Structs

Intro

- Structs, or structures, give a way of implementing complex data structures in c.

- They are collections of different variable types.
    - The variables inside a struct are called "members"
    - Members exist in their own namespace and have their own scope.

- Unlike object oriented languages, c does not allow structures to directly contain their own methods.

- Default way to access: "struct name.member = …"
  for a ptr to a struct you can access members of structure pointed to using "->"

Example

```c
#include<stdio.h>
#define MAXLEN 120 //Macro defining the length of the strings in the structure

struct prisoner_record{ //this declare a tag so that the c recognizes the type when invoked below
  char name_first[MAXLEN];
  char name_last[MAXLEN]; //These are known as members
  int number;
}; //The semicolon here denotes that this is a tag not a declaration of a variable.

int main()
{
  struct prisoner_record PR = {"Jean","Valjean",24601};
  printf("Last Name: %s, First Name: %s, Prisoner Number: %d\n",  PR.name_last, PR.name_first, PR.number);
}
```

```
Last Name: Valjean, First Name: Jean, Prisoner Number: 24601
```

Typedef

- It is often useful to construct type definitions.

- Typedef can do this so you don't have to type struct over and over.

```c
#include<stdio.h>
#define MAXLEN 120 //Macro defining the length of the strings in the structure

typedef struct prisoner_record{ //this declare a tag so that the c recognizes the type when invoked below
  char name_first[MAXLEN];
  char name_last[MAXLEN]; //These are known as members
  int number;
}Prisoner_Record; // declares the type Prisoner_Record as an alias for the structure type

int main()
{
  Prisoner_Record PR = {"Jean","Valjean",24601};
  printf("Last Name: %s, First Name: %s, Prisoner Number: %d\n",  PR.name_last, PR.name_first, PR.number);
}
```

```
Last Name: Valjean, First Name: Jean, Prisoner Number: 24601
```

Memory Layout for structs

Address	Value	Name
0x200	"Jean\0"	name_first
0x319		
0x320	"Valjean\0"	name_last
0x439		
0x440	24601	number
0x441		
0x442		
0x443		

```c
#include<stdio.h>
#define MAXLEN 120 //Macro defining the length of the s

struct prisoner_record{ //this declare a tag so that th
    char name_first[MAXLEN];
    char name_last[MAXLEN]; //These are known as members
    int number;
}; //The semicolon here denotes that this is a tag not

int main()
{
    struct prisoner_record PR = {"Jean","Valjean",24601};
```

Functions and structs

- Variables in c are passed by creating a copy.

- For this reason, c defines that structs are also passed by copy.

- This means that arrays that are member of a structure are copied.

- Pass by copy can be very wasteful and so it is usually recommended to pass by reference via a pointer to the struct.

```
Last Name: Valjean, First Name: Jean, Prisoner Number: 24601
Last Name: Jean, First Name: Valjean inside function
Last Name: Valjean, First Name: Jean outside function
Last Name: Jean, First Name: Valjean inside function (ptr)
Last Name: Jean, First Name: Valjean outside function (ptr)
```

```c
void swap_names(Prisoner_Record PR)
{
  char name_temp[MAXLEN];
  strcpy(name_temp,PR.name_first);
  strcpy(PR.name_first,PR.name_last);
  strcpy(PR.name_last,name_temp);
  printf("Last Name: %s, First Name: %s inside function\n",  PR.name_last, PR.name_first);
}

void swap_names_ptr(Prisoner_Record *PR)
{
  char name_temp[MAXLEN];
  strcpy(name_temp,PR->name_first);
  strcpy(PR->name_first,PR->name_last);
  strcpy(PR->name_last,name_temp);
  printf("Last Name: %s, First Name: %s inside function (ptr)\n",  PR->name_last, PR->name_first);
}

int main()
{
  Prisoner_Record PR = {"Jean","Valjean",24601};
  printf("Last Name: %s, First Name: %s, Prisoner Number: %d\n",  PR.name_last, PR.name_first, PR.number);
  swap_names(PR);
  printf("Last Name: %s, First Name: %s outside function\n",  PR.name_last, PR.name_first);
  swap_names_ptr(&PR);
  printf("Last Name: %s, First Name: %s outside function (ptr) \n",  PR.name_last, PR.name_first);

}
```