

CSc 352

C - Conditions, Loops

Benjamin Dicken

```
#include <stdlib.h>
#include <stdio.h>

int main() {
    int x = 0;
    scanf("%d", &x);
    if (x > 100) {
        fprintf(stderr, "BAD INPUT\n");
        return 1;
    } else if (x > 80) {
        printf("large\n");
    } else if (x > 50) {
        printf("medium\n");
    } else {
        printf("small\n");
    }
    return 0;
}
```

```
#include <stdlib.h>
#include <stdio.h>
```

```
int main() {
    int x = 0;
    scanf("%d", &x);
    if (x > 100) {
        fprintf(stderr, "BAD INPUT\n");
        return 1;
    } else if (x > 80) {
        printf("large\n");
    } else if (x > 50) {
        printf("medium\n");
    } else {
        printf("small\n");
    }
    return 0;
}
```

if / else if / else chains also looks the same as Java

A thruthy / nonzero value will cause a true condition

A falsy / zero value will cause a false condition

How to print to stderr,
Also note return value

The diagram consists of four arrows pointing from the explanatory text on the right to the code on the left. One arrow points from the first explanatory text to the 'if' statement. Another arrow points from the second explanatory text to the 'return 1;' statement. A third arrow points from the third explanatory text to the 'fprintf(stderr, ...)' statement. A fourth arrow points from the fourth explanatory text to the 'return 0;' statement.

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
int main() {  
    int i = 0;  
    while (i < 20) {  
        printf("counting: %d\n", i);  
        i += 1; // or i++;  
    }  
    return 0;  
}
```

```
#include <stdlib.h>
#include <stdio.h>
```

While functions basically the same
as Java



```
int main() {
    int i = 0;
    while (i < 20) {
        printf("counting: %d\n", i);
        i += 1; // or i++;
    }
    return 0;
}
```

A thruthy / non-zero value will
cause loop to continue

A falsy / zero value will stop the
loop

++ and += both work in C



```
#include <stdlib.h>
#include <stdio.h>

int main() {
    for (int i = 0; i < 20; i++) {
        printf("counting: %d\n", i);
    }
    return 0;
}
```

For loops also work!

```
#include <stdlib.h>
#include <stdio.h>
```

```
int main() {
    for (int i = 0; i < 20; i ++ ) {
        printf("counting: %d\n", i);
    }
    return 0;
}
```

3 components within parentheses:

- (1) Run before first iteration
- (2) Condition checked before each iteration
- (3) Code executed at end of each iteration

Tribonacci Numbers

Write a C program that:

- Asks the user for one numeric value **N** from standard input
- Prints out the first **N** numbers in the Tribonacci sequence
- The Tribonacci sequence is defined by:
 - First three numbers in sequence: 0, 0, 1
 - Fourth and on: Defined by the sum of the previous three numbers in the sequence
 - 0, 0, 1, 1, 2, 4, 7, 13, 24 . . .
- Use conditions / loops, not recursion!

What will happen?

```
#include <stdlib.h>
#include <stdio.h>

int main() {
    int x;
    printf("%d\n", x);
    return 0;
}
```

Value initialization

- C does not automatically initialize data of primitive types (Java too)
 - In Java - will give an error if you try to use one of these variables
 - In C - Unpredictable value
- C also does not auto-init array values, unlike Java

```
#include <stdio.h>
```

```
int main() {
```

```
    double x = 1.3928304980192481234;
```

```
    double y = 1.3928304980192481234;
```

```
    double pi = 3.141592653589793238;
```

```
    y = y * (pi / 180.0);
```

```
    y = y * (180.0 / pi);
```

```
    if (x == y) {
```

```
        printf("same\n");
```

```
    } else {
```

```
        printf("different\n");
```

```
    }
```

```
    return 0;
```

```
}
```

What will
this print?

Floating-point Imprecision

- Recall: floats and doubles are not infinitely precise!
- When checking if two floats are equal, can compare with a margin or error

Man pages

Look at the man pages for **scanf** and **printf**

What header file do those come from?

```
#include <stdio.h>
```

```
int main() {
```

```
    int x = 5;
```

```
    float y = 10.17;
```

```
    x = (int) y;
```

```
    y = (float) x;
```

```
    printf("x: %d\n", x);
```

```
    printf("y: %.5f\n", y);
```

```
}
```

What will
this print?