THE UNIVERSITY OF SYDNEY

SCHOOL OF COMPUTER SCIENCE

Info1910 Cheat Sheet

2025

C

Types in C,

Type	Bits
int	16
long	32

Structs in C,

```
struct frog {
    char* name;
    int ribbits;
};
```

Typedefs, allows for defining custom types

```
typedef int ribbits_per_hour;
```

Arrays, are defined as a list of certain types of elements,

```
#define arr_len 10;
#define str_len 12;
char frogs[arr_len][str_len];
```

Looping through an array,

```
int len = sizeof(frogs) / sizeof(frogs[0]);
for (int i = 0; i < len; i++) {
    printf("i is %d, and frog is %c", i, frogs[i]);
}
```

Can loop through arrays with recursion or for-each loop or while loop.

Strings are just arrays of characters terminated by the null character, '\0'

Can use these functions to act on strings,

- `strlen`, finds length of string including '\0'
- `strcpy` copies a string from `src` to a destination `dest`
- `strcmp` compares two strings can be used for sorting.

Recursion in C involves recalling the called function but always constructing a base case,

```
int fibbonacci(int n) {
   if(n == 0){
    // base case
      return 0;
   } else if(n == 1) {
      return 1;
   } else {
      return (fibbonacci(n-1) + fibbonacci(n-2));
   }
}
```

To pass a pointer to a function in C,

```c
#include <string.h>
char 2nd_char(char* str) {
    if strlen(str) < 1 {
        return '';
    }
    return str[1];
}
```

To sort with `qsort` you need to provide a `base` which is the original array to sort, then size of the array (the number of elements) and the size of each element and then a comparison function.

```c
#include <string.h>
#include <stdlib.h>
char frogs[arr_len][str_len];

// sort the array
void sort_frogs(char frogs[][STR_LEN], int arr_len) {
    return qsort(frogs, arr_len, str_len, )
}

int str_compare(const void* str1, const void* str2) {
    return strcmp((char*) str1, (char*) str2);
}
```

with the `strcmp` providing the algorithm for comparing the strings and ensuring that the strings are in the correct order.

Then to search through array to find a 'needle',

```c
int compare(const void* a, const void* b)
{
    return (*(int*)a - *(int*)b);
}

arr[5] = {1, 2, 3, 4, 5};

int n = sizeof(arr) / sizeof(arr[0]);

int key = 4;
int* item;

item = (int*)bsearch(&key, arr, n, sizeof(int), compare);
// finds the index of the particular item - otherwise returns NULL
```

Arguments to the main function,

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

To interact with files, typically this flow is used,

1. The item is opened
2. The file is read into a buffer
3. The file is acted on

4. The file may be overwritten, appended to or closed.

Opening a file

Using the `fopen` to open a file, use `fopen(...) != NULL` to check exists.

```
#include <stdio.h>
FILE *fp = fopen("file.txt", "r")
```

With "`r`" mode being read mode.

Scanning file stream

Using the `fscanf` to scan in characters,

```
int number;
char word[50];

fscanf(file, "%d %s", &number word);
```

The number returned is the number of elements successfully *scanned* in to the program.

Printing to stream

Prints particular characters with a format to a stream.

```
#include <stdio.h>
fprintf(file_opened, "Name %d %s", n, name);
```

Format Specifier

Specifier	Use
`d`	Decimal
`s`	String (of characters)
`c`	Character
`f`	Floating point number

Pointers

Pointers point to an address in memory and are designated by the `&` syntax.

```
int j = 10;
int* k = &j;
```

The `int*` syntax refers to the fact that k refers to a part of memory which harbours an integer.