Welcome!

COMP1511 18s1
Programming Fundamentals

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 chars
understand what ASCII is
understand the basics of strings
write programs using strings to solve simple problems

(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!

course style guide published

week 4 weekly test due friday
don't be scared!

assignment 1 out now
work on it regularly!

additional autotests added to the assignment
don't forget about help sessions!
see course website for details

Beyond Numbers

we've mostly seen numbers thus far

int age = 18;
double pi = 3.14

what else might we want to store?

Letters and Words

what about words?

printf("andrew is awesome");

Letters and Words

what about words?

printf("andrew is awesome");

"andrew is awesome"
Letters and Words

words in C are called **strings**

```c
printf("andrew is awesome");
```

"andrew is awesome"

^ this is a **string**

---

**Strings**

A **string** is an **array** of **characters**.

Note: A **character** is a “printed or written letter or symbol”.

---

**Characters**

A **character** generally refers to a letter, number, punctuation, etc.

In C we call it a **char**
Characters in C

in C we call it a char

```c
// making an int
int age = 18;

// making a char
char letter = 'A';
```

char s go inside single quotes, i.e. ‘’. strings go inside double quotes, i.e. “”.

---

ASCII

ASCII is a way of mapping numbers to characters. It contains:

- upper and lower case English letters: A-Z and a-z
- digits: 0-9
- common punctuation symbols
- special non-printing characters: e.g. newline and space.

---

ASCII

you don’t have to memorize ASCII codes!

single quotes give you the ASCII code for a character:

```c
printf("%d", 'a'); // prints 97
printf("%d", 'A'); // prints 65
printf("%d", '0'); // prints 48
printf("%d", ' ' + '\n'); // prints 42 (32 + 10)
```

don’t put ASCII codes in your program - use single quotes instead!
ASCII

let's try it out!

Reading chars

getchar()
reads a byte from standard input
returns an int
returns a special value if it can't read a byte
otherwise returns an integer (0..255)
(ASCII code)

let's try it out!

getchar

consider the following code:

```
int c1, c2;
printf("Please enter first character:\n");
c1 = getchar();
printf("Please enter second character:\n");
c2 = getchar();
printf("First %d\nSecond: %d\n", c1, c2);
```

what should this do?
what does it actually do?
(how can we fix it?)

int c1, c2;
printf("Please enter first character:\n");
c1 = getchar();
printf("Please enter second character:\n");
c2 = getchar();
printf("First %d\nSecond: %d\n", c1, c2);

what should this do? read two typed characters
what does it actually do? read one typed character + enter
getchar
how can we fix it?

```c
int c1, c2;
printf("Please enter first character:\n");
c1 = getchar();

getchar(); // extra getchar to catch the newline

c2 = getchar();
printf("Please enter second character:\n");
printf("First %d\nSecond: %d\n", c1, c2);
```

End Of Input

scanf or getchar will fail if there isn't any more input
e.g. if you're reading from a file and reach the end of the file

getchar returns a special value to indicate no more input is available
we call this value EOF
(how could you check this with scanf?)

Reading until End of Input

```c
int c;
c = getchar();
while (c != EOF) {
    printf("'%c' read, ASCII code is %d\n", c, c);
    c = getchar();
}
```

Strings

strings are an array of characters

```c
int num;
// scanf returns the number of items read
while (scanf("%d", &num) == 1) {
    printf("you entered the number: %d\n", num);
}
```
Remember Arrays?

A series of boxes with a common type, all next to each other

Arrays in C

// Declare an array with 10 elements
// and initialises all elements to 0.
int myArray[10] = {0};

Character Arrays

we can make an array of chars in the same way

char myArray[10] = {0};
// Put some values into the array.
myArray[0] = 65;
myArray[5] = 70;
How long is a piece of string?
you don't always know the length of a string in advance

e.g. name could be "Andrew", or "Tom" (6 characters vs 3 characters)

```c
// "Andrew" is 6 letters long
name[0] = 'A';
name[1] = 'n';
name[2] = 'd';
name[3] = 'r';
name[4] = 'e';
name[5] = 'w';

// "Tom" is 3 letters long
name[0] = 'T';
name[1] = 'o';
name[2] = 'm';
```

How long is a piece of string?
we need a way to know how long the string is!

```c
```

Null Terminator
we do this in C using a null terminator

any function (e.g. printf) working with a string interprets this as "end of string".

```c
name[0] = 'T';
name[1] = 'o';
name[2] = 'm';
name[3] = '\0';
```

(please never write code on one line like that! it's only here so the slides fit)

printing name would print TOMREW
Null Terminator

<table>
<thead>
<tr>
<th>'T'</th>
<th>'O'</th>
<th>'M'</th>
<th>\0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

printing `name` would now print `TOM`

Sidenote: Uninitialised Arrays

What happens if we don't initialise our array?

Let's try it and see!

```
int array[SIZE];
int i = 0;
while (i < SIZE) {
    printf("%d\n", array[i]);
    i++;
}
```

Sidenote: Uninitialised Arrays

What's wrong with this code?

The array has not been initialised.

```
int array[SIZE];
int i = 0;
while (i < SIZE) {
    printf("%d\n", array[i]);
    i++;
}
```
Sidenote: Uninitialised Arrays

solution: initialise the array first
(note: you could also initialise all the values in a loop)

```c
int array[SIZE] = {0};

int i = 0;
while (i < SIZE) {
    printf("%d\n", array[i]);
    i++;
}
```

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Sidenote: Uninitialised Arrays

dcc can catch this for you if you tell it to use valgrind

dcc -o blah blah.c --valgrind