Some C

September 1, 2016

Last Time

- compiling a C program
- big differences between languages
- preprocessor, compiler, assembler, linker
- data sizes
- printf( ), and its placeholders
- some things that might feel slightly weird:
  - char as a kind of integer
  - integers as booleans
- things that are completely the same (or at least almost):
  - operators
  - if if-else switch for while do-while

---

### boolean

<table>
<thead>
<tr>
<th>expression</th>
<th>Java result</th>
<th>C result</th>
</tr>
</thead>
<tbody>
<tr>
<td>10&lt;20</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>10&gt;20</td>
<td>false</td>
<td>0</td>
</tr>
</tbody>
</table>

means you can have things like

```c
if (1) {
    /* always runs */
}
```

```c
if (0) {
    /* never runs */
}
```

```c
int i=50;
while (i) {
    i--;
}
```
but also means ...

```c
1 int x=10, y=20;
2
3 if (x=y) {
4    printf("equal\n");
5 } else {
6    printf("not equal\n");
7 }
```

Output
```
equal
```

so then you get confused, angry and add

```c
1 int x=10, y=20;
2
3 if (x=y) {
4    printf("equal\n");
5 } else {
6    printf("not equal\n");
7 }
8
9 printf("x=%d, y=%d\n", x, y);
```

```c
1 int x=10, y=20;
2
3 if (x=y) {
4    printf("equal\n");
5 } else {
6    printf("not equal\n");
7 }
8
9 printf("x=\%d, y=\%d\n", x, y);
```

Output
```
equal
```

```
x=20, y=20
```
int x=10, y=20;
if (x=y) {
    printf("equal\n");
} else {
    printf("not equal\n");
}
printf("x=%d, y=%d\n", x, y);

What's Happening?

- assignment then test
- no compiler error

Things that are almost completely the same in C and Java

- if, if-else
- for, while, do-while
- switch though not Strings.
- operators +, -, mostly
- comments: mostly
  - /* supported everywhere */
  - // mostly supported

Arrays

OK
- int A[5];
- int A[]={10,20,30,40,50};

Not OK
- int A[];
- int []A={10,20,30,40,50};
Arrays

OK

- int A[5];
- int A[]={10,20,30,40,50};

Not OK

- int A[];
- int []A={10,20,30,40,50};

Legal but will get you in trouble

- A[-3]=5;

Difference: Auto-Initialization

Java: what's in the array now?
- int A[] = new int[5];

What about C?
- int A[5];

Arrays

- What do we pass when we pass an array in Java?
  - size
    - no .length field
    - pass length with array
More on Arrays

In Java

- what's stored in an array variable?
- what happens when we pass to a method?

In C, exactly the same idea

```java
import java.util.Arrays;

public class Array1 {
    public static void func(int A[]) {
        for (int i=0; i<A.length; i++)
            A[i]*=2;
    }

    public static void main(String args[]) {
        int A[] = {10,20,30,40,50};
        func(A);
        System.out.println(Arrays.toString(A));
    }
}

Output

[20,40,60,80,100]
```
What's Printed

```java
import java.util.Arrays;

public class Array2 {
    public static void func(int A[]) {
        int B[] = new int[A.length];
        for (int i=0; i<A.length; i++)
            B[i]=2*A[i];
        A=B;
    }

    public static void main(String args[]) {
        int A[] = {10,20,30,40,50};
        func(A);
        System.out.println(Arrays.toString(A));
    }
}
```

Output

```
[10,20,30,40,50]
```

What's in the array after `func()`?

```c
#include <stdio.h>

void func(int A[], int)
{
    func(A, 5);
    return 0;
}

void main(int argc, char **argv)
{
    func(A, 5);
    return 0;
}
```

Answer

```
[20,40,60,80,100]
```
So far, only `printf()`.

We’ll do things like Java’s `Scanner` later.

---

**Getting a character at a time**

```
int getchar();
```

- reads the next character from `STDIN`
- returns
  - `success`: the next character
  - `failure`: `EOF`

---

**Where do I learn more about these things?**

- `man` pages!
- tells
  - C library functions, e.g., `man getchar`
  - Unix shell commands, e.g., `man ls`
  - the manual itself, e.g., `man man`

---

**Writing a single character to the screen**

Could still use `printf()`. There’s also:

```
int putchar(int c);
```

- writes the character `c` to `STDOUT`
- returns
  - `success`: 0
  - `failure`: `EOF`
Examples

Using shell redirection (< and > operators):
- cat
- char counter
- line counter
- word counter