

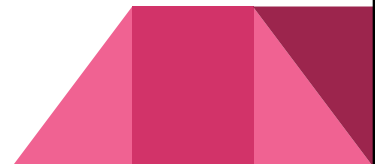


Repetition

For Loops

Any loop has 3 parts

- A setup or initialization.
- A condition that must be true in order to loop.
- Code that executes



FOR Loops

- FOR loops make it easy to include these 3 parts all in the for statement.
- We use for loops when we know how many times the loop needs to be repeated before going into the loop.
 - ie: It's GREAT for counting

Syntax

- ```
for (i=0 ; i<100 ; i++) {
 printf("%i ", i);
}
```
- The parts of a FOR are separated by semi-colon;
- First,  $i = 0$  initializes integer  $i$  to 0
- Second  $i < 100$  the loop will continue as long as  $i < 100$
- Third  $i++$  , increase  $i$  by one each time

## For Example

```
int i;
 Initial Value Condition After each loop
for (i=1; i<=4; i++){
 printf("%i\n", i);
}
return (0);
```

**What will this loop do?**

## For Example

```
int i;

for (i=1; i<=4; i++){ i = 1
 printf("%d\n", i);
}
return (0);
```

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
```

```
}
```

```
return (0);
```

**i = 1**

output

1

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
```

```
}
```

```
return (0);
```

**i = 2**

output

1

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
```

```
}
```

```
return (0);
```

**i = 2**

output

1

2

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
```

```
}
```

```
return (0);
```

**i = 3**

output

1

2

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
```

```
}
```

```
return (0);
```

**i = 3**

**output**

1

2

3

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
```

```
}
```

```
return (0);
```

**i = 4**

**output**

1

2

3

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
}
return (0);
```

**i = 4**

output

1  
2  
3  
4

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
}
return (0);
```

**i = 5**

output

*Loop is finished! i <= 4 is FALSE.*

1  
2  
3  
4

## For Example

```
int i;
```

```
for (i=1; i<=4; i++){
 printf("%d\n", i);
}
```

```
return (0);
```

*Loop is finished! i <= 4 is FALSE.*

**i = 5**

**output**

1

2

3

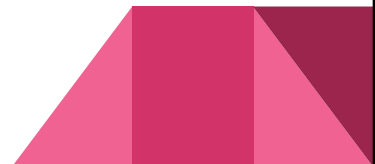
4

## Counting by Twos

- While “i++” is most common, you *can* put **any** expression after the 2nd semi-colon to change i. You can even count backwards, or count by letters.

```
□ for (i=0; i<100; i+=2) {
 printf(“%d “, i);
}
```

- Will print the even numbers starting at 0 until 98.
- How many times does the loop execute?





# Meaningful Variable Names

While we usually want explicit variable names. FOR loops there is a convention of using variables *i*, *j* and *k*, or *x*, *y*, and *z*

The code is easier to read than:

```
for (counter=1 ; counter<10 ; counter++)
```

## Run-time decisions

- While we need to know how many times to loop this can be determined during the running of the program.

- Example:

```
scanf("%d", &iMax);
scanf("%d", &iMin);
scanf("%d", &iStep);
for (i=iMin; i<=iMax; i+=iStep)
 ...
```

## Looping zero and many times

- The code in an for loop may not get executed.
  - `for (i=5; i < 4; i++)`
- Or it may loop forever
  - `for (i=0; i <100; i--)`
- However it is okay to go backwards
  - `for (i=99; i>=0; i--)`

## Syntax problems

- `for (i=0; i<10; i++);  
printf("%d\n", i);`
- `for (i=0, i<10, i++)  
printf("%d\n", i);`
- `for (i=0; i<10; i++)  
printf("%d ", i);  
printf(" time through loop");`

## More on the FOR.

- It is poor style to modify the integer we use to control the for loop within the loop.
- We can “break” out of the loop by using the break statement.
- We can force a loop to immediately “continue” to force a skip and jump to the next iteration of the loop.

## Nested fors

- It is common to nest for loops.
- `// print a grid`

```
int x; char y;
for (x=1; x<10; x++) {
 for (y='A'; y<'I'; y++) {
 printf("%d%c ", x, y);
 }
 printf("\n");
}
```

`// yes you can count by letters.`

# Tracing through a for loop

- A common test question will ask you how many times a loop executes.
- The questions are small enough just to write down the values of the control loop.
- e.g. for (i = -6; i <14; i+=5)
  - start at -6, continue as long as i less than 14

-6      -1      4      9      is 14 less than 14?

