COMP1511 PROGRAMMING FUNDAMENTALS

WEEK 2 LECTURE 1

Structs and Enums

LAST LECTURE...

LAST WEEK, WE TALKED:

- Welcome and Introductions
- Started looking at C
- Our first Hello! program
- Compiling and running your code
- **`printf()`** and **`scanf()`**
- Variables (.**`int`**, **`double`**, **`char`**)
- Maths :)
- Basic IF statements

IN THIS LECTURE...

TODAY...

- Recap
- Structs
- Enums

“

WHERE IS THE CODE?

Live lecture code can be found here:

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/22T2/LIVE/WEEK02/

RECAP

RELATIONAL OPERATORS

NOTICE: IN C, WE HAVE == AND =

THESE ARE NOT THE SAME AND DO NOT MEAN WHAT YOU ARE USED TO IN MATHS!

USING = WHEN YOU ASSIGN VALUES
USING == WHEN YOU ARE CHECKING FOR EQUIVALENCE

- Relational Operators work with pairs of numbers:
  - < less than
  - > greater than
  - <= less than or equal to
  - >= greater than or equal to
  - == equals
  - != not equal to

- All of these will result in 0 if false and a 1 if true

SOME EXAMPLES

LET'S TRY THIS OUT...

- True (1) or False (0)?

```
if (7 < 15 && 8 >= 15) {
    //do something
}
```

```
if (7 < 15 || 8 >= 15) {
    //do something
}
```

```
if !(5 < 10 || 6 > 13) {
    //do something
}
```

HOW DOES SCANF() REALLY WORK?

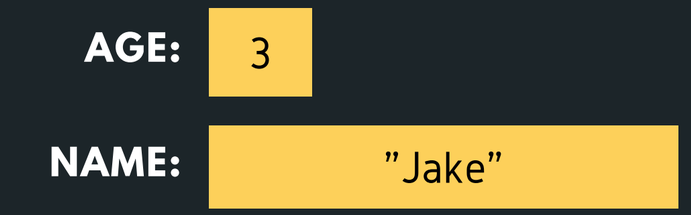
A MAGICAL POWER...

- Gives us the ability to scan stuff in from the terminal (standard input)
- We have to tell the computer what we expect to scanf() - is it an **`int`**, **`double`**, or **`char`** ?
- But since scanf() is a function does it return something?
    - Yes, scanf() returns the number of input values that are scanned
    - If there is some input failure or error then it returns EOF (end-of-file) - we will look at this more tomorrow!
    - This is useful to check for any errors

VARIABLES

- Variables allow us to store data in the memory of a program
- This is short term memory (we ”forget” our stored data once the program or the scope of the variable ends)
- C needs to know the type of the variable, to know how much memory to ask the operating system for.

- int age = 3;
- string name = ”Jake”;



SCANF

- Gives us the ability to scan stuff in from the terminal (standard input)
- We have to tell the computer what the type we want to read scanf() - is it an **`int`** , **`double`**, or **`char`** ?
- But since scanf() is a function does it return something?
    - Yes, scanf() returns the number of input values that are scanned
    - If there is some input failure or error then it returns EOF (end-of-file) - we will look at this more tomorrow!
    - This is useful to check for any errors

CUSTOM DATA TYPES

ENUMS

- ENUMS (enumerations) is a custom data type, which describes set of possible values in a programmer-defined category

- For example, days of the week.

CUSTOM DATA TYPES

ENUMS DEMO

- Let’s demo why structs are useful

BREAK

- Let’s demo why structs are useful

CUSTOM DATA TYPES

STRUCTS

- Structures…. Or **`struct`** (as they are known in C!)
- Structs are variables that are made up of other variables

STRUCTURES

WHAT? WHY? EXAMPLES?

- What happens if you wanted to group some variables together to make a single structure?
- Why do we need structures?
  - Helps us to organise related but different components into one structure
  - Useful in defining real life problems
- What are some examples in real life where some things go together to make a single component?

HOW DO WE CREATE A STRUCT?

To create a struct, there are three steps:

1. Define the struct (outside the main)
2. Declare the struct (inside your main)
3. Initialise the struct (inside your main)

1. DEFINING A STRUCT

WHAT AM I GROUPING TOGETHER INTO ONE WHOLE? LET'S USE AN EXAMPLE OF A COORDINATE POINT

Because structures are a variable that we have created, made up of components that we decided belong together, we need to define what the struct (or structure is). To define a struct, we define it before our main function and use some special syntax.

```
struct struct_name {
      data_type variable_name_member;
      data_type variable_name_member;
      ...
};
```

For example, using the coordinate point example, to declare a variable, cood_point, of type struct coordinate

1. DEFINING A STRUCT

WHAT AM I GROUPING TOGETHER INTO ONE WHOLE? LET'S USE AN EXAMPLE OF A COORDINATE POINT

For example, using the coordinate point example, to make a structure called coordinate, that has two members - the x_coordinate and the y_coordinate:

```
struct coordinate {
      int x_coordinate;
      int y_coordinate;
  };
```

2. DECLARING A STRUCT

INSIDE YOUR MAIN

To declare a struct, inside the main function (or wherever you are using the structure - more on this later)...

```
struct struct_name variable_name;
```

For example, using the coordinate point example, to declare a variable, cood_point, of type struct coordinate

```
struct coordinate cood_point;
```

3.INITIALISE A STRUCT

INSIDE YOUR MAIN

We access a member by using the dot operator .

```
variable_name.variable_name_member;
```

For example, using the coordinate point example, with variable name: cood_point, trying to access the x coordinate:

```
cood_point.x_coordinate;
```

LET'S SEE IT ALL TOGETHER FOR A COORDINATE POINT

1. DEFINE
2. DECLARE
3. INITIALISE

1. DEFINE

Inside the main function

2. DECLARE

Inside the main function

3. INITIALISE

Inside the main function

```
// Define a structure for a coordinate point

struct coordinate {
      int x_coordinate;
      int y_coordinate;
 };

// Declare structure with variable name

struct coordinate cood_point;


// Access stuct member to assign value

cood_point.x_coordinate = 3;
cood_point.y_coordinate = 5;
```

LET'S SEE STRUCTS IN ACTION

CODE DEMO

You can see structs in action:

`struct_intro.c`

REACH OUT

CONTENT RELATED QUESTIONS

Check out the forum

ADMIN QUESTIONS

cs1511@cse.unsw.edu.au