :   0.1
        Date     :
*/

#include "MT262io.h"
#pragma hdrstop

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

#pragma argsused

int main(int argc, char **argv)
{
    AnsiString Line
    char Letter;
    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;
}
```

a) Identify all the syntax errors in the program above and correct them

b) There is a semnatic error in the program above. Say what the semantic error is and how would you correct it

c) Write a final correct design for this program and annotate it to explain key features

3) This question relates to functions which will be used in a later tutorial to build a database program.

a) Write a function which takes a string and converts all its characters to uppercase.

b) Write a function that compares two arrays

c) Write a function which finds the larger of two integers

d) Write a function which takes an integer and tests whether its positive or negative

4) A software company keep in the form of a table information about their employees. For each employee ther are to be five pieces of information :

- Name
- Address
- Project Name
- Project Leader
- Salary

Name, Address, Project Name and Project Leader should all be of type AnsiString and salary should be an integer ranging from 0 (note discrepency with real life here) and 500,000. There are maximum of 20 employees in the company.

a) Give C++ declaration of a varaible employee to hold this table of information as an array of records
b) Initialise the records to blank for characters and 0 for integer
c) Write employee class and with method called Initialise. Don’t worry about the implementation of Initialise.

Answer to Question 1

(a) **The C++ code**

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 2
        Title   :   To display words between brackets
        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 Sentence;
      int Index;

      Sentence = ReadStringPr(“Enter your line of text containing one
pair of brackets: “);

      Index = 1;
      while (Sentence[Index] != ‘(‘)
            Index = Index + 1;
      Index = Index + 1;

      while (Sentence[Index] != ‘)’)
      {
            WriteChar(Sentence[Index]);
            Index = Index + 1;
      }

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

Data table would contain :

AnsiString Sentence
Integer Index   do the table yourself as exercise

(b) Use the same two variables in (a), so a design similar to the following would give full marks :

```
1.1  write out “Enter your line of text containing one pair of brackets: “
1.2  read in Sentence
1.3  Index <- 1
2.1  **loop while** Sentence[Index] ≠ ‘(‘
2.2          write out Sentence[Index]
2.1          Index <- Index + 1
2.2  **loopend**
3.1  **loop while** Sentence[Index] ≠ ‘)’
3.2          Index <- Index + 1
3.3  **loopend**
4.1  **loop while** Index < Length(Sentence)
4.2          Index <- Index + 1
4.3          write out Sentence[Index]
4.4  **loopend**
```

The C++ code is as follows :

```
/*
        Unit    :   MT262
        Tutorial:   Tutorial 2
        Title   :   To display words outside brackets
        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 Sentence;
      int Index;

      Sentence = ReadStringPr(“Enter your line of text containing one
pair of brackets: ”);

      Index = 1;
      while (Sentence[Index] != ‘(‘)
      {
            WriteChar(Sentence[Index]);
            Index = Index + 1;
      }

      while (Sentence[Index] != ‘)’)
            Index = Index + 1;

      while (Index < Length(Sentence))
      {
            Index = Index + 1;
            WriteChar(Sentence[Index]);
      }

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

<u>Answer to Question 2</u>

(a) Syntax errors are :

1) AnsiString Line does not have semi colon, to correct type **AnsiString Line;**
2) Found varliable is not defined, to correct it type **bool Found;**
3) While Begin < End does not have bracket, to correct do **while (Begin < End)**
4) If (Found = true) assigns Found to true (i.e. 1) so Found will always be true. To correct do **if (Found == true)**

(b) The semantic error results from the fact that the index in Line keeps going so if a letter is found within the sentence and is not the last letter, then this letter will be missed and the program reports that the letter has not been found. A way to correct this is to set the Line index to the end once a letter is found in Line, this will break the while loop and reports the correct result to the user. To do this you need to add another step (which is 3.2.2 Begin = End) in the design as below:

```
3.1   if Line[Begin] = Letter then
3.2.1            Found <- true  // indicates that the letter was found
3.2.2            Begin = End
3.3   ifend
```

(c) **<u>Final correct design</u>**

```
1.1 write out "Enter your non-empty sentence : " // Program caters for any string
1.2  read in Line        // Get sentence
1.3  read in Letter      // Get letter to search for
1.4  Begin <- 1
1.5  End <- Length(Line)
1.6  Found <- true
2.1  loop while Begin < End
3.1      if Line[Begin] = Letter then
3.2.1            Found <- true
3.2.2            Begin = End             // Letter is found in sentence
3.3      else
3.4              Found <- false
3.5      ifend
4.1  Begin <- Begin + 1          // Move to the next letter
5    loopend
6.1  if Found = true then                // Display outcome
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
```

<u>Answer to Question 3</u>

(a) The function is as follows

```
AnsiString UpperCase (AnsiString Line)
{
        int Index;
        for (Index = 1; Index <= Length(Line); Index = Index + 1)
                if ((Line[Index] >= 'a') && (Line[Index] <= 'z'))
                        Line[Index] = Line[Index] – 32;
        Return(Line);
}
```

This algorithm ch = ch – 32 works because 'A' to 'Z' have ASCII numbers 65 to 90 and 'a' to 'z' have ASCII number 97 to 122. Thus difference between A and a is 32 and so on.

What are the formal parameters and prototype here?
Do a function to convert characters in a string to lowercase as exercise?
Why is the above code not efficient and what do you need to modify to make it more efficient?
Hint : Length(Line)

(b) The function is as follows

```
Bool ArrayCmp(AnsiString SourceArray, AnsiString DestArray)
{
        int Index;
        int maxlength;
        bool Equals;

        Index = 0;
        maxlength = Length(SourceArray);
        Equals = true;

        while ((Index < maxlength) && (Equals))
        {
                if (SourceArray[Index] != DestArray[Index])
                        Equals = false;

                Index = Index + 1;
        }

        return Equals;
}
```

Do a design as exercise.

(c) The function is as follows :

```
int maxinteger(int First, int Second)
{
        if (First > Second)
           return First;
        else
           return Second;
}
```

(d) The design is as follows :

```
bool   ispositive(int Number)
{
        if (Number > 0)
            return true;
        else
            return false;
}
```

<u>Answer to Question 4</u>

(a) You need to define this as array of records, first define the data structure of the record based on specification in the question then initialise the array of records.

```
struct Emptype
{
        AnsiString Name;
        AnsiString Address;
        AnsiString Project_Name;
        AnsiString Project_Leader;
        int Salary;
};

Emptype employee[20];
```

(b) Use a variable like Index to do that

```
int Index;

for (Index = 0; Index < 20; Index = Index + 1)
{
        employee[Index].Name = " ";
        employee[Index].Address = " ";
        employee[Index].Project_Name = " ";
        employee[Index].Project_Leader = " ";
        employee[Index].Salary = 0;
}
```

(c) Here you are converting data structure to class.

```
class employee
{
   private:
        AnsiString Name;
        AnsiString Address;
        AnsiString Project_Name;
        AnsiString Project_Leader;
        int Salary;
   public:
        void Initialise();
};
```