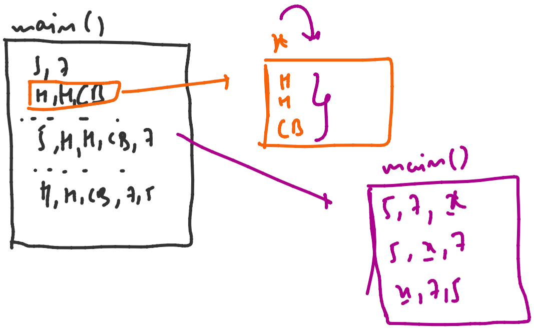
Functions

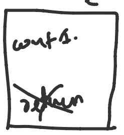
20 June 2023 20:59

They are blocks of code containing some logic/code which can be called/used from anywhere inside program.



4 types of containers that we can make in C++:

void



it wont take any value

[return;]

it won't give/return any value;

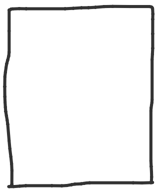
```
void container_name(){
// code.
}
```

pass X

usage → container_name(); → value X

proper syntax of functions / diff types of functions/containers we can make

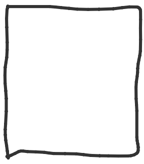
(II) void



it does take some value ✓

X doesn't return any value.

(III)



X doesnot take any value

but returns some value. or a value.

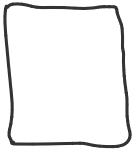
```
datatype fn(){

return value;
}
```

```
int main()
{

return 0;
}
```

(IV)



take some value/values

return a value.

```
datatype fn(params)
{
return value;
}
```

take some value means :- when it is called/ invoked from anywhere inside the program it

take some value means :- when it is called/ invoked from anywhere inside the program it requires some value to be given by that part of program to it, in order to achieve/complete its task.
and these values are called as parameters.

return some value means :- the value, the function/ containing returns/ gives back to the point (from where it was invoked/called) is called return value. And whatever is the type of return value a function gives back, it is the typename of function.

syntax for writing a function.

```
typename function_name(parameter1, param2....
                       ){
    // tasks
    return value
}
```

(int, char, bool, float, double, void)

and the type of this value is called typename

syntax for calling a function.

if return value/type is void.

→ function_name (value1, value2 ....) ;

else.

→ ① cout << function_name(v1, v2---) ;

② datatype/returntype variable = function_name(v1, v2 ...) ;

int int