# Stuctures in C

# (struct)

# + some I/O with structs

structs are complex datatypes which allow the programmer to group together different datatypes in a single unit

Similar to the “class” datatype in object-oriented programming languages

A struct (a user-defined datatype):

```
struct people {
    char firstname [50];
    char surname [50];
    char address [100];
    int age;
};
```

With this declaration we can declare struct variables:

```
struct people person1, person2;
```

Now we have two person variables...

```
person1, person2
```

...that we can use to store information about people

Assignment to the various parts of the struct can be done in the following manner:

```
strcpy(person1.firstname,”John”);
strcpy(person1.surname,”Smith”);
strcpy(person1.address,”123 Main St.”);
person1.age = 32;
```

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

struct people {
    char firstname [50];
    char surname [50];
    char address [100];
    int age;
};

void print_person(struct people person);
```

```
int main()
{
  struct people person1, person2;

  strcpy(person1.firstname,"John");
  strcpy(person1.surname,"Smith");
  strcpy(person1.address,"123 Main St.");
  person1.age = 32;
  strcpy(person2.firstname,"Dina");
  strcpy(person2.surname,"Hall");
  strcpy(person2.address,"334 High St.");
  person2.age = 29;

  print_person(person1);
  print_person(person2);

} //end main()
```

```
void print_person(struct people person) {

 printf( "First name: %s\n", person.firstname);
 printf( "Surname: %s\n", person.surname);
 printf( "Address: %s\n", person.address);
 printf( "Age: %d\n", person.age);

} //end print_person ()
```

What is written to the screen:

```
$ ./a.out
First name: John
Surname: Smith
Address: 123 Main St.
Age: 32
First name: Dina
Surname: Hall
Address: 334 High St.
Age: 29
$
```

We can write whole structs directly to a binary file with the function fopen() and fwrite():

```
fopen(name_of_file,mode);
```

mode can be “r” for readonly and “w” for writeonly...

```
fwrite(&the_struct,size_in_bytes,nr_of_structs
  ,the_file);
```

```
//same declarations as in the previous example


int main()
{
  struct people person1, person2;
  FILE *fp; //file variable

  strcpy(person1.firstname,"John");
  strcpy(person1.surname,"Smith");
  strcpy(person1.address,"123 Main St.");
  person1.age = 32;
  strcpy(person2.firstname,"Dina");
  strcpy(person2.surname,"Hall");
  strcpy(person2.address,"334 High St.");
  person2.age = 29;
```

```
  fp = fopen("personfile.dat","w");
  if (fp != NULL){
    fwrite(&person1,sizeof(struct people),1,fp);
    fwrite(&person2,sizeof(struct   people),1,fp);
    fclose(fp);
  } else {
    fprintf(stderr,"Problem opening file\n");
    exit(1);
  }

} //end of main()
```

The last bit of code created a binary file called personfile.dat which now can be read with fread()

```
fread(&the_struct,size_in_bytes,nr_of_structs,
  the_file);
```

```
//same declarations as in the previous example


int main()
{
  struct people person1;
  FILE *fp;

  fp = fopen("personfile.dat","r");
  if (fp == NULL) {
    fprintf(stderr,"Problem opening file\n");
    exit(1);
  }

  while (!feof(fp)) {
    fread(&person1,sizeof(struct people),1,fp);
    print_person(person1);
  }
  fclose(fp);

}
```