# C++ IN A NUTSHELL

1. C++ ⇒ Bjarne Stroustrup 1979 (low level)
   - C with classes (extension of C)
   - C++ fast programs, more control over system resources + memory management.
   - High performance

   2011 → C++11
   2014 → C++14
   2017 → C++17

2. Basic structure of C++ programs

```
#include <iostream>          → Header File : Defined in compiler
using namespace std;

int main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

- `cout` → Used to display output (object of class ostream)
- `<<` → Insertion Operator (`>>` = Extraction Operator)
- `"Hello World!"` → string
- `endl` → Manipulators → Used to end the line
- `main()` → main function of C++ program, always returns an integer value.
- `{ }` → curly braces are used to define the body of the function. Inside it the instructions are written and terminated with a semicolon (;).
- `#` → Preprocessor directive

## ③ Variables & Comments

↓

containers for storing data values. { type variable = value ; }

Data types ⇒ int 123 or -123
double 19.99 or -18.76
char 'A' or 'a'
string "Hi" or "Bye"
bool true or false
float 17.8 or 3.6

```
// This is a single line comment
/* this
is a
multi line
comment */
```

Variable scope → It is the region in code where the existence of variable is valid.

- Local Variables : Declared inside a function can only accessed there.
- Global Variables : Declared outside " " can access anywhere.

(:: → scope resolution operator used to print value of global variable)

DataTypes continued ⇒

| PRIMITIVE | | DERIVED | USER - DEFINED |
|---|---|---|---|
| int | 4byte = 32 bit | Function | Class |
| float | 4byte = 32 bit | Array | structure |
| char | 1 byte | Pointers | Union |
| bool | 1 byte | Reference | Enum |

★ 1 byte = 8 bits

* In C++ sequence of bytes corresponding to input and output are commonly known as streams.

(4) Operators
⇒ Arithmetic ( + , - , * , / , % , ++ , -- )
⇒ Relational ( == , != , > , < , >= , <= )
⇒ Logical ( && , || , ! )
⇒ Bitwise ( & , | , ^ , ~ , << , >> )
AND OR XOR LEFT SHIFT OPERATOR
COMPLEMENT RIGHT SHIFT OPERATOR
⇒ Assignment ( = += -= *= /= %= <<= >>= &= ^= |= )
⇒ Misc Operators

(5) Reference Variables & Typecasting

```
int x = 455;
int &y = x;        then   x = y = 455
```

```
float b = 45.56;
cout << int(b);          o/p  45.
```

(6) Constants, Manipulators & Operator Precedence
↓
Literals ⇒ Ex: const int a = 3;

Manipulators ⇒ Modify the iostream
1) endl, ws, ends, flush
2) setw(), setfill(), setprecision(), setbase()  [Main with arguments]

() ①
*/ ②
+- ③

## (7) Control structures in C++

① Sequence structure : Statements are executed in the same order in which they are specified in the program.

② Selection structure :

if (condition == true) {

}

else {

}

if-else if ladder  switch-case

③ Loop structure : for, while, do, do-while.

* Break & continue statement.

## (8) Pointers

It is a variable whose value is the address of another variable.

type *var_name ;

```
int a = 3 ;
int *b = &a ;
```

Though, the value of b == &a == 0x1732ff... (Address)

cout << *b ⇒ o/p 3  Dereference operator * → value

cout << b << &a ⇒ o/p 0x1732ff...  address of operator. & → address

Pointer to pointer

```
int **c = &b ;
```

## (9) Arrays & Pointers Arithmetic

Collection of items of similar types stored in contiguous memory locations.

```
int marks[10] = {10, 20, 30, 40, 50};

int *p = marks;
```

## (10) Structures, Unions & Enums

Structures:

It is a user defined data type, used to group items of possibly different types into a single type.

```
struct student {
    string name;
    int age;
    bool gender;
};
```

* can also use typedef (st)
  ↓
  Keyword to assign alternative name.

## (11) Union:

Like structures, union is a user defined data type.

* In union all members share the same memory location.

union

* Size of a union is taken according the size of largest member in union.

Enum: It is a user defined data type which can be assigned some limited values.

```
enum Meal { breakfast, lunch, dinner };
            0          1       2
```

## (12) Functions & Function Prototypes

type function_name (parameters/arguments ...) {

}

* The value which is passed to a function inside main() are arguments.

## (13) Call by value & Call by reference

Call by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function.

Ex:

```
void swap (int x, int y) {
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

THIS WILL NOT WORK

To avoid this call by reference method is used, which copies the reference of an argument into the formal parameter.

Ex:

```
void swap (int &x, int &y) {
    int temp;
    temp = x;
    x = y
    y = temp;
}
```

* Short Hand If... else (Ternary Operator)

variable = (condition) ? expressionTrue : expressionFalse;

## (14) Inline Functions, Default & Constant Arguments

Used for only small function (less no. of bytes / space in memory)
↓ the compiler

Inline functions ∨ Replaces the function call with the corresponding function code, which reduces the OH overhead of function calls.

```
inline int max_num (int x, int y){
    return (x>y) ? x:y ;
}
```

Default Argument ↑

```
float moneyYouwillGet (int currentMoney, float factor = 1.04) {
    return currentMoney * factor ;
}
main() {
cout<<moneyYouwillGet (1000, 0.01) ;
}
```

* Default arguments should always be in the extreme right (last).

## (15) Recursions & Recursive Functions

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

```
int fib (int n){
  if(n<2) return 1;
  return fib(n-2) + fib(n-1);
}
```

## (16) Function Overloading

In "Function Overloading" function name should be same and the arguments should be different.

## (17) OOPs

Classes Objects → Basic Run time entities

↓

Basic template for creating objects (Blue Print)

⇒ Paradigms ⇒ Abstraction
Encapsulation
Inheritance
Polymorphism

Encapsulation : Binding of data (methods + variables) into single entity.

Abstraction : It means showing only the essential information and hiding the internal details.

| Access Modifiers : | Access in Own Class | Derived Class | Outside the Class |
|---|---|---|---|
| PUBLIC | ✓ | ✓ | ✓ |
| PRIVATE | ✓ | × | × |
| PROTECTED | ✓ | ✓ | × |

* Static Data Members & Member Function

## (18) Constructors

It is a special member function with the same name as of the class.

- It is automatically invoked.
- It is used to initialize objects of its class.

(i) Default
ClassName()

(ii) Parameterized
ClassName(int a, float b) {..};

## (19) Destructor

It never takes an argument, nor does it return any value.
~ClassName() ~(tilde).

## (20) Inheritence

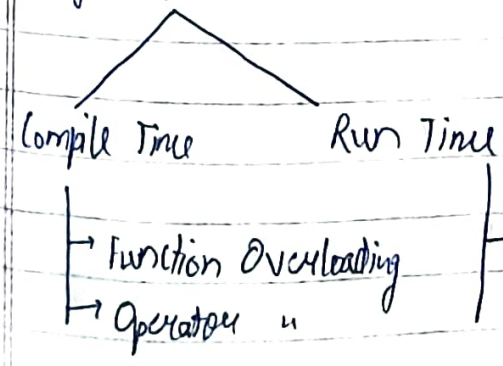
```
class MainClass () {
    ...
};

class InheritedClass : public MainClass {
    ...
};
```

Types :-
(i) Single
(ii) Multiple
(iii) Hierarchical
(iv) Multilevel
(v) Hybrid (Virtual)

## (21) Polymorphism

(*) One name many forms.
→ Function Overloading, Operator overloading
→ Virtual Functions



* int a = 10 ; // Global variable

int main() {
    int a = 15 ; // Local Variable

    cout << a << endl ; // local variable
    cout << ::a << endl ; // Global variable ( scope resolution ).

## (22) Standard Template Library (STL)

set of C++ template classes to provide common programming data structures & functions.

Components of STL ⇒
(i) Containers
(ii) Algorithm
(iii) Iterators

(i) Containers : → stores data {Sequence, Associative, Derived}
→ Use template classes
(ii) Algorithms : → sorting, searching
→ Use template functions
(iii) Iterators : → object points to an element in a container.

* STL is used because its a good idea not to reinvent the wheel.

① Vector

vector < type > vector_name ;

② List (Linked List can be declared by struct, class, vector --)

③ Map

Map is an associative array. [Just like Dict in python]

```
main () {
    map <string, int> MarksMap ;

    MarksMap ["Harry"] = 98 ;
    marksMap ["Jini"] = 59 ;
    marksMap ["Ron"] = 2 ;

    map <string, int> :: iterator itr ;
    for (itr = marksMap.begin() ; itr != markMap.end() ; itr++){
        cout << (*itr).first << " " << (*itr).second << endl ;
    }

    Return 0 ;
}
```

O/P

```
Harry    98
Jini     59
Ron      2
```