Coding

→ Coding sometimes called computer programming, is how we communicate with computers. Code tells a computer action to take, so writing code is like creating a set of instructions.

Programming

→ Programming is the process of creating a set of instructions that tell a computer how to perform a task.

Code

→ Code refers to a statements written in a programming language, processed by a compiler to run on a computer.

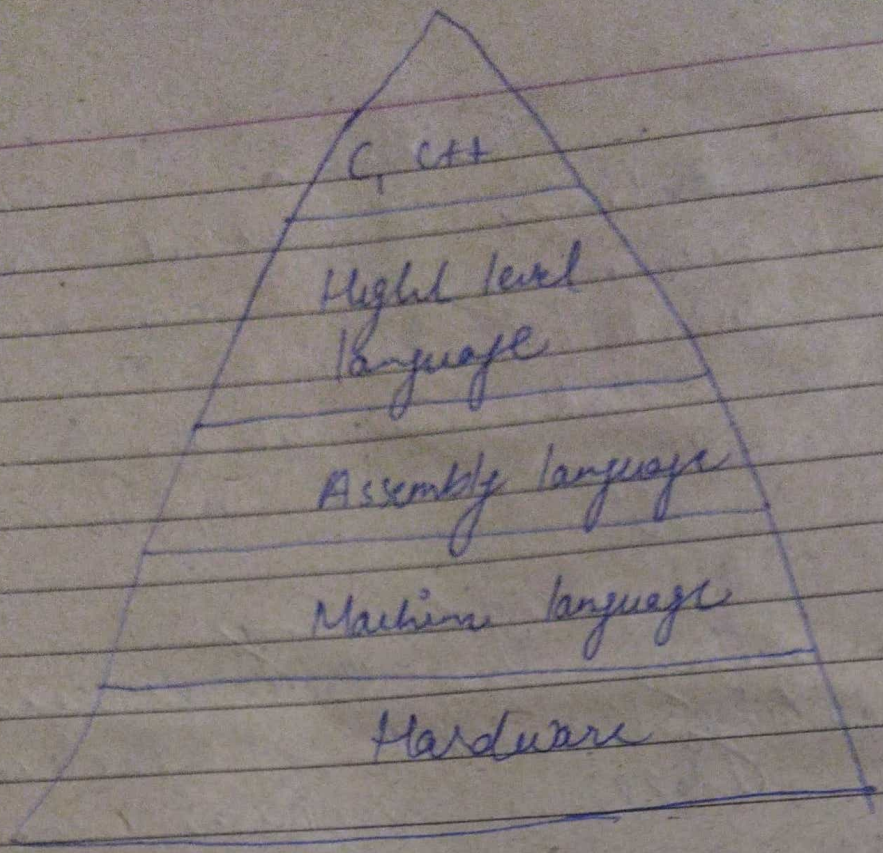
Pseudo Code

→ Pseudo code is an artificial & informal language that helps programmers develop algorithms. Pseudo-code is "text based" detail (algorithm) design tool. The rules of Pseudo code are reasonably are straightforward.

Algorithm

→ An algorithm is a set of commands that must be followed for a computer to perform calculations or other problem-solving operation.

→ An algorithm is defined as a step-by-step process that will be designed for a problem.



· Highlevel language

→ A high-level language (HLL) is a programming language such as C, fortan, or Pascal that enables a programmer to write programs that are more or less independent of a particular type of computer.

Assembly language

→ An assembly language is a type of low-level language that is intended to communicate directly with a computer's hardware eg: MIPS, INTEL64/AMD64, ARM64.

Machine Language

→ Machine code, also known as machine language is the elemental language of computers. It is read by the computer's central processing unit is composed of digital number of look like very long sequence of zeroes and ones.

Assembler

→ An assembler is a program that takes basic computer instruction & converts them into a pattern of bits that the computer processor can use to perform.

Intrepreter

→ An Interpreter translates code into machine code, instruct^{n} by instruction the CPU executes each instruct^{n} before the interpreter moves on to translate the next instruction.

Compiler

→ The language processor that reads the complete source program written in high-level language as a whole in one go & translates it into an equivalent program in machine language is called Compiler.
ex: C, C++, C#, java.

Programming language

→ A programming language is one kind of computer language, and are used in computer programming to implement algorithms.

* C++ Programing Language

→ C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
Designed by: Bjarne Stroustrup.

*Hello World Program

```
// Hello world program in c++

#include<iostream>
using namespace std;

int main()
{
    cout<<"Hello World"\n";
    return 0;
}
```

1. Comments:
The two slash(//) signs are used to add comments in a program. It does not have any effect on the behaviour or outcome of the program. It is used to give description of the program you're writing.

2. #include <iostream>
→ it is pre-processor directive i.e use to include files in our program.
→ Here we are including the iostream Standard file which is necessary for the declaration of basic standandard input/output library in C++.

3. Using namespace std;
→ All elements of the standard C++ library are declared with namespace, here we are using std namespace.

4. int main()
→ The execution of any C++ program starts with the main function, hence it is necessary to have a main function in your program.
→ 'int' is the return value.

5. Cout<<"Hello World"\n";
→ This is a C++ statement. Cout represent the standard output stream in C++.
→ It is declared in the iostream Standard further std namespace.

6. return 0;
→ return signifies the end of a function. here the function is main, so when we put return 0, it exist the program. We are returning 0 because we mentioned the return type of main funct^n as Integer. If a one indicate that something has gone wrong.

Variable

→ A variable is a container used to hold data.
→ Each variable should be given a unique name (identifier)

Fundamental Data Types in C++.

→ Data types are declaration of variable.
→ This determine the type & size of data associated with variable which is essential to know since different data types occupy different size of memory.

Data Type	Meaning	Size (in Byte)
int	Integer	4
float	Floating Point	4
double	double floating-point	8
char	Character	1
wchar_t	wide Character	2
bool	Boolean	1
void	empty	0

- int → −2147483648 to 2147483647
- float & double → decimal & exponential upto 7 digi
- char → '' '' double quote, ASCII Code used
- bool → 2 value True or False, 1 → T & 0 → F.

Decision Making

1. if/else
→ The if block is used to specify the cond^n code to be executed if the specified in if is true, the else block is executed otherwise.

2. else if
→ To specify multiple if condition, we first used if at then the consecutive statement used else if.

3. nested if
→ To specify condition within condition we make the use of nested ifs.

Loops in C++
→ A loop is used for executing a block of statements repeatedly until a particular condition is satisfied.
→ A loop consist:
i. initial statement
ii. test statement
iii. increament statement.

* Loops are -
1. for loop: syntax for(int i=0; i<n; i++).
2. while loop: syntax while(i<n) { }
3. do while: syntax do { } while(i<n);

Jumps in Loops.

→ Jumps in loops are used to control the flow of loops.
→ There are two statement, i) continue, ii) Break.
→ We are using this when we want to change the flow of the loop, when some specified condition is met.

1) Continue:
→ Continue statement is used to skip to the next iteration of that loop.
→ This means that it stops one iteration of the loop.
→ All the statements present after the continue statement in that loop are not executed.

2) Break:
→ Break statement is used to terminate the current loop.
→ As soon as the break statement is encountered in a loop, all further iterat^n of the loop are stopped & control is shifted to the first statement after the loop.

Switch Statement ★

→ switch case statement are a substitute for long if statements that compare a variable to multiple values. After a match is found, it executes the corresponding code of that value case.

operator in C++

→ Operators are nothing but symbol that tell the compiler to perform some specific operation.

* Operater are of following type.

1. Arithmetic operator
   - Unary → one operand
   - Binary → two operand.

- Pre-increment: It increament the value of the operand instantly.

ex ++a, a=5 → 6.
b = ++a → b→5, a→6.

- Post-increment: It stores the current value of the operand temporarily & only after that statement is completed; the value of the operand is incremented.

ex: a=5, a++, a=6
a=5, ++a, b=++a, b→5, a→6 sol?

★ Pre-decrement & Post decrement same work as post & pre decrement.

2. Relational operators

→ Relational operators define the relⁿ bw 2 ele i.e true or false.

3. Logical operators

→ logical operators are used to connect multi expression or conditions together.

→ && → give true if both are true

→ || → give true if either one is non-zero

→ ! → Reverse the logical state. !A is true.

4. Bitwise Operators.

→ Bitwise operator are the operators that operate on bits of perform bit-by-bit operat^n.

- & And
- | Binary OR
- ^ Binary XOR
- ~ Binary ones
- << Binary left
- >> Binary Right shift

5. Assignment operators

- =
- +=
- -=
- *=
- /=

6. Misc Operators.

- sizeof( ) → if a is integer then sizeof(a) will return 4.
- Condition ? 2 : 4.
- Cast → Casting operator converts one data type to another.
- Comma (,).

C++ Strings

→ String are used for storing text.
→ A string variable contains a collection of characters surrounded by double-quotes.
* We have to include a header file for strings i.e #include<string>

* Concatenation

i The + operator can be used to concatenate to add the or join two or more strings

ii We also can use the append() function to concatenate or join string.

NB USE String keyword to make string.

eg
```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName.append(lastName);
cout << fullName;
// output
John Doe.
```

String Length.

→ To get the length of the string we use length() or size() function

Access string.

→ We simply Access the element of string like we access the array from 0 index pp n-1.

C++ Function

→ The function in C language is the block of code.
→ To perform any task we can create function.
→ A function can be called many times.

* Advantages of function in C.

i. Code reusability.
ii. Code optimization.

* Types of functions.

→ There are two types of function in C programming
i. Library function → The funct° which are already defined in C++ language.
ii. User-defined functions → The function which are created by the user, to use many times.
→ It reduces complexity of a big problem & optimizes the code.

* Declaration of function.

declaration {
return type function name (data type parameter)
{
// code to be created
};

int main
{
function name (); // funct° call
}
initialization

There are two ways to pass value or data to a function

i. Call by Value

ii. Call by Reference

Key point: Original value is not modified in call by value, but it is modified in call by reference

#Call by Value

Original value is not modified.
Value being pass to the function is locally stored by the function parameter in stack memory location.
* STACK is based on LIFO
If you change the value of function parameter, it is changed for the current function only.
Its value is not change in the main() funct°.

Ex:
```
#include<iostream>
void func(int data);
int main()
{
    int data=3
    func(data);
    cout<<data;
    return 0
}
```

```
void func(int data)
{
    data=5;
}
```

Output = 3

Call by Reference.

* → In call by value, orizional value will modified because we pass reference (address).

→ Here, address of the value is passed in function. so actual & formal arguments share the same address. Hence, value changed inside the function, is reflect inside as well as outside the function.

ex:-
```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
    int swap;
    swap = *x;
    *x = *y
    *y = swap;
}
int main()
{
  int x=500, y=100;
  swap(&x, &y);
  cout<<"value of x is:"<<x<<endl;
                "  "  y
  return 0;
}
```
output:
x = 100
y = 500

Difference call by value & call by reference.

Call by Value	Call by reference
i Value passed to the funct^n	i Address is passed to the function
ii change made inside the funct^n is not reflected on other function	ii change made inside the function is reflected outside the funct^n also.
iii Actual & formal arguments will created in different memory allocation	iii. Actual & formal arguments will be created in same memory location.

C++ Recursion

When the function is called within the same function, it is known as recursion.
funct^n which calls the same function is known as recursive function.

Tail Recursion → A function that calls itself, & doesn't perform any task after function calls, is known as tail recursion.

ex: recursionfunction() {
    recursionfunction(); // calling self function
}

C++ Storage Classes.

→ storage class is used to define the lifetime & visibility of a variable and/or function within a program.

→ lifetime: refers to the period during which the variable remains active & visibility refers to the modules of a program in which the variable is accessible.

There are five types of storage classes.
1. Automatic
2. Register
3. External
4. Mutable
5. Static

Automatic Storage class.

The auto keyword provides type Inference capabilities, using which auto can deduct the data type of an expression in a programming language can be done.

This Consume less time having to write out things the compiler already knows.

As all the types are deduced in compiler phase only.

The time for compilation increase slightly but it does not affect the run time of program.

It is default storage for all local Variable.

Register Storage class

The register variable allocates memory in ~~mem~~ register than RAM.

Its size is same of register size.

It has faster accust than other variable.

Note: We can't get the address of register variable

ex: register int a = 0;

↳ keyword.

its store in or register only if only if space is free, if not then in memory

Works faster as compared to memory.

Static

→ Static variables have a property to preserving their value even after they are out of their scope.

→ Static variables preserve the value of their last use in their scope.

→ We can say that they are initialized only once & exist until the termination of the program.

→ Thus, no new memory is allocated because they are not re-declared.

→ Their scope is local to the function which they defined.

Note: By default, they are assigned the value 0 by the compiler.

Extern

→ Extern storage class simply tell us that the variable is defined elsewhere & not within the same block.

→ Basically, the value is assigned to it in a different block & this can be overwritten/changed in a different block as well.

ex: extern int counter=0;

Mutable

→ Sometimes there is a requirement to modify one or more data members of class/struct through const function even though you don't want to this funct" to update other members of class/struct.

When we declare a function as const, the pointer passed to funct" becomes const. Adding mutable to a variable allows a const pointer to change member.

ex:
```
#include<iostream>
using namespace std;

class Test {
public:
    int x;
    mutable int y;
};

Test()
{ x=4;
  y=10;
}
};
int main()
{
const Test t1;
t1.y=20
cout<<t1.y;
return 0;
```