Structures

Topics

- What is a structure
- How to define a structure
- How to populate and use structures

What is a structure

- An array can only contains elements of the same type
- What if you need to store data about a student including the name, and GPA:
  - Name is a string
  - GPA is float
- Instead of using two separate arrays, a structure is a container for related data

How it works

- A structure allows you to store information related to one entity
  - An entity can be a student, a book, a bank customer, and so on
- Each piece of information (member) in the structure can be of any other data type.
- For example, a member could be: name, GPA, Date of birth, and Email.
- You can then search for a student by their name, or DOB, or both.

A structure

- A structure is a data type
  - You can create an array of int, and
  - Similarly, you can create an array of a structure
  - Each element of the array will be made of the members of the structure

Declaring A Structure

```
struct struct_name {
   DataType member1_name;
   DataType member2_name;
   DataType member3_name;
   …  //you can add more members
};
```

Declaring A Structure

```
struct struct_name {
   DataType member1_name;
   DataType member2_name;
   DataType member3_name;
   …  //you can add more members
};
```

- struct_name is a name you pick.
- Members data type can be any type.
- After this declaration struct struct_name would act as a data type.

Examples I

Declare a structure whose tag name is Point2D and that contains exactly two fields (or members), both of type double. The first field is x and the second field is y.

```
struct Point2D {
        double x;
        double y;
};
```

←again, this is a definition of a data type

←Next you need a variable of that data type

Examples II

Struct Point2D firstpoint;

- In the above the type Point2D has been already defined, and firstpoint is a variable of type Point2D
- Next you need to assign each member of firstpoint a value.
- How?

Examples III

Two ways to assign values:

1. Recall a point has two members of type double:
   firstpoint = {12.45, 17.283};
2. Use the dot operator, and recall the first member name is x, and the second is y:
   firstpoint.x = 12.45;
   firstpoint.y=17.283;

Another Example

```
struct studentdata {
    char *name; //this member is a string
    int id;          // int
    float gpa;     //float
};
```

Declaring a variable of type struct

struct studentdata student;

Declare Struct and Var in One

```
struct studentdata {
    char *name; //this member is a string
    int id;          // int
    float gpa;     //float
} student;
```

Saving data into a struct variable

```
 struct studentdata  student;   //declare
//Two ways to fill with actual data:
 student = { "John", 1234, 3.15};
// OR use the dot
student.name = "John";
student.id = 1234;
student.gpa = 3.15;
```

Using the DOT operator

- Makes code clearer
- Same as using struct variable

Accessing the members

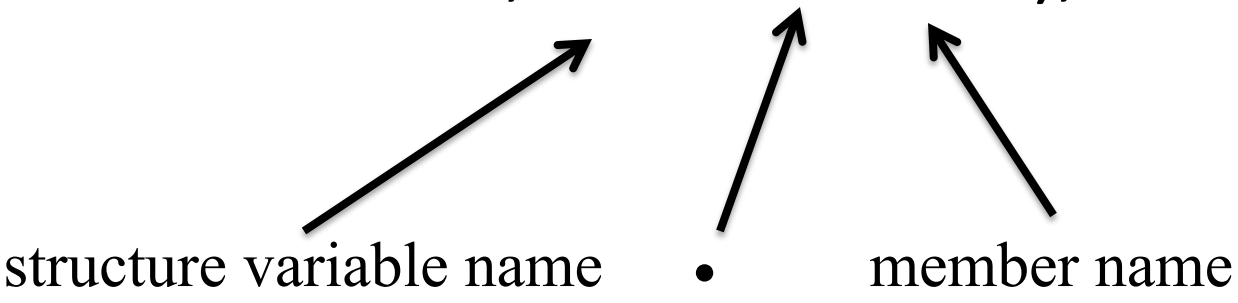
Use the dot operator for each member to access

```
printf("Student Name is: %s", student.name);
```

Accessing the members

Use the dot operator for each member to access

```
printf("Student Name is: %s", student.name);
```



Fun With Structures - FYI

- A structure may contain members that are int, *char, or struct
- Syntax Example:

```
Struct StudentData
{
   int id;
   int age;
   char* name;
   struct stu_address stuAddress; //this member is a struct
};
```

No Can Do

- Structures may not be compared.
- Hence for:

  struct studentdata s1={123,18,“Jack”};
  struct studentdata s1={whatever values};

  Can not do:

  ```
  if s1==s2
  ```

  But you can compare the members of s1 and s2

Structures are meant for Pointers

- Traditionally data related to multiple items of a structure type are managed through pointers or dynamic arrays
- This requires memory allocation techniques which are beyond the scope of our class
- In a C++ or Data Structures this topic is covered in details

What Can We Do

To store student data for 20 students:

Create a structure declaration

Create an array of the structure type

Arrays of Structure

To store student data for many students:

- Declare a structure

```
struct StudentData {
    char* name; //this member is a string
    int id;          // int
    float gpa;     //float
};
```

- Create an array of type structure

```
struct StudentData record[19]; //record is an array
```

Now What?

- Fill the array with data:

```
for(i=0;  i<20; i++)
        { // skipping the print prompts
              // the question is, can we do
record[i].id=????
    //
    // answer: No!
 ….Now what?
```

Why Not?

- When declaring an array, the array name is a pointer to the first element of the array.
- First element is a structure
- Use a variable of the structure type, assign it to the array element, then use the dot operator
- See next demo

Filling the Array of Struct

```
/*Fill the array with data, we can not use
record[i].membername = …
Instead use:   */
struct StudentData {
   char name[10]; //this member is a string
   int id;       // int
   float gpa;     //float
};

int main()
{
        struct StudentData records[2];
        struct StudentData record;

int i;
for (i=0; i<2;  i++) {
       printf("Records Database : %d \n", i+1);
       printf("\n Enter student name:");
       gets(record.name); //string member

       printf("\n Enter student id:");
       scanf_s("%d",&(record.id));
       printf(" \n Enter student GPA:");
       scanf_s("%f", &(record.gpa));
       records[i] = record;
}

  printf("Now all records are full \n");
  printf("Enter which record you'd like to review \n");
  scanf_s("%d",&i);

record = records[i-1];
printf("Student name: %s \n ", record.name);
printf("Student ID: %d \n ", record.id);
printf("Student GPA: %f \n ", record.gpa);

printf("\n press return when done");

return 0;
}
```

Summary

- Structures are one category of complex types
- There are other categories but we'll skip them
- This was meant as an intro
- A data structures class provides more in-depth coverage