C++

- Bjarne Stroustrup (1979)

→ Fast programs, More control over system resources
→ Memory management
→ High performance.

* Header Files
① System defined header files : Defined in the compiler itself.
   # include <iostream>
② User defined header files : Made by the programmers

* Variable Scope [Global Variables   cout << :: variable_name
  (To use a global variable in main ())
  Local Variables

* 34.4f is a float.
  34.4 is a default double data type in C++

* Reference Variables
```
int x = 40 ;
int & y = x ;
cout << y ;
```
$\xrightarrow{}$ am %

O/P $\underline{\underline{40}}$.

* cout $\boxed{<<}$ → Insertion Operator
  cin $\boxed{>>}$ → Extraction Operator

* Typecasting : Converting, a variable from one data type to another.
```
float b = 45.46 ;
cout << (int) b ;
```
O/P $\underline{\underline{45}}$.

* transform () $\longrightarrow$ upper to lowercase string.

* constants
Eg. const int a = 10 ;

* Manipulators :
① << endl ;
② setw (4) : set width (Output mai width set karta hai)
etc.

(BODMAS)
* Operator Precedence. Imp

* Control Structures
① sequence structure
② selection Structure – If-else , If-else ladder, Switch-case
③ Loop Structure – For, while , do-while

* Pointer
A pointer is a variable whose value is the address of another variable i.e. direct address of the memory location

```
type  *var-name ;
 int main() {
   int *b ;
   int a = 3 ;
     b = &a ;
```

* & ---> (Address of) operator

 * ---> Dereference operator (value at)

```
cout << *b            o/p   3//  { bcoz value at *b is 3 }
cout << b or &a       o/p   0x126 ~ { Address }
```

* structures, unions & Enums

SYLLABUS

- ✓ "Hello, World!"
- ✓ Input & Output
- ✓ Basic Data Types
- ✓ Conditional statements
- ✓ For Loop
- ✓ Functions
- ✓ Pointers
- ✓ Arrays Introduction
- ✓ Strings
- ✓ Structs, Unions & Enums
- ✓ OOPS [CLASSES N OBJECTS]
- ✓ OOP Concepts
- Smart Pointers
- Exception handling
- ✓ I/O and streams
- Standard Template Library (ST
- ✓ Operators

* Recursive Functions

~~int main() {~~ ~~return~~
~~int recursion (n){~~
~~if~~

* Functions calls itself directly or indirectly.
* Components → Base condition
  → Parallel logic
  → Recursive call

```
int factorial (int n){
    if (n<2){
       return 1;
    }
  else return n * factorial(
}
```

```
arr [] {1,2,3,4,5};

int length = sizeof(arr) / sizeof(arr[0]);
```

4 byte × 5 = 20 byte

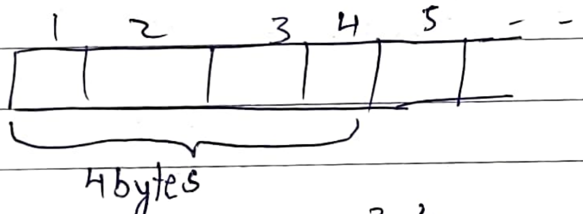
* Data types

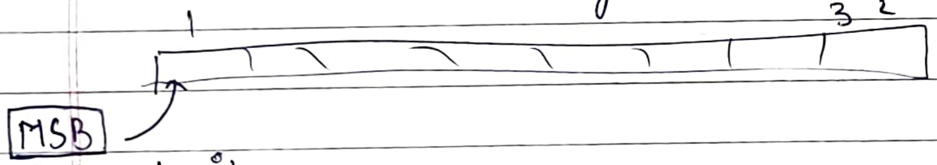
PRIMITIVE		DERIVED	USER - DEFINED
1 Integer	4 byte = 32 bit	Function	Class
2 Float	4 byte = 32 bit	Array	Structure
3 Character	1 byte	Pointer	Union
4 Boolean	1 byte	Refrence	Enum

★ 1 byte = 8 bits

⇒ Integer (signed (-ve) (1), Unsigned (+ve) 0)
(1) (0)

Memory block





Most significant bit

char 'a' [ASCII Value = 97]

★ Type Modifiers
Modifier is used to alter the meaning of the basic type so that it more precisely fits the need of various situations.

Ex: signed, unsigned, long $ short.

#→ Preprocessor directive
used to include files.

* BINARY NUMBER SYSTEM

Binary to Decimal

1	0	1	1	0	1
2^5	2^4	2^3	2^2	2^1	2^0

$2^5 + 0 + 2^3 + 2^2 + 1$

$32 + 8 + 4 + 1 = 4$

64
16
80

Decimal to binary.

45 ⇒

64	32	16	8	4	2	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1	0	1	1	0	1

$45 - \underset{\checkmark}{\underline{32}} = 13 - \underset{\checkmark}{\underline{8}} = 5 - \underset{\checkmark}{\underline{4}} = 1 - \underset{\checkmark}{\underline{1}} = 0$

$\boxed{45 \Rightarrow 101101}$

1y1

* Subarrays : is a contiguos part of an array.

$\{\underbrace{1, -4, 7}, 2\}$

① $= \underbrace{1, -4, 7}_{\checkmark}$ ② $= \underbrace{1, 7, 2}_{\times}$

* Structures, Unions & Enums
↓
→ It is a user defined data type in C++. A structure creates a data type that can be used to group items of possibly different data types into a single type.
Basically like classes 'n' objects

typedef

```
struct employee {
    /* Data Members */
    int eId ;
    char ~ ;
    float ~ ;
} ep ;
```

→ Union is for better memory management. Though we can only use any one of the data member.

```
enum Meal { breakfast, Lunch, Dinner } ;
               ↓        [1]     [2]
              [0]
```

⊛ PENDING WORK: VID NO. - 16-17-18-19 | 23-28

* Call by Value / Call by reference.

Page No.
Date:

OBJECT ORIENTED PROGRAMMING
IN C++ (C with classes)

Diff b/w structs & classes is that, classes have public & private access modifiers.

Ex:
```
class Geeks {
    public :

    string geekname ;                          // Data Member

    void printname()                           // Member Function
    {
        cout << " Geekname is :" << geekname ;
    }
};

main() {

  Geeks obj1 ;                        // Declare an object of class geeks

  obj1 . geekname = "RITWIK" ;        // Accessing data member

  obj1. printname () ;                // Accessing member function.

  return 0;
}
```

OOPs

```
class — {
    Data Members;
    Member Functions;
}
```

→ Encapsulation:
Hiding "sensitive" data from the user

⇒ Access Modifiers:

	Access in Own Class	Derived class	Outside the class
1. Public	✓	✓	✓
2. Private	✓	×	×
3. Protected	✓	✓	×

Pros:
- Good coding practise, useful in interviews.
- Increased security of data.

→ Inheritance
It is possible to inherit attributes and methods from one class to another.

① Derived class (child) - the class that inherit from another
② Base class (parent) - the class being inherit from.

* Types :-
1. Single
2. Multiple
3. Multi level
4. Hybrid
5. ~~Hex~~ Hierarchical