Multi-Paradigm Programming - Structuring Data in C

Dominic Carr

Atlantic Technological University

dominic.carr@atu.ie

What We Will Cover

1. Goals of this Session
2. Structuring Data in C

Goals of this Session

Goals

- To understand....
  - How to group variables together in C
  - Create a structure of related data
  - How to refer to the memory address of variables

Structuring Data in C

Structuring Data in C I

Listing 1: C Example of representing a Person

```
struct person
{
   int age;
   float weight;
};
```

Structuring Data in C II

- A struct is a way of grouping individual variables together
- It can be used to create a *representation* of something
  - A person, a clock, an animal, an ticket etc.
  - As defined above a person has an age and a weight
  - A ticket could be defined to have an owner, a price, a date, a title, a venue etc. All *bundled* in a ticket struct.
- We will contrast this with OOP approaches which we will learn about soon.

Structuring Data in C III

Listing 2: Struct representing a clock

```
struct clock {
    int hours;
    int mins;
    int secs;
};

void printClock(struct Clock c)
{
    printf("\n Clock");
    printf("\n%02d:%02d:%02d", c.hours, c.mins, c.secs);
}
```

Structuring Data in C IV

Listing 3: Struct representing an address

```
struct address
{
   char name[50];
   char street[100];
   char city[50];
   char county[20];
   char eircode[10];
};

int main()
{
    struct address atu = { "ATU Galway", "Old Dublin Road", "
        Galway", "Co. Galway", "H91 T8NW"};
    printf("The eircode is %s\n", atu.eircode);
    return 0;
}
```

Structuring Data in C V

Listing 4: Modifying values in a struct

```
int main()
{
    struct address atu = { "ATU Galway", "Old Dublin Road", "
        Galway", "Co. Galway", "H91 T8NW"};
    printf("The eircode is %s\n", atu.eircode);
    atu.eircode[5] = '9';
    printf("The eircode is %s\n", atu.eircode);

    return 0;
}
```

Structuring Data in C VI

Listing 5: Filling the Person from User Input

```
struct person
{
   int age;
   float weight;
};
int main()
{
    struct person *personPtr, person1;
    personPtr = &person1;
    printf("Enter age: ");
    scanf("%d", &personPtr->age);
    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);
    printf("Age: %d\n", personPtr->age);
    return 0;
}
```

Structuring Data in C VII

- What's this & and $->$ stuff all about?

& - The Reference Operator

- When a variable is created a memory address is assigned to the variable.
  - This is where the variable is stored on the computer.
- When we assign a value to the variable, it is stored in this memory address.
- To access the address we use the reference operator (&).

```
int myAge = 43;
printf("%p", &myAge);
```

Structuring Data in C VIII

- &myAge is as a pointer.
  - A pointer stores the memory address of another variable as its value.
- A pointer points to a data type (like int) of the same type, and is created with the * operator.

```
int myAge = 43;   // An int variable
int* ptr = &myAge; // A pointer variable, with the name ptr,
    that stores the address of myAge

// Dereference: Output the value of myAge with the pointer
    (43)
printf("%d\n", *ptr);
```

Structuring Data in C IX

The $->$ operator is used to reference internal members of a struct which is referenced by a pointer.
Structure dereference ("member b of object pointed to by a")

```
&personPtr->age
// a is &personPtr and b is age
```

This is important because we need to use a reference when passing structs into **functions for modification.**

The End