**ASSIGNMENT ON CSC 302- SURVEY ON PROGRAMMING LANGUAGES**

1. How programming languages are organized along the given rubrics;
2. Programming languages organized based on the basic 7 programming paradigms;
* **UNSTRUCTURED PROGRAMMING**

This programming is identified by the following features;

* Uses unstructured jumps to labels or instruction addresses.
* The lines are usually numbered; allow the flow of execution to jump to any line in the program.
* In contrast to structured programming which uses structured constructs of selection (If/then/else) and repetition (while and for).

EXAMPLES OF DIFFERENT PROGRAMMING LANGUAGES ARE; **BASIC, FORTRAN, COBOL, FOCAL ETC**

* **STRUCTURED PROGRAMMING(MODULAR PROGRAMMING)**

This programming id identified by the following features;

* Top-down implementation which improves readability and maintainability of code.
* Promotes code reuse.
* Makes use of structured control flow constructs of selection (if/then/else), and repetition(while and for), block structures and subroutines.

EXAMPLES ARE; **ALGOL, PASCAL, ADA.**

* **OBJECT ORIENTED PROGRAMMING**

This programming id identified by the following features;

* Use objects which can contain data
* Object data are in form of fields.
* Object codes are in form of procedures.

EXAMPLES ARE; **PYTHON, RUBY, EMERALD ETC.**

* **ASPECT ORIENTED PROGRAMMING**

This programming id identified by the following features;

* Top-down implementation which improves readability and maintainability of code.
* Promotes code reuse.
* Makes use of structured control flow constructs of selection (if/then/else), and repetition(while and for), block structures and subroutines.

EXAMPLES ARE; **.**

* **ACTIVITY ORIENTED PROGRAMMING**

This programming id identified by the following features;

* Top-down implementation which improves readability and maintainability of code.
* Promotes code reuse.
* Makes use of structured control flow constructs of selection (if/then/else), and repetition(while and for), block structures and subroutines.

EXAMPLES ARE; **ALGOL, PASCAL, ADA.**

* **EVENT-ORIENTED PROGRAMMING**

This programming id identified by the following features;

* Top-down implementation which improves readability and maintainability of code.
* Promotes code reuse.
* Makes use of structured control flow constructs of selection (if/then/else), and repetition (while and for), block structures and subroutines.

EXAMPLES ARE; **ALGOL, PASCAL, ADA.**

**ii) Based on domain requirements;**

**iii) Based on requirements i and ii above**

1. **EVOLUTION OF PROGRAMMING LANGUAGES IN CHRONOLOGICAL ORDER**

1950s

Autocode (1952): This family of “simplified coding systems” was created in the 1950s specifically for use with the digital computers at the universities of Manchester, Cambridge and London. Considered by many to be the first complied programming language ever invented, Autocode was developed by Alick Glennie to be both comprehensible and high-level.

Fortran (1957): Fortran is a general-purpose, imperative programming language suited to numeric computation and scientific computing. In use for over half a century, Fortran was developed by IBM in 1957 for both scientific and engineering applications.

1960s

Algol 68 (1968): Short for Algorithmic Language 1968, Algol 68 was an imperative programming language designed as a successor to Algol 60. With a wider scope of application and rigorously defined syntax, this language was the first to be fully defined before it was implemented.

1970s

Pascal (1970): Named in honor of the French mathematician Blaise Pascal, this programing language was developed by Niklaus Wirth. Pascal enabled programmers to define their own complex data types and made it easier to build dynamic and recursive data structures like lists, trees and graphs.

C (1972): One of the most widely used programming languages of all time, C is a general-purpose language designed for structured programming. C program source text is free-format, using the semicolon as a statement terminator and curly braces for grouping blocks of statements.

1980s

C++ (1980): This programming language was designed mainly for system programming but has expanded to be used in desktop, servers and performance-critical applications. It inherited most of C’s syntax and has imperative, object-oriented and generic programming features.

Perl (1987): Perl is a family of high-level, general-purpose programming languages. It borrows features from other programming languages, such as C, AWK and sed. Originally, the only documentation for Perl was a single manual page, but it has gone through several revisions and changes.

1990s

Python (1991): Python’s design philosophy focuses on readability. A successor to the ABC language, its syntax allows programmers to express concepts in fewer lines of code than is possible in languages such as C++ or Java.

Java (1995): This programming language was designed to have as few implementation dependencies as possible, giving it a wide variety of applications. It is intended to let application developers “write once, run anywhere,” so that Java can run on any platform that supports it without the need to recompile.

PHP (1995): PHP is a server-side scripting language used for both Web development and general-purpose programming. This language can be mixed with HTML code or used in combination with templating engines and Web frameworks. It was originally not meant to be a programming language, but grew organically over time.

2000s

C# (2001): This multi-paradigm programming language was developed by Microsoft within the .NET framework. It was intended to be simple, modern and object-oriented. The most recent version of C# was released in 2012.

Visual Basic .NET (2001): A successor to the original Visual Basic language, Visual Basic .NET is a high-level programming language implemented on the .NET framework. It uses statements to specify actions and is one of the two main languages targeting the .NET framework, along with Visual C#.

2010s-Present

Swift (2014): Swift was created by Apple for iOS and OS C development. It was introduced in 2014 at Apple’s Worldwide Developers Conference. Designed to work with Apple’s Cocoa and Cocoa Touch frameworks, Swift is meant to be more concise and resilient to erroneous code.



1. **Distinguishing between modular programming and object oriented programming ;**

|  |  |
| --- | --- |
| **MODULAR PROGRAMMING** | **OBJECT-ORIENTED PROGRAMMING** |
| **-**Focuses on process. | **-**Focuses on data. |
| **-**Follows top-down approach. | **-**Follows bottom-up approach. |
| -Programs are divide into small self-contained functions. | **-**Programs are divided into small entities called objects. |
| **-**Provides les reusability, more function dependency. | -Provides more reusability, less function dependency. |
| -Less abstraction and less flexibility. | **-**More abstraction and more flexibility. |