Software Development

Oops Concepts in C++

$$\begin{tikzcd}
\text{Object} \arrow[r] & \text{entity} \arrow[r] \arrow[dr] & \text{State} \\
 & & \text{behaviour}
\end{tikzcd}$$

eg :

$$\begin{tikzcd}
\text{Paul} \arrow[r] \arrow[dr] & \text{name} \arrow[r] \arrow[dr] & \text{Tekken} \\
 & \text{health} \arrow[dr] & \text{Paul} \\
 & & \text{70.}
\end{tikzcd}$$

→ User Defined Datatype
Class → Hero (eg)

Eg :
```
class Hero ()
        char name [100] ;
properties {     int health ;
        char level;
}
```

→
```
int main (){
    Hero h1;
```

→ cout << "size" << sizeof(h1) ;

}

→ if object is empty the it allocates
1 byte of memory.

→ To access variable in class. from
main function.

Hero h1;
eg: h1. health ();
(need to public)

→ Access Modifiers:

(only access in class) Public (default in struct) :
(Access anywhere) Private (default in class) :
Protected :

→ Setter & Getter Method.
→ in class Hero (){
char getname () {
return name;
}
} get name from private.

(you can also used functions i.e setter & getter)

```
void set_health (int h) {
        health = h;
}
```

to set health in private member from outside

-> in Main fuction.

```
cout << "H1' health :" << h1.gethealth
                                    << endl;
```

=> Dynamic Allocation.

eg: int* i = new int ();

int Main fuction =

```
        Hero* h = new hero();
        cout << "H1' health :" << h->health
                                      << endl;
```

(-> is used when Dynamic allocation is done)

Constructor:

↓

Object create

↓

1) No return type
2) No insert parameter
3) Object creation invoke.

eg: Hero. Ramesh (); by default.
Back side -> [ramesh . Hero ()]

To test default Constructor write in class. ->
```
Hero () {
cout << " Called Constructor"
<<endl;
}
```

-> & just define a variable of Hero in Main function.

-> Parameterized Constructor.

in class ->
```
Hero (int h, String s){
    health = h;
    ~~S~~level = s;
}
```

In main function ->
```
Hero initial ( 90, "Dom" );
cout << initial . health <<endl;
```
(Prints = 90).

→ Copy Constructor -

in class → Hero (const Hero& copy)
: health(copy.health), level(copy.level), name(copy.name) {}

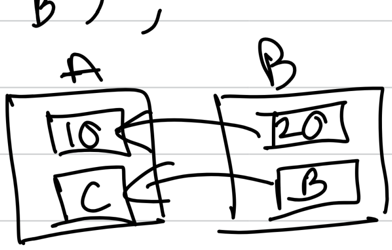
in main function - Hero Copy(h1);
(will print exactly same as h1 values)

h1 = 70 → cout << copy.health << endl
health // 70

→ Copy Assignment Operator :

eg : Hero a (10, 'c');
Hero b (20, 'B');

⇒ a = b;



Destructor : To deallocate heap memory

Eg: ~ Hero () {
cout << "Destructor Called" << endl;
}

(Only Static Allocation has automatic destructor)

→ For Dynamic constructor. → $\boxed{\text{delete b;}}$

How const keyword ?

→ Static keyword

→ make a variable in class-

static int time_to_complete;

Hero | enemy
\ /
Both will same time to complete.

→

Data Members ⟶ No need to access through object.

→ outside class
(datatype ClassName :: fieldName = value;)
int Hero :: timeToComplete = 5;
↙
Scope resolution operator

→ in Main function
→ cout << Hero :: timeToComplete << endl;

// answer is 5.

→ **Static functions :**
↓ ↳ No need to create object.

$\boxed{\text{Can only Access Static members.}}$