Object Oriented Programming
Dr Robert Harle

IA CST, PPS (CS) and NST (CS)
Lent 2011/12

The OOP Course

- Last term you studied **functional** programming (ML)
- This term you are looking at **imperative** programming (Java primarily).
  - You already have a few weeks of Java experience
  - This course is hopefully going to let you separate the fundamental software design principles from Java's quirks and specifics

- Four Parts
  - From Functional to Imperative
  - Object-Oriented Concepts
  - The Java Platform
  - Design Patterns and OOP design examples

Java Practicals

- This course is meant to *complement* your practicals in Java
  - Some material appears only here
  - Some material appears only in the practicals
  - Some material appears in both: deliberately*!

* Some material may be repeated unintentionally. If so I will claim it was deliberate.

Books and Resources I

- OOP Concepts
  - Look for books for those learning to first program in an OOP language (Java, C++, Python)
  - *Java: How to Program* by Deitel & Deitel (also C++)
  - *Thinking in Java* by Eckels
  - *Java in a Nutshell* (O' Reilly) if you already know another OOP language
  - Java specification book: http://java.sun.com/docs/books/jls/
  - Lots of good resources on the web

- Design Patterns
  - *Design Patterns* by Gamma et al.
  - Lots of good resources on the web

Books and Resources II

- Also check the course web page
  - Updated notes (with annotations where possible)
  - Code from the lectures
  - Sample tripos questions

http://www.cl.cam.ac.uk/teaching/1112/OOProg/

Section: From Functional to Imperative Programming

Explicit Start Points

Standard

Arguments

Java: `public static void main(String args[])`

C/C++: `int main(int argc, char **argv)`

Return Type

python:

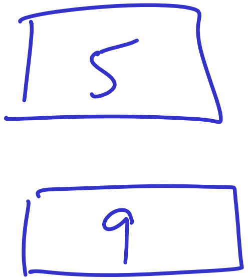
```
def main():
     # main code here

if __name__ == "__main__":
     main()
```

Immutable to Mutable Data

ML

```
- val x=5;
> val x = 5 : int
- x=7;
> val it = false : bool
- val x=9;
> val x = 9 : int
```

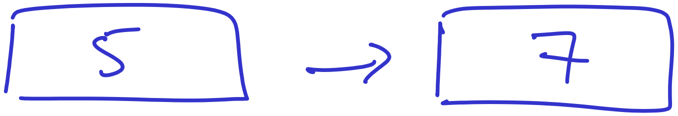
Test



Java

```
int x=5;
x=7;

int x=9;
```

fail



Types and Variables

- We write code like:

Manual types

```
int x = 512;
int y = 200;
int z = x+y;
```

- The high-level language has a series of *primitive* (built-in) types that we use to signify what's in the memory
  - The compiler then knows what to do with them
  - E.g. An "int" is a primitive type in C, C++, Java and many languages. It's usually a 32-bit signed integer
- A variable is a name used in the code to refer to a specific instance of a type
  - x,y,z are variables above
  - They are all of type int

E.g. Primitive Types in Java

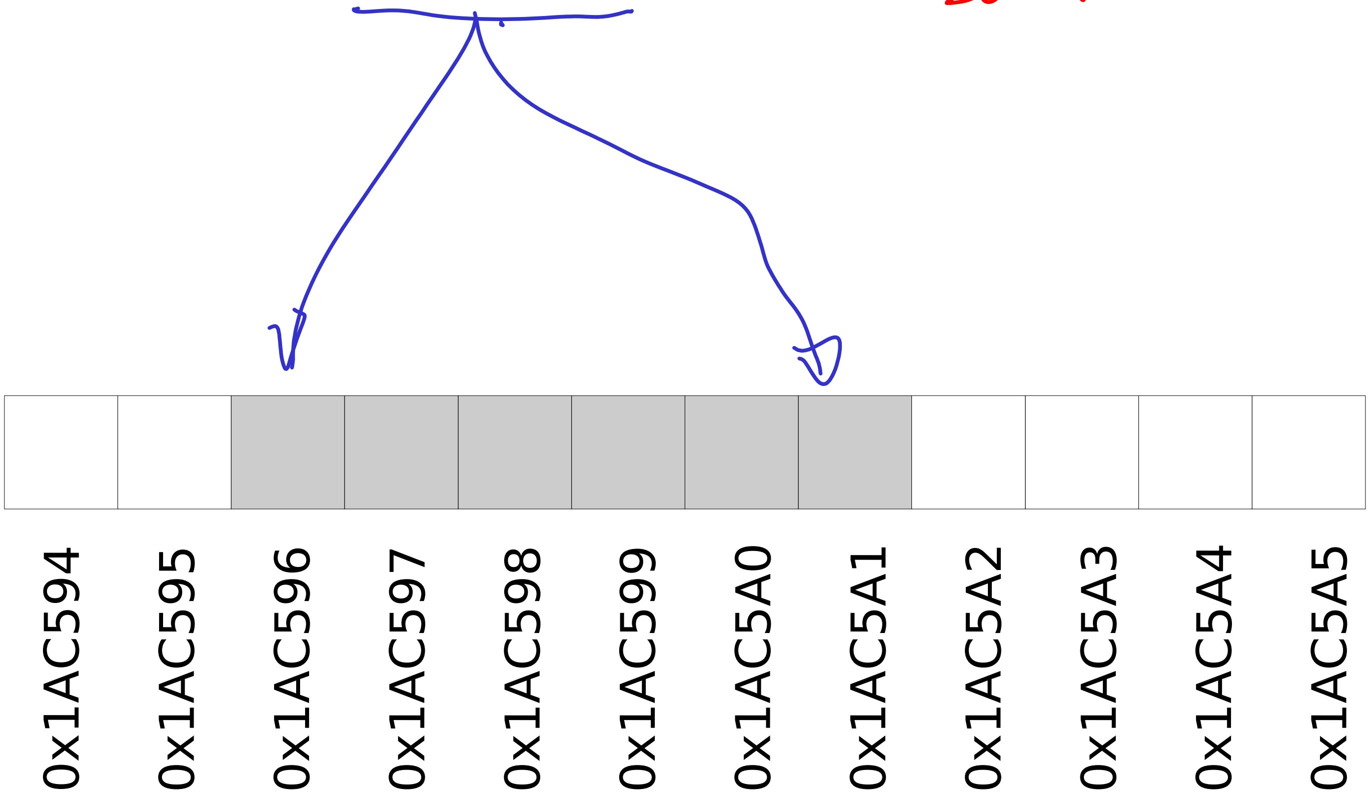
- "Primitive" types are the built in ones.
  - They are building blocks for more complicated types that we will be looking at soon.
- boolean – 1 bit (true, false)
- char – 16 bits ← UNICODE
- byte – 8 bits as a signed integer (-128 to 127)
- short – 16 bits as a signed integer
- int – 32 bits as a signed integer
- long – 64 bits as a signed integer
- float – 32 bits as a floating point number
- double – 64 bits as a floating point number

See Workbook 1

Arrays

```
byte[] arraydemo = new byte[6];
byte   arraydemo2[] = new byte[6];
```

Both work



Functions to Procedures

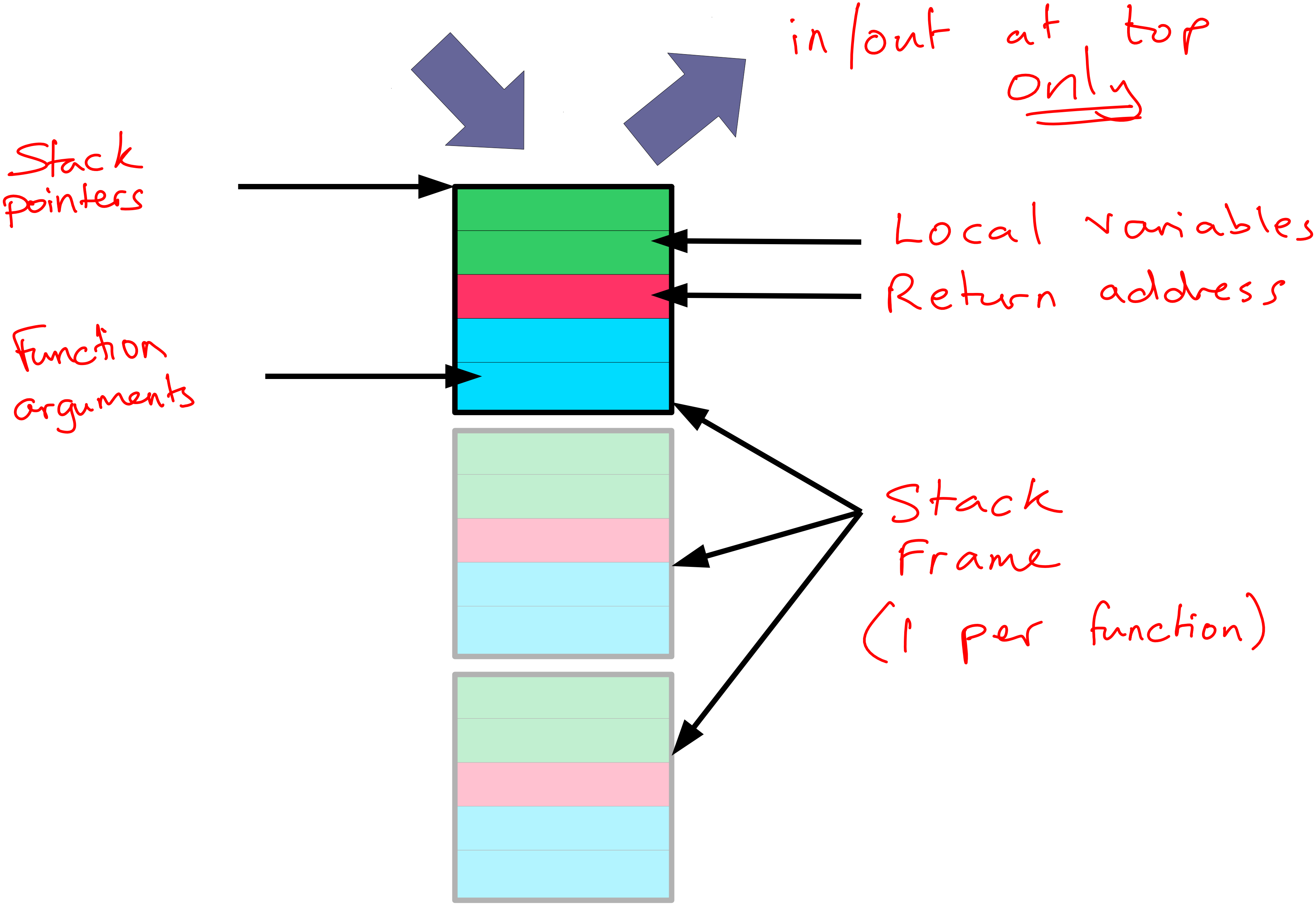
Maths: $m(x,y) = xy$ ✓

ML: `fun m(x,y) = x*y;` ✓

Java: `public int m(int x, int y) = x*y;` ✓

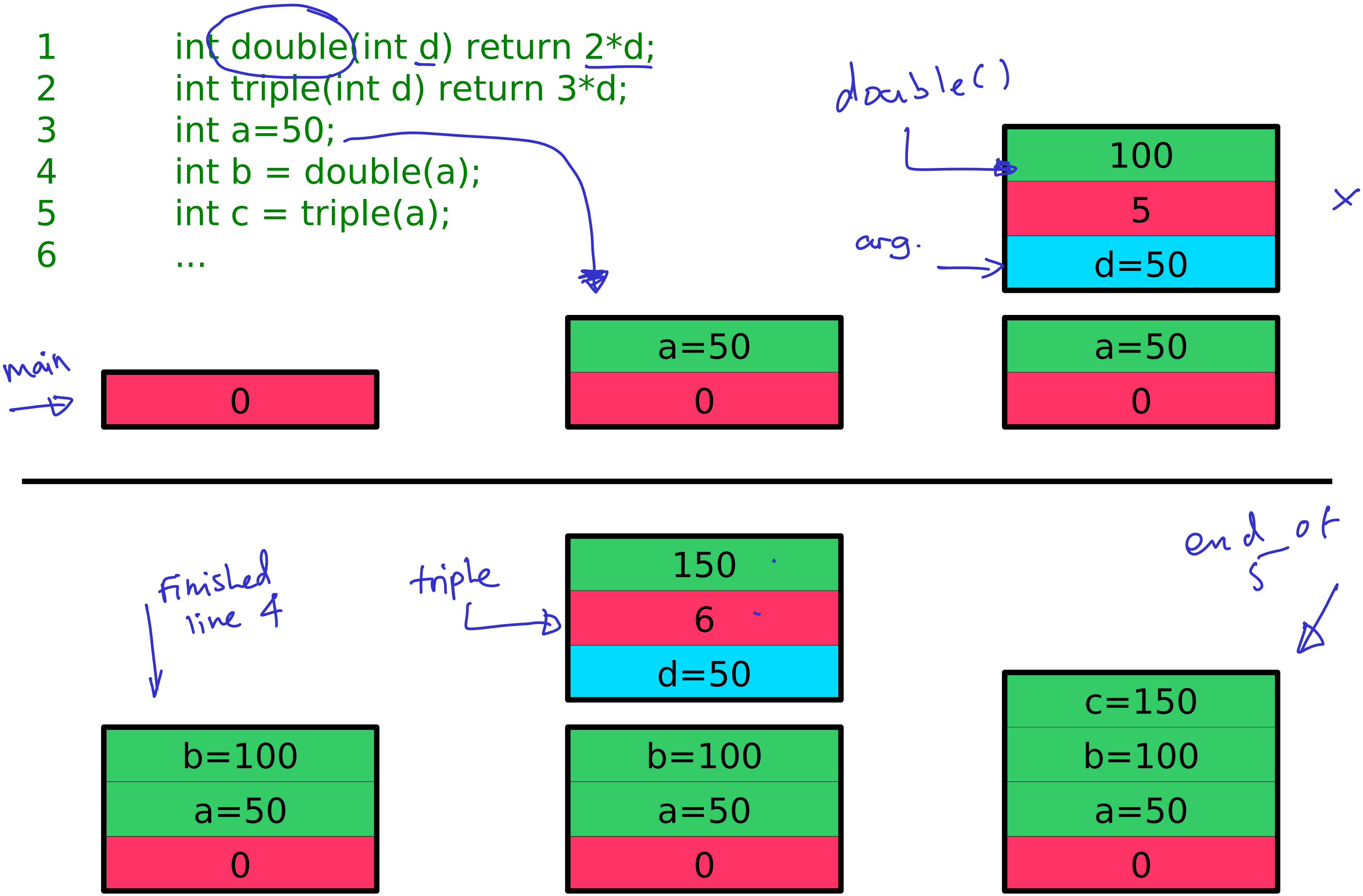
```
int y = 7;
public int m(x) {
        y=y+1;
        return x*y;
}
```

The Call Stack



The Call Stack: Example

```
1        int double(int d) return 2*d;
2        int triple(int d) return 3*d;
3        int a=50;
4        int b = double(a);
5        int c = triple(a);
6        ...
```



main

0

a=50
0

double( )

arg.

100
5
d=50
a=50
0

x

Finished line 4

b=100
a=50
0

triple

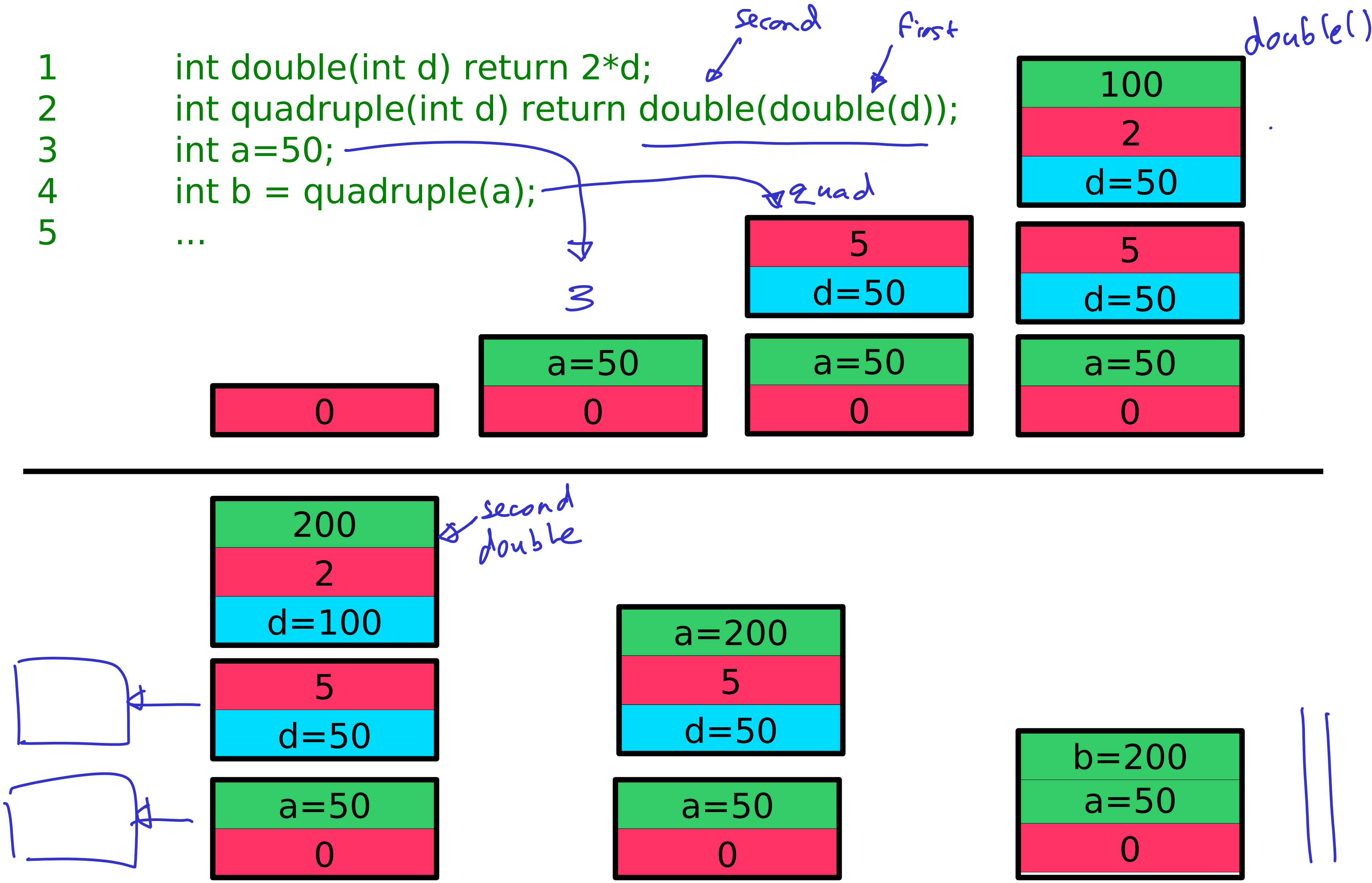
150
6
d=50
b=100
a=50
0

end of 5

c=150
b=100
a=50
0

Nested Functions

```
1    int double(int d) return 2*d;
2    int quadruple(int d) return double(double(d));
3    int a=50;
4    int b = quadruple(a);
5    ...
```



second
first
double()
3
quad
0
a=50
0
5
d=50
a=50
0
100
2
d=50
5
d=50
a=50
0
200
2
d=100
second double
5
d=50
a=50
0
a=200
5
d=50
a=50
0
b=200
a=50
0