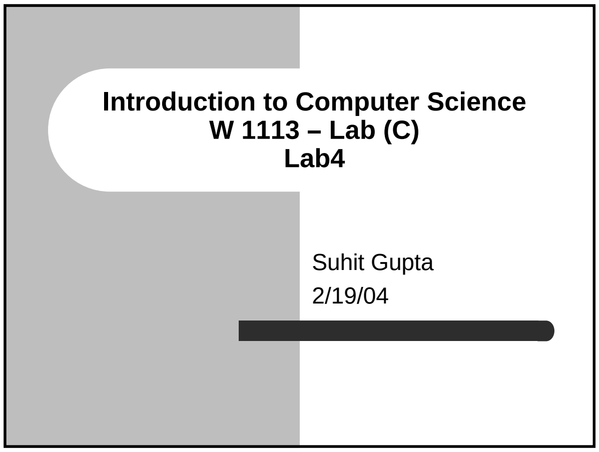
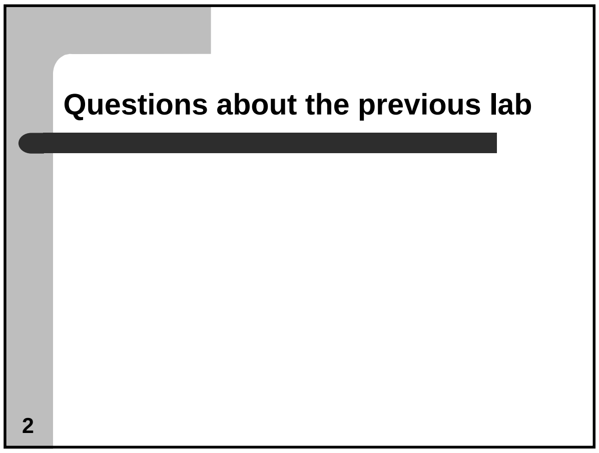
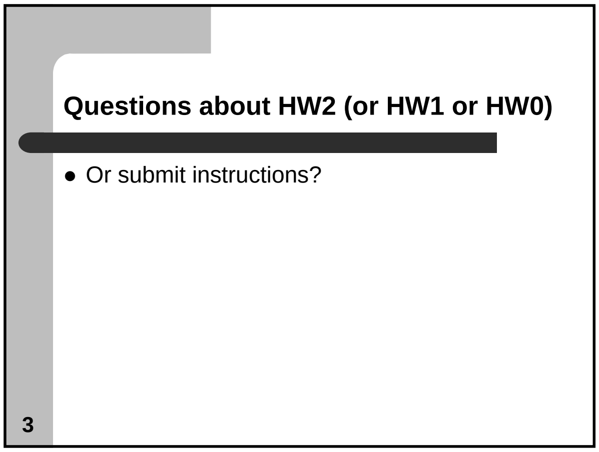


Introduction to Computer Science W 1113 – Lab (C) Lab4

Suhit Gupta
2/19/04



Questions about the previous lab



Questions about HW2 (or HW1 or HW0)

- Or submit instructions?

Recap from Lab 2

- Details on printf
- Details on scanf
- Conversion between data types
- Math operators
- Command Line Parameters

4

Recap from Lab 3

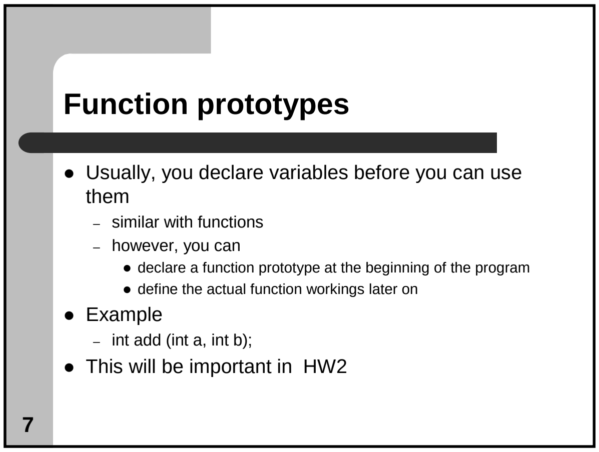
- Math operators
- Arrays (assignment and reference)
- Strings
  - string manipulation
  - fgets
  - sscanf

5

Quick quiz…

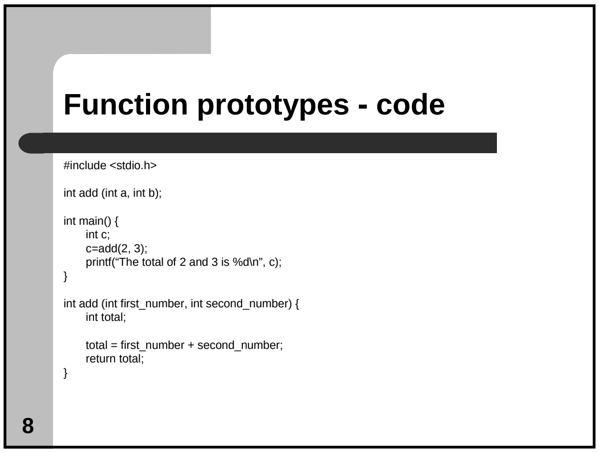
- BTW, I will be asking one (or two) questions every class that are in the reading only… brownie points ☺
- +=, -=, *=, /=
  - What do these do?

6



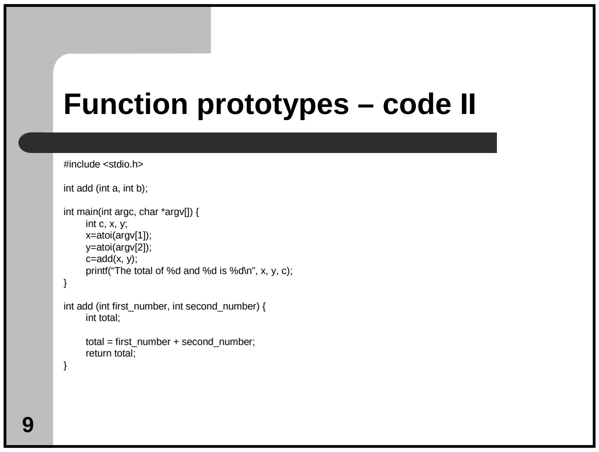
Function prototypes

- Usually, you declare variables before you can use them
  - similar with functions
  - however, you can
    - declare a function prototype at the beginning of the program
    - define the actual function workings later on
- Example
  - int add (int a, int b);
- This will be important in HW2



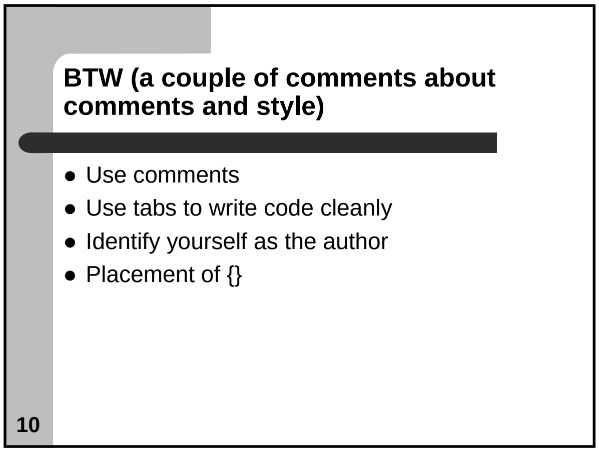
Function prototypes - code

```
#include <stdio.h>

int add (int a, int b);

int main() {
       int c;
       c=add(2, 3);
       printf(“The total of 2 and 3 is %d\n”, c);
}

int add (int first_number, int second_number) {
       int total;

       total = first_number + second_number;
       return total;
}
```



Function prototypes – code II

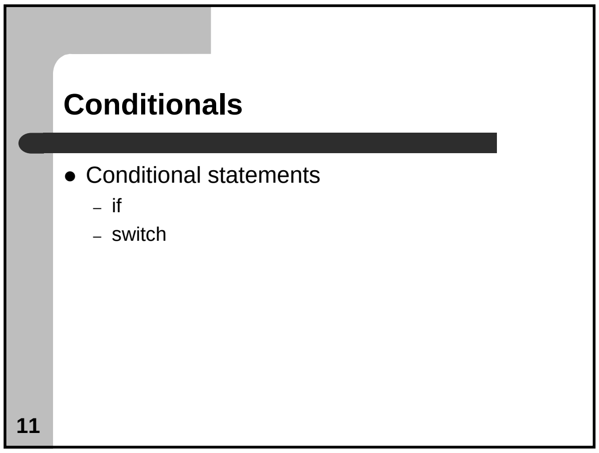
```
#include <stdio.h>

int add (int a, int b);

int main(int argc, char *argv[]) {
       int c, x, y;
       x=atoi(argv[1]);
       y=atoi(argv[2]);
       c=add(x, y);
       printf(“The total of %d and %d is %d\n”, x, y, c);
}

int add (int first_number, int second_number) {
       int total;

       total = first_number + second_number;
       return total;
}
```



BTW (a couple of comments about comments and style)

- Use comments
- Use tabs to write code cleanly
- Identify yourself as the author
- Placement of {}

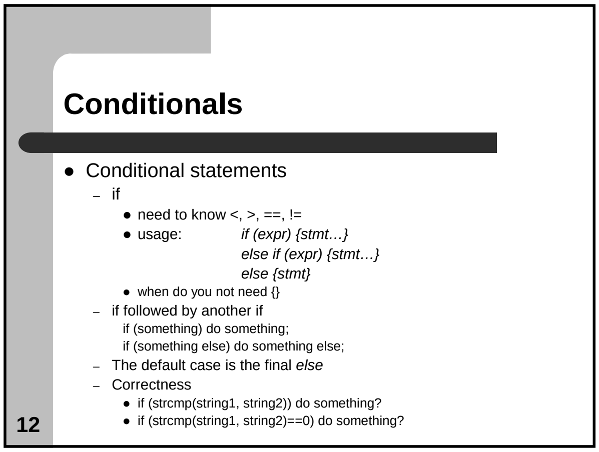
10



Conditionals

- Conditional statements
  - if
  - switch

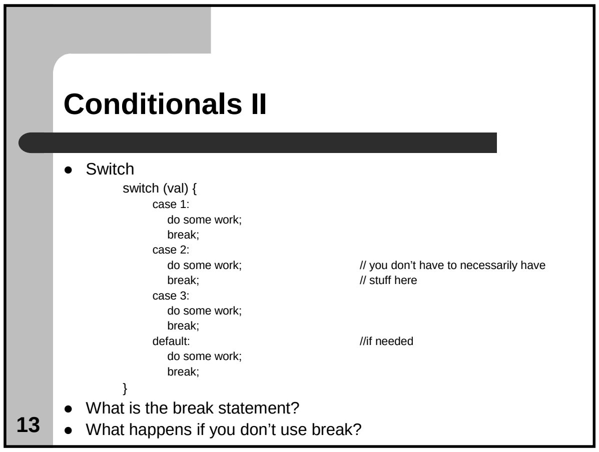
11



Conditionals

- Conditional statements
  - if
    - need to know <, >, ==, !=
    - usage: *if (expr) {stmt…}*
      else if (expr) {stmt…}
      else {stmt}
    - when do you not need {}
  - if followed by another if
    if (something) do something;
    if (something else) do something else;
  - The default case is the final *else*
  - Correctness
    - if (strcmp(string1, string2)) do something?
    - if (strcmp(string1, string2)==0) do something?

12



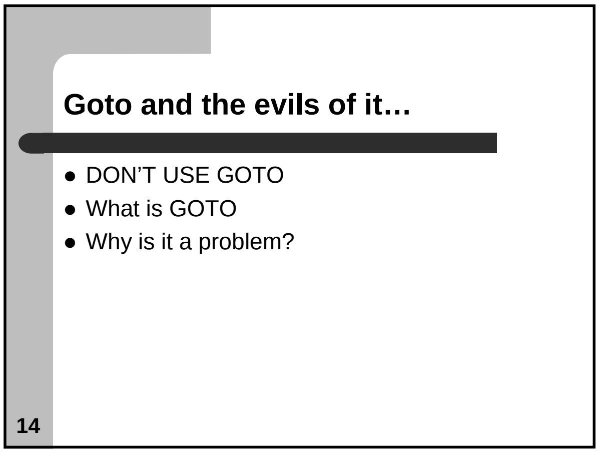
Conditionals II

- Switch

```
switch (val) {
    case 1:
        do some work;
        break;
    case 2:
        do some work;                    // you don't have to necessarily have
        break;                           // stuff here
    case 3:
        do some work;
        break;
    default:                             //if needed
        do some work;
        break;
}
```

- What is the break statement?
- What happens if you don’t use break?

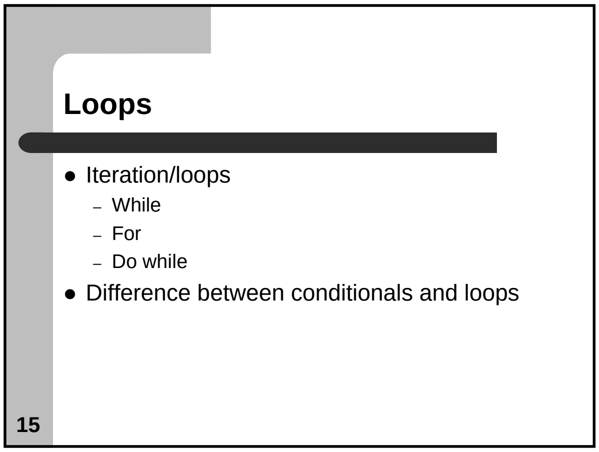
13



Goto and the evils of it…

- DON’T USE GOTO
- What is GOTO
- Why is it a problem?

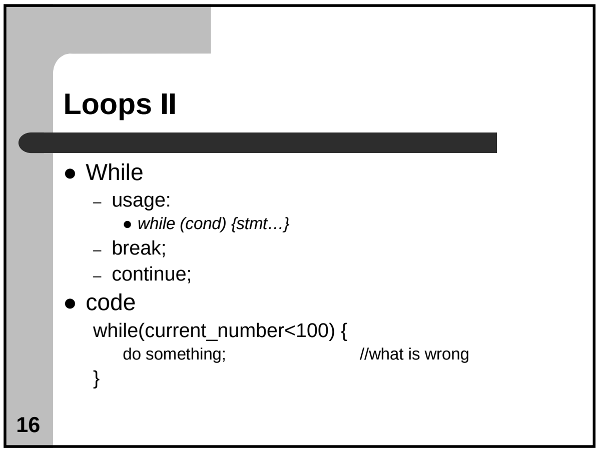
14



Loops

- Iteration/loops
  - While
  - For
  - Do while
- Difference between conditionals and loops

15

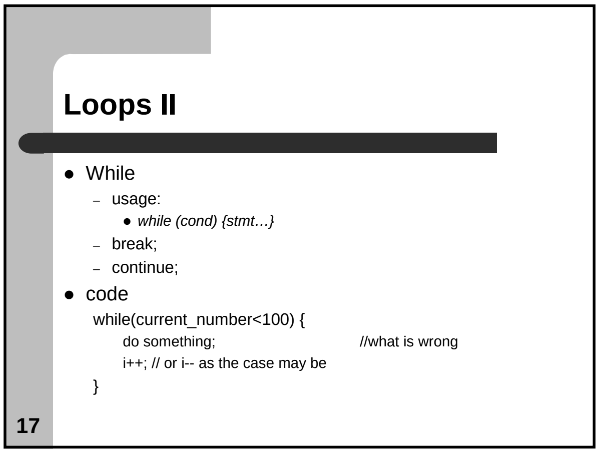


Loops II

- While
  - usage:
    - *while (cond) {stmt…}*
  - break;
  - continue;
- code

```
while(current_number<100) {
    do something;                    //what is wrong
}
```

16

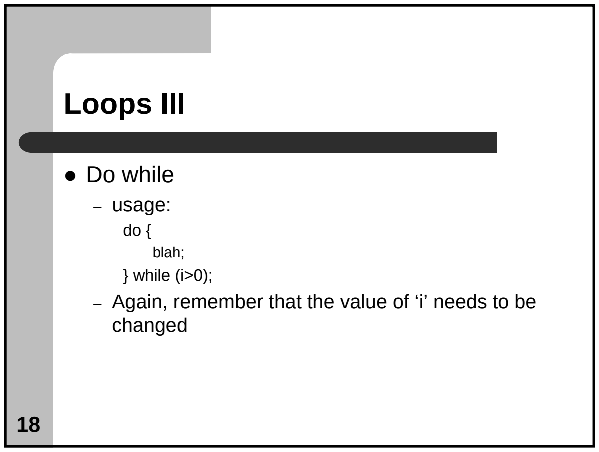


Loops II

- While
  - usage:
    - *while (cond) {stmt…}*
  - break;
  - continue;
- code

```
while(current_number<100) {
    do something;                    //what is wrong
    i++; // or i-- as the case may be
}
```

17



Loops III

- Do while
  - usage:

```
    do {
        blah;
    } while (i>0);
```

  - Again, remember that the value of ‘i’ needs to be changed

18

Loops IV

- For
  - usage:
    - for (… ; … ; …) {
      do something here;
      }
  - There is other acceptable syntax (sort of)
- BTW, this is where the ++i and i++ becomes relevant and useful
- Everything in for can be done in a while
  - Think about it

19

Loops V

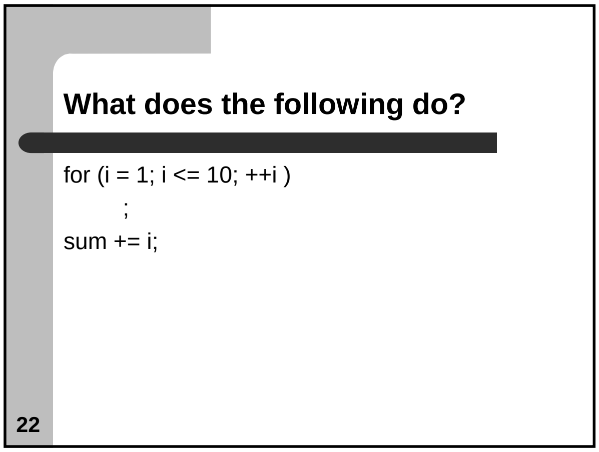
- The comma operator
  - Things are evaluated from left to right
- for (sum=0, i=1; i<=n; ++i)
  sum += i;
- for (sum=0, i=1; i<=n; sum += i, ++i)
  ;
- for (sum=0, i=1; i<=n; ++i, sum += i)
  ; *// this may give wrong results as i is*
  // incremented before added to sum

20

Loops VI

- Why can we use the ; just like that
- Infinite loops – beware
  - while (1) { …}
  - for (; ;) {…}
    - Use it at your own risk (system administrator may kill ;-))
    - Use it instead of running your program again and again

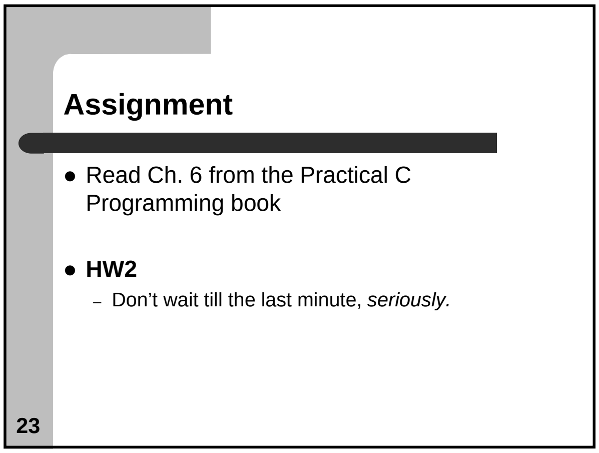
21



What does the following do?

```
for (i = 1; i <= 10; ++i )
        ;
sum += i;
```

22



Assignment

- Read Ch. 6 from the Practical C Programming book
- **HW2**
  - Don’t wait till the last minute, *seriously.*

23