# STRUCTURES

BEESHANGA ABEWARDANA JAYAWICKRAMA

UTS:
ENGINEERING AND
INFORMATION
TECHNOLOGY



feit.uts.edu.au

# A BETTER WAY TO STORE STRUCTURED DATA

A program is expected store information about planets.

Option 1: Have separate variables of C primitive data types to store different information.

```
double diameter; int nmoons; ...
```

Option 2: Define a new structured data type that contains all information about a planet.

```
planet_t earth;
```

`planet_t` is a custom defined structure type.

# STRUCTURE DEFINITION

```
struct planet
{
    double diameter;
    int nmoons;
    ...
};

typedef struct planet planet_t;
```

Now `planet_t` is a variable type. You can use it similar to any other primitive data type in C.

i.e. make variables of the new structure type, pass/return structures to/from functions, make arrays of structures, ...

# VARIABLES OF STRUCT TYPE

Declare a variable of `planet_t` type:

```
planet_t earth;
```

Data type — Variable name

Access the `diameter` of `earth`:

```
earth.diameter = 12000.0;

printf("%d\n", earth.diameter);
```

When passing a structure to a function a local copy is made.

# STRUCT IN A STRUCT

It is possible to have nested structures as shown below. The order you define the structures matter here.

```
struct moon
{
    double diameter;
    char name[20];
};
typedef struct moon moon_t;

struct planet
{
    double diameter;
    char name[20];
    moon_t moons[10];
};
typedef struct planet planet_t;
```

UTS
UNIVERSITY OF TECHNOLOGY SYDNEY

# FILE PROCESSING

BEESHANGA ABEWARDANA JAYAWICKRAMA

UTS:
ENGINEERING AND
INFORMATION
TECHNOLOGY

# FILE TYPES

Text files:

> Generally contains human readable characters

> End of each line is marked by LF (Linux) or CR LF (Windows)

> Read/write functions from stdio.h - `fscanf`, `fprintf`, `fgetc`, `fputc`

Binary files:

> Generally not limited to human readable characters

> Read/write functions from stdio.h - `fread`, `fwrite`, `fgetc`, `fputc`

**TODO**: Read about above functions on http://www.cplusplus.com

Have you already been writing/reading text files? Devices as files:

> `stdin`, `stdout` and `stderr`

# OPENING AND CLOSING FILES

Opening files:

```
FILE* fopen(const char* filename, const char* mode);
```

Common modes: "r" (read text file), "w" (write text file), "rb" (read binary file), "wb" (write binary file).

Closing files:

```
int fclose (FILE* stream);
```

Close any file stream operating in any mode.

**TODO**: Read about above functions on http://www.cplusplus.com

# EXAMPLE: READING CHARS UNTIL THE END OF A TEXT FILE

Using fgetc read one char at a time. When the end of file is reached fgetc returns EOF.

```
/* input file stream finp is open */
char ch;
for (ch=fgetc(finp); ch!=EOF; ch=fgetc(finp))
{
    /* do something with ch */
}
```