Welcome!

COMP1511 18s1

Programming Fundamentals

COMP1511 18s1

— Lecture 10 —

Strings+Arrarys+Functions

~~Andrew Bennett~~

~~<andrew.bennett@unsw.edu.au>~~

Jashank Jeremy

<jashank.jeremy@unsw.edu.au>

Before we begin…

introduce yourself to the person sitting next to you

why did they decide to study **computing**?

Overview

after this lecture, you should be able to…

understand the basics of working with **getchar**, **putchar**, **fgets**

write programs using **strings** to solve simple problems

have a deeper understanding about **arrays**

have a deeper understanding about **calling functions** and **function parameters**

have a deeper understanding about **passing values** into functions

(**note**: you shouldn't be able to do all of these immediately after watching this lecture. however, this lecture should (hopefully!) give you the foundations you need to develop these skills. remember: programming is like learning any other language, it takes consistent and regular practice.)

Admin

Don’t panic!

assignment 1 due **TONIGHT**

you can do it!

week 5 weekly test due thursday

don’t be scared!

lab marks released

post in class forum || email your tutor

don’t forget about **help sessions**!

see course website for details

remember strings?

a **string** is an **array** of **characters**

```
char name[] = "ANDREW";
```

remember strings?

characters store **ASCII** values

```
char name[] = "ANDREW";
```

is equivalent to

remember strings?

never use the ASCII values directly

```
char name[] = "ANDREW";

// Prints out A
printf("name[0] as a char is: %c\n");

// Prints out 65
printf("name[0] as an int is: %d\n");
```

remember strings?

never use the ASCII values directly

```
int some_letter = 'A';
int another_letter = 65;

assert(some_letter == another_letter);
```

we can acccess the ASCII value for the letter A with ‘A’.

much better to use ‘A’ than 65 – **why**?

letters are just ASCII values are just letters

ASCII values are **sequential**

```
printf("the ascii value for %c is: %d\n", 'A', 'A');
printf("the ascii value for %c is: %d\n", 'B', 'B');
printf("the ascii value for %c is: %d\n", 'C', 'C');
```

letters are just ASCII values are just letters

this means we can do cool things

```
// what will something be?
int something = 'B' - 'A';
```

getchar and putchar

`getchar()`

reads a character from standard input

returns an **int**

`putchar('A')`

prints a character to standard output

let's try it!

using getchar and putchar in a loop

```
while (c != EOF) {
    printf("%c", c);
    c = getchar();
}
```

using getchar and putchar in a loop

```
int c = ????
while (c != EOF) {
    printf("%c", c);
    c = getchar();
}
```

using getchar and putchar in a loop

```
int c = getchar();
while (c != EOF) {
    printf("%c", c);
    c = getchar();
}
```

up next: beyond getchar()

More input

what if we wanted to scan more than one character at a time?

More input

(re-)introducing: **fgets**

More input

`fgets(array, array size, stream)` reads a line of text

array - char array in which to store the line

array size - the size of the array

stream - where to read the line from, e.g. stdin

fgets won't try to store more than **array size** chars in the array

let's try it out!

fgets vs gets

never use the function `gets`! (why?)

man pages + demo

up next: where is everything?

Arrays

what **are** arrays?

Arrays in memory

how are they **stored** in memory?

Arrays in memory

what **else** is stored in memory?

hint: everything!

Everything in memory

why does this matter?

Function memory

variables in a function can only be accessed by that function.

why?

up next: calling functions

Passing values into functions

functions receive a **copy** of the **value** of the function parameter

Passing arrays into functions?

if functions can’t modify anything outside of their function

how do arrays work?

farnarkle.c

Farnarkles

```
int hidden_tiles[N_TILES];
printf("Enter hidden tiles: ");
read_tiles(hidden_tiles);
print_tiles(hidden_tiles);
test_farnarkle_ai(hidden_tiles)
```

farnarkle.c