<https://www.youtube.com/watch?v=wN0x9eZLix4&t=4646s>

Errors- compile time – run time error

C++ - case sensitive

IDE – Itegrated development environment

Iostream – input output stream

Access modifiers – private, public and protected

Object oriented- to represent real life objects and explain them to computer- many attributes or characteristics of the objects like car and methods or behaviour

Class are user defined data type similar to int, float vagera

Class Employee{

}; - semiclone expected

Std::string, a data type from <iostream> library

Everything in class is private by default

Graphical user interface, application

Description automatically generatedText, application

Description automatically generated

Both are same situations in above images

Private - hidden – not accecible outside class – even objects of this class cannot use the private ones.

Class method == function

A picture containing text

Description automatically generated

Std::cout<<

Everytime an object is created a contructor is created- default constructer gives random values to objects- we can create our own constructor

Constructor is a method used to create an object to the class with default values to the data types.

Rules for constructor

Constructor doesn’t have a return value

Constructiore has same name as class

Constructor needs to be public

**Encapsulation** – bundling or tying together the data within the class, to avoid direct access to data to another class

Very specific public functions to allow modification of data – getters and setters

Make data private

Use getters and setters to modify the data

Advantage – we can set validation rules in which the data can be modified- for eg. Set an age limit

Graphical user interface, application

Description automatically generated

**Abstraction –**

Hiding complex things behind the simple functions

There is a class named Abstract class-

**Inheritence-**

Derived class – subtype of a class-

Car -> subset electric car

For employee -> developer



If parent has a user defined constructor child must also have its own user defined constructor

Child has all the properties that parent has and more

Child= derived class

Parent = base class

Derived class constructor-

Graphical user interface, text, application, email

Description automatically generated

Protected – child can access the property, if private it cannot access and getters are used

If the inheritance is public – child object can use the method of the parent class-



**Polymorphism- (many forms)**

When a parent class reference is used to refer to an object of the child class

Graphical user interface, text, application

Description automatically generated

The pointers are defined for the parent class which are assigned the reference of the child objects

-> Is used to access the members of the pointer

The methods work() are the members of the child class

A picture containing text

Description automatically generated

Virtual work method in employee class and their respective implementations in the child classes

STL- standard template library

Containers – prebuilt data structures

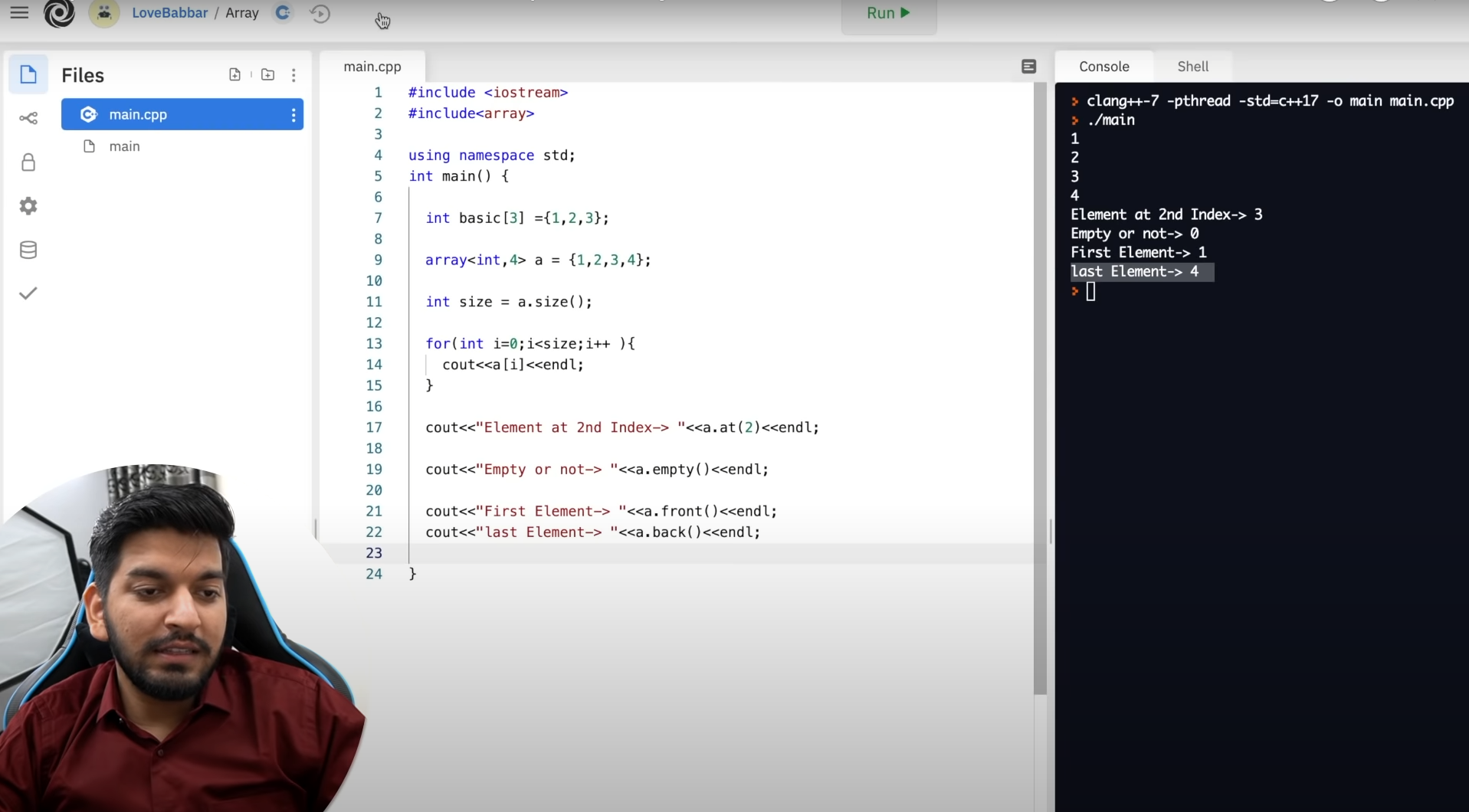
Diagram

Description automatically generated

Array – same data type data stored in it

#include<array>

Stl array is static array



Vector = dynamic array

If vector is full and we try to add more data its size will double

Can change at runtime

#include<vector>

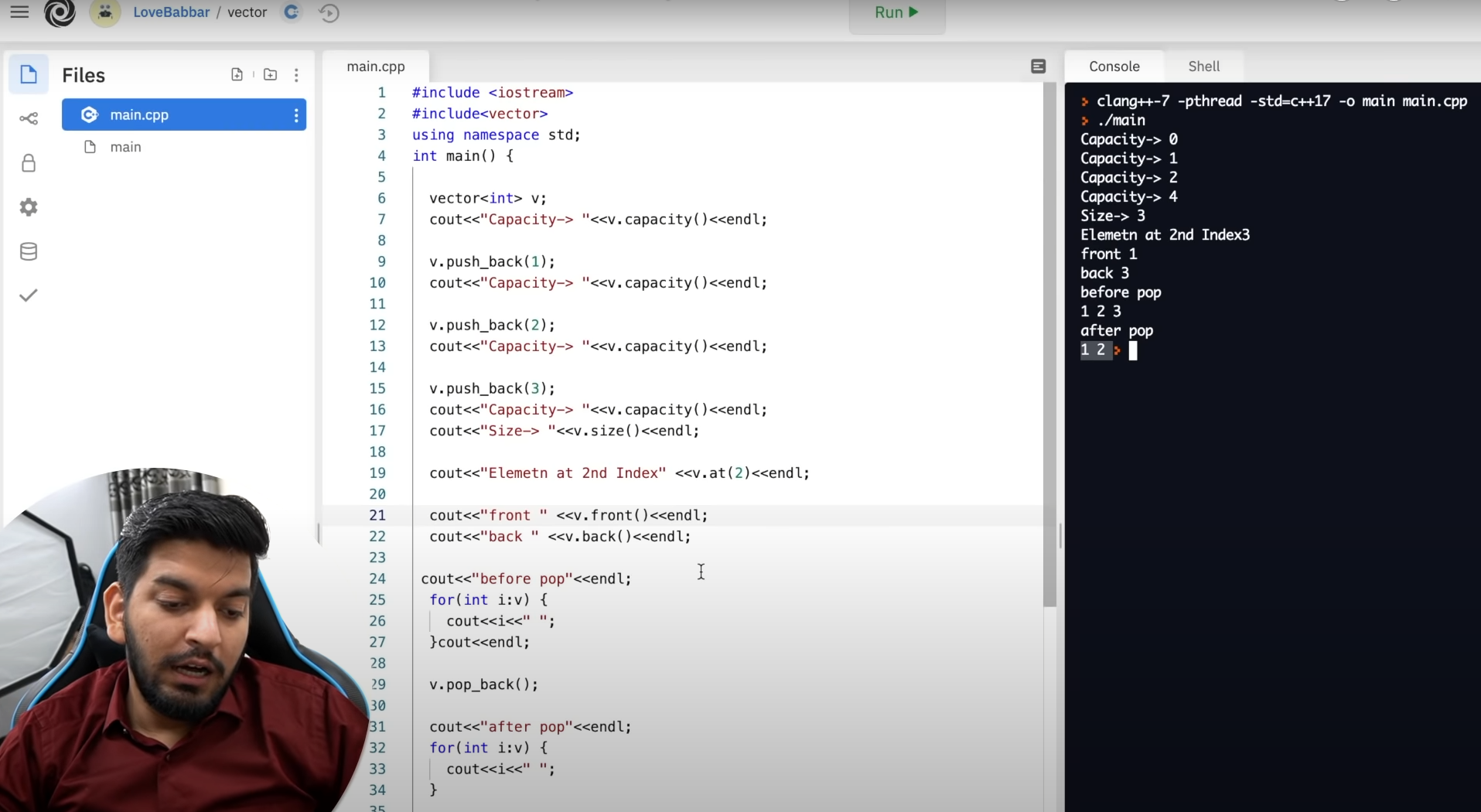
Initial size of vector is 0

v.push to add element

The capacity keeps on doubling

Size= no. of element contained

Capacity = capacity



Vector<int> a(5,1) = [1 1 1 1 1]

Double ended queue- deque – deletion and insertion on both sides

‘at’ operation for random access

#include<deque>

push\_back push\_front pop\_back pop\_front

#include<list>

push\_back push\_front pop\_back pop\_front

stack- we can only access the element that was added at last

last in first out

#include<stack>

s.top – access element

s.pop – gela

s.size, s.empty

#queue – first in first out

q.front – first element

q.pop- first element removed

q.push

Priority queue-

Max-heap – first the greatest element will be accessed

Min-heap – reverse

m.push m.top m.pop

m.empty - returns true or false

Set-

All elements are unique – no element can be repeated – modifications not allowed- only insert and delete- gives output in sorted order

#include<set>

s. insert s.erase

s.count(5) – is 5 present

s.find(5) – returns the interation of the element – reference of the value

Map- data stores in key value

Key = love , value = scorpio

Love will only point to scorpio

Key points to the value

#include<map>

Map = sorted

m.first; m[1] = “babbar”; m.insert({5, “bheem”});

m.erase

Algorithms-

1. Binary search – element present or not,
2. Lower bound – gives iterator ( itr = v.begin = position)
3. int