BISHOP SCOTT BOYS' SCHOOL
(Affiliated to CBSE, New Delhi) Affiliation No.: 330726, School Campus: Chainpur, Jaganpura, By-Pass, Patna 804453.
Phone Number: 7061717782, 9798903550. ,
Web: www.bishopscottboysschool.com Email: info@bishopscottboysschool.com

Sarvada Sarva shreshth
HOLISTIC DEVELOPMENT

STUDY COURSE MATERIAL
COMPUTER
SESSION-2020-21

TOPIC: CHAPTER-4 INTRODUCTION TO PROGRAMMING CONCEPTS

DAY-1

❖ TEACHING MATERIAL

Generations of programming language

Programming languages have been developed over the year in a phased manner. Each phase of developed has made the programming language more user-friendly, easier to use and more powerful. Each phase of improved made in the development of the programming languages can be referred to as a generation. The programming language in terms of their performance reliability and robustness can be grouped into five **different generations**,

1. First generation languages (1GL)
2. Second generation languages (2GL)
3. Third generation languages (3GL)
4. Fourth generation languages (4GL)
5. Fifth generation languages (5GL)

1. First Generation Language (Machine language)

The first generation programming language is also called low-level programming language because they were used to program the computer system at a very low level of abstraction. i.e. at the machine level. The machine language also referred to as the native language of the computer system is the first generation programming language. In the machine language, a programmer only deals with a binary number.

Advantages of first generation language

- They are translation free and can be directly executed by the computers.
- The programs written in these languages are executed very speedily and efficiently by the CPU of the computer system.
- The programs written in these languages utilize the memory in an efficient manner because it is possible to keep track of each bit of data.

❖ VIDEO-LINKS

LINK-1

https://www.youtube.com/watch?v=cbck7g2S6Io

❖ DOCUMENTS LINKSj

https://www.microcontrollertips.com/compilers-translators-interpreters-assemblers-faq/
https://www.includehelp.com/basics/generations-of-programming-language.aspx

DAY-2

❖ TEACHING MATERIAL

2. Second Generation language (Assembly Language)

The second generation programming language also belongs to the category of low-level- programming language. The second generation language comprises assembly languages that use the concept of mnemonics for the writing program. In the assembly language, symbolic names are used to represent the opcode and the operand part of the instruction.

Advantages of second generation language

- It is easy to develop understand and modify the program developed in these languages are compared to those developed in the first generation programming language.
- The programs written in these languages are less prone to errors and therefore can be maintained with a great case.

3. Third Generation languages (High-Level Languages)

The third generation programming languages were designed to overcome the various limitations of the first and second generation programming languages. The languages of the third and later generation are considered as a high-level language because they enable the programmer to concentrate only on the logic of the programs without considering the internal architecture of the computer system.

Advantages of third generation programming language

- It is easy to develop, learn and understand the program.
- As the program written in these languages are less prone to errors they are easy to maintain.
- The program written in these languages can be developed in very less time as compared to the first and second generation language.

Examples: FORTRAN, ALGOL, COBOL, C++, C

4. Fourth generation language (Very High-level Languages)

The languages of this generation were considered as very high-level programming languages required a lot of time and effort that affected the productivity of a programmer. The fourth generation programming languages were designed and developed to reduce the time, cost and effort needed to develop different types of software applications.

Advantages of fourth generation languages

- These programming languages allow the efficient use of data by implementing the various database.
- They require less time, cost and effort to develop different types of software applications.

- The program developed in these languages are highly portable as compared to the programs developed in the languages of other generation.

Examples: SOL, CSS, coldfusion

5. Fifth generation language (Artificial Intelligence Language)

The programming languages of this generation mainly focus on constraint programming. The major fields in which the fifth generation programming language are employed are Artificial Intelligence and Artificial Neural Networks

Advantages of fifth generation languages

- These languages can be used to query the database in a fast and efficient manner.
- In this generation of language, the user can communicate with the computer system in a simple and an easy manner.

Examples: mercury, prolog, OPS5

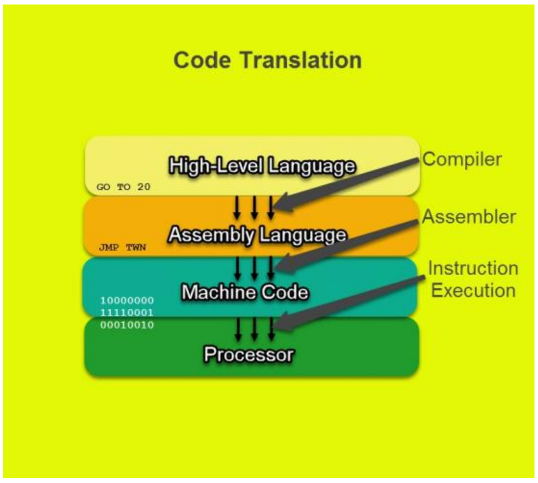
❖ VIDEO-LINKS

https://www.youtube.com/watch?v=cbck7g2S6Io

DAY-3

Translators
The most general term for a software code converting tool is "translator." A translator, in software programming terms, is a generic term that could refer to a compiler, assembler, or interpreter; anything that converts higher level code into another high-level code (e.g., Basic, C++, Fortran, Java) or lower-level (i.e., a language that the processor can understand), such as assembly language or machine code. If you don't know what the tool actually does other than that it accomplishes some level of code conversion to a specific target language, then you can safely call it a translator.

Compilers
Compilers convert high-level language code to machine (object) code in one session. Compilers can take a while, because they have to translate high-level code to lower-level machine language all at once and then save the executable object code to memory. A compiler creates machine code that runs on a processor with a specific Instruction Set Architecture (ISA), which is processor-dependent. For example, you cannot compile code for an x86 and run it on a MIPS architecture without a special compiler. Compilers are also platform-dependent. That is, a compiler can convert C++, for example, to machine code that's targeted at a platform that is running the Linux OS. A cross-compiler, however, can generate code for a platform other than the one it runs on itself.



❖ VIDEO-LINKS

MUST WATCH

https://www.youtube.com/watch?v=cbck7g2S6Io

DAY-4

❖ **Interpreters**
Another way to get code to run on your processor is to use an interpreter, which is not the same as a compiler. An interpreter translates code like a compiler but reads the code and immediately executes on that code, and therefore is initially faster than a compiler. Thus, interpreters are often used in software development tools as debugging tools, as they can execute a single in of code at a time. Compilers translate code all at once and the processor then executes upon the machine language that the compiler produced. If changes are made to the code after compilation, the changed code will need to be compiled and added to the compiled code (or perhaps the entire program will need to be re-compiled.) But an interpreter, although skipping the step of compilation of the entire program to start, is much slower to execute than the same program that's been completely compiled.

❖ Interpreters, however, have usefulness in areas where speed doesn't matter (e.g., debugging and training) and it is possible to take the entire interpreter and use it on another ISA, which makes it more portable than a compiler when working between hardware architectures. There are several types of interpreters: the syntax-directed interpreter (i.e., the Abstract Syntax Tree (AST) interpreter), bytecode interpreter, and threaded interpreter (not to be confused with concurrent processing threads), Just-in-Time (a kind of hybrid interpreter/compiler), and a few others. Instructions on how to build an interpreter can be found on the web.[i] Some examples of programming languages that use interpreters are Python, Ruby, Perl, and PHP.

❖ **Assemblers**
An assembler translates a program written in assembly language into machine language and is effectively a compiler for the assembly language, but can also be used interactively like an interpreter. Assembly language is a low-level programming language. Low-level programming languages are less like human language in that they are more difficult to understand at a glance; you have to study assembly code carefully in order to follow the intent of execution and in most cases, assembly code has many more lines of code to represent the same functions being executed as a higher-level language. An assembler converts assembly language code into machine code (also known as object code), an even lower-level language that the processor can directly understand.

- Assembly language code is more often used with 8-bit processors and becomes increasingly unwieldy as the processor's instruction set path becomes wider (e.g., 16-bit, 32-bit, and 64-bit). It is not impossible for people to read machine code, the strings of ones and zeros that digital devices (including processors) use to communicate, but it's likely only read by people in cases of computer forensics or brute-force hacking

- Procedural programming means that you define a program and its subprograms as a series of steps. In contrast, declarative programs try to describe the result without regard to the steps taken to computer it but rather some description or denotation of the desired result.

- Object oriented programming is a way of organizing code around the principles of encapsulation, inheritance, substitution, programming to interfaces, and so on. Object oriented programs are usually mostly procedural.

- Event based programming is about writing event handling procedures and having the core event loop provided by the underlying system. In this way you can save the trouble of writing your own event loop and benefit from various libraries that already work with the system provided event loop. Event based programs are very often writing using object oriented style, but not always.

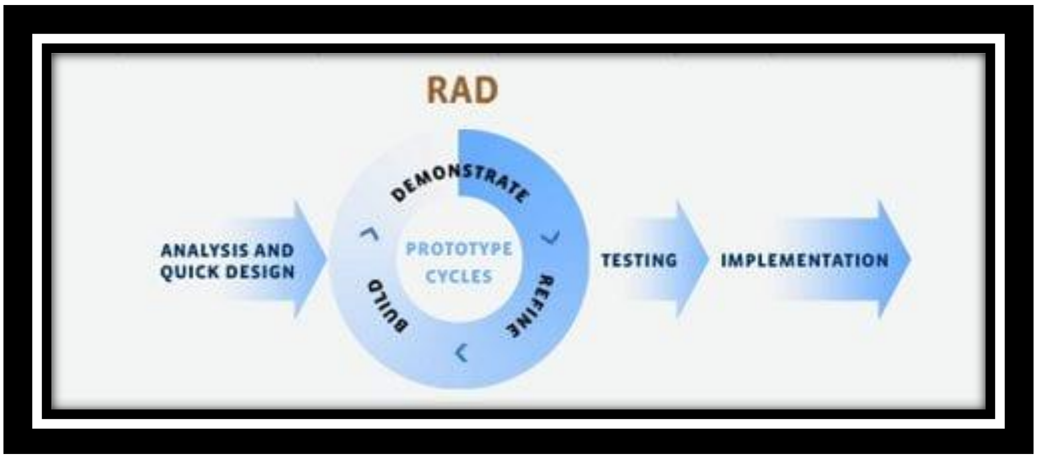
- These three categories are thus not related strictly hierarchically, but in common usage they are mostly nested within one another.

Rapid Application Development (RAD) is a form of agile software development methodology that prioritizes rapid prototype releases and iterations. Unlike the Waterfall method, RAD emphasizes the use of software and user feedback over strict planning and requirements recording.

Some of the key benefits and advantages of RAD are:

- Enhanced flexibility and adaptability as developers can make adjustments quickly during the development process.
- Quick iterations that reduce development time and speed up delivery.
- Encouragement of code reuse, which means less manual coding, less room for errors, and shorter testing times.
- Increased customer satisfaction due to high-level collaboration and coordination between stakeholders (developers, clients, and end users).
- Better risk management as stakeholders can discuss and address code vulnerabilities while keeping development processes going.
- Fewer surprises as, unlike the Waterfall method, RAD includes integrations early on in the software development process.

5 steps or phases in RAD



❖ DOCUMENTS LINKS

https://blog.capterra.com/what-is-rapid-application-development/

https://www.quora.com/What-are-the-relationships-between-programming-procedural-object-oriented-and-event-driven-paradigms

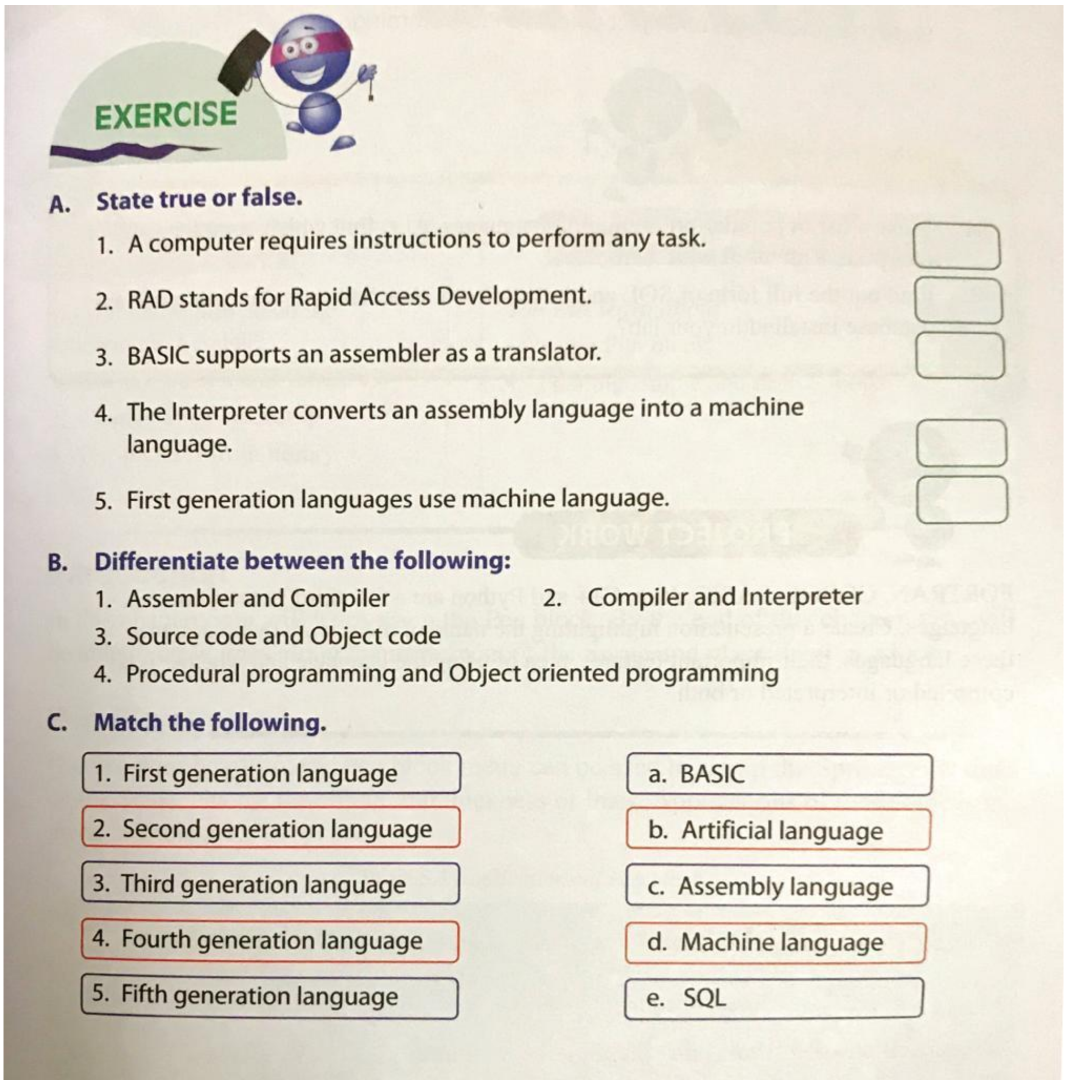
❖ VIDEO-LINKS

https://www.youtube.com/watch?v=cbck7g2S6Io

DAY-5

EXERCISE:

1.



EXERCISE

A. State true or false.

1. A computer requires instructions to perform any task. ☐
2. RAD stands for Rapid Access Development. ☐
3. BASIC supports an assembler as a translator. ☐
4. The Interpreter converts an assembly language into a machine language. ☐
5. First generation languages use machine language. ☐

B. Differentiate between the following:

1. Assembler and Compiler
2. Compiler and Interpreter
3. Source code and Object code
4. Procedural programming and Object oriented programming

C. Match the following.

1. First generation language	a. BASIC
2. Second generation language	b. Artificial language
3. Third generation language	c. Assembly language
4. Fourth generation language	d. Machine language
5. Fifth generation language	e. SQL

2. Answer the following questions:

I. What is program?
II. Name the five generations of computer languages.
III. State any two important points of a first generation computer languages?
IV. What are the important features of a fourth generation computer languages?
V. State the advantages of Object Oriented Programming.