UNIT 2
PROGRAMMING METHODOLOGY

Stylistic Guidelines:

Writing good program is a skill. This can be developed by using the following guidelines.

1. **Meaningful Names for identifiers:** A programmer should give the meaningful names to each section of the program so that it can help him to identify the variable used for specific purpose. This helps him to execute the right elements during the complex run of a program.
2. **Ensure clarity of expression**: Expression carry out the specified action. Thus they must be clearly understood by the users. There should not be any compromise with the clarity of expression.
3. **Use of comments and indentations**: Comments generally are used as the internal documentation of a program .if comments are used in the program they will guide the program while debugging and checking. While indentation is the proper way of writing to avoid the confusion regarding the flow of program. These highlights nesting of groups of control statements.
4. **Insert blank lines and blank spaces**: Blank lines should be used to separate long, logically related blocks of code. Specifically Normally in programming the standard for the use of spaces is to follow normal English rules. This means that: Most basic symbols in C++ (e.g., "=", "+", etc.) should have at least one space before and one space after them.
5. **Statements**: Each statement should appear on a separate line. The opening brace following a control statement such as if or while should appear on the line after the if or while, lined up with the left of the control statement, and the closing brace should appear on its own line, lined up with the left of the control statement. The opening and closing braces for a function should be lined up in the same way. The statements within a {_____} pair are indented relative to the braces.

Characteristics of a Good Program:

Following are the characteristics of a good program.

1. **Effective and efficient:** The program produces correct results and is faster, taking into account the memory constraints.
2. **User friendly:** The program should be user friendly. The user should not be confused during the program execution . The user should get correct direction and alerts when he is going through the program.
3. **Self documenting code:** A good program must have self documenting code. This code will help the programmer to identify the part of the source code and clarify their meaning in the program.
4. **Reliable:** The good program should be able to cope up from any unexpected situations like

wrong data or no data.
5. **Portable:** The program should be able to run on any platform, this property eases the use of program in different situations.
6. **Robustness**: Robustness is the ability of the program to bounce back an error a and to continue operating within its environment.

PROBLEM SOLVING METHODOLOGY AND TECHNIQUES

To develop an efficient and effective programs we should adopt a proper problem solving methodology and use appropriate techniques. Following are some of the methods and techniques to develop a good program.
1. **Understand the problem well**: for a good program one should understand the problem well . one should know what exactly is expected from the problem. Knowing the problem well is the half way done.
2. **Analyze the program. :** Analyzing the problem involves identifying the program specification and defining each program's minimum number of inputs required for output and processing components.
3. **Design** : In this phase of design a Model is developed which look alike a program . This phase gives the face to the program. Outputs are designed in this phase.
4. **Code program :**This step is the actual implementation of the program. In this program algorithm is translated into programming language. in this it is essential to decide which technique or logical will be more appropriate for coding.
5. **Test and Debug program.:** Once the solution algorithm is coded the next step is to test and debug the program. Testing is the process of finding errors in a program and debugging is of correcting the errors. The developed program is put to test in different conditions and verified at different level for its proper and efficient working.
6. **Implementation & Documentation:** After successful execution of the program it is implemented. Documentation refers to written descriptions specification, design code and comments, internal and external to program which makes more readable and understandable.

Uses of documentation:
1. This becomes an useful interface between a technical personnel and non technical personnel.
2. This is very useful for upkeep and maintenance.
3. Documentation makes ease for any technical emergencies.
4. Very useful in operating for learners and trainers.

Type of errors:

There are three types of errors generally occur during compilation and running a program. They are (i) Syntax error; (ii) Logical error; and (iii) Runtime error.

Syntax error: Every programming language has its own rules and regulations (syntax). If we overcome the particular language rules and regulations, the syntax error will appear (i.e. an error of language resulting from code that does not conform to the syntax of the programming language). It can be recognized during compilation time.

Example

```
int a = 0;
while (a < 10)
{
a = a + 1
cout<<a;
}
```

In the above statement, the fourth line is not correct. Since the statement does not end with **;**. This will flash a syntax error.

Logical error: Programmer makes errors while writing program that is called logical error. It is an error in a program's source code that results in incorrect or unexpected result. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running. The logical error might only be noticed during runtime, because it is often hidden in the source code and are typically harder to find and debug.

```
int a = 100;
while (a < 10)
{
a = a + 1;
cout<< a;
}
```

In the above example, the while loop will not execute even a single time, because the initial value of a is 100.

Runtime error: A runtime error is an error that causes abnormal termination of program during run time. In general, the dividend is not a constant but might be a number typed by you at runtime. In this case, **division by zero** is illogical. Computers check for a "division by zero" error during program execution, so that you can get a "division by zero" error message at runtime, which will stop your program abnormally.

Testing and Debugging

Testing is the process of finding errors in a program, and debugging is the process of correcting errors found during testing process.

Documentation refers to written descriptions specification ,design ,code and comment , internal and external to a program more understandable , readable and more easily modifiable

Modules of Documentation :

1. Modules makes information more easily accessible to the specific user for which they were prepared , and reduce the cost of production and maintenance.
2. The documentation modules are generally referred to as manuals .
3. In detail it depends upon Complexity of system, Technical sophistication of user and People involved in development and use.

List of Manuals

1. User manual
2. Input Preparation Manual
3. Operation Manual
4. Equipment Manual
5. Programmer Manual
6. System Manual
7. Standards Manual

What is internal documentation?

The internal documentation includes comments, self documenting code and program's formatting. The goal of this features is to make program readable, understandable , and easily modifiable.

What is self documenting code?

It is a code that uses meaningful names for constants, variables & subprogram Identifiers to clarify their meaning in the program.

Program maintenance

Adaptive maintenance To accommodate changing needs , time to time , maintenance is done and is called adaptive maintenance. For example new government may need to process new reports or market conditions etc.

Preventive maintenance If possible errors could be anticipated , before they actually occur, the maintenance could be done to avoid them and the system down time(time for which system remains out of order) can be saved. This type of maintenance aims at preventing errors is called preventive maintenance.

How to develop information that are easy to maintain?

1. The systems should be planned with an eye on the future.
2. User specification should be correct.
3. The system should be modular
4. Documentation should be complete.
5. Standards should be followed.

6. Testing should be through.

7. Adequate time should be allowed for development cycle.

8. Attetion should be paid to end-users , health and human factors should be considered.

9. The development team should be fully aware at the relationship of system design and system maintenance.