**STEPHEN SMITH ONIMISI**

**18/SCI01/088**

**CSC 206**

**1)i)** Programming is the process of writing codes

**ii)** A program is a set of codes that instructs the computer to perform processes

**iii)** Programming languages are languages through which we can instruct the computer to perform processes

 **2)i) Machine language:** Machine language or machine code is the native language directly understood by the computer’s central processing unit or CPU. This type of computer language is not easy to understand, as it only uses a binary system, an element of notations containing only a series of numbers consisting of one and zero, to produce commands.

**ii) Assembly language:** It is a language that consists of some symbolic codes, which are easier to remember than machine codes. it makes use of an assembler to translate codes into machine language.

**iii) High level language:** High Level Languages are user-friendly languages which are similar to English with vocabulary of words and symbols.

* These are easier to learn and require less time to write.
* They are problem oriented rather than ‘machine’ based.
* Program written in a high-level language can be translated into many machine languages and therefore can run on any computer for which there exists an appropriate translator.

**3)**

* It must have syntactic rules for forming statements.
* It must have a vocabulary that consists of letters of the alphabet.
* It must have a language structure, which consists of keywords, expressions and statements.
* It may require a translator before it can be understood by a

 computer.

* Programming languages are written and processed by the

 computer for the purpose of communicating data between the human being and the computer.

**4)**

**Procedural programming:** A procedural programming is a series of steps, each of which performs a calculation, retrieves input, or produces output. Concepts like assignments, loops, sequence and conditional statements are the building blocks of procedural programming. E.g: COBOOL, FORTRAN, c and c++.

**Object-Oriented programming:** An object oriented program is a collection of objects that interact which each other by passing messages that transforms their state. The fundamental building blocks of object oriented programming are object modelling, classification and inheritance. Major object-oriented laguages are: Python, Java, etc.

**Functional Programming:** this is a collection of mathematical functions, each with an input (domain) and a result (range). Interaction and combination of functions is carried out by functional compositions, conditionals and recursion. Major functional programming languages are Lisp, Scheme, Haskell, and ML.

**Logic (Declarative) Programming:** logic programis a collection of logical declarations about what outcome a function should accomplish rather than how that outcome should be accomplished. Logic programming provides a natural vehicle for expressing non-determinism, since the solutions to many problems are often not unique but manifold. The major logic programming language is Prolog.

**Event Driven Programming:** An event driven program is a continuous loop that responds to events that are generated in an unpredictable order. These events originate from user actions on the screen (mouse clicks or keystrokes, for example), or else from other sources (like readings from sensors on a robot). Major event-driven programming languages include Visual basic and Java.

**Concurrent Programming:** A concurrent program is a collection of cooperating processes, sharing information with each other from time to time but generally operating asynchronously. Concurrent programming languages include Linda and High performance FORTRAN.

**5)**

* **Problem Analysis:** This is where the clear statement of the problem is stated. The programmer must be sure that he understands the problem and how to solve it. He must know what is expected of the problem, i.e. what the program should do, the nature of the output and the input to consider so as get the output. He must also understand the ways of solving the problem and the relationship between the input and the expected output.
* **Design:** The planning of the solution to the problem in the first stage takes place in this stage. The planning consists of the process of finding a logical sequence of precise steps that solve the problem. Such a sequence of steps is called an **algorithm.** Every detail, including obvious stepsshould appear in the algorithm. The three popular methods used to develop the logic plan are: flowcharts, a pseudo code, and a top-down chart. These tools help the programmer break down a problem into a sequence of small tasks the computer can perform to solve the problem. Planning may also involve using representative data to test the logic of the algorithm by hand to ensure that it is correct.
* **Coding:** Translation of the algorithm in stage two into a programming language takes place here. The process for writing the program is called **coding.** The programmeruses the algorithm devised in the design stage along with the choice of the programming language he got from stage three.
* **Testing and Debugging:** The process involves the location and removal of error in the program if any. Testing is the process of checking if the program is working as expected and finding errors in the program, and debugging is the process of correcting errors that are found (An error in a program is called a bug.).
* **Documentation:** This is the final stage of program development. It consists of organizing all the material that describes the program. The documentation of the program is intended to allow another person or the programmer at a later date, to understand the program. Documentation might also consist of a detailed description of what the program does and how to use the program. Although documentation is listed as the last step in the program development cycle, it should take place as the program is being coded.

**6)**

* **Naming Conventions:** It is very important to give meaningful names to all your constructs. A name like **get\_Height() or get\_avg\_height()** gives us much more information than **ctunde().** Also, a variable name**- total** for addition is more meaningful than **pen.** The name of a class should communicate its purpose. Class name should start with an uppercase letter, e.g. **class AddPrime.**
* **File Naming and Organization:** Files should be organized into directories in a module-wise fashion instead of having a monolithic structure where all source code files and all header files are in a single directory. This should be part of the design process.
* **Formatting and Indentation:** The lines within the code should be clearly organized in a way that it will be easy to read and understand even for the writer. Proper identification should be used to show subordinate lines.
* **Comments and Documentation:** Introducing comments and proper explanations (documentation) of the program aid in understanding the code. They help us in following the program flow, and skip parts for which we are not interested in details. This allows for program amendment and extensibility.
* **Classes**: Ensure that all the classes in your application have a default constructor, copy constructor and overloaded operator. Also ensure that all the class data items are appropriately initialized in constructor and assigned to each member of the class.
* **Minimizing Bugs by Testing:** Testing is an integral part of software development. Tests help us not only in making sure that what we have written is correct, but also in finding out if someone breaks the code later. So, it is a good programming style to thoroughly test a program.

**7)**

Structured programming is a programming paradigm aimed at improving the clarity, quality and development time of computer program by making extensive use of subroutines, block structures, for and while loops.

**8)**

* To increase programmers’ productivity
* To increase program clarity by reducing complexity
* To reduce program testing time
* To reduce program maintenance time and effort.
* Structured programming gives room for well-thought-out program logic and provides an attempt to keep programs as simple and straight forward as possible.

**9)**

* **Sequence:** sequence refers to an ordered execution of statements. It has one entry and exit points.
* **Selection**: it uses conditions and one of a number of statements is executed depending on the state of the program. This is usually expressed with keywords such as if …then…else…endif, switch, or case.
* **Repetition**: in repetition, a statement 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, repeat, for or do…until. Often it is recommended that each loop should have only one entry point.

**10)**

* Firstly, I will write down my algorithm to this problem.
* Then I will ascertain all courses offered by all students in the college of sciences.
* Then using a conditional statement if a particular course is held by a particular number of students, and taught by a particular lecturer in a particular location, then those students will have no other class, and no other set of students are to have another class at that same location, then the data would not be accepted. Else it would be denied.

**11)**

**Algorithm:**

* Start
* Input value of PI as 3.14
* Input R as 23.12
* Compute SPHERE AS (4/3)\*(PI)(R^3)
* Print SPEHERE
* Stop

 **Flowchart:**

INPUT PI=3.14

INPUT R=23.12

SPERE=(4/3)\*(PI)\*(R^3)

PRINT SPHERE

**12)**

**Algorithm:**

* Start
* Input volume(v)of cone as 200(cm^3) and height(h) as 18.13m
* Input pi as 3.14
* Compute radius(r) = $√($(v\*3)/(pi\*h))
* Print r
* Stop

 **Flowchart:**

INPUT v=200cm₃ and h=18.13m

INPUT pi=3.14

(r) = $√($(v\*3)/(pi\*h))

PRINT radius

**13)**

 Structured programming object oriented

1. A programming paradigm that divides A programming paradigm based on

The code into modules or function the concept of objects, which

 Contain data in the form of fields

 Known as attributes, and code in

 The form of procedures known

 As methods.

1. Difficulty to modify structured Easier to modify object oriented

Programs programs

1. There are no access to specifiers There are access specifiers such as

 Private, public and protected

1. Data is not secured Data is secure.
2. Difficult to rescue code easy to rescue code

**14)**

 **program to find the circumference of a circle.**

 #include

 #define pi 3.142

int main ()

{ const float rad = 7.5;

 float cir;

cir = 2 \* pi \* rad;

 printf(“the value of circumference of a circle is %d”, cir);

return 0; }

**EXPLANATION**

1. The first line “#include” includes or adds header files in could come with the compiler or you could make your own header file.
2. The second line ”define pi 3.142” makes pi as 3.142 anytime it is called in the program.
3. The third line “int main()” shows the beginning of the main program.
4. The fourth line “const float rad= 7.5;” makes 7.5 as constant for rad variable under float data type
5. The fifth line “float cir;” is basically declaring cir variable as a float.
6. The sixth line “cir=2\*pi\*rad;” is just computing the value of cir that is the circumference.
7. The seventh line “printf(“the value of circumference of a circle is%d”,cir);” is just to print out the value of a circle and the ”%d” there is very wrong there because “cir” was declared as a float so it is meant to be “%f”.
8. The last line “return 0;” is basically to return value back to origin.