Date
Page 2/12/2021

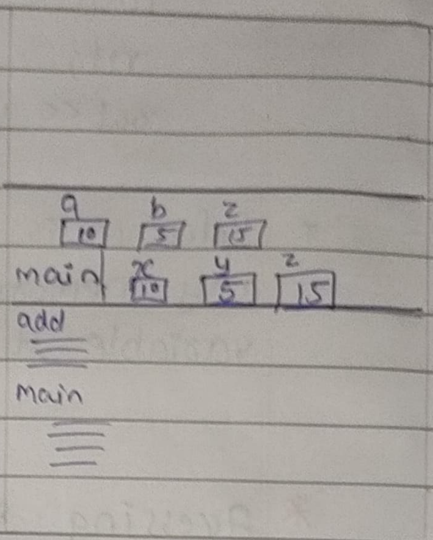
monolithic Program	Modular or procedural program
int main()	func()
{	{ }
—	main()
—	{
	}

→ **Functions**

```
int add ~~main~~ ({int a, int b)   → Prototype or Header of function
                  ↑ formal parameter
{
    int c;
    c = a+b;
    return(c);
}
```

definations of function

```
int main ()
 {
    int x, y, z;
    x = 5;
    y = 10;
    z = add (x, y);     ~~Actual parameters~~
    print(z); → 15
 }
```



→ function variable dies as soon as controls returns from function

*P → Dereferencing.
Pointer takes 2 byte.
reference dont take any memory.

Date
Page 2/12/2021

→ Parameter passing.

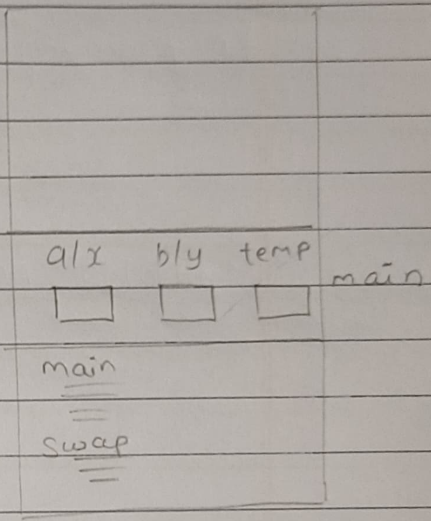
~~(suitable for returning the result)~~
call by value: formal parameter does not affect actual parameter

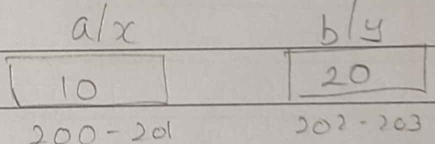
~~void~~ ~~(suitable for returning more than 1 variable or changing actual parameter)~~
call by address: Any change in formal parameter affect actual parameter.

c++
call by reference:- although the source code in procedural the program is monolytic while compiling becz as soon as swap is called it becomes part of main funct^n

```
void swap (int &x, int &y)
 {
     int temp;
     temp = x;
     x = y;
     y = temp;
 }

int main()
 {
     int a, b;
     a = 10;
     b = 20;
     swap (a,b);
     print (a, b)
```


a/x
b/y
temp
main
main
swap


a/x
10
200 - 201
b/y
20
202 - 203