C++ 11

* Initializer list

eg:-

int arr[4] = {1, 2, 3, 4}

* + C++11 extends initializer list to STL containers.
  + Calls initializer\_list constructor

eg: vector<int> v = {2, 3, 4, 5}

Initializer list can be applied to our own classes.

Eg:

class MyVector

{

vector<int> myVec;

public:

MyVector(const initializer\_list<int>& v)

{

for(auto i:v)

{

myVec.push\_back(\*i);

}

}

};

MyVector myVector = {1, 2, 3, 4};

MyVector myVect1{2, 3, 4, 5}; //effectively the same

* Uniform initialization
  + Aggregate class or struct can be initialized with curly braces enclosed list.
  + Eg: dog d1 = {5, “Jimmy”};
  + Aggregate initialization
  + Order of choice:-
    - Initializer\_list constuctor
    - Regular constructor that takes appropriate parameters
    - Aggregate initialization
* Auto type
  + Automatically infer data type from r-value.
* for
  + eg:- for(int i:v)
  + for(auto i: v)
  + for(auto& i: v) -> To change value while iteration
* nullptr
  + NULL – (int)0
  + nullptr – null pointer
* enum class
  + In C++03, enums are integers
  + eg:- enum class apple{red, green};

enum class orange{big, small};

apple a = apple::green;

orange o = orange::big;

if(a ==o) //produces error.

* static\_assert
  + Make assertions during compile time whereas assert() will do it in run time.
  + static\_assert(sizeof(int) == 4)
* Delegating constructors / In class initialization

eg:

class dog

{

int age = 9;

dog()

{

}

dog(int a) : dog()

{

doOtherThings();  
}

};

* override
  + When defining virtual functions in derived class override keyword is put at the end to specifically tell this function is overridden the base class function, If any change in parameters occur accidentally, C++ compiler throws an error.
* Final – for virtual functions and classes

eg: class dog final

{

};

class dog

{

virtual void bark() final;

};

* Compiler generated default constructor

eg:

class dog

{

dog(int a){}

dog() = default; //force compiler to generate default constructor even there is parameterized constructor

}

* delete
  + Deleting functions

eg:

class dog

{

dog(double) = delete;

};

* constexpr