A computer program is a series of organised instructions that directs a computer to perform tasks. Without programs, computers are useless.

A program is like a recipe. It contains a list of variables (called ingredients) and a list of statements (called directions) that tell the computer what to do with the variables.

Like a recipe, a program can be written in different programming languages which may express the steps differently according to the programming language syntax, but deliver the same end result.

Programming is a creation of a set of commands or instructions which directs a computer in carrying out a task.

Later these commands or instructions will be compiled and/or interpreted and then transformed to executable instructions that a computer or electronic device can execute or run.
A programming language is a set of words, symbols and codes that enables humans to communicate with computers.

It is a language used for writing computer programs, that direct a computer to perform computation and to organise the flow of control between mechanical devices.

Hundreds of programming languages exist today. Each language has its own standard or rules for writing the commands and/or instructions.

Examples of programming languages are:
- **BASIC** (Beginner’s All Purpose Symbolic Instruction Code)
- **Pascal**
- **C**
- **Smalltalk**.

A programmer is someone who writes computer programs. One who adopts and practices a formal approach to programming is sometimes also referred to as a programmer analyst, computer scientist, software engineer and software analyst.

A programmer analyst designs computer programs besides writing them.

A computer scientist is a generic or broad term for a professional with expertise in computer software.

These different job titles are quite subjective as different companies may define them differently. Professional programmers may work in corporate IT departments, software houses and service companies.

Sometimes professional programmers work for consulting companies and their work often takes them to their client’s workplace.
LEsson 2
Generations of Programming Language

A low-level programming language is a programming language that provides little or no abstraction from computer’s microprocessor.

A high-level programming language is a programming language that is more abstract, easier to use, and more portable across platforms.

Levels of Programming Language
FIRST GENERATION OF PROGRAMMING LANGUAGE

The first generation of programming language, or 1GL, is machine language. Machine language is a set of instructions and data that a computer's central processing unit can execute directly.

Machine language statements are written in binary code, and each statement corresponds to one machine action.

SECOND GENERATION PROGRAMMING LANGUAGE

The second generation programming language, or 2GL, is assembly language. Assembly language is the human-readable notation for the machine language used to control specific computer operations.

An assembly language programmer writes instructions using symbolic instruction codes that are meaningful abbreviations or mnemonics.

An assembler is a program that translates assembly language into machine language.

Since assembly language consist of human-readable abbreviations, the assembler must first convert assembly language into machine-readable language before the computer can readily understand its instructions.
THIRD GENERATION PROGRAMMING LANGUAGE

The third generation of programming language, 3GL, or procedural language uses a series of English-like words, that are closer to human language, to write instructions.

High-level programming languages make complex programming simpler and easier to read, write and maintain. Programs written in a high-level programming language must be translated into machine language by a compiler or interpreter.

PASCAL, FORTRAN, BASIC, COBOL, C and C++ are examples of third generation programming languages.

FOURTH GENERATION PROGRAMMING LANGUAGE

The fourth generation programming language or non-procedural language, often abbreviated as 4GL, enables users to access data in a database.

A very high-level programming language is often referred to as goal-oriented programming language because it is usually limited to a very specific application and it might use syntax that is never used in other programming languages.
SQL, NOMAD and FOCUS are examples of fourth generation programming languages.

**FIFTH GENERATION PROGRAMMING LANGUAGE**

The fifth generation programming language or visual programming language, is also known as natural language.

Provides a visual or graphical interface, called a visual programming environment, for creating source codes.

Fifth generation programming allows people to interact with computers without needing any specialised knowledge.

People can talk to computers and the voice recognition systems can convert spoken sounds into written words, but these systems do not understand what they are writing; they simply take dictation.

Prolog and Mercury are the best known fifth-generation languages.
EXT : OPEN PROGRAMMING LANGUAGE

The Open Programming Language (OPL) is an embedded programming language found in portable devices that run the Symbian Operating System.

For example mobile telephones and PDAs.

OPL is an interpreted language that is analogous to BASIC.

In the early years, before the computer was invented, there are several inventions of counting machines.
Structured programming often uses a top-down design model where developers map out the overall program structure into separate subsections from top to bottom.

In the top-down design model, programs are drawn as rectangles. A top-down design means that the whole program is broken down into smaller sections that are known as modules. A program may have a module or several modules.

Structured programming is beneficial for organising and coding computer programs which employ a hierarchy of modules. This means that control is passed downwards only through the hierarchy.

Examples of structured programming languages include Ada, Pascal and Fortran.
OBJECT-ORIENTED PROGRAMMING

The object-oriented approach refers to a special type of programming approach that combines data with functions to create objects.

In an object-oriented program, the object have relationships with one another.

One of the earliest OOP languages is Smalltalk. Java, Visual Basic and C++ are examples of popular OOP languages.

DIFFERENCE BETWEEN STRUCTURED AND OBJECT ORIENTED PROGRAMMING

- Structured programming often uses a top-down design model.
- The object-oriented programming approach uses objects.
LESSON 4
TRANSLATOR

Sometimes two people cannot understand each other because they don’t speak the same language. So they need the help of a third person who understands both languages. This third person is known as a translator.

All software packages or programs are written in high-level languages, for example, C++, Visual Basic and Java.

However, in order for the computer to be able to carry out the instructions, the high-level languages must be translated into machine language before the computer can understand and execute the instructions in the program.

The translation of high level languages to machine language is performed by a translator.

PROGRAM

Have you ever wondered how your computer runs your favourite software? Your favourite software is a program that consists of several instructions that perform its operation.

A programmer will write a source code which consists of the instructions needed to run a program. Then the compiler or interpreter with assembler will translates the source code into machine language which is made of a sequence of bits (eg. 01100011).

The computer will load the machine code and run the program.
ASSEMBLER

An assembler is a computer program for translating assembly language — essentially, a mnemonic representation of machine language — into machine language.

For example in intel 80836, the assembly language for the ‘no operation’ command is NOP and its machine code representation is 10010000.

Example of assemblers are MACRO-80 Assembler and Microsoft MASM.

INTERPRETER

Interpreter is used to interpret and execute program directly from its source without compiling it first. The source code of an interpreted language is interpreted and executed in real time when the user execute it.

The interpreter will read each codes converts it to machine code and executes it line by line until the end of the program.

Examples of interpreter-based language are BASIC, Logo and Smalltalk.
The source code (in text format) will be converted into machine code which is a file consisting of binary machine code that can be executed on a computer. If the compiler encounters any errors, it records them in the program-listing file.

When a user wants to run the program, the object program is loaded into the memory of the computer and the program instructions begin executing.

A compiled code generally runs faster than programs based on interpreted language. Several programming languages like C++, Pascal and COBOL used compilers as their translators.
LESSON 5
HOW TO INSTALL VISUAL BASIC 6.0

VISUAL BASIC 6.0 INSTALLATION

Microsoft Visual Basic 6.0 was designed to be easy to learn and use. The language not only allows programmers to easily create simple GUI applications, but also has the flexibility to develop fairly complex applications as well.

Programming in Microsoft Visual Basic 6.0 is a combination of visually arranging components or controls on a form. Hence a simple program can be created without the programmer having to write many lines of code.

VISUAL BASIC 6.0 FEATURES

Microsoft Visual Basic is fast and easy with intuitive tools that enable you to rapidly build your own Windows applications. You can use Microsoft Visual Basic to develop programs such as games, calculator, phonebook database and lots more.

The word “Visual” refers to the technique used to build the Graphical User Interface (GUI). Visual Basic features an easy “drag and drop” method to produce an interactive Graphical User Interface (GUI) for your program.

The “Basic” part refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a high level programming language develop by Microsoft.
VISUAL BASIC 6.0 FEATURES

Among the main features of Visual Basic are drag and drop user interface, data access features, ActiveX technologies and internet capabilities. Drag and drop user interface allows you to instantly create an interactive user interface for your software without dozens of codes.

Data access features allow your application to access information from a database such as a telephone book program.

ActiveX technologies allow you to make use of the functionality provided by other applications.

Internet capabilities allow your program to interactively utilise the Internet.
LESSON 6
BASIC ELEMENT IN PROGRAMMING

5 Basic elements in programming.

1. Constant.
2. Variable.
3. Data Type.
4. Operators.
5. Control Structures.

CONSTANTS AND VARIABLES

Constants
Constant is a data container that stores information. The value will never change (remains constant) at any time during the course of a program.

Declare is the official term used in programming to announce to the program the condition of statement in programming.

Variables
Variable is a data container that stores information. The value inside may change at any time during the course of a program.
DATA TYPES, OPERATOR AND CONTROL STRUCTURES

Data Types

String
Example: "This is String:"

Integer
Example: 1, 20, 1050

Floating point
Example: 1.25, 3.141

Operator is a symbol that tells what action to perform.

<table>
<thead>
<tr>
<th>Operators symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ (plus)</td>
<td>marks = 20 + 50</td>
</tr>
<tr>
<td>- (minus)</td>
<td>marks = 31 - 12</td>
</tr>
<tr>
<td>/ (divided)</td>
<td>marks = 100 / 2</td>
</tr>
<tr>
<td>* (multiplied)</td>
<td>marks = 50 * 2</td>
</tr>
<tr>
<td>&gt; (greater than)</td>
<td>marks &gt; 1</td>
</tr>
<tr>
<td>&lt; (less than)</td>
<td>marks &lt; 100</td>
</tr>
<tr>
<td>= (equal to)</td>
<td>marks == 100</td>
</tr>
<tr>
<td>&lt;&gt; (not equal to)</td>
<td>marks &lt;&gt; 100</td>
</tr>
</tbody>
</table>

Control structures allow the programmer to control the flow of a program.

if marks>50 then
{status=pass}
else
{status=fail}
LESSON 7
THE DIFFERENCES BETWEEN ETHICS AND LAW

CONSTANTS AND VARIABLES
Constant is a virtual data container that stores information. The value will never change (remains constant) at any time during the course of a program.

Variables is a virtual data container that stores information. The value inside may change at any time during the course of a program.

DIFFERENCES BETWEEN CONSTANTS AND VARIABLES
### Constants versus Variables

**Usage**

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Constants</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value is not changeable during the course of the program.</td>
<td>Value can be changed anytime during the course of the program.</td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>Use constant when you want to declare something that won’t change midway in your program execution.</td>
<td>Use variable to store data that may or will change during the running of the program.</td>
</tr>
</tbody>
</table>
DATA EXAMPLES FOR DIFFERENT DATA TYPES

Data type determines the type of data a variable can store, for example a number or a character. Examples of data types are integer, double, string and boolean.

<table>
<thead>
<tr>
<th>Data Types</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>18, 79, 21</td>
</tr>
<tr>
<td>Double</td>
<td>41.5, 31.4</td>
</tr>
<tr>
<td>String</td>
<td>Kok Keong, Amira, Eden, Alex</td>
</tr>
<tr>
<td>Boolean</td>
<td>TRUE, FALSE</td>
</tr>
</tbody>
</table>

DATA EXAMPLES FOR DIFFERENT DATA TYPES

**Integer**

Integer data type contains any whole number value that does not have any fractional part.

```vbnet
Const Year_Birth = 1990
```

This is how we declare an integer type constant in Visual Basic statement.

```vbnet
Dim Age As Integer
Age = 17
```

This is how we declare an integer type variable in Visual Basic statement.

**Double**

Any number value that may and could contain a fractional part.

```vbnet
Const PI = 3.142
```

This is how we declare a double type constant in Visual Basic statement.

```vbnet
Dim Marks As Double
Marks = 60.5
```

This is how we declare a double type variable in Visual Basic statement.
**String**
Any value that contains a sequence of characters.

```
Const Name = "AHMAD"
```

This is how we declare string type constant in Visual Basic statement.

```
Dim Address As String
Address = "Kuala Lumpur"
```

This is how we declare string type variable in Visual Basic statement.

**Boolean**
Boolean type consists either a True or False value. Programmers usually use it to store status.

```
Const Input_Status = true
```

This is how we declare a boolean type constant in Visual Basic statement.

```
Dim Input_Status As Boolean
Input_Status = true
```

This is how we declare a boolean type variable in Visual Basic statement.
### OTHER DATA TYPES

<table>
<thead>
<tr>
<th>Data type</th>
<th>Size</th>
<th>Sample usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>2 bytes</td>
<td>Dim Discount As Integer</td>
</tr>
<tr>
<td>Long integer</td>
<td>4 bytes</td>
<td>Dim Loan As Long</td>
</tr>
<tr>
<td>Single-precision floating point</td>
<td>4 bytes</td>
<td>Dim Price As Single</td>
</tr>
<tr>
<td>Double-precision floating point</td>
<td>8 bytes</td>
<td>Dim Pi As Double</td>
</tr>
<tr>
<td>Currency</td>
<td>8 bytes</td>
<td>Dim Debt As Currency</td>
</tr>
<tr>
<td>String</td>
<td>1 byte per character</td>
<td>Dim Input As String</td>
</tr>
<tr>
<td>Boolean</td>
<td>2 bytes</td>
<td>Dim Flag As Boolean</td>
</tr>
</tbody>
</table>
LESSON 9
MATHEMATICAL AND LOGICAL OPERATORS

MATHEMATICAL, RELATIONAL AND LOGICAL OPERATORS
Operator is a symbol or notation that tells a computer to perform certain actions or operations.

An example: the plus (+) notation will tell the computer to perform the "add" operation.

Let's look at some examples of Mathematical Operators.
Let's look at some examples of Relational Operators.
Let's look at some examples of Logical Operators.

**FUNCTION OF MATHEMATICAL OPERATORS**

Mathematical operators are notations that tell the computer to perform mathematical operations.

**FUNCTIONS OF RELATIONAL OPERATORS**

Relational operators perform comparison between two elements.

They return an element of logical 1 (True) where the relation is true, and element of logical 0 (False) where the relation is false.

The diagram shows some common relational operators and their expression.
Let’s understand a relational operation with the following examples.

In this expression, A >= B is used to test if the value of left expression (A) is greater than or equal to that of the right expression (B).

If the conditions are met, then the program will return a logical 1 and proceed to print "A is greater than or equal to B" else it will return a logical 0 and proceed to print "A is less than B".

**FUNCTIONS OF LOGICAL OPERATORS**

Logical operators are notations that tell the computer to perform logical operations.

Examples of Logical operation are: AND, OR, and NOT.

Logical operator compares 2 conditions and returns a TRUE or FALSE value.
**AND operator**

The diagram shows a truth table of AND operator.

Notice that truth value of $X \text{ AND } Y$ is True (1) if only both $X$ and $Y$ are True (1).

Else it is false (0)

**OR operator**

The diagram shows a truth table of OR operator.

Notice that truth value of $X \text{ OR } Y$ is only True (1) if either $X$ or $Y$ are True (1) or both $X$ and $Y$ are true (1)

Else it is false (0)

**NOT operator**

The diagram shows the truth table of NOT operator $\neg X$ is the negation of $X$, it is essentially the 1's complement operation.

Notice that truth value of $\neg X$ is True (1) when $X$ is False and vice versa.
DIFFERENCES IN MATHEMATICAL AND LOGICAL OPERATORS
As we have learned, there are some differences between mathematical, relational and logical operators.

**Function:**

\[
20 + 15 = 35 \\
20 - 15 = 5 \\
20 \times 15 = 300 \\
20 \div 15 = 1.33
\]

Mathematical operators perform mathematical operations such as plus or subtract.

\[
A = B \\
C > D \\
E < F \\
G \geq H \\
I \leq J
\]

Relational operators perform element-by-element comparisons between two arrays.

\[
A \text{ AND } B \\
C \text{ OR } D \\
\text{NOT } E
\]

Logical operators perform logical operations such as checking the condition of two Boolean values.
Symbols:

These operators have their own symbols based on the programming language.

**EQUALITY OPERATORS**

The equality determine an operand equal to or not equal to another operand.

Remember we must use "=" and not "<>" when testing if two primitive values are equal in Microsoft Visual Basic 6.0.
Pseudo code is text only sentences that describe the logic and program flow of a computer program. Pseudo code resembles plain English.

It usually does not have any specific programming language syntax and grammar.

Pseudo code is directly linked to the computer codes because each pseudo code statement can often be converted into the programming language virtually line by line. There are no set rules for writing pseudo code.

A programmer can have his or her personalised pseudo code.

He or she must use consistent language and syntax in the pseudo code, so that he or she can understand it at a later stage.

AN EXAMPLE PROGRAM BASED ON A PSEUDO CODE

This program will basically calculate the Volume of a sphere based on the given value \( r \).

BEGIN
Declare PI as a constant
Declare "r" and "volume"
Variable "r" equal to 10
Execute sphere's volume formula
Print result in message box (MsgBox)
END

This is the output
EXT: STANDARDS OF GOOD PSEUDO CODE

Pseudo-Code is simply a numbered list of instructions to perform some task. In this course we will enforce three standards for good pseudo code

Number each instruction. This is to enforce the notion of an ordered sequence of the operations.

Furthermore we introduce a dot notation (e.g. 3.1 come after 3 but before 4) to number subordinate operations for conditional and iterative operations

Each instruction should be unambiguous and effectively computable.

Completeness. Nothing is left out.

Pseudo-code is best understood by looking at examples.

Each example below demonstrates one of the control structures used in algorithms: sequential operations, conditional operations, and iterative operations.

We also list all variables used at the end of the pseudo-code.
Example:

Computing discount:

Pseudo-code:
Computing the final price of an item after figuring in discount.

Note the three types of instructions:
input (request),
process/calculate (=) and
output (print)

BEGIN
1. Request price of item
2. Request discount rate
3. discount = price of item times discount rate
4. final price = price of item subtract discount
5. Print final price
END
variables: price of item, discount rate, final price
LESSON 11
FLOW CHART

Let’s identify some of the main elements in the flow chart. We have five main elements in a flow chart.

**Terminator** shows the beginning or end of a program.

**Flowline and arrowhead** use to connect symbols and indicate the sequences of operation.

**Input or output** shows either an input operation (e.g. an INPUT from the user) or an output operation (e.g. PRINT some messages).

**Process** shows a process to be carried out (e.g. calculation).

**Decision** shows a decision (or choice) to be made. The program should continue along one of two routes (e.g. if...else).

A flow chart is a diagram using symbols to show the step-by-step sequence of procedures in a program. A flow chart describes the logic and program flow of a computer program graphically.

**Example:**

A student requests his marks from the program.

The program will then check whether his marks is more/equal to 50 or not.

If the marks are over or equal to 50, the program will print a congratulatory message together with the marks.

If the marks are lower than 50 then the program will print a motivational message together with the result.
LESSON 12
CONTROL STRUCTURES

Control structure is a structure of statements in programming that allows the programmer to control the flow of a program.

Control structure can be divided into sequence, selection and repetition control structures.

SEQUENCE CONTROL

Sequence control refers to the linear execution of codes within a program. In sequence control, the statements are executed one by one in consecutive order.

In sequence control, the statements are executed one by one in consecutive order.
Let's see an example of pseudo code that has sequence control structure.

This program will request the user's date of birth and then request today’s date, calculate the age and finally will print the user's age.

For example, today’s date is 1-JAN-2006, then the results will be shown as follows:

<table>
<thead>
<tr>
<th>Student's name</th>
<th>Student's date of birth</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hassan b. Zambiri</td>
<td>1-3-1988</td>
<td>17</td>
</tr>
<tr>
<td>Rosita bt. Kamaruddin</td>
<td>22-6-1986</td>
<td>19</td>
</tr>
<tr>
<td>Sirajuddin b. Najib</td>
<td>14-4-1987</td>
<td>18</td>
</tr>
</tbody>
</table>

Let's see another example of pseudo code that has sequence control structure.

The following is the result of the example.

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Salary (RM)</th>
<th>Allowance (RM)</th>
<th>Total (RM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32156</td>
<td>2500</td>
<td>100</td>
<td>2600</td>
</tr>
<tr>
<td>41324</td>
<td>3200</td>
<td>150</td>
<td>3350</td>
</tr>
<tr>
<td>32667</td>
<td>2500</td>
<td>180</td>
<td>2680</td>
</tr>
</tbody>
</table>

Let's see the flow chart for a general sequence control structure.

As we can see, the flow chart for sequence control is very simple.

It will execute statement 1 followed by statement 2 and any following statements.
EXAMPLE

The flow chart represents a program that will request the user’s date of birth and then request today’s date, calculate the age and finally will display the user’s age.

SELECTION CONTROL

There are times when you want your program to make a decision based on the situation given.

For example, a program that stores student’s marks may respond differently to different marks.

Or maybe a simple mathematical program will display its result as odd or even, based on the result.

Selection control enables the programmer to assign different events for different situations.

An example of selection control is “If...Then...Else” statement. The basic pseudo code for “If...Then...Else” statement is as follows.

Let’s see a pseudo code example for the “If...Then...Else” statement for a program that will print “You are too heavy to ride the toy car” if the student’s weight is more than or equal to 50.
Otherwise it will print “You can ride the toy car”

Let’s see the flow chart for a similar selection control example.

Let’s see a program that will respond differently for different service hours in a school library.

A student requests her service hours in a library from the program, the program will then check whether her service hours are more than/equal to 30 or not.

If the service hours are over or equal to 30, the program will print a message, “Thank you for your service”.

If the service hours are lower than 30 then the program will print a message, “Please continue to serve in the library”.

Let’s write an example of a program that implements sequence control structure.

This program will declare a constant pi equal to 3.142. Then the program declares two variables to be used in the program (r and volume).

The program will then assign some value to the r variable.

The program will then calculate the volume using the formula volume = (4/3)πr

The program will then display the volume calculated with a message box.
Now write an example program that implements selection control structure.

Write a program that will retrieve the current date from the system.

If today’s date is more than 15 then the program will display a message box with the message “We are towards the end of the month”.

Else the program will print “We are at the beginning of the month”.

DIFFERENTIATE BETWEEN SELECTION CONTROL AND SEQUENCE CONTROL

---

**Execution flow**

<table>
<thead>
<tr>
<th>sequence control</th>
<th>selection control</th>
</tr>
</thead>
<tbody>
<tr>
<td>executes one by one in linear or consecutive order</td>
<td>execute different statements for different conditions</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>sequence control</th>
<th>selection control</th>
</tr>
</thead>
<tbody>
<tr>
<td>use when want to execute code line by line</td>
<td>use when want to implement decision making process in the program</td>
</tr>
<tr>
<td>does not use the decision symbol</td>
<td>use the decision symbol</td>
</tr>
</tbody>
</table>

**Flow Chart**

![Flow Chart Diagram](image-url)
EXT : REPETITION CONTROL STRUCTURES

Let’s learn about the last control structure which is repetition control structure. A repetition control structure allows the programmer to specify an action to be repeated while some condition remains true.

Pseudo Code:

```
BEGIN
  While there are more items on my shopping list
    Purchase next item and cross it off my list
  END
```

This is a pseudo code of "While" structure.

```
Dim product As Integer
product = 3
While product <= 3000
  product = product*3
END While
```

The value of product will increase in this order:
3, 9, 27, 81, 243, 729, 2187, 6561

This program segment will continue its looping until the condition of variable named product is greater than 3000.

This is a flow chart of a repetition control structure.
LESSON 13
OVERVIEW OF PROGRAM DEVELOPMENT PHASES

In program development, there are five main phases. These phases are a series of steps that programmers undertake to build computer programs. The program development phases guide computer programmers through the development of a program.

The five main phases of program development are as follows:
PROBLEM ANALYSIS PHASE
During the problem analysis phase, the programmer will interview the client to find out what the client’s needs are.

For example, the client might be a school that wishes to set up a school registration program. So the school administrator might tell the programmer that they need to record students’ data such as name, date of birth, gender, class, parents’ names, address and contact numbers.

PROGRAM DESIGN PHASE
Based on that, the programmer will design a flow chart that represents the needs of the client, which in this case is the school registration program.

CODING PHASE
Once the flow chart is confirmed, the programmer will perform coding.

TESTING AND DEBUGGING PHASE
The school registration program will be tested by the users at the client’s site. In this case, it will be the school office administrators. If there are any errors, the programmer will do a debugging of the program.

DOCUMENTATION PHASE
After this, the programmer will complete the documentation for the program; this includes the user manual, a clear layout of the input and output records and a program listing.

EXT: SIX STEPS OF SOFTWARE DEVELOPMENT

Step 1
Know your target user.

The software we develop is for our user. Learn their preferences.

Step 2
Search for the exciting possibilities in your software.

In this way, it is possible to identify their likely concerns and anxieties and thus take them into account during the design stage.
**Step 3**
Correlate these possibilities with the target user. Find how to combine these possibilities with user personalities.

**Step 4**
Immerse yourself in the process. You have to be involved deeply in the subject that you develop.

**Step 5**
Engage a talented and visionary designer or developer in the team. They are responsible for all aspects of the software.

**Step 6**
Make quality and fantastic features as the priorities among software development team members. The software should deliver the complete, accurate, reliable and available information to the right person at the right time in a suitable format.
LESSON 14
PROBLEM ANALYSIS

PROGRAM DESIGN

The first step in the program development phase is to analyse the problems faced by target users. In this phase, a programmer reviews and defines the problems. Identifies the data input, process and output for the program.

CASE STUDY

The school needs a school registration program to computerise the registration of new students.

Any new student who enrolls in the school should have all his or her data keyed into the school registration program. They will notified of his or her class for the coming academic year or term.

This case study will be used throughout the program development phases.

In this case study, the programmer reviews and defines the problems faced by the school administrators.

Increase of students make the registration more and more difficult.

Registration must be made by using the computer.

The programmer must indicate the purpose of the program before continuing to the next task.

Purpose of the school registration program is to computerise the registration of new students.

In this case, the purpose of the school registration program is to computerise the registration of new students.
In this situation, the programmer identifies the data input, process and output for the program.

First step: The programmer identifies the data input, for example student’s name, student’s NRIC, gender, telephone number, date of birth, parents’ name and address.

Second step: The programmer identifies the process for the program. Any new student who enrolls in the school should have all his or her data keyed into the school’s registration program.

PROBLEMS AND SOLUTIONS

In program development, the target user and programmers will face some problems. Here are some common problems faced by them, and solutions to overcome them.
LESSON 15
PROGRAM DESIGN

Many inventions today are driven by computer programs. To develop good programs, program design is a very important phase.

PROGRAM DESIGN PHASE

In program design, there are three popular tools used, namely the top-down design model, pseudo code and flow chart.

In this case study, the programmer generates a top-down design model.

This top-down model consists of three modules, namely input module, process module and output module.

Then the programmer writes the pseudo code for the program based on the top-down design model.

Case Study

The school needs a school registration program to computerise the registration of new students.

Any new student who enrolls in the school should have all his or her data keyed into the school registration program. They will notified of his or her class for the coming academic year or term.

This case study will be used throughout the program development phases.
Next, the programmer draws the flow chart that shows the data flow of the program.

Besides flow chart, a programmer also produces input and output user interfaces base on the existing form.

EXT : UNIFIED MODELING LANGUAGE

Unified Modeling Language (UML) has been adopted as a standard notation for object modeling and development.

These notations are used in diagrams that present various views of the program being developed.

These are some symbols used in Unified Modeling Language Diagrams.
The process of writing instructions or code for computers is known as coding. Hence without coding, there would be no software to help us perform useful tasks on computers.

**CODING**

In this phase, a programmer uses a program development tool which generates or provides some or all codes.

Coding is the process of writing the solution using the computer programming language. After finishing the coding process, the programmer will type the programming language code into the computer.
PROGRAMMING LANGUAGE

There are many different programming languages in the market. Examples of the popular programming languages used in Malaysian secondary schools are Visual Basic and C language.

Each of these programming languages has its own particular syntax.

CASE STUDY

Student Registration
Name: __________________________
NRIC: _________________________
Gender: ________________________
Tel: ___________________________
Date of Birth: _________________
Parent’s Name: _________________________
Address: ___________________________
Student’s Class: _________________________

Private Sub Command1_Click()
Dim Name, NRIC, Gender, Tel, DOB, PN, Ad As String
Dim Class As String
Name = Text1.Text
NRIC = Text2.Text
Gender = Text3.Text
Tel = Text4.Text
DOB = Text5.Text
PN = Text6.Text
Ad = Text7.Text
If Name <> "" And NRIC <> "" And 
Gender <> "" And Tel <> "" And
DOB <> "" And PN <> "" And Ad <> "" Then
Class = "6 Sapphire"
Label10.Caption = Class
Else
MsgBox("Data error!! Please try again")
End If
End Sub
EXT : EXTREME PROGRAMMING

Extreme programming is a strategy that proposes that a programmer should immediately begin coding and testing solution as soon as requirements are defined. The code is continually tested and refined until it works.

Extreme programming also suggests that programmers work in pairs.

They communicate with each other to generate ideas for proper solutions.

IT experts claim that extreme programming reduces the time required for the program development phases.

Extreme programming most often is used in object-oriented design.
LESSON 17
TESTING AND DEBUGGING

For computers to perform their tasks smoothly, the software or program must be error-free.

Imagine doing an important assignment on the computer. Suddenly, there is a program error and you are unable to complete your work and meet the deadline.

That is why before any software is released to the market or given to the users, it must undergo testing and debugging. Program testing is done right after coding. The purpose of program testing is to ensure the program runs correctly without any syntax or logic errors.

If errors are uncovered during testing, the programmer will proceed to debugging. Debugging involves tracing the source of errors in the program and making the necessary corrections.

After correcting the errors, it is normal for programmers to test run the program again just to make sure that the errors have been properly fixed.

TESTING AND DEBUGGING

Once the programmer has finished the coding, the next step is to test it.

The purpose of program testing is to ensure the program runs correctly and is error-free.

There are three types of errors uncovered during this phase: syntax errors, logic errors and run-time errors.
SYNTAX ERROR

When the code violates the syntax of the programming language, a syntax error occurs.

A syntax error is caused by a wrong spelling in command and declaration. Other syntax errors include case sensitive, punctuation and wrong word command.

They either show up as you type the program code, or if you have turned off the automatic syntax error check, the syntax errors show up when you try to run or compile the program.

When syntax error is located, a message is displayed on the screen.

The programmer must review and correct all syntax errors.
LOGIC ERROR

If the expected output and actual output do not match for any set of data, the program has a logic error.

A logic error is an error in the design that causes inaccurate outputs.

We can use desk check technique to test logic errors. There are 3 steps to perform desk checking.

First, create a set of test data that includes input and output data.

Second, use the test data to test the coding and record the result.

Lastly, compare the manual results with the computer generated results.
RUN-TIME ERROR

Run-time error is an error that occurs while the program is running or executing.

A run-time error may cause the program to stop running.

DEBUGGING

The process of locating and correcting of syntax and logic errors in a program is known as debugging the program.

The program errors themselves are called bugs. Thus removing the errors are called debugging.
EXT : BETA

Some software companies distribute a beta version of their software to users. A beta version is a program that has most or all of its features and functionality implemented.

Users test the beta program and send in bug reports to the software company.

This enables the software manufacturer to fix any errors before the software is released to the public for sale.
In program development, documentation refers to the written material generated throughout all the phases of program development.

Thus documentation includes the detailed problem definition, the program plan (flow chart or pseudo code), comments within the source program and testing procedures.

It also includes a detailed description of the program, clear layouts of input and output records and a program listing.

All documentation must be ready by the end of program development.

The programmer should ensure that all documentation is complete and accurate.

Documentation is very valuable if the program requires changes in the future as documentation enables new programmers to learn about existing programs much easily and quickly.

**DOCUMENTATION**

Documentation refers to the written description and pseudo code of computer programs.
Although documentation is stated as the last phase in program development, it is actually performed throughout all the five main phases of program development.

Documentation consists of materials generated during each phase.

Thus the documentation package is made up of the detailed problem definition, the program plan (flow chart or pseudo code), comments within the source program and testing procedures.

Other items in the documentation package also include a detailed description of the program, clear layouts of input and output records, and a program listing.

The programmer should ensure that all documentation is complete and accurate.

Documentation becomes especially valuable when the program requires changes in the future.

Proper documentation greatly reduces the amount of time a new programmer spends in learning about the existing programs.

**INTERNAL DOCUMENTATION**

Internal documentation consists of comments within the program. These two types of comments are global comments and internal comments.
Global comments, usually stated at the top of the program, explain the program’s purpose and identify the program’s name, its author and the date the program was written.

Internal comments, which appear throughout the body of the program, explain the purpose of the code statements within the program.

Internal comments are mostly addressed to other programmers who may have to make corrections or other modifications in programming language.

**EXT : MARKETING DOCUMENTATION**

It is important to provide some informative materials that enable people to learn more about the software product.

This form of documentation, called marketing documentation, has three purposes.

1. To interest potential users with the product and encourage them to get more involved.

2. To inform potential users about the product so that they know what to expect in return.

3. To compare the product with other alternatives.
LESSON 19
COMPUTER USERS

DEVELOP A PROGRAM

A project is a set of activities with a fixed start date and end date. Program development is also a type of project.

A successful program development project will result in one or more programs that are error-free, affordable, relevant to the needs of the users and delivered on time.

All the software that you find so helpful, educational and enjoyable today is the outcome of successful program development projects.

For a program development project to succeed, the people involved must possess the appropriate set of communication, management and technical skills.

APPLY PROGRAM DEVELOPMENT PHASES TO SOLVE PROBLEMS

Let’s see how a programmer applies program development phases to solve the library problems.

**Problem Analysis Phase**

identifies the data input, processing and output for the program.
Program Design Phase

Top-down Design Model

MAIN

Input

Process

Output

Check any field empty and data input

Request book_title
Request author
Request publisher

Print book registered

Pseudo code

BEGIN
Request book_title, author, publisher from user
Check data input
If any field empty Then
    print "Please fill up all the data!"
Else
    print "Book registered!"
End If
END

Flow Chart

BEGIN
Declare book_title, author, publisher as variable
Request book_title, author, publisher from user
Check data input

Any field empty?

True

print "Please fill up all the data!"

False

print "Book registered!"

END
5.0 PROGRAMMING

Coding Phase

Existing Form

Sample
Book enquiry form

<table>
<thead>
<tr>
<th>Book Title</th>
<th>Author</th>
<th>Publisher</th>
<th>Status</th>
</tr>
</thead>
</table>

Library System

Input Interface

<table>
<thead>
<tr>
<th>Title</th>
<th>Text Box String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Text Box String</td>
</tr>
<tr>
<td>Publisher</td>
<td>Text Box String</td>
</tr>
</tbody>
</table>

Output Interface

<table>
<thead>
<tr>
<th>Status</th>
<th>Label String</th>
</tr>
</thead>
</table>

Program Interface

Private Sub Command1_Click()
    Dim book_title As String
    Dim author As String
    Dim publisher As Integer
    Dim status As String
    book_title = Text1.Text
    author = Text2.Text
    publisher = Text3.Text
    If book_title = Empty And author = Empty Or publisher = Empty Then
        MsgBox("Please fill up all the detail!")
    Else
        Label5.Caption = "Book registered!"
    End If
End Sub
Testing and Debugging Phase

Documentation Phase
EXT : TIPS AND TECHNIQUES

Let’s learn a few tips and techniques on developing a new program using Microsoft Visual Basic 6.0.

- In Microsoft Visual Basic 6.0, undeclared variable will be assigned as variant type. Variant type is slow, consuming more memory than the others.

- Every procedure and module should have comments explaining their function. It will be easier for you to maintain the code later.

- Use descriptive words for your variables and control. It will be easier for other people to understand the program if you use this technique.

- When using graphic, use *.gif, *.jpg, and *.wmf picture formats instead of *.bmp. Bitmaps format consume more memory and may slow your program down.

- If you have some code which is repeatedly used, code it as independent module or function so that you can easily reuse them later.
LESSON 20
OVERVIEW OF THE LATEST TYPE OF PROGRAMMING LANGUAGES

THE LATEST TYPE OF PROGRAMMING LANGUAGES

FIFTH GENERATION LANGUAGES

Fifth generation programming language (5GL) is an advance programming language which concentrates on solving problems using constraints given to the program.

In fifth generation language, the programmer just need to define the problem to be solve and the program will automatically code the program based on the problem definition.

Fifth generation languages are designed to make the computer solve the problem for you.

Fifth generation languages are mostly used in artificial intelligence research.

Examples of fifth generation languages include Prolog and Mercury.

NATURAL LANGUAGE

Natural Language programming aims to use natural language such as English to write a program.

Instead of using a specific programming language syntax, natural language
programming will use normal English as the input to program software.

Such a technique would mean less technical programming knowledge is required to write a program.

The programmer needs to define the program using normal language.

**OPENGL (GRAPHIC LIBRARY)**

OpenGL (Graphics Library) is a standard specification to describe the standard Application Programming Interface (API) for 3D/2D computer graphic applications.

OpenGL specification describes a set of functions and the exact behaviours that the 3D/2D application must perform.

OpenGL was developed by Silicon Graphics.

OpenGL is widely used in virtual reality, scientific visualisation, flight simulation and video game development.
EXT: THE LATEST PROGRAMMING LANGUAGES: CAPABILITIES & DIFFERENCES

What can the latest programming languages do? How are they different from the previous or past programming languages?

<table>
<thead>
<tr>
<th>Latest Programming Languages</th>
<th>Previous Programming Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• simplicity: aims to make programming simpler and more accessible to the public</td>
<td>• would require a person to understand the computer system architecture and programming language syntax</td>
</tr>
<tr>
<td>• less work: aims to move the hard work from the human to the system</td>
<td>• the computer that has to work hard computing the program code</td>
</tr>
<tr>
<td>• cross platform: it also eliminates the different hardware factor, the same code or input can be used in different target platform</td>
<td>• is tightly related with the system’s hardware and operating system</td>
</tr>
</tbody>
</table>