Post lock-down programming

ESC101: Fundamentals of Computing

Nisheeth

Announcements

- Where to begin?
- Its been a loooooong midsem break
- Very unlikely that this semester's classes will resume in-person
- I am uploading slides, with lecture notes where appropriate, to the course website
- Very unlikely that any of this material will be graded
- Therein lies opportunity ……

Opportunity

- The second half of ESC101 has always been painful on the regular schedule for people new to programming
- Too many new concepts come at you in a very short time-frame
- Now we can take our time with them
- All slides posted to the course website
  - Along with some practice problems
- I will do Zoom broadcasts 1-3 times a week (depending on number of people interested)
  - Will mostly solve practice problems in real-time during the broadcast
  - Will take questions from attendees
  - None of this will be graded, so no performance pressure
- Zoom meeting link (also posted on course webpage)

Recap

In the beginning, there was nothing	printf(“Hello World”);
Then we discovered simple data types	printf(“Hello %d”, number);
We learned how to play with them using logic	if(temp>99) screamf(“infected!”);
We learned to break up code into little bits	a = simple_interest(400,5,2);

So far, everything was simple and intuitive

Then we discovered arrays and pointers!	printf(“%p”, &a[2], a+2);
And life became complicated (and beautiful)	printf(“%d”, a[2], *(a+2));

ESC101: Fundamentals of Computing

Recap

When you talk to a computer, you are talking to a library with a clear indexing scheme.

Can ask for entries by title, or by location.

Massive flexibility in what you can ask for

```
printf("%p", &a[2], a+2);

printf("%d", a[2], *(a+2));
```

Print address of third element in array a

ESC101: Fundamentals of Computing

Recap

When you talk to a computer, you are talking to a library with a clear indexing scheme.

Can ask for entries by title, or by location.

Massive flexibility in what you can ask for

```
printf(“%p”, &a[2], a+2);

printf(“%d”, a[2], *(a+2));
```

Print address of element two elements away from first element of array a

Recap

When you talk to a computer, you are talking to a library with a clear indexing scheme.

Can ask for entries by title, or by location.

Massive flexibility in what you can ask for

```
printf("%p", &a[2], a+2);

printf("%d", a[2], *(a+2));
```

Print value stored in third element in array a

ESC101: Fundamentals of Computing

Recap

When you talk to a computer, you are talking to a library with a clear indexing scheme.

Can ask for entries by title, or by location.

Massive flexibility in what you can ask for

```
printf("%p", &a[2], a+2);

printf("%d", a[2], *(a+2));
```

Print value stored in element two elements away from first element of array a

We saw what arrays look like, and then discovered pointers and how they decide why arrays look the way they do

ESC101: Fundamentals of Computing

Looking ahead

- We will see how arrays are just pointers in disguise
- We will see how pointers can be used to make more complicated data structures
  - Struct, union, linked lists, stacks, heaps etc.
- We will see how to use algorithms to process complicated data structures more efficiently
  - We will understand what computational efficiency means a little bit

ESC101: Fundamentals of Computing

Course logistics post lock-down

- I had a team of 100 TAs and tutors to help me work with you
- Now its just me
- Won’t be able to answer questions on Piazza regularly
  - Unless it’s a question about some potential mistake in slides
  - Will definitely answer those
- One mostly learns programming by doing
  - The Zoom meetings will focus almost entirely on solving practice problems in real-time
  - Will only help if you have already worked your way through the corresponding slides

ESC101: Fundamentals of Computing

Today’s program

- We saw multi-dimensional arrays a little bit in the last week before the mid-sem break
- Can you write a program to multiply two matrices?

- Printing out a 5x6 2D array

```
int i,j;
    for (i=0; i < 5; i=i+1) {
      for (j=0; j < 6; j = j+1) {
                      printf(“%f ”, mat[i][j]);
      }
      printf(“\n”);
    }
```

```
#include<stdio.h>

int main(void) {
        int i, j, p, q, m, n, k, tot = 0;
        int first[10][10], second[10][10], result[10][10];

        \\ enter matrix sizes and elements

        \\ check if matrices can be multiplied

        if (n != p)
                printf(" Matrix multiplication not possible \n ");
        else {

                \\ can you complete the code?

                for (i = 0; i < m; i++) {
                        for (j = 0; j < q; j++)
                                printf("%d \t", result[i][j] );
                        printf(" \n ");
                }
        }
        return 0;
 }
```