

Intro to Programming II

More C

Chris Brooks

Department of Computer Science

University of San Francisco

20-2: Exercise 2

- Write a C program that takes the string “Oh how I love to program in C” and prints out all the vowels.

20-3: More on printing

- `printf` is a bit different from `println()` in Java.
- Takes an arbitrary number of arguments.
- First argument is the string to print.
 - It may contain control characters.
- Remaining arguments are values to replace in the string.

20-4: More on printing

```
printf(``hello world``);  
printf(``hello %s``, ``bob``);  
printf(``The first number is %d and the second number is %lf``, i, j)  
printf(``average: %2lf total %4.2lf``, i, j);
```

20-5: Pointers

- The biggest difference between C and Java is the use of *pointers*.
- A pointer is the actual address that a variable is stored at.

```
int
main(void) {
    char *testStr = "hello world";

    printf("%s", testStr);
    printf("%d", *testStr);
    printf("%d", *testStr + 3);
}
```

20-6: Pointers

- To declare a pointer to a variable, use `*`
 - `char *hello = "hello world";`
- To get at the data, use the variable name
 - `printf("%s", hello);`
- To *dereference* the data and use the address, add a the `'*` to the front of the variable name.
 - This produces an integer.
 - `printf("%d", *hello);`

20-7: Dynamically allocating objects

- In Java, you allocate objects by using `new`.
- In C, you use `malloc`
 - Big difference: you need to specify the size of the memory chunk you want and the number of chunks.

```
/* make an array of 10 integers */  
int *i = (int *)malloc(10, sizeof(int));
```

20-8: Exercise 3

- Write a C program called `fillArray` that:
 - prompts the user for a number of integers
 - Uses `malloc` to allocate an array of that size
 - prompts the user for each number
 - stores each number in the array
 - prints out the average of those numbers

20-9: Deallocating memory

- In Java, the garbage collector cleans up objects once you're done using them.

```
public void foo() {  
    String s = new String();  
    ...  
}
```

- When the method `foo` exits, the garbage collector realizes that `s` is no longer being used.
- The memory `s` used is reclaimed automatically.

20-10: Deallocating memory

- C does not have a garbage collector.
- Instead, dynamically allocated memory must be freed up when your program is done with it.
- This is done using `free`.
- Forgetting to free no-longer-used memory can cause a *memory leak*.

20-11: Deallocating memory

```
int
main(void) {
    int *array = (int *)malloc(10 * sizeof(int));
    int j;
    for (j = 0; j < 10; j++) {
        array[j] = j;
    }
    free(j);
}
```

20-12: Exercise 4

- Add a free statement to your fillArray program.
- Change fillArray to allocate and read in an array of doubles.

20-13: Pass by value

- Function calls can pass parameters in two ways:
- pass-by-value: a copy of the object is passed.
- Any changes made inside the function affect only the local copy.
- pass by reference: the address of a variable is passed into the function.
- changes made inside the function are reflected outside the function.

20-14: Parameters in Java

- In Java, primitives are passed by value.
- With references, it's a little trickier.
 - For objects, a copy of the reference is passed.
 - This means that methods called on that reference affect the same global object.
 - But, if the reference itself is changed, only the local copy is affected.

20-15: Parameters in C

- In C, things are more straightforward.
- Variables are passed by value.
- To pass by reference, you can provide a reference to the variable using &.

20-16: References vs Pointers

- Let's start with a variable:
 - `int x`
- To refer to the address that `x` is stored at, we use the address operator
 - `&x`
- To create a pointer to an integer, we use the `*` operator:
 - `int *iptr;`
- `iptr` is a variable of type `int *` - that is, it's the address of an integer.
- So, we can do:
 - `iptr = &x;`
 - (the variable `iptr`, which is an address of an `int`, is set to the address of `x`, which is an `int`.)

20-17: References vs Pointers

- So, & is used to get from a variable to its address.
 - scanf
 - calling a function with pass by reference
- * is used when you want to start with an address, and then create a variable.
 - dynamically allocating memory
 - being called with pass by reference

20-18: Exercise 5

- Write a function called `swap`.
 - It should take as arguments pointers to two integers.
 - It should exchange their values.
- Then write a main that calls this function with references to two variables.