Real life is not just a collection of small bits of information. Bits of information are grouped together in an organized way. Structures allow us to combine several variables into a single unit with multiple properties.

### Data for a Student

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student ID</td>
<td>Integer</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
</tr>
<tr>
<td>GPA</td>
<td>Decimal number</td>
</tr>
</tbody>
</table>

Instead of defining three variables each time a student is stored, let’s create a structure that contains all necessary information for a student.
Structure Syntax

- First, we need the `typedef` and `struct` keywords.
- The `typedef` keyword allows types to be given shorter names.
- The `struct` keyword means that the description of a structure will follow, in curly braces (`{` and `}`).

```c
typedef struct {
    ...
} student;
```

- After this definition, any use of `student` will register as a type, just as `int` and `char` do already.
The biggest part of a structure are its members. We add them by defining variables within the curly braces.

```c
typedef struct
{
    int id;
    char name[100];
    double gpa;
} student;
```
Using a Structure

```c
student s;     /* Define s with type student */
s.id = 555111022;
strcpy(s.name,"Franky Frankerson");
s.gpa = 3.4;
```
Structures Pay Off in a Big Way

- Structures seem pointless when dealing with one at a time.
- It becomes a much more important experience when many are involved.
- Like any other type, we can define an array of structured types:

  ```
  student people[100];
  ```

- As usual, setting an index on the array results in the type `student`, and we can access the members in the same way:

  ```
  if ( x == people[i].id )
  ```
Doing it the Old Way

- Keeping track of these three values before required three different arrays:

  ```
  int ids[100];
  double gpas[100];
  char names[100][100];
  ```

- The only thing holding them together is the array index.

- What would happen if I wanted to sort by id?
Doing it the new Way

- Now, a single array stores all information for a student in a single package.

```c
printf("%s has GPA %lf\n",
       people[i].name,
       people[i].gpa);
```
Cheat Sheet: Structures

Structure Definition:

typedef struct {
    type1 memberName1;
    type2 memberName2;
    ...
} structName;

Structure Declaration:

structName varname;

Structure Use:

varname.memberName1;
varname.memberName2;