Structures

What is a Structure Variable

- Traditional terms like *fields* in other programming languages are the same as *structures* and *members* in C++
- Think of ***structures*** as ***containers*** for various pieces of information about an object
- You will define an object and then define the related information about that object – the members

Defining a Structure

- General Syntax for declaring a structure – defines how it is organized and what members it will have (same as how classes are defined)

```
struct type_name
    {
        int member_number     // these are standard C++
       string member_name    // variable declarations
    };  // note the closing semi-colon
```

Structure example - student

- Member list of student

```
string name;
     int id;
     int mark[3];
```

- Putting member list inside structure:

```
struct Student
    {
            string name;
            int id;
        int mark[3];
    };
```

- This does not actually define the variable or set aside any space or memory – it is just a specification of what the variables will look like when they are defined.

Declaring a Structure *Instance*

- You need to declare an INSTANCE of the Structure to actually create it

```
Structure_Name      Nameof_Instance;
Student             stu;
```

(do not need key word struct again here)

- You define structure variables just as you would any other basic built-in data type like `int`

```
Datatype    variable_name ;
string  name;
int         id;
int         mark[3];
```

Referencing structure members

- You will use a “ dot operator “ to reference structure members
- You will write the member name in three parts

```
stu.name
```

  - Structure variable (stu)
  - Dot operator (.)
  - Member name (name)

Structure Members

- You treat structure members just like any other variable

```
stu.name = "Porthos";
```

- assigns the value Porthos to name

```
cout  << "Name = " << stu.name;
```

- Output: Name = Porthos

Nested structures

- You can nest structures within other structures.
- You would reference them also with the dot operator;
- Three nested structures would result in a name such as:

  - `apartment1.laundry_room.washing_machine.feet`
  - `(struct 1) (struct 2)  (struct 3)  (member)`

Structure Summary

- Structures are important because you can group several data items together to form a single entity
- Structure declaration lists the variables
- Structure definition then sets aside memory for those variables
- Can refer to members such as id = 1234 or as `cout <<stu.id>>`

A word about enumeration

- You can write very good C++ programs without ever using enumeration
- Lists all possible values for a data type (usually very short list)

```
enum days_of_week (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
```

- Very specific, normally short list of ALL possible values
- Unlike other data types like `int` – which could be any infinite number of values