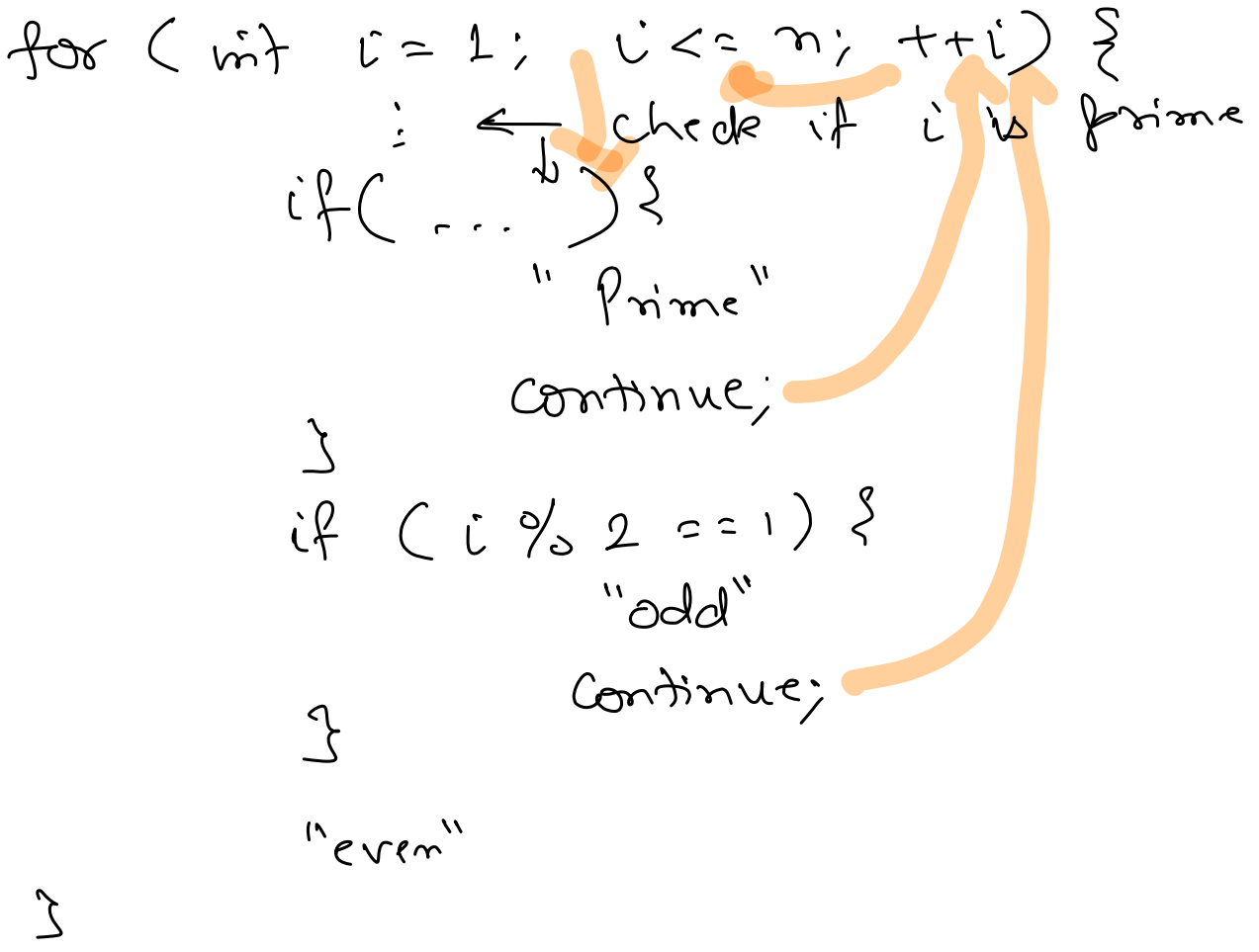
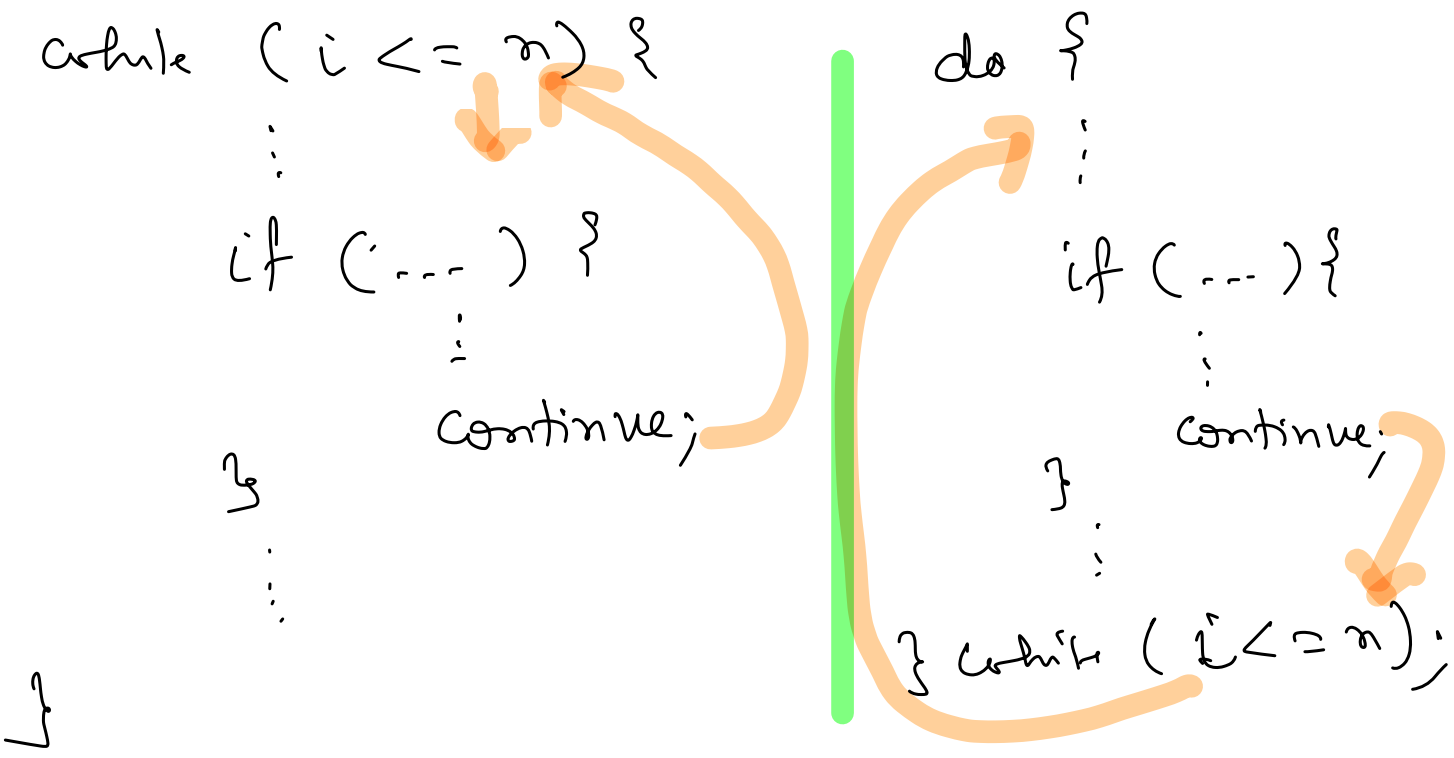
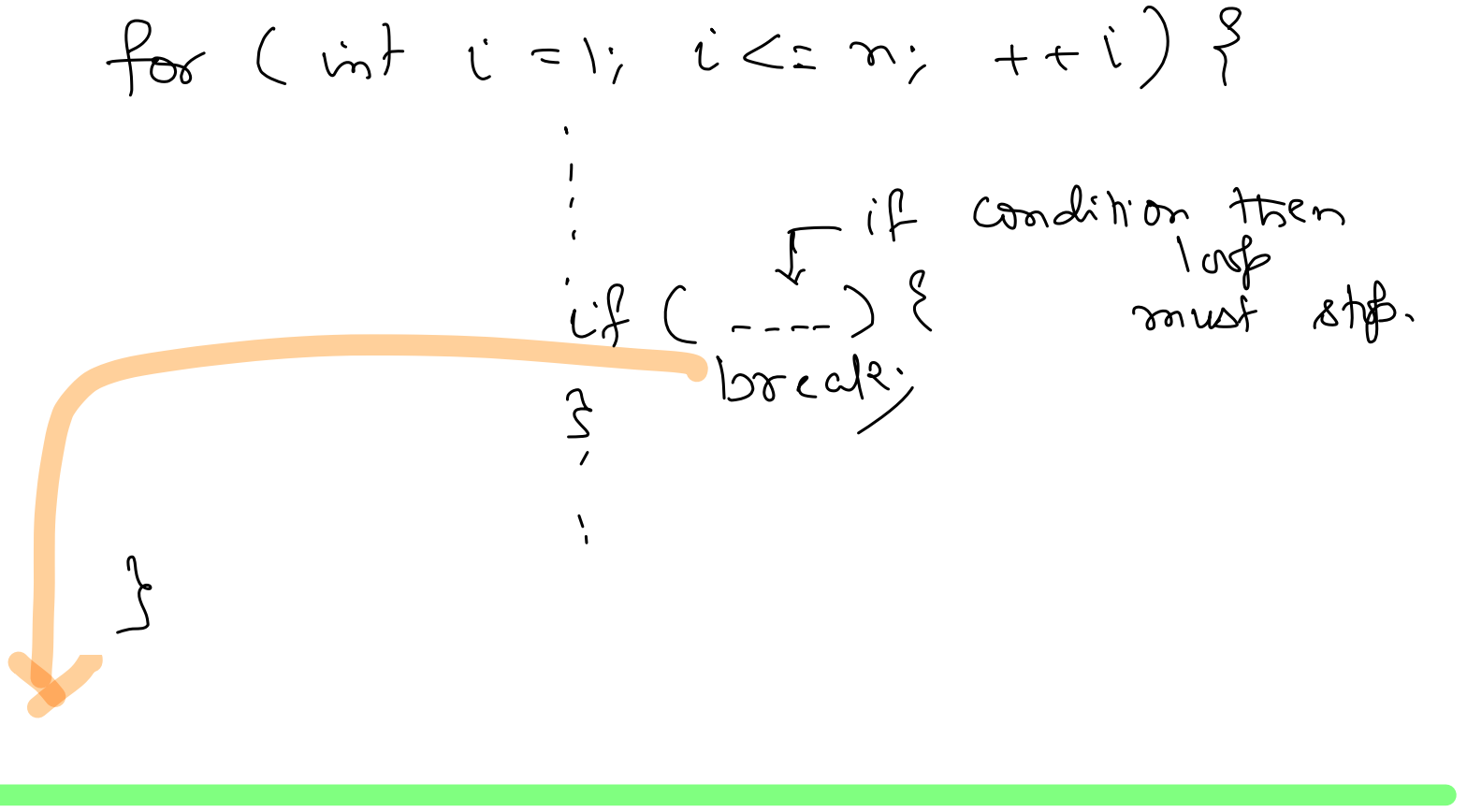
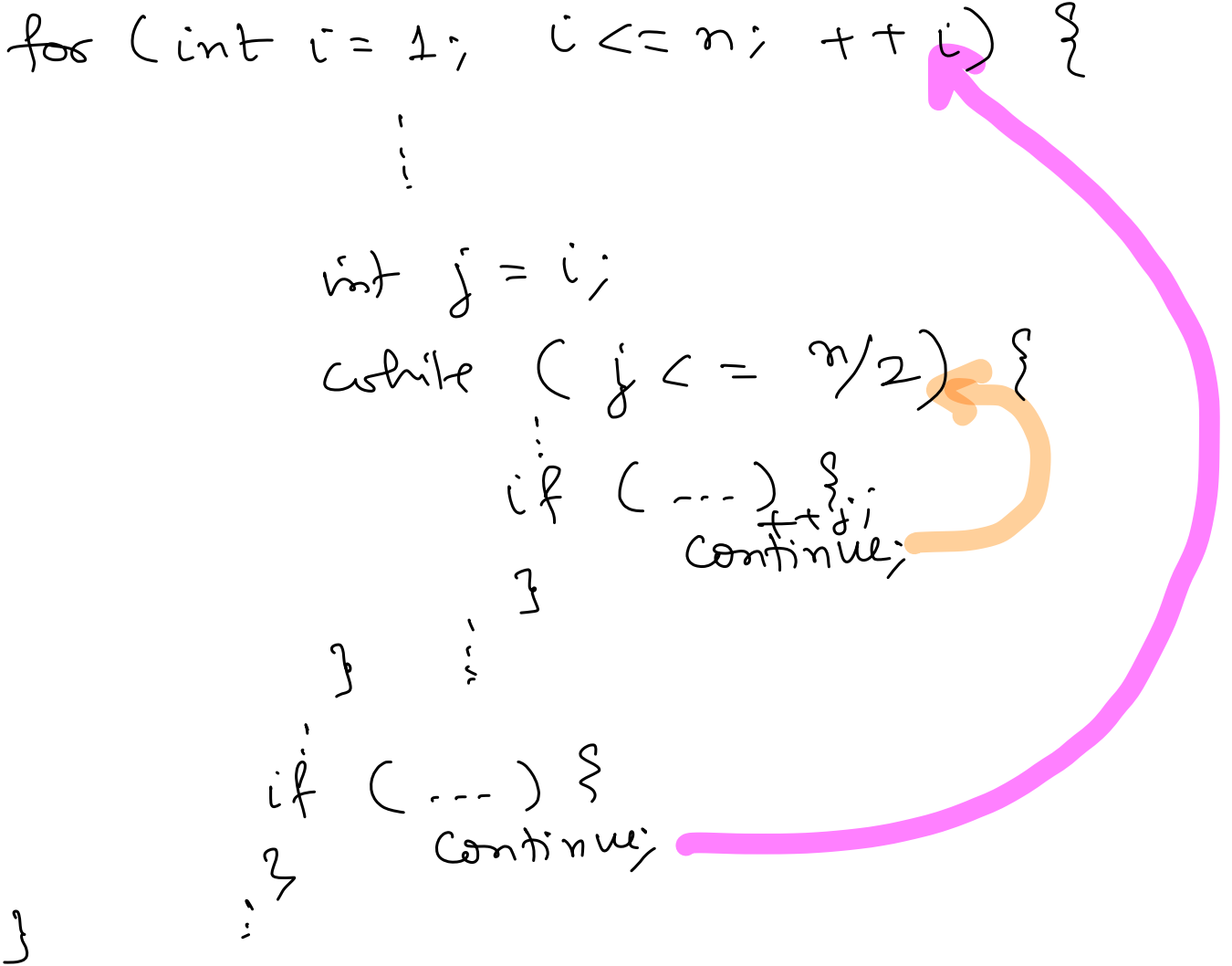
break continue

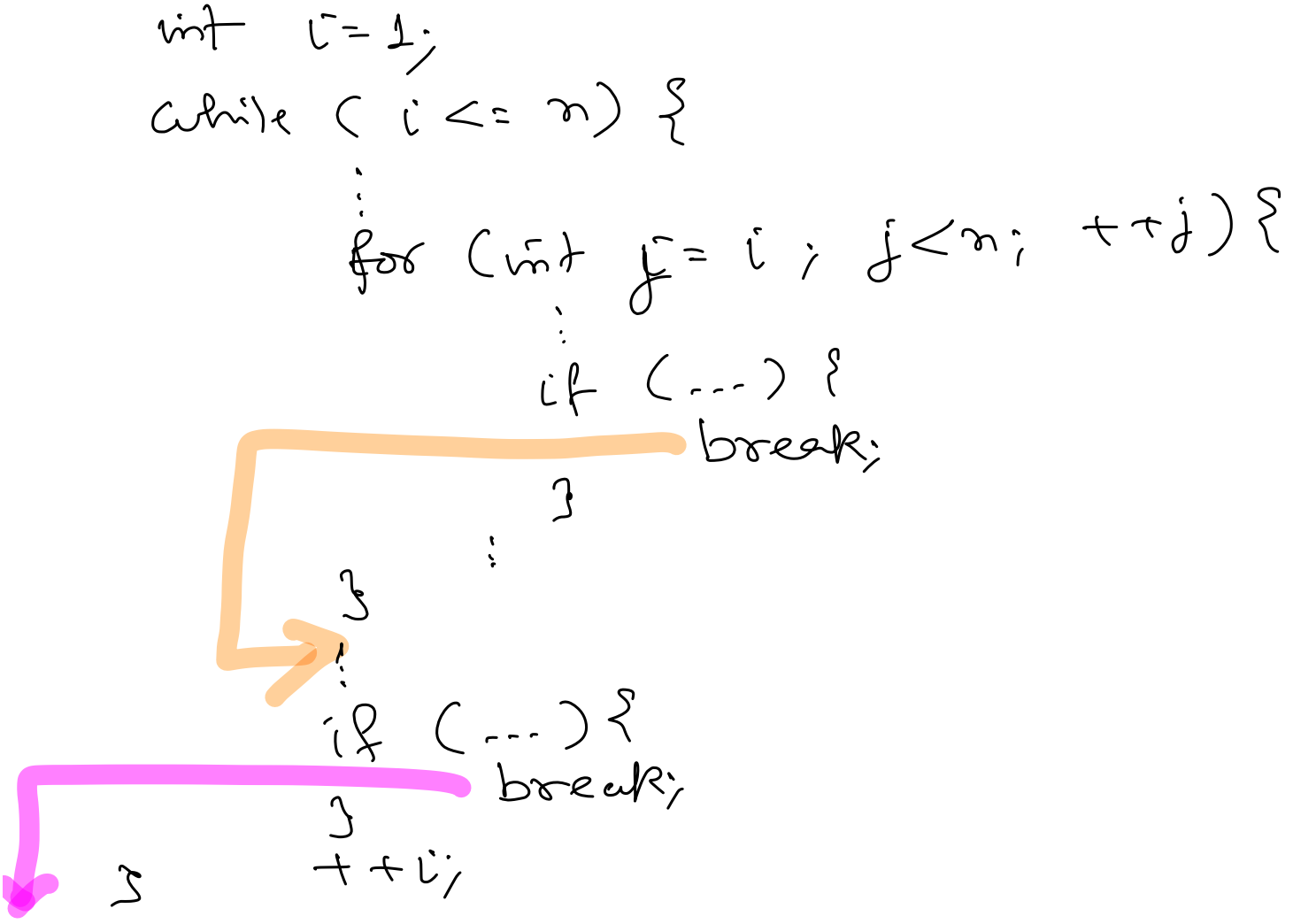
```
for (int i = 1; i <= n; ++i)
    : <- check if i is prime
    if ( .... ) {
                "Prime"
    }
    else {
        if ( i % 2 == 1){
                "odd";
        }
        else {
                "even";
        }
    }
}
```


for (int i = 1; i <= n; ++i) {
check if i is prime
if(...) {
" prime"
continue;
}
if (i % 2 == 1) {
"odd"
continue;
}
"even"
}

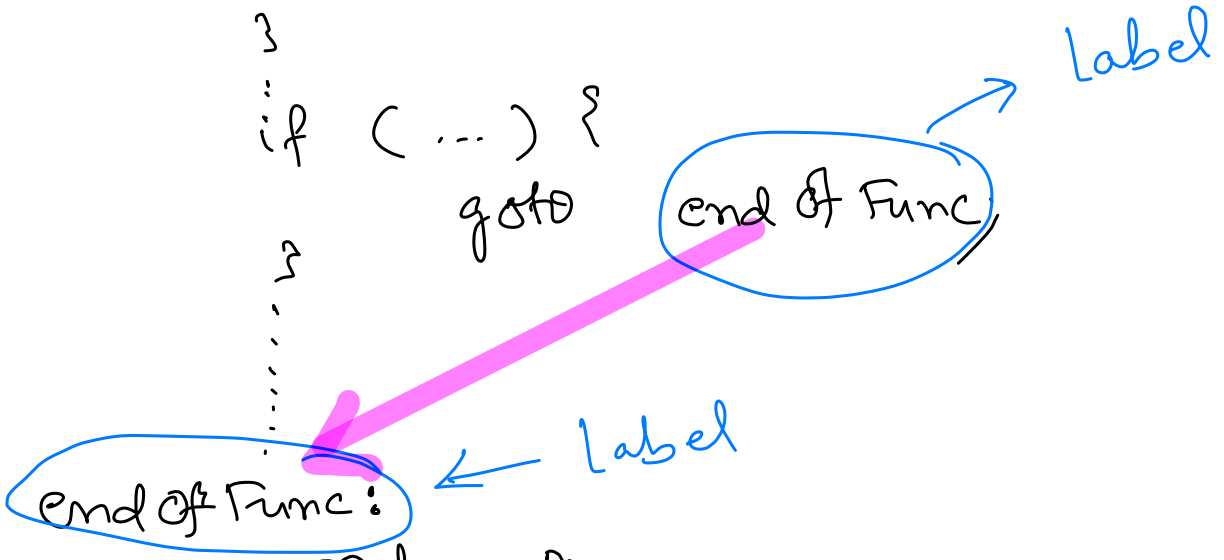

while (i <= n) {
if (...) {
continue;
}
}
do {
if (...) {
continue;
}
} while (i <= n);


for (int i = 1; i <= n; ++i) {
if (----) {
break;
}
}
if condition then
loop
must stop.

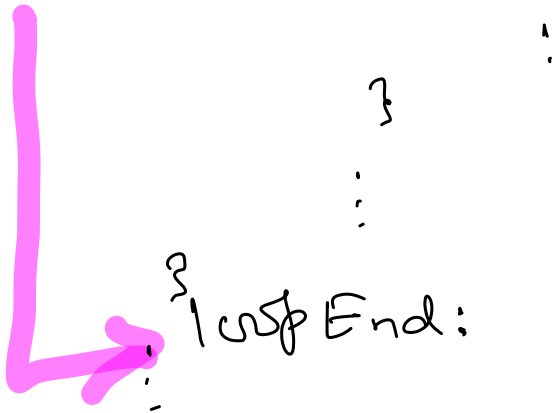

for (int i = 1; i <= n; ++i) {
int j = i;
while (j <= n/2) {
if (...) {
++j;
continue;
}
if (...) {
continue;
}
}


int i = 1;
while (i <= n) {
for (int j = i ; j<n; ++j) {
if (...) {
break;
}
}
if (...) {
break;
}
++i;
}

```
int main() {
    int i = 1;
    :
    while (i <= n) {
        :
    }
    :
    if (...) {
        goto endOfFunc;
    }
    :
    :
endOfFunc:
    return 0;
}
```



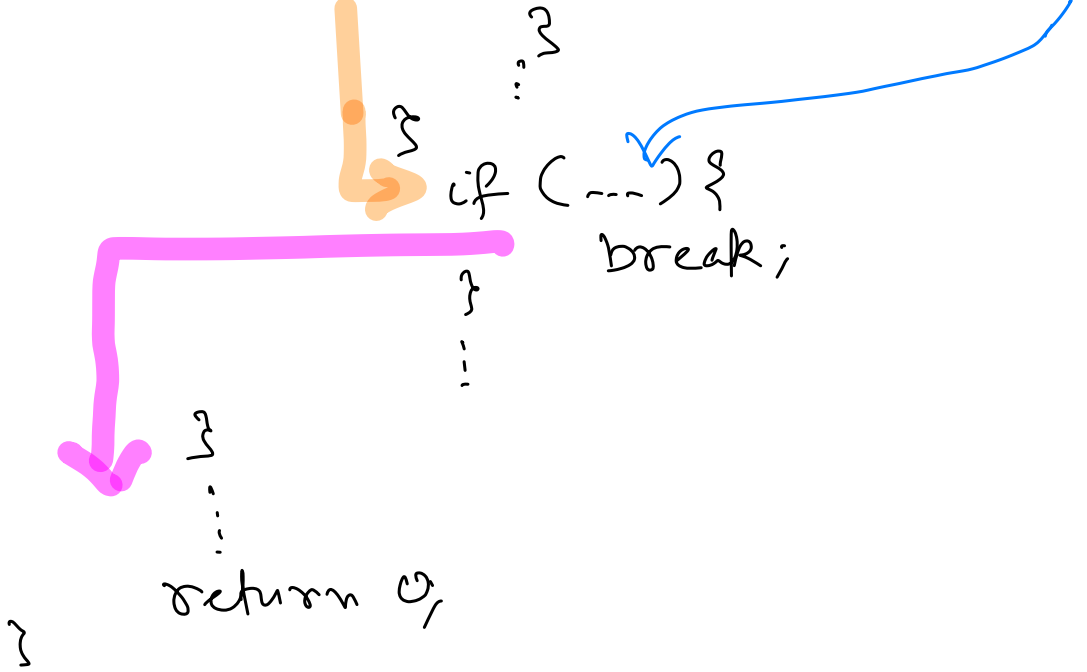
```
int main() {
    ⋮
    for (int i = 0; i <= n; ++i) {
        ⋮
        for (int j = i; j <= n; ++j) {
            ⋮
            for (int k = j + 1; k <= n; ++k) {
                ⋮
                if ( ... ) {        ← if condition true then end all loops.
                    goto loopEnd;
                }
                ⋮
            }
```



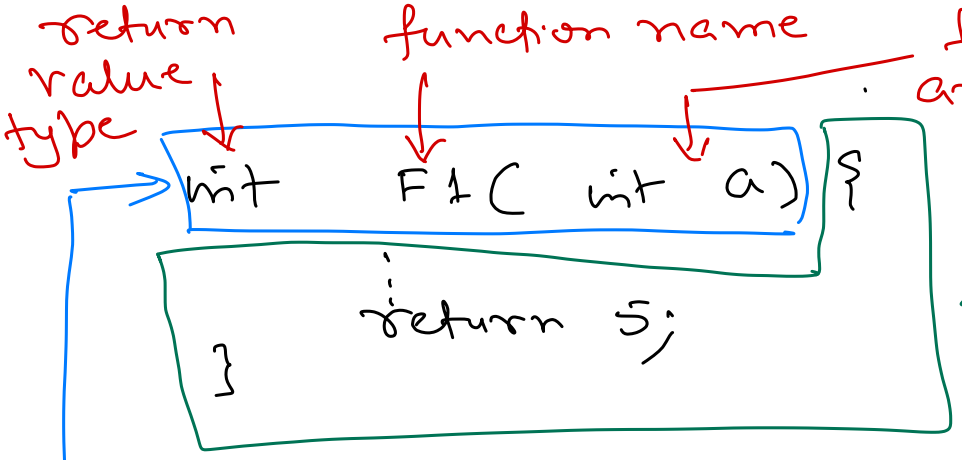
```
    }
    :
}
loopEnd:
:
return 0;
}
```

```
int main() {
    for (int i = 0; i <= n; ++i) {
        for (int j = i+1; j <= n; ++j) {
            :
            if (...) {
                break;
```

condition to end all loops.


}
if (...) {
break;
}
}
return 0;
}

Functions




```
bool isPrime (int n0);
int main() {
    int n;
    std:: cin >> n;
```

function prototype

Print all prime numbers between 2 to n.

```
for (int i = 2; i <= n; ++i) {
    if (isPrime(i)) {
        std::cout << i << " isPrime\n";
    }
}
return 0;
}

bool isPrime(int n0) {
    for (int i = 2; i <= n0/2; ++i)
        !
    return true;
}
```

function name

function call operator

pass argument to function, if required.

Calling a function

```
bool isPrime(int n0) {
        for (int i = 2; i <= n0/2; ++i)
                ;
        return true;
}

int main() {
    int n;
    std:: cin >> n;
```

Print all prime numbers between 2 & n.

```
for (int i = 2; i <= n; ++i) {
    if (isPrime(i)) {
        std::cout << i << " isPrime\n";
    }
}
return 0;
}
```

function name

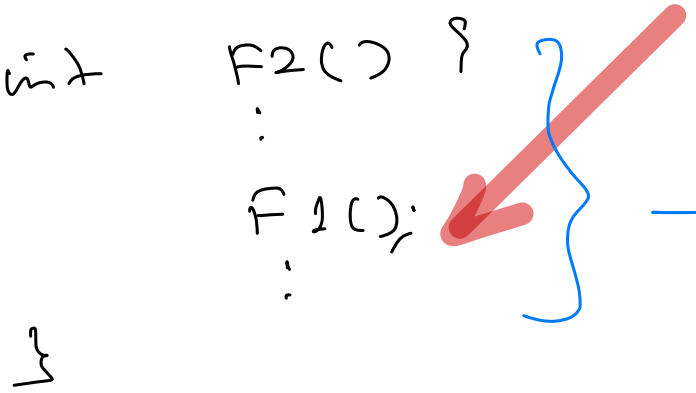
function call operator

pass argument to function, if required

Calling a function

```
int F1() {
    :
    F2();
    :
}
```

which function to be implemented first?



→ function defination

function header

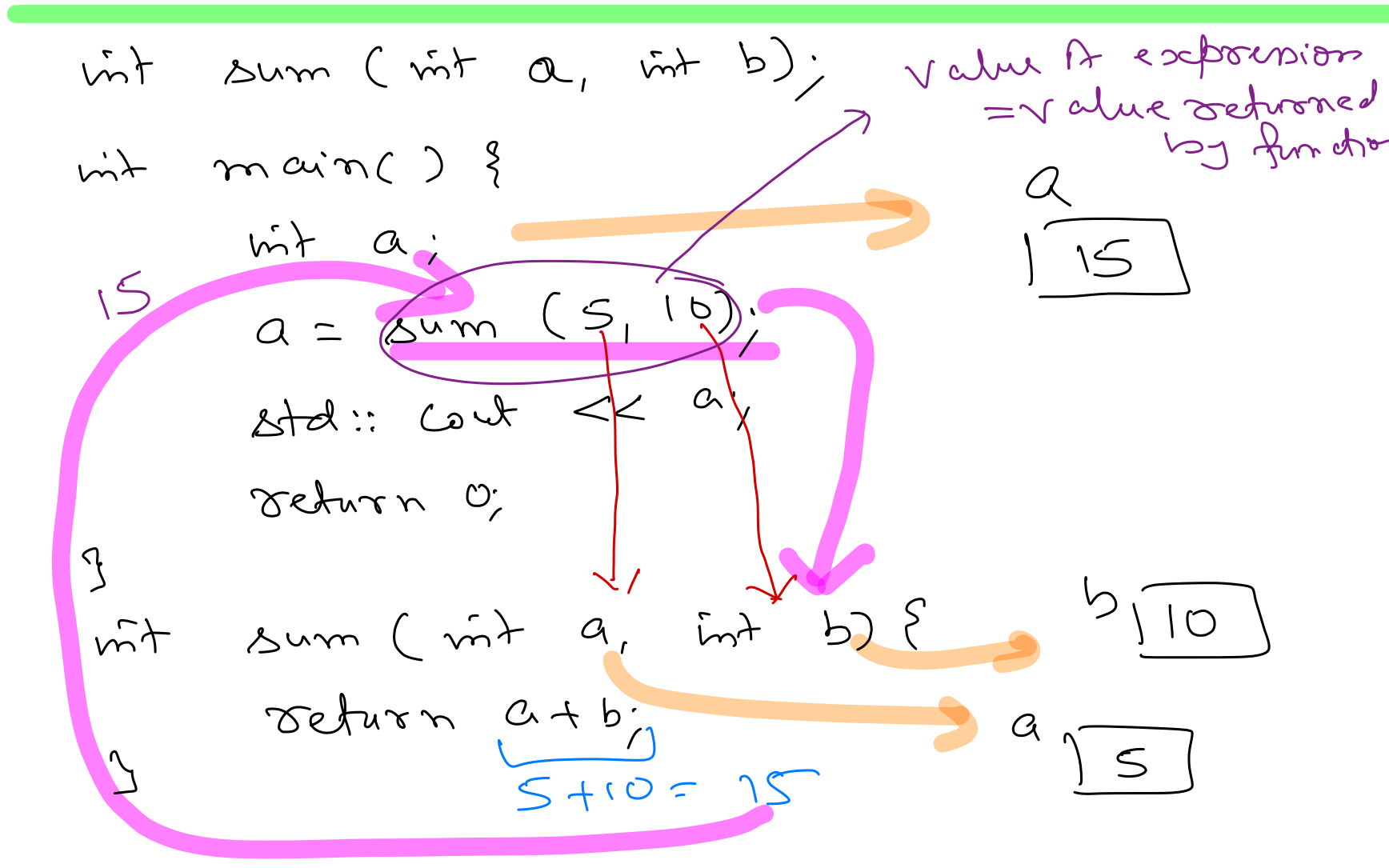
function body

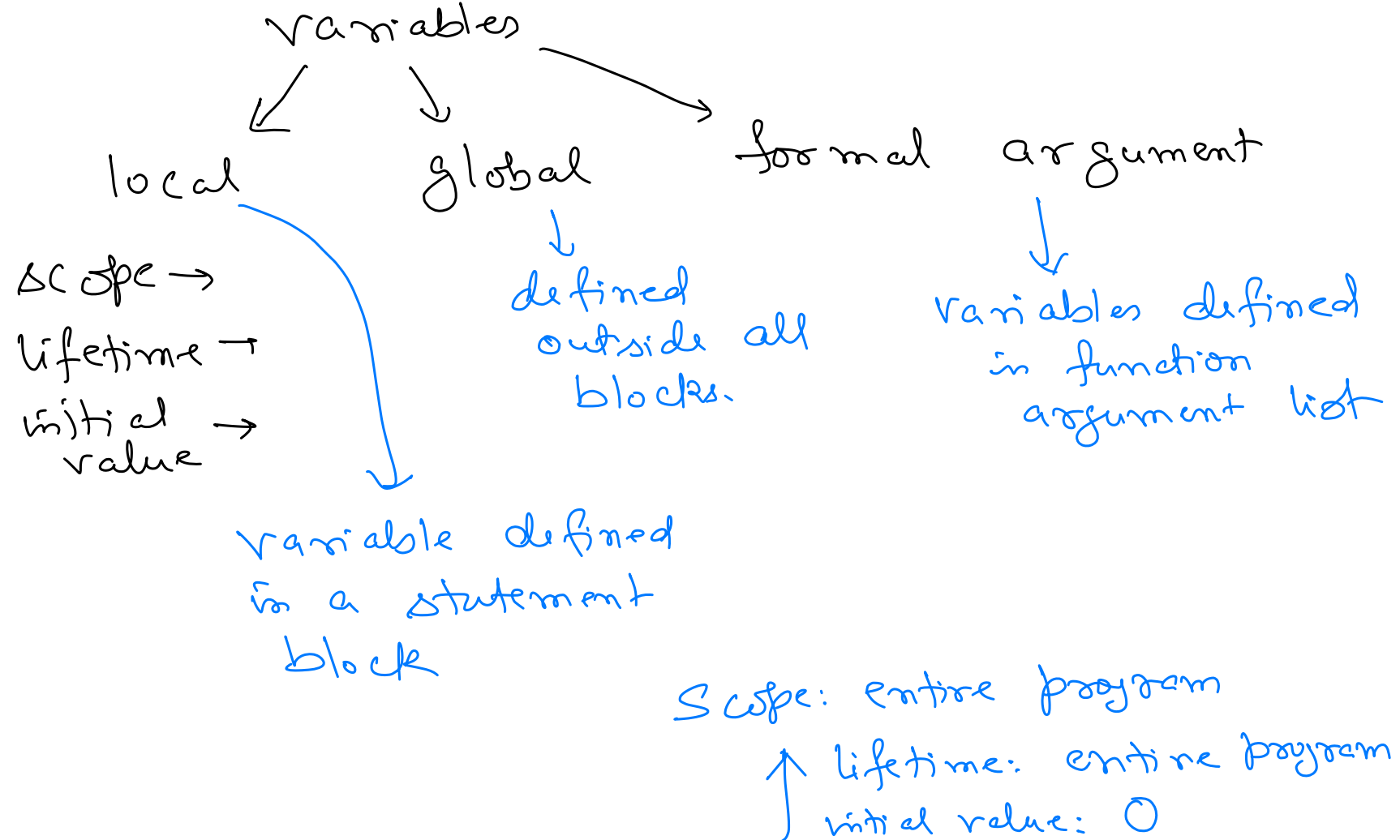
function declaration → just function header

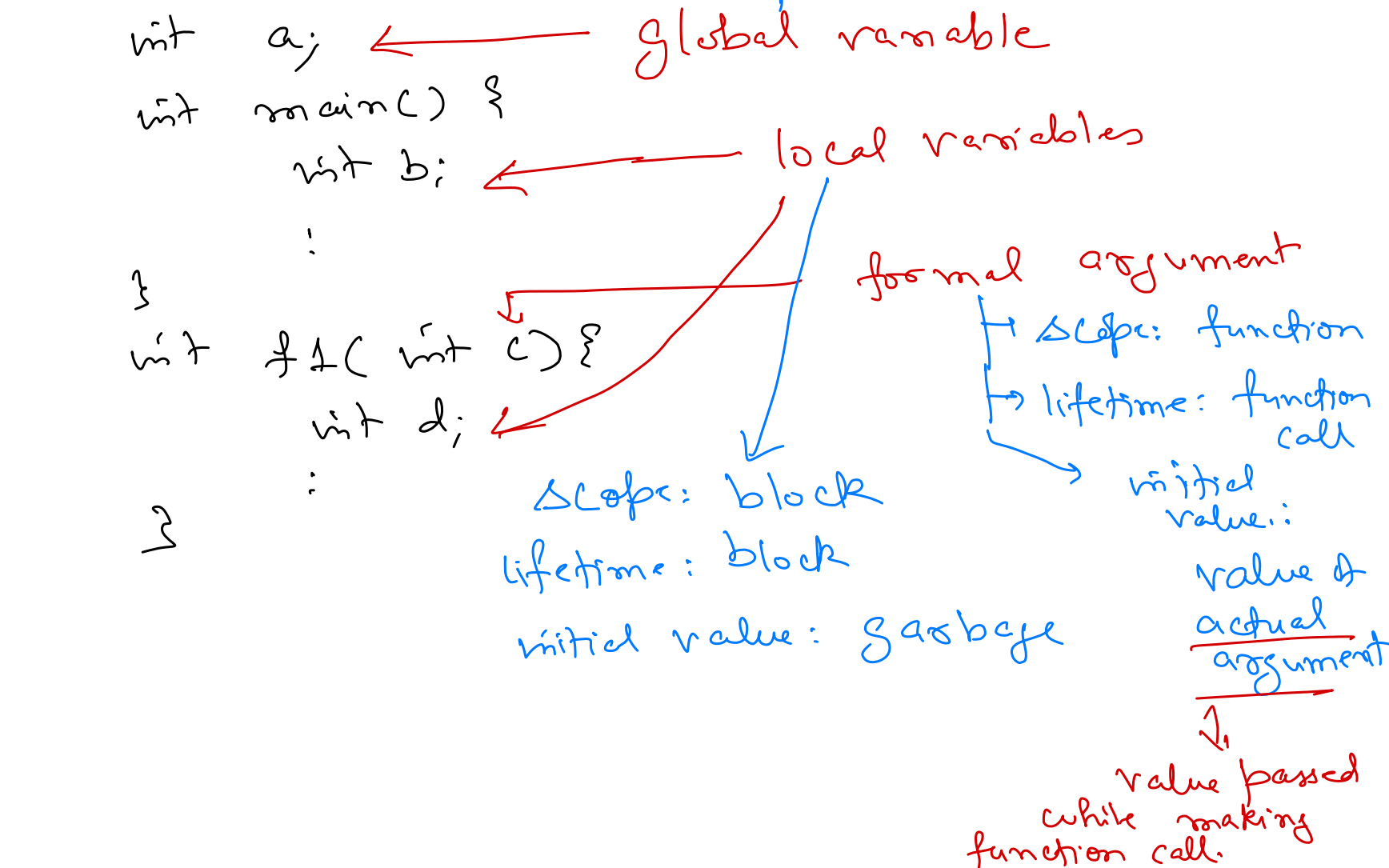
function prototype.

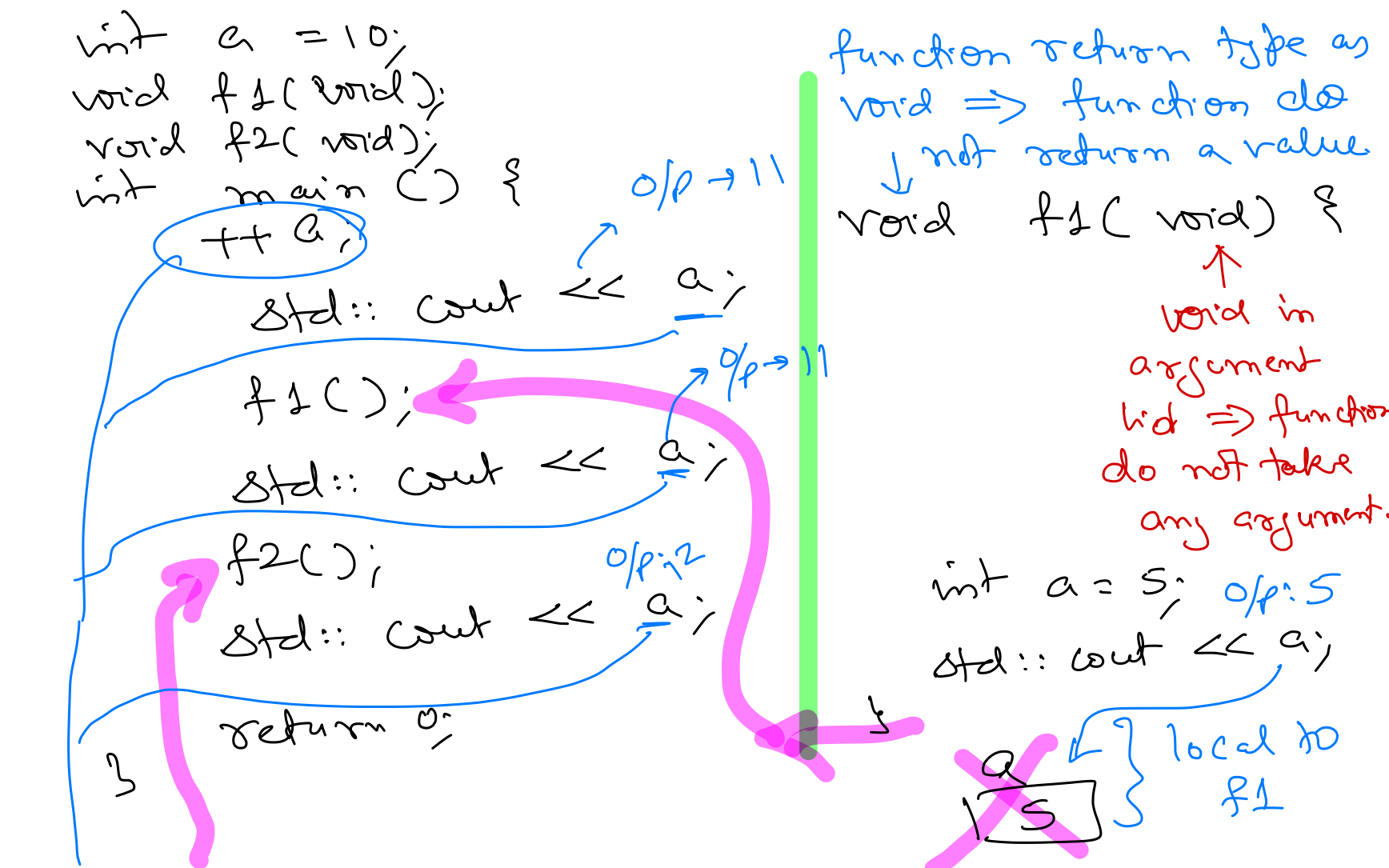
You can declare as many times as you want.

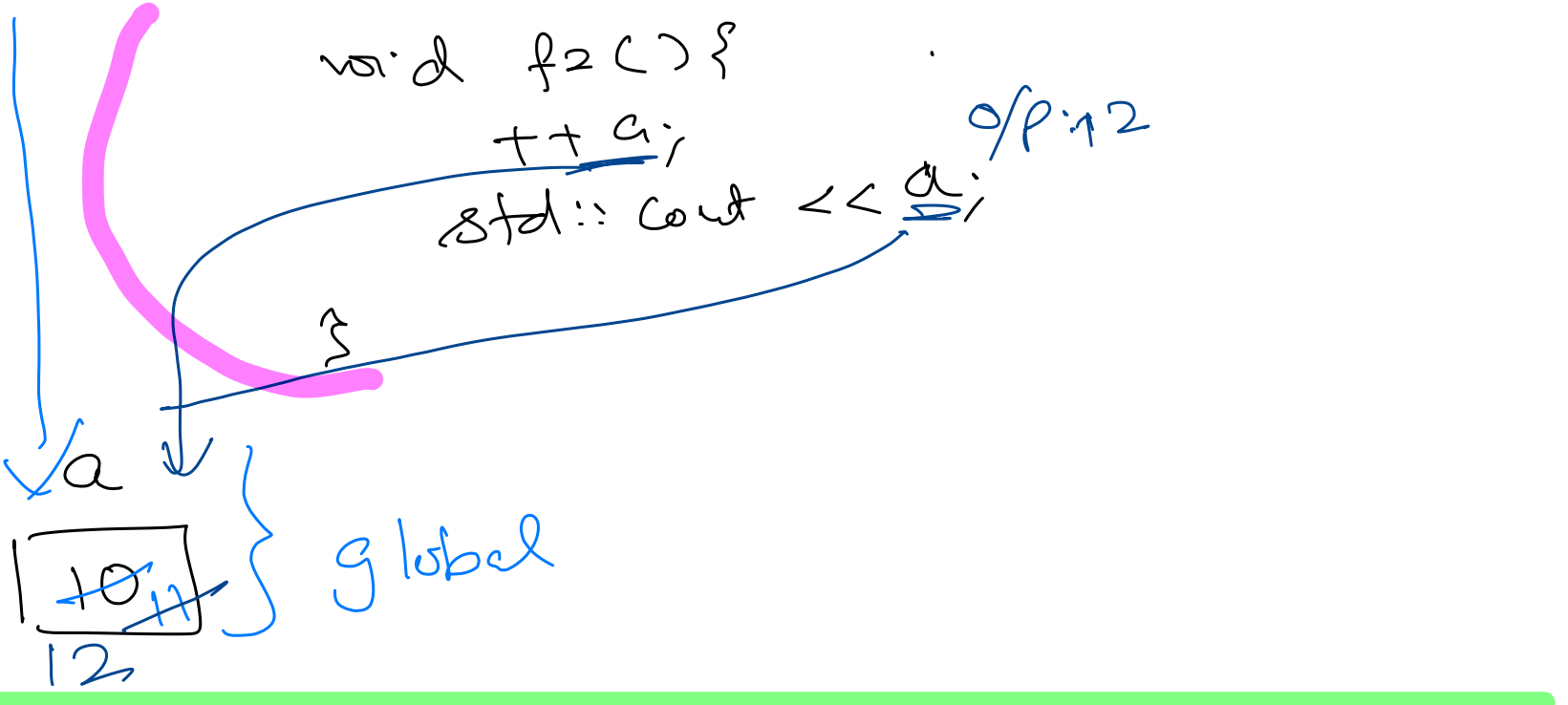
But must be defined exactly once.


int sum (int a, int b);
int main() {
int a;
a = sum (5, 10);
std:: cout << a;
return 0;
}
int sum (int a, int b) {
return a+b;
}
value of expression
= value returned
by function
a
15
b
10
a
5
15
5+10 = 15


variables
local
global
formal argument
Scope →
lifetime →
initial value →
defined outside all blocks.
variables defined in function argument list
variable defined in a statement block
Scope: entire program
lifetime: entire program
initial value: 0


int a;
global variable
int main() {
int b;
local variables
:
}
int f1(int c){
int d;
:
}
formal argument
Scope: function
lifetime: function call
initial value:
value of actual argument
value passed while making function call.
Scope: block
lifetime: block
initial value: garbage


int a = 10;
void f1(void);
void f2(void);
int main() {
++a;
std:: cout << a;
o/p → 11
f1();
std:: cout << a;
o/p → 11
f2();
std:: cout << a;
o/p: 12
return 0;
}
function return type as void ⇒ function do not return a value
void f1(void) {
void in argument list ⇒ function do not take any argument.
int a = 5;
o/p: 5
std:: cout << a;
}
a
5
local to f1


void f2(){
++a;
std::cout << a;
}
o/p: 12
a
10
11
12
global

Exercise

① write function to print 'n' lines of

```
*
* *
* * *
* * * *
```

n = 4

② write function to find n!

③ write function to print 'n' lines of

```
  *
 * *
* * *
```

n = 3

```
    *
  * * *
* * * * *
```

n = 3