CS2100 - L06 - Structures

Week 3

- Defining a New Structure
- Declaring a Structure Variable
- Usage of Structure Variables
- Passing Structures into Functions
- Combining Structures with Other Data Types

```c
struct Fraction {
    int num;
    int den;
};

int main()
{
    struct Fraction frac1 = { 0 };
    int common;

    printf( "Numerator and denominator: " );
    scanf( "%d%d", &(frac1.num), &(frac1.den) );

    common = GCD( frac1.num, frac1.den );
    frac1.num = frac1.num / common;
    frac1.den = frac1.den / common;
```

fields

data type

variable

need to initialize at least one field, rest will be initialized to 0 by default

accessor

Similar to:

- **class** in Python & JavaScript

The **GCD()** function implementation is not shown in this code

Defining a New Structure

SYNTAX

Definition:

```c
struct struct_name {
    datatype fieldname1;      //one or more fields
    [ datatype fieldname2; ]
};
```

EXAMPLE

```c
struct Fraction {
    int num;          //numerator
    int den;          //denominator
};
```

Declaring a Structure Variable

SYNTAX

Declaration:

```
struct struct_name identifier;
```

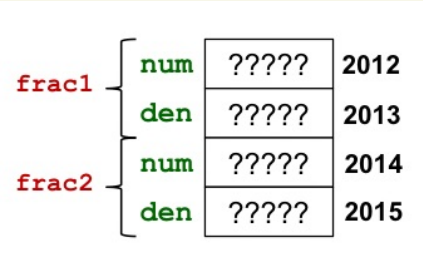
OR

```
struct struct_name identifier = init_values;
```

EXAMPLE

```
struct Fraction myFraction;
```

- **Behavior:**
  - Structure variable contains multiple fields as defined in the structure
  - Each structure variable has an **independent set of the fields**
  - The fields of a structure are placed in adjacent locations in memory



Usage of Structure Variables

accessor

SYNTAX	
	`structure_var.fieldname`

Passing Structures into Functions

- Structure variable **is passed by value**
  - A **copy of the actual argument will be made**

- Structure variable **is commonly passed by address instead:**
  - To avoid memory and time wastage
  - To allow a function to modify the actual argument

```
void printFraction( struct Fraction *fptr )
{
    printf( "%d / %d", (*fptr).num,
                       (*fptr).den );
}
```

myfraction

```
int main()
{
   struct Fraction myFraction = { 123, 456 };

   printFraction( &myFraction );

   return 0;
}
```

- The "->" is known as **indirect field selector**

```
void printFraction( struct Fraction *fptr )
{
    printf( "%d / %d", (*fptr).num,
                       (*fptr).den );
}
```

≡

```
void printFraction( struct Fraction *fptr )
{
    printf( "%d / %d", fptr->num, fptr->den );
}
```

Combining Structures with Other Data Types

- **Structure** and **array** can be "combined" to meet more complicated needs
- For example:
  - **Array** of **structures**:
    - Array of fractions, array of students etc
  - **Structure** with **array** as field:
    - Student's name is a string (char array)
  - **Structure** with **structure** as field:
    - A **line** in a 2D plane can be defined with **two points** (X_1, Y_1) and (X_2, Y_2)