More on Strings, File I/O

September 8, 2016

Administrative Stuff

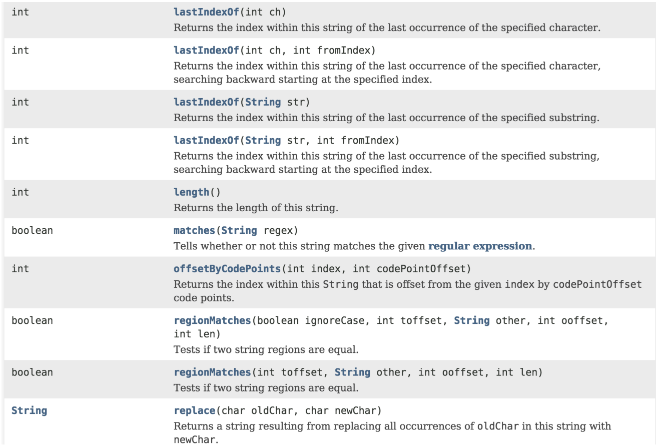
- Assignment 2 posted soon
- Please make sure that you're getting comfortable with *some* C development environment.
- ACM

Last Time

- word count
  - finite state machines
  - turning the diagram into running code
  - more practice using shell redirection ops $<$, $>$.
- more on arrays
- more on Java references
- refresher on Java Strings
- lots on C strings

Java API

http://docs.oracle.com/javase/8/docs/api/



int	lastIndexOf(int ch) Returns the index within this string of the last occurrence of the specified character.
int	lastIndexOf(int ch, int fromIndex) Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
int	lastIndexOf(String str) Returns the index within this string of the last occurrence of the specified substring.
int	lastIndexOf(String str, int fromIndex) Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
int	length() Returns the length of this string.
boolean	matches(String regex) Tells whether or not this string matches the given **regular expression**.
int	offsetByCodePoints(int index, int codePointOffset) Returns the index within this String that is offset from the given index by codePointOffset code points.
boolean	regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len) Tests if two string regions are equal.
boolean	regionMatches(int toffset, String other, int ooffset, int len) Tests if two string regions are equal.
String	replace(char oldChar, char newChar) Returns a string resulting from replacing all occurrences of oldChar in this string with newChar.

etc.

C string API

- `string.h`
- finding what's there:
  - man pages
  - Google
  - appendix of K&R
- many other APIs exist
- a sample:

```
     ...
char *stpcpy(char *dst, const char *src);
char *strcat(char * s1, const char *s2);
char *strchr(const char *s, int c);
int strcmp(const char *s1, const char *s2);
char *strcpy(char *s1, const char *s2);
size_t strcspn(const char *s1, const char *s2);
char *strerror(int errnum);
size_t strlen(const char *s);
char *strncat(char * s1, const char *s2, size_t n);
     ...
```

Experimenting with Strings

- creating with double quotes.
  - printing
  - modifying individual characters
  - is `'\0'` really there?
- creating with array notation
  - what happens when we forget `'\0'`?

Fake Quiz

Write a method called `copy_A( )`, which is passed three arguments:

- `int A[]`
- `int B[]`
- `int len`

The method should copy the first `len` items from `A` to `B`.

What Happens?

What happens if we try this:

```
#define CAPACITY 100

...

int A[CAPACITY];
int B[CAPACITY/2];

...


copy_A(A, B, CAPACITY);
```

Same Problem. What happens here?

```
#define CAPACITY 10

...

char s[CAPACITY];

...


strcpy(s, "There’s always money in the banana stand.");
```

Practice Problem

- Implement `strcat`

Practice Problem

- Implement `strcat`
- What happens when the destination array isn't long enough?

Practice Problem

- Implement `strcat`
- What happens when the destination array isn't long enough?
- `strncat`

strncat( )

```c
void strncat(char d[], char s[], int n) {
  int i;

  int dlen = strlen(d);
  for (i=0 ; i<n && s[i]!='\0'; i++)
     d[dlen + i] = s[i];
  d[dlen + i] = '\0';
}
```

strncat( )

```
void strncat(char d[], char s[], int n) {
  int i;

  int dlen = strlen(d);
  for (i=0 ; i<n && s[i]!='\0'; i++)
     d[dlen + i] = s[i];
  d[dlen + i] = '\0';
}
```

- Stop when:
  - we've hit the end *OR*
  - we've run out of space
- similar bounds-checking equivalents for others, *e.g.*, `strncpy()`

files

Like Java, multiple steps:

- open
- use
- close

files

Like Java, multiple steps:

- open
  - `fopen( )`
- use
  - many, many functions
- close
  - `fclose( )`

fopen

```
FILE *fopen(char path[], char mode[]);
```

- `path`
  - the name of the file. Just like you'd pass to the `File` constructor in Java
- `mode`
  - "r" for reading
  - "r+" for reading and writing
  - "w" for writing. start writing at the beginning of the file
  - . . . and others

fopen

```
FILE *fopen(char path[], char mode[]);
```

- `path`
    - the name of the file. Just like you'd pass to the `File` constructor in Java
- `mode`
    - "r" for reading
    - "r+" for reading and writing
    - "w" for writing. start writing at the beginning of the file
    - ... and others

returns

- a `FILE*` on success
- `NULL` on failure

FILE*

- What's a FILE*?
- Who cares? Just a handle:
  - returned from `fopen`
  - passed to functions that operate on files.
- Remember that we really didn't care what was inside a `Scanner` object either

`fgetc`, `fputc`

reading a single char

- **from kbd**: `int getchar()`
- **from file**: `int fgetc(FILE *fp)`

`fgetc`, `fputc`

reading a single char

- **from kbd**: `int getchar()`
- **from file**: `int fgetc(FILE *fp)`

writing a single char

- **to screen**: `putchar(int c)`
- **to file**: `fputc(int c, FILE *fp)`

fclose()

A Simple File Reader

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

#define EXIT_FILE_OPEN_FAILURE 1

int main(int argc, char **argv) {
  int c;
  char filename[]="stuff.txt";
  FILE *fp;

  if ((fp=fopen(filename, "r"))==NULL) {
    printf("error opening %s for reading. Quitting.\n", filename);
    return EXIT_FILE_OPEN_FAILURE;
  }

  while ((c=fgetc(fp))!=EOF) {
    putchar(c);
  }

  printf("\nDone\n");
  fclose(fp);
  return EXIT_SUCCESS;
}
```

A Simple Copier

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

#define EXIT_FILE_OPEN_FAILURE 1

int main(int argc, char **argv) {
  int c;
  char input_filename[]="instuff.txt";
  char output_filename[]="outstuff.txt";
  FILE *infp;
  FILE *outfp;

  if ((infp=fopen(input_filename, "r"))==NULL) {
    printf("error opening %s for reading. Quitting.\n", input_filename);
    return EXIT_FILE_OPEN_FAILURE;
  }

  if ((outfp=fopen(output_filename, "w"))==NULL) {
    printf("error opening %s for writing. Quitting.\n", output_filename);
    return EXIT_FILE_OPEN_FAILURE;
  }

  while ((c=fgetc(infp))!=EOF) {
    fputc(c, outfp);
  }

  printf("Done\n");
  fclose(infp);
  fclose(outfp);

  return EXIT_SUCCESS;
}
```