Functions

Dr. Nachiket Tapas

Function Prototype

- Similar to variable declaration, if the function is defined after it is called, you need to tell the compiler about the function.

```
void main() {
        ...
        x = max(c, d);
}
int max(int a, int b) {
        x = a>b?a:b;
        return x;
}
```

You will get a warning message as function is used before defined based on the compiler.

Solution

```
int max(int a, int b);          //function declaration or prototype
void main() {
        ...
        x = max(c, d);
}
int max(int c, int d) {
        x = a>b?a:b;
        return x;
}
```

Syntax

return_type function_name (list_of_arguments);

Examples:

```
int max(int a, int b);
int max(int, int);
```

Position in program: Before the call to the function

Interesting fact: header files (stdio.h, math.h) contain prototype/declaration of frequently used functions. In Linux/Unix, the definition of function is stored in library file libc or libm.

Nested Function Calls

- Functions can call each other
- A declaration or definition (or both) must be visible before the call.
  - Help compiler detect any inconsistencies in function use
  - Compiler warning, if both (declaration and definition) are missing

Example

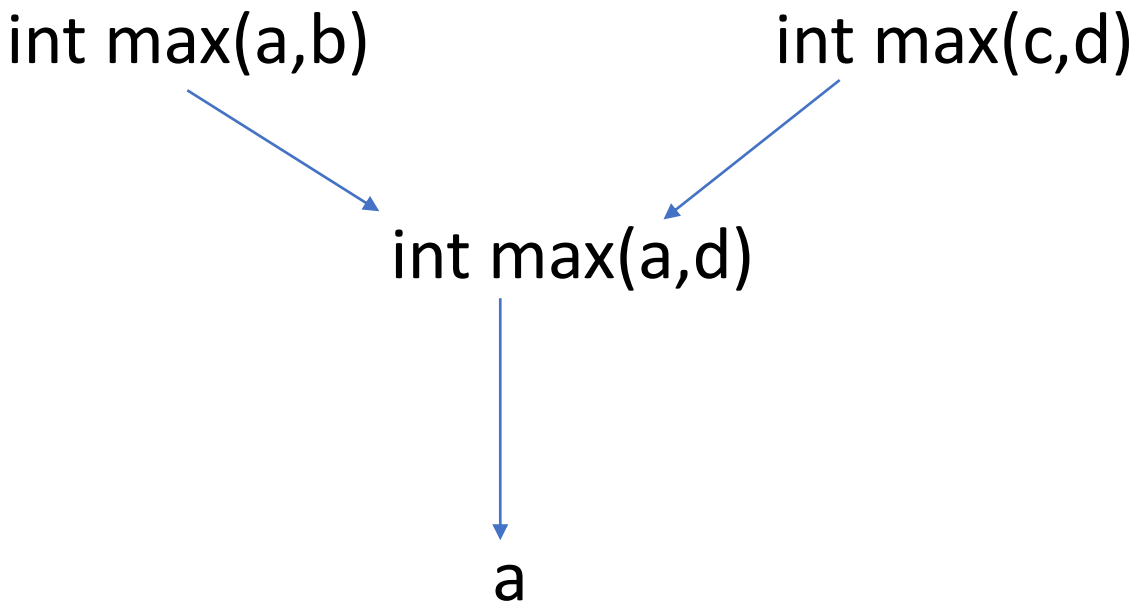
```
#include<stdio.h>
//int min(int, int);
//int max(int, int);
```

```
int main() {
        printf("%d", min(6,4));
}

int max(int a, int b) {
        return (a>b) ? a : b;
}

int min(int a, int b) {
        return a + b – max(a, b);
}
```

Maximum of 4 numbers

```
int max(int a, int b) {
        return a>b?a:b;
}
int max4(int a, int b, int c, int d) {
        return max(max(a,b), max(c, d));
}
void main() {
        int a, b, c, d, m;
        m = max4( a, b, c, d);
}
```

Execution



Function to swap two numbers

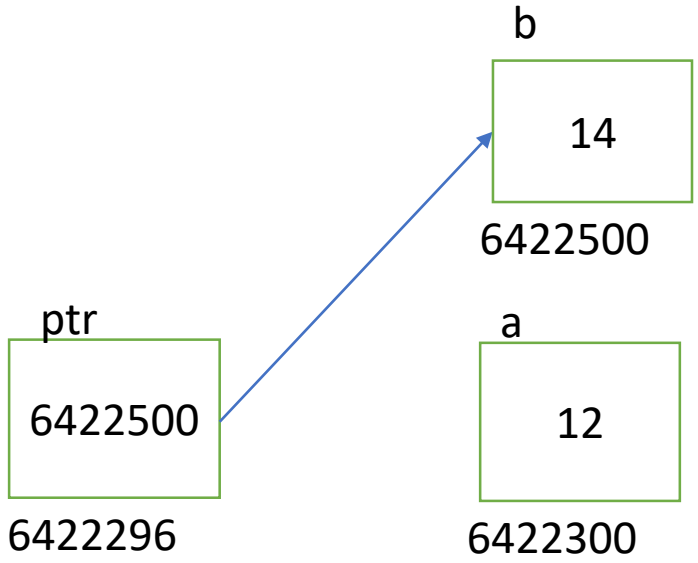
```
#include <stdio.h>
void swap(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
        printf("Value in function swap, a=%d, b=%d\n", a, b);
}
```

```
void main() {
        int a=5, b=6;
        printf("Value before swap, a=%d, b=%d\n", a, b);
        swap(a, b);
        printf("Value after swap, a=%d, b=%d\n", a, b);
}
```

Does this code swap the values inside main?
If not, how to do it then?

Call-by-Reference Parameters

- Instead of value, address of a parameter is passed.
- In function definition, a special variable is used to hold the address of a variable called pointer.
- A pointer looks like int * prt;



- int a; &a=100 a=5

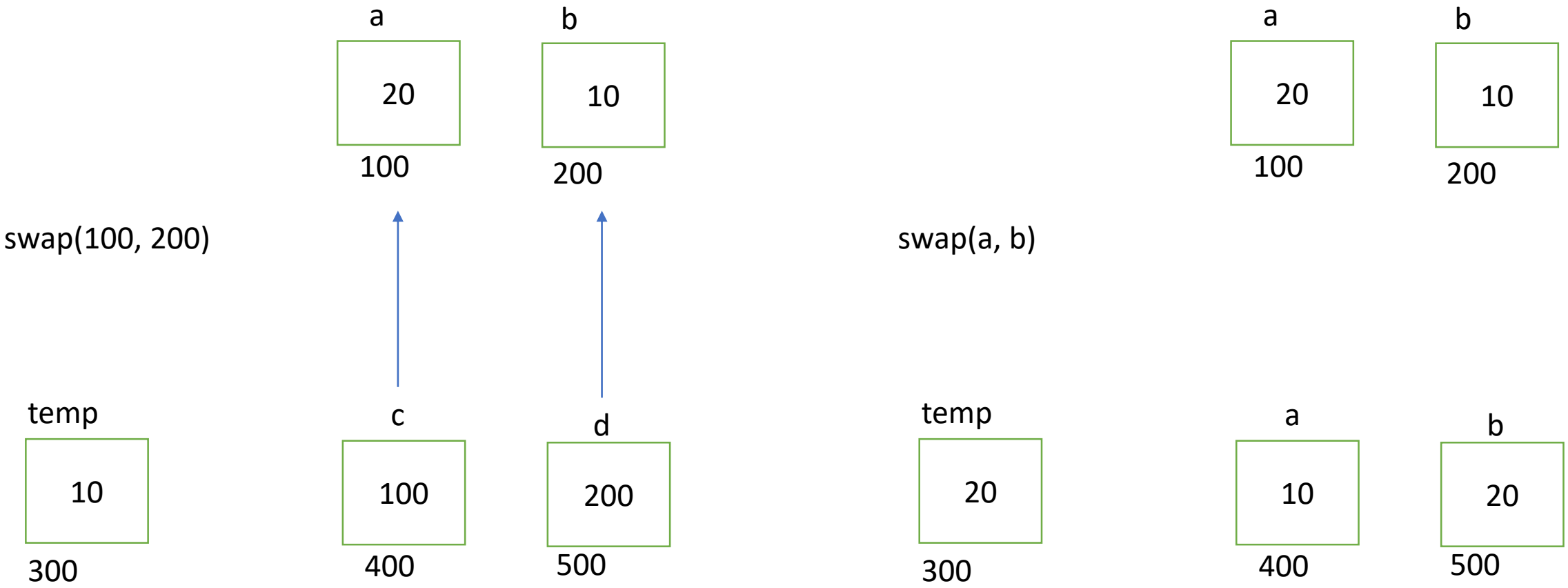
- int * b; b = 100;
- printf(“%d”, a); //12
- printf(“%u”, &a); //6422300
- printf(“%u”, ptr); //6422300
- printf(“%d”, *ptr); //value at address stored in ptr 12

Function to swap two numbers

```
#include <stdio.h>
void swap(int * c, int * d) {
        int temp=*c;
        *c=*d;
        *d=temp;
        printf("Value in function swap, a=%d, b=%d\n", *c, *d);
}
```

```
void main() {
        int a=5, b=6;
        printf("Value before swap, a=%d, b=%d\n", a, b);
        swap(&a, &b);
        printf("Value after swap, a=%d, b=%d\n", a, b);
}
```

Does this code swap the values inside main?

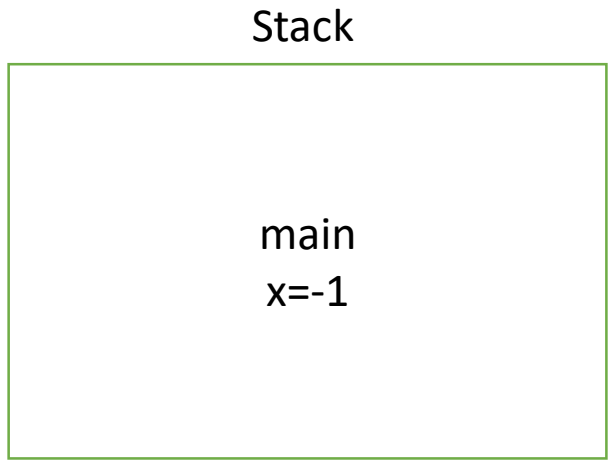

a
b
20
10
100
200
swap(100, 200)
swap(a, b)
temp
c
d
10
100
200
300
400
500
temp
a
b
20
10
20
300
400
500

Execution of a function

- An important role is played by a data structure called stack.
- A stack only grows in one direction.
- You can consider books kept on top of each other as stack.
- It contains information about the functions
- When a function is called, stack grows. When function terminates, stack shrinks.
- Two stack functions: push and pop.

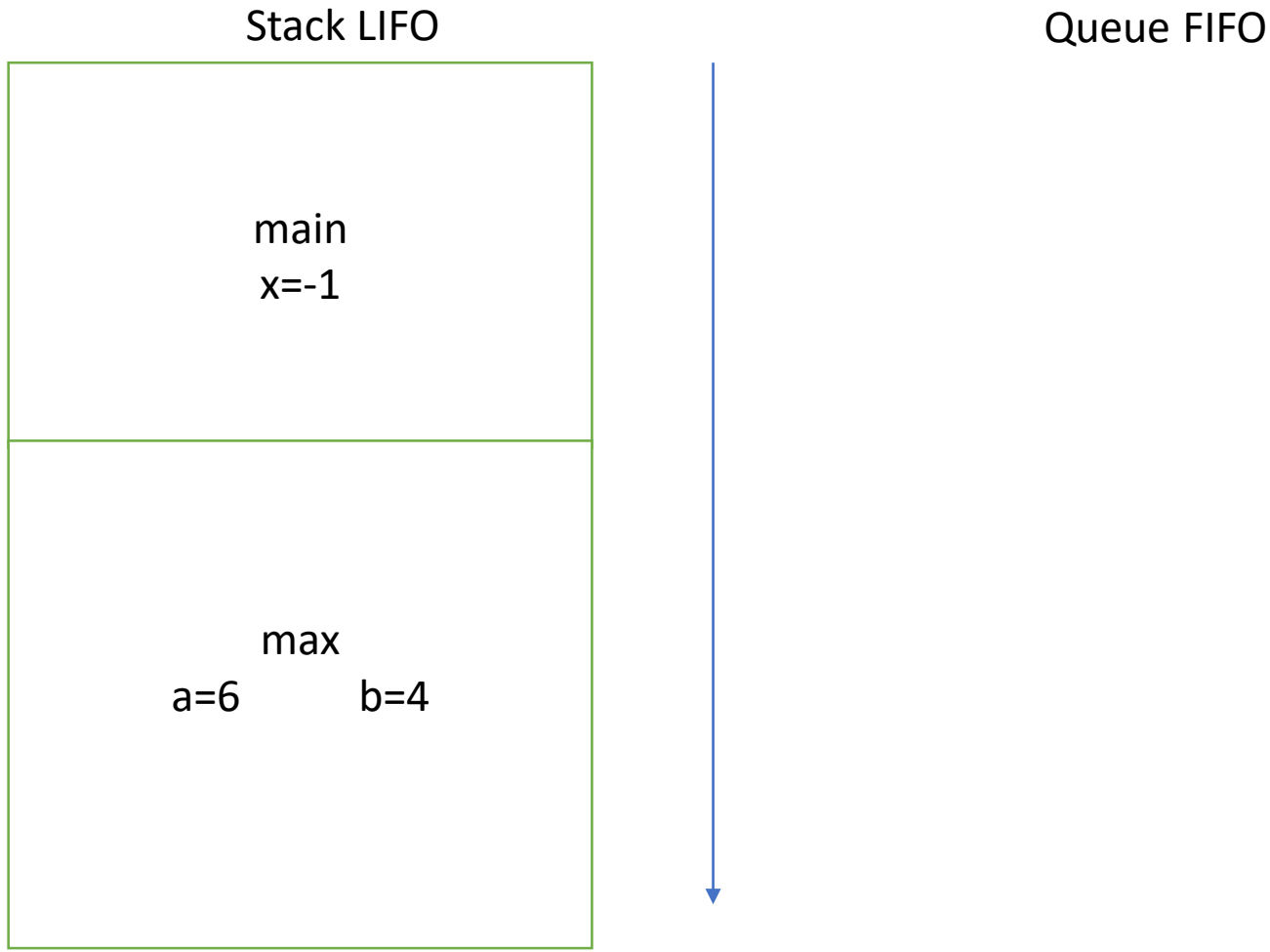
Example

```
#include<stdio.h>
int max(int a, int b) {
        if(a>b)
                return a;
        else
                return b;
}
int main() {
        int x = -1;
        x = max(6,4);
        printf("%d", x);
        return 0;
}
```



Example

```
#include<stdio.h>
int max(int a, int b) {
        if(a>b)
                return a;
        else
                return b;
}
int main() {
        int x = -1;
        x = max(6,4);
        printf("%d", x);
        return 0;
}
```



Example

```
#include<stdio.h>
int max(int a, int b) {
        if(a>b)
                return a;
        else
                return b;
}
int main() {
        int x = -1;
        x = max(6,4);
        printf("%d", x);
        return 0;
}
```