Lecture 1

11 September 2024 09:06 AM

Lecture 2

11 September 2024 09:06 AM

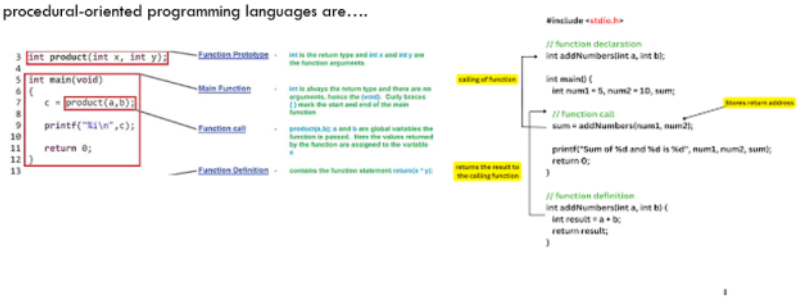
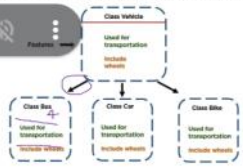
Recap-

1. Procedural Programming focuses on Functions which are like the steps
2. Object Oriented Programming focuses on Classes/Real World entity which are like objects

Examples

Procedural Oriented Programming	Object Oriented Programming
Student Wakes Up -> Gets Ready -> Attend Class Teacher Wakes Up -> Get Ready -> Prepare -> Give Lectures	Student Class Name, Roll No, Branch Functions -Wakes Up, Gets Ready, Attend lecture, Write Exam Teacher Class Name, ID, Branch, Wakes Up, Get Ready, Prepare, Give Lectures, Evaluate

There is no ownership in POP
If(Gives_Lecture){
Attend_Class;
}
There is no description for different programs. There are only functions.
	The function attend lecture is dependent on the teacher
	Eg. If(Teacher.Gives_Lectures == True){
	Student.Attend_lecture;
	}
	If(student.write_exam==True){
	Teacher.evaluate;
	}
	If instead of student we write teacher, then the programming language will show an error.

Procedural Oriented Programming	Object Oriented Programming
Functions are steps which are executed in a particular sequence.	Object is a real world entity, which can be defined by characteristics and functions.
Procedural Programming is not very organised as any function can be associated to any entity.	Object Oriented is more organized; Because we structure the functions in different entities
•PROCEDURAL-ORIENTED PROGRAMMING LANGUAGES: •Procedural-oriented programming languages are based upon procedural call concepts. •This kind of programming language divides the entire program into small portions called routines or functions. •This kind of programming language makes it easy for the user to easily track the flow of control of the program and code re-usability can be done throughout the program. •Some of the procedural-oriented programming languages are…. •FORTRAN •ALGOL •COBOL •BASIC •PASCAL •C 	•OBJECT-ORIENTED PROGRAMMING LANGUAGES: • Object-oriented programming languages are a kind of language, which mainly depends on objects. • In this type of programming language, the entire program is divided into parts, which are known as objects. • These objects are used to implement some of the programming methods like polymorphism, inheritance, abstraction, etc. • The main advantage of object-oriented programming languages is that these kinds of languages are faster and easier to execute, and easy to maintain. Some of the object-oriented programming languages are…. Scala C++ Java Python C# ruby 
Communication between the functions	Communication between the objects

Function Declaration - Tells us the datatype and the name of its function.
Function Definition - Tell us the functionality and the logic of the function.

High Level Language and Low Level Language.

High-Level Language	Low-Level Languages
High-Level Languages are simple to pick up and comprehend.	Low-level languages may be difficult to learn and comprehend.
Because they need translation software, they run slower than lower-level languages.	They work at a breakneck pace.
They allow for a great deal more abstraction.	They don't allow much if any, abstraction.
At the hardware level, they don't have a lot of options.	They are extremely near to the hardware and may assist in the development of software at the hardware level.
Hardware expertise is not necessary for creating programs.	Hardware expertise is required while creating programs.
The programs are simple to change.	It's tough to change programs.
Several instructions may be executed by a single statement.	The assertions may be directly translated to instructions on the CPU.

High Level Language is same on all devices

Low level language maybe specific for a singular device because of different versions CPU,

Context for Tomorrow's Class
Linker - Links two or more functions and explains the sequence.
Loader - Loads the functions into the CPU.
Compilers

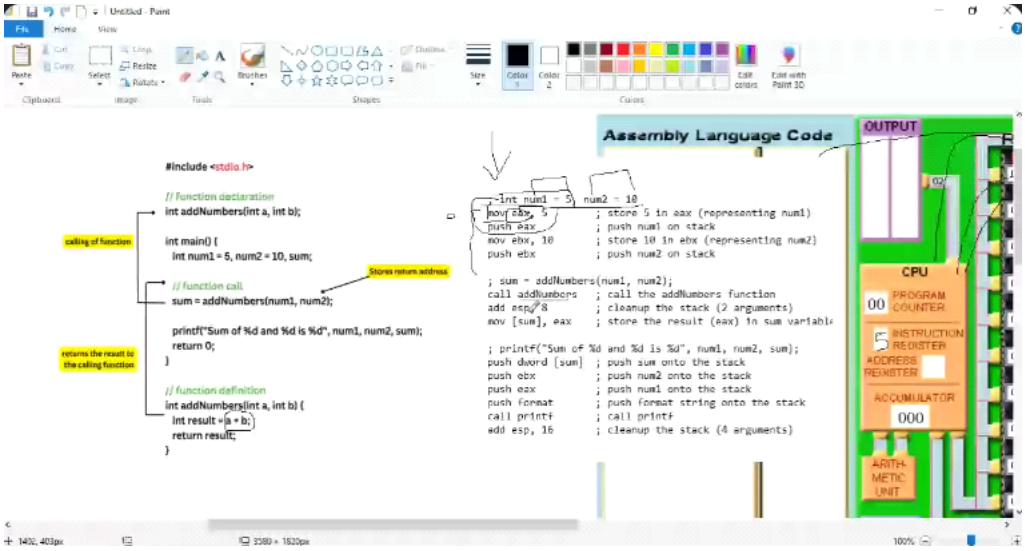
Lecture 3

11 September 2024 09:07 AM

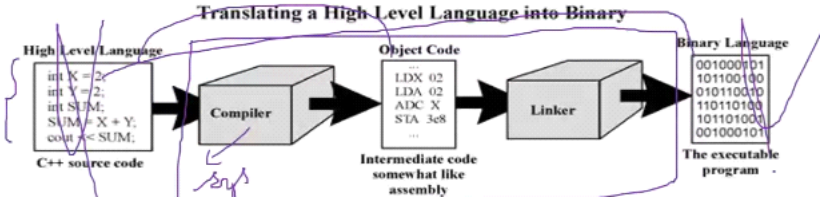
Lecture 4

12 September 2024 08:53 AM

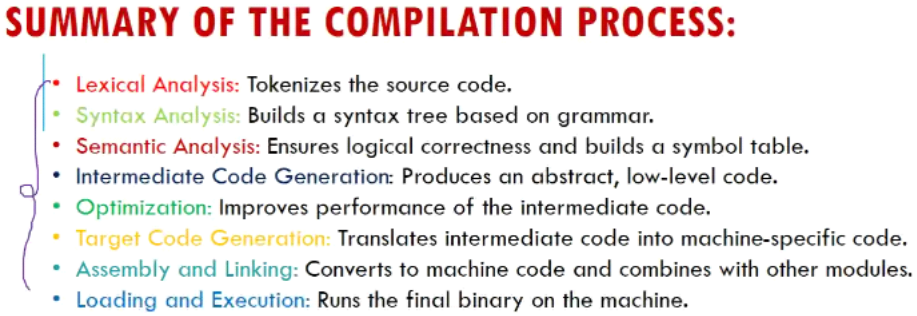
1. Assembly language acts as a middle ground between our code and memory language(binary)
2. Execution of code is happening on CPU.

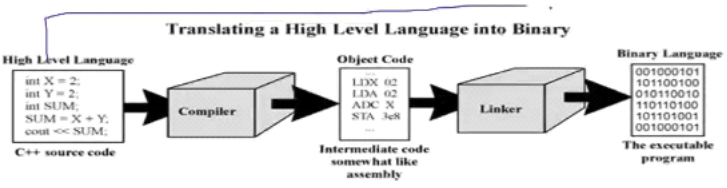


- High Level Language --> Compiler --> Assembly Code --> Linker--> Binary Code
-



-





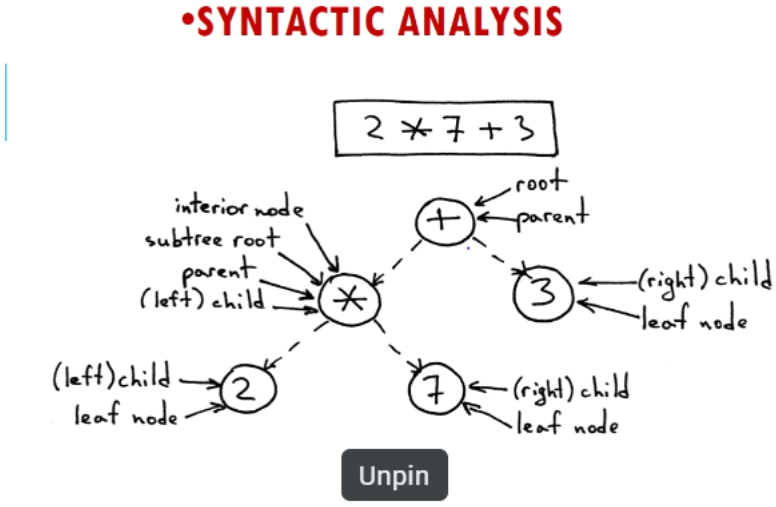
Lexical Analysis :
Syntax Analysis
Semantic Analysis

Lexical Analysis : Tokenizes the source code

1. Token - smallest unit code in a programming language.
2. Example int x = 5;
   Tokens KEYWORD "int", IDENTIFIER "x", OPERATOR "= ", CONSANT "5" , PUNCTUATION ";" are the 5 tokens.

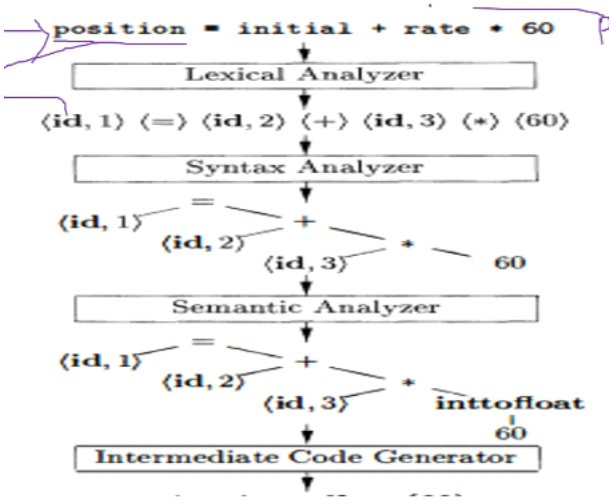
Syntax Analysis - Builds a syntax tree based on grammar

1. Syntax - Programming rules which are language specific.



1. The parser checks the sequence of tokens, confirms the grammar rules of the language.
2. It constructs a parse tree that shows the syntactic structure of the program.

Sematic Analysis

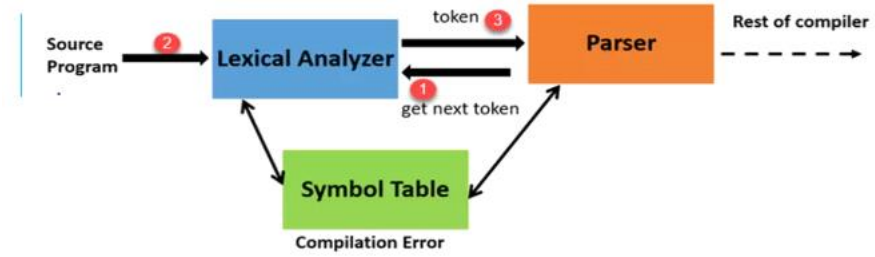


Converts int into float if, the value of data changes during operation.
Or if the value of data becomes float and if we only want int then semantic analyser converts it back to
Checks if the code us logically correct or not.

Lecture 5

13 September 2024 09:10 AM

Symbol table



Eg. Int x =10; --> <id int>, <id x>, <id =>, <id 10> <id ;>
After performing the lexical analysing and after dividing into tokens, we put everything Into a table called Symbolic Table.

Example -

Identifier	Type	Scope	Memory Address	Additional Info
x	int	main	0x1000	Initialized to 10
y	float	main	0x1004	Initialized to 20.5
z	int	main	0x1008	Result of x + y

-> Symbol Table contains - name of identifier, the type of data, the scope of the data, its memory location and additional value associated with it.

Types -
To operate on any variable, the both variables should have the same data type. So we use Symbol Table to check the variable data type. If not we correct it through semantic analysis.