# Introduction to C++

## Programming Paradigms

Different approaches to design and structure a program.

Types of programming paradigms:

- Imperative Programming: `Assembly`, `C`
- Procedural Programming: `C`, `Pascal`
- Object-oriented Programming: `C++`, `Java`
- Declarative Programming: `React`, `SQL`
- Functional Programming: `Haskell`

## Difference between `C` and `C++`

> ⚠️ Make sure to go through the course **Problem Solving and Programming in C** from **Semester 1** before reading this

| | C | C++ |
|---|---|---|
| **IO Library** | `stdio.h` | `iostream` |
| **Input** | `scanf()` | `cin >>` |
| **Output** | `printf()` | `cout <<` |
| **Memory Allocation** | `malloc()` | `new` |
| **Memory Freeing** | `free()` | `delete` |
| **Ability to create Objects** | No | Yes |

## `std` and its Usage

`std` referes to standard and all functions and syntax defined in libraries must be used with this. For example: `std::cout`, `std::cin`, `std::endl`

To avoid typing that every time, use this statement at the beginning of the program: `using namespace std`

# Memory Allocation in C++

- `new` : Used to allocate memory
  - `new type;` — Allocate a single memory location
  - `new type[size];` — Allocate a block of memory (for an array)
  - `new type(value);` — Allocate memory + Initialize
  - `new type[size]{v_1, v_2, ..., v_n};` — Alocate memory + Initialize (for an array)
- `delete` : Removes a pointer refering to allocated memory
  - `delete ptr` — Free memory refered to by a pointer (single location)
  - `delete[] ptr` — Free memory refered to by a pointer (block location)

**NOTE:** Always set pointer to `nullptr` after freeing memory.

# Function Overloading

A method to assign multiple functionalities to an identifier name. For example

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

int add(int a, int b); // Adds two numbers
int add(int a, int b, int c); // Adds three numbers

int main() {
    cout << add(5, 7) << endl; // Prints 12
    cout << add(5, 7, 6) << endl; // Prints 18
}

int add(int a, int b) {
    return a + b;
```

```
}

int add(int a, int b, int c) {
    return a + b + c;
}
```

Three ways of function overloading:

- Same return type, different number or type of parameters
- Different return type, same number and type of parameters
- Different return type, different number or type of parameters