Teaching Information Technology

M. Shahzad Mughal

Contents

Preliminary Remarks

This study text "Teaching Information Technology" is slightly different from other study texts of the B. Ed. in Technical Education programme.

Firstly, we have organised the chapters of this study text according to the outline of the entire specialised area "Information Technology", which includes the content both for face-to-face and for self-learning phases. For a better understanding, you find the number of the chapter of the outline within brackets at the end of each chapter heading of this text.

Secondly we allocated the learning objectives to the respective chapters. We did this because of the large and comprehensive catalogue of learning objectives in order to facilitate the allocation of specific learning objectives to the referring chapter.

We hope to have simplified the understanding of the text by these measures.

Literature

- "Beginning Microsoft Visual BASIC 2010" by Thearon Willy and Bryan Newsome.
- "Professional Visual BASIC 2010 and .NET 4" by Bill Sheldon, Billy Hollis and others.
- Sams "Teach Yourself Visual Basic 2010 in 24 Hours".
- http://www.vbtutor.net/index.php/visual-basic-2010-tutorial/
- *Beginning* C: Fifth Edition, by Ivor Horton. APress, ISBN13: 978-1-4302-4881-1.
- C++ in a NutShell, A Language and Library Reference, O'Reilly.
- C++ How to Program: 8th Edition, by Dietel & Dietel.
- http://www.tutorialspoint.com/cplusplus
- http://www.cprogramming.com/tutorial
- "Beginning Database Design – from Novice to Professional", 2nd Ed. by Clare Churchul; APress.
- "Microsoft Access 2010 – Programmer's Reference", by Teresa Hennig, Rob Cooper, Geoffrey Griffith, Jerry Dennison; Wrox.
- "Microsoft Access 2010 All-in-One for Dummies", by Alison Barrows, Margaret Levine Young, Joseph C. Stockman.
- http://office.microsoft.com/en-001/access-help/getting-started-with-access-2010-HA010341722.aspx
- http://www.gcflearnfree.org/access2010

List of Figures

List of Tables

Unit Programming (C/C++)

1 Expressions and Operators (Chapter 2)

Learning Objectives

The main objective of the course is to teach basic computer programming concepts, skills and apply them to computer–based problem-solving approaches and methods. After going through this module you will be able to:

- Use of computers as a powerful tool in solving the common problems within various areas of our society.
- Understand the generic rules and principles of computer programming which can be applied directly to common situations.
- Comprehend the procedures, functions, algorithms, and processes of a computer programming language.
- Apply and Use the knowledge of both the algorithmic functions and of computer programming in definite application settings.
- Demonstrate and explain the use of computer programming concepts and knowledge in the production of programs which have the application to diverse problem settings.
- Understand the fundamentals of programming such as variables, constants, compiler, interpreter, conditional and iterative execution, methods and functions, etc.

1.1 Introduction

Hi Students! Hope, you are doing well in your programming course. You have learnt basics of computer programming. We are going to discuss about expressions and operators in programming.

As in our daily life we come across many problems and calculations in which we have to use some expressions and operators. These operators may be Arithmetic or may belong to some other category of evaluation and calculation.

Have you ever thought that what the expressions are? Expressions are just like sentences. Just as words in a sentence, an expression is a combination of literals/ constants, identifiers/names, operators and other symbols. Generally an expression represents a calculation. In a computer program, an expression evaluates a values as per given operations and data. Every expression results in a value and type of that value. Type of that value is also considered as the type of that expression.

1.2 Expressions and Operators

In C/C++, expressions consist of two parts:

- LValues
- RValues

In an expression, LValue is an identifier or reference of an object where as the RValue is a value. LValue and RValue are the foundations of expression in C/C++.

LValue may consists of an identifier or variable name, a pointer, an object reference or an expression or array subscript or function call which results in a reference/ pointer to an object. **RValue** is an expression or may be direct value, it may the result of some arithmetic operations, some function calls. The constants cannot be used as LValue but can be used as RValue.

1.3 Assignment Statement

Dear students! In our daily life we perform many calculations. Some of these calculations are formal and other may be non-formal. From non-formal, I mean that the calculations which we just quickly perform to deal our daily activities. These may be price calculations or rate calculations or total of some amounts, etc. In all these calculations we ultimately calculate the value of some object, piece. It may be as simple as adding to values or as complex to calculate the area under a curve.

Consider the following example:

B = 45 / 5

In this statement we calculate the value of 45/5 and assign the result to a variable B. This type of statements are known as assignment statements. Assignment statement is generally used to assign a value to an identifier or a variable. An assignment statement is written as:

Identifier = expression

In a C/C++ program the assignment statements may be of the following format:

```
{
    A = 10;
    B = 70
    C = (A + B) / 2;
}
```

These are three assignment statements, in first statement value 10 is assigned to A, then value 70 is given to B and finally the average of A & B is calculated and result is assigned to C.

Assignment operator (=) is used in assignment statement. On the left-hand-side of = is always a valid LValue and on the right-hand-side is a valid RValue. RValue is any valid expression which results in a valid calculation and value.

1.4 Arithmetic Operators and Operator Precedence

We have discussed the Expressions and Assignment statement till now. When we are going to write an arithmetic expression then we use arithmetic operators to perform calculations or arithmetic operations. These arithmetic operators include addition operator (+), subtraction operator (-), multiplication operator (*), etc. Table 1 shows the Arithmetic operators available in C/C++ programming language. Each operator is illustrated by an example. We use two variable X and Y where X=20 and Y=5;

Operator	Explanation	Example
+	Addition	X + Y will result in 25
-	Subtraction	X – Y will result in 15
*	Multiplication	X * Y will result in 100
/	Division	X / Y will result in 4
%	Modulus	X % Y will result in 0 Y % X will result in 5
++	Increment	X++ will result in 21
--	Decrement	X -- will result in 19

Table 1 Arithmetic Operators in C/C++

Arithmetic Expressions are written using these arithmetic operators. Arithmetic expressions are written in a straight line. Now it is the question that if an Expression is written using these arithmetic operators then in which sequence these operators will perform the arithmetic operations. This phenomenon is known as operator precedence.

Example:

If we have an expression x + y * z then what will be the sequence of operation. Either x + Y will be calculated first and the result will be multiplied with z or y * z will be calculated first and the x will be added in the result. In normal Mathematics multiplication has the higher precedence and hence y * z will be performed first and then x will be added in the result.

C/C++ also follows the same precedence rule. Table 2 shows the precedence of arithmetic operators in descending order.(lower number shows higher precedence i.e. 1 has highest precedence and 2 next lower and so on).

Precedence	Operator	Category
1	+, -, ++, --	Unary operators
2	*, / , %	Multiplicative
3	+, -	Additive
4	&	Bitwise AND
5	^	Bitwise XOR
6	|	Bitwise OR

Table 2 Arithmetic Operators Precedence

Activity 1

Write a C/C++ program which converts a Celsius temperature into Fahrenheit value. It should take value of Celsius temperature from the user then perform calculation and display the result in Fahrenheit.

1.5 Comparisons

Computer is considered to be an intelligent device. It is intelligent because it can be programmed. It can analyze the situations on the basis of some conditions. Depending upon the conditions it can make decisions. The conditions are evaluated on the basis of comparisons of two values. These comparisons are made using conditional statements. These conditional statements are also like expressions but the difference is that a conditional statement can yield only one of the two possible results i.e. TRUE or FALSE. The conditional statements are written using relational operators. Each relational operator takes two operands and compares their values and give the result as TRUE or FALSE. Table 3 shows the relational operators being using used by C/C++ to make comparisons.

Operator	Symbol	Form	Operation
Greater than	>	x > y	TRUE if x is greater than y, FALSE otherwise
Less than	<	x < y	TRUE if x is less than y, FALSE otherwise
Greater than or equals	>=	x >= y	TRUE if x is greater than or equal to y, FALSE otherwise
Less than or equals	<=	x <= y	TRUE if x is less than or equal to y, FALSE otherwise
Equality	==	x == y	TRUE if x equals y, FALSE otherwise
Inequality	!=	x != y	TRUE if x does not equal y, FALSE otherwise

Table 3 Relational Operators in C/C++[1]

Comparisons can be made between numeric values, and literals as well as char type variables and literals. In C/C++, FALSE is represented by 0 (zero) and any non-zero values is treated as TRUE.

Activity 2

Write a C/C++ program. It should take input from user. The input should be an integer. Then this program should display that whether this integer is Even or Odd as well as if it is a Prime integer or not.

1.6 Logical Connectors

Previously we have discussed about conditions in C/C++ programming. We have discussed about single conditions till now. There are many situations when we have to combine more than one condition. To combine the results of more than one condition we use logical connectors. These logical connectors are also call logical operators. Logical operators are also binary operators. The operands of these operators are logical values or logical expressions. The logical connectors/operators are described in the Table 4.

1 http://www.learncpp.com/cpp-tutorial/35-relational-operators-comparisons/

Operator	Description
&&	It is known as Logical AND Operator. It returns a TRUE value if the both of its operands have TRUE or non-zero values. If one of the operands have FALSE or zero value then the result will be FALSE.
||	It is known as Logical OR Operator. It returns a TRUE value if any one of operands has TRUE or non-zero value. If both of the operands have FALSE or zero values then the result will be FALSE.
!	It is known as Logical NOT Operator. It is a urinary operator. It simply reverses the value of the operand. If operand has TRUE value then it will FALSE and vice versa.

Table 4 Logical Operators in C/C++

To exactly understand the impact of Logical Operators on the conditions, we can use Table 5. The following is a Truth Table (Table 5) showing the impact of Logical Operators/Connectors.

A Truth Table with two Logical Expressions P and Q				
P	Q	P && Q	P || Q	! P
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

Table 5 Truth Table for Logical Connectors/Operators

Using this truth table we can understand and plan that how we can formulate our conditions in logical expressions and decision making situations especially where multiple conditions are to be considered.

2 Control Statements (Chapter 3)

Learning Objectives

On the successful completion of this chapter, the students will be able:

- To solve the problems using computer programming with control statements.
- To design solutions by understanding the problem through a top-down approach.
- To use if, if … else, switch, loop, break and continue statements in their programs.
- To differentiate between the if, if … else and switch statements.
- To have skill about the effective use of For, while, do … while, loops for problem solving techniques.
- To use the conditions, relational and logical operators to control the loops and branching in C/C++ programs.

Preface

We have gone through the basics of C/C++ programming in our previous sections. In all our daily tasks and even throughout the life we have to make different types of decisions. These decisions may be about purchasing something, adopting some education, accepting some job, or many other occasions like those. We have certain parameters in our mind according to which we decide. Similarly in the life-cycle of a computer programme there may be points where some decision has to be taken. So in this section we will discuss about this process.

2.1 Introduction

In the previous chapter we have discussed about expression, logical expressions, conditions, arithmetic operators and logical operators. In our daily life we often come across the situations where we have to make decisions. For example, if we are driving then at a signal we check whether signal is red or green to pass through the signal or stop there. When we have to make the decisions, then certainly we have some options to choose from. Computers are also machines which can make decisions. These decisions are made by computers depends upon the programming which programmer has done to make use of computers. To make decisions in programming we use control statements.

2.2 Control Statements

Computer programs are executed in a sequence and control statements can change this sequence depending upon some condition. There are two types of control statements:

- Branching statements
- If
- If … else
- Switch
- Repetition statements (loops)
- Counter-controlled loops (for)
- Conditions controlled loops (while, do … while)
- Continue and Break

2.3 Branching Statements

2.3.1 "The If else statement"

In C/C++ programs "if" statement is a single statement. It selects one of the two paths of execution of a program depending upon the result of some condition, i.e. if the result is true then it perform one task and then continue normal execution and if the condition results in false then it ignore the task within the if statement and simple continue the part after the if statement block. Whereas the "if … else" statement perform one task if the condition is true and other task if the statement is false and then continue the normal sequential execution.

The following flow-charts describe the flow of programs in these statements.

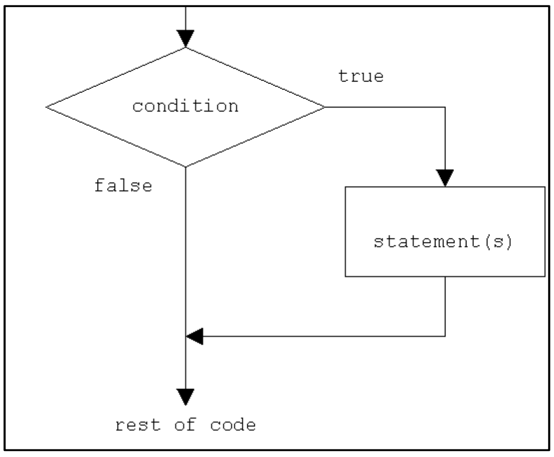


Figure 1 - Flow Chart for "if" statement[2]

[2] http://www.ustudy.in/sites/default/files/if.gif

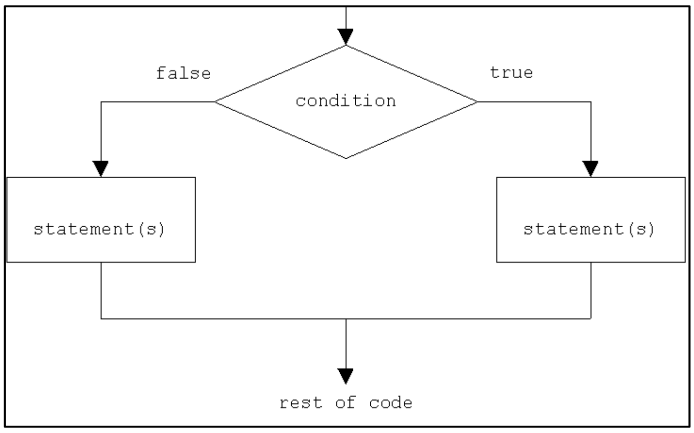


Figure 1 Flow Chart for "if ... else" statement[3]

If we have choice between more than two, which depends upon more than one conditions then we can use "if … elseif" statement. It is generally known as "if...elseif ladder". Figure 2 depicts the "if…elseif" ladder.

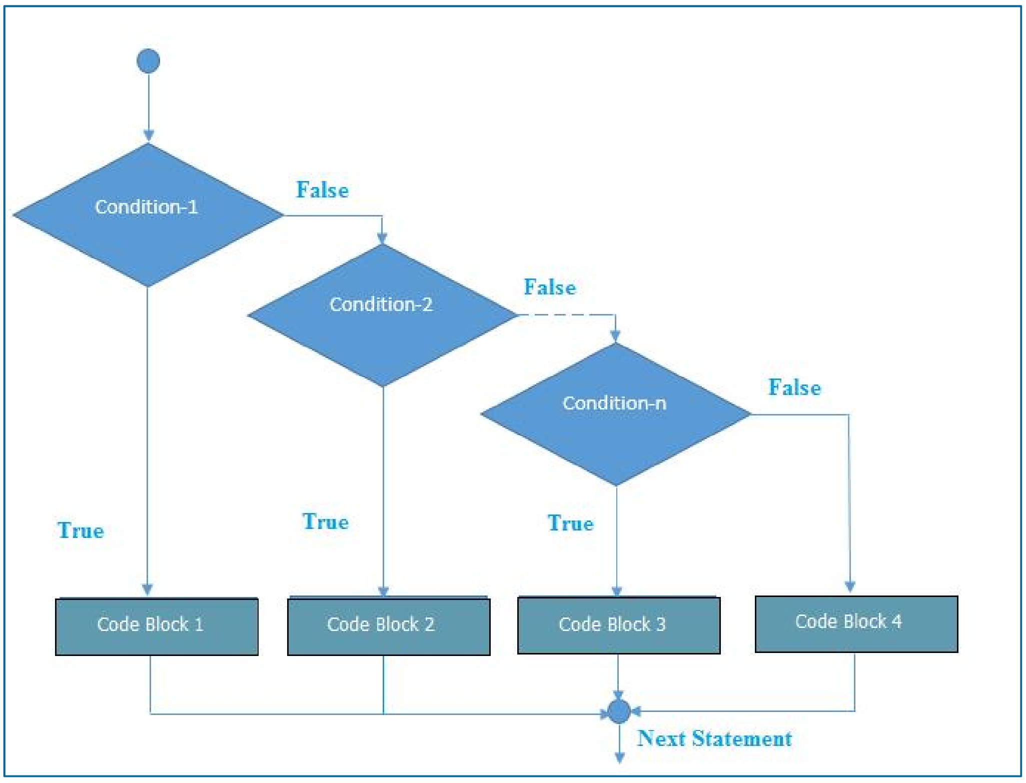


Figure 2 Flow Chart for "if … elseif …" statement4

This ladder helps us in solving the problems involving multiple conditions. Example of this type of problem is to calculate the Income-Tax which depends upon different tax rate on different level of income, or to calculate the letter grade of students depends upon the percentage marks of students.

[3] http://www.ustudy.in/sites/default/files/if_else.gif
[4] http://www.javatpoint.com/cpages/images/elseifladder.png

Syntax of "if", "if…else" and "if…elseif" are as follows:

If	If…else	If…elseif

●
●
●
if (condition)
{
Code Block – TRUE
}
●
●
●
	●
	●
	●
	if (condition)
	{
	Code Block – TRUE
	}
	else
	{
	Code Block – ELSE
	}
	●
	●
	●
		●
		●
		●
		if (condition-1)
		{
		Code Block – TRUE-1
		}
		elseif(condition-2)
		{
		Code Block – TRUE-2
		}
		.
		.
		.
		else
		{
		Code Block – ELSE
		}
		●
		●
		●

Activity 3

Write down a program in C/C++ which follows the following instructions:

- Take input from the user about a students' :
  Roll No
  Gender
  Marks of any 5 subjects
- Then calculate the average marks and percentage marks
- Then display the information as (if Gender if F - female):
  Roll No. xxxx is a Girl
  Average Marks: nn
  Percentage Marks: nn%
  Letter Grade: A ….
  (Letter grades could be from A to F, depending upon percentage marks from 100 down to 40)

Compile the program and execute it.

Test the program against different type of inputs.

2.3.2 The switch statement

The switch statement is an alternate of the "if…elseif…else" structure. We have discussed about it in the class. You have learned that how we can use a switch statement in-place of an "if…elseif…else" structure.

2.4 Repetition statements (loops)

In our daily life problems which we try to solve using computer programming, many time involves a situation when it is required to repeat a piece of code many times. For example, if we have to display a message or a string many times, taking input from a user until he enters a specific text or value, etc. This is the situation where loops enter into our programming life. To be able to perform a block of code repeatedly is a very important and basic skill in programming. Many complex tasks can be solved by just repeating a simple block of code. All the programming languages provide the statements to performing looping tasks possible. By having a Label in the code and using a GoTo statement can also help to create a loop in a program. Figure 3 shows a general concept of a loop.

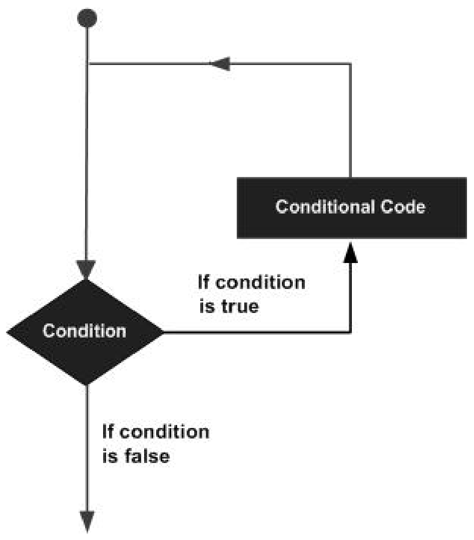


Figure 3 Figure showing a loop construct

C/C++ provides different looping constructs. One of these is counter-controlled and others are conditions controlled. Counter controlled loop is "for" loop and condition-controlled loops are "while" and "do…while". Generally "for" loop continue till a predetermined number of times where as condition-controlled continue until a certain condition remains true. A loop becomes an **infinite loop** is its condition always remains true, sometimes it may be a programming error.

2.4.1 Counter-controlled loops (The "for" Loop)

For loop is a counter-controlled loop. The syntax of a "for" loop is C/C++ is as follows:

"for" loop

for (variable initialization; condition; variable update)
{
Code to execute while the condition is true
}

Figure 4 explains the components of the header of "for" loop statement.

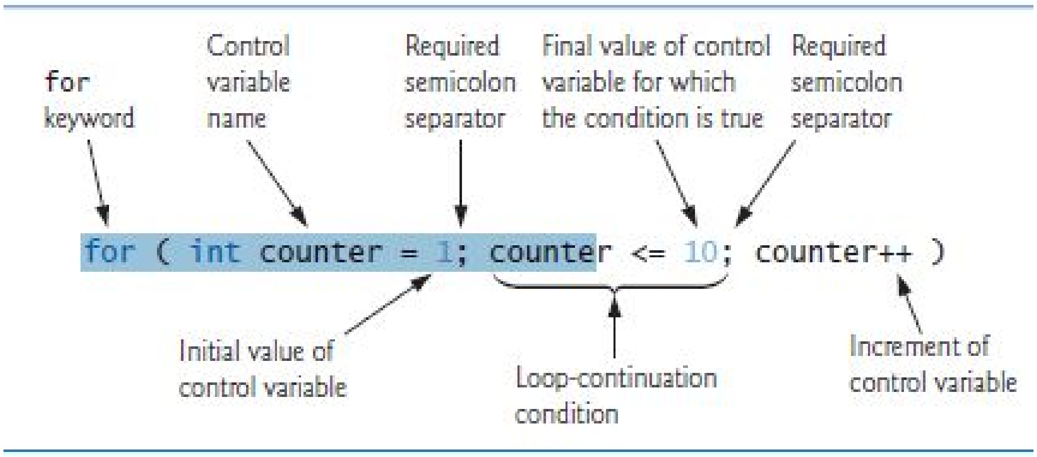

for keyword
Control variable name
Required semicolon separator
Final value of control variable for which the condition is true
Required semicolon separator
for (int counter = 1; counter <= 10; counter++)
Initial value of control variable
Loop-continuation condition
Increment of control variable

Figure 4 Components of "for" statement header[5]

If we have implemented "for" loop in our code then it will be executed by following the steps given below:

- First of all the control variable is initialized.
- The value of control variable is tested against the loop-control condition.
- If the condition is true, then loop body starts executing.
- When the body is executed once, then the value of control variable is updated in the last statement of the header of loop statement.
- Then the value of the control variable is again tested in the loop-control condition and then if the condition results in false then loop is terminated, otherwise again body is executed. And so on …

In this way we can solve a problem which requires a counter controlled loop.

Activity 4

Write a program in C/C++ using "for" loop which follows the following instructions:

- Take input from the user :
  a start value **S**
  an end value **E**
  an integer **N**
- Then calculate execute the **for** loop from **S** to **E** and count the values which are divisible by **N**.
- Display the value of this count.

Compile the program and execute it.

Test the program against different type of inputs.

5 "C++ how to program", p:156

3 Recursive Functions (Chapter 4)

Learning Objectives

On the successful completion of this chapter, the students will be able:

- To breakdown a larger problem into smaller set of problems.
- To solve a problem in modular approach.
- To develop large programs modularly from the small code components known as functions.
- To write down their own small functions for re-usability in different programs.
- How to pass information to functions.
- How to receive results from functions.
- How to build, code, use and re-use functions from outside and from within the same function

Preface

In this chapter I will ask you to make your observation deep and keen about the happenings of daily life. Have a look around you! You may notice many events, many projects in progress around you. If we look closely, you will find that the large projects have been divided into small tasks and each small task is handed over to a single person or a small group to complete it. When all such sub-groups complete tasks assigned to them, then the large project is completed. In this way a project can be completed in an efficient and more manageable manner. The same approach is being in large computer programs and projects.

3.1 Introduction

The programs which solve the real-world problems are generally very large and complex. Then it is a better idea to divide these programs into small programs which are known as functions and then build the big programs using these small pieces. These small pieces of programs are more manageable and easy to debug, modify and maintain. C/C++ provides us with many built-in functions which help us a lot in building larger and variety of programs. In this chapter we will learn how to write, use and maintain functions.

3.2 Recursive Functions

After the discussion that how to write and function; scope of variables of functions and modification of function arguments, now we will discuss a special type of functions in C/C++. This is a type of functions which call themselves repeatedly to solve a complex problem. Generally the functions call one-another in a sequence or a hierarchical fashion but for some problems it is easy and more manageable to write functions which call themselves. These functions are called recursive functions. A recursive function is called recursive if the function calls itself directly or indirectly via other function.

To understand the concept of recursive functions we will discuss the phenomenon of recursions first. Recursion is a bit complex to understand. We will try to make it simple by giving examples of simple recursion problems. Generally a recursive function is designed to solve some simple problem and it has some known values for some simple cases of the problem. These known values are called base case. When a recursive function is called with a parameter, then if the parameter has this base case then a recursive function simply returns a result otherwise it calls itself for some simpler case. The new call of the function has a simpler case and performs again the same. When a function calls itself then it is known as recursive call. A recursive function must have to be coded very carefully, because if we fail to code a base case from where the function starts to return values, then there will be infinite recursive calls and soon the memory of the computer will be exhausted and system will become irresponsive and may be halted. A recursive function may have one or multiple calls to itself from within its body.

It is always possible to have both the iterative and recursive solution of a problem. But sometimes it is easy to have an iterative solution and for some problems it is easy to work with a recursive solution. To have good understanding of recursion and recursive functions now we will discuss few simple programs with recursive functions to perform some common mathematical calculations.

Iterative solution	Recursive solution

`int sum (int n)`
`{`
`        int nSum = 0;`
`        // the for loop to calcu-`
`late the`
`        sum from 1 to n`
`        for ( int i = 1; i <= n;`
`i++ )`
`        {`
`                nSum = nSum + i;`
`        }`
`        return nSum;`
`}`
	`int sum(int n)`
	`{`
	`        //Return 0 when n is 0`
	`        //this is the base case`
	`        if ( n <= 0 )`
	`                return 0;`
	`        else`
	`                return n + sum(n-`
	`1);`
	`}`

Program 1: Calculating the sum of first n integers. Creating a sum() function.

Iterative solution is almost clear to understand as we have worked with this type of programming. Now we will discuss that how the recursive solution will work. Let the function sum() is called with n=3, i.e. sum(3). Then Figure 5 describes the recursive function calls and return values from each call.

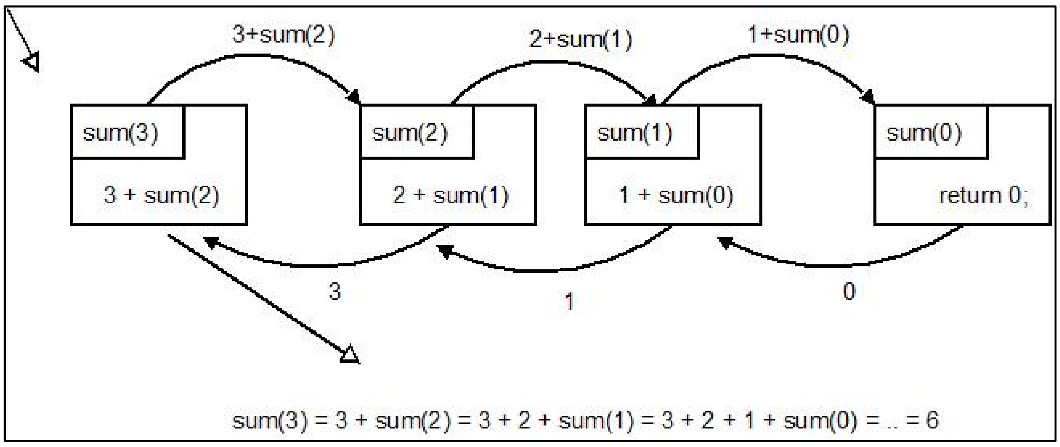


Figure 5 Recursive execution of sum() function.[6]

[6] http://talkbinary.com/wp-content/uploads/2009/11/summation21.jpg

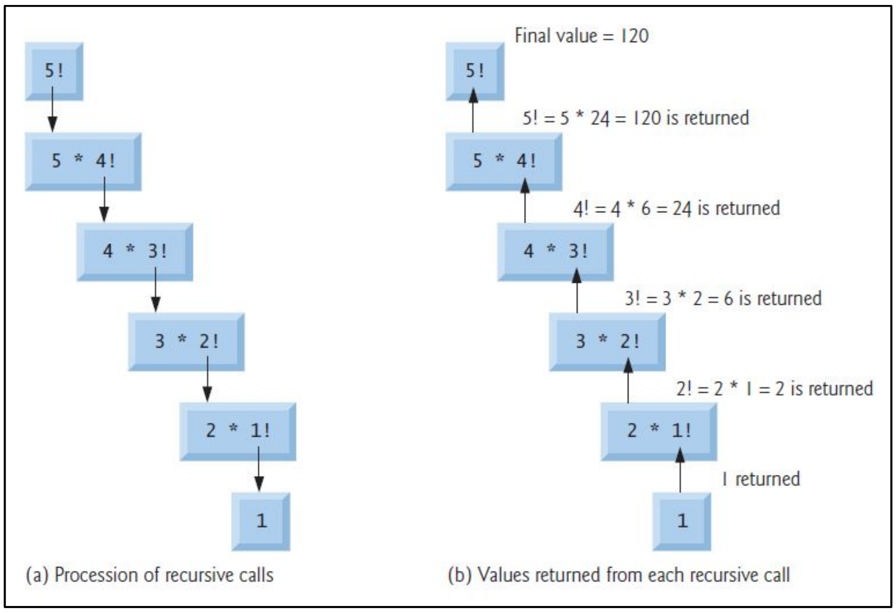
Initially the function sum() will be called with n=3, from its body a call will be generated with n=2, then from the body of this call next call will be generated with n=1, then from this call next call will be generated with n=0; now this is the base case, it will return a value 0, then from its previous call the value 1+0 will be returned, then back further and value 2+1 will be returned and from the very first call the value 3+3 will be returned to the calling code which will be the sum of first 3 integers.

Iterative solution	Recursive solution

`int fact (int n)`
`{`
`        int nFact = 1;`
`        // the for loop to calcu-`
`late the`
`        factorial of n`
`        for ( int i = 1; i <= n;`
`i++ )`
`        {`
`                nFact = nFact * i;`
`        }`
`        return nFact;`
`}`
	`int fact(int n)`
	`{`
	`        //Return 1 when n is 1`
	`        //this is the base case`
	`        if ( n == 1 )`
	`                return 1;`
	`        else`
	`                return n * fact(n-`
	`1);`
	`}`

Program 2: Calculating the factorial of a non-negative integer n. Creating a fact() function.

Iterative solution is very much clear. Now we will discuss the recursive version of the fact() function. Let the function fact () is called with n=5, i.e. fact(5). Then Figure 6 describes the recursive function calls and return values from each call.



5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!
1
(a) Procession of recursive calls
Final value = 120
5!
5! = 5 * 24 = 120 is returned
5 * 4!
4! = 4 * 6 = 24 is returned
4 * 3!
3! = 3 * 2 = 6 is returned
3 * 2!
2! = 2 * 1 = 2 is returned
2 * 1!
1 returned
1
(b) Values returned from each recursive call

Figure 6 Recursive execution of fact() function[7]

[7] "C++ how to program", p:240

In Figure 6 part (a) shows the steps in recursive calls with each decreasing value of n and the part (b) shows the returned values from each call and also shows the final result.

Activity 5

Write the C/C++ programs to implement both above discussed functions sum() and fact() in both types i.e. iterative and recursive. Then compile, run and test the programs with different values of n.

Task 1

Write C/C++ programs to Calculate:

- Highest Common Factor (HCF)
- Least Common Multiple (LCM)

Both in iterative and recursive format.

The program should take two integer values as input, ask the user what to calculate either HCF or LCM and then display the result accordingly.

4 Input and Output in C/C++ (Chapter 5)

Learning Objectives

On the successful completion of this chapter, the students will be able:

- To implement input techniques in C/C++ programming.
- To implement output techniques in C/C++ programming.
- To understand the input and output from different types of files.
- To open, close, modify the data files.
- To implement file processing in sequential-access and random-access manner.

Preface

Almost in every program, either it is large or small, it is required to take input from user either direct from keyboard or from a file stored on some storage media. After the completion of data processing, it is required to display the output/results. This output may be on a visual display, printed on a paper or stored as a file on storage. So to implement all these techniques we should be familiar with the tool and techniques provided by C/C++ to achieve these goals.

What you have learnt during face-to-face so far:

Chapter 5.1 Standard Input/Output :scanf, cin, printf, cout

These concepts have been discussed in the class and we have learnt about the uses of these standard input and output functions provided by C/C++.

Chapter 5.2 Other input/output:

Besides the standard input/output functions and techniques, C/C++ also provides some other functions for input and output. These include: getchar() and putchar() for single character input and output from the standard stream, printf() and scanf() for formatted input/output and gets() and puts() for string input and output purposes. We also have discussed about all these in the class.

4.1 Introduction

During the programming when we store the data in variables and arrays, then the data is stored temporarily in these data holding places. As soon as the life of program ends the data stored in these variables and arrays is also lost. If we need to store such data permanently then we have to use files and store data in files. Files are stored on secondary storage devices where data resides until we delete it intentionally. Keeping in view this scenario, file handling is an important in programming era. In the large programming application we come across the situations where we have to manipulate the data stored in files.

4.2 C/C++ File Handling – File Pointers:

C/C++ provides different programming functions and constructs which help us in creating and manipulating files. From the C/C++ point of view, a file is a sequence or stream of bytes. Each file starts with a beginning-of-file (BOF) marker, followed by none or some bytes (this some may be huge) of data, then and end-of-file (EOF) marks which denotes the ending of the file. When we open a file in C/C++, then a stream is associated with it, which act as a communication channel between the file and the program. In C/C++ program, this stream is handled by using a FILE type pointer. After the processing of the file is completed we close the file.

C/C++ provides two functions *fopen()* and *fclose()* to open and close the file streams. We discuss these one-by-one.

4.2.1 fopen() and fclose()

fopen() function is used to open a stream associated with a file in C/C++ program. Conceptually it opens an already existing file, but if file do not exists then it creates the new file. The syntax of *fopen()* is given as under:

```
FILE * fopen ( const char * filename, const char * mode );
```

This function opens a file whose name is specified by filename parameter, and the afterward operations can be performed by the FILE type pointer returned by this function. A NULL pointer is returned if the file cannot be opened due to any reason. The mode parameter mentioned that which operations on this stream can be performed and which cannot be performed.

The possible values of the mode parameter are given here:

- **"r"**: (read) this mode value opens a file for input which already exists.
- **"w"**: (write) fopen() with this mode values creates a new empty file for output operations. If the file with the same name exists, then this file is over-written.
- **"a"**: (append) this value of mode also opens a file for output purpose. If the file already exists then it write the data at the end of the file but if the file does not exists then it will create the file.

- **"r+"**: (read/update) the file, already exists, will be opened for update purpose- both read and write.
- **"w+"**: (write/update) a new file is opened for write and update purpose. If the file already exists then it is over-written.
- **"a+":** (append/update) a file is opened for update (read and write) purpose, but all the output data is written at the end of the file. A new file is created if the file does not exist.

We should take of the value of the mode parameter to properly manipulate the data in the file. Using wrong value of mode parameter may result in data loss or wrong results.

fclose() function is used to close any opened file. It also closes the stream attached to this file. The syntax of *fclose()* function is given as under:

```
        fclose(FILE *a_file)
```

The file associated with a_file pointer will be closed when we call this function and also the stream will be closed and the file associated will be saved (written) automatically to the storage media.

A sample C/C++ program demonstrating the use of fopen() and fclose() functions is given here:

```
        int main ()
        {
                FILE * file_ptr;
                File_ptr = fopen ("myfile.txt","w");
                if (file_ptr!=NULL)
                {
                        fputs ("file is opened and closed ",file_ptr);
                        fclose (file_ptr);
                }
                else
                {
                        printf("\n ERROR: mufile.txt cannot be
opened");
                }
                return 0;
        }
```

In this program file_ptr, a FILE pointer is declared. The fopen() is called with the desired file name and mode values. If the file is opened successfully the file_ptr will have some valid value otherwise it will have a NULL value. Then the value of the file_ptr is tested, if the file is opened successfully then a message is written to the file and file is closed using fclose() function. If the file is not opened successfully then an error message is displayed and the program is ended.

Activity 6

Write a C/C++ program which opens a text file, name of the file may be given as argument to the program, then it should go through the whole file, should display and count all the characters in the file, it should also count the number of lines in the file. At the end it should display the number of characters and number of lines in the files. The program should properly close the opened file.

5 Structures and Unions in C/C++ (Chapter 6)

Learning Objectives

On the successful completion of this chapter, the students will be able:

- To conceptualize the structures and unions.
- To define, declare and use structures and unions in C/C++ programming.
- To pass the complex data to functions either by value or by reference.
- To process data with the concept of bit-fields or unions.
- To implement file processing like database with the help of structures to store and retrieve data and perform reporting.

Preface

Dear students you have studied about different C/C++ programming techniques. To process data we use variables to store data. We use different types of variables. These data types include int, char, float, double, etc. we use these variables of these types directly or pointers to these variables. These variables provide facility to store the data of simple one type. If we have to store complex data or data which consists of different components which are of different data types then we have to use some other technique.

5.1 Introduction

To store collection of data, we may use arrays but arrays manage the data of same type of objects or variables. Arrays are suitable for a number of different types of problems but in real life problems we have to use a data type which is suitable for storing multiple types of data components in it. In this situation arrays do not solve the problem. For example if we have to save the data which may store a Date, then a simple integer variable or array will not solve the problem. For this purpose we will need a variable which have three components each consists of an int type; one for Month, second for Day and third for Year value. To handle this situation we will use structures.

Unions are also the same type of objects. It helps to store different data components in a compact form in a single variable. It is useful in communication and especially hardware programming and the situation where memory constraint is very crucial.

5.2 Structures and Unions in C/C++

5.2.1 Defining a Structure/Union

Structures and Unions are the data types which are derived from other data types. So these are derived data types. The structure in C/C++ language is defined using **struct** keyword. It is a derived and complex data type declaration. Using struct, we can define logically grouped variables of different types which can be used under single name. It is also represented and stored physically in a block of memory. We can access different variables using single name. The **struct** can also contain other complex and simple data types.

The syntax to declare a struct, are as follows:

struct structure_name{	typedef struct structure_name{
data_type member1;	data_type member1;
data_type member2;	data_type member2;
.	.
.	.
.	.
};	} struct_data_type_name;
then the variable of this type will be declared as follows:	then the variable of this type will be declared as follows:
struct structure_name var1;	struct_data_type_name var1;

These both declarations are used to define a struct. The difference will be when we declare the variable of these types.

The union can also be defined in the same manner. The only difference is that we have to use **union** keyword instead of struct. The syntax to declare a union, are as follows:

union union_name{	typedef union union_name{
data_type member1;	data_type member1;
data_type member2;	data_type member2;
.	.
.	.
.	.
};	} union_data_type_name;
then the variable of this type will be declared as follows:	then the variable of this type will be declared as follows:
union union_name var1;	union_data_type_name var1;

Now, for example we want to define a struct of date data type, then we will use the following codes:

struct **date**{
    int month;
    int day;
    int year;
};
	typedef struct new_date{
	    int month;
	    int day;
	    int year;
	} **date**;

then the variable of date type will be declared as follows:	then the variable of date type will be declared as follows:
struct **date** date_var1;	**date** date_var1;

In the next section we will discuss about that how we can access the components of these structures and unions.

5.2.2 Accessing Members of a Structure/Union

Hi dear students! In the previous section we have learnt that how we can create structures and unions. Now we will discuss that after creating these types and declaring these types, how we can access the components of these compound data objects.

Following the example of date type struct which we have declared in the previous section, if we have declared a variable date_var1, then the components of the date type will be accessed using dot (.) operator as follows:
(let we have a date with month value 11, day value 15 and year value 2010)

```
.
.
.
date_var1.month = 11;
date_var1.day = 15;
date_var1.year = 2010;
.
.
.
```

The struct construct facilitates us with the assignment operator. If we have a second variable date_var2 and we use the following statement

```
date_var2 = date_var1;
```

Then the values of the components of date_var2 will be the same as were assigned to the components of date-var1.

If we use a pointer to represent a struct type variable then we will have to use arrow (->) operator to access its components. The following code is an example with the date_var1 variable and a pointer.

```
date *date_ptr;
date_ptr = &date_var1;
date_ptr->month=12;
date_ptr->day=16
date_ptr_year=2012;
```

After the execution of this code the components of date_var1 variable will have new values assigned through the date_ptr, a pointer to date_var1.

Structures and unions also provide the facility to pass the complex data to the functions either by value or by reference. There are also the cases when we have the structure which is self-referencing. They have a pointer inside their body which has the same data type defined by that structure. These types of structures are used to build some important and useful data structures known as lists, tree, stack, etc.

Example

As an example of a complex data types, we here discuss a very simple data structure. Let we build a struct which will represent our personal address book. The components of our address book may be:

- A code for each entry
- Person name
- Person mobile number
- Person National ID card Number.
- Person Salary

Now to declare such struct we will use the following code:

```
struct contact {
    int code;
    char name[20];
    char mobile[12];
    char NIC[20];
    float salary;
}
```

Now to build an address book with 100 entries in it, we can declare an array of this type of data as given under:

```
struct contact address_book[100];
```

Then the components of this address_book can be accessed as :

address_book[1].code, address_book[1].NIC, address_book[1].salary and so on.

In this way we can manage an address book.

Activity 7

As discussed in the previous sections, write a C/C++ program to implement the **date** struct. You should implement the following functionality:

- Take the input of date variable components.
- Check the input for errors, i.e. month value should not be 0 and should not be greater than 12; the day value should not be 0 and should be according to the month, and year should not be 0.
- After taking input, display the input value as the format:
  mm-dd-yyyy
- Take the two variables of date type and then take the input, and check the two variables that either these two dates are equal or which one is greater and which one is smaller.

Task 2

Implement the address book application. One address book entry should have the following components:

- Code
- Name
- Date of Birth (month, day, year)
- Address (house number, street number, block, etc.)
- City
- Mobile Number

Then implement the following functionalities:

- Should add, edit, delete an address book entry with proper ERROR checking
- Should be able to display an entry in proper format
- Should be able to store data in a file
- Should be able to search for a particular Code or Name or Mobile Number.
- Should be able to show all the entries one by one.
- Should display a Menu for user choice that what he/she want to do from the above described functions. Example of Menu is as under:

```
***************************************

>>1<< To add new Record

>>2<< To find a Record by Code

>>3<< To find a Record by Name

>>4<< To find a Record by Mobile Number

>>5<< To Delete a Record

>>6<< To display all Records


>>0<< To Quit

***************************************

Enter Your Choice:

***************************************
```

The program should continue to execute until user enter 0 to quit.

Unit Programming with Visual BASIC

6 Building a User Interface (Chapter 10 – 10.5)

Learning Objectives

After the completion of this chapter students will be able to:

- Create a form
- Change the different properties of form, e.g. Name, Text, Background, Icon, etc.
- Add different controls to the form.
- Add response to different events of the form and other components
- Show and hide the forms according to the requirement
- Setting the startup form of the project/application

Preface

So far we have gone through a series of programming sessions. We have discussed the programming concepts using C/C++ language. C/C++ is generally known as a system programming language. It is a very hard to build attractive and user-friendly environment using C/C++ programming language.

User interface is the basic element and requirement of any computer program or application. It defines that how user will interact with the computer to use a particular application.

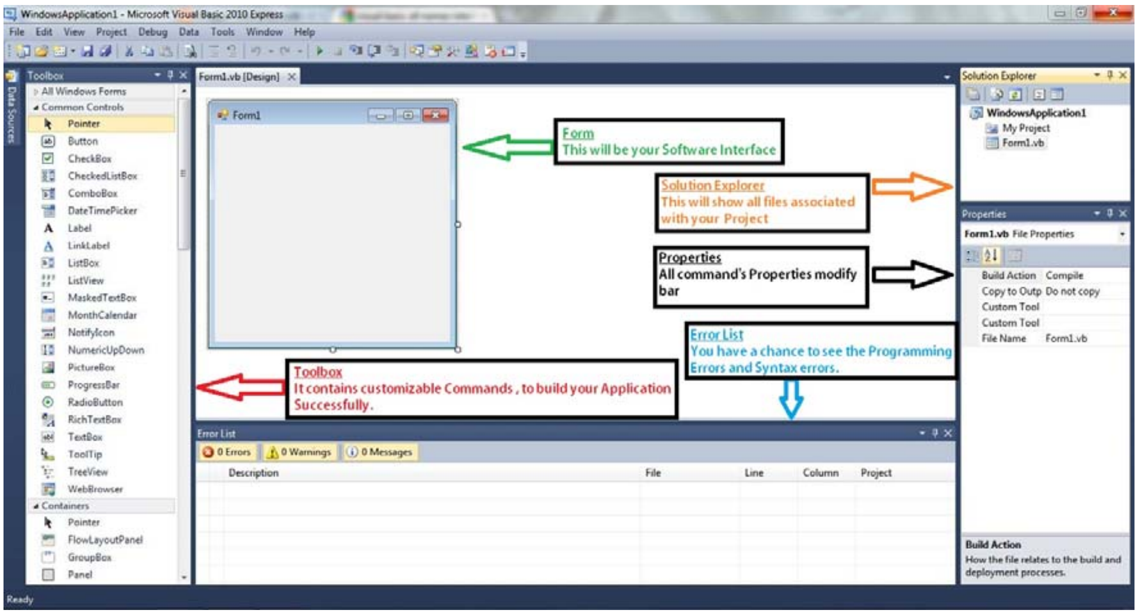
Visual BASIC is a programming language which provides facilities, tool and techniques which help the programmer to build attractive and user-friendly GUI. In this chapter we will learn to build and use GUI (known as Forms) in Visual BASIC environment.

In the following sections you will learn how to build applications and programs in a visual environment. You will become familiar with the development of a Graphic User Interface. The use of different visual components will be explained. It will be explained that how we can get input in GUI (Forms) and how we can display information for user on forms. You learn to build lists, menus and other advanced features of GUI. Visual BASIC is a very programmer friendly visual programming environment. You can easily build database applications using this wonderful language.

6.1 Building a User Interface

6.1.1 Building Forms

In Visual BASIC user interface is build using forms. A form is a screen which is displayed when an application build in Visual BASIC starts. Forms are actually windows of some application. Using forms user can enter data and can view data and its processed results. It is very essential to learn how to design a form. Form is just like a canvas on which we draw the user interface using components provided by Visual BASIC. After placing the components on a form, we can adjust their positions, colors and other properties. When we start to build an application then a blank form is automatically displayed. If it is not displayed then we can add a form to the project.



Form
This will be your Software Interface

Solution Explorer
This will show all files associated with your Project

Properties
All command's Properties modify bar

Error List
You have a chance to see the Programming Errors and Syntax errors.

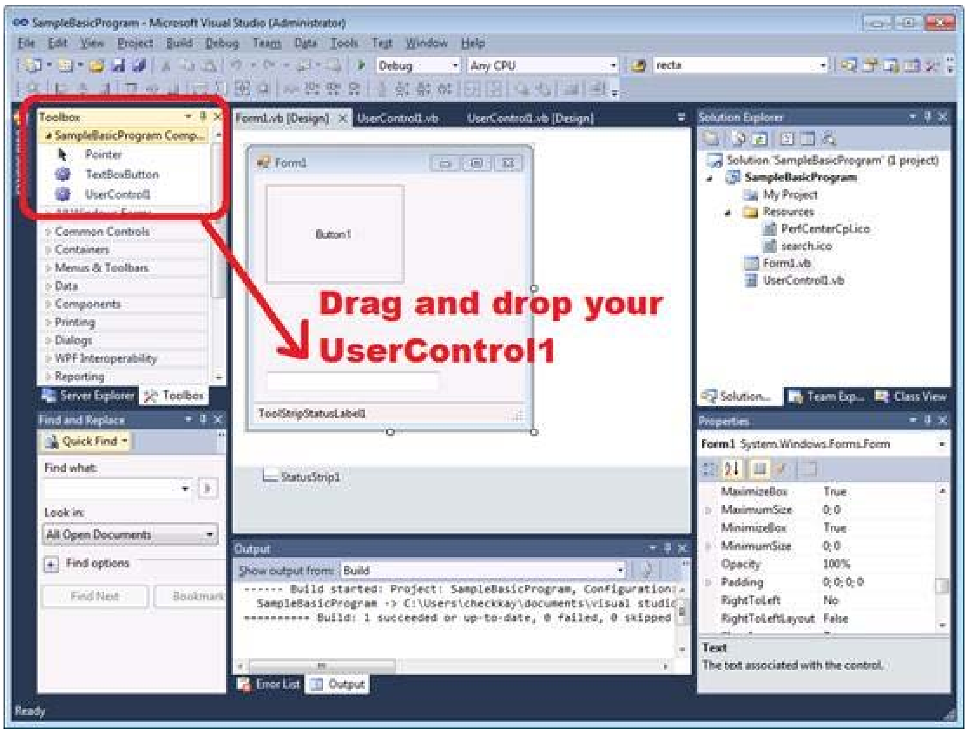
Toolbox
It contains customizable Commands , to build your Application Successfully.

Figure 7 Visual Basic 2010 interface[8]

Form is treated as an object in a Visual Basic application. Whenever we add an object to any project then Visual Basic assign it a default name but it is a good and professional approach to assign a descriptive name to each object. Same is the case with forms. As soon as we add a form to the application we assign it a descriptive name using its property dialog box or property sheet. We can also change the appearance of the form using properties sheet. We can change its title, its text, its background color, etc. We can also add a picture to the background of a form.

8 http://2.bp.blogspot.com/-hddOrWdHNJU/T65I446D2tI/AAAAAAAAANo/lbLo-j2toOo/s1600/Capture.JPG

We can add different components to the Form by drag-n-drop form the Tool Box:



Show output from: Build

```
------ Build started: Project: SampleBasicProgram, Configuration:
  SampleBasicProgram -> C:\Users\checkkay\documents\visual studio
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped
```

Properties

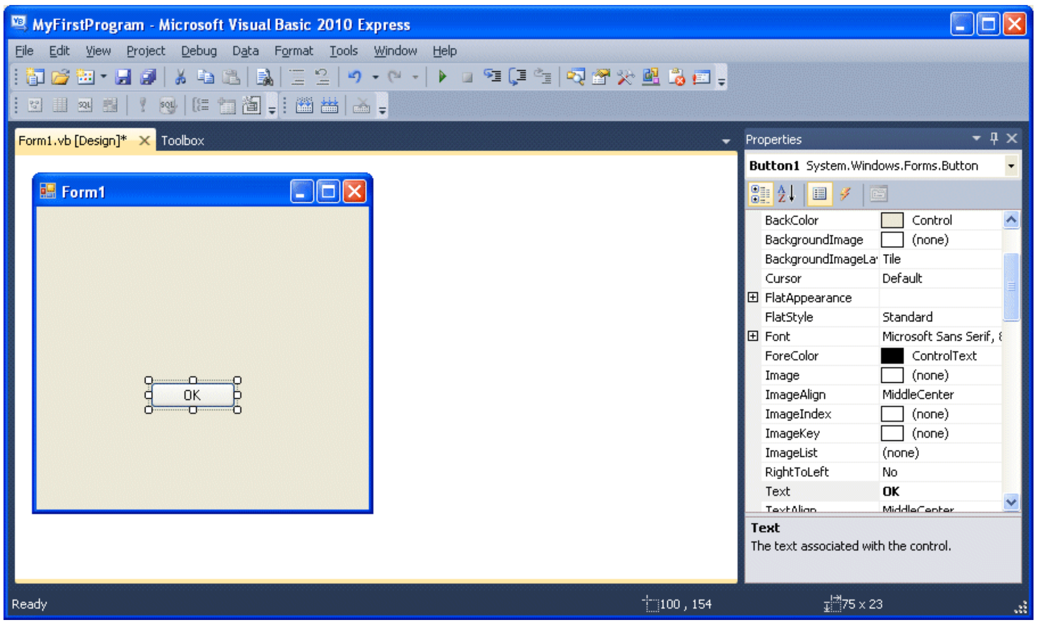
Form1 System.Windows.Forms.Form

MaximizeBox	True
MaximumSize	0; 0
MinimizeBox	True
MinimumSize	0; 0
Opacity	100%
Padding	0; 0; 0; 0
RightToLeft	No
RightToLeftLayout	False

Text
The text associated with the control.

Figure 8 Adding user controls to the form[9]

If we want to add a command button, we drag command button from the tool box to the form canvas. The text displayed on the command button is called caption. We can change the caption of the command. For this, from the properties sheet we type *OK* in the text field. Form and button will look like the following:


MyFirstProgram - Microsoft Visual Basic 2010 Express
File Edit View Project Debug Data Format Tools Window Help
Form1.vb [Design]*
Toolbox
Form1
OK
Ready
100 , 154
75 x 23

Properties

Button1 System.Windows.Forms.Button

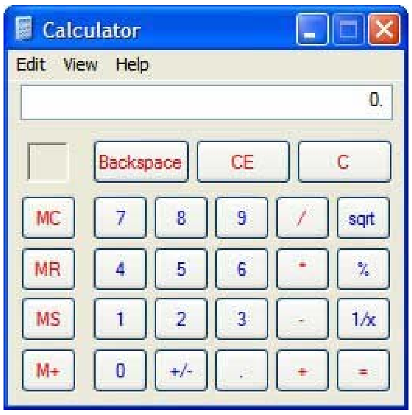
BackColor	Control
BackgroundImage	(none)
BackgroundImageLa	Tile
Cursor	Default
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif, 8
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	**OK**
TextAlign	MiddleCenter

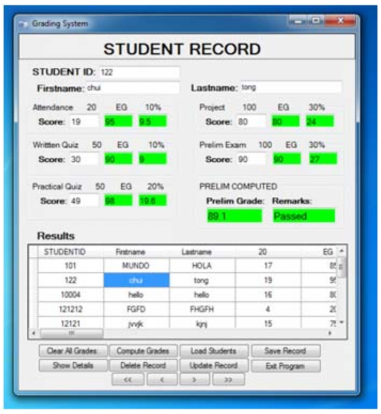
Text
The text associated with the control.

Figure 9 Form with OK command button[10]

9 http://1.bp.blogspot.com/-SuUcV5-pcks/ULvpuqAhPDI/AAAAAAAABJE/rkfhp5vGjs8/s1600/image007-746581.jpg

When we have added a component then we can program its different events. Events means that if something happen to the object then it may respond to that action according to code written in its event procedure. For a command button we generally program its click event. The name of the click event in visual basic code window is ButtonName_Click. So whatever code we write in this event, it will be executed when we click the button with mouse or we press Enter when this button is selected. We can add multiple components to our form as per our requirements. The form may have a simple look just like calculator in windows to a complex one to show a Bank account or student grading system.




Grading System

STUDENT RECORD

STUDENT ID: 122
Firstname: chui Lastname: tong

Attendance 20 EG 10% Score: 19 95 9.5
Project 100 EG 30% Score: 80 80 24
Written Quiz 50 EG 10% Score: 30 90 9
Prelim Exam 100 EG 30% Score: 90 90 27
Practical Quiz 50 EG 20% Score: 49 98 19.6

PRELIM COMPUTED
Prelim Grade: 89.1 Remarks: Passed

Results

STUDENTID	Firstname	Lastname	20	EG
101	MUNDO	HOLA	17	85
122	chui	tong	19	95
10004	hello	hello	16	80
121212	FGFD	FHGFH	4	20
12121	jvvjk	kjnj	15	75

Figure 10 Different forms in Visual Basic: Calculator / Grading System[1]

6.1.2 Showing and Hiding Forms

A fairy simple application in Visual Basic may consist of only one form. The best example of this type of application is windows Calculator. But normally even simple applications may have more than one form. So it is required to show or hide the forms according to the response of user to choices offered. Some forms are shown just to show some message, or to take some input from the user while other may be shown to get complex data input from the user.

When the application is started a form is displayed in the start. We can change the settings that which form should be shown at the startup of the application. This is done using the project settings in the Visual Basic. To do this, from the project menu we select the Application properties. A new tab will be opened in the project area. Form this tab we can set the Start Up form. Against this there is a drop down list which will show all the available forms within the project and we can use any of these as startup form but certainly we have designed some specific form to be displayed at the start of the application and we choose that one for this purpose.

We have form_name.show() and form_hide() methods to show and hide any form programmatically.

---

10 http://www.vbtutor.net/index.php/vb2010-lesson-1-introduction/

6.1.3 Working with Controls

As in our daily life, we come across different type of problems to be solved using programming languages. To solve a problem we certainly need some input from the user and then we display the results after processing that input. So we need some controls on the form which help us to take input from the user and we also need some controls on the form which help us to display the results.

Visual Basic provides us different controls to serve the purpose of input and output. To get input some controls are for free form input, which allow the user to input any value as he wish and some are for controlled input which provide some fixed choices for input. We will discuss these in the following paragraphs.

We can add a control to the form by just double-click it in toolbox, by drag-n-drop from tool box to the form, to copy and paste a control, or by drawing a control on the form. It is very easy to add controls to the form. The hard part is to arrange them on the form. It is just an art. The arrangement of the controls is the ultimate product which a user will get when you deploy or give the application to the user. Now it is always different for different users to like one arrangement or the other. The task of a developer is to arrange the controls in a logical and aesthetical way so the user feel easy to use the form. The layout of the form may correspond to some existing document to which user is familiar.

Here we discuss few of the controls:

Text Box: it is used to get free form input from the user. User can enter any alphabet in this control. We can write code on its different events.

Label: it is used to display some text on the form. The property of the label is text which can show the text or some results. The value of this text can be change using properties sheet or in the code.

Command Button: this control is used for controlled input. It just gets the response from the user by the click of mouse or by pressing enter. We can program its events.

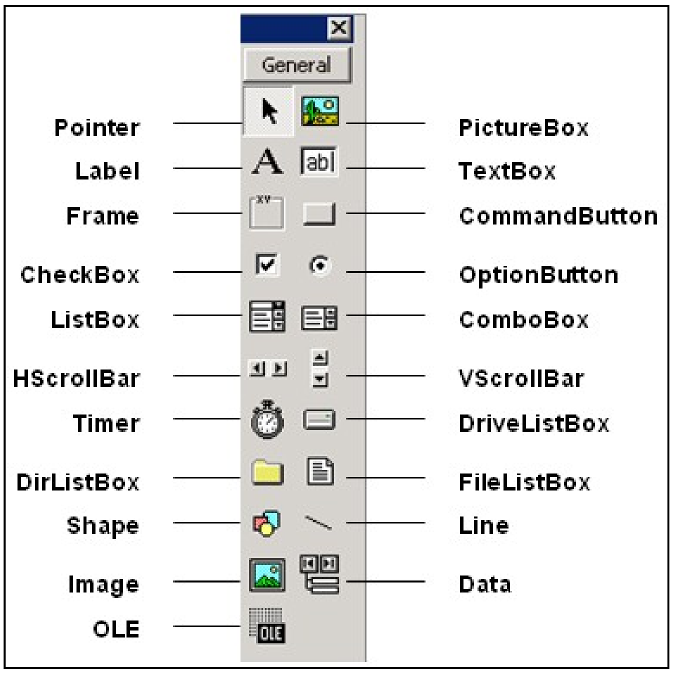


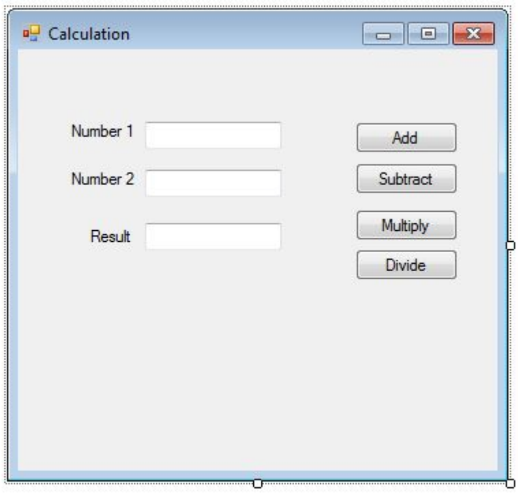
Figure 11 *Tool box with Controls*[1]

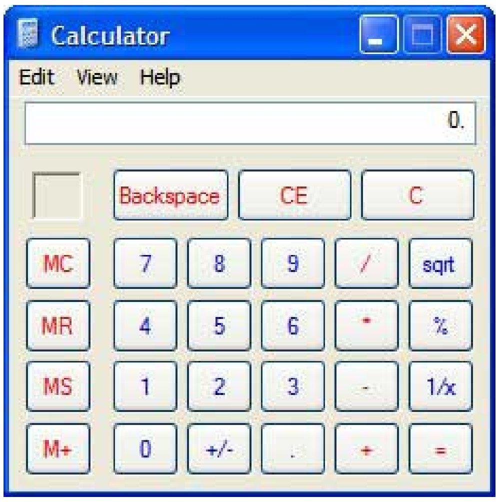
List Box and **Combo Box:** these two controls are used to provide the pre-defined choices to the user. In both the controls a list is fed in the control and user can chose an input just from these choices. The difference between these two controls is that in List Box more than one list element can be displayed at a time where as in combo box the list is displayed when we click on the arrow of the combo box and the selected value is displayed in the text box of the combo box.

You can find the detailed information about the other controls on the following link:
http://msdn.microsoft.com/en-us/library/aa733708(v=vs.60).aspx .

Activity 8

Create/design the forms shown in the figure below:





6.1.4 Creating MDI Forms

Dear students we have discussed about the visual Basic project and different components of Visual Basic project. The projects we have discussed so far are mostly SDI project. SDI means single-document interface. SDI is a project which opens all the forms at the same level and there is no hierarchy.

The MDI in Visual Basic is used for Multi-Document Interface. All the applications designed today's are mostly MDI applications. It means that they can open many child documents (forms) within a parent form. Parent form acts as a container. In an MDI application, there is a parent form and one or more child form which can be opened in parent form. You can find many examples of such applications. One good example could be Adobe Photoshop. In Adobe Photoshop, on running it, we have many sub windows. Some of these are canvas or working area, other could be tools, layers, etc. The child form can exist only within the parent window.

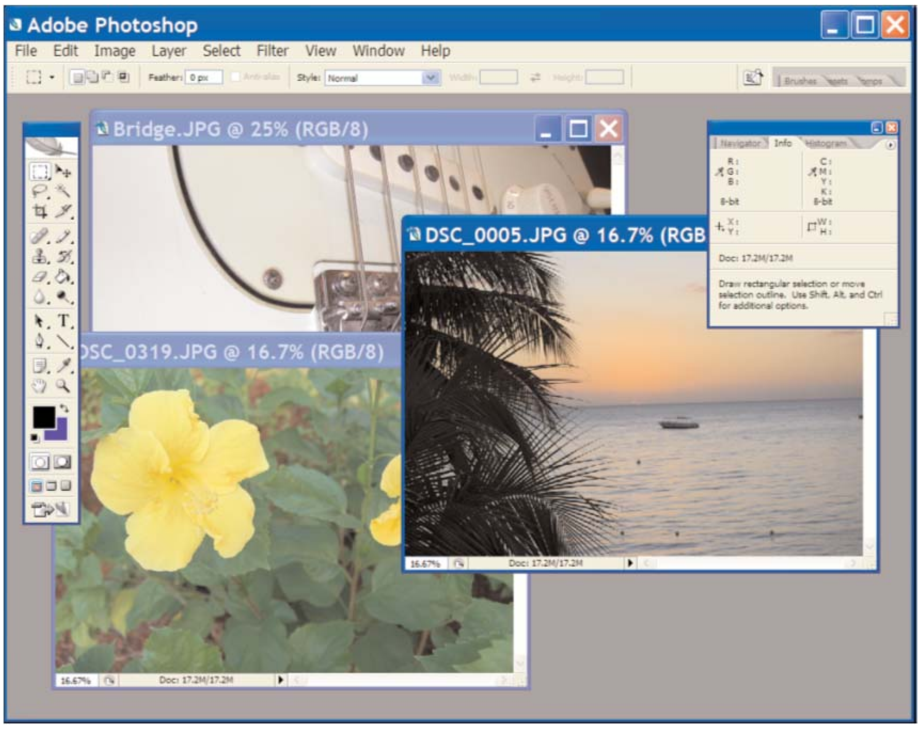

Adobe Photoshop
File Edit Image Layer Select Filter View Window Help
Bridge.JPG @ 25% (RGB/8)
DSC_0005.JPG @ 16.7% (RGB
DSC_0319.JPG @ 16.7% (RGB/8)
Navigator Info Histogram

Figure 12 Adobe Photoshop, an MDI application.[11]

When we start to create and MDI form then first of all we create a parent form. When we start Visual Studio and create a new project then this project will contain a single form. We can set the name of this form to some descriptive name using the Properties sheet. And from the same Properties sheet we set the *IsMdiContainer* property to *true*. This will make the current form a parent form. The background of the form will be changed to dark grey, which is the default background for a parent form in Visual Basic. From here we can start to build an MDI application. Now from the *Add New Item* toolbar button we can add more forms to this project.

[11] Sams' Teach Yourself Visual Basic 2010 in 24 Hours. p:137

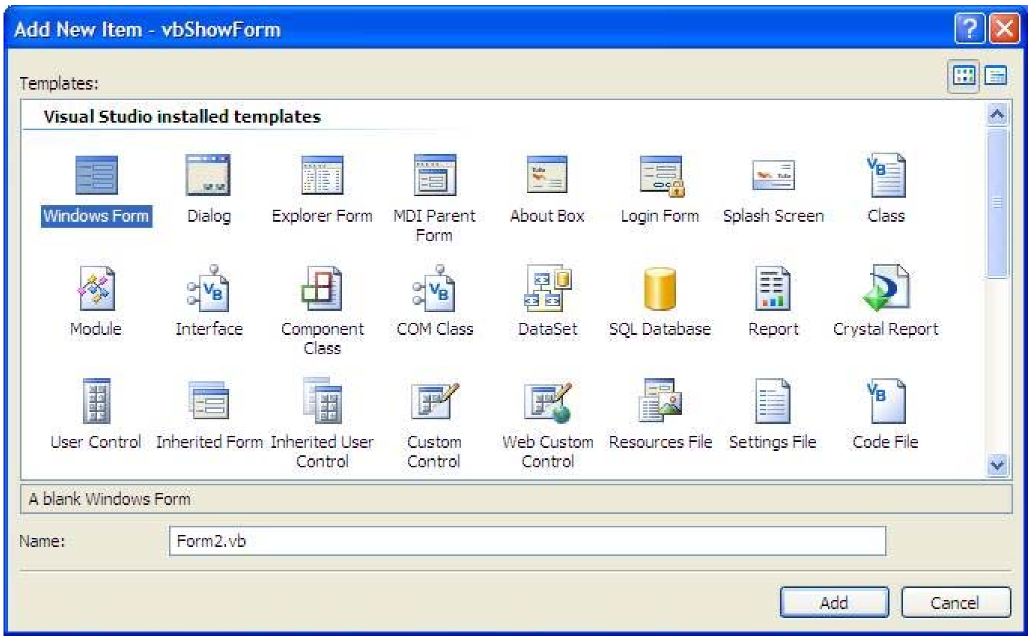


Figure 13 Add New Item Dialog Box[12]

Let we have given the name MDIParentForm to the parent form. Then we add to child forms with the names MDIChildOneForm and MDIChildTwoForm. Now we changed the Text properties of these forms to MDI Parent, MDI Child 1 and MDI Child 2 respectively.

Now in the Load Event of the MDIParentForm write the following VB code:

```
ChildOneForm.MdiParent = Me
ChildTwoForm.MdiParent = Me
ChildOneForm.Show()
```

Now if we run the project then it will have a Parent form and two child forms. The statement *ChildOneForm.Show()* will show the *childOneForm* in the parent form. If we have a button on the *ChildOneForm* or a menu item in parent form the by having the following code in its click event we can show *ChildTwoForm* in the parent form as well.

```
ChildOneForm.Show()
```

At this time the application will have the appearance to the right.

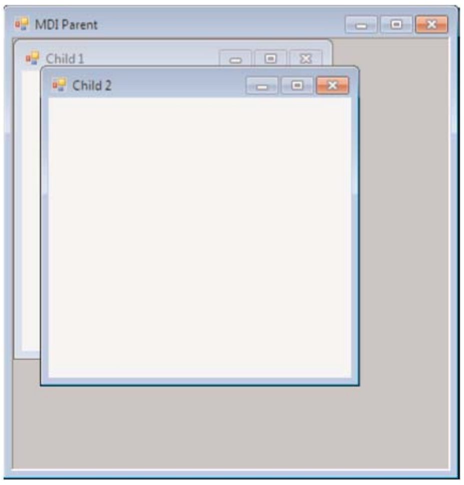


Figure 14 An MDI application with 2 child forms

[12] http://www.techotopia.com/index.php/Image:Visual_studio_add_new_item.jpg

Activity 9

Create an MDI application as discussed in the above text.

6.1.5 Setting Startup Form

We discussed about SDI and MDI application designing in Visual Basic. If we have designed a SDI application then there may be many forms which exist in the application. We will have to decide that which form will appear when we start the application. If we do not set any form as startup then the Visual Basic itself will make the form startup form which we have added as very first form in the project. If we decide to make some other form as startup form, then we can do so by right-clicking the mouse on project name and then properties. Here we change the startup form property and give the name of the form which we want to work as startup form. In an MDI application we should set the Parent form as startup form otherwise the application will not work as we will be expecting.

7 Making Things Happen—Programming and Coding (Chapter 11.3 – 11.6)

Learning Objectives

After the completion of this chapter students will be able to:

- Create Visual Basic code modules and procedures
- Call procedures and to pass parameters to procedures
- Understand different data types provided by Visual Basic
- Define and use variable and constants
- Work with data collection like arrays
- Perform arithmetic and Understand operator precedence
- Compare values and understand Boolean logic
- Manipulate strings, dates and times
- Make decisions using If...Then and If...Then and ElseIf and to use Select Case
- Use Loop for a specific number of times using For...Next
- Use Loop based on a condition using Do...Loop

What you have already learnt in the beginning of this chapter:

11.1 Creating and Calling Code Procedures

We write code in procedures in Visual Basic and Visual Basic uses Modules to store these procedures. We can write procedures with any name and with any number of parameters but generally the procedures are attached to the events of the object which are placed on the GUI of the Visual Basic application. We have studied and discussed about these in our class in detail.

11.2 Using Constants, Data Types, Variables, and Arrays

Initially the programming was started to process the data and it was the only use of computer at that time. Even nowadays, the purpose is almost the same. The only difference is that the nature of data has been changed. Now we can process many types of data. To process the data we have to represent this data by some variables in our program. Depending on the nature and type of data we use different types of variables and constants. These data types include Integer, String, Double, Date, Time and many others. We have studied and discussed about these constants, variables and data types in our class sessions.

7.1 Introduction

A Visual Basic application can perform all the tasks automatically at its own or it may responds to the user actions which he/she performs when the application is running. In Visual Basic, all the code is written in different events of the objects which are included in the visual interface of the application. These components include Forms, Buttons, Menus, and all the other components which we can add to a form from the control box.

Here in this chapter we will discuss about the tool and techniques which will use to build the applications and also to write the code to make the things happen. It is our code which makes an application intelligent.

7.2 Making Things Happen—Programming and Coding

7.2.1 Performing Arithmetic, String Manipulation, comparisons, working with Dates and Times

In our daily life we face many situations where we will have to perform calculations. These may be the calculation of marks, grades, calculation of a purchase bill, calculating the balance amount, etc. Same is the situation in programming. Most of the time we will be performing calculations/arithmetic in our programs/codes. There will also be situations when we will have to compare the values. Visual Basic uses Boolean logic to compare values. These may be to compare the income to calculate tax about tax slabs, to compare the ages against a date, etc.

7.2.1.1 Performing Arithmetic

One must have good basic mathematical skills to be a good programmer. Because in a program we mostly perform arithmetic. So one should know that at what time he has to perform which mathematical operation to get the desired results. Following are the basic arithmetic operations which we can perform in Visual Basic.

- Addition: It can be performed by simply using "+" character in-between the values or variables which we want to add. For example:
  ```
  value1 = 6
  value2 = 7
  result = value1 + value2
  ```
  the variable result will have the value 13.
- Subtraction: It can be performed by simply using "-" character in-between the values or variables which we want to subtract. For example:
  ```
  value1 = 9
  value2 = 7
  result = value1 - value2
  ```
  the variable result will have the value 2.

- Multiplication: it can be performed by using "*" character in-between the values or variables which we want to multiply (we cannot use "x" in our code to multiply). For example:
  ```
  value1 = 6
  value2 = 7
  result = value1 * value2
  ```
  the variable result will have the value 42.

- Division: it can be performed by using "/" or "\" character in-between the values or variables which we want to divide. The difference will be, "/" will divide along with fractional part in the answer whereas the "\" will give only the integer part and the fractional part (remainder) will be discarded. For example:
  ```
  value1 = 10
  value2 = 4
  result1 = value1 / value2
  result2 = value1 \ value2
  ```
  the variable result1 will have the value 2.5 where result2 will have the value 2.

- Exponentiation: the power of a value or variable can be calculated by using "^" character. For example:
  ```
  value1 = 6
  value2 = 2
  result = value1 ^ value2
  ```
  the variable result will have the value 36.

- Modulus: Modulus is an operation in which we get the remainder value of the division operation. Mod operator is used for this purpose. For example:
  ```
  value1 = 14
  value2 = 5
  result = value1 Mod value2
  ```
  the variable result will have the value 4.

7.2.1.2 String Manipulation

The strings are type of data which consist of text. The length of the text may be 1 or more than 1. If the length of a string is 0 (zero) then it is called a NULL string. The real world is full of strings. All the communications take place using words and sentences which all are strings. So the string manipulations is very important component of the programming. In common applications mostly we will have to deal with strings, e.g. we have to compare the names, find an address, deal with cities, etc. in this section we will discuss that how we can deal with strings. The following are the string manipulation facilities provided by Visual Basic.

- Concatenation: concatenation is the operation which helps to add two strings. This is not the actual arithmetic addition rather it will take two strings and make a longer string consisting of the initial two strings. The "+" or "&" characters can be used to perform the concatenation. For example:
  ```
  String1 = "Visual"
  String2 = "Basic"
  result1 = String1 + String
  result2 = String1 & String
  ```
  both the variables result1 and result2 will have the same value "Visual-Basic".
- Number of Characters in a String: we can determine the number of characters in a string we use Len() function. For example:
  ```
  String1 = "Visual"
  result1 = Len(String1)
  ```
  the variable result1 will have the value 6.
- Text from the start of a String: we can get the specified number of characters from the start of a string by using Microsoft.VisualBasic.Left() function. This function takes the string and the number of characters as parameters and return the specified number of characters from the start of the string. For example:
  ```
  String1 = "Visual Basic"
  result1 = Microsoft.VisualBasic.Left(String1,5)
  ```
  the variable result1 will have the value "Visual".
- Text from the End of a String: we can get the specified number of characters from the right side of a string by using Microsoft.VisualBasic.Right() function. This function takes the string and the number of characters as parameters and return the specified number of characters from the End of the string. For example:
  ```
  String1 = "Visual Basic"
  result1 = Microsoft.VisualBasic.Right(String1,5)
  ```
  the variable result1 will have the value "Basic".

- Text from in-between a String: we can get the specified number of characters from the in-between a string by using Mid() function. This function takes the string, starting position and the number of characters as parameters and return the specified number of characters from the starting position. For example:

  ```
      String1 = "Visual Basic 2010"
      result1 = Mid(String1,8,5)
  ```

  the variable result1 will have the value "Basic".

- Trimming spaces from the start and end of a String: Visual Basic provides functions which help us to trim spaces from the start or the end or from the both sides of a string. These are as follows:
  _ Trim() : it will trim spaces from both sides
  _ LTrim() : it will trim spaces from the start
  _ RTrim() : it will trim spaces from the end

7.2.1.3 Comparisons

Comparing the values either constants or in variable is also an important component of the programming. These comparisons are needed when we are going to make decisions in our code. In these comparisons we may compare numeric values and string values. Even we can compare other types of variables but the data type should be the same to perform comparison. In general practice we compare the value of one variable to the value of another variable. We have also learnt this concept in C/C++ programming unit. The comparison operator returns the result as *True* or *False*. Followings are few examples of comparisons.

- Let
  Val1 = 7
  Val2 = 3
  Val3 = 7
  then comparisons
  (Val1 = Val3) will give True
  (Val2 > Val3) will give False
  (Val1 > Val2) will give True
  (Val2 < Val3) will give True
  (Val2 >= Val3) will give False
  (Val2 <= Val3) will give True

In our coding it is very simple to perform comparisons just we have to be clear in our mind that what we want to do. If we find it difficult to write comparisons then it is a good practice to first phrase the situation in an English sentence.

We can combine multiple comparison expression by using Boolean operators. These Boolean operators are And, Or, Not. Use of these Boolean operators is just like those as we have discussed in C/C++ Unit.

7.2.1.4 Working with Dates/Time

There may be situation when we come across the data of Date and Time types. And it is very often that we have to process this type of data in our programming. Sometimes these (Dates/Time) are like Strings and sometimes these are like numbers as we can perform some arithmetic operations on these.

It is very common to process Date type data. Some time we subtract two dates to find a period, sometime we add some dates to a given date to find a target date. We can define and initialize a date type variable as under:

```
Dim dteMyBirthday As Date = #7/22/2010#
```

Sometime we have to do the following to convert a string to a Date value.

```
dteMyDateVariable = CDate(txtBirthDay.Text)
```

We can retrieve different parts of a date by using the following functions:

```
DatePart(DateInterval.Month, #7/22/2010#) ' Returns 7
DatePart(DateInterval.Hour, #3:00:00 PM#) ' Returns 15 (military format)
DatePart(DateInterval.Quarter, #6/9/2010#) ' Returns 2
```

We can calculate the difference of two dates by using the following functions:

```
DateDiff(DateInterval.Year, #7/22/1969#, #10/22/2001#) ' Returns 32
DateDiff(DateInterval.Month, #3/3/1992#, #3/3/1990#) ' Returns -24
DateDiff(DateInterval.Day, #3/3/1997#, #7/2/1997#) ' Returns 121
```

We can get the current Date and Time of the system as follows:

```
Dim dteToday As Date = DateTime.Today
Dim dteToday As Date = DateTime.Now
```

Activity 10

1) Design a Visual Basic application which have
   - One form
   - 2 Text Box on the form for input purpose
   - 2 labels to show the results
   - 6 buttons for each of the arithmetic operation described above.
   - After entering the values in the text boxes when an operator button is clicked then the relative result should be shown in the labels.
2) Design a Visual Basic application which have
   - One form
   - 4 Text Box on the form for input purpose (2 for strings and 2 for numeric values)
   - 1 labels to show the results
   - 6 buttons for each of the string operation described above.
   - After entering the values in the text boxes when an operator button is clicked then the relative result should be shown in the labels.

7.2.2 Making Decisions in Visual Basic Code, Using If...Then

Most of the decision coding in programming is done by using *If…Then* construct. This construct look like:

```
If expression Then
    ' code to execute when expression is True.
End If
```

Here the expression is a logical test which results in a Boolean value. The value may be True or False. If the expression is evaluated as True then the code after Then is executed and the code after End If is continued after the completion of code in the if block but if the expression is evaluated as False then the code in If…Then construct is skipped and the execution is continued after End If.

If we have a situation in which we have to execute one of the two code block depending upon the condition then we use If…Then…Else construct. It has the following structure:

```
If expression Then
    ' code to execute when expression is True.
Else
    ' code to execute when expression is False.
End If
```

In this construct if the expression is True then True block will be executed and False will be skipped and if the expression then False block will be executed and True block will be skipped.

7.2.3 Branching Within a Procedure Using GoTo

Most important and notable comment is that it is a bad practice to use GoTo in a code. It is very hard to debug and maintain a code which have included GoTo statement.

Anyway to use a GoTo statement we must have to create a Label for a GoTo statement. GoTo is a simple jump statement which shifts the control to some location in the code. Following is a code which uses GoTo statement to form a loop:

```
IncrementCounter:
    intCounter = intCounter + 1
    If intCounter < 5000 Then GoTo IncrementCounter
End Sub
```

Here IncrementCounter: is a label and the if statement shifts control to the label depending upon the condition.

7.2.4 Looping in Visual Basic

Loops are very important construct in programming. Loops are used to repeat a certain block of code to a specific number of times or to the point where a certain condition is fulfilled. There are two types of looping construct in Visual Basic.

7.2.4.1 Looping a Specific Number of Times Using For...Next

For…Next loop is the simplest loop construct. It was introduced in the very initial versions of the BASIC programming language. In this construct we use a counter variable. In the start of the loop we assign it a start value and tell the end value. The loop goes through the body of the loop depending upon the start value and counting till the end value. For example if the start value is 0 and end value is 10 then the body of the loop will be executed 10 times. There is an option part which is called Step. The step value is used to increment the counter variable. If the step value is not mentioned then it is 1. The following is the structure of this construct.

```
For countervariable = start To end [Step step]
    ... [statements to execute in loop]
Next [countervariable]
```

7.2.4.2 Using Do...Loop to Loop an Indeterminate Number of Times

If we want to repeat a block of code for indeterminate number of times, then we use Do…Loop construct of Visual Basic. It is used in the situations where we don't know in prior that how many time the loop will be executed. The structure of the this loop is as follows:

```
Do while expression
    [Statements]
Loop
```

In this construct the statements will be executed until the expression is evaluated to True.

Task 3

Create an application which has multiple documents. It should be created using MDI format. This application should mean to get data of a student (personal and academic). The feature and functionality should be as follows:

- It should contain 1 parent form
- From the parent form, user should be able to select between different forms.
- 1 form should be for student personal information
- 1 form for student academic data (5 subjects, obtained marks and maximum marks)
- 1 form to show the result card, which should contain
- Name
- Roll No
- Date of Birth
- Marks for Each subject (%age marks, Grade)
- Over-all %age and Grade
- Remarks for Excellent, Very Good, Good, Satisfactory and Fail
- The application should have proper exit buttons.

8 Working with Data and Databases (Chapter 13.3 – 13.5)

Learning Objectives

After the completion of this chapter students will be able to:

- Create and Save files in Visual Basic
- Use the OpenFileDialog and Save FileDialog
- Manipulating data in files
- Making a connection to a Database
- Accessing the Records and Fields of a Database Table
- Navigating through Records in a Database Table

Preface

Dear Student, in the following sections you will learn how to build applications and programs in a visual environment. You will become familiar with the development of a Graphic User Interface. The use of different visual components will be explained. It will be explained that how we can get input in GUI (Forms) and how we can display information for user on forms. You learn to build lists, menus and other advanced features of GUI. Visual BASIC is a very programmer friendly visual programming environment. You can easily build database applications using this wonderful language.

8.1 Introduction

What you have learnt in the prevision section of Chapter 13:

13.1 Performing File Operations

File operations include to Create a File, To Delete a File, To Edit the contents of a File, to Open a File, To Close a File. We have discussed and learnt these concepts in our class session of this module.

13.2 Using the OpenFileDialog and SaveFileDialog Controls

Dear Students ! Visual Basic provides us with two very useful controls. One is OpenFileDialog and the other is SaveFileDialog. The name of these two controls reveals the facility provided by these two controls. The OpenFileDialog is used to display the contents of the computer storage and if a file is selected then the file can be opened by the code in the application. On the same lines, if we want to save a file then the SaveFileDialog control is used. We have studied these both controls in our classroom sessions.

8.2 Working with Data and Databases

8.2.1 Manipulating files with the File Object

To interact with the files we have to interact with the file system. Visual Basic has provided with a namespace, System.IO to handle the file operations. System.IO comes with many properties and methods, and we can perform any file operation using these properties. In the following sections we will discuss some of these operations.

To know whether a file exists

Before performing any operation to a file we should know that whether the specified file exists in the file system or not. Because if a file does not exists and we perform some operation on it then there may be a system EORROR and our application may halt or malfunction. To implement this we use Exists() method of the System.IO.File object.

```
System.IO.File.Exists(FileName)
```

This method returns True if the file exists with the specified name and path and false otherwise. So we can test the existence of a file using this method.

Copying a file:

We can use Copy() method the System.IO.File object to copy a file from one location to another.

```
System.IO.File.Copy(SourceFile, DestinationFile)
```

Moving a file:

The MoveI() method the System.IO.File object is used to move a file from one location to another location.

```
System.IO.File.Move(SourceFile, DestinationFile)
```

Deleting a file:

Similarly Delete() method is used to delete a file from the storage. It is dangerous operation, so use some confirmation method before deleting a file.

```
System.IO.File.Delete(SourceFile)
```

Activity 11

Create a project that enables the user to select a file with the OpenFileDialog control. Store the filename in a text box. Provide another button that, when clicked, creates a backup of the file by making a copy of it with the extension .bak.

8.2.2 Working with Text Files - Reading and Writing Text Files

Text Files are widely used to store data. Text files are used from the very early ages of the computing. Although these are not an efficient way to store data in text file but even then these have very important concepts related to data processing. The text files are very easy to handle and process. Visual Basic provides us with some classes which make it much easier to play with text files. These classes are StreamWriter and StreamReader. We have to use two different objects to read from the file and to write to the file. At a given time, only one of these can access a file.

Writing to a Text File

We use StreamWriter object to write text to a text file. To perform this operation first of all we have to create a StreamWriter class object. It will be created as:

```
Dim objFile As New System.IO.StreamWriter("c:\test.txt")
```

After the successful execution of this line we will have an object names objFile which points to the file c:\test.txt. now we can add text or data to this file by using any of the following method.

- WriteLine(): it writes a single line of text to the file and shift the cursor to the next line in the file.
- Write(): it simply writes the text/data to the text file where the current cursor is.

Following is an example:

```
Dim objFile As New System.IO.StreamWriter("c:\test.txt")

objFile.WriteLine("text1")

objFile.WriteLine("text2")

objFile.WriteLine("text3")

objFile.Close()
```

Reading From a Text File

We use StreamReader object to read text from a text file. To perform this operation first of all we have to create a StreamWriter class object. It will be created as:

```
Dim objFile As New System.IO.StreamReader("c:\test.txt")
```

The difference between StreamWriter and StreamReader object is that, if file doesn't exist then StreamWriter will create the file but StreamReader will generate an error.

You can read entire file by using ReadToEnd() method or can read a file line by line by using ReadLine() method of the System.IO.StreamReader object.

```
strContents = objFile.ReadToEnd()

OR

strLine = objFile.ReadLine()
```

Activity 12

Create an application which implement a text box. It should read a file into this text box and user can make changes in this text and then it should write these modified lines to a new file. The form should have buttons to open a file, save a file, close a file and Exit the application.

8.2.3 Working with a Database, ADO.NET, Manipulating Data

Dear Students! We have discussed about the application development. We have discussed about simple applications and applications which can access and data stored in Text Files. The limitation of text file is that we cannot read and write at the same time in Visual Basic. It is also not possible to process data in different sequences and orders. If we want to process data in different sequences and in many ways then we use Database systems.

Most of the computer applications in our daily life we deal with different of data and information. The data may include Social Security Numbers, National Identity Card numbers, Names, Addresses, Dates, etc. In some business organizations like banks and airlines even more data is to be processed. We use database to manage such huge amount of data.

There exist many Database Management Systems like Microsoft Access, Oracle, Microsoft SQL Server, etc. These database systems are being used widely to handle and manage large amount of data. A Database Management Systems is used for storing, retrieving, deleting and modifying the data stored in a database. It is very difficult for a common person to handle data directly using DBMS. To make it easy for everyone, the developers develop database applications which perform different operation on data by providing a user friendly interface.

We will study this concept by building an application in Visual Basic (here we will discuss it for Visual Basic 2010). We will build a contact list project.

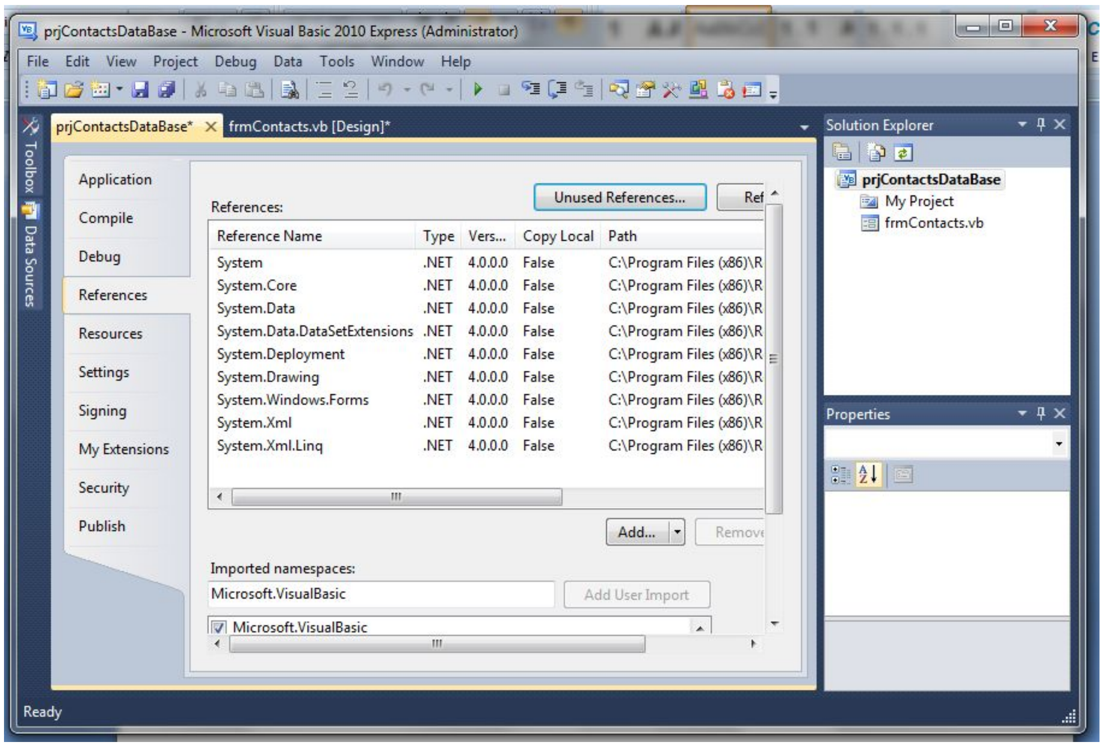
To deal with the databases, Microsoft Visual Basic provides ADO.NET technology. It is the latest technology which can handle most of the database systems available these days. We will also need to have some database system installed in or PC to work with ADO.NET technology. So for a better compatibility you may install Microsoft SQL Server on your PC. ADO.NET provides different objects which help us to play with data stored in a database. These objects include DataSet, DataTable, DataReader.

8.2.3.1 Starting a DataBase Project

Here we will learn that how to start building a database project/application in Visual Basic 2010. To build a database project in VB, start the Visual Basic 2010. Then chose New Project and give a descriptive name to your project, like prjContactsDataBase. The default form will appear. As we are going to build a contact list, so set the Text property of this form as Contacts. To build a database application we need some ADO.NET objects. These objects are:

- SqlConnection – this object is required to build a connection with Microsoft SQL Server.
- DataTable – this object is required to store the data fetched form a database table and then manipulate it.
- DataAdapter – this object is required to read the data using DataReader from a database table.

All the objects in a project in Visual Basic belong to some namespace. The above three objects belongs to System.Data and System.Xml namespace. So we have to include the references of these namespaces in our project before we can use them in our project. To create the reference of ADO.NET object, select the project menu and then chose prjContacsDataBase properties. Now the properties of our current project will be shown. From the properties dialog click the References tab; it will display the current reference available in this project as shown in Figure 15.



References:

Reference Name	Type	Vers...	Copy Local	Path
System	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Core	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Data	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Data.DataSetExtensions	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Deployment	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Drawing	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Windows.Forms	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Xml	.NET	4.0.0.0	False	C:\Program Files (x86)\R
System.Xml.Linq	.NET	4.0.0.0	False	C:\Program Files (x86)\R

Imported namespaces:
Microsoft.VisualBasic

☑ Microsoft.VisualBasic

Figure 15 Active Reference in Visual Basic Project

All the active references will have a check mark in the relative checkbox on their left in imported namespaces. To create our reference for ADO.NET we have to place a check mark in the boxes with System.Data and System.Data.SqlClient namespaces. After selecting these click on the Save All button from the toolbar.

8.2.3.2 Using ADO.NET to Create Connection to a Database

Now we will learn that how we can create a connection to a database with the help of ADO.NET object. To create a connection to a database it is required that the database must exist. So we have to create a database before we can use it in our project. As we will use Microsoft SQL Server as a database management system so we will create our database using this DBMS. We should use a descriptive name for this database. We will give dbContacts name to this database. When we create a database, it is basically a container. We have to create Tables to store data. In this database we create a table with the name tblContacts having two fields in it; ContactName and State. We enter few records in this table and then save it. Now we can create a connection to this database and can manipulate data stored in this database.

We can create a connection with the help of DO.NET with many different objects which ADO.NET provides. These objects include OleDbConnection, SqlConnection and many others. As we are using Microsoft SQL Server, we will use SqlConnection object of ADO.NET to build the connection to Microsoft SQL Server. First of all we'll create a SqlConnection object using the following syntax:

```
Private sqlCn As New SqlConnection()
```

After creating this connection object, we have to specify the data source to which we want to build connection. To specify the data source we use ConnectionString property of ADO.NET connection object. The connection string is an important property. It contains the information about connection, which includes: name of the provider, username, password and some other parameters. We must specify the ConnectionString property of our ADO.NET connection. We can do so by writing the following code in the Load event of our form so that when the form starts, this code execute.

```
SqlCn.ConnectionString = "Data Source=.\SQLEXPRESS; AttachDbFilename = " & _
"C:\Temp\dbContacts.mdf;Integrated Security=True; Connect Timeout=30;" & _
"User Instance=True"
```

We can replace "C:\Temp\" with the actual path where we have stored our database when creating it. When once we have defined the connection string then we can establish a connection to the data source as follows:

```
SqlCn.Open()
```

if we build a connection to an unsecured database then it is not required to pass the username and password parameters, otherwise we have to provide the username and password. The above connection string is for an unsecured database connection.

We can end the connection by using the following statement:

```
SqlCn.Close()
SqlCn.Dispose()
```

8.2.3.3 Data Manipulation

Data manipulation means that we can add new records to the data, modify the existing records, delete a record, search for particular records and moving around within the records. ADO.NET provides DataTable object to manipulate data conveniently. A DataTable object contains the records which are fetched from a table using some query or a procedure.

To fetch the data from a data source into a DataTable object we have to use another ADO.NET object which is called DataAdapter. DataAdapter is created by using the connection object which we have already created. Following is the syntax to create a Data Adapter:

```
Dim SqlDAr As New SqlDataAdapter([CommandText],[Connection])
```

The Command Text could be a query or SQL statement and connection is the connection object which we have created earlier.

To access data from our source database we will write the following code immediately after the command SqlCn.Open() in form Load event.

```
Dim SqlDA As SqlDataAdapter
Dim SqlCB As SqlCommandBuilder
Dim SqlDataTable As New DataTable
Dim SqlrowPosition As Integer = 0
SqlDA = New SqlDataAdapter("Select * From tblContacts", SqlCn)
SqlCB = New SqlCommandBuilder(SqlDA)
SqlDA.Fill(SqlDataTable)
```

SqlDA.Fill() method will fetch the data from the data source according to the query given in DataAdapter.

We can set the values of fields of a row and also can get the value of a field by creating a DataRow object and then referring the fields in that object. The following code is showing the code used for this purpose. This code will access the very first record of the DataTable object.

```
Dim SqlDataRow As DataRow = m_DataTable.Rows(0)
SqlDataRow("ContactName") = "Eddison Jhon"
strContactName = SqlDataRow ("ContactName")
```

Second line of the code will set the value of "ContactName" field whereas third line will get the value of "ContactName" field of DataTable object.

8.2.3.4 Navigating in a DataTable

Often we have to move to and fro in a DataTable to process the data. This is known as navigating through the data or table. ADO.NET provides different methods to perform the navigation through the data. We can access the fields of a particular row/record by using the following syntax:

```
strContactName =  SqlDataTable.Rows(SqlRowPosition)("ContactName").ToString()
```

now by incrementing or decrementing the value of "SqlRowPosition" variable and using the same line of code we can move to next record or previous record in a DataTable object and then can access the fields of that row for data processing purpose. We can create buttons on our form to navigate through the records.

To add a new record or row to a table we first have to call NewRow() method of the DataTable object. Then we assign the values to all the fields and then we call Add() method to actually post the data to the data source. If we do not call Add() method then the data will not be saved to the data source.

To delete a record we call Delete() method of the DataTable object with the row-position and call Update() method to make this change permanent in the data source.

Unit Microsoft Access (DBMS)

9 Getting Started, Getting Around (Chapter 16)

Learning Objectives

After the completion of this chapter students will be able to:

- Start Microsoft Access
- Open and existing database in Access
- Understand the Microsoft Access interface (Menus, Ribbons, Toolbars, etc.)
- Use different commands from Ribbon and Toolbars
- View and work with different objects in Microsoft Access
- Manage the database

Preface

Dear Students! Hope you are doing well in your IT learning sessions. We have discussed about the programming in previous sections. In the following sections you will learn different concepts about DataBase Management System (DBMS).

In our daily life we are always dealing with data. It may be intentional dealing or just be chance. In a class a teacher keeps data about his/her students like their attendance record, grades, marks, etc. We keep our contact lists either on paper, in our phone, electronic diary or even as simple as just try to remember in our mind.

Large companies and organizations spends huge amount of money to manage data of their customers, employees, inventory, sales and other transactions. To manage all these data some DBMS is being used.

You will learn how to handle, manage and manipulate data using a DBMS. We will discuss about different components of DBMS. You will learn that how to conceptualize data hierarchy. We will use Microsoft Access 2010 as our DBMS for discussion and learning. So you will become familiar with the interface and working of Microsoft Access 2010. We will discuss that how we can design a database, tables, forms, queries and reports using Microsoft Access 2010.

9.1 Introduction

In our daily life applications we mostly deal with data. We need to store these data, retrieve these data and manipulate these data. To store these data we use the files and to use files we have to use the file system operations. DBMS provides us an easy way to perform all operations required to store and manipulate data. Whenever we are going to use a new application, then it is important to become familiar with the interface of that application. If you become familiar with the Microsoft Access interface then it will make your learning straight forward. In this chapter we will get familiar with the initial steps required to start and manage objects in Microsoft Access 2010. You will become familiar with Ribbon, Navigation Pan, Record Navigation bar, etc.

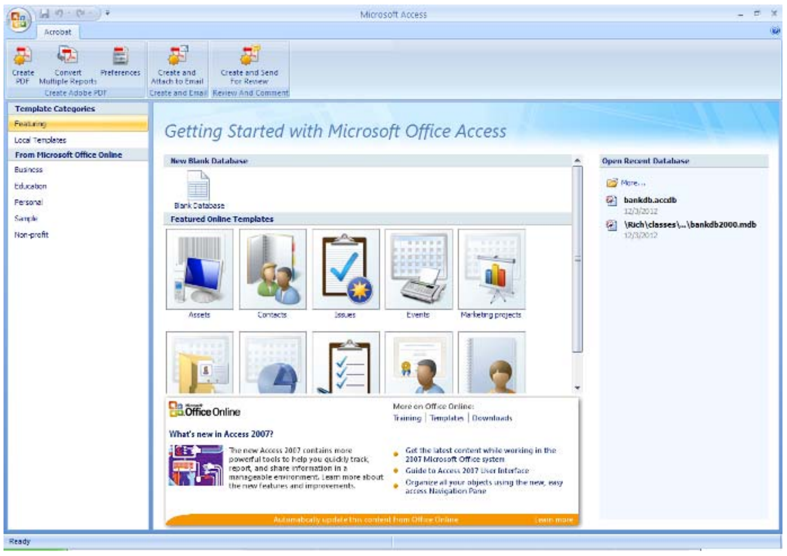
9.2 Getting Started, Getting Around

9.2.1 Running Microsoft Access

Microsoft Windows provides different ways to run or start the same application. The same is true for Microsoft Access. To run Microsoft Access follow the steps:

1) Click *Start*.
2) Choose *All programs*.
3) Choose *Microsoft Office*.

When you complete the above four steps the PC will start processing to run Microsoft Access. After some time (depending upon PC configuration) the initial Microsoft Access screen will be shown as displayed in Figure 16. This is known as *BackStage View*. This view offers you the choices for creating a new database or open an already build database.



What's new in Access 2007?

The new Access 2007 contains more powerful tools to help you quickly track, report, and share information in a manageable environment. Learn more about the new features and improvements.

- Get the latest content while working in the 2007 Microsoft Office system
- Guide to Access 2007 User Interface
- Organize all your objects using the new, easy access Navigation Pane

Figure 16 Initial Screen of Microsoft Access[13]

[13] http://holowczak.com/microsoft-access-2007-and-2010-tutorial/4/

There may be other ways; you may have a shortcut of Microsoft Access on your desktop and when you double click that icon the Microsoft Access will start; if you double click a database already created in Microsoft Access, it will also start Microsoft Access.

9.2.2 Opening a Database

If we have created a database in Microsoft Access and we want to work further on it then first of all we have to open it in Microsoft Access. Microsoft Access has the limitation to open only one database at a time. If you try to open second database at the same time then Microsoft Access will automatically close the previously opened database.

There are more than one methods to open an existing database. One of the methods is described in the following steps:

1) Click the tab *File* on the Microsoft Access Ribbon.
2) Click *Open*.
3) The dialog box *Open* will be shown, now choose the desired path and filename and click *Open*.

After performing the above 3 steps, Microsoft Access will open the selected database. Sometimes Microsoft Access shows some warning message, generally it will be about disabling some contents of the database. You can respond to these messages as per situation. If you are going to open a database which you have used recently, then you can choose from the list of recently opened files because Microsoft Access keeps the history of last 4 opened databases. These recently used databases also appear in the *Backstage View*. You can open these by just double clicking the required database.

Another method to open a database is just double click a database in Windows Explorer, then Microsoft Access will run and the database will be opened.

When you open a database then different components and pans appear in the Microsoft Access window. It all depends upon the database that what will be displayed. Some database may display a navigation pan while other may display a startup form. This startup form may require some login information and may perform some tasks.

You can open the older version database in Microsoft Access 2010. Microsoft Access 2010 and Microsoft Access 2007 have the same file format, so the files created in these two versions can be opened alternatively but the files created in these two versions cannot be opened in Microsoft Access 2003 or earlier. The files created in Microsoft Access 2003 or earlier can be imported in these two versions of Microsoft Access.

9.2.3 Creating a sample database from a template

Hello! As we have discussed a lot about data, database, DBMS, and Microsoft Access. Now you certainly want to explore things in a database in Microsoft Access 2010. To do this we must have a database. To have a database we have to create it. As we are in learning phase, so we can create a database using any of the templates provided by Microsoft Access 2010. These templates are installed with the Microsoft Access when we install Microsoft Office. To create a database using template perform the following steps:

1) Run Microsoft Access if you have not opened it already.
2) Choose *New*.
3) In the Templates area choose *Sample Templates*.
4) Choose the template which you want to use to create your database.
5) Click *Create*; Microsoft Access will create the data base. If you are using Windows XP then the database will be created in the folder MyDocuments and if you are using later version of Windows then the folder will be created as per the template you selected. For example if you have selected Contacts template then the folder will be created with the name Contacts.accdb.
6) If Microsoft Access shows Security warning then click *Enable Content*.

In this way you will have a database with which you can play and try all the feature and functions of Microsoft Access as a DBMS.

Activity 13

Create a database for storing contacts using template in Microsoft Access 2010. Explore different components of the database and use different features of this database.

9.2.4 Viewing Objects in Your Database

There are many different types of objects in a Microsoft Access database. General they fall in six different categories. These categories are:

- Tables
- Forms
- Queries
- Reports
- Macros
- Procedures and Modules (VBA code)

You can open any of this by choosing the object category tab. And then you can open any object as per screen-format. For example you can open a Form in design view or in running view. We can also view an object by double clicking it in Navigation Pane. When we double click an object, Microsoft Access will open that object in its default view, then you can change its view. For example a form can be opened in Design View as shown in Figure 17.

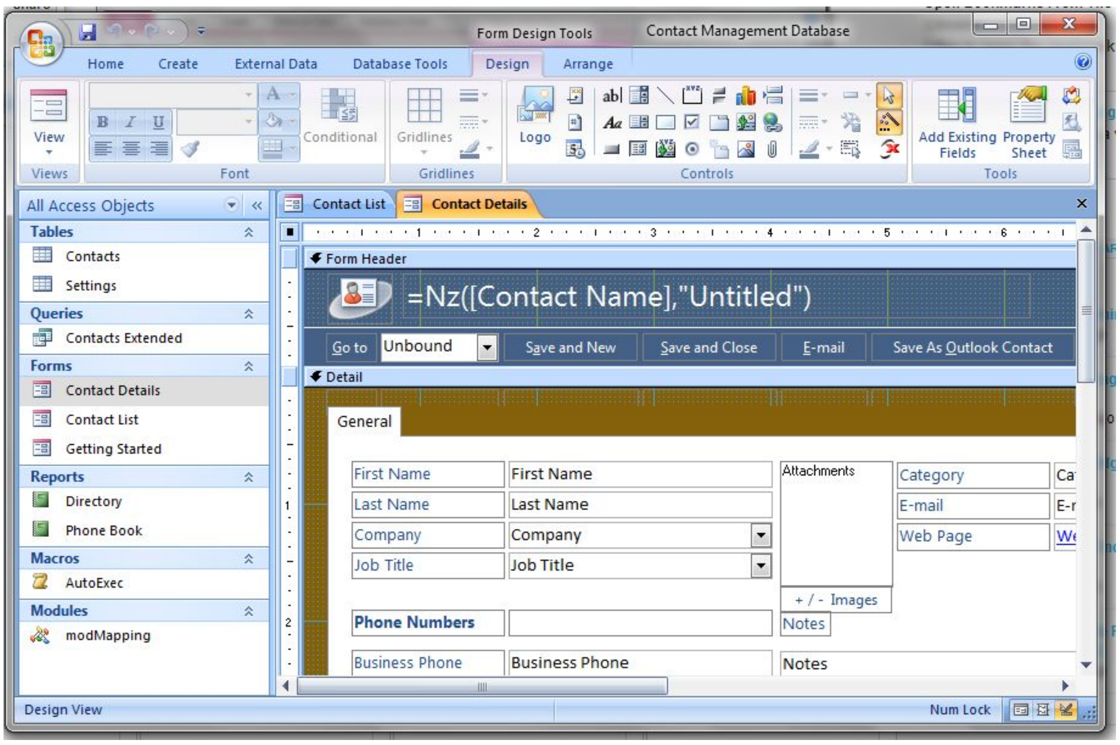


Figure 17 *Viewing* Microsoft Access objects

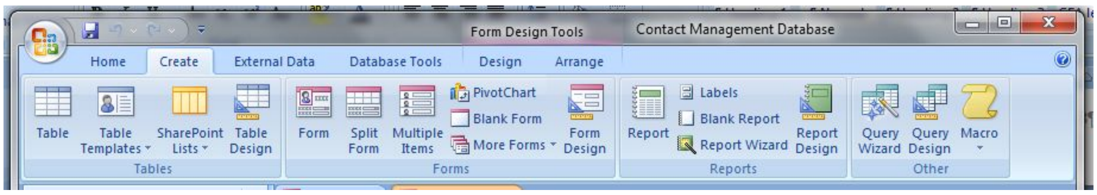
If you have opened an object then at some point you certainly want to close this object. For this purpose simply click X button in the control box of the window showing that object. You can also close the object by right clicking it and then choosing close from the popup menu.

9.2.5 Creating, Deleting, Renaming, Copying, and Printing Objects

We have discussed about database and objects present in a database. We can create new instances of these objects. We can delete, copy, rename or print these objects as per the requirements of our task and database. Here is a brief introduction of these operations in Microsoft Access.

To create a new object follow the steps given below:

1) Locate and click the Tab *Create* from the Ribbon.
2) Select the desired type of object to be created.



3) Select the option to create an object using wizard or in *Design View*.
4) Then follow the steps as per guided by the Microsoft Access.

You will get the desired object created.

To delete an object from the database follow the steps given below:

1) Select the object in the Navigation Pane.
2) Press <DEL>.
   OR
1) Right click on the object in the Navigation Pane.
2) Select the option *Delete* from the pop-up menu.

You will get the object deleted. Keep in mind that if you delete a table then its data will also be deleted.

To rename an object
To rename an object in the database follow the steps given below:

1) Click the object name in the Navigation pane and press <F2> or right click the name and choose *Rename* from the pop-up menu.
2) You will see a text box containing the name of the object which will be editable.
3) Now simply type the new name you want to give that object.

You will get the object renamed.

To copy an object
You can copy an object in the *Navigation Pane* just like the operation in other Windows applications. Select the object to be copied and then press <Ctrl+C> to copy and <Ctrl+V> to paste. You can also use right-click and choose *Copy* and then right-click and choose *Paste*.

To Print an Object
To print an object in the database follow the steps given below:

1) Select or open the desired object.
2) Press <Ctrl+P> OR select the tab *File* and then choose *Print* as per your convenience.
3) Choose the appropriate printer and printer setup.
4) Click *Print*.

You will get the object printed.

9.2.6 Using Wizards

We can design or create an object in two ways. One way is design the object from scratch by using design view. The other way is by using Wizards. Wizards are programs which guide us step-by-step getting some information from us and create the desired object according to the information provided by us. Generally wizard gets information from us one to two at a time in a dialog box and then we click the Next button to proceed to the next step. This process is continued until wizard has gathered all the required information from us. Then at the last step there is a Finish button. When we click that Finish button the wizard creates the object. We can create Tables, Forms, Queries and even databases using wizards.

9.2.7 Getting Help

In the computer world, almost any organization who have developed some application or program, has also included its help and guideline for using it within that program. The same is the true Microsoft Access. The help in Microsoft Access is a very useful and helping feature. Microsoft Access provide different ways to get information from it.

You can get help by clicking the Question mark in the upper-right corner of the Microsoft Access window. You can achieve the same goal by pressing <F1>.

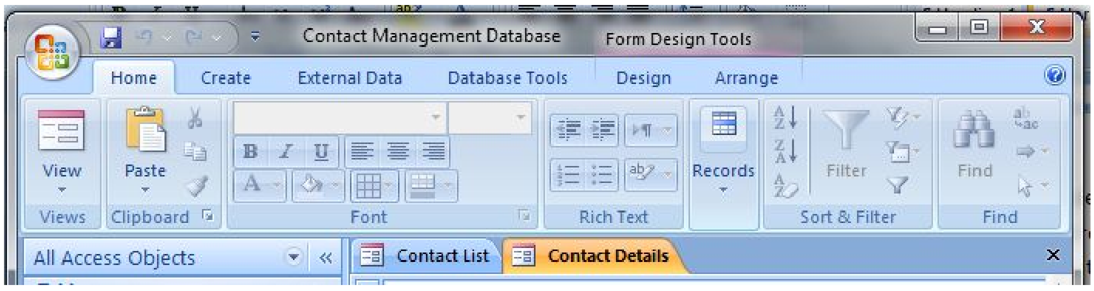


Figure 18 Mark in the upper-right corner of the window

Then *Help dialog* box will appear. You can type and question or a keyword in the text box provided and then press enter or click the search button as shown in Figure 19.

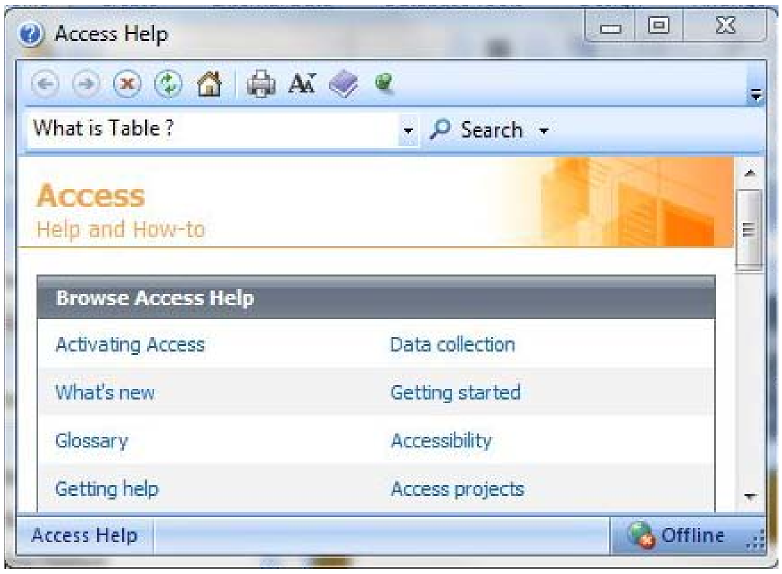


Figure 19 Help Dialog Box

If you want to get help on a specific topic then you can click on any of the topic presented in the *Microsoft Access Help dialog box*.

Activity 14

In Activity 13 you have created a database for storing contacts using template in Microsoft Access 2010. Explore different components of the database and Add, Delete, Rename and print different objects. Also get Help about different components using Microsoft Access help.

10 Designing Database the Relational Way (Chapter 17)

Learning Objectives

After the completion of this chapter students will be able to:

- Design and create a database
- Design and create tables
- Choose the right data types for fields in the tables
- Refine the database design to make it a Relational Database
- Create the relation between tables using joins to make the database relational
- Use templates and wizards provided by Microsoft Access.

Preface

To design a database means that how the data will be stored, how the data will be split in different tables and fields so that the data can be accessed in a structured way. In this process we decide that how many tables will be there in the database, how many fields different tables will have and what will be their data types. We also take decisions about primary keys and foreign keys for each table. We decide that how different tables will be joined to relate different information in different tables. If the database is designed in this way then it will be easy to maintain it and process the data efficiently.

10.1 17.1 to 17.5

These sections have been discussed in the class sessions. These sections discussed the concepts about tables, fields and keys. These also discussed about different data types which we can assign to fields of tables. The relationship types have also been discussed. We have studied that how we can break up our data to form different tables and then linking these tables.

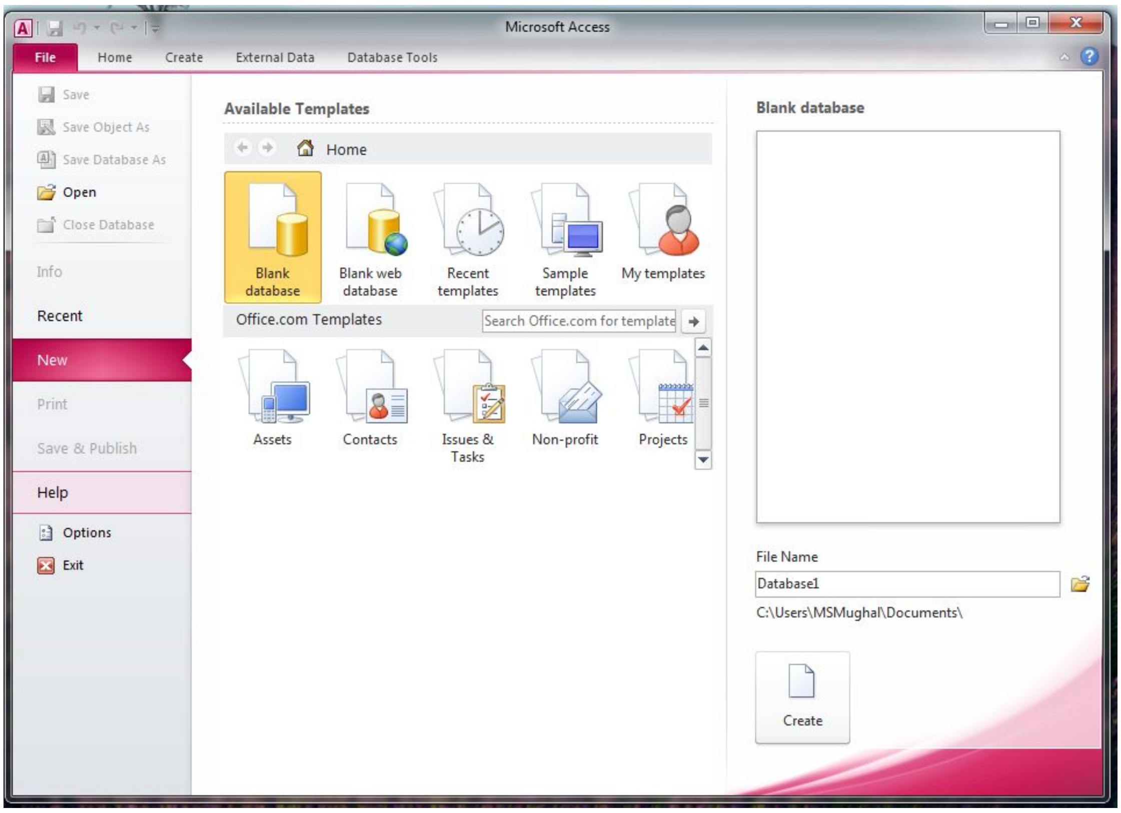
10.2 Creating a Database

In the previous sections we have discussed that how we can design a database. If we have designed the database carefully then we can create it using any DBMS. We will learn that how we can create a database in Microsoft Access. A Microsoft Access database is certainly a Windows file. When we create it, then it is stored at some location called path on our PC.

10.2.1 Creating a Database from Scratch

If your database design has very specific requirements and specifications then it will be a good idea to create a database from the scratch. In this way you can create a database with too much flexibility and specifications. To create a database from scratch in Microsoft Access follow the steps described below.

1) Click the tab *File* on the Ribbon.
2) Click *New*.
3) Choose *Blank Database*.
4) Choose the path to store this database.
5) Type a name for the database.
6) Click *Create*.



File Name
Database1
C:\Users\MSMughal\Documents\

Figure 20 Screen to create Database from scratch.

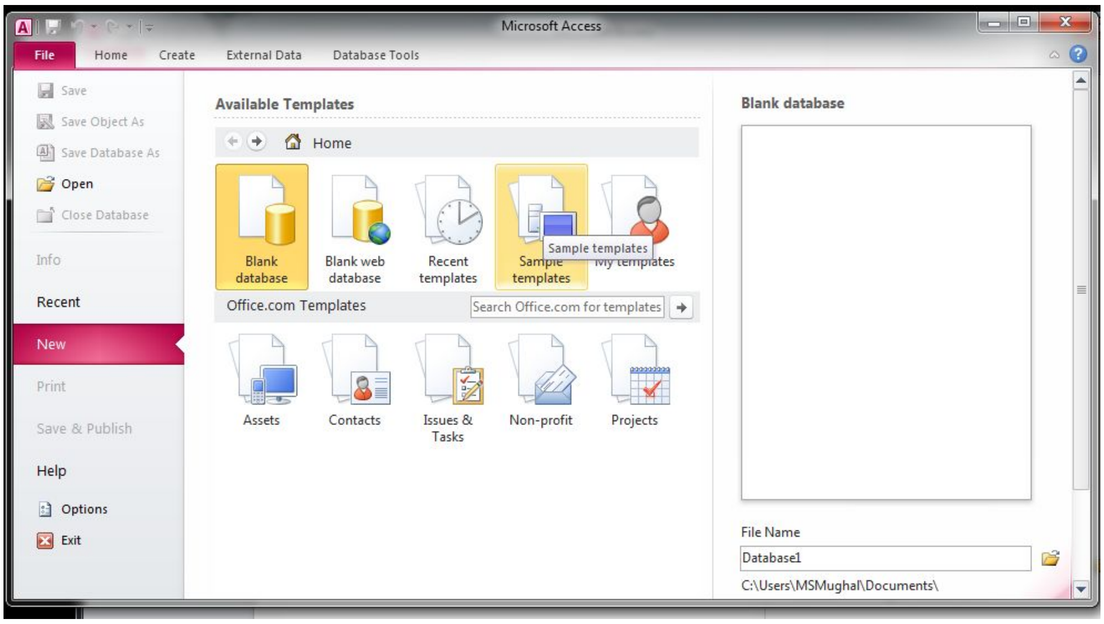
A blank database will be created and opened. A table will be added automatically to this database having the name "Table1". You can add more tables and other objects to this database as per your requirements and specifications.

10.2.2 Creating a New Database Using a Template

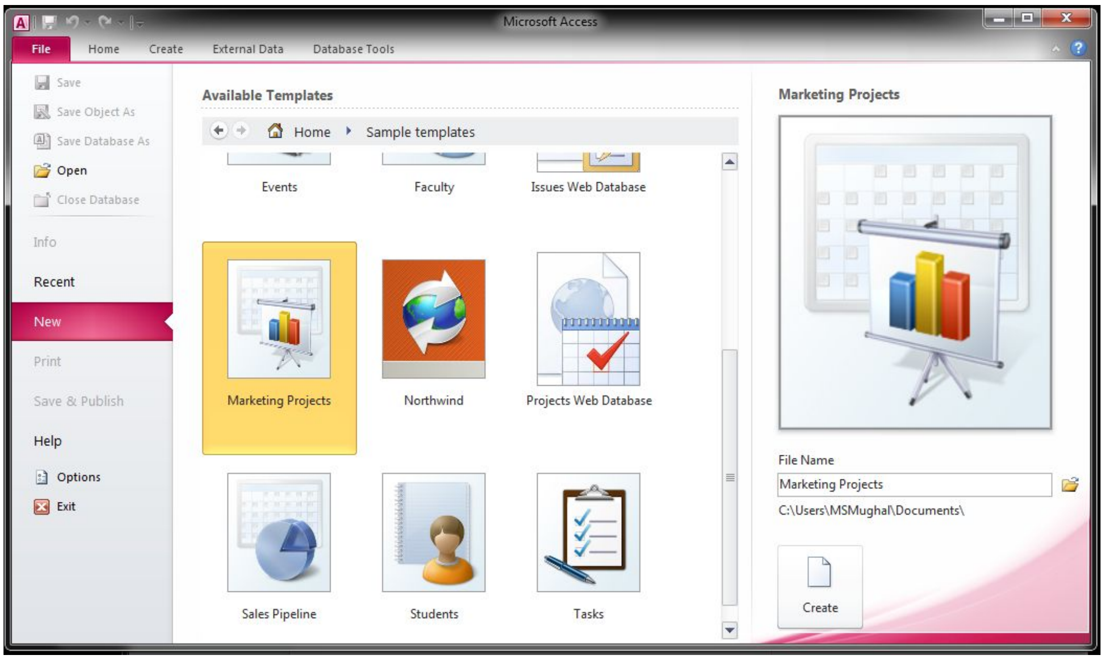
If you have designed a database in such a way that it resembles to any of the templates provided by Microsoft Access then you can use that Template to create your database. It is a convenient way to create a database.

To create a database using a template follow the steps given below.

1) Open Microsoft Access on the start page.
2) Click the tab *File* from the Ribbon and then select *New*.
3) Click *Sample Templates*.



4) Choose the template.



5) Choose the path where you want to store this database.
6) Type the name for the database.
7) Click *Create*.

Your database will be created according to the selected Template. Now you can use this database and its objects or you can modify the objects created by the Templates. You can also add new objects or can delete any of the objects as per your requirement. If the database created using template doesn't fulfill your requirements then simply delete it and try some other template or build a database from scratch.

Task 4

1) Create a database "Marketing Projects" using templates and then input some sample data.
2) Create the same database starting from scratch and then add also those objects which were added by the template.

11 Avoiding the Garbage Data (Chapter 19)

Learning Objectives

After the completion of this chapter students will be able to:

- Design and create tables with proper fields
- Create fields with proper data type to avoid data entry of wrong types
- Create input-masks to get proper data in proper format from the user
- Understand that what are the look up lists
- Create look up lists to get pre-defined values as input in a field
- Create validation rules to validate data at the time of input

Preface

Any database application is designed to store and manipulate data. Certainly the data stored in the database comes from the user. User can enter the data through GUI known as forms or can enter data directly into tables. User may enter any data which may be irrelevant or may be of wrong data type or may not match the required format or may not follow some limits and restrictions. Use may do this intentionally or just by mistake. So the data entered may be garbage. If the data entered is garbage then after processing the results will certainly be garbage.

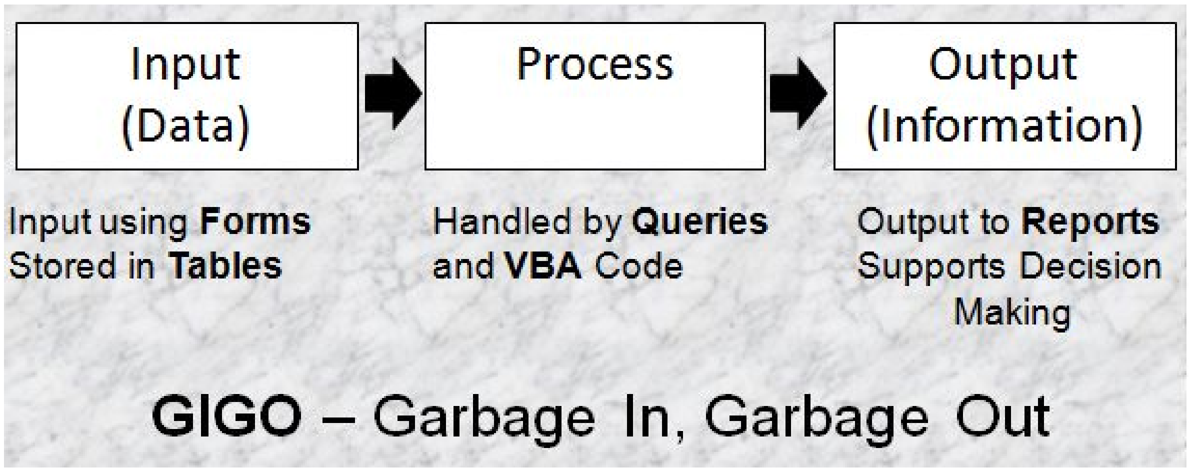


Figure 21 Garbage In, Garbage Out

We should design our database and table in such a way that we can avoid garbage data entry into the tables. Microsoft Access provides may options to do this. In the following sections we will discuss about the solutions, tool and techniques which we can use to prevent garbage data entry.

11.1 Using Input Masks to Validate and Format Data

All the database applications are intended to use by the people who have not developed it. Generally they are not aware of the rules and data formats to be entered. So different users can enter data in different ways and formats. It is the duty of the developer that he/she should design and develop the application in such a way that all users enter the data in the same type and format. For this purpose Microsoft Access offers the facility of Input-Mask.

Input mask controls the data entry by limiting the data entry to some specific characters and format. These are the best tool to format and limit the data if we know the data format in prior. For example the Telephone numbers and Zip codes have specific formats so we can use input masks for the fields in which such information is to be entered. Input mask is useful both to format the data and restrict the type of characters and their order in which they must be entered. Input masks provides the formatting of data by inserting punctuation or some other characters, for example adding "," in currency value and displaying "*" instead of text in password type fields. If the data to be entered in a field has no specific order or format then it is not a good candidate to use an Input Mask with it. Input Masks can be used with Text, Number, Date/Time and currency type data type of fields. If you choose other data type for your field then Input Mask property will not be available for these fields.

We can define the Input Mask for a field when we are in Design View. When we define Input Mask in design view then it will become a part of that field and table and it will validate the data both in direct entry to the table or when data is entered using a form.

An Input Mask consists of three parts. It has the format like: Part1 ; Part2 ; Part3 . The three parts within the mask string are separated by semicolon. From these three parts, Part1 is mandatory while other two are optional. These three parts are described below:

- **Part1:** It is the mandatory part. This part contains the mask string (consisting of 1 or more than 1 characters). This part may also include some placeholders, parenthesis, hyphens, periods or other literals.
- **Part2:** This is the second part which is optional. It contains a "0" or "1" only. The values "0" or "1" tells that how the embedded characters will be stored in the field. "0" means that the characters will be stored with the data and "1" means the characters will be displayed only and will not be stored. So you can save some storage space by setting the value to "1"
- **Part3:** As mentioned above, this part is also optional. It contains a single character which may be just a space as a placeholder. Normally Microsoft Access uses "_" as placeholder but you may use some other character and place that character in this part of the Input Mask.

Lets discuss an example. Consider the input mask **(999) 000-000;0;-.** Its three parts are elaborated as under:

- First part is a string consists of some 9s and 0s. These are used as placeholders. 9 is used for an optional digit and 0 is used for mandatory digit.
- In the second part 0 is used. It means that mask characters will be stored in the field along with the data entered.
- "-" is used in the third part. It means "-" will be used as placeholder character instead of "_" when the data is entered.

Table 6 describes different characters being used to create input masks. This table also shows what a character means and what is its role if used in creating an Input Mask in Microsoft Access.

Character	**Explanation**
0	User must enter a digit (0 to 9).
9	User can enter a digit (0 to 9).
#	User can enter a digit, space, plus or minus sign. If skipped, Access enters a blank space.
L	User must enter a letter.
?	User can enter a letter.
A	User must enter a letter or a digit.
a	User can enter a letter or a digit.
&	User must enter either a character or a space.
C	User can enter characters or spaces.
. , : ; - /	Decimal and thousands placeholders, date and time separators. The character you select depends on your Microsoft Windows regional settings.
>	Coverts all characters that follow to uppercase.
<	Converts all characters that follow to lowercase.
!	Causes the input mask to fill from left to right instead of from right to left.
\	Characters immediately following will be displayed literally.
""	Characters enclosed in double quotation marks will be displayed literally.

Table 6 Input Mask characters[14]

We can define Input Mask in two ways:

- Using Input Mask Wizard
- Manually create an Input Mask

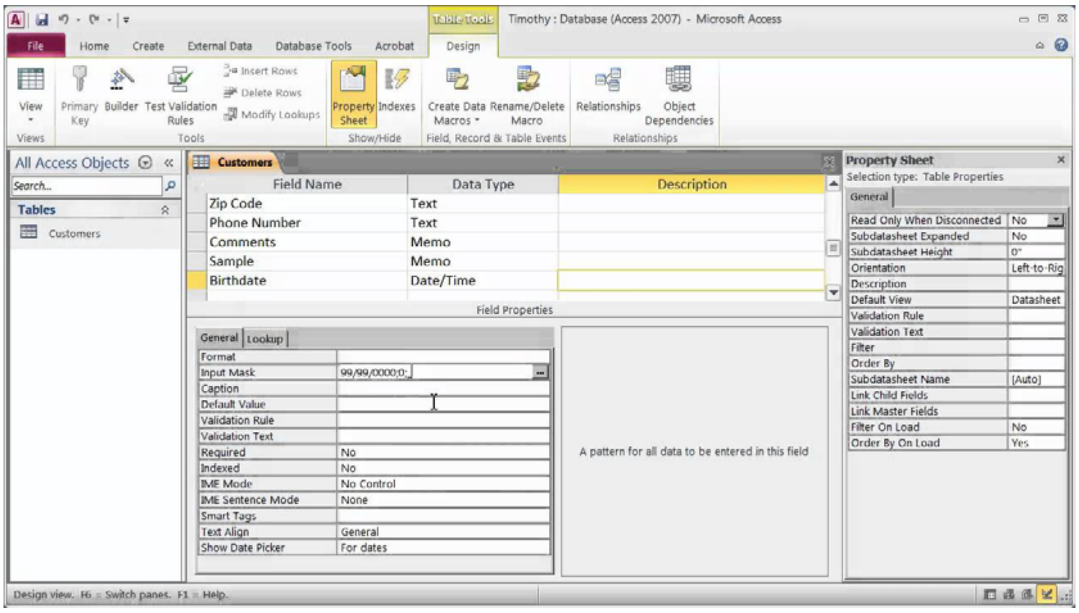
[14] http://office.microsoft.com/en-001/access-help/control-data-entry-formats-with-input-masks-HA010096452.aspx

11.1.1 Using Input Mask Wizard

Microsoft Access provides Input Mask wizard to define an Input Mask in a convenient way. Wizard guide us step-by-step to define the Input Mask. Input Mask wizard is very much helpful if your data is of the type and format which are used very commonly, for example a specific date format, phone number, a zip code, social security number, etc. In such cases we create the Input Mask using wizard and then can edit the mask to make it the way we exactly want the functionality from it.

The following steps describe that how we can use Input Mask wizard to create an Input Mask:

1) Select the table for which fields you want to create the input mask.
2) Open the table in *Design View*.
3) Select the required field for which Input Mask is required. When you select the field, its properties will be displayed.
4) From the properties click the tab *General*, then click the property *Input Mask*.



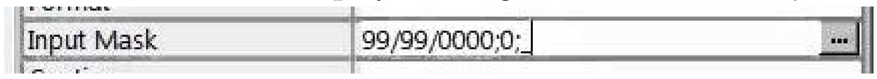
Field Name	Data Type	Description
Zip Code	Text	
Phone Number	Text	
Comments	Memo	
Sample	Memo	
Birthdate	Date/Time	

Format	
Input Mask	99/99/0000;0;_
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	No
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	
Text Align	General
Show Date Picker	For dates

A pattern for all data to be entered in this field

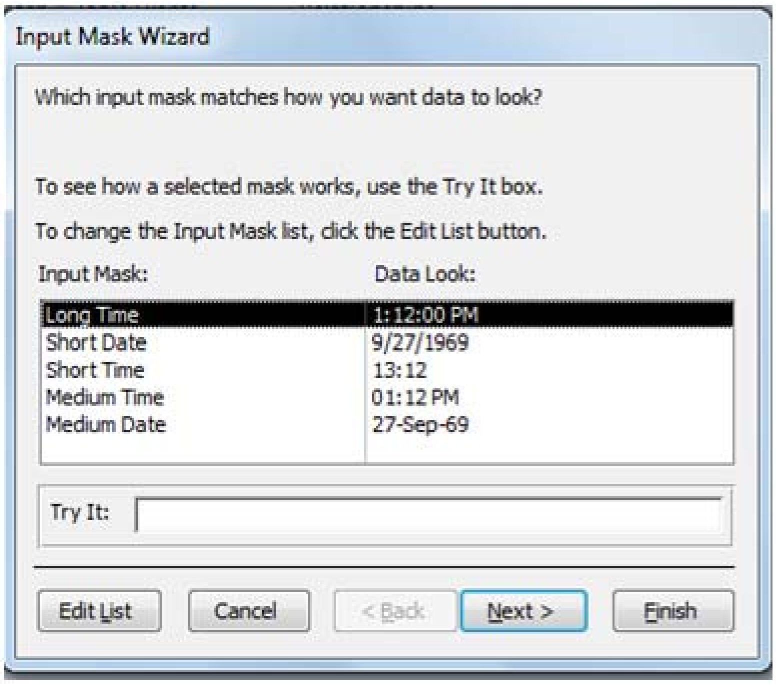
Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0"
Orientation	Left-to-Rig
Description	
Default View	Datasheet
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes

5) A button ... will be displayed on right side of the field *Input Mask*.



Input Mask	99/99/0000;0;_

6) Click this button to launch the *Input Mask Wizard*.


Input Mask Wizard

Which input mask matches how you want data to look?

To see how a selected mask works, use the Try It box.

To change the Input Mask list, click the Edit List button.

Input Mask:	Data Look:
Long Time	1:12:00 PM
Short Date	9/27/1969
Short Time	13:12
Medium Time	01:12 PM
Medium Date	27-Sep-69

7) From this dialog box you can choose an input mask which you think is suitable for you data input.
8) Now click in the box *Try It* and enter some data which you want to input in this field. Here you can test how the data will look using this input mask.
9) If you are satisfied in step 7, then click *Next*. You will be asked some more questions depending upon the data and Input Mask you selected.
10) You may have the option to modify this selected input mask if required. (We will discuss this editing in section 0)
11) Microsoft Access will offer you the option to select some place holder characters like punctuation marks. You can add these placeholders to your field. After choosing a placeholder click *Next*.
12) At the final screen you will be asked that how you want to store the data in the field. You may save the data entered only or you may also save the placeholders in the field. Generally you don't need to store the placeholders as they will be displayed with the input mask created every time the data is displayed. Selecting appropriate option click *Next*.
13) Now click *Finish* to place the created input mask in the input mask property of the field.
14) Now save the table to make this change permanent.

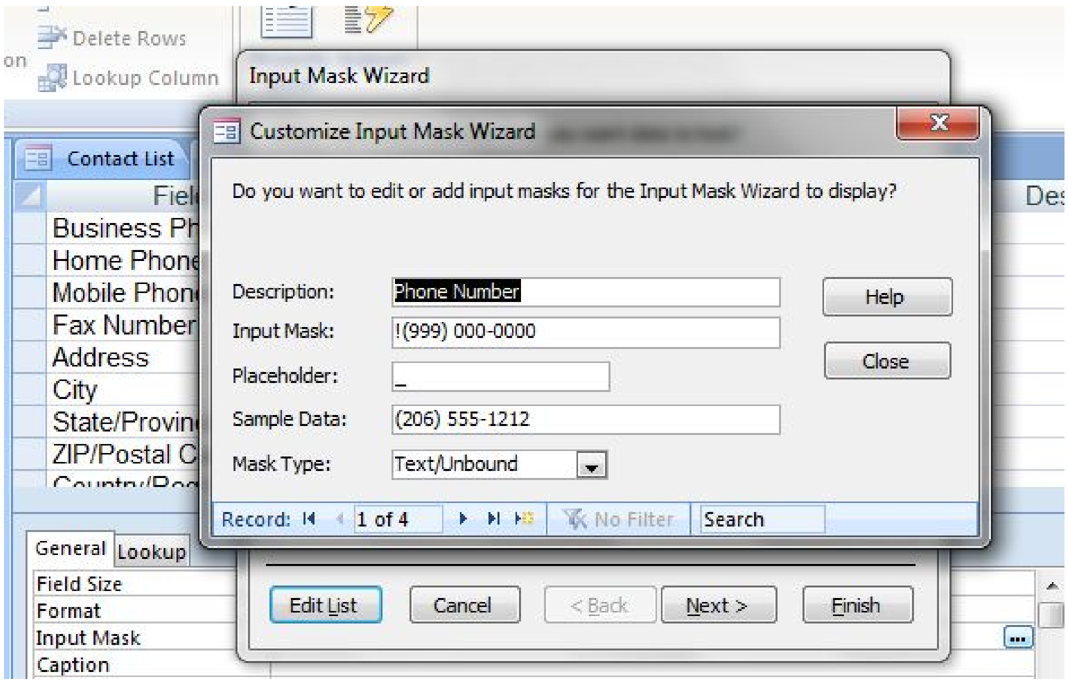
You have created an input mask by using input mask wizard successfully.

11.1.2 Manually create an Input Mask

Dear students! We have discussed to create an Input Mask using wizard. Now we will discuss that how we can create an Input Mask manually. To do this, we just have to enter the previously discussed three parts of the input mask in input mask property box of the field.

The following steps describe that how we can create Input Mask manually:

1) Select the table for which fields you want to create Input Mask.
2) Open the table in *Design View*.
3) Select the required field for which Input Mask is required. When you select the field, its properties will be displayed.
4) From the properties click the tab *General*, then click the property *Input Mask*.
5) Now enter the Input Mask string in this box.
6) You can also do this by clicking the button *Edit List* in step 5 and then enter the string in *Input Mask box*.



Do you want to edit or add input masks for the Input Mask Wizard to display?

Description: Phone Number
Input Mask: !(999) 000-0000
Placeholder: _
Sample Data: (206) 555-1212
Mask Type: Text/Unbound

Table 7 shows and explains some Input Mask examples.

This input mask	Provides this type of value	Notes
(000) 000-0000	(206) 555-0199	In this case, you must enter an area code because that section of the mask (000, enclosed in parentheses) uses the 0 placeholder.
(999) 000-0000!	(206) 555-0199 ( ) 555-0199	In this case, the area code section uses the 9 placeholder, so area codes are optional. Also, the exclamation point (!) causes the mask to fill in from left to right.
(000) AAA-AAAA	(206) 555-TELE	Allows you to substitute the last four digits of a U.S. style phone number with letters. Note the use of the 0 placeholder in the area code section, which makes the area code mandatory.
#999	-20 2000	Any positive or negative number, no more than four characters, and with no thousands separator or decimal places.
>L????L?000L0	GREENGR339M3 MAY R 452B7	A combination of mandatory (L) and optional (?) letters and mandatory numbers (0). The greater-than sign forces users to enter all letters in uppercase. To use an input mask of this type, you must set the data type for the table field to **Text** or **Memo**.
00000-9999	98115- 98115-3007	A mandatory postal code and an optional plus-four section.
>L<??????????????	Maria Pierre	A first or last name with the first letter automatically capitalized.
ISBN 0-&&&&&&&&&-0	ISBN 1-55615-507-7	A book number with the literal text, mandatory first and last digits, and any combination of letters and characters between those digits.
>LL00000-0000	DB51392-0493	A combination of mandatory letters and characters, all uppercase. Use this type of input mask, for example, to help users enter part numbers or other forms of inventory correctly.

Table 7 Examples of Input Masks with explanation.[15]

From these examples you can get guidelines to create your custom Input Masks.

[15] http://office.microsoft.com/en-001/access-help/control-data-entry-formats-with-input-masks-HA010096452.aspx

Activity 15

Create a table in any database with different data types in fields. Apply all the input masks explained in Table 7. Check their behavior. Also check their behavior by modifying these.

11.2 Creating a Lookup Field

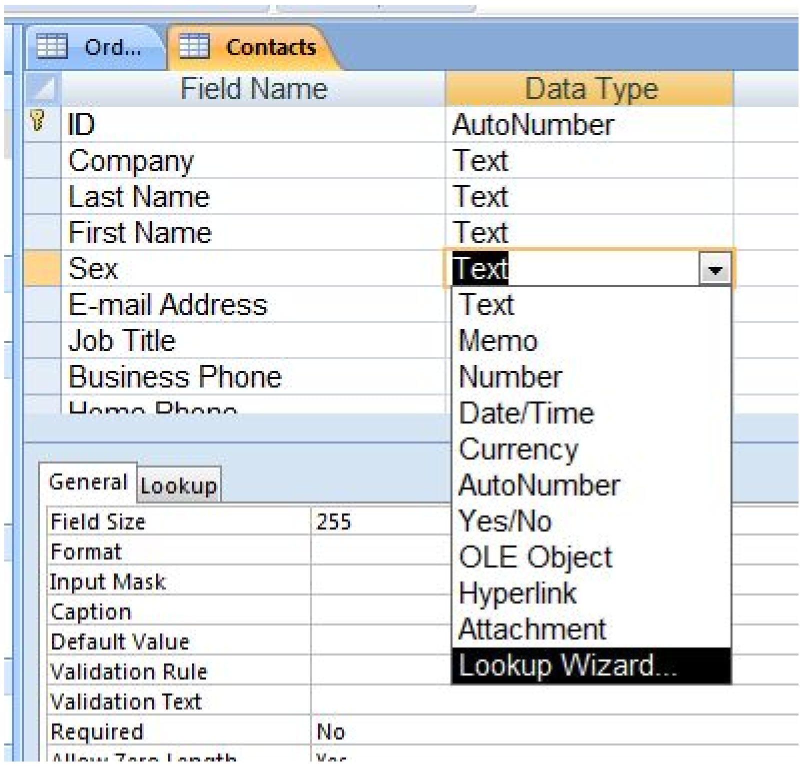
Whenever a database application is developed, it certainly meant to manage and manipulate data. The data mostly comes from the user either direct entry in the table or from a form. There may be many users who use the same application. Now it is very important and critical that all the users enter data with the same pattern and with the same valid values. To make it sure and convenient for the users, it is best choice if we can provide them with the valid lists of values for a field or data item. Microsoft Access provides the facility to create Lookup fields for this purpose. By creating lookup fields we can improve the data integrity, data accuracy and efficiency of data entry. It makes the use of database application very easy and the data entered is always consistent because the user is provided with the options to select rather than to type the value.

A lookup field in a table or a form displays a list of valid data values to the users which user can select to be placed in that field. The values displayed by a lookup field may come from a list we type in or from a table or query. It is always a good practice if we keep our lookup values in a table instead of typing a list. In this way it is more flexible to change the list of values or modify the values without the need of changing the table and field in design view. We can create a lookup field very easily by using Lookup Wizard. We can create a Look up fields in two different ways described below:

11.2.1 Create Lookup Field By Typing in the List

If we are going to create a Lookup field using this option, then we have to type in the list. We will discuss this by following the example. For this open the "Contacts" database provided by Microsoft Access as sample database. Here we will add a Sex field to the contacts table and then add a lookup field which will show a list, "F" for female and "M" for male. Follow the following to create lookup field:

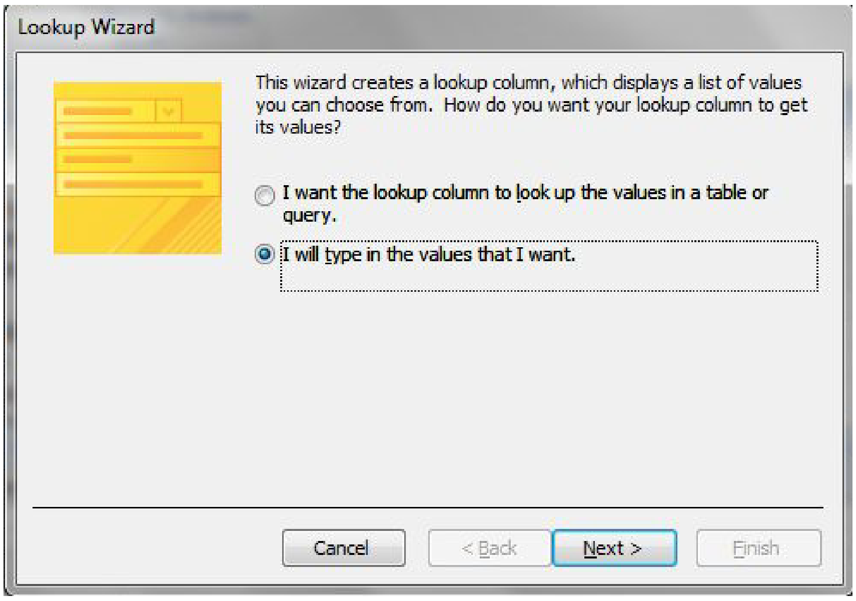
1) Select the table *Contacts*.
2) Open it in *Design View*.
3) Now click the *Sex field* (If it doesn't exist then add a field with this name).
4) In *Data Type* select *Lookup Wizard*.



Field Name	Data Type
ID	AutoNumber
Company	Text
Last Name	Text
First Name	Text
Sex	Text
E-mail Address	
Job Title	
Business Phone	

Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No

5) This will launch the *Lookup Wizard*.

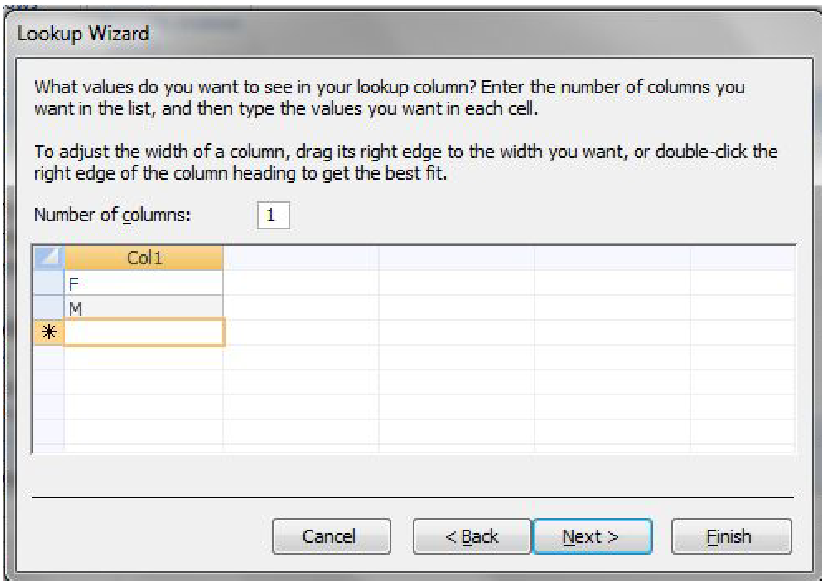

Lookup Wizard

This wizard creates a lookup column, which displays a list of values you can choose from. How do you want your lookup column to get its values?

I want the lookup column to look up the values in a table or query.

[✓ I will type in the values that I want.]

6) Choose *I will type in the values that I want* and click *Next*.
7) Enter *1* in the *Number of Columns box* and enter *F* and *M* in two rows in the provided place as shown below:


Lookup Wizard

What values do you want to see in your lookup column? Enter the number of columns you want in the list, and then type the values you want in each cell.

To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

Number of columns: 1

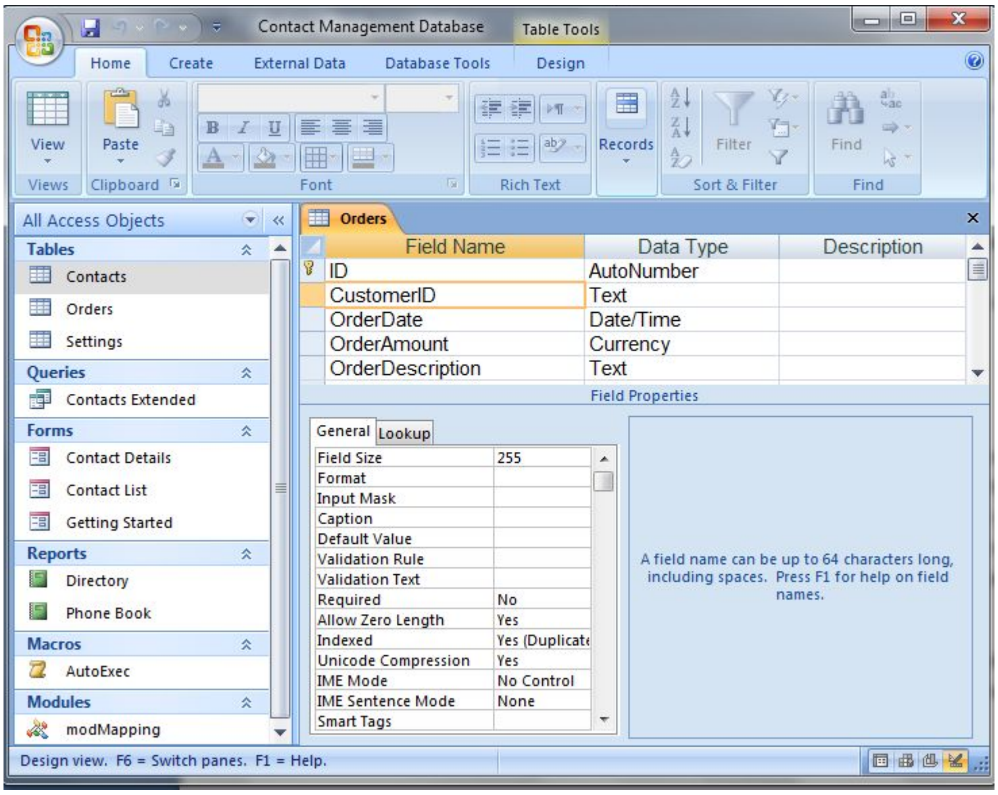
Col1
F
M

8) Then click *Next*.
9) Next Screen will ask you for a label for this lookup column/field. *Sex* will already be shown, type in if you want something else. Click *Finish*.
10) Save the table and view it in *Datasheet View*. You can check that in *Sex field* a drop down list will be displayed for the F and M values.

We have created a lookup field successfully by entering a list.

11.2.2 Create Lookup Field by selecting the Values from a Table

When creating Lookup field using this option, then we have to mention some table or a query from which we will populate our lookup field list. We will discuss this by following the example. For this open the "Contacts" database provided by Microsoft Access as sample database. Add a table orders as shown in the Figure 22. We will create lookup field for *CustomerID* for which the values will come from the *ID* field of *Contact Table*.



Field Name	Data Type	Description
ID	AutoNumber	
CustomerID	Text	
OrderDate	Date/Time	
OrderAmount	Currency	
OrderDescription	Text	

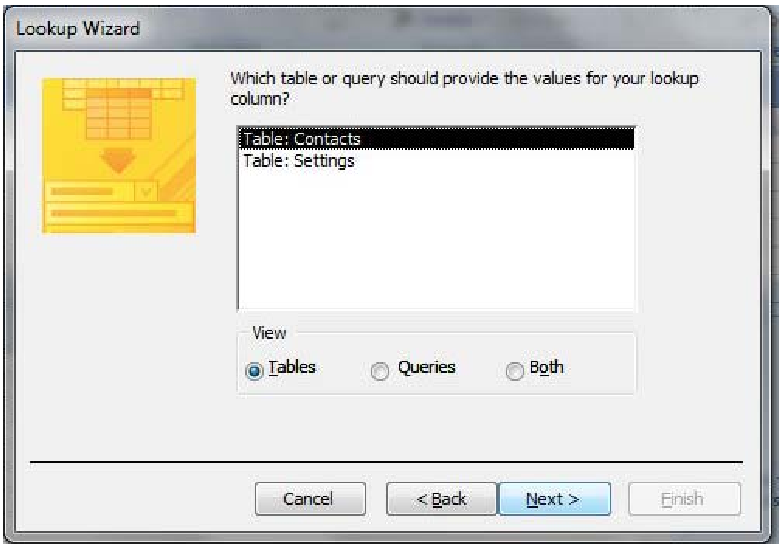
Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	Yes (Duplicate
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Figure 22 The structure of table "Orders".

Follow the following to create lookup field:

1) Select the table *Orders*.
2) Open it in *Design View*.
3) Click the field *CustomerID*.
4) In the *DataType* select *Lookup Wizard*.
5) Select *I want the lookup column to lookup the values in a table or query* and click *Next*.
6) Now select the table *Contacts* from the list provided and click *Next*.

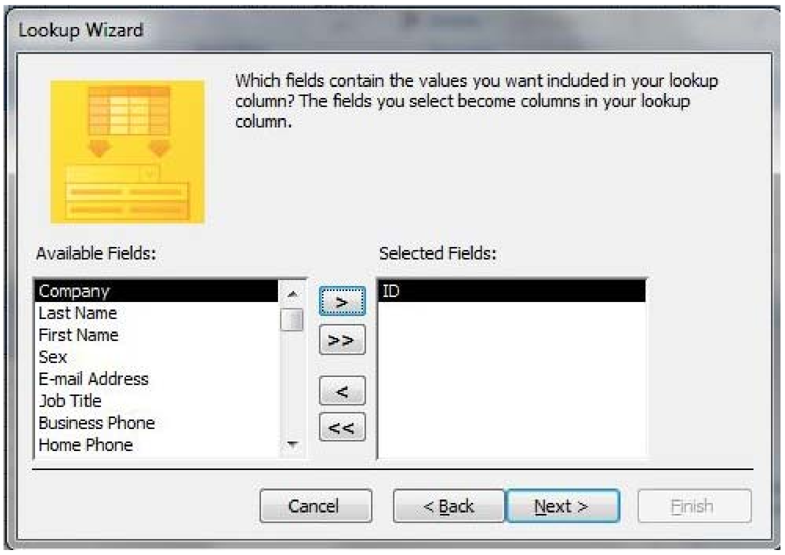

Lookup Wizard

Which table or query should provide the values for your lookup column?

Table: Contacts
Table: Settings

View: [✓ Tables] Queries Both

7) Now select the field *ID* from the available fields list and click > to shift this field to the selected fields list as shown.


Lookup Wizard

Which fields contain the values you want included in your lookup column? The fields you select become columns in your lookup column.

Available Fields:
Company
Last Name
First Name
Sex
E-mail Address
Job Title
Business Phone
Home Phone

Selected Fields:
ID

8) Click *Next*.
9) Now choose sorting order of the values; select the column on which you want to sort and select the sort order by just clicking the button given on the right of the column box. Then click *Next*.
10) From the next screen you can select the width of the lookup column. Click *Next*.
11) On the next screen it will ask for a label for this column. Give a label as you want. This is the last screen. Just click *Finish*.
12) Save the table, check the table in Datasheet view and see the value drop down list in field *CustomerID* coming from the field *ID* of the table *Contacts*.

We have successfully created a lookup field by using the values stored in another table.

11.3 Validating Data as It's Entered

When designing a database we generally know that which type of data will be entered in a field. Mostly we also know the range or domain of values which will be the valid values for a field. So it is often possible to formulate some rule which can validate a data which is being entered by a user. The rules may that a student's admission date must not be a date prior to his/her Date of Birth. The name of a person must not be of zero length. Microsoft Access provides the "Validation Rule" property of a field to establish such checks on the data entered in the field. A simple validation rule is that a value must be entered in a field. If this is the requirement then we set the "Required" property of the field to "Yes". Then a user cannot leave this field empty. We can set rules for both: a field or a record. For example if two dates, OrderDate and ShipDate are entered in the same record, then we can set a validation rule for the record that ShipDate must not be a date before the OrderDate.

We can create a Validation Rule by following given steps below:

1) Select the target table.
2) Open the table in *Design View*.
3) Click the field for which *Validation Rule* is to be created.
4) Now click in the property *Validation Rule* in the lower half of the table *Design window*.
5) Type in the *Validation Rule* according to the requirements.
6) You may enter a descriptive text in the property *Validation Text* which will be shown if the data entered in the field violate the validation rule.

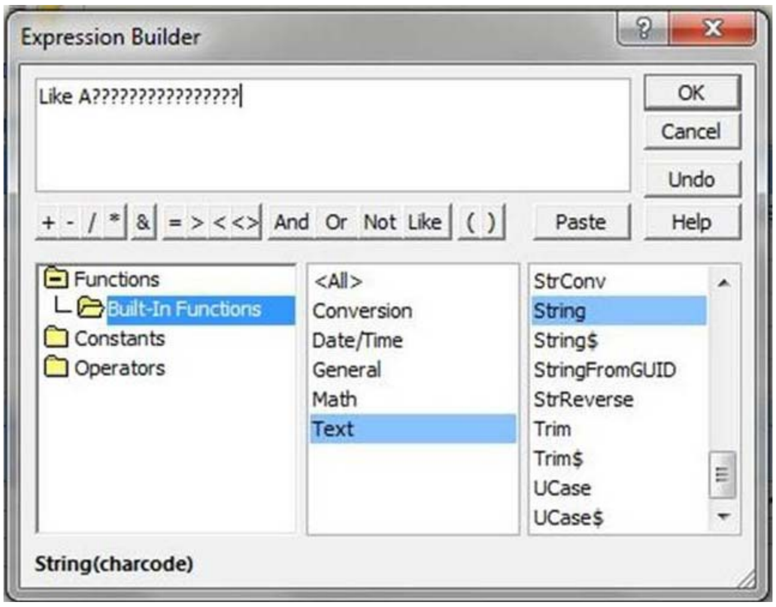
Now you can test the created Validation Rule by entering data in the field.

Rule for the Field	Validation Rule
Date not before 2006	>#12/31/05#
Price zero or greater	>=0
Five characters beginning with P	Like P????
Ship date equal to or later than order date	[Ship Date]>=[Order Date]

Table 8 Validation rules and their syntax in the field validation rule

When entering the data, if user enters a value which violates the validation rule then a message will be displayed in a popup which you have entered in box *Validation Text* which may provide a guideline about the valid data to be entered in that field.

We can build expression by using expression builder to formulate validation rules. Expression builder also provide us to use functions to perform some complex validation task and return a True or False value. The expression builder can be launched by clicking the button … on the right of the box *Validation Rule* as shown in Figure 23.



```
Like A?????????????????
```

String(charcode)

Figure 23   Expression Builder

Table 9 shows some further examples of the Validation Rules. You may find the guidelines from these to formulate your validation rules.

Validation Rule	What does it mean
“Frankfurt” OR “Berlin”	Limits input in the field to just those two cities
Is Null	Allows the user to leave the field blank
<20	Allows values less than 20
>20	Allows values greater than 20
<=20	Allows values less than or equal to20
>=20	Allows values greater than or equal to 20
=20	Allows values equal to 20
<>0	Allows values not equal to 0
In(“Boing”,“Jet”)	Allows text that is Boing or Jet
Between 30 And 40	Allows values between 30 and 40
Like “#####”	field to only allow five digits
Like “S*”	Only strings starting with S
LIKE “S???”	Only strings having 4 characters and starting with S

Table 9   Examples of validation rules

Activity 16

Use the table created in Activity 15. Try to apply validation rules along with input masks to validate the data entered.

12 Creating Queries (Chapter 20)

Learning Objectives

After the completion of this chapter students will be able to:

- Design and create different types of Queries
- Create Queries in Design View and using Query Wizard
- Create parameterized queries
- Create action queries (Create table, delete data, add data, update, insert)
- Use Select, Sort and Filter criteria to get the desired data only in queries
- Create queries to get data from multiple related tables
- Create and use calculated fields
- Perform calculations, arithmetic, date and time, text, etc. in queries
- To find duplicate data and records

Preface

As we have discussed that all database applications are designed to store and manipulate data. Once data is stored in the database then it is to be retrieved according to the requirement. Some it may be the requirement to get the whole data from a table and some time it is to get selected data from one or multiple related tables. The records may be selected by following certain criteria and we have to perform calculations as we select the data. Microsoft Access provides query objects to fulfill all such requirements. To build a query is just like to build an object which asks question about the data from the database. It may be like that who lives in "Lahore" city from my contacts list; what items were ordered by a certain customer on a specific data; etc. By using queries we can find all such related data from the database. We can build as many queries in a database as we require. In this chapter we will learn all about queries.

12.1 Types of Queries

Microsoft Access provides us with many different types of queries. These include:

- Select Query
- Filter and Sort Query
- Action Query
- Parameterized Query
- Summary Query
- Lookup Query
- Crosstab Query

We have learnt all about these in our class work.

12.2 Creating a Query in Design View

In Microsoft Access we can create a query starting from scratch. In this technique we select the tables which we want to include in the query, we select fields from selected tables from which the data is to be fetched, we specify criteria and other information which is required by the query to perform the desired action. We have discussed about it in our class room sessions.

12.3 Creating a Query with the Query Wizard

Microsoft Access provides a guided mechanism to create a query. It is known as Query Wizard. It guides us step-by-step to create a query. In our class room session we have discussed and used query wizard to create different types of queries.

12.4 Understanding Design View

We have discussed about designing a query in design view. This is the interface in which we select our tables, fields, and put criteria to fetch data. In this view we can also tell the Microsoft Access that what is the type of our query. The upper half of the interface is table pane where we place tables which will be used as data source for query and in the lower half we select the fields from these tables. We have covered this interface and its components in our class room discussions.

12.5 Viewing Your Query

The ultimate objective of designing a query is to fetch the filter, sorted and selective data. When we have created a query in Microsoft Access then we can view this query in different ways including Design and Datasheet views. We have learnt about viewing the query in our class lectures and practices.

12.6 Limiting Records with Criteria Expressions

In Microsoft Access we create queries to display the selected fields form a table or set of related tables. Microsoft Access also facilitates us to specify criteria to show the limited records. If we specify criteria in a query then only those records will be shown in a query which satisfies the criteria. The criteria specified applied on the values in the fields of the tables.

A criteria is just an expression. It is a logical expression which returns True or False values after evaluation. If the result is True then the record is included in the resultant data and not otherwise. A criteria work like a formula. It is also like a validation rule which we have studied in table design. Criteria are a string which has field references, literals and operators in it. Different criteria look different in their look; it all depends upon the data type of the field on which we are using criteria. Some criteria use basic operator while other may use functions and complex operators.

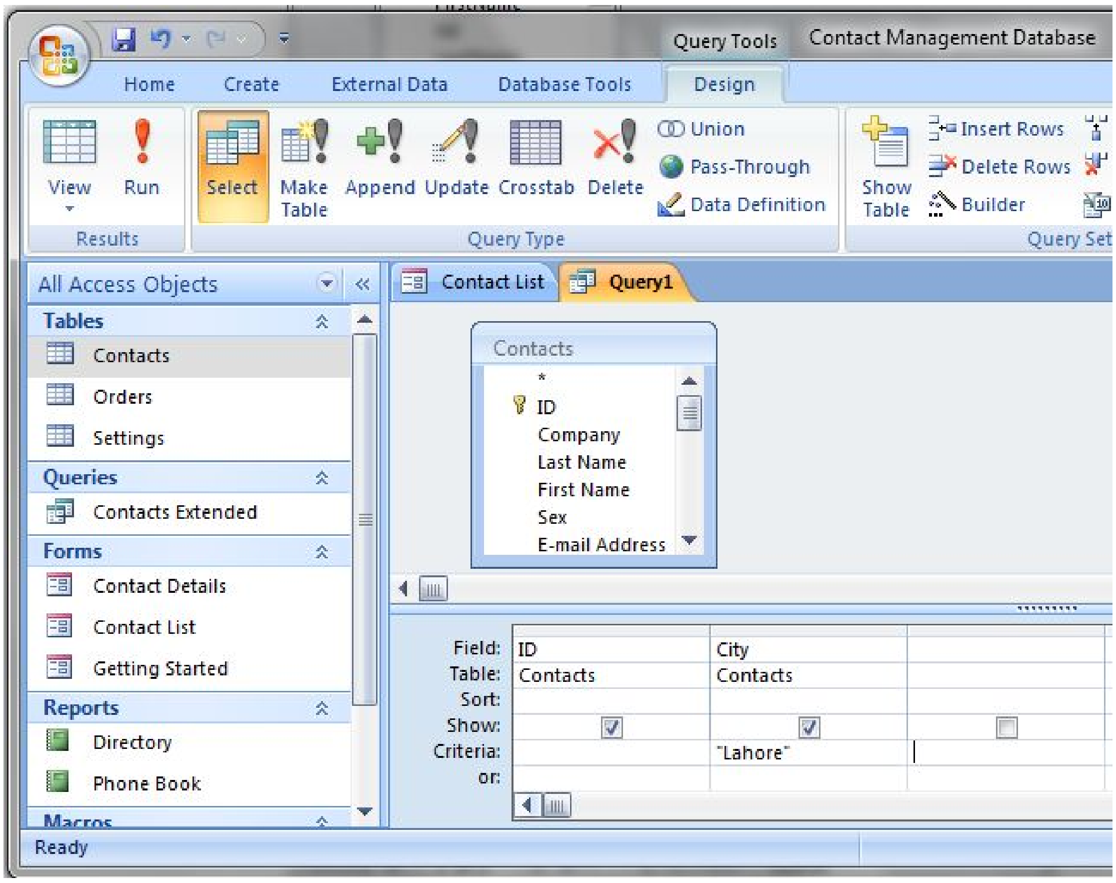
We can specify the criteria in the following different ways:

- Querying by example (QBE)
- Using Field, Literal, Date, Time, Text or other Literal Values in Criteria
- Operators being Used to write Criteria Expressions

12.6.1 Querying By Example (QBE)

Microsoft Access provides a Querying By Example method to define the criteria for a query. In this method we tell the Microsoft Access that what is our requirement or for which data we are looking for, then Microsoft Access can proceed accordingly and fetch the data which we are searching. For example if we want to find the students with age 15 then we simply type 15 in the criteria and Microsoft Access will search the data accordingly.

Simplest criterion is a logical expression in which we test the value of a field against some other constant or literal value and the result of this expression is True or False. To find all the contacts which have city "Lahore" will be as in Figure 24.



Field:	ID	City	
Table:	Contacts	Contacts	
Sort:			
Show:	✓	✓	☐
Criteria:		"Lahore"	
or:			

Figure 24 Criteria in a Query for City field

12.6.2 How to use *Field, Literal, Date, Time, Text* or other Literal Values in Criteria

In our data we come across different types of values. In our criteria sometime we have to use a table field or a date value or a time value or value of text type. Here we will discuss that how we can use such data types in our criteria. Table 10 describes that how we can use these data types in expression defining criteria.

Type of Data	In an Expression
Text	"String"
Date	#2-April-2010#
Time	#11:00am#
Number	60
Field name	[field name]

Table 10 Criteria values

As shown above, if we want to use a text in our criteria then we have to enclose this value into double quotes. We can use all the allowed date and time formats supported by Microsoft Access in our criteria expressions.

12.6.3 Operators being Used to write Criteria Expressions:

Criteria expressions are just like the validation rules. We use different Relational and Logical operators to write these criteria expressions. Table 11 shows and describes different operators being used in writing these expressions.

Relational Operator	What It Does it stands for
=	Finds values equal to text, a number, or date/time.
<>	Finds values not equal to text, a number, or date/time.
<	Finds values less than a given value.
<=	Finds values less than or equal to a given value.
>	Finds values greater than a given value.
>=	Finds values greater than or equal to a given value.
BETWEEN	Finds values between or equal to two values.
IN	Finds values or text included in a list.
LIKE	Finds matches to a pattern.

Table 11 Operators to write criteria expressions

Microsoft Access provides the facility that we don't have type the name of the field when writing the criteria expression; when we type criteria in the column of a field then its name is automatically considered. When you type your criterion, you don't have to tell Access the field name. But if we have to refer some other field then we have to mention its name using "[]" pair.

Table 12 describes some examples of data types used in criteria expressions.

Criteria Name	Write it like...	Function
Equals	"**x**"	Searches for values equal to **x**
Does Not Equal	Not in ("**x**")	Searches for all values except those equal to **x**
Null	Is Null	Searches for empty fields
Not Null	Is Not Null	Searches for non-empty fields
Criteria Name	Write it like...	Function
Contains	Like "***x***"	Searches for all values that contain **x**
Does Not Contain	Not like "***x***"	Searches for all values except those that contain **x**
Begins with	Like "**x***"	Searches for all values beginning with **x**
Ends with	Like "***x**"	Searches for all values ending with **x**
Comes After	>= "**x**"	Searches for all values that come after **x** in alphabetical order.
Comes Before	<= "**x**"	Searches for all values that come before **x** in alphabetical order.
Between	Between "**x**" and "**y**"	Searches for values in the range between *x* and **y**
Less Than	< **x**	Searches for all values smaller than *x*
Less Than or Equal To	<= **x**	Searches for all values smaller than or equal to **x**
Greater Than	> **x**	Searches for all values larger than **x**
Greater Than or Equal To	>= **x**	Searches for all values larger than or equal to **x**
Between	Between "**#mm/dd/yy#**" and "**#mm/dd/yy#**"	Searches for dates that fall between two dates.
Before	**<#mm/dd/yy#**	Searches for dates before a certain date
After	**>#mm/dd/yy#**	Searches for dates after a certain date
Today	=Date()	Searches for all records containing to-day's date
Days Before Today	<=Date()-**x**	Searches for all records containing dates *x* or more days in the past

Table 12 Data types in criteria expressions

You may see more examples and information about criteria from the following link: http://office.microsoft.com/en-001/access-help/examples-of-query-criteria-HA010341674.aspx

Activity 17

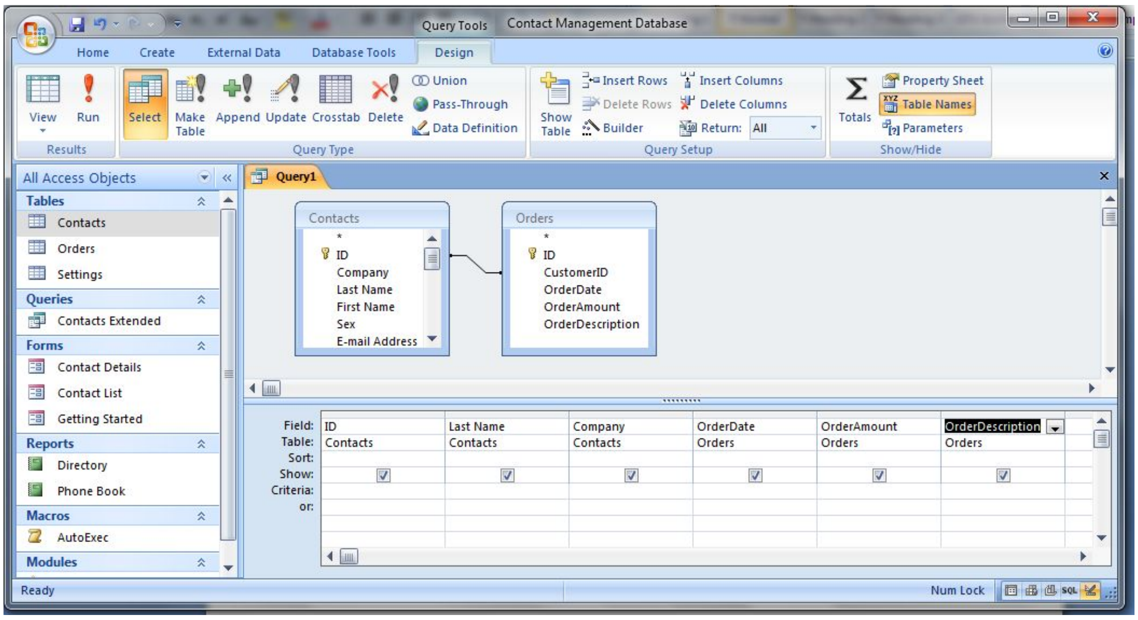
Open the database *Contacts* in Microsoft Access and design a query which shows all records:

- from city "Lahore"
- with Sex as F (female)
- which have not mentioned their email address
- which are Male and from Germany

12.7 Working with Multiple Related Tables

In a well-designed database application there is always more than one table. And these tables have primary and foreign keys. These primary and foreign keys define that how the data in different tables relate to each other. We can establish these relations in Database tools tab in Microsoft Access. If we have established the relations between different tables then we can also these related tables in queries together based on their relationship.

In our example of Contacts database, we have two tables, Contacts and Orders. These both have a relationship on ID field in Contacts and CustomerID in Orders. There is One-To-Many relationship between these two tables on these fields; means that one record in Contacts table can have none or multiple corresponding records in Orders table which are linked on the fields mentioned above. Figure 25 shows the Design View of a query which uses both the Contacts and Orders form to display the related records from both the tables.



Field:	ID	Last Name	Company	OrderDate	OrderAmount	OrderDescription
Table:	Contacts	Contacts	Contacts	Orders	Orders	Orders
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

Figure 25 Query with multiple tables

If we are using some field in the criteria then we must include it in the query. If we don't want to display a field in the query then we can turn is "show" mark off by click the clicking the check box in its column.

12.8 Doing Math in Queries

It is very often that when we retrieve data from a database then we perform calculations on this data. It is very rare case that we don't have to do so with the data. If you are able to perform math with a paper-pencil then it will be of the same comfort level to do the same in the queries in Microsoft Access.

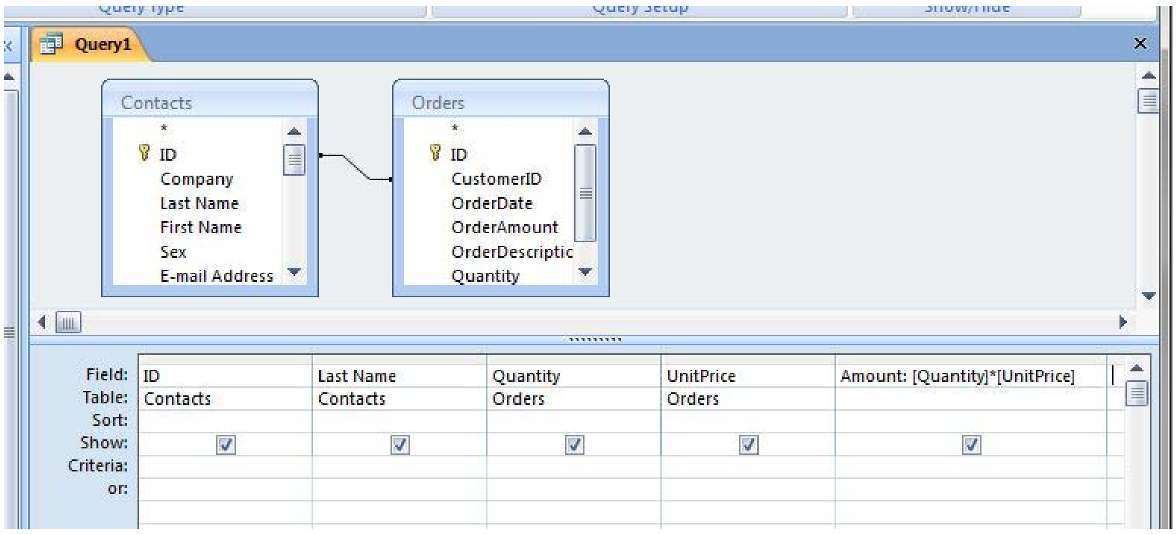
Calculation may of the type that, we have date of birth of some person and we know the current date, on the basis of these two facts we can calculate the age of some person. Or we can have the number of items and the unit price in two different fields of the record and we can calculate the total amount from these two data items. These types of calculations in the queries save us from saving extra information in our table and also give us relief from doing such calculations by ourselves. If we have put a calculated field in query then any object in Microsoft Access using that query with automatically get that calculated value.

To do calculations and math in a query, we need to create an additional field in the query which is known as Calculated Field. The value of a Calculated Field doesn't come directly from any field of the table rather it is the result of some arithmetic expression written in its source. When we add a calculated field in a query then we give it a name and this name should be unique within that query. By adding this field, we do not add any field to the table, rather just a field to the query only.

Let we have to fields Field1 and Field2 in a table. We are designing a query and we want to add a calculated field CalcField in a query based on the Field1 and Field2. Then we will write the following string in the calculated field box:

CalcField:Field1+Feild2

Figure 26 shows a query which have a calculated field Amount which is the result of multiplication of Quantity and UnitPrice fields of the table orders.



Field:	ID	Last Name	Quantity	UnitPrice	Amount: [Quantity]*[UnitPrice]
Table:	Contacts	Contacts	Orders	Orders	
Sort:					
Show:	✓	✓	✓	✓	✓
Criteria:					
or:					

Figure 26 Query with calculated field (Amount)

In Figure 26 the first 4 fields ID, Last Name, Quantity, UnitPrice and regular fields which get their values directly from the relative tables whereas the fifth field

Amount:[Quantity]*[UnitPrice]

is a calculated field. The field name is Amount and it gets its value from the expression [Quantity]*[UnitPrice] where both the Quantity and UnitPrice and the fields in the Orders table.

We can have as many calculated fields in our query as we require. It is an art to write the expressions for calculated fields and you have numerous possibilities to write and get results from the calculated fields. You can use simple arithmetic operators or even complex functions provided by Microsoft Access to write these expressions.

12.9 Creating Action Queries

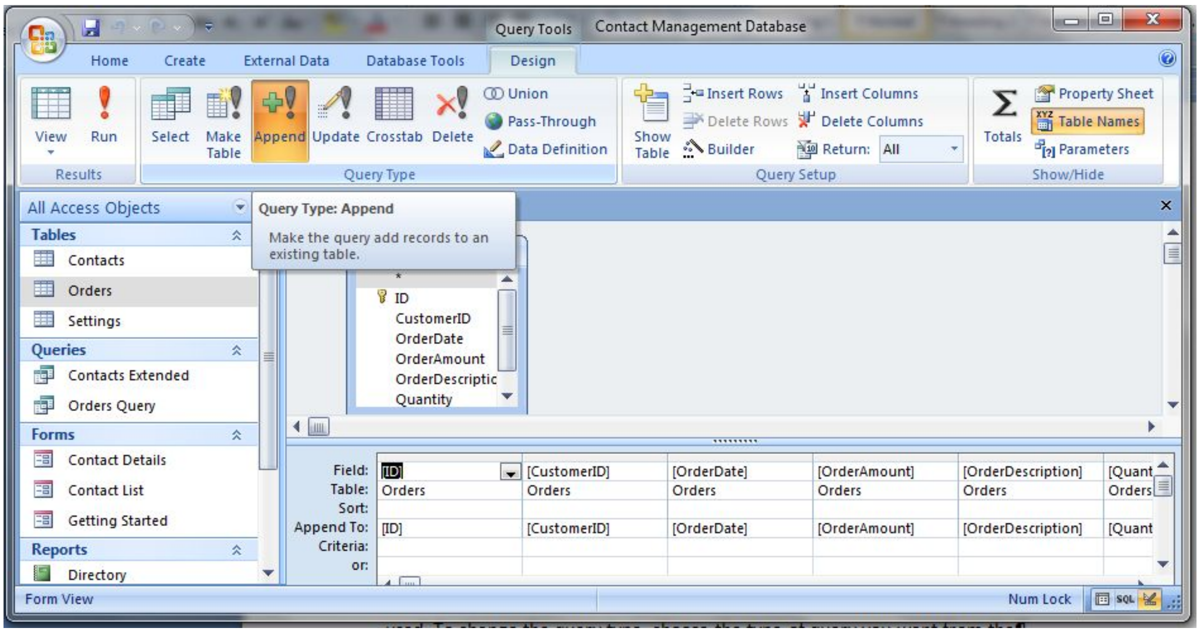
Dear Students! You have learnt how to create simple select queries. These queries help to get selected results from the data. There exist other kinds of queries in Microsoft Access which are used to perform actions on the database. These are known as action queries. Action queries are very much different from select queries. You can use these queries to make changes to the data, to delete data, to add data from one table to the other and to create a table. Action queries are powerful which can help you to make corrections and perform maintenance of your database. But if the actions queries are not used with much care then these may mess up your database and you may run in a real trouble.

There are four different kinds of Action Queries as described below:

- Update Query: these queries are used to make changes in the selected records.
- Append Query: these queries are used to append data to a table from another table depending upon some criteria.
- Delete Query: these queries are used to dele some records from a table which satisfy some criteria.
- Create-Table/Make-Table Query: these queries are used to create new tables from the data of other tables.

To run an action query safely, it is a good practice to make a backup of your data because the changes made by the action query are irreversible and there is no UnDo for it.

You can create an action by any of two methods which we discussed to create select query i.e. either by Query Wizard or by using Design View. Using any of these methods will create a Select Query. Now you will have to change its type. You can change the type of the query by selecting the type of query from the Query Type group of the Design tab on the Ribbon.



Query Type: Append

Make the query add records to an existing table.

Field:	[ID]	[CustomerID]	[OrderDate]	[OrderAmount]	[OrderDescription]	[Quant
Table:	Orders	Orders	Orders	Orders	Orders	Orders
Sort:						
Append To:	[ID]	[CustomerID]	[OrderDate]	[OrderAmount]	[OrderDescription]	[Quant
Criteria:						
or:						

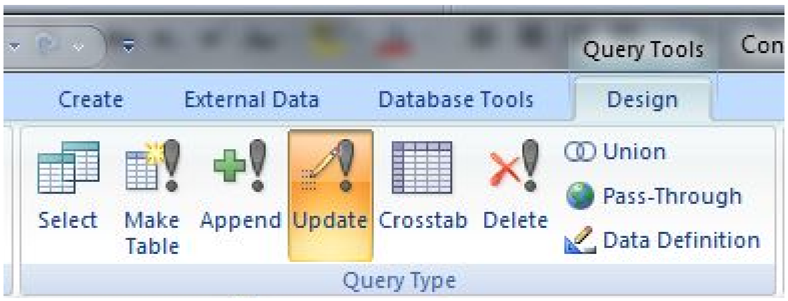
Figure 27 Changing the Type of Query

12.9.1 Update Queries:

Update queries are used to make the changes in bulk of data in a single go. For example if we want to modify the comments about students in a result database depending upon the percentage marks then we can use Update Queries. We may be interested in the list of students who have not submitted their dues for a particular semester.

You can create an Update Query by following the given steps:

1) For your safety purpose backup the database or the tables to be altered by the action query you are going to create.
2) Create a select query by using Query Wizard or in *Design View*.
3) Place the fields in the query grid which are to be affected by the Update Query and which are to be used in criteria. You may use the calculated fields.
4) Establish the criteria to update the desired data.
5) Check the data by clicking *View* that which data will be affected.
6) Make any change in criteria or fields if you find some errors or mistakes in step 5.
7) Now change the *Query Type* from the group *Query Type* of the tab *Design* on the Ribbon. Click *Update*.



8) If everything is alright then click Run button to run the update.
9) Access will show a confirmation dialog box, click the Yes button to update the data.
10) Check the updated data for the correctness of data.
11) You may save the query if you want to use it in future but always be careful in opening this type of update queries to avoid some data loss or data mess up.

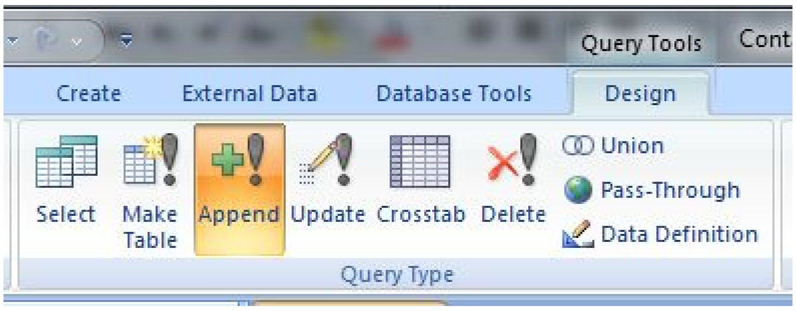
You have created an Update Query successfully.

12.9.2 Append Queries:

If we want to add data to a table from another table then we use Append Queries. These queries are used to copy selected records from one table to another which satisfy some criteria.

We can create an Append Query by following the steps given below:

1) For your safety purpose backup the database or the tables to be altered by the action query you are going to create.
2) Create a *Select Query* by using *Query Wizard* or *Design View*.
3) Run the Query to see that whether the query displays the desired records only, otherwise make required corrections and changes in the criteria in the select query.
4) Now change the *Query Type* from the group *Query Type* of the tab *Design* on the Ribbon.



5) Select the Table Name to which data is to be appended.
6) Now click *OK*.
7) Now check for any errors in the appended data.
8) Run the query to copy the data by running the append query.
9) Access will show a confirmation dialog box, click *Yes* to update the data.
10) You may save the query if you want to use it in future, but always be careful in opening this type of update queries to avoid some data loss or data mess up.

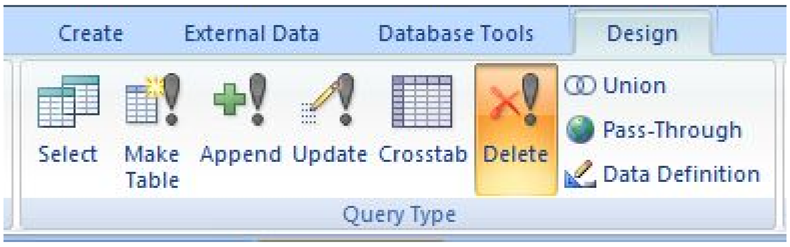
You have created an Append Query successfully.

12.9.3 Delete Queries:

Delete Queries are meant for deleting bulk of records from a table in one go. The only requirement is that the records should satisfy some criteria. Delete queries are powerful but dangerous at the same time. You should use them with great care because you can lose whole of you data in just one click. You should also be very careful in deleting data from a table as the table may be related to other tables in the database and by deleting some records data integrity and data dependency may be violated and you may run in a trouble.

We can create a *Delete Query* by following the steps given below:

1) For your safety purpose backup the database or the tables to be altered by the action query you are going to create.
2) Create a Select Query by using Query Wizard or Design View.
3) Run the Query to see that whether the query displays the desired records only, otherwise make required corrections and changes in the criteria in the select query.
4) Now change the Query Type from the Query Type group of the Design tab on the Ribbon.



5) Now click *OK*.
6) Run the query to delete the records.
7) Access will show a confirmation dialog box, click *Yes* to update the data.
8) You may save the query if you want to use it in future but always be careful in opening this type of update queries to avoid some data loss or data mess up.

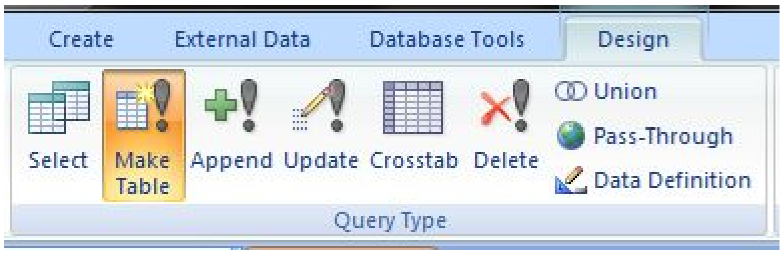
You have created a Delete Query successfully.

12.9.4 Create-Table/Make-Table Queries:

When it is required to make back of data of selected records or all records from one table or multiple tables in new table, then these queries are really useful. Microsoft Access provides this easy way to backup our data. These queries are generally not dangerous as these do not destroy any data, these just create a duplicate of the data in a new table.

You can create an Make-Table Query by following the steps given below:

1) Create a *Select Query* by using *Query Wizard* or *Design View* by setting the criteria for the desired set of records and fields to be exported to new table.
2) Run the Query to see that whether the query displays the desired records, otherwise make required corrections and changes in the fields and criteria in the select query.
3) Now change the *Query Type* from the group *Query Type* of the tab *Design* on the Ribbon.



4) When you click the Make-Table, Microsoft Access will display a dialog box which will ask you to enter name of New Table.

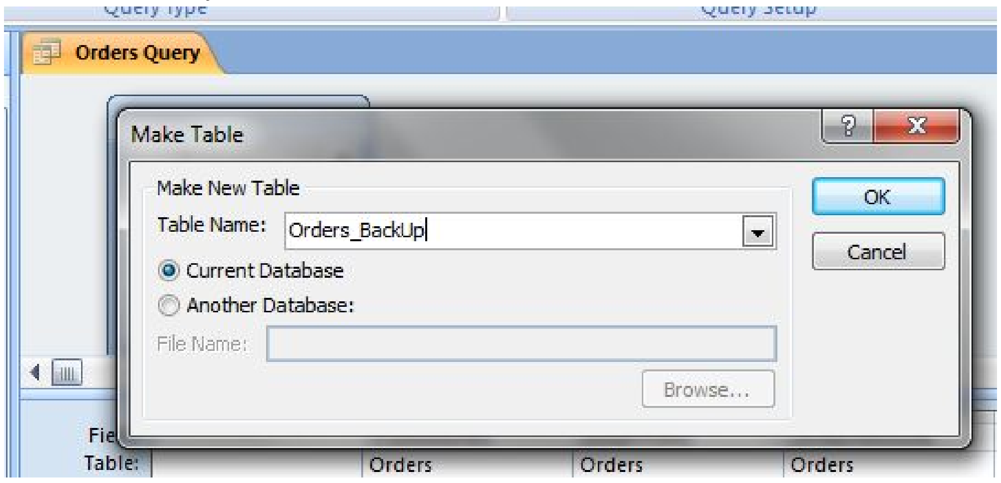


Table Name: Orders_BackUp
[✓ Current Database]
Another Database:
File Name:

5) Type in the name for new table. Choose from the two options that either the new table should be created in *Current Database* or in *Another Database*.
6) Now click *OK*.
7) Run the query to create new table containing the backup data.
8) Access will show a confirmation dialog box, click *Yes* to create the table.
9) You may save the query if you want to use it in future.

You have created a Make-Table Query successfully.

Activity 18

Create all the Action Queries in Contacts database and check by changing the criteria.

12.10 Calculating subtotals in a query

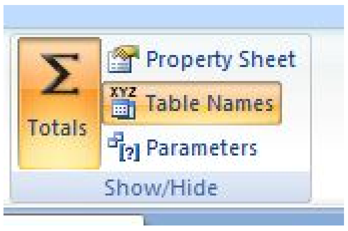
In Microsoft Access we can create a query which can calculate different arithmetic on multiple records. We have discussed about the arithmetic on fields, in this section we will discuss the arithmetic on records. These arithmetic include Total, SubTotal, Average, Count, Min, Max, etc. The queries in which we perform such record level arithmetic are known as Total Query. These queries just show the results only and not the every record included in these arithmetic. Table 13 shows the multi-record level calculation operations.

Keyword/Function	Result
Avg	Average of records in field
Count	How many records
First	Value stored in first record
Group by	Nothing — this is used only for grouping
Last	Value stored in last record
Max	Highest value in all records
Min	Lowest value in all records
StDev	Standard deviation
Sum	Sum of records in field
Var	Variance

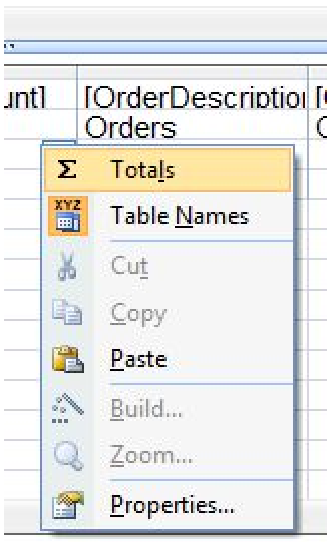
Table 13 Multi-record level calculation operations

We can create a query that performs such calculations by following the steps given below:

1) Create a *Select Query* by using *Query Wizard* or *Design View* by setting the criteria for the desired set of records.
2) In the tab *Design* of Ribbon click *Totals* in the group *Show/Hide*.

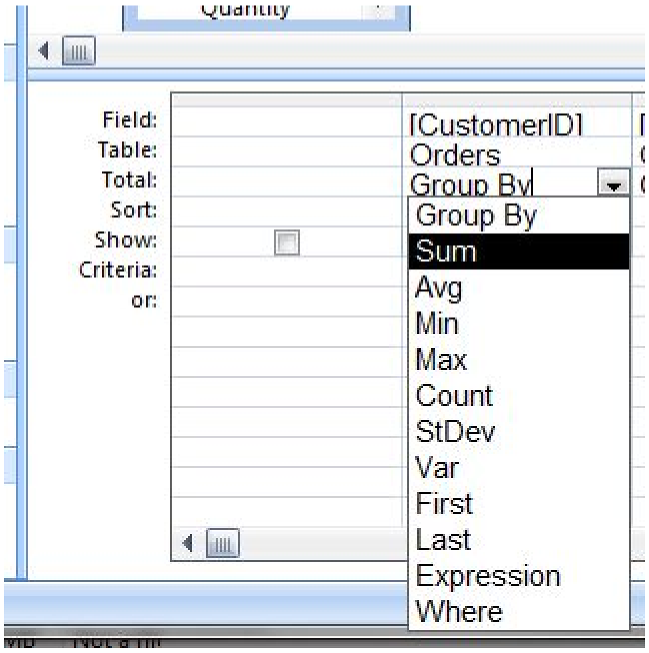


3) To get pop-up menu Right-click in the query grid and then choose *Totals* from this menu.



4) Now you will see a new row with label *Totals* in the Query Grid.
5) Now select all the fields for which you want to perform calculations.

6) When you have selected the fields, then click in the row *Total* below the field and choose required option from this drop down list.



7) Do this for all the fields for which the calculations are required.
8) If we want to calculate subtotals in the query then we have to add a field on which the can be grouped. Then from the Total row select Group By option. Then subtotal for each group will be calculated and shown in the DataSheet view.

In this way we can create to calculate multiple record level calculations in Microsoft Access.

Task 5

Open the Northwind sample database provided by Microsoft Access. Create different queries to do multiple record level calculations in different tables. For Example in Orders table:

1) Calculate Shipping fee subtotal on date.
2) Calculate total shipping fee by Ship Name, etc.
3) Calculate total shipping fee pay by:
   a. Credit Card
   b. Check
   c. Cash

13 Designing and Using Forms (Chapter 21)

Learning Objectives

After the completion of this chapter students will be able to:

- Design and create different types of forms
- Create a form with the help of Form Wizard provided by Microsoft Access
- Create a form to navigate and manipulate records/data
- Create a form from scratch in Design View
- Modify a form in Design View already created
- Add, Arrange and format different Controls on the Form
- Set the properties of From and Controls on the form
- Create Menus, List and Combo Boxes
- Use option buttons
- Add Header and Footer to the Form
- Search data using forms
- Create Tabbed forms
- Create and use sub-forms
- Create applications with Master-Detail forms

Preface

You may have seen different type of database applications. These may be a Point-of-Sale application for a super store, a banking system, a ticketing system, etc. These all application gets data from the users in one form or the other. Users enter data in the database of these systems. Users are not provided direct access to the tables of the database application. If the users are given direct access to the tables then there will be no or limited control on the data entered by the users. To have proper control on the data entered by the users and even to facilitate the users to enter the data in the database in a convenient way, a proper user interface to the database should be provided to the users. Microsoft Access provides "Forms" object to create the GUI for entering, navigation, and manipulation of data. It is very easy to design user friendly GUI by using forms and controls provided by Microsoft Access. In this chapter we will learn to create and use different types of forms.

13.1 Making and using a Form

A form is an interface to the database. It doesn't store data in it. A form shows the data and when the user chose the option in one or the other ways then form saved the data in the related data sources (tables). In general, the data shown on the form is from the fields of a single record of the table, but the data may come from multiple fields of different tables. We can also design a form which can show multiple records at the same time. We can also create Master-Detail form by using sub-forms. We have learnt all these concepts in the class.

13.2 Creating Forms with Wizards

Microsoft Access provides an automated tool which can create forms for us. This tool is knows as Form Wizard. It let you chose different options, asks few questions from you and then create form for you. This may not the form which you desire or looking for but it is a good way to get started with. You can modify this created form using Design view. You can re-arrange the fields and controls on the form and may add VBA code to add required functionality. We have discussed and practiced this in our class room sessions.

13.3 Viewing a form

Microsoft Access provides three different types of a form view, form view, layout view and design views. All have their own functionality. In our class room session we have discussed and learnt these concepts.

13.4 Modifying Existing Forms

Forms created by Wizards are generally not according to the requirements. We often have to modify these forms. Even the forms created manually in Design View may need modifications often. This is the process which we have studied in our class sessions.

13.5 Where records come from

As we know that the forms are interface to the database, so the data in the forms come from different tables and fields of the database. We can design forms to display data from tables or queries in different ways. We have studied this in the class room.

13.6 Form control types

Microsoft Access provides many controls which we can add to a form. These controls belong to different types. Some are action controls like Command Buttons, some are data controls which provide editable text boxes, some are used just to display information, and so on. This is a topic which we have studied in detail in the class.

13.7 Setting control properties

In the process of designing a form, we add controls to the form. It is not enough to just add the controls to the form. To add proper functionality and to give proper look-and-feel we have to setup different properties of the controls. These properties include: control name, control Text, foreground color, background color, border, etc. The properties of the controls have discussed extensively in our class discussing the form design,.

13.8 Binding a control to data in the record source

We design the form to display the data from the tables. Then it is important to set a relation between a control on a form and the field of a table from where the data will come in this control. This phenomenon is known as to bind a control to the data source. This topic has also been discussed in the class.

13.9 Making a button to display a related form

We can add different controls on a form according to our requirement. Some of these controls are of action type controls which can perform some action as we click on them. One of these is Command Button. We can define any action on its click event. One of these actions is to display some related form. We have studied this concept is our class sessions.

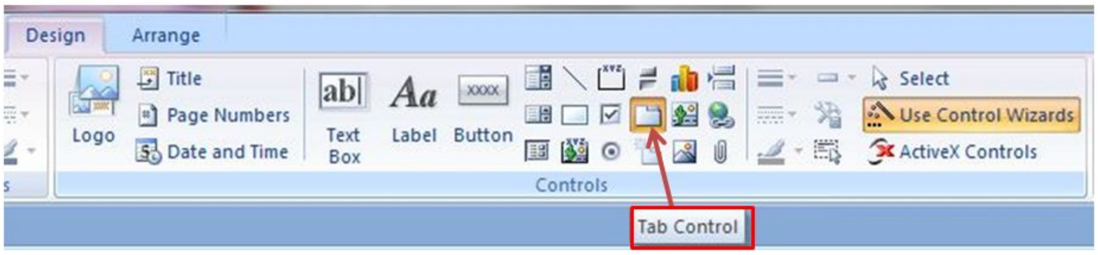
13.10 Creating Tabbed Forms

We are discussing all about database applications. In this chapter we are studying about the interface provided to the users to manipulate data in the database. While developing database applications sometime we face a situation where we have a lot of data which should be displayed or manipulated by using a single form. Now if we design a classical form, then this form may span too wide and too lengthy to be fit on our monitor screen. It will also seem to be over crowded and data arrangement and management will be a nightmare.

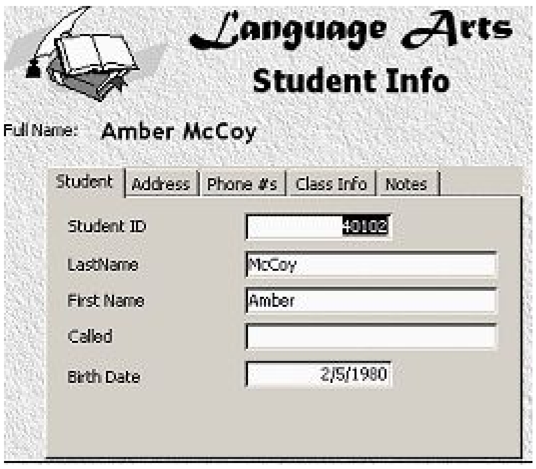
Microsoft Access provides us with a control known as Tab Control to manage large amount of data on a small sized form having multiple pages arranged in a small area. A tab control has multiple pages where each page behaves like a form and contains different controls on it. So we can use Tab control to have multiple pages in our single form. For example in our Contacts database the information in Contacts table can be divided into two sections: personal information and official information. Then we can arrange this information on two different pages of Tab control on the form. In this section we will learn that how we can create a form with Tab control. When we place tabs on a form then it will become better organized and more user friendly. We can use tabbed pages to arrange data logically.

We can create a form with a Tab control from the scratch or we can add a Tab control to a form which we have already created. We can create a form with Tab control by following these steps:

1) Open the form in *Design View*. Create adequate space to place the Tab control on the form.
2) Click the *Tab Control* in the Control *group* of the *Design tab* of the Ribbon.



3) Click in the form where you want to place the control and drag to adjust the size of the control.
4) Now change the name of the Tab control to some descriptive name.
5) Change the names of the pages in Tab control as per the data category or group.
6) Place related controls on the related tables and arrange them as per example shown in the following figure.


Language Arts
Student Info
Full Name: Amber McCoy
Student | Address | Phone #s | Class Info | Notes
Student ID: 40102
LastName: McCoy
First Name: Amber
Called:
Birth Date: 2/5/1980

7) Now save and test the form for desired results.

In this way you can create a form with a Tab control on it.

13.11 Doing Elementary Calculations

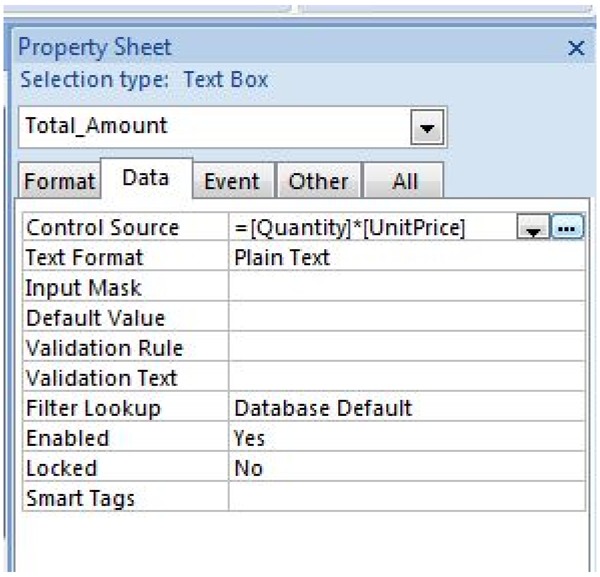
When we place controls on the form then we generally set their data source property. Normally this data source is a field of a table from the database. Microsoft Access offers the opportunity to place controls on the form for which data source is not a field rather it could be an expression which perform calculations and put the result in that control. Putting these calculations are as easy as the elementary mathematics. To create a calculated control generally we place a text box on the form and write an expression in its Control Source property. Expressions are just formulas to perform arithmetic. When writing an expression in Control Source then we start it with an "=" sign. If any field names are included in the expression then these are enclose in "[]" pair. A control named Total amount may be calculated as follows from the fields Quantity and UnitPrice:

= [Quantity] * [UnitPrice]

We have already discussed about expression in our previous sections when creating calculated fields in query.

We can create a calculated control on a form by going through the following steps:

1) Open the form in *Design View*.
2) Place a Text box on the form.
3) Show the property sheet of the Text box.
4) Change the name of the Tex box to some descriptive name.
5) Select the Data tab on the property sheet.
6) Click the property *Control Source* of the Text box.



Control Source	=[Quantity]*[UnitPrice]
Text Format	Plain Text
Input Mask	
Default Value	
Validation Rule	
Validation Text	
Filter Lookup	Database Default
Enabled	Yes
Locked	No
Smart Tags	

7) Type the expression in the property box *Control Source*. We can also use "…" (build) button from the property *Control Source* to build expression. We can use table fields and form controls both in an expression, just be careful not to name a control on the form same as of the field name in a table otherwise the expression will not know that which to use in the calculation.

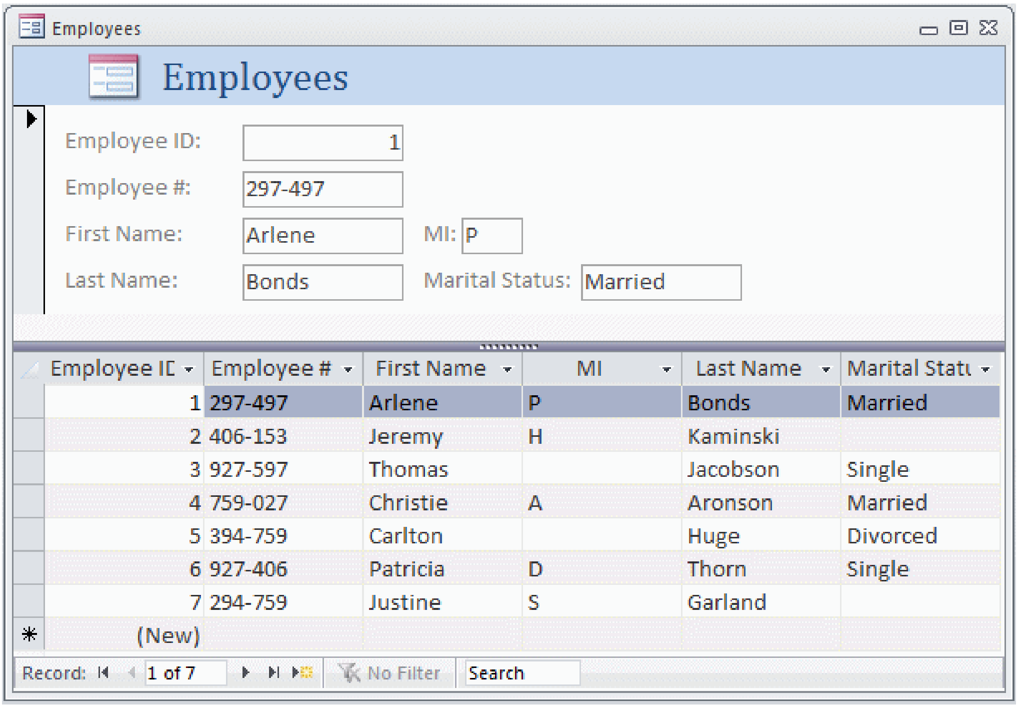
In this way we can perform elementary calculations on our form.

13.12 Using a Split Form to Display a Datasheet

When displaying the data in the forms, sometimes it is convenient to browse through the data using a datasheet view and then select a record and to display it or manipulate its fields in a form. Microsoft Access has the facility to arrange both these views (datasheet view and form view) in a single form. For this purpose Microsoft Access has the concept of Split Form. In the Split form we can logically divide a form in two parts. One part can show the datasheet view and other half have the arrangement of the fields and may also have some calculated fields. When we click a record in the datasheet view then its contents are displayed in the form view.

We can create the split form as per the following process:

1) Select the table or query which will play the role of record source for the split form.
2) From the *Create tab* of the Ribbon choose Split form.
3) Microsoft Access will create a form split from the middle, lower part will show the datasheet view and upper half will have the controls for all the fields from the base table or query.



Employees

Employees

Employee ID:	1		
Employee #:	297-497		
First Name:	Arlene	MI:	P
Last Name:	Bonds	Marital Status:	Married

Employee ID	Employee #	First Name	MI	Last Name	Marital Statu
1	297-497	Arlene	P	Bonds	Married
2	406-153	Jeremy	H	Kaminski	
3	927-597	Thomas		Jacobson	Single
4	759-027	Christie	A	Aronson	Married
5	394-759	Carlton		Huge	Divorced
6	927-406	Patricia	D	Thorn	Single
7	294-759	Justine	S	Garland	
(New)					

Record: 1 of 7 No Filter Search

4) We can adjust different properties of the split form by selecting the property sheet from the ribbon.

Using these steps we can create a split form to make the data navigation and data manipulation more user friendly.

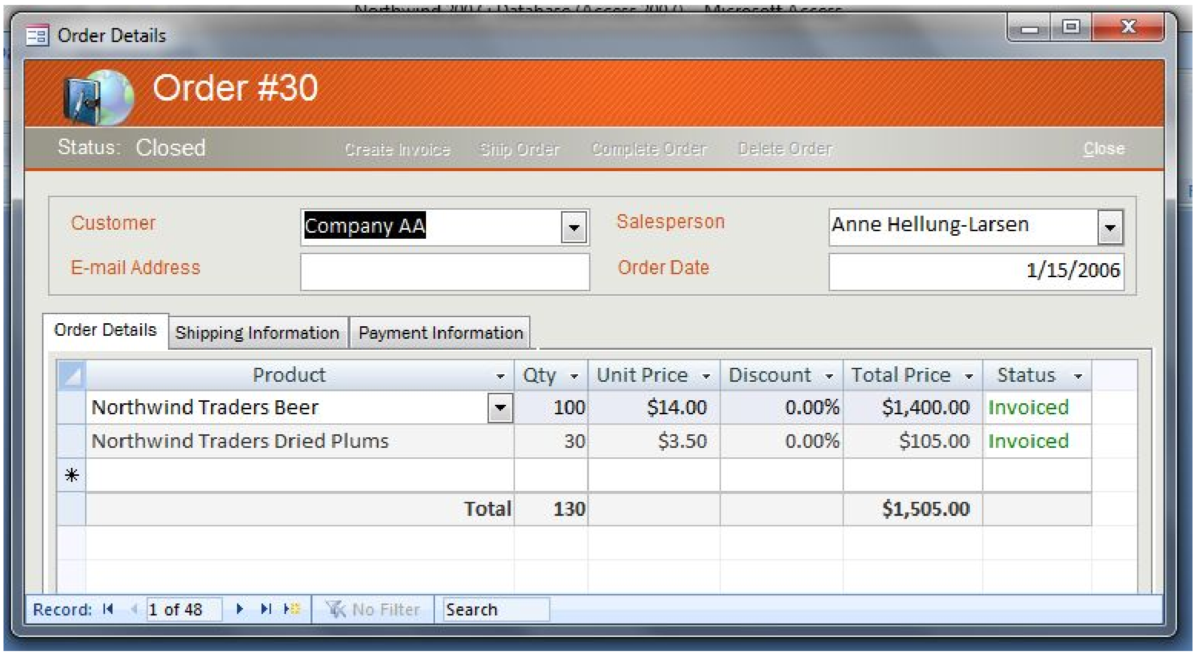
Activity 19

Create split forms for different queries and tables in the Northwind sample database provided with Microsoft Access.

13.13 Using a Subform to Display Detail Records

If we observe different database applications then we will see that there are situations when we have many database records against one entity. For example if we consider a database of Students then for a single student there will many records of subject marks, dues, assignments, etc. When we purchase some items from a store then there are multiple items in the bill against one bill and one customer. For an Employees database there are many entries of Salary and Tax against one employee. Such situations are very common in our daily life.

To handle such situations we generally handle more than one tables, queries or data sources. The data source from which single record is coming is known as Master and the data source from where multiple related records are to be fetched or displayed is known as detail. It will be a good solution if we can display both the information in a single form. Figure 28 shows the example of such form. It is commonly known as Master-Detail form. Such type of forms are created by using subform feature provided by Microsoft Access.



Status: Closed

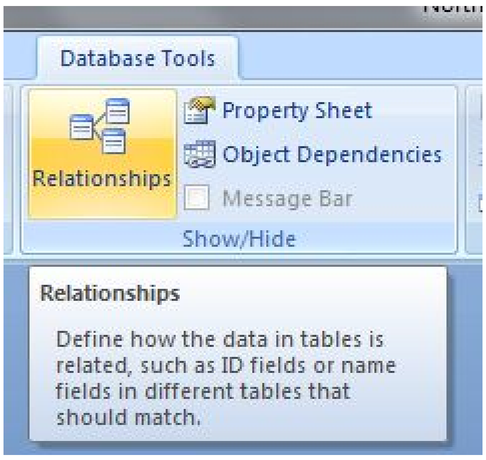
Customer	Company AA	Salesperson	Anne Hellung-Larsen
E-mail Address		Order Date	1/15/2006

Product	Qty	Unit Price	Discount	Total Price	Status
Northwind Traders Beer	100	$14.00	0.00%	$1,400.00	Invoiced
Northwind Traders Dried Plums	30	$3.50	0.00%	$105.00	Invoiced
Total	130			$1,505.00	

Figure 28 Master Detail Form

The entries in the detail part (subform) are related to the main record in the main section (main form).

The tables being used in such Master-Detail must have a relationship established. And the relation between Master and Detail data sources should be One-to-Many i.e. one record in the Master data source should have multiple records in the Detail data source. If this relationship is not already established then we can establish it using Relationship button in the Database tools group from the Ribbon as shown in Figure 29.



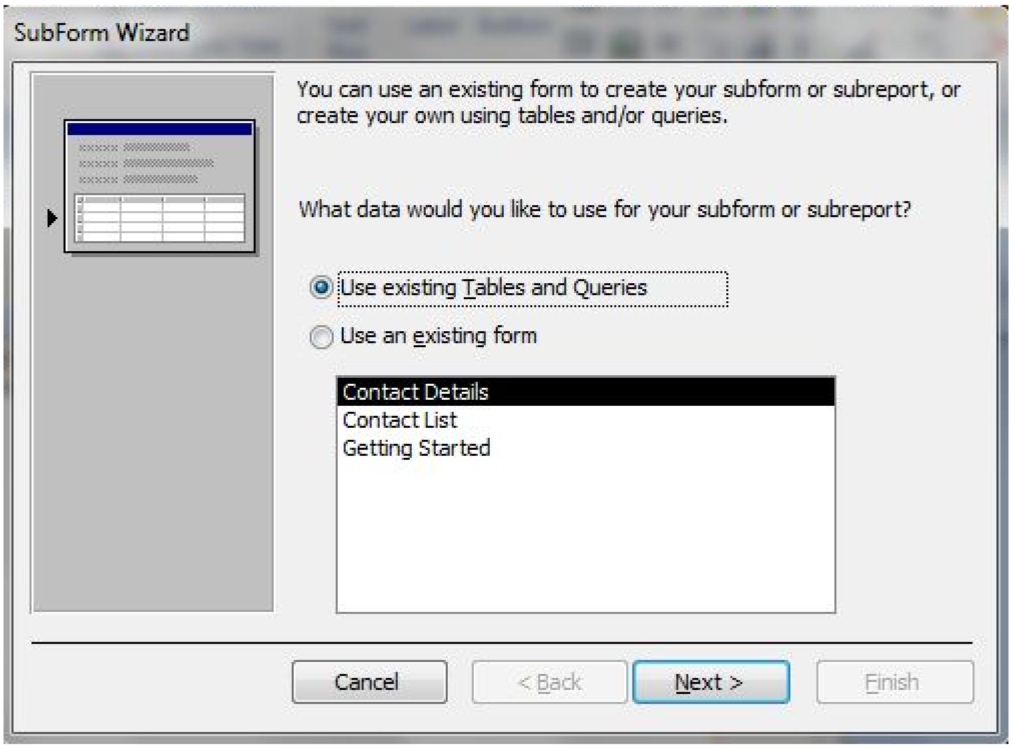
Relationships

Define how the data in tables is related, such as ID fields or name fields in different tables that should match.

Figure 29 Database Relationships

To design a Master-Detail form we have to create and design two separate forms. One form is used as Master or main form and the other form is used as Detail or subform. We can use both the forms as independent forms as well. To create Master Detail perform the steps give below:

1) Open the main form in the Design *View*.
2) From the main data source, place the controls on the main form.
3) Arrange the main data source controls in such a way that there is enough space for placing a subform in proper and reasonable size.
4) Choose the subform control from the controls.
5) Place the subform control on the main form at the proper place and adjust its size reasonably to show an adequate quantity of data.
6) If you have already created a form to act as subform then we can place that form in this control form the wizard opened by Microsoft Access as you place the subform control on the main form. If no subform has already been designed then choose *Use Existing Tables and Queries* and click *Next*.


SubForm Wizard

You can use an existing form to create your subform or subreport, or create your own using tables and/or queries.

What data would you like to use for your subform or subreport?

(●) Use existing Tables and Queries

( ) Use an existing form

Contact Details
Contact List
Getting Started

7) Select table and fields to be displayed on the subform and click *Next*.
8) Tell Microsoft Access how the form will show the data. It will follow a pre-defined relation or you can create a new relation. The click Next button on the subsequent dialog boxes.
9) Next dialog box will ask you for the name of the subform. Type in a descriptive name for the subform and click *Finish*.
10) A new form will be created in the database and the same will be displayed in the subform control on the main form. You can modify the design of the subform as per your desire and requirement.

In this way you can create a Master-Detail form using subform.

13.14 Adding Subtotals and Totals from Subforms

Often it may be the requirement to calculate totals, subtotals or different multi-record level arithmetic on form the subform. In this section we will discuss that how we can do these. For example you want to add the total for quantity on the form. We can create a control on the subform which can calculate and show the arithmetic. then if it is the requirement to show the same value on the main form then create a control on the main form which refer to the control on the subform.

To perform multi-record calculations like total, average, count, etc. Microsoft Access provide different functions known as aggregate functions. Some of these functions are listed in Table 14.

Function	Description
Sum()	Totals the values.
Count()	Counts the values.
Avg()	Averages the values (sum divided by count).
Min()	Calculates the smallest value (for numeric values), the earliest date (for date values), or the first value in alphabetical order (for text values).
Max()	Calculates the largest value (for numeric values), the latest date (for date values), or the last value in alphabetical order (for text values).
First()	Uses the value from the first record.
Last()	Uses the value from the last record.

Table 14 Aggregate functions

To calculate the total of a field we will use Sum() function and other aggregate functions for desired calculations and results.

To calculate the total of Quantity field we will use the following expression:

```
= Sum([Quantity])
```

If we want to have the sum of total amount depending upon the fields Quantity and UnitPrice then we will use the following expression:

```
= Sum([Quantity] * [UnitPrice])
```

And you can use other function like these in the Control Source property of the controls on the forms.

13.15 Referring to a control on a subform

There may be the situation when we need to display the values from the subform controls in the main form controls. To do this we have to make a reference to the control in the subform in the Control Source property of the control on the main form.

To refer a control on the subform in the main form is as simple as writing simple expression in the Control Source property of a control. It has the following syntax:

```
= [subform control name].Form![control name]
```

We can explain it with an example.

Let we have the main form named as frmOrderMainForm. We have placed a subform control on this for with the name ctrlOrderSubForm. There is a Text box control on the subform in this control with the name txtSubTotal. Now we want to refer to this control in the main form. Then we will use the following expression:

```
= [ctrlOrderSubForm].Form![txtSubTotal]
```

In this way we can access any of the control on the subform in the main form and access and manipulate its value.

Task 6

User previously created Contacts database. Create a Master-Detail form for Orders. If we choose a Contact from the main form then all its related records for orders by that contact should be displayed in the detail form. The detail form should show the total amount for each order and the total amount for all orders of that contact.

14 Creating and Making-over Reports (Chapter 22)

Learning Objectives

After the completion of this chapter students will be able to:

- Create and use reports
- Create report manually or by using Wizard
- Modify reports using Design View
- Add headers and footers to the report
- Formulate the groups and sort the data in report
- Calculate totals and subtotals based upon groups of data
- Set the page layout for a report
- Export a report in different data and document formats
- Create charts in report
- Preview and print a report

Preface

All the database application developed are basically the part of Management Information System (MIS). Each organization which implements some sort of database application certainly wants to have some information at the end from this database. This information are generally known as reports. So we can say that the reports are the final product of any database application. The reports collect the data from the database and then perform some analysis and calculations on the data and display the results in some properly formatted manner. Reports can show the data as it, or show data summary, or show comments depending upon the data value and data nature. Microsoft Access provides the report as a database object. Using reports we can also get the printed output, or the final result converted to a PDF, or DOCX, or Web Page format. You can choose the way the data should be displayed and formatted in the reports. Reports can contain data from a single table or from multiple related tables. We can create reports from scratch manually or by using a Report Wizard. In this chapter we will discuss that how we can create reports, modify these reports and how to make the reports to have a good and impressive look.

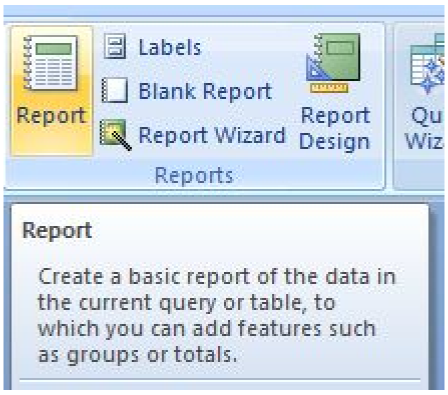
A reported created in Microsoft Access is generally consists of the five sections. These are Header, Group Header, Detail Section, Group Footer and Footer. Headers and Footers are the optional part of the report. Only the detail section is a must part of the report because it is used to display all the data. In this chapter we will learn to create reports.

14.1 Creating Reports Automatically

Microsoft Access provides the facility to generate reports based on tables or queries in a very easy way. For this purpose you just have to press a button and the report will be generated by Microsoft Access. Although this process does not provide you with much options but it is a very convenient way to create and initial report.

You can create such report by following the steps given below:

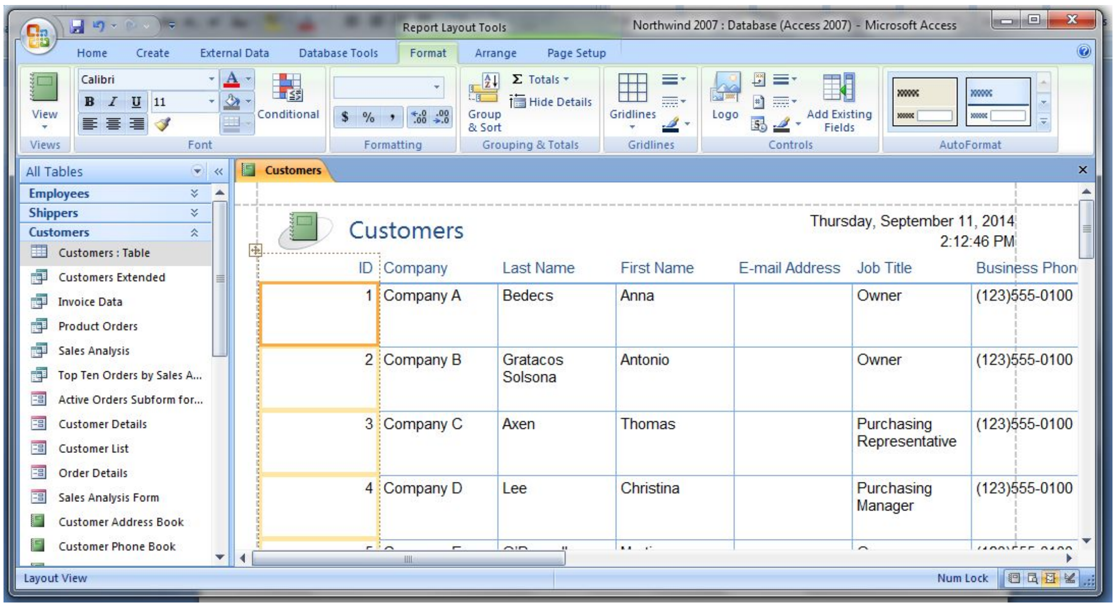
1) Select the table or query for which you want to create the report.
2) Now from the Ribbon *Report group* click *Report*.



Report

Create a basic report of the data in the current query or table, to which you can add features such as groups or totals.

3) Microsoft Access will create the report. This report will have a tabular format and look. This report will include all the fields from the Table or Query you selected in step 1. Figure 30 shows such created report for Customers table.



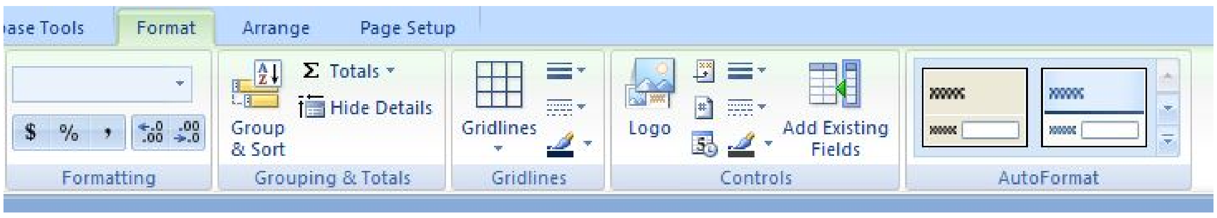
Customers

Thursday, September 11, 2014
2:12:46 PM

ID	Company	Last Name	First Name	E-mail Address	Job Title	Business Phon
1	Company A	Bedecs	Anna		Owner	(123)555-0100
2	Company B	Gratacos Solsona	Antonio		Owner	(123)555-0100
3	Company C	Axen	Thomas		Purchasing Representative	(123)555-0100
4	Company D	Lee	Christina		Purchasing Manager	(123)555-0100

Figure 30 Report example

4) You can format and modify the report using *Layout View* or *Design View*. You can remove the fields which you don't want to be in the report. You can also adjust the width of fields to adjust its contents properly. You can also set the borders and other properties.



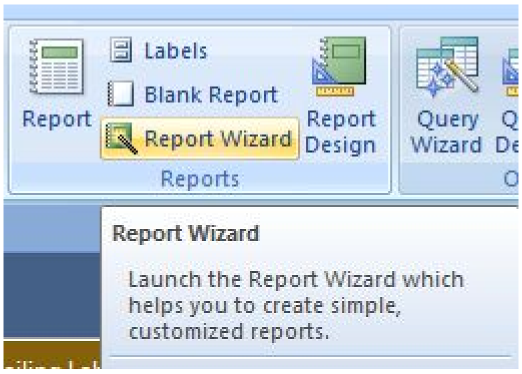
This is an automatic process provided by Microsoft Access to create a report.

14.2 Running the Report Wizard

In the previous section we have discussed that how we can create a report automatically in Microsoft Access. Here we will discuss to create a report using report provided by Microsoft Access. This is also an automatic way to create a report but is little more that that process described in the previous section. It is a convenient way to create a report using wizard by selecting fields from single or multiple tables or queries and also by defining the groups and subgroups. This process asks few questions and let you do some selections and put some values, all step-by-step and then at the end it creates a report according to you input and responses you made to the wizard.

In Microsoft Access you can create a report using wizard by following the steps given below:

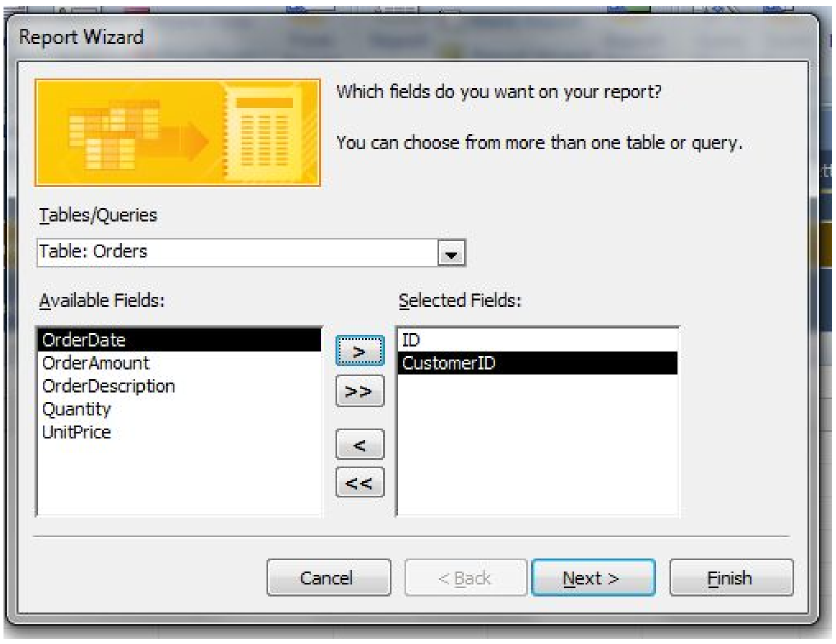
1) In the Ribbon *Microsoft Access* click the tab *Create*, then click the button *Report Wizard* in the group *Reports* as shown in the following figure.



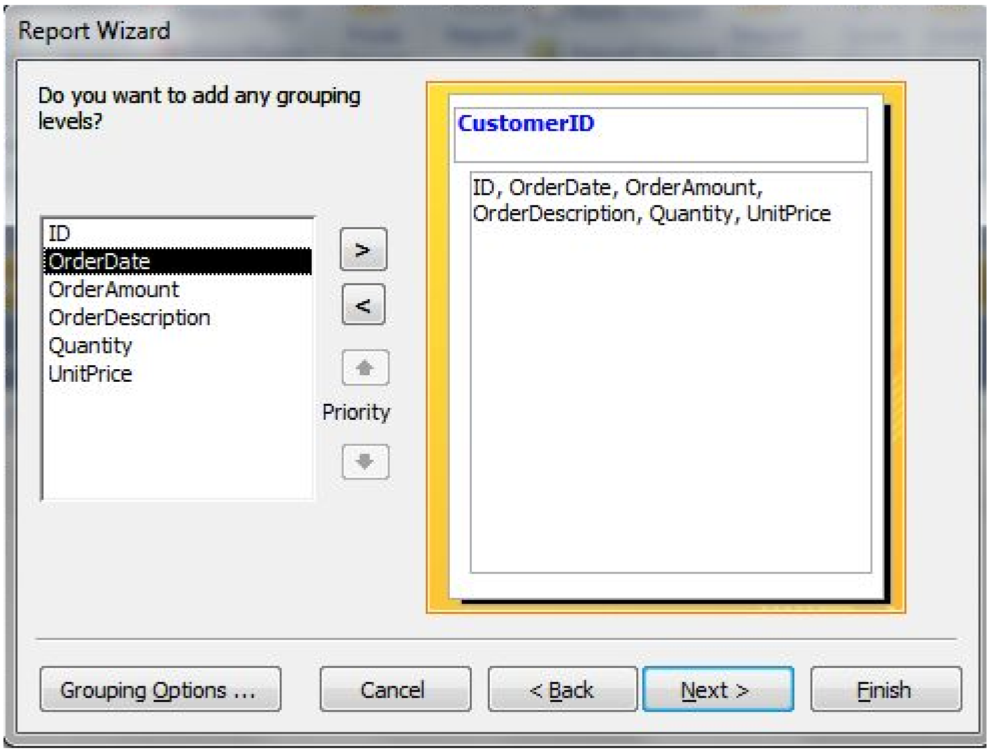
Report Wizard

Launch the Report Wizard which helps you to create simple, customized reports.

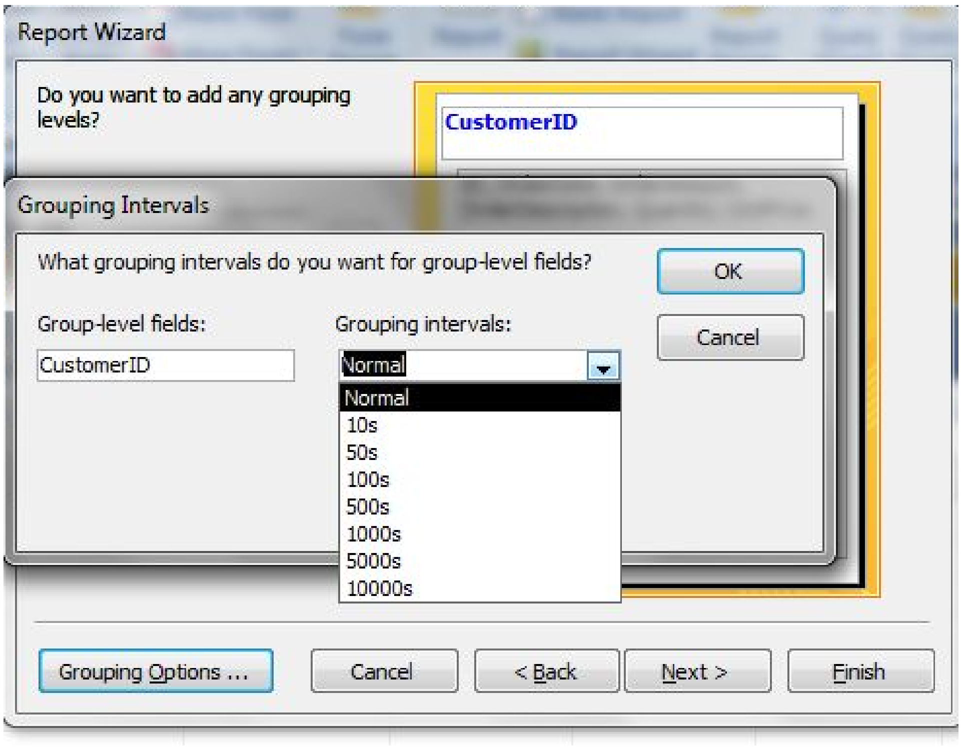
2) The dialog box *Report Wizard* will be displayed.



3) Select the tables and queries from the list one by one and choose the fields from those which you want to include in the report.
4) You can select a field by clicking > and all fields at once by clicking >> .
5) In the same way you can remove a field from the report by clicking < or all fields at once by clicking << .
6) Repeat steps 3 to 5 for all the tables or queries from which you want to include the fields in the report.
7) Click *Next* to show next dialog box. This dialog box will let you define any grouping levels. Again you can choose the fields on which you want to make groups in the report data. You can add as much grouping levels as you require sensibilly.

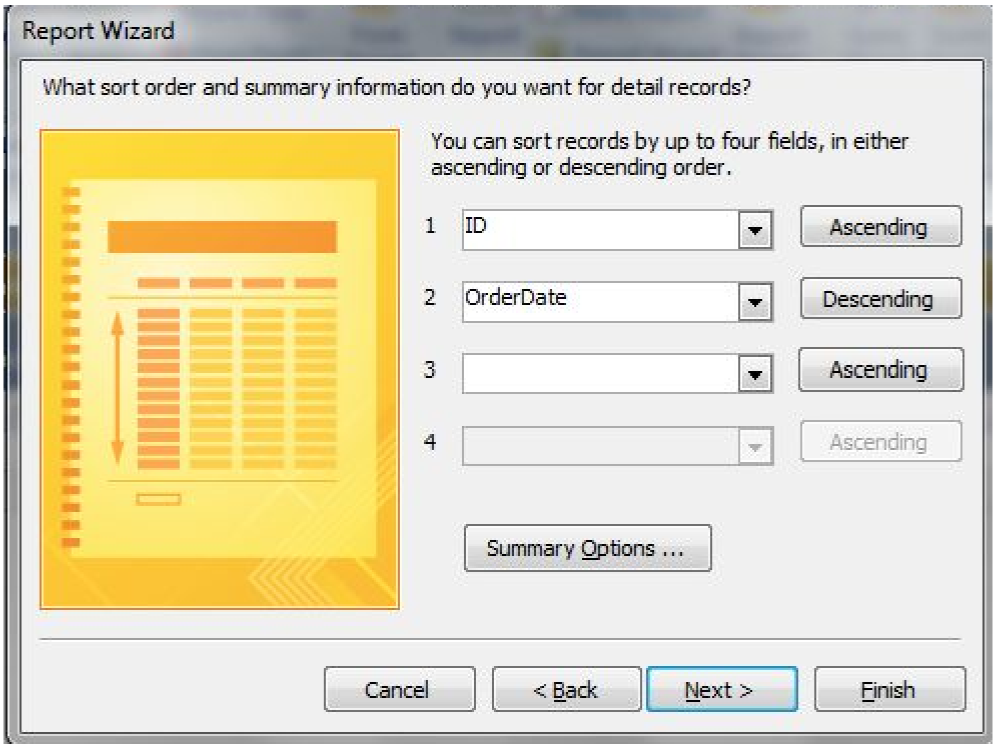


8) Now click Grouping Options ... to set the other grouping and sub-grouping options as shown the figure below.



9) You can set grouping options depending upon the data type. If the field is numeric then you can group on intervals, if the data is of date type then you can set the interval on weeks, months, years, etc. Click *OK*. Then click *Next*.

10) The dialog box will give you the option to set the sorting orders and options for the data displayed in the report. You can set the sort order as *Ascending* or *Descending* for multiple fields as per the requirement of report.



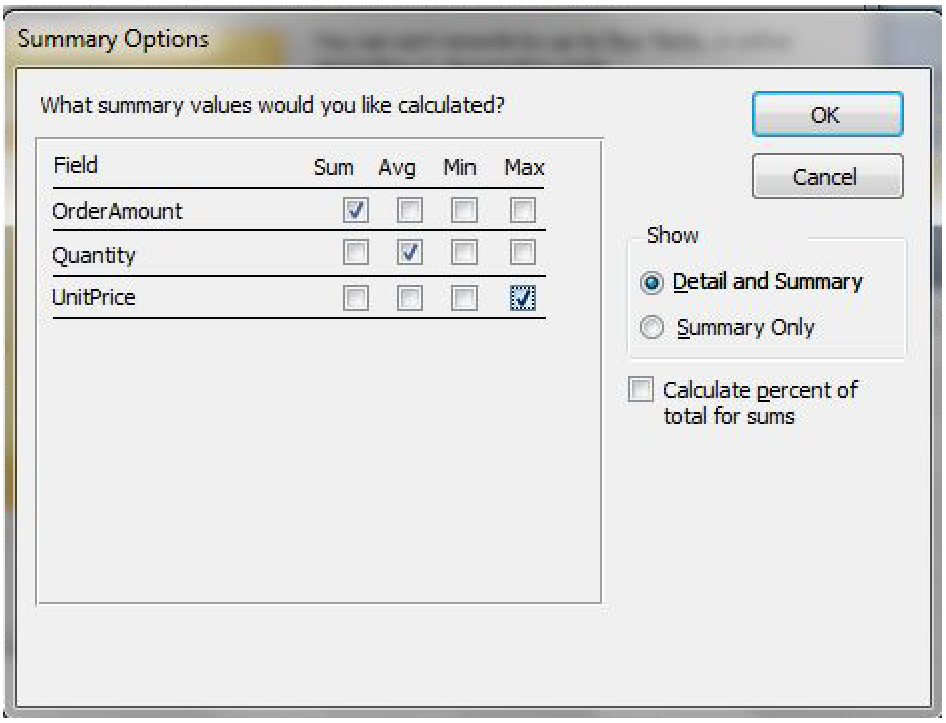
Report Wizard

What sort order and summary information do you want for detail records?

You can sort records by up to four fields, in either ascending or descending order.

1	ID	Ascending
2	OrderDate	Descending
3		Ascending
4		Ascending

11) Now click to set the summary options of the report. You can set different summary options as shown below:



Summary Options

What summary values would you like calculated?

Field	Sum	Avg	Min	Max
OrderAmount	✓			
Quantity		✓		
UnitPrice				✓

Show
[✓ Detail and Summary]
Summary Only

Calculate percent of total for sums

12) After setting the summary options click *OK*. Then click *Next* to show the next dialog box.

13) Then click *Next* and set report appearance settings and page setups.

14) Then again click *Next* to set the layout and format.
15) Now you will be on the final setup. Here you can set the title for the report. You can preview the report and even try to modify the report.

Following these steps you have created a report using wizard in Microsoft Access.

14.3 Editing Reports in Layout and Design View

Database reporting is a very rich and powerful feature of Microsoft Access. we can create reports by fully automatic way, by using report wizard or by totally manual method starting from scratch. The better approach is to first to create a report using report wizard and then edit it and fine tune it in Layout or Design View.

The editing of reports in Layout or Design View is almost the same. The only difference is that in Layout View Microsoft Access displays a report having data populated in it and you can arrange, design and manage the fields accordingly. And in Design View, we are provided with the design and fields without data in them. Figure 31 shows a report in *Design View*.

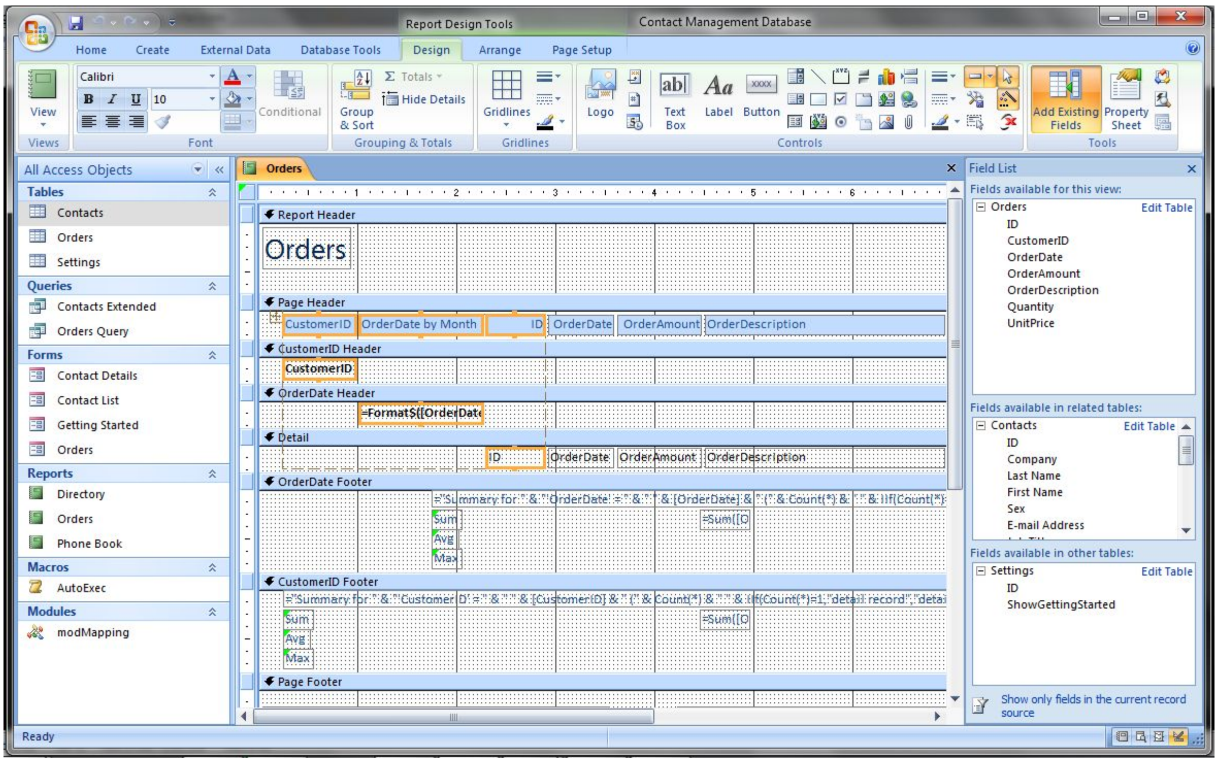


Figure 31   Microsoft Access report in Design View

We can modify and edit a report for different purposes. We may edit a report to add, edit, change layout or delete the pre-added fields or controls in the report. We can change the properties of the controls. To change the properties we select the control and from the Property sheet of the control we can set different properties. The properties include the format, font, size, color, etc. We can also create the calculated controls which follow the same process as we have discussed in the forms creation and editing.

A report has different sections like: *Report Header, Report Footer, Page Header, Page Footer, Group Header, Group Footer* and *Detail section*. Almost all the section names are self-explanatory and describe their functionality. Header and Footer sections of different type may be printed many times in the report. For example page header and footer will be printed on each page, group header and footer will be printed whenever a group starts and ends and so on.

You can change the properties of different sections by just double clicking them or by right click and select properties. You can add page numbers, totals and subtotal and date and time to the report. You can add the summary options and information to the report footer sections.

So you can modify and edit your reports for full customization in Layout or Design view in Microsoft Access.

14.4 Viewing Your Report

After creating the report, the ultimate purpose is to print the report for MIS purposes which helps in future decisions based on the database and information in the database.

When the report is completed then before printing it on the paper we have the option to check or view it on the display. Figure 32 shows the choices to view a report.

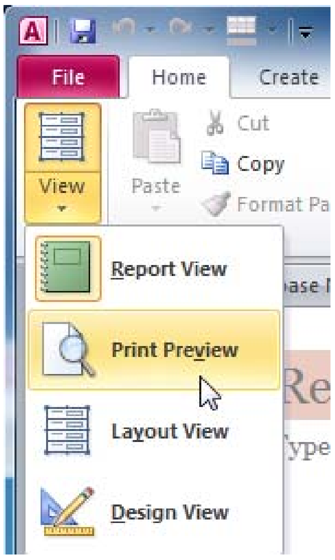
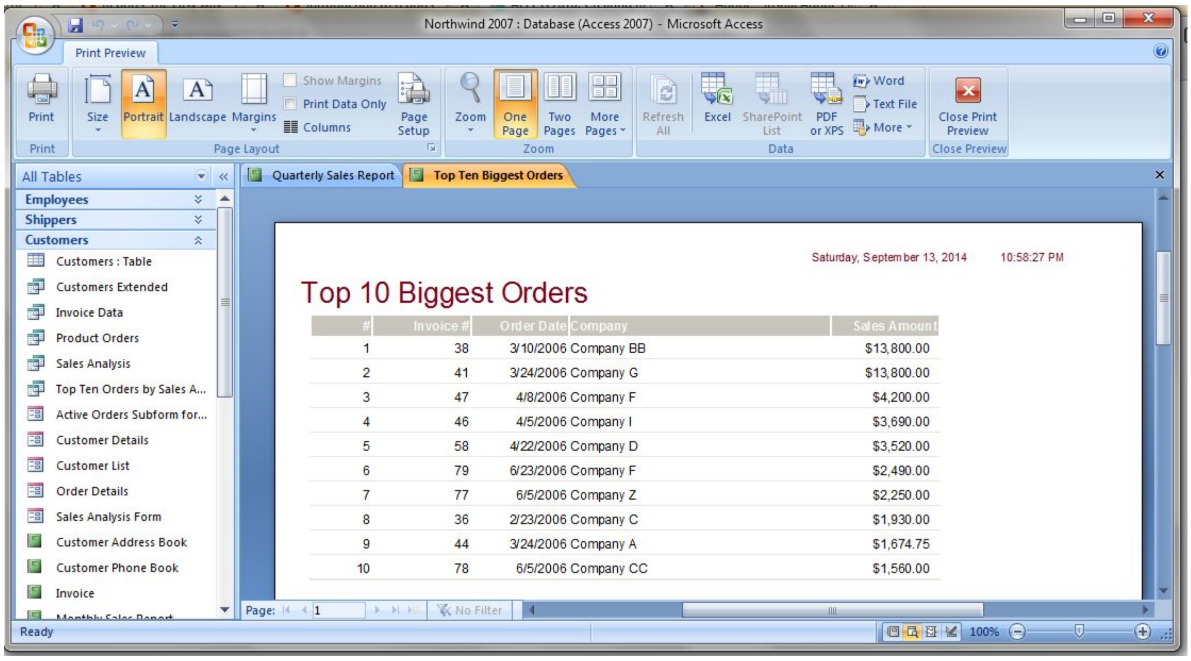


Figure 32 Views

When we see a report in the report view, then it is displayed as it will be printed on the paper. Then the Ribbon is related to the Report and provides different buttons and facilities related to the report view and data manipulation, sorting and filtering. A report in *Print Preview* will be shown as one is displayed in Figure 33.



Saturday, September 13, 2014 10:58:27 PM

Top 10 Biggest Orders

#	Invoice #	Order Date	Company	Sales Amount
1	38	3/10/2006	Company BB	$13,800.00
2	41	3/24/2006	Company G	$13,800.00
3	47	4/8/2006	Company F	$4,200.00
4	46	4/5/2006	Company I	$3,690.00
5	58	4/22/2006	Company D	$3,520.00
6	79	6/23/2006	Company F	$2,490.00
7	77	6/5/2006	Company Z	$2,250.00
8	36	2/23/2006	Company C	$1,930.00
9	44	3/24/2006	Company A	$1,674.75
10	78	6/5/2006	Company CC	$1,560.00

Figure 33 Print Preview

Using this print preview facility of Microsoft Access you can get the idea about the print out of this document even without getting it printed on the paper. In this way you are able to make changing and modification if the report is not as per your requirements and specifications. You can see whole page or even multiple pages at the same time on the screen.

In this preview mode you can adjust the page size, page margins, page orientation and page-layout settings. These options are provided in Page Layout Group on Print Preview tab in Microsoft Access Ribbon as shown in Figure 34.

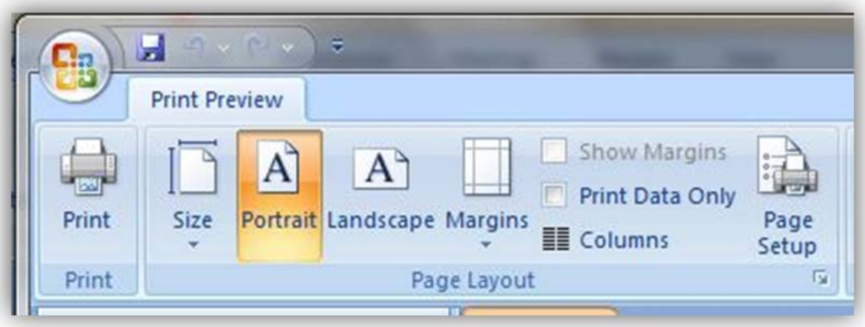


Figure 34 Print Preview

14.5 Previewing reports with parameters:

In Microsoft Access when reports have been designed then some of these reports may require some values to be provided before the reports start to print or view. These values are known as parameters. This can be achieved by creating prompts to get data from the users or by passing values from the form which initiate the report. If we have defined some parameters in the reports then Microsoft Access will automatically ask for the values of these parameters as soon as you initiate the report for viewing or printing as shown in Figure 35.

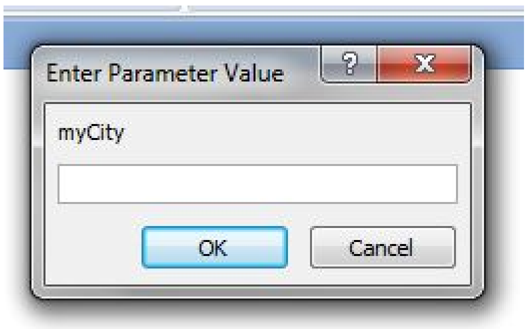


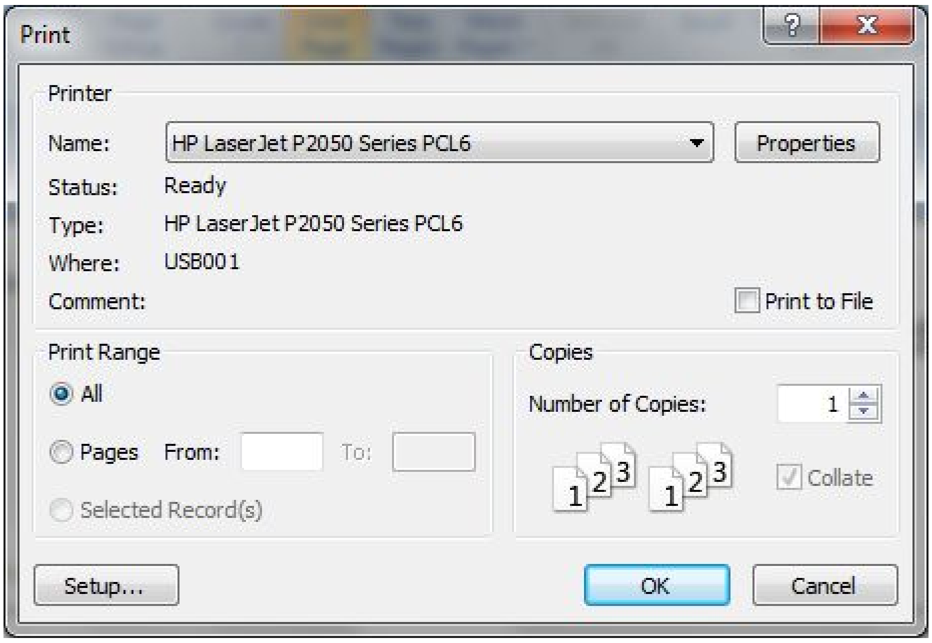
Figure 35 Dialog box – Parameter values

You can avoid this if you initiate such reports from the forms and then provide parameters from the form.

14.6 Printing the Report

After creating a report using wizard and then editing it for fine tuning, we view it in preview mode to adjust its different settings. Now is the time to get it printed on the paper.

We have multiple options to print the report. When the report is opened in Preview mode, or in *Layout View* or in *Design View*, we can print it. In *Print Preview*, a button for Print is displayed on the ribbon as shown in Figure 34, we can print the report by simply clicking this button. We can also print the report by clicking the tab *File* on the Ribbon and then click *Print* and then again clicking *Print*. We can also print the report by pressing <Ctrl+P>. All of these options will show the dialog box *Print*, which is the same for most of the Microsoft applications. This dialog box is shown in Figure 36.


Print

Printer

Name: HP LaserJet P2050 Series PCL6

Status: Ready

Type: HP LaserJet P2050 Series PCL6

Where: USB001

Comment:

Print to File [ ]

Print Range

[✓ All]

Pages From: To:

Selected Record(s)

Copies

Number of Copies: 1

[✓ Collate]

Figure 36 Dialog box - Print

From this dialog box we can select our printer, can also adjust the printer properties, can select range of pages of the report to be printed, can adjust the number of copies of the selected pages. After setting all these values, click the OK button. The report will be sent to the printer by Microsoft Windows. If the printer is ready then you will get the report printed on the pages.

14.7 Creating a PDF, XPS, HTML or other file of your report

In the previous section we have discussed that how we can get a report printed. Sometimes it is not required to get the report printed on the paper. The report may also be produced in soft form using different document formats. These formats include PDF, Doc, Docx, XML, XPS, etc. Following this method we can share our report online. We can distribute our report electronically. You don't need to do anything complex to create such documents from the reports. Microsoft Access provides very simple mechanism to achieve this. Microsoft Access can export the report in any of the format which user requires. When we have opened our report in the Print Preview, then there is a group Data, on the Ribbon which have the options to export the data in these formats. It can be seen in Figure 37.

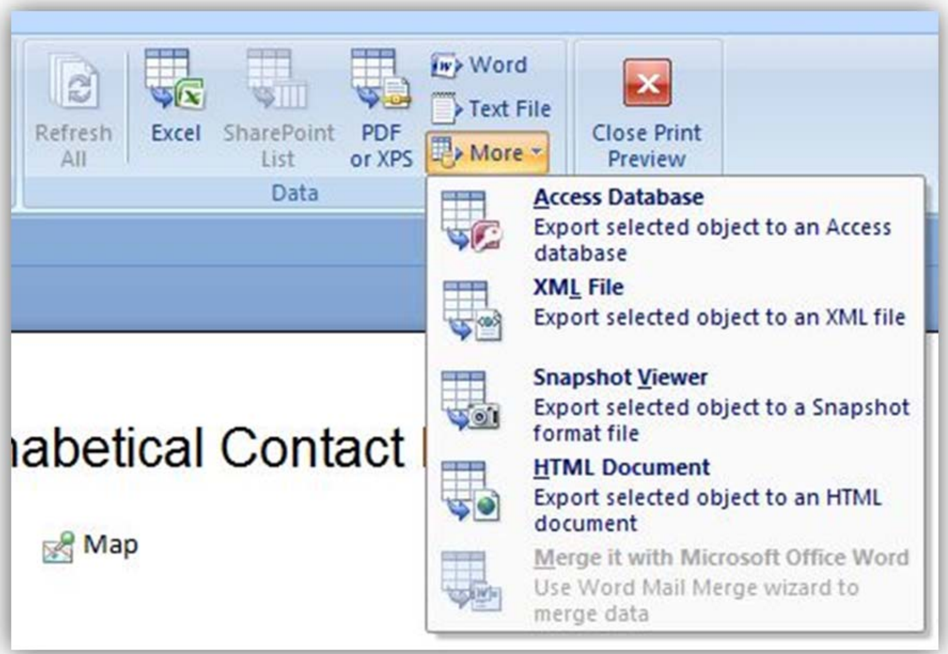

Refresh All
Excel
SharePoint List
PDF or XPS
Word
Text File
More
Close Print Preview
Data
Access Database
Export selected object to an Access database
XML File
Export selected object to an XML file
Snapshot Viewer
Export selected object to a Snapshot format file
HTML Document
Export selected object to an HTML document
Merge it with Microsoft Office Word
Use Word Mail Merge wizard to merge data
habetical Contact
Map

Figure 37 Export formats

When you click any of these buttons, Microsoft Access will display a relative dialog box which will ask you save the file in some destination or specify the destination of the exported data depending upon the selection of your export document type.

Task 7

Open the database *NorthWind* provided by Microsoft Access as sample database. Open all the reports provided by the database and write down the field level and table level description of each component included on these reports.

15 Database Housekeeping (Chapter 24)

Learning Objectives

After the completion of this chapter students will be able to:

- Identify the growth of database size
- Identify how to compact the database
- Understand why to compact the database
- Explain why to compact the database
- Take backup of the database
- Find the problems arising within the database
- Restore a database from the backup

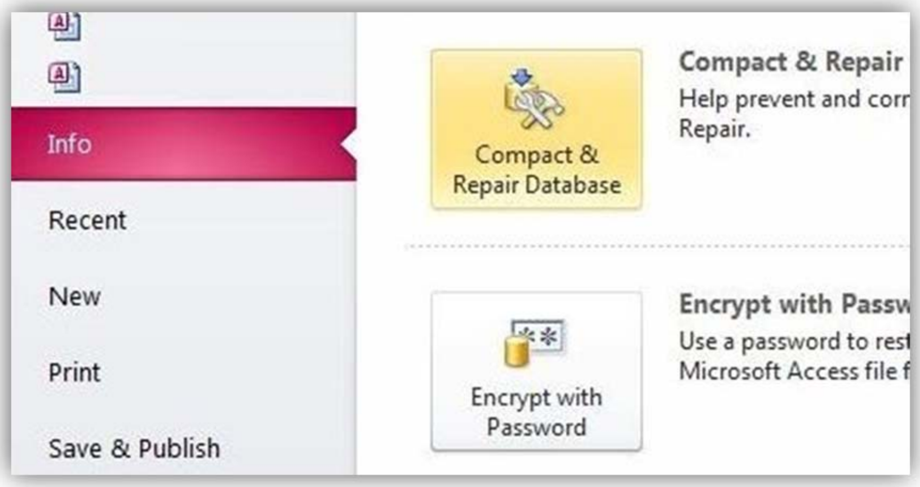
Preface

As the time pass on and database application is being used for a long time the database size increases. Same is the case with Microsoft Access databases. These databases increase in size day-by-day and will use more space on storage media. And as the database is used by multiple users and data is modified deleted and added, the chances of data corruption increase. It is required to keep the database neat and tidy. So it is a good practice if the data is backed up on regular basis. Microsoft Access provides the facility to compact the data. This process reduces the size of data making the backup to consume less storage space. We will discuss in this chapter the use compact and repair database tool provided by Microsoft Access.

15.1 Compacting and Repairing your Database

With the passage of time database grows in size as users enter data into the database. There may be developmental changes in the database. Some reports may be added as per the requirements of day-by-day management. With every change and any type of change, Microsoft Access store new information in the database file. The new information is made active and old information is marked as old which is to be deleted but the old information are kept by Microsoft Access database and not deleted straight forward. To remove these old information from the database file Microsoft Access has the process of compacting a database. This process also repairs the errors that may have occurred during the growing of database file. The process of compacting and repair the database can be performed in the following steps:

1) Open the database, if it is not already opened.
2) Make a backup/copy of the database before the process of compacting. It is an optional step but it is always a good practice to make a backup before compacting.
3) You have to close all the open objects as first step as the process operation cannot access an object if it is open. These objects include tables, queries, form, reports, VBA modules and editors, etc.
4) From the Ribbon *Microsoft Access* click the tab *File*, then click *Info*. The following options will be displayed.



5) Choose *Compact & Repair Database* from the options shown.
6) Microsoft Access will start compacting the database. A progress bar in the status bar will show the progress. It will take time depending upon the size of the database. If there is some error then error or warning messages will be displayed. If the progress bar shows a 100% complete then it means that compacting has been done perfectly and no repair is needed.

After the compacting is complete the database will be re-opened automatically and may require some login information if set so.

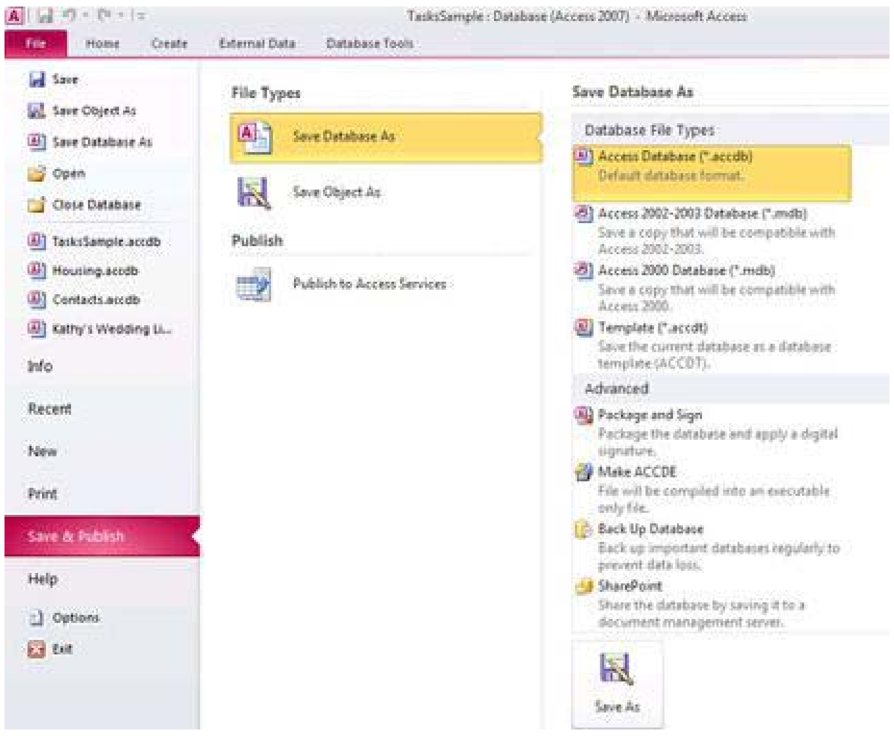
15.2 Making Backups

Electronic and paperless environment has made our life very easy. But there is one major flaw. If the electronic equipment goes down or starts malfunctioning then we may lose our important data and documents. The same can happen with a database file. To avoid such disaster it is a good practice to make the backups of the data and database on regular basis and especially if some change have been made by the development team. If such disaster takes place then you will have to type all the data again.

If you are a good MIS manager, then you certainly will make sure that all the important documents and data on your hard disk is being backed up regularly. The backups can be made on CDs, Tapes, DVDs or on additional or External Hard Drive.

The process to make a backup of the database is as follows:

1) Open the database, if it is not already opened.
2) From the Ribbon *Microsoft Access* click the tab *File*, then click *Save & Publish.*



Default database format.

Save a copy that will be compatible with Access 2002-2003.

Save a copy that will be compatible with Access 2000.

Save the current database as a database template (ACCDT).

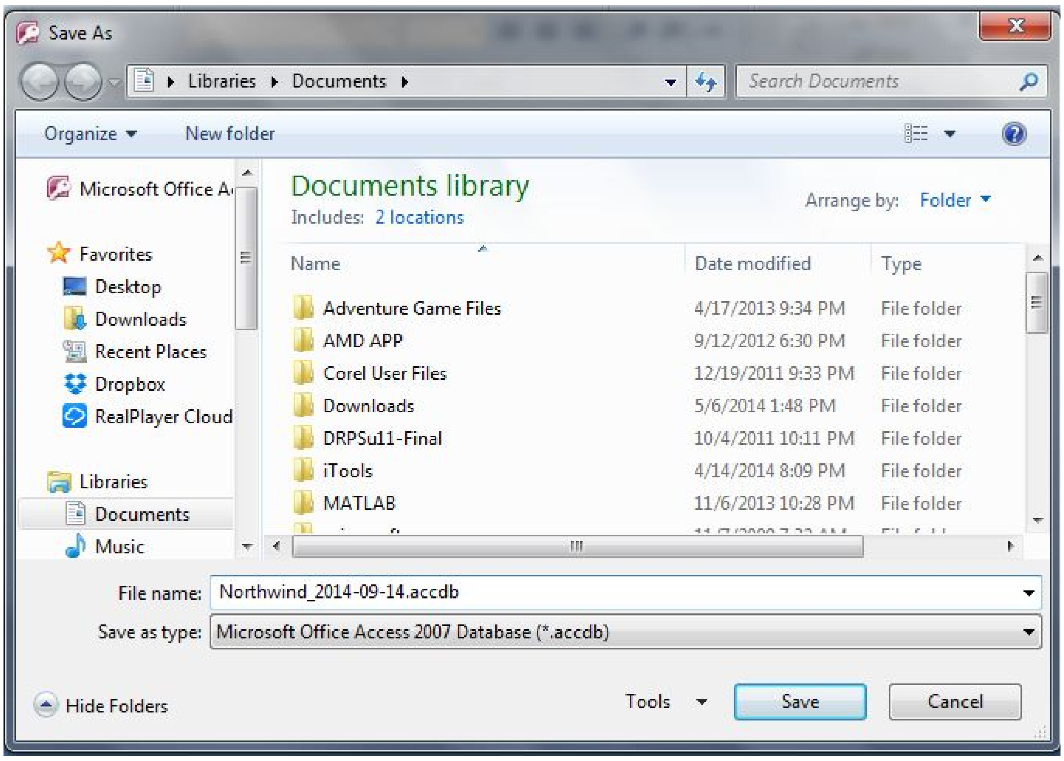
Package the database and apply a digital signature.

File will be compiled into an executable only file.

Back up important databases regularly to prevent data loss.

Share the database by saving it to a document management server.

3) Click *Save Database As* and chose *Access Database* from the right pan.
4) The click *Save As*.
5) In the dialog *Save As*, a new database name will be displayed with the current database name and today's date padded at the end of the database name as shown in the following figure the new name for the Northwind database made on the date September 14,2014.



Name	Date modified	Type
Adventure Game Files	4/17/2013 9:34 PM	File folder
AMD APP	9/12/2012 6:30 PM	File folder
Corel User Files	12/19/2011 9:33 PM	File folder
Downloads	5/6/2014 1:48 PM	File folder
DRPSu11-Final	10/4/2011 10:11 PM	File folder
iTools	4/14/2014 8:09 PM	File folder
MATLAB	11/6/2013 10:28 PM	File folder

File name: Northwind_2014-09-14.accdb

Save as type: Microsoft Office Access 2007 Database (*.accdb)

6) Select the proper path and folder where you want to store the backup of the database and change the database file name if you want to.
7) Click *Save*.

Backup of the database will be created in the specified folder with the specified name of the Microsoft Access database.

15.3 Converting Databases

As the Microsoft Access is a database which exist for many years in the past. It has gone through many evolution phases. During this evolution process Microsoft Access has changed the file format which it use to store the data. Microsoft Access 2010 and Microsoft Access 2007 use the same file format and can open the database file interchangeably. But the earlier versions cannot open these database files and the database files created in earlier versions also require to be converted to Microsoft Access 2010 or Microsoft Access 2007 file format to be opened in these versions of Microsoft Access.

When opening a database file from the older versions of Microsoft Access, we just have to tell the Microsoft Access that of which version of Microsoft Access file we want to open. Follow the given steps to open and older database in Microsoft Access 2010 or Microsoft Access 2007.

1) Open the database.
2) From the Ribbon *Microsoft Access* click the tab *File*, click *Save* and *Publish.*
3) Click *Save Database As* and choose *Access Database* from the right pan.
4) Click *Save As*.
5) In the dialog *Save As*, a new database name will be displayed with the current database name and today's date padded at the end of the database, choose the new name as per your requirement.
6) Select the proper path and folder where you want to store the new converted database.
7) Click *Save*.

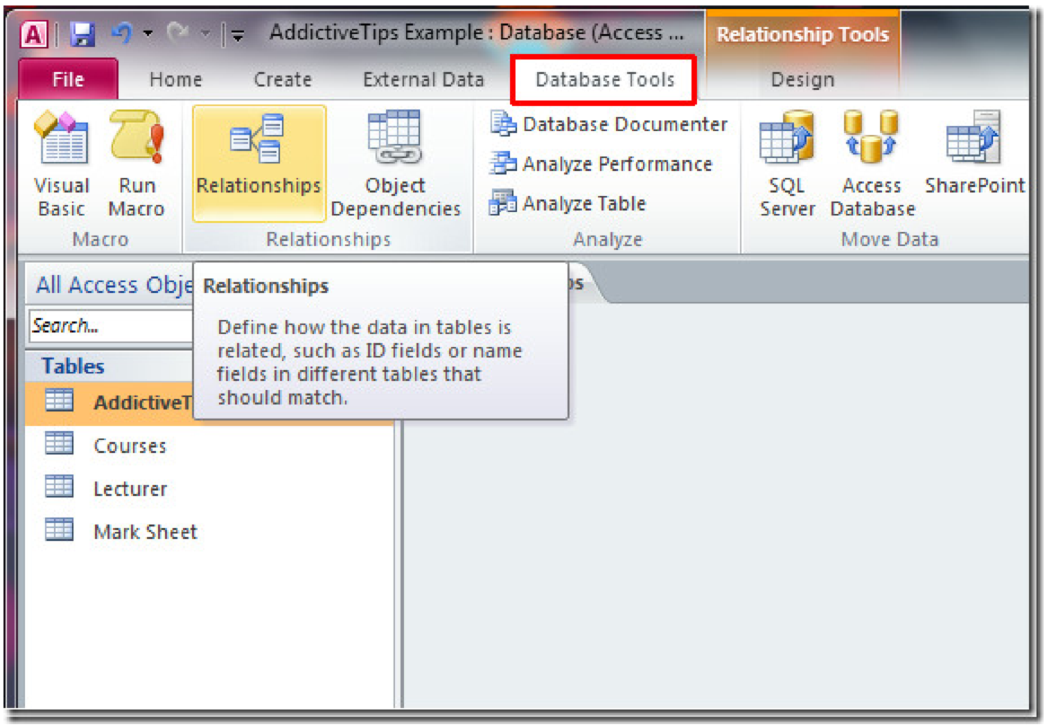
You have converted an old version database to a database of Microsoft Access 2010 or Microsoft Access 2007 version database.

You can follow the same steps if you want to convert a database file of Microsoft Access 2010 or Microsoft Access 2007 format to older versions format. Just in the step 3 choose the required Microsoft Access version and format.

15.4 Viewing relationships in the Relationships window

Microsoft Access is a relational database. It handles the data as relations. User/developer can establish different types of relationships between the tables depending upon the Primary and Foreign keys of the tables. You can see and examine these relationships graphically at any time.

To display the relationships from the Ribbon *Microsoft Access*, select the group *Relationships* and then click *Relationships*.



Relationships

Define how the data in tables is related, such as ID fields or name fields in different tables that should match.

Figure 38 Button - Relationships

This will display the relationship window showing the tables and their relations in your database. Figure 39 shows a relationship window for Northwind sample database provided by Microsoft Access.

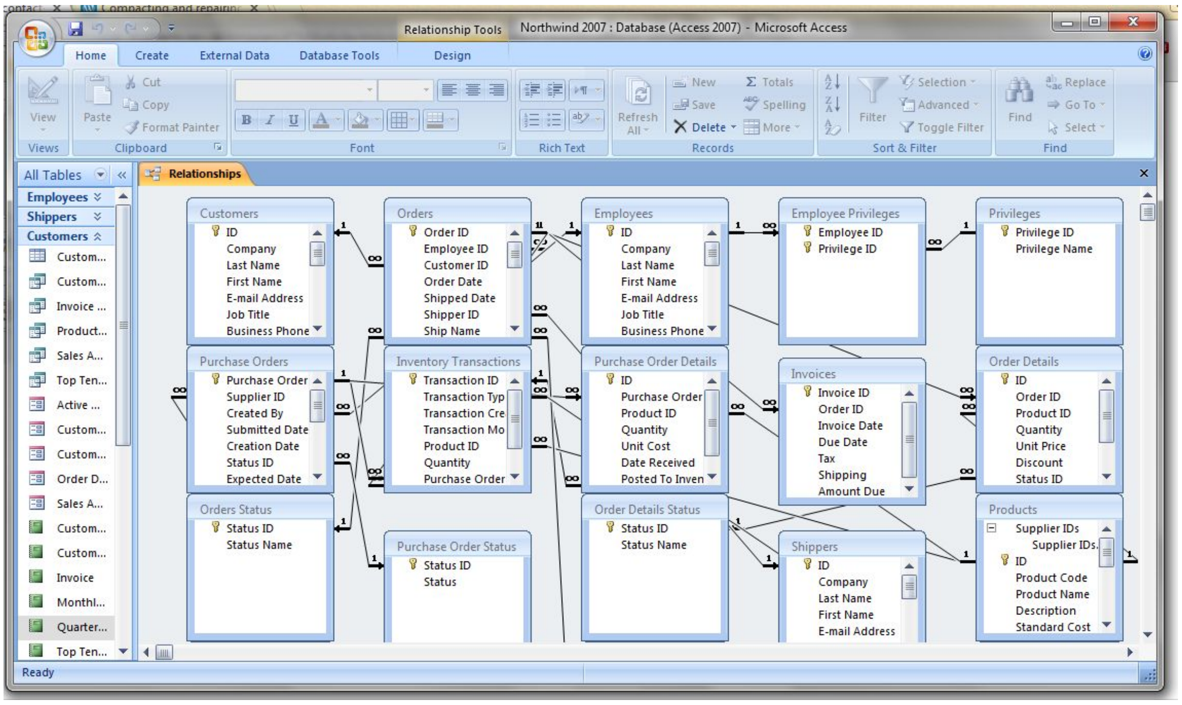


Figure 39 Relationships Overview

You can examine the existing relationships of the tables and queries, and can even create new relationships and delete the existing relationships avoiding the data dependencies and data integrity.

15.5 Documenting your database

The documentation of any application is very important and vital. The same is the case with database applications. These are meant for the proper usability of the application as well as for the successful management and database administrations.

In this process we create a document which explain and describe each object included in the database. It also contains the details about the controls which are placed on different object. The details include the important information like for which purpose a control is used, from where it will receive data, what will be the range of data and what will be the domain of data and so on.

Microsoft Access provides a tool named as Database Documenter which creates the documentation for your database. You can use this tool be following the steps given below.

1) Open the database.
2) Close all the open objects
3) From the Ribbon *Microsoft Access* choose the group *Analyze* and then click *Database Documenter*.

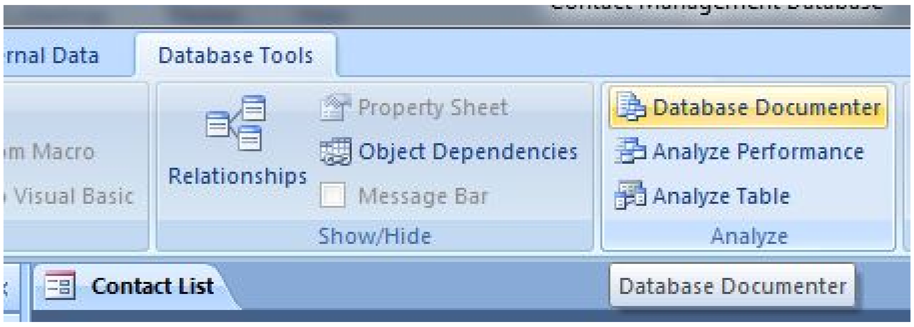


Figure 40 Button - Database Documenter

4) The dialog box *Documenter* will be displayed as shown in Figure 41

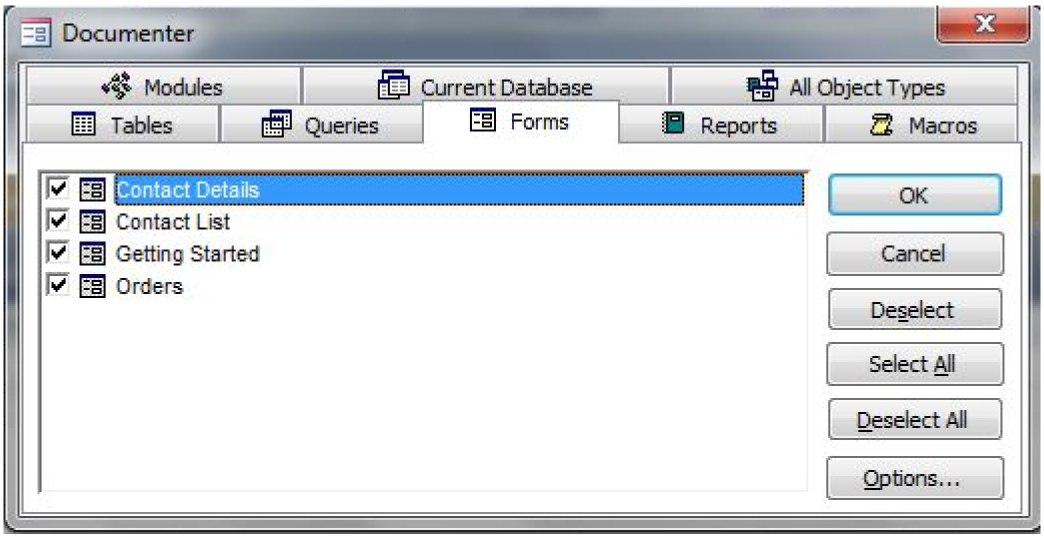


Figure 41 Database Documenter

5) From this dialog box select different tabs and select the objects for which the documentation report is needed.
6) Click *OK.*

Microsoft Access will create the documentation report for the selected objects. It is a good idea to select one object at a time and then create its documentation report because sometime the report will span many pages.

Task 8

Open the Northwind sample database and perform the following activities:

1) Compaction of the database.
2) Make a backup of the database with the name suggested by the Microsoft Access.
3) Save a copy of this database in the file format *Microsoft Access 2003*.
4) Display the table relationship window and make a report about the relationship types and the keys/fields participating in these relationships.
5) Create documentation reports for:
   a. Any 2 forms
   b. Any 2 tables
   c. Any 2 queries
   d. Any 2 reports

16 Using VBA with Microsoft Access (Chapter 25)

Learning Objectives

After the completion of this chapter students will be able to:

- Understand how to code and program
- Understand and use Visual Basic for Applications (VBA)
- Understand and explain VBA code and syntax
- Create and use variables and constants in VBA code
- Create, Run and Test VBA Code and Procedures
- Write decision making code
- Write code for repeated tasks
- Show responses to the user
- Open different Microsoft Access objects from within the VBA code
- Create, use and manipulate data using record sets
- Debug the VBA code

Preface

Microsoft Access is a complete Database Management System. It provides the tools to create different objects in database as well as tool to write your own code to enhance the functionality of the built-in objects. So it is a complete development environment to develop database applications. A programming language named Visual Basic for Application (VBA) is provided in Microsoft Access to write your own code and programming in Microsoft Access. When designing a database application we start by creating Tables, Queries, Forms, Reports, etc. Then at a point we need some automation in our application. Here the VBA comes into action. VBA is used to enhance the functionality and to add the automation in the Microsoft Access and other Microsoft Office applications. With the help of VBA coding you can develop customized solutions to almost all the MIS problems. You should know the syntax, techniques, other requirements and limitations posed by VBA. VBA follows almost all the concepts being used by other programming languages. If you have a programming background in some other language then it will be quite easy to grasp the concepts in VBA programming. In this chapter we will discuss about the concepts and techniques required to do programming in VBA.

16.1 Finding VBA Code

VBA has the statements written using its editor. In Microsoft Access VBA code can be found in two different paces.

- Class modules (code written for forms)
- Standard modules

We have studied these concepts in our class sessions.

16.2 Opening a class module

You may need to review the VBA code to make changes or for some other purpose. To view/open the code, first of all open the object in *Design View* for which code is written. Then click the tab *Design* and then *View Code*. You can also go to the code using Property sheet and then Event tab.

This topic has been discussed in class room sessions.

16.3 Creating or opening a standard module

In a database in Microsoft Access there may be certain chunks of VBA code which are not attached to any object. These are known as Standard modules. This VBA code is accessible to all the objects in the database. We have studied to built and use such modules in our class.

16.4 Enabling VBA Code

Microsoft Access has a built-in security mechanism to protect you from unknown risks. These risks may be posed through VBA code. Like all other programming languages, a programmer can write a program which is useful or may be harmful. So Microsoft Access doesn't run a VBA code directly. Whenever you open a database in Microsoft Access which has VBA code in it then Microsoft Access will display a warning message about it and asks you to enable it or not. You can choose the option as per the situation.

16.5 VBA Syntax

Just like any other programming language VBA also follows some syntax rules. The sentences written in VBA are called statements. All the statements written in VBA code must be written according the syntax rules of VBA. These syntax rules are too strict, and if we write any statement even with a minor mistake then the statement will not be executed and an error message will be generated by Microsoft Access.

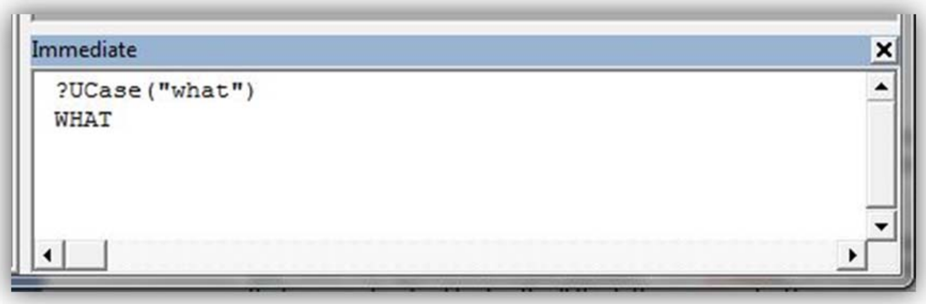
We have studied and discussed a lot about these syntax rules in our class.

16.6 Writing Your Own VBA Procedures

VBA provides many built-in functions and procedures to be used in programming in Microsoft Access but the actual power is that we can write our own functions and procedures. We have learnt in our class that how we can write and use our own VBA functions and procedures.

16.7 Passing arguments to procedures

In Microsoft Access VBA provide many functions to perform useful tasks. We often use these functions in writing VBA code and expressions. Let we consider a built-in function UCase(). This function gets a string as its argument and then convert it to upper case. So the argument is a number or text which we give to a function and the function perform some processing on it and then give back the results or produce some output by itself. Whenever we call a function and need to provide it arguments then we place the arguments in the parentheses following the function name. In Figure 42 a call to UCase() function is shown in the Immediate window with the result.



```
?UCase("what")
WHAT
```

Figure 42 Passing argument to a function

We can also create our own functions and procedures which can get arguments and then display or generate results. When writing our code then we define that how many and what type of arguments our function or procedure will accept. The syntax of defining a procedure and function to accept arguments is as follows:

```
Sub name(argument list)
    {body of the procedure}
End Sub

Function name(argument list) As type
    {body of the function}
End Function
```

The different between a procedure and function is that the function returns a value and procedure doesn't. Argument list may consist of zero or more arguments. Arguments may have different data types like integer, string, date and time, Boolean, etc. The argument names and types are defined in the parentheses. For example if a procedure is to use three arguments of type integer, Boolean and string, then it will look like:

```
ASubProcedure (arg1 As Integer, arg2 As Boolean, arg3 As String)
    {body of procedure}
End Sub
```

If the argument is of type String then it should be passed by enclosing it in double quotes if it is literal and only name of identifier if it is assigned to an identifier.

16.8 Returning a value from a function

As we have discussed earlier that we can write code in two different type of modules. One is known as procedure and other is function. The difference between these two is that a function type module returns a value after processing and the procedure doesn't.

The mechanism to return value from the function is that somewhere in the body of the function we write the following statement:

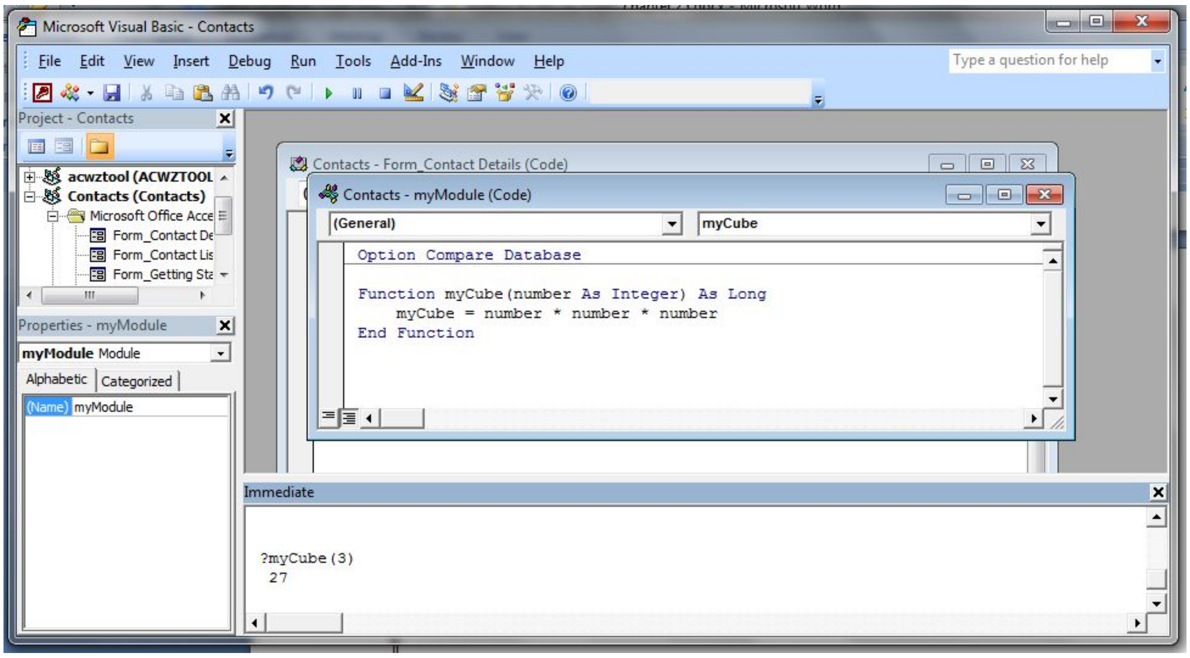
```
functionName = value
```

here the functionName is the name of the function you created and the value represents the value which the function will return.

If we write a function myCube() which returns the cube of an integer argument passed to it, then this function will look like as follows:

```
Function myCube(number As Integer) As Long
    myCube =  number * number * number
End Function
```

Then we can use this function in immediate window as:



```
Option Compare Database

Function myCube(number As Integer) As Long
    myCube = number * number * number
End Function
```

```
?myCube(3)
 27
```

Figure 43   Function myCube() and its use in immediate window

Using this technique you can write any function to compute a simple or a much complex expressions and return the results.

16.9 Creating Variables and Constants

In all the programming languages we have to store data temporarily for processing purpose for referring the data within the code. For this purpose we use the concept of variables and constants. Variables denote such objects which value can be changed or modified within the code but the value of a Constant cannot be modified. Microsoft Access uses the same concepts in VBA programming.

The Constants and Variables are named locations in the computer memory (RAM). We can give any name to a variable or constant but good practice is that we should give such a name that describes the content and type of the data in that variable or constant. To name a variable or constant we have to follow some rules. These are as follows:

- The name will always start with a Letter (capital/small)
- Any combination of letters, numbers and Underscore (_) can occur after the first letter.
- There cannot be any punctuation mark, arithmetic operator or space in the name.
- Any keyword of VBA or Microsoft Access cannot be used as name of a variable or constant.

In VBA a variable can be created in many different ways. These methods are categorized as implicit declaration and explicit declaration. The following are few examples of implicit declaration of variables:

- Age = 23
- Qty = 6
- UnitPrice = 287
- Amount = Qty * UnitPrice

In the above examples Age, Qty, UnitPrice and Amount variables will be created automatically.

For explicit declaration of variables it is required to mention the data type to the variable before using it. In this sense it requires a little bit more effort but this effort pay in the sense of efficiency of the program. To explicitly declare we use the concept of Dimension and short Dim is used for this purpose. Dim declare a variable associating a data type with it.

The syntax for declare a variable explicitly is as follows:

```
Dim variableName As dataType
```

If we want to use a variable UnitPrice then we will need to perform the following two steps:

- Dim UnitPrice As Currency
- UnitPrice = 12.00

By using different data types you can declare variables explicitly.

To create a constant we replace the word Dim by Const and rest of the procedure is the same.

16.10 Scope and lifetime of variables

As we write functions and procedure we declare variables and constants within the code. All the variables and constants declared within the code have certain scope and lifetime. Scope means that where in the code this variable or constant are visible. We specify the scope of a variable at the time of declaration. This scope specification may be implicit or explicit. The scope specification keywords are “Private” and “Public”.

To specify the scope of a variable we use keyword “Private” or “Public” instead of “Dim” while declaring the variable. If we don’t use any scope keyword and just declare a variable using “Dim” then it is automatically assumed as “Private”. And all the variables declared implicitly are also considered as private.

All the variables which are Private are visible only within the module. And the variables declared using Public keyword are visible to all the modules.

Generally the lifetime of all the variables is till the life of the procedure. When the execution of the procedure is completed then all the variables created in that procedures are finished and even the same procedure is called again the variables will be created from the scratch.

16.11 Organizing variables into arrays

In VBA we declare variables to store data temporarily. Sometimes it is the case that we have to save many variables of the same type and to store the related data. To handle such situations VBA provides a way to group the variable in a collection which is known as Array. Array is just like a table. The multiple variables are stored under the same name having an index or subscript within the name. These subscripts indentify all the elements of the array uniquely. These subscripts also correspond to the positions of the elements in the array. For examples if the names of the students are stored in an array named Students, then we can refer to each element like Students(1), Students(2), etc. An array can have multiple columns like a table in the Microsoft Access database. Such arrays are called multidimensional arrays.

An array can be declared just like we declare a variable explicitly. We have to give the dimension size for each dimension of the array. The syntax to declare a one dimension array in VBA is as follows:

```
Dim arrayName(Dimension_Size) As type
```

This will create a single dimension array and number of elements in this array will depend upon the value of Dimension_size. The index/subscript of arrays in VBA starts from zero (0) and go to the Dinension_size specified when the array is declared. For example if we want to store the four directions then we can declare and use the array as follows:

```
Dim Directions(3) as String
Directions(0)=”East”
Directions(1)=”West”
Directions(2)=”North”
Directions(3)=”South”
```

If you want to use an array from the index 0 then you can do a simple trick. Declare the array with size 4 and then start to assign values from index 1 and never refer to index 0 in your VBA code. The proper way is to use the following statement before the very first module in the code:

```
Option Base 1
```

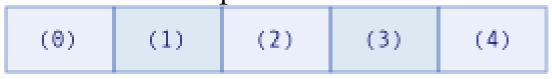
Then all the arrays will start from 1 rather than 0.

VBA also provides the feature of multidimensional arrays. The simplest multidimensional array is a table having two dimensions, one denoting rows and other columns in the table. There could be arrays of dimensions more than two dimensions.

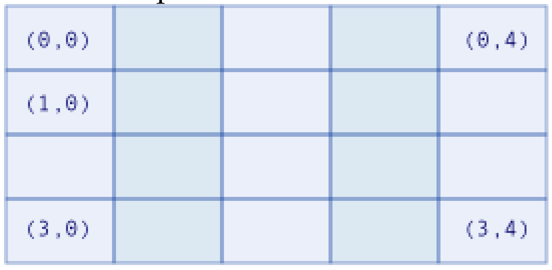
The following figures and examples describe the multidimensional arrays.

Example

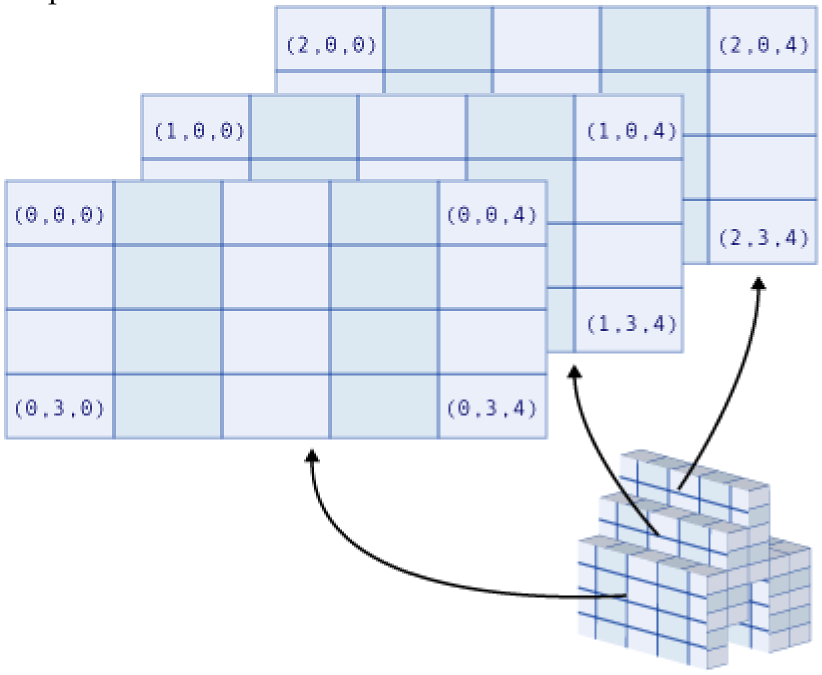
- Most of the arrays are one dimension arrays, e.g. StudentNames(4) which contains the names of the students. The following figure shows its structure and subscript for each element.



- Many arrays can be two-dimensional arrays just like a table. Its structure and subscripts are shown in the following figure.



- Few arrays will have three dimensions. It is little bit hard to conceptualize a three dimensional array. Following figure shows its structure and subscripts.



Although VBA provides the support for multi-dimensional arrays, but programmers very rarely use this feature in the programming and coding. The mostly used arrays are one-dimensional and two-dimensional arrays.

16.12 Making Decisions in VBA Code

As all the database applications are used for reporting purpose and decision making, so we have to make decisions and take actions according to certain conditions in our VBA code. To make decisions in the code is an important part of all the programming languages.

This decision making means that the code should be intelligent to respond to different situation in a proper and intelligent way. To implement such situations Microsoft Access provides decision making structures in VBA. These decision making structures are:

- if…endif
- if…else…endif
- if…elseif…elseif…   elseif…endif
- if…elseif…elseif   elseif…else…endif

We can write intelligent programs and code using these structures. These all structures base upon certain conditions, which are evaluated as True or False using conditional expression. A conditional expression generally has the following structure:

Value ComparisonOperator Value

The both values are information and the ComparisonOperator could be one of the operators given in the Table 15 showing operators and their description.

Operator	Description
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<>	Not equal to

Table 15 Comparison operators

Following are the few examples of the conditional expressions:

- [City]="Lahore" : will compare the contents of the City field with the string "Lahore" and returns the value as True or False.
- [Age]>=30 : will compare the value of Age field against the value 30. If the value of the Age field is greater than or equal to 30 then it is evaluated as True otherwise False.

Sometimes there are situations when we have to test more than one condition and we take action depending upon the collective effect of these conditions. We can combine the effect of more than one conditions using logical operators. Logical operators are shown in Table 16 along-with their description.

Operator	Description
And	Both are true.
Or	One or both are true.
Not	Is not true.
Xor	Exclusive, or: One — but not both — is true.

Table 16 Logical operators

Following are the few examples of the conditional expressions containing more than one conditions:

- [City]=”Lahore” And [LastName]=”Abeer”: will compare the contents of the City field with the string “Lahore” and contents of the LastName with the string “Abeer”. It will return True if the results of both the individual conditions are true and False otherwise.
- [Age]>=30 OR [EducationYears]>=16 : will compare the value of Age field against the value 30 and compare the value of the EducationYear against the value 16. If will return the value True if the result of any one the conditions is true and returns False if both the conditions are false individually.

You can combine as many conditions as you require and wish being clear in your mind that what the effect would be.

16.12.1 Using If...End If statements

When writing VBA code in Microsoft Access we can use decision structures. The simplest structure is If…EndIf structure. In VBA the syntax of the structure is:

```
If condition Then
   {statements}
End If
```

This structure will execute the statements within the if…endif block if the condition is true otherwise these statements will be skipped.

Another variation of this structure is with else part. It has the syntax as follows:

```
If condition Then
   {statements_1}
else
   {statements_2}
End If
```

This structure will execute the statements_1 if the condition is true. In this situation statements_2 will be skipped. If the condition is false then statement_2 will be executed and statement_1 will be skipped.

If we have more than one conditions and want to implement the decision making structure then it could be implemented using the following structure:

```
If condition_1 Then
    {statements_1}
Else if condition_2 Then
    {statements_2}
Else if condition_3 Then
    {statements_3
    . . .
else
    {statements_n}
End If
```

You can predict the behavior of this code as per the discussion above.

16.12.2 Nesting If...End If statements

In programming there may be situations when more than one conditions are to be tested depending upon a previous condition. Then the combining of the conditions do not solve the problem. In such cases we have to nest the if…endif structure. Nesting means to write a structure within the other structure.

The syntax of the nested if…endif structure could be as follows:

```
If condition_1 Then
    … {statements_1}
        if condition_2 Then
            …{statements_2}
        End If
End If
```

In this situation, statements_2 will be executed only if the condition_1 and condition_2 both are true.

16.12.3 Using a Select Case block

In a simple if…else…endif structure we can only work for two alternative situations of a single condition. What if we have to check multiple options of the same condition? Then we will have to nest the if…endif structure very intelligently and with great care. In Microsoft Access, VBA provide an easy way to handle such situations. VBA has a structure known as Select…Case block/structure to test the multiple outcomes of the same condition.

Select…Case block works on the value of a control variable. The syntax for a Select…Case block will be as follows:

```
Select Case Variable
    Case Value1
        …statements_1…
    Case Value2
        …statements_2…
    …
    Case Else
        …statements_else
End Select
```

In this block all the Case and Else are part are optional. The Variable is the control variable for which different values, different statements of the code will be executed. If the one Case Value condition becomes true then the statements in that part will be executed and rest will be skipped.

16.13 Executing the Same Code Repeatedly

Often there are situations in programming when we have to repeat a certain set of statements many times. The same could happen in VBA programming in Microsoft Access. To handle such situations all programming languages provide structures called loops. By using loops we can repeat the execution of certain set of statements as many times as we require. VBA provides the following different loop structures. In all the looping structures we can use Exit loop statement to terminate a loop from within the body of the loop.

16.13.1 Using Do...Loop to create a loop

This is one of the structures in VBA which help us to repeat a set of statements. The syntax of this loop is as follows:

```
Do {{While | Until} condition}
    …statements…
Loop
```

In this syntax of the loop, the very first line checks the condition and if the condition is true the body of the loop continues to execute. Another variation of this structure exists in which the condition is tested by the last line of the structure. This variation looks like as follows:

```
Do
    …statements…
Loop {{While | Until} condition}
```

The different between these two is that in the second version of the loop the body is executed at least once, even the condition is false but in the first one the body of the loop is executed only if the condition is true.

Examples

Code of these structures with a recordset rst already created from a Students database:

```
Rst.MoveFirst
Do Until Rst.EOF()
    Debug.Print Rst.Fields("StudentName")
    Rst.MoveNext
Loop
```

This loop will be executed as follows:

1) Rst.MoveFirst statement will move the cursor to the first record in the recordset.
2) In the loop
   Until Rst.EOF() will check for the end of file condition for the recordset. EOF() will return true if the end of file has been reached and false otherwise. If the EOF() returns false then the body of the loop will be executed otherwise loop will be terminated.
3) Debug.Print Rst.Fields("StudentName") will print the value of the field "StudentName" in the Debug window.
4) Rst.MoveNext will move the cursor to the next record in the recordset.
5) Loop will transfer the control to the step 2 and the process will repeated.

We can also write the same code using the second syntax as follows:

```
Rst.MoveFirst
Do
    Debug.Print Rst.Fields("StudentName")
    Rst.MoveNext
Loop Until Rst.EOF()
```

The difference in execution is that the condition EOF() will be tested in the last statement of the Loop. Hence the body of loop will be executed at least once in this case.

16.13.2 Using While...Wend to create a loop

This is another looping structure supported by VBA in Microsoft Access. This structure also helps us to repeat a set of statements. It is very similar to the Do…Loop structure. The syntax of this loop is as follows:

```
While condition
    …statements…
Wend
```

Condition is an expression which is evaluated in True or False value. If the result of the condition is True then the body of the loop will be executed. It will continue to execute till the condition remains True and terminated otherwise.

16.13.3 Using For...Next to create a loop

The looping structures we have discussed are generally categorized as condition based loops. Sometimes we want to repeat a code for a set number of times. The value of this number is generally predetermined. For this purpose we can use For…Next looping structure provided by VBA in Microsoft Access. The syntax of this loop is as follows:

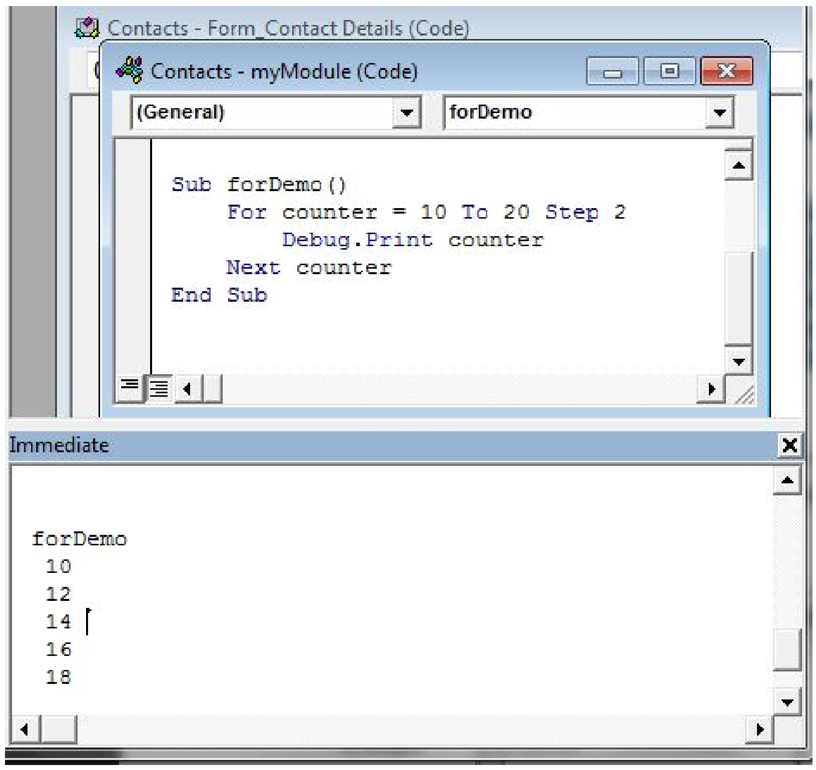
```
For counter = startValue to endValue [Step stepValue]
    …statements…
Next [counter]
```

The components of this looping structure are:

- counter : it is any variable which keep the value of repetition for the loop.
- startValue: it is the value from which the loop start to count. In other words the value of counter will be initialized to this value to start the loop.
- endValue: it is the value which will be the ending value for the counter.
- stepValue: it is an optional part. It tells the loop that with which value the value of counter will be incremented after a complete iteration of the loop body statements. If not mentioned then its value is assumed to be 1.
- …statemtents… : these are any valid statements of VBA.

The counter, startValue, endValue and stepValue are numbers.

A For…Next loop which starts from 10 and print the value of counter up to 20 with a step value 2 is shown in Figure 44 along-with its results in the immediate window.


Contacts - Form_Contact Details (Code)
Contacts - myModule (Code)
(General)
forDemo
Immediate

```
Sub forDemo()
    For counter = 10 To 20 Step 2
        Debug.Print counter
    Next counter
End Sub
```

```
forDemo
 10
 12
 14
 16
 18
```

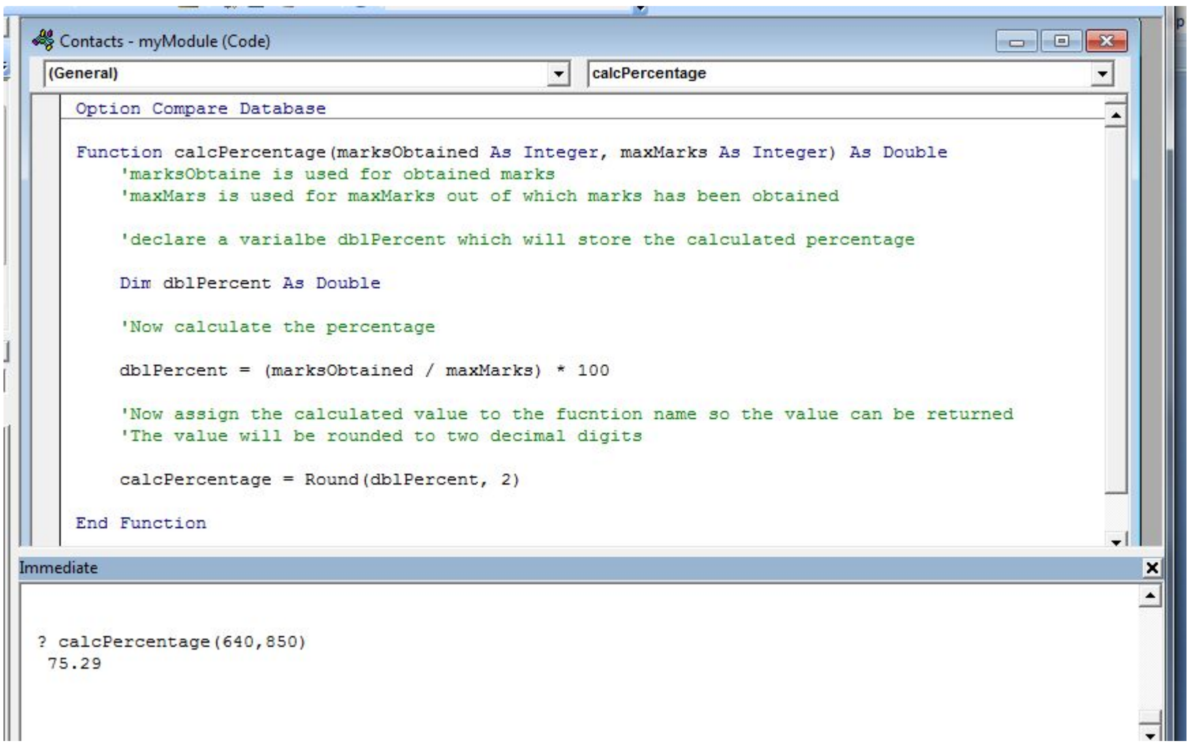
Figure 44   Loop example

16.14 Using Custom Functions

Microsoft Access provides many useful functions in it. These functions can be used in VBA programming. While programming in VBA at some point programmer will find that Microsoft Access doesn't have a built-in function to perform the required processing. What the programmer should do in such situation. VBA in Microsoft Access provides the facility to the programmers to write their own functions. These functions are called Custom Functions or User Defined Functions. Once you create a function in a module then you can use it just like a built-in function.

We can demonstrate the use of Custom functions through the example below:

We will create a function calcPercentage() and then use it. This function will take two parameters, obtainedMarks and maxMarks, and will return the percentage marks. We will use proper comments in the function for clarity and for future maintenance purpose of the code. Figure 45 shows this functions and its result in immediate window.

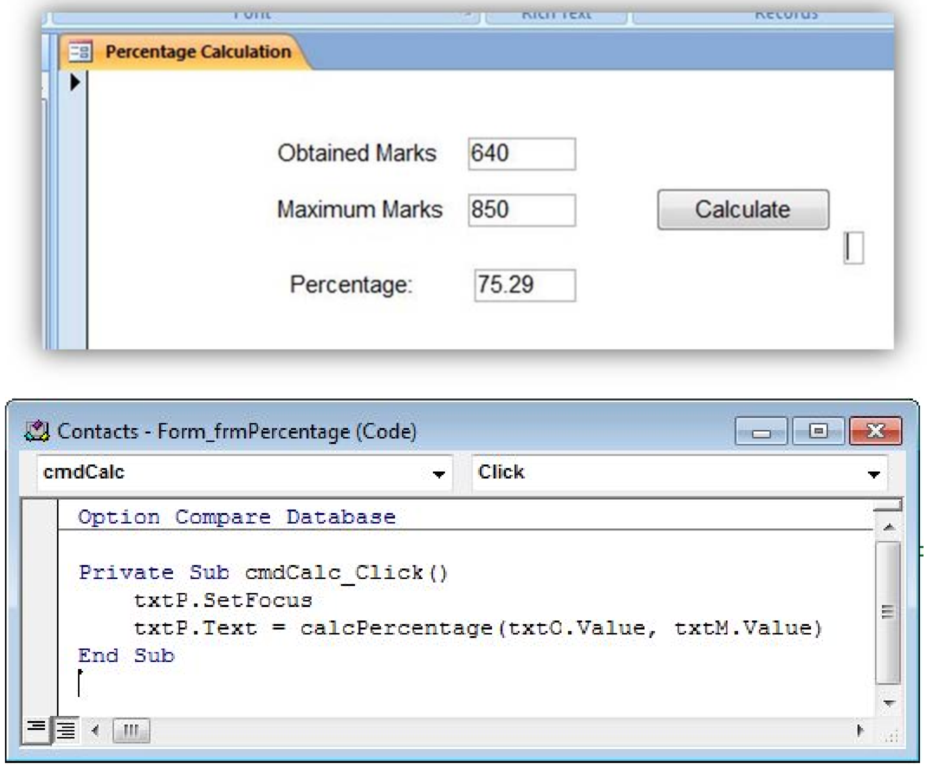


```
Option Compare Database

Function calcPercentage(marksObtained As Integer, maxMarks As Integer) As Double
    'marksObtaine is used for obtained marks
    'maxMars is used for maxMarks out of which marks has been obtained

    'declare a varialbe dblPercent which will store the calculated percentage

    Dim dblPercent As Double

    'Now calculate the percentage

    dblPercent = (marksObtained / maxMarks) * 100

    'Now assign the calculated value to the fucntion name so the value can be returned
    'The value will be rounded to two decimal digits

    calcPercentage = Round(dblPercent, 2)

End Function
```

```
? calcPercentage(640,850)
 75.29
```

Figure 45 A custom function calcPercentage() and its result

We can use this function in our Forms and reports. In Figure 46 we have used this function in the click event of *Calculate*.



Obtained Marks	640
Maximum Marks	850
Percentage:	75.29

cmdCalc — Click

```
Option Compare Database

Private Sub cmdCalc_Click()
    txtP.SetFocus
    txtP.Text = calcPercentage(txtO.Value, txtM.Value)
End Sub
```

Figure 46 Form and VBA code of click event

Activity 20

Create and test a UDF caclGrade() which takes percentage marks as parameter and return the Letter Grad (A,B,C,D,E,F) according to the general criteria being used in different educational institutes.

16.15 Changing Form Controls with VBA

When programming in VBA, the code is generally dealing with the controls on the forms. We access values or contents from the controls and set the values of some controls on the forms. We can change different properties of controls using VBA code. We use the following syntax in VBA to change the properties of controls.

ControlName.PropertyName = newValue

The form containing the control must be opened when this VBA code is executed otherwise there will an ERROR. The ControlName should be the complete name of the control. The complete name consists of the form name and control name joined by dot (.) operator. If the control is on the current form then the "Me" can be used instead of the form name. Table 17 shows some examples of setting different properties of controls on the current opened form.

Statement	Effect
Me.ControlName.Visible = False	make a control invisible
Me.ControlName.Visible = True	make the control visible
Me.ControlName.Enabled = False	disable a control
Me.ControlName.Enabled = True	enable a control
Me.ControlName.Value = newValue	change the value (contents) of a control
Me.TotalPrice.Value = Me.Quantity.Value * Me.UnitPrice.Value	set the value to a calculation based on the values of two controls on the form

Table 17 Different properties of controls

16.16 Properties, methods, and events

In Microsoft Access a database is the collection of different objects. These objects further contain different controls in it. All the controls and objects have their properties, methods and events. In our common life we also have objects and each object has its own properties, methods and event. If we consider students then each student is an object. Each student has its own properties like Name, Age, Date of Birth, Degree, etc. In Microsoft Access we have the following meanings of these.

- **Property:** it is some characteristic of the object. In Microsoft Access it may be name, size, colour, value, etc.
- **Method:** it is the action of a control which a control can perform. For example for a form Close and Open.
- **Event:** event represent any action which can happen and to which a control can response, like click, open, load, etc. Programmer can write its own code to respond to different events as per requirement.

We can use these properties, methods and events to develop different database applications for different organizations.

16.17 Looping through collections

When programming in VBA in Microsoft Access, sometimes it is the requirement to go through all the objects in a collection. This collection may be all the forms in current project, all controls on a form, all fields in a table, etc.

Microsoft Access provides a little variation in For…Next loop to through a collection of objects. This is For Each…Next loop. This structure is designed in such a way that it automatically go through all the objects in a collection. The syntax of this loop is as follows:

```
For Each element IN collection
    …statements…
Next element
```

This loop will help to process all the elements in a collection.

Task 9

Create a new database project with the following specifications:

1) A user defined function strDay() which takes a Date as argument and return the day name as a String. i.e. Monday, Tuesday,…,Sunday.
2) A user defined function numToStr() which take an amount as argument and return the amount in words.
3) A form which has three controls:
   a. one Textbox for input a Date value.
   b. one Textbox in which the day name of the date in the first Text box will be displayed.
   c. one Command button which call UDF function strDay() on the value of Textbox having date and display the result in second Textbox.
4) A form which has three controls:
   a. One Text box for input Amount.
   b. One Textbox to display the amount in the first Textbox in words.
   c. One command button which call UDF function numToStr() on the value of the first Text box and display the result in the second Textbox.