C++ Structures

- C/C++ arrays allow you to define variables that combine several data items of the **same** kind, but structure is another user defined data type which allows you to combine data items of **different** kinds.
- Structures are used to represent a record, suppose you want to keep track of your books in a library.
- You might want to track the following attributes about each book:
  - **Title**
  - **Author**
  - **Subject**
  - **Book ID**

Defining a Structure

- To define a structure, you must use the **struct** statement.
- The **struct** statement defines a new data type, with more than one member, for your program.
- The format of the struct statement is this:

```
struct [structure tag] {
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

- The `structure tag` is optional and each member definition is a normal variable definition, such as *int i*; or *float f*; or any other valid variable definition.
- At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional.
- Here is the way you would declare the **Library** structure:

```
struct Library {
string  title;
string  author;
string  subject;
int   book_id;
} book;
```

Accessing Structure Members

- To access any member of a structure, we use the member access operator (.)
- The member access operator is coded as a period between the structure variable name and the structure member that we wish to access.
- You would use **struct** keyword to define variables of structure type.
- Following is the example to explain usage of structure:

```
#include <iostream>

using namespace std;

struct Books {
```

```
string title;
string author;
string subject;
int   book_id;
};

int main() {
 struct Books Book1;        // Declare Book1 of type Book

   // book 1 specification
Book1.title =  "Learn C++ Programming";
Book1.author = "Chand Miyan";
Book1.subject = "C++ Programming";
Book1.book_id = 6495407;

struct Books Book2;        // Declare Book2 of type Book

   // book 2 specification
Book2.title =  "Telecom Billing";
Book2.author = "Yakit Singha";
Book2.subject = "Telecom";
   Book2.book_id = 6495700;
   // Print Book1 info
cout<< "Book 1 title : " << Book1.title <<endl;
cout<< "Book 1 author : " << Book1.author <<endl;
cout<< "Book 1 subject : " << Book1.subject <<endl;
cout<< "Book 1 id : " << Book1.book_id <<endl;
   // Print Book2 info
cout<< "Book 2 title : " << Book2.title <<endl;
cout<< "Book 2 author : " << Book2.author <<endl;
cout<< "Book 2 subject : " << Book2.subject <<endl;
cout<< "Book 2 id : " << Book2.book_id <<endl;
   return 0;
}
```

- When the above code is compiled and executed, it produces the following result:

```
Book 1 title : Learn C++ Programming
Book 1 author : Chand Miyan
Book 1 subject : C++ Programming
Book 1 id : 6495407
Book 2 title : Telecom Billing
Book 2 author :Yakit Singha
Book 2 subject : Telecom
Book 2 id : 6495700
```

Structures as Function Arguments

- You can pass a structure as a function argument in very similar way as you pass any other variable or pointer.
- You would access structure variables in the similar way as you have accessed in the above example:

```
#include <iostream>
#include <cstring>
using namespace std;

struct Books {
string  title;
string  author;
string  subject;
int   book_id;
};

void printBook( struct Books book ) {
cout<< "Book title : " <<book.title<<endl;
cout<< "Book author : " <<book.author<<endl;
cout<< "Book subject : " <<book.subject<<endl;
cout<< "Book id : " <<book.book_id<<endl;
}

int main() {
 struct Books Book1;        // Declare Book1 of type Book
 struct Books Book2;        // Declare Book2 of type Book
// book 1 specification
Book1.title =  "Learn C++ Programming";
Book1.author = "Chand Miyan";
Book1.subject = "C++ Programming";
Book1.book_id = 6495407;

   // book 2 specification
Book2.title =  "Telecom Billing";
Book2.author = "Yakit Singha";
Book2.subject = "Telecom";
   Book2.book_id = 6495700;

   // Print Book1 info
printBook( Book1 );

   // Print Book2 info
printBook( Book2 );

   return 0;
}
```

- When the above code is compiled and executed, it produces the following result:

```
Book title : Learn C++ Programming
Book author : Chand Miyan
Book subject : C++ Programming
Book id : 6495407
Book title : Telecom Billing
Book author :Yakit Singha
Book subject : Telecom
```

```
    Book id : 6495700
```

Pointers to Structures

- You can define pointers to structures in very similar way as you define pointer to any other variable as follows:

```
    struct Books *struct_pointer;
```

- Now, you can store the address of a structure variable in the above defined pointer variable.
- To find the address of a structure variable, place the & operator before the structure's name as follows:

```
struct_pointer = &Book1;
```

- To access the members of a structure using a pointer to that structure, you must use the -> operator as follows:

```
struct_pointer->title;
```

- Let us re-write above example using structure pointer, hope this will be easy for you to understand the concept:

```
#include <iostream>
#include <cstring>

using namespace std;

    struct Books {
    string  title;
    string  author;
    string  subject;
```

```
     int   book_id;  };


// This function accept pointer to structure as parameter.
void printBook( struct Books *book ) {
cout<< "Book title : " << book->title <<endl;
cout<< "Book author : " << book->author <<endl;
cout<< "Book subject : " << book->subject <<endl;
cout<< "Book id : " << book->book_id<<endl;
}


int main() {
   struct Books Book1;        // Declare Book1 of type Book
   struct Books Book2;        // Declare Book2 of type Book

      // book 1 specification
     Book1.title =  "Learn C++ Programming";
     Book1.author = "Chand Miyan";
     Book1.subject = "C++ Programming";
     Book1.book_id = 6495407;

        // book 2 specification
     Book2.title =  "Telecom Billing";
     Book2.author = "Yakit Singha";
     Book2.subject = "Telecom";
        Book2.book_id = 6495700;

   // Print Book1 info, passing address of structure
printBook( &Book1 );

   // Print Book1 info, passing address of structure
printBook( &Book2 );

   return 0;
}
```

- When the above code is compiled and executed, it produces the following result:

```
Book title : Learn C++ Programming
Book author : Chand Miyan
Book subject : C++ Programming
Book id : 6495407
Book title : Telecom Billing
Book author :Yakit Singha
Book subject : Telecom
Book id : 6495700
```

Complex number example using struct

- A complex number consists of two parts: **real** and **imaginary**
- Both parts are floating point numbers
- This example involves using function that receives and return structures

```
#include<iostream>
using namespace std;
struct Complex {
     float real;
     float img;
 };

 Complex add_complex( Complex z1, Complex z2 ) {
   Complex z3;
   z3.real = z1.real + z2.real;
   z3.img  = z1.img  + z2.img;
   return z3;
 }

 int main()
 {
   Complex z1, z2, z3;
cout<<"Enter the first complex number \n";
cout<<"Enter the real part " ;
cin>>z1.real ;
cout<<"Enter the imaginary part " ;
cin>>z1.img ;

cout<< "Enter the second complex number \n";
cout<< "Enter the real part " ;
cin>>z2.real;
cout<<"Enter the imaginary part " ;
cin>>z2.img ;

cout<<endl;
   z3 = add_complex( z1, z2 );
   //show the sum
cout<< z3.real<< "\t"<<z3.img ;

   return 0;
 }
```

Array of structures

- Arrays can be declared of type **struct**
- In this case, each element of the array will be an independent structure (record)
- Each array element can be accessed using its index
- Each structure field can be accessed using the dot . operator
- Consider the following example:

```
#include <iostream>
using namespace std;
struct student
{
  int roll_no;
  string name;
  int phone_number;
}stud[5];

int main(){

  int i;

for(i=0; i<5; i++){                    //taking values from user
cout<< "Student " <<i + 1 <<endl;
cout<< "Enter roll no" <<endl;
cin>> stud[i].roll_no;
cout<< "Enter name" <<endl;
cin>> stud[i].name;
cout<< "Enter phone number" <<endl;
cin>> stud[i].phone_number;
}

for(i=0; i<5; i++){              //printing values
cout<< "Student " <<i + 1 <<endl;
cout<< "Roll no : " << stud[i].roll_no<<endl;
cout<< "Name : " << stud[i].name <<endl;
cout<< "Phone no : " << stud[i].phone_number<<endl;
}
return 0;
}
```