COMP20005 Workshop Week 4

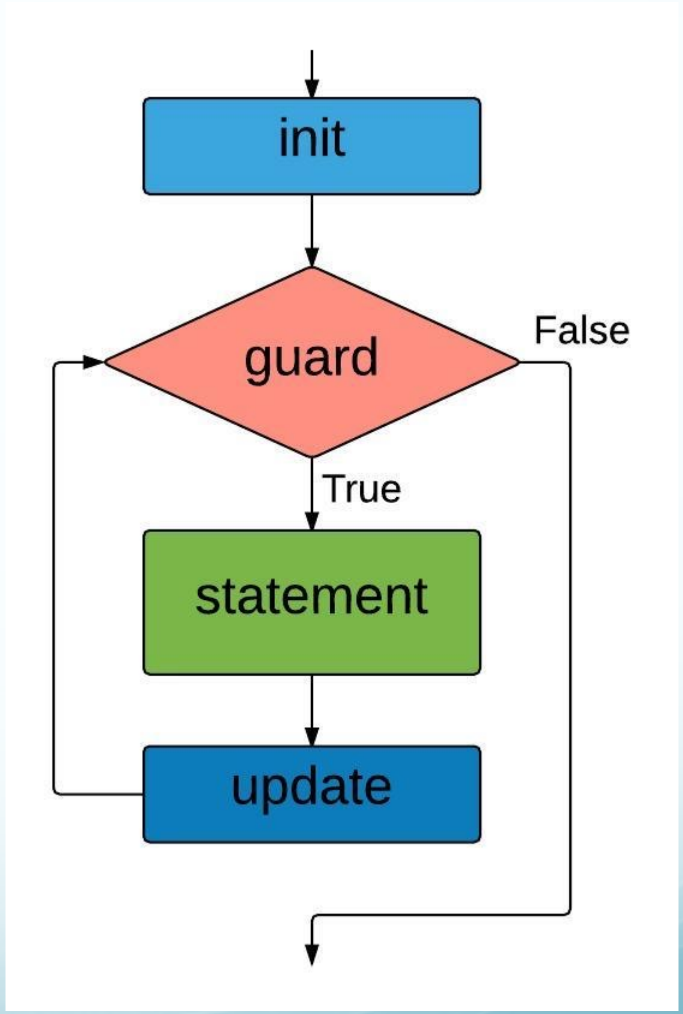
1	Review: loops; Discuss **`Ex 4.01`** and **`4.02`** Do Together: **`Ex 4.05`**
2	Functions: what, why, how? **`Ex 5.01`** Implement **`Ex 4.09`** using functions
LAB	Implement in the order: **`4.05 4.09 4.04 4.11+function 5.02 5.03`** min requirement: finish **`4.05`**, **`4.09`**
Past & Future	**Lectures W3:** everything about loops; **`getchar()`**, **`putchar()`**; functions **Quiz 1** (Week 6): covers chapters 1—5 (ie. till the end of functions) Try Sample Tests *before* Workshop Week 5

can be empty, empty means 1

```
for ( init ; guard       ; update ) {

        statement;...

}
```

All the boxes can be empty! The must-be parts are:

```
for (  ;  ;  )
```



continue (go to <update>) & break (exit loop)

```
for ( s=0, i=1 ; i<=10 ; i++ ) {

    if (i==2) {
        continue;
    }

    s += i*i;
    if (i==4) {
        break;
    }

}
printf("the end: i=%d, s= %d\n",i,s);
```

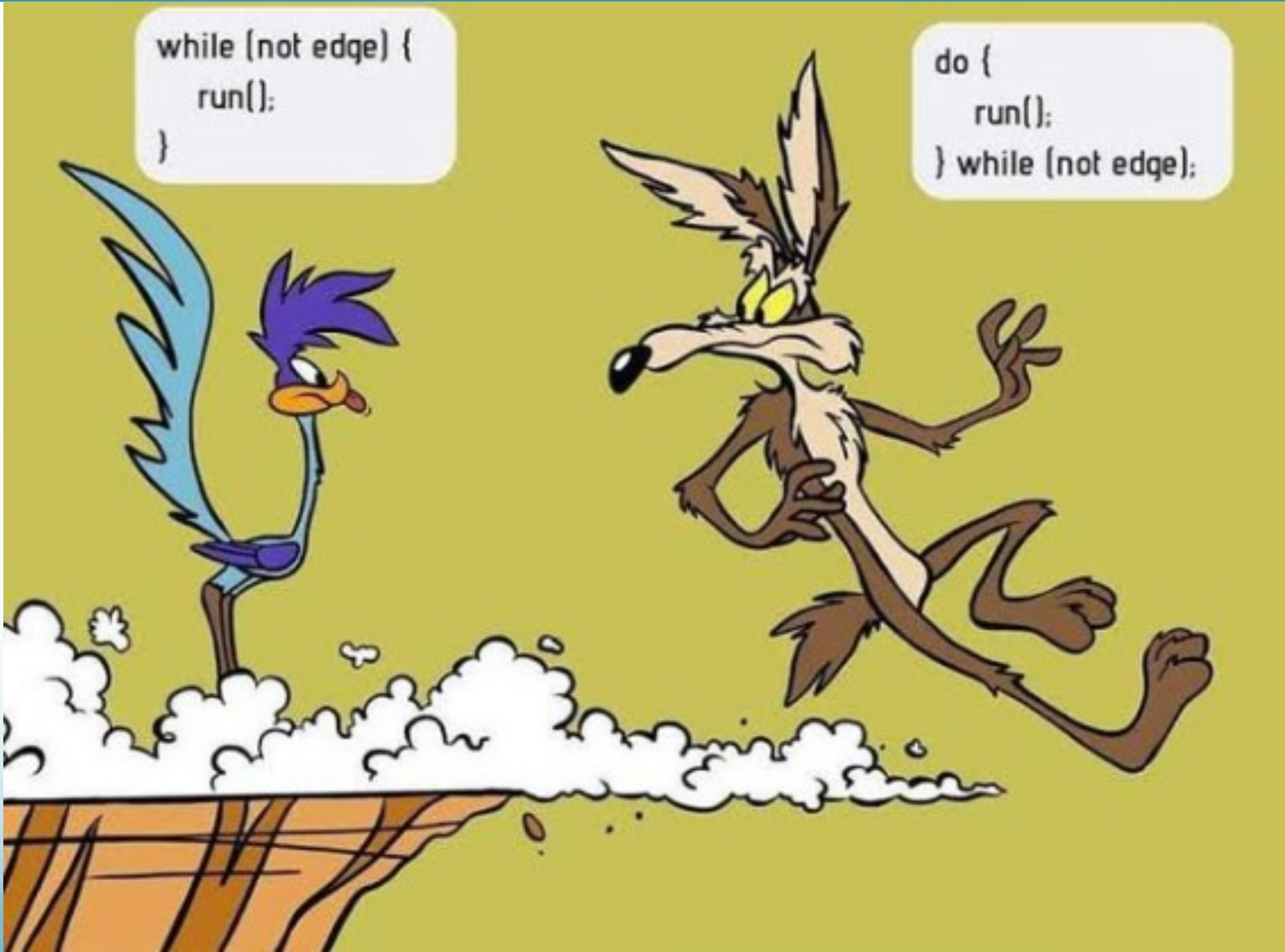
s= 0, i= 1

1 <=10: s=   update i=

Loops: while and do…while

```
while ( guard ) {
        statements;
}
```

```
do {
        statements;
} while (guard);
```



Ex 4.02

Give a general construction that shows how any `do` *statement can be converted into an equivalent* `while` *statement.*

```
do {
        statement;
} while (guard);
```

```
while ( guard ) {
        statement;
}
```

Ex 4.01 c) (changed)

Trace the action of the loop, and determine the values printed out by the `printf` *statement. Assume that all variables have been declared to be of type* `int`*.*

```
sum = 0;
for  (i=0; i<4; i++) {
        sum = sum + i;
        printf (“S(%2d) = %2d\n”, i, sum);
}
```

output of the code fragment

```
S() =
```

Ex 4.01 d)   (changed)

```
for (i= 0; i < 2; i++) {
  // i=
  for  (j= i+1; j < 8; j += 3) {
    // j=
    printf (“i= %d, j= %d\n”, i, j);
  }
}
```

```
i= , j=
```

Ex 4.01 e)   (changed)

```
for (i= 2; i < 5; i++) {
  for  (j= i+1; j < 8; j += 3) {
    if (i+j == 7) {
      break;
    }
    printf (“i= %d, j= %d\n”, i, j);
  }
}
```

```
i= , j=
```

Ex 4.01 f)

```
j = 5;
for (i= 0; i < j; i++) ; {
    printf (“i= %d, j= %d\n”, i, j);
}
```

```
i= , j=
```

How many lines printed out?

Ex 4.01 g)

```
j = 5;
for (i= 0; i < j; j++) {
    printf (“i= %d, j= %d\n”, i, j);
}
```

```
i= , j=
```

How many lines printed out?

Do-Together in grok: **Ex 4.05**

Design and implement a program `grapher.c` *that reads integers and draw a simple graph. Assume that all of the values read are between 1 and 70. Example:*

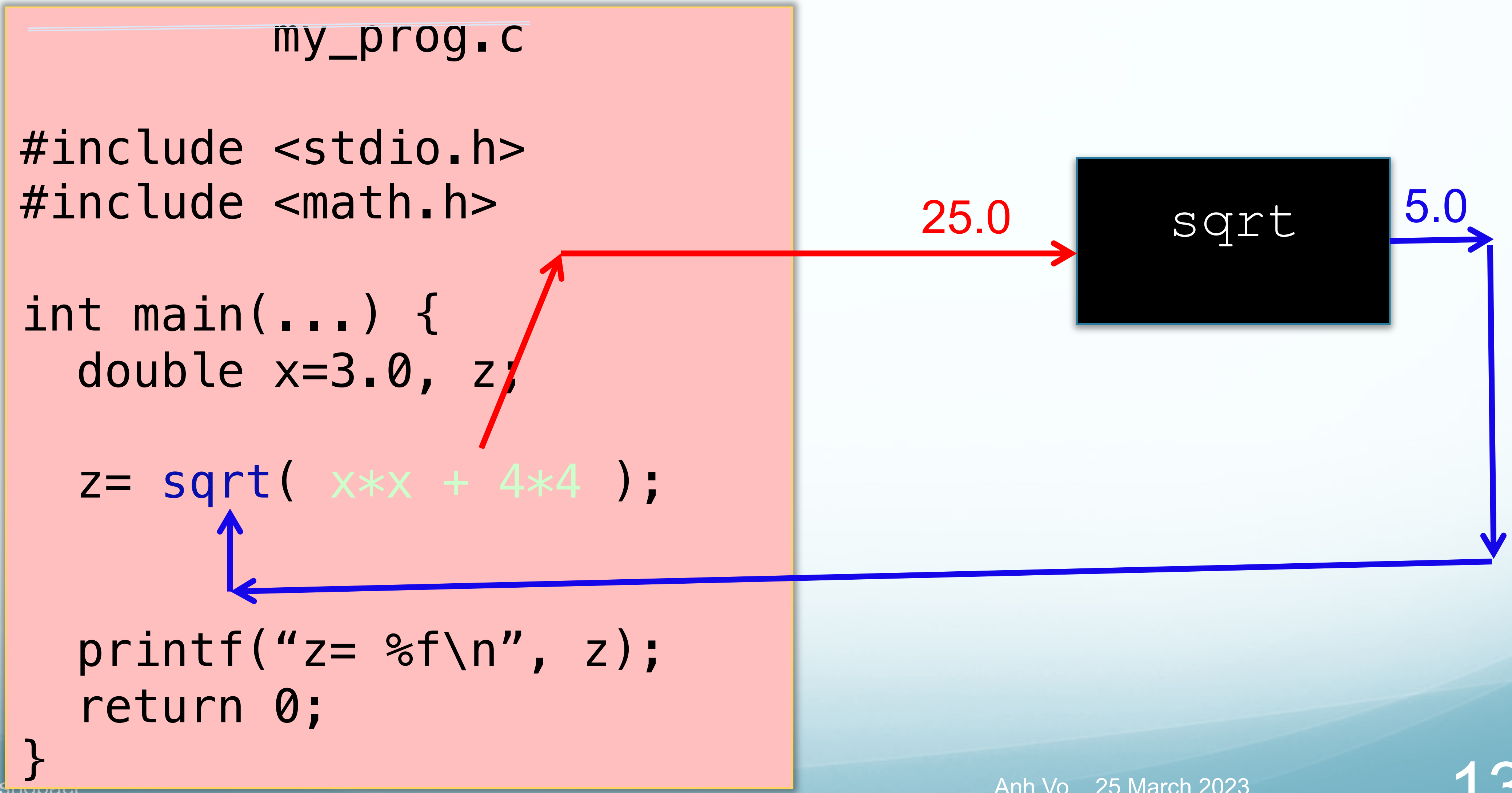
```
./grapher
Enter integers between 1 and 70 inclusive:
3  7  11
  3 |***
  7 |*******
 11 |***********
```

Functions

x= 2.0, need to print out √x

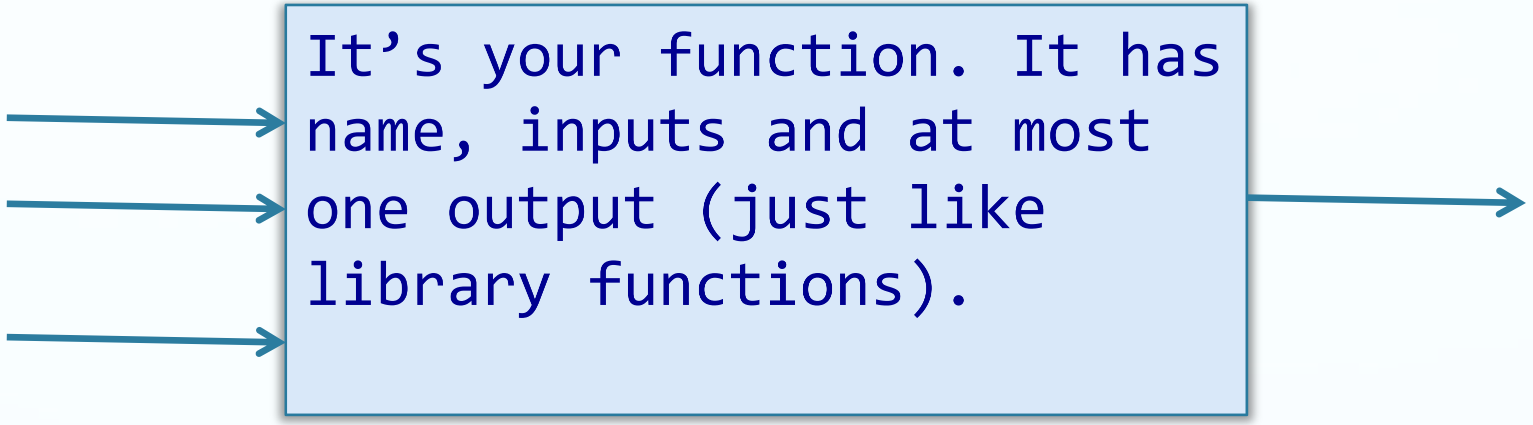
How?

(Library) functions are black boxes



```
              my_prog.c

#include <stdio.h>
#include <math.h>

int main(...) {
  double x=3.0, z;

  z= sqrt( x*x + 4*4 );


  printf("z= %f\n", z);
  return 0;
}
```

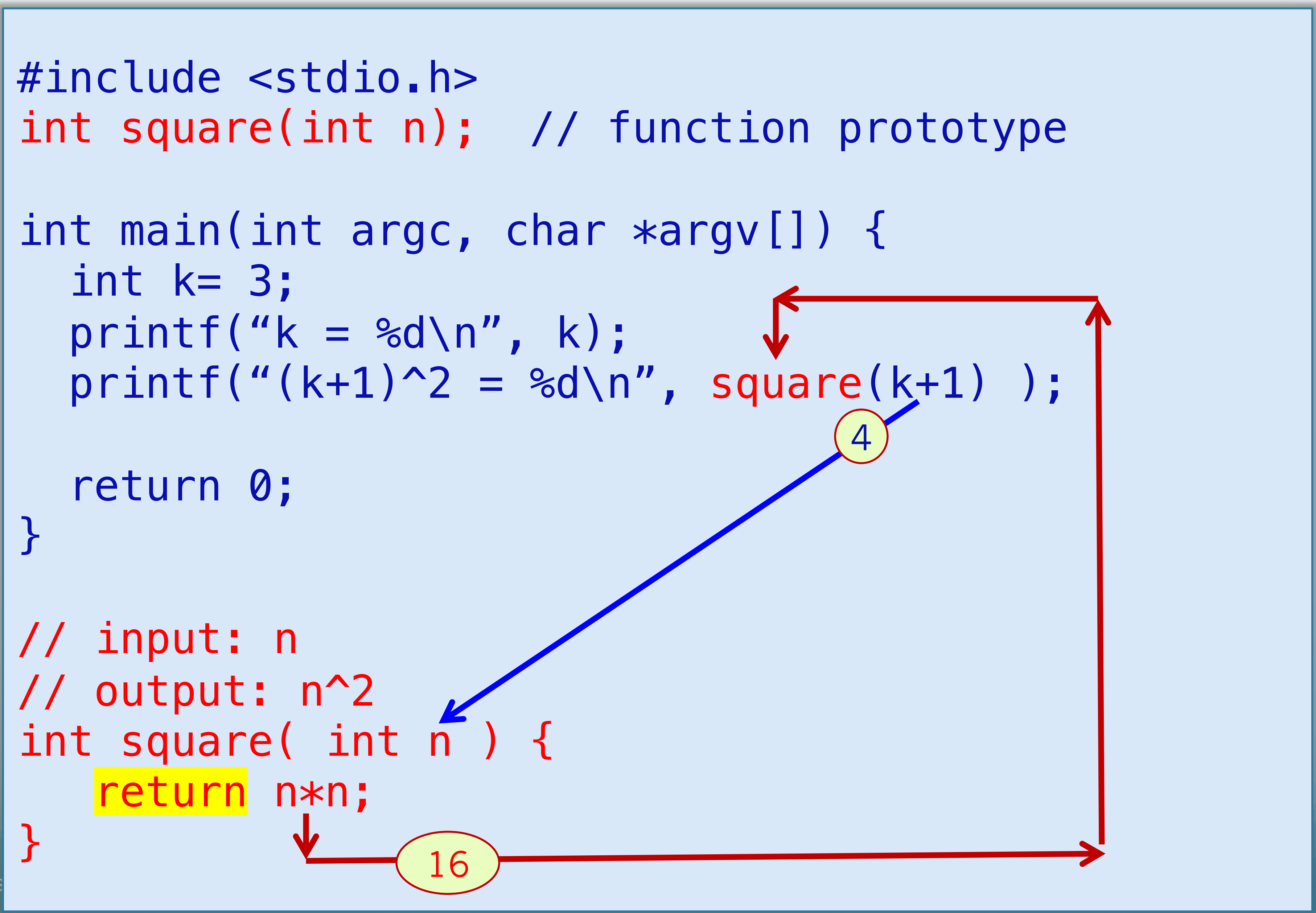
we also write our own *user-defined functions*



It’s your function. It has name, inputs and at most one output (just like library functions).

User-defined functions & program structure

my_prog.c



```
#include <stdio.h>
int square(int n);  // function prototype

int main(int argc, char *argv[]) {
  int k= 3;
  printf("k = %d\n", k);
  printf("(k+1)^2 = %d\n", square(k+1) );

  return 0;
}

// input: n
// output: n^2
int square( int n ) {
   return n*n;
}
```

how to write a function

First, write the function header, therefore determine:

```
int          square        (   int n )
```

data type of function result	name of the function	list of inputs, for each: • name • data type

Then, design the algorithm and write the function body: use the *inputs*, compute and *return* the result.

Do-Together Ex 5.01, then start Ex 4.09

Time for fun: Quick Test

Note: *In all questions, all variables are pre-declared as* `int`.
Q1: What are the values of `s`, `i`, and `c` after the following statement:

```
for (s=0, i=0, c= 0; i<5; i++) {
   s += i;
   c++;
}
```

A	`s= 10, i= 5, c= 5`
B	`s= 10, i= 5, c= 4`
C	`s= 15, i= 6, c= 6`
D	none of the above

Time for fun: Quick Test

Q2: How many lines and numbers are printed by the following segment:

```
for (i=0; i<2; i++) {
  for (j=0; j<3; j++) {
    printf(“%d “, i*j);
  }
  printf(“\n”);
}
```

A	6 lines, 6 numbers
B	3 lines, 12 numbers
C	2 lines, 6 numbers
D	none of the above

Time for fun: Quick Test

Q3: How many lines and numbers are printed by the following segment:

```
for (i=0; i<5; i++) {
  for (j=0; j<4; j++) {
    if ( j >i ) continue;
   printf(“%d “, i*j);
  }
  printf(“\n”);
  if (i==2) break;
}
```

A	5 lines, 20 numbers
B	3 lines, 6 numbers
C	2 lines, 3 numbers
D	none of the above

Time for fun: Quick Test

Q4: Which fragment compute `s=` $1^2 + 2^2 + \ldots + n^2$?

A	`i=0;`
	`while (i<=n) s += i*i;`

B	`for (s=0; n > 0; n--) s += n*n;`
C	`for (i=1; i<=n; i++) s += i*i;`
D	`for (s=0,i=1; i<n; i++) s = s + i*i;`

Lab:

Implement in the order: `4.05 4.09 4.04 4.11-using-function 5.02 5.03`

min requirement: finish `4.05`, `4.09`

Remember:

- Discuss with your classmates
- Ask Anh questions and/or tell him some exciting things

The Coming Quiz 1:

- Do the practice test as many times as possible (you'll probably have different questions each time)
- Bring doubted questions to the next week workshop
- Make sure about the Quiz Time, how to start & submit, and don't forget!

Wrap Up

```
for (<I> ;  <?> ;  <U>) {
      //do one iteration
}
```

```
while (<?>) {
      //do one iteration
}
```

In a for loop:

- `<I>`, `<?>`, and `<U>` can be empty

(`?` in `if (?) ...` and `while (?) ...` **cannot** be empty)

```
int foo(int a, int b) {
   // a & b have values
   // no scanf needed
   int output;
   output= ...(using a, b)
   return output;
   // no printf needed
}
```

functions are great for abstraction

- header: formally specifies inputs and output
- body : calculates output from inputs
- `return`: ends function and returns output

library functions:

`sqrt`, `sin`, ... in `#include <math.h>`

Learnt in lectures: `getchar`, `putchar`, `...`