Step-In:-

C++ Awareness
- Classes & Objects
- Inheritance
- Virtual Functions
- Operator Overloading

Foundation/Intermediate:-
* Insights on above topics (Reinforce) Cpp Insights
* Templates, STL Containers & Iterators
* Emphasis on code quality
* Unit Testing

Hands-on:- Assignmenets(set 1-4, Coding Tasks)

Specialization/Advanced:-
* C++11/14 additions
* Concurrency & IPC
* Basic design aspects
* More focus on code quality

Reimplement coding tasks using C++11 & 14
Applu C++11 & 14, threads & ipc in Mini projects

Books:-
* Effective C++ by Scott Meyers
* Effective Modern C++ by Scott Meyers
* Clean code by Robert Martin (or) Code Complete by Steve McConnell

C++11/14 additions on language basics:-
* constexpr
* auto type
* decltype
* range based for loops
* static_assert
* nullptr
* Scoped/Strongly typed enums
* strict initializers with {}
* using keyword for aliasing
* user defined literals
* binary literals
* digit separators
* User defined Literals
* Raw string literals

Anatomy:-

https://godbolt.org/

https://cppinsights.io/

https://quick-bench.com/

constexpr ==> .rodata section (ROMability)

observer symbol table using nm/objdump, for constexpr variables

g++ -v

g++ hello.cpp -std=c++11

```
auto sum(auto x,auto y) --> int {

} // in C++11

auto sum(auto x,auto y) {

} // in C++14
```

```
template<typename decltype(sum(a,b)) >
class MyArray {


};

std::map<int, std::string> cities;
for(auto p : cities) {
        //p.first, p.second
}

ptr=nullptr;
if(ptr)       , if(!ptr)

nullptr                    ==> false
not a nullptr              ==> true

assert      ==> runtime error (abnormal termination)
                    if cond is false
```

What is namespace?
How to create namespace?
How to access symbols from namespace?
Usage of "using" keyword
Multilevel/Nested namespace
Default namespace
Anonymous namespace

cpp-essentials-all ==> namespace

Box(int,int,int);

Box(2.3f,4.5f,5.6f);

Box(float,float,float)=delete;

Helper functions

Sample(double);

Sample s1(2.3f);
Sample s2(10);

Sample(float)=delete;
Sample(int)=delete;

```
class Box{
  int l=10;
  int b=12;
  int h=5;
  public:
  Box(int x,int y,int z):l(x),b(y),c(z) {  }
  Box(int x,int y):l(x),b(y) {  }
};

class Sample {                                                    int Sample::k=10;
  const int maxlen;
  static int k=10; //error
  static int k;
  const static int tmax=30;      //allowed in C++98 also
  public:
  Sample(int len):maxlen(len) { }
};
```

```
class Pack {
   int price;
   Box b1;
   public:
   Pack(int p,int x,int y,int z):price(p), b1(x,y,z) {  }
}

class Pack {
  int price;
  Box b1{10,12,5};
  public:
   Pack(int p,int x,int y,int z):price(p), b1(x,y,z) {  }

};

Box(10,12,5)
Box{10,12,5}
{10,12,5}        ==> anonymous object of Box class
```

```
class A  {
  public:
  int sum(int,int);
};
class B {
  public:
  int sum(int,int,int);
};

B b1;
b1.sum(10,12,5);     //ok
b1.sum(10,12);       //error

using A::sum(int,int);       //in class B
//now B objects can call sum with two arguments
```

Skip:-

- Type Traits
- Read only objects

Move Semantics

---

r-value references
std::move
move constructor
move operator=