Intro to Programming II
Project Recap

Chris Brooks

Department of Computer Science
University of San Francisco
24-0: Recapping what you need for Project 5

- Working with 2-dimensional arrays.
- Reading from files.
- Command line arguments
- Program structure
Multidimensional Arrays
What if we don’t know ahead of time how big our array should be?

Then we need to use malloc to allocate memory on the fly.

In this case, we treat our 2D array as an array of pointers (or, an array of arrays)

```c
int **intArray;
```

intArray is a pointer to an array of pointers.
We start out using malloc as usual (almost):

```c
int **intArray = (int **)malloc(10 * sizeof(int *));
```

We used malloc to create an array of 10 int pointers

But, none of those pointers point to anything yet.

We have to go through and use malloc to allocate space for each of those arrays as well.

```c
for (i = 0; i < 10; i++)
    intArray[i] = (int *)malloc(10 * sizeof(int));
```
Let’s think a bit about what the board struct for project 5 should look like.

It needs:
- number of rows
- number of columns
- a representation of the board. (a two-dimensional array)
It should probably look something like:

```c
typedef struct {
    int nrows;
    int ncols;
    int **boardArray;
} board;
```
We’re now ready to write `createRandomBoard`.

It should take three arguments: a pointer to a board, a number of rows, and a number of columns.

It should allocate memory for the `boardArray` and then fill in each cell with a random value (0 or 1).

Write this function in `board.c`, and then write a small main in `gameOfLife.c` that creates a board and calls this function.
The next step is to write printBoard.

This should take a board as input and print it out to the screen.

I’d suggest printing ’1′ for cells containing 1, and ’ ’ in cells containing 0.
File I/O
24-9: Opening files for reading

- fopen() opens a file and returns a file pointer.

- It takes two arguments:
  - The file name
  - A string indicating whether we’re opening for reading or writing.
For example

```c
FILE *fp = fopen("myfile","r");
```

opens myfile for reading.
fscanf is used to read from a file.

Works exactly like scanf, except that the first argument is the file pointer.

You can also use getc - it returns the integer representing the next character in the file.

getc() returns EOF if you’re at the end of the file.
To open a file for writing, use the "w" argument to fopen.

```c
fopen("foo","w");
```

You can then write to a file with:

- `fprintf(FILE *fp, char *stringToPrint, arg1, arg2 ...)`
- `putc(intc, FILE *fp)`
Command Line Arguments
To provide command line arguments, we need to specify that main() will receive arguments.

- Remember, main is just another function.

It takes its arguments in a special form:

```c
int
main(int argc, char **argv)
```

- argc is the number of command line arguments
- argv is an array of strings, one for each argument.
- argv[0] is the name of the program.
Modify cat to read in the file indicated by the first argument and write it out to the file indicated by the second argument.
24-16: Converting strings to ints

What if you need to convert a string to an integer?
• Say you want to provide an integer as a command line argument?

atoi(char *s) will return an integer representation of the string s.
Modify the multiplication program from Wednesday to take three command line arguments:

- The number of rows
- The number of columns
- A file to write the table to.
You should now have the necessary tools to do the following parts of project 5:

- createRandomBoard()
- readBoardFromFile()
- printBoard()
- getAliveNeighbors()
- isValidCell()
- updateBoard()
- main

I’d suggest starting with createRandom board, readBoardFromFile, and printBoard. Then start on your main, so that you can test each of these.
What would a structure chart for this program look like?