Main function

- Core of all C programs!
  - Must be defined
  - Returns 0 (EXIT_SUCCESS) or 1 (EXIT_FAILURE)

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    return EXIT_SUCCESS;
}
```
Output

- Getting into working with stdout
- Format specifiers: why are they important?

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int num = 90;
    printf("%d\n", num);
    return EXIT_SUCCESS;
}
```

C Programming: Section 7.2
Output cont.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int num = 90;
    printf("%c\n", num);
    return EXIT_SUCCESS;
}
```

Now?
Output cont.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int num = 90;
    printf("%i\n", num);
    return EXIT_SUCCESS;
}
```
Format specifiers...

- Just like it sounds, specifies a format for the output.
- Can we specify a format for the input though?

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int num = 0x5A;
    printf("%i\n", num);
    return EXIT_SUCCESS;
}
```
Strings?

- Why do we keep specifying **char** if we’re dealing with strings?
  - What does this tell us about the size (in bytes) of the strings?

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char* intro = "hi\0";
    char intro2[] = "hey!\0";
    char intro3[5] = { 's', 'u', 'p', '!', '\0' };
    printf("%s\n", intro);
    printf("%s\n", intro2);
    printf("%s\n", intro3);
    return EXIT_SUCCESS;
}
```

intro and intro2 don’t technically require \0, they’re automatically null-terminated since they’re string literals!
Structs

- Two basic ways to declare structs
  - Typedef
  - Standard
- Properties (members) that are basic data types (or pointers to them)
- How do we find size (roughly)?
- Does * affect member size?

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int age;
    char* name;
} Person;

struct Pet {
    int age;
    char* name;
};

int main() {
    Person Ryan;
    Ryan.age = 23;
    Ryan.name = "Ryan Good";

    struct Pet Trent;
    Trent.age = 4;
    Trent.name = "Trent Good";
    return EXIT_SUCCESS;
}
```

C Programming: Section 6.1+
Unions

Two basic ways to declare unions
  ○ Typedef
  ○ Standard

Properties (members) that are basic data types (or pointers to them)

How do we find size (roughly)?

How does size pertain to the members of the union?

What’re the current values of the two wallets?
  ○ What specifiers would we use to print them?

C Programming: Section 6.8+
Bit fields (special structs)

- Able to use : to specify fields of a certain width
- Simply useful for flags to keep track of state, etc.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    char* name;
    unsigned int gps_on : 1;
    unsigned int flashlight_on : 1;
    unsigned int bluetooth_on : 1;
} phone;

int main() {
    phone pixel;
    pixel.name = "Pixel 6A";
    pixel.gps_on = 0;
    pixel.flashlight_on = 0;
    pixel.bluetooth_on = 1;
    return EXIT_SUCCESS;
}
```