
UNSW
SYDNEY
Australia's
Global
University
COMP1911 - Computing 1A

sort3

- sort3.c reads 3 integers and prints them in from largest to smallest.

```
int a, b, c;
scanf("%d %d %d", &a, &b, &c);
if(a>=b && b>=c) {
    printf("%d %d %d\n", a, b, c);
}
if(a>=c && c>b) {
    printf("%d %d %d\n", a, c, b);
}
if(b>a && a>=c) {
    printf("%d %d %d\n", b, a, c);
}
...
```

```
int a, b, c;
int tmp;
scanf("%d %d %d", &a, &b, &c);
if(b<c) {
    tmp=b; b=c; c=tmp;
}
if(a<b) {
    ...
}
...
printf ("%d %d %d\n", a, b, c);
```

6. While Loops

In this lecture we will cover:

- While Statements/Loops
- Conditions: Loop Counters
- Conditions: Sentinel Variables
- Infinite Looping
- Nested While Loops

While Statements

- We often need to execute code (statements) many times.
- **if** statements only allow us to execute or not execute code. In other words they allow us to execute code 0 or 1 times **while** statements allow us to execute code 0 or more times
- Like **if**, **while** statements have a controlling expression but **while** statements execute their body until the controlling expression is false

```
while (EXPRESSION) {
  stmt1;
  stmt2;
  ...
  stmtn;
}
```

while Loop - Loop Counter Example

Often use a **loop counter** variable to count loop repetitions. We then have a **while** loop execute **n** times.

```
// read an integer n
// print numbers from 0..n
int i, n;

printf("Enter a value for n: ");
scanf("%d", &n);

//initialise our loop counter, i.
i = 0;
while (i < n) {
   printf("%d ",i);
   i = i + 1;
}
printf("\n");
```

while Loop - Loop Counter Pattern

Here is the programming pattern for a while that executes *n* times:

```
//initialise loop counter
loopCounter = 0;
while (loopCounter < n) {
    //
    // statements the loop needs to perform
    //

    // increment loop counter
    loopCounter = loopCounter + 1;
}
```

while Loop - Another Loop Counter Example

```
// read an integer n
// print n asterisks
int i, n;

printf("How many asterisks? ");
scanf("%d", &n);

i = 0;
while (i < n) {
    printf("*");
    i = i + 1;
}
printf("\n");
```

Question: How could we change it to make each * be on its own line? i.e. a vertical column of asterisks instead of a horizontal row of asterisks?

How could we change it to make each * be on its own line?

while Loop - Sentinel Variable Pattern

Sometimes instead of a loop counter, we use a sentinel variable. A sentinel variable is a variable that is used to stop a while loop when a condition occurs in the body of the loop.

```
// read numbers, printing them out
// stop if zero read
int stopLoop, numbers;

 stopLoop = 0;
 while (stopLoop != 1) {
     scanf("%d", &number);
     if (number == 0) {
         stopLoop = 1;
     } else {
         printf("%d\n", number);
     }
 }
```

while Loop - Sentinel Variable Pattern

Here is the programming pattern for a while that executes *n* times:

```
stopLoop = 0;
while (stopLoop != 1) {
    //
    // statements the loop needs to perform
    //
     if (.......) {
         stopLoop = 1;
     }
    //
    // perhaps more statements
    //
}
```

While Statements - Termination

Easy to write while loops that do not terminate.

- A classic example is forgetting to update the loop counter

```
// i never gets updated so is always 0
// this is an infinite loop!
i = 0;
while (i < n) {
   printf("*");
}
```

- Note: You may need to hit Ctrl C to stop your program if you accidentally do this.

Nested While Loops

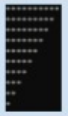
- Often need to nest while loops.
- Need a separate loop counter variable for each nested loop.

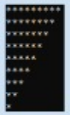
```c
//print a square of 10x10 asterisks
#include <stdio.h>

int main(void) {
    int i, j;

    i = 0;
    while (i < 10) {
        j = 0;
        while (j <10) {
            printf("*");
        j = j + 1;
        }
        printf("\n");
        i = i + 1;
    }
    return(1);
}
```

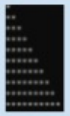
UNSW SYDNEY

What is the output?

```
int i, j;

i = 0;
while (i < 10) {
    j = 0;

    while (j < i ) {
        printf("*" );
        j = j + 1;
    }

    printf("\n");
    i = i + 1;
}
```







None of the above

Total Results: 0

What is the output?

```
int i, j;

i = 0;
while (i < 10) {
    j = 0;

    while (j <10-i) {
        printf( "*" );
        j = j + 1;
    }

    printf("\n");
    i = i + 1;
}
```

A

B

C

D

None of the above

E

Total Results: 0

Looping Summary

- C has other looping constructs - but **while** is all you need
- **for** loops can be a little more concise/convenient, we'll see them later - for now use **while**
- Often use a **loop counter** variable to count loop repetitions
- Can then have a **while** loop execute **n** times.

Questions