**QUESTION 1**

**DEFINE COMPUTER PROGRAMMING AND COMMENT BRIEFLY ON ANY FIVE PROGRAMMING STYLE EXPRESSION KNOW TO YOU**

**Answer**

**Computer programming**: is the process of writing or developing and implementing useful, maintainable, and extensible source code or various sets of instructions which can be interpreted or compiled by a computing system to perform a meaningful task

**FIVE PROGRAMMING STYLES**

1. **Object-Oriented**: Programming by defining objects that send messages to each other. Objects have their own internal (encapsulated) state and public interfaces. Object orientation can be:

* **Class-based**: Objects get state and behavior based on membership in a class.
* **Prototype-based***:* Objects get behavior from a prototype object.

1. **Functional Programming:** control flow is expressed by combining function calls, rather than by assigning values to variables
2. **Flow-Driven**: Programming processes communicating with each other over predefined channels.
3. **Event-Driven**: Programming with emitters and listeners of asynchronous actions.
4. **Structured**: is a kind of imperative programming where control flow is defined by nested loops, conditionals, and subroutines, rather than via gotos. Variables are generally local to blocks (have lexical scope).

**QUESTION 2**

**WITH THE AID OF A SIMPLE PROGRAM CODE DIFFERENTIATE BETWEEN MONOLITHIC PROGRAMMING AND MODULAR PROGRAMMING CONCEPT**

**Answer**

**Modular programming** is a software design technique that emphasizes separating the functionality of a **program** into independent, interchangeable **modules**, such that each contains everything necessary to execute only one aspect of the desired functionality.

**Example code in php**

<?php

//simple modular code adding two numbers

*function* take\_inputs(){

$num1 = $\_POST [‘num1’];

$num2 = $\_POST [‘num2’];

}

take\_inputs($um1,$num2);//calling the function to take the inputs from input page

*function* calculate($num1,num2){

$result = $num1 + $num2; //calculate the result

} //display the result

Echo “the result of the addition is”. $result;

Include (‘inputpage.php’); //call the page that sends the input from html

?>

**Monolithic Programming Approach:** In this approach, the program consists of sequence of statements that modify data. All the statements of the program are Global throughout the whole program. The program control is achieved through the use of jumps i.e. goto statements. In this approach, code is duplicated each time because there is no support for the function. Data is not fully protected as it can be accessed from any portion of the program. So this approach is useful for designing small and simple programs. The programming languages like ASSEMBLY and BASIC follow this approach.

**Example code in Q-Basic**

CLS

REM program to add two numbers

DIM num1, num2 AS INTEGER

DIM result AS DOUBLE

PRINT "Enter a Number"

INPUT num1

PRINT "Enter another Number"

INPUT num2

LET result = num1 + num2

PRINT "Your Answer is ", result

END

**QUESTION 2B**

**DISCOURSE THE ADVANTAGES OF MODULAR PROGRAMMING OVER MONOLITHIC CONCEPT OF PROGRAMMING**

### The advantages of using modular programming:

* Fewer bug because each set of programming commands is shorter and can be dealt with separately unlike the monolithic programming.
* Algorithm is more easily understood because each module looks independent as a unit of its own.
* many programmers can be employed, one on each of the modules
* programmers can use their expertise on particular techniques
* allows library programs to be inserted easily
* all of which saves time and means the finished program can be completed more quickly
* In addition, a module can be radically changed without affecting other modules as long as its original function remains unchanged, unlike the monolithic programming which modification can affect the whole program

**QUESTION 3**

**COMMENT BRIEFLY ON THE MAJOR ELEMENT OF STRUCTURED PROGRAMMING**

**Answers**

**The major elements of structured programming are as follows**

1. **Subroutine**: is a sequence of program instructions that perform a specific task, packaged as a unit. This unit can then be used in programs wherever that particular [task](https://en.wikipedia.org/wiki/Task_(computing)) should be performed.
2. **Block or code block**: is a lexical structure of [source code](https://en.wikipedia.org/wiki/Source_code) which is grouped together. Blocks consist of one or more [declarations](https://en.wikipedia.org/wiki/Declaration_(computer_science)) and [statements](https://en.wikipedia.org/wiki/Statement_(programming))

**QUESTION 3B**

**WHAT ARE THE CONTROL STRUCTURES 2 HENCE GIVE THREE EXAMPLES OF CONTROL STRUCTURES KNOWN TO YOU.**

**Control structures**: A control structure is a block of programming that analyzes variables and chooses a direction in which to go based on given parameters. The term flow control details the direction the program takes (which way program control "flows"). Hence it is the basic decision-making process in computing; flow control determines how a computer will respond when given certain conditions and parameters

**Examples of control structures**

* **Selection**: one or a number of statements is executed depending on the state of the program. This is usually expressed with [keywords](https://en.wikipedia.org/wiki/Keyword_(computer_programming)) such as [if..then..else..endif](https://en.wikipedia.org/wiki/Conditional_(programming)).
* **Iteration**: a statement or block is executed until the program reaches a certain state, or operations have been applied to every element of a collection. This is usually expressed with keywords such as [while](https://en.wikipedia.org/wiki/While_loop), [repeat](https://en.wikipedia.org/wiki/Do_while_loop), [for](https://en.wikipedia.org/wiki/For_loop) or [do..until](https://en.wikipedia.org/wiki/Do_while_loop). Often it is recommended that each loop should only have one entry point (and in the original structural programming, also only one exit point, and a few languages enforce this).
* **Recursion**: a statement is executed by repeatedly calling itself until termination conditions are met. While similar in practice to iterative loops, recursive loops may be more computationally efficient, and are implemented differently as a cascading stack.

**QUESTION 3C**

**Global variables** are variables declared outside the main function or the main method, and can be accessed (used) on any function or anywhere in the program.

**EXAMPLE OF GLOBAL VARIABLE.**

//program to find the sum and difference between two numbers

int a,b,result;

void main(){

sum();

sub();

getch(); }

sum(){

printf("Enter two numbers to find their sum");

scanf("%d%d",&a,&b);

result=a+b;

printf("\n the sum of two numbers is %d",result);

return 0;}

sub(){

printf("Enter two numbers to find their difference");

scanf("%d%d",&a,&b);

result=a-b;

printf("\n the difference between two numbers is %d",result);

return0; }

***Here, a,b and result are global variables which are declared before the main function.***

**LOCAL VARIABLES** *are variables declared inside a function, and can be used only inside that function.*

**Example of local variable.**

//program to add any two integers

void main(){

function locavar(){

int a,b,sum;

printf("Enter any two integer value");

scanf("%d%d",&a,&b);

sum=a+b;

printf("\nSum of two integers %d and %d is %d", a, b, sum);

} localvar();//calling the function .

}

***Here, a, b, and sum are local variables which are declared in main function.***

**QUESTION 4A**

**DISCUSS ANY TWO MAJOR PROGRAMMING APPROACH OF PROGRAMMING LANGUAGE DESIGN KNOWN TO YOU AND GIVE EXAMPLE OF TWO LANGUAGES THAT EMPLOY THE PRINCIPLE OF EACH OF THE DESIGN OF EACH OF THE DESIGN APPROACH YOU MENTION.**

**Answer**

The two major programming approaches are the **Top down Approach** and the **Bottom up Approach.**

**TOP DOWN APPROACH**: is taking a **"COMPLEX CODE (PROGRAM)"** and breaking it down to the **"SIMPLE INDIVIDUAL COMPONENTS"** to gain a better understanding of the inner layers. Simply putting, it's like decomposition, i.e. breaking into smaller parts.

In top-down approach, main function is written first and all sub functions are called from main function thus, sub-functions are written based on the requirement

Structure / procedure oriented programming languages like **C** programming language and **QBASIC** follows top-down approach.

## DISADVANTAGES OF TOP-DOWN PROGRAMMING

1. Top-down programming complicates testing. Noting executable exists until the very late in the development, so in order to test what has been done so far, one must write [stubs](http://dept-info.labri.fr/~strandh/Teaching/MTP/Common/Strandh-Tutorial/glossary.html).
2. Furthermore, top-down programming tends to generate modules that are very specific to the application that is being written, thus not very reusable.
3. But the main disadvantage of top-down programming is that all decisions made from the start of the project depend directly or indirectly on the high-level specification of the application. It is a well-known fact that this specification tends to change over time. When that happens, there is a great risk that large parts of the application need to be rewritten.

**BOTTOM UP APPROACH**: is building up a **"COMPLEX PROGRAM"** by piecing together "**SIMPLE INDIVIDUAL COMPONENTS"**. You work your way up to the final outcome by inter-relating individual blocks.

In bottom-up approach, code is developed from modules and then these modules are integrated with main function

Object oriented programming languages like **C++** and **JAVA** programming language follows bottom-up approach.

## ADVANTAGES OF BOTTOM-UP PROGRAMMING

Bottom-up programming has several advantages over [top-down programming](http://dept-info.labri.fr/~strandh/Teaching/MTP/Common/Strandh-Tutorial/top-down-programming.html).

Testing is simplified since no [stubs](http://dept-info.labri.fr/~strandh/Teaching/MTP/Common/Strandh-Tutorial/glossary.html)are needed. While it might be necessary to write [test functions](http://dept-info.labri.fr/~strandh/Teaching/MTP/Common/Strandh-Tutorial/glossary.html), these are simpler to write than stubs, and sometimes not necessary at all, in particular if one uses an interactive programming environment such as Common Lisp or GDB.

Pieces of programs written bottom-up tend to be more general, and thus more reusable, than pieces of programs written top-down.

**QUESTION 4B**

**COMMENT BRIEFLY ON THE FOLLOWING TERMS**

1. **Block**: is a lexical structure of source code which is grouped together, written to perform a specific task, example , if(a<10){

a++// increase the value of a if it is less than 10

}

1. **Statement:** is the smallest element of a programming language that expresses some action to be carried out. It is a written instruction in a high-level language that commands the computer to perform a specified action. Example conditional statements like == (equal to), > (greater than), < (less than.) etc**.**
2. **Expression**: is any legal combination of symbols that represents a value or is any valid unit of code that resolves to a value. Example. x = 7, 2+3, c = 3+2 etc.
3. **Function**: is a block or group of statements, which is called upon by the main program to perform a specific task. Example return type function name (argument list){

Set of statements – Block of code

}

1. **Procedure**: Procedures, also known as routines, [subroutines](https://en.wikipedia.org/wiki/Subroutine), or functions simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

**QUESTION 5A**

**WITH THE AID OF A SIMPLE PROGRAM CODES DIFFERENTIATE BETWEEN**

**STRUCTURED PROGRAMMING AND OBJECT ORIENTED PROGRAMMING**

**Answer**

**OOP PROGRAM ON JAVA**

/\*

\* Example to illustrate Method Overloading

\*/

public class **TestMethodOverloading** {

public static int average(int n1, int n2) {

//version A

System.out.println("Run version A");

return (n1+n2)/2;

}

public static double average(double n1, double n2) {

//version B

System.out.println("Run version B");

return (n1+n2)/2;

}

public static void main(String[] args) {

System.out.println(average(1, 2)); // Use A

System.out.println(average(1.0, 2.0)); // Use B

System.out.println(average(1.0, 2)); // Use B - int 2 implicitly casted to double 2.0

//average (1, 2, 3, 4); // Compilation Error - No matching method

}

}

**STRUCTURED PROGRAM**

**Example: Add Two Numbers in Basic**

CLS

10 REM FIND THE SUM OF THE TWO NUMBERS

20 LET A=9

30 LET B=6

40 LET C=A+B

50 PRINT A, B, C

60 END

**SUMMARY**

|  |  |
| --- | --- |
| **STRUCTURED PROGRAMMING** | ****OBJECT ORIENTED PROGRAMMING**** |
| 1. Structured Programming is designed which focuses on **process**/ logical structure and then data required for that process. 2. Structured programming follows**top-down approach**. 3. Structured Programming is also known as **Modular Programming** and a subset of **procedural programming language**. 4. In Structured Programming, Programs are divided into small self-contained **functions**. 5. Structured Programming is **less** secure as there is no way of **data hiding**. 6. Structured Programming can solve **moderately** complex programs. 7. Structured Programming provides **less** **reusability**, more function dependency. 8. Less abstraction and less flexibility. | 1. Object Oriented Programming is designed which focuses on **data**. 2. Object oriented programming follows **bottom-up approach**. 3. Object Oriented Programming supports **inheritance, encapsulation, abstraction**, **polymorphism**, etc. 4. In Object Oriented Programming, Programs are divided into small entities called **objects**. 5. Object Oriented Programming can solve any **complex** programs. 6. Object Oriented Programming provides more reusability, less function **dependency**. 7. Object Oriented Programming provides more reusability, less function **dependency**. 8. More abstraction and more **flexibility** |