Data structure in C / C ++

Struct in C / C ++ Arrays in C / C ++ allow you to define several types of variables that can hold the values ??of several members of the same data type. But the structure is another type of data in the C / C ++ programming language, which allows you to combine other types of data.

Arrays in C / C ++ allow you to define several types of variables that can hold the values ??of several members of the same data type. But the structure is another type of data in the C / C ++ programming language, which allows you to combine other types of data.

The structure is used to represent a record. Suppose you want to store the value of a book in your library. You can store the properties of the following book:

Title

Author

Theme

Book ID

Define a structure in C ++

To define structure, you must use **struct** statement. The struct statement defines a new data type, with more than one member in your program. The general form of the struct statement is as follows:

```
struct [ ten cau truc ] {  phan dinh nghia thanh vien ;  phan dinh nghia thanh v
```

Here, the option can be arbitrary and one member defines the variables as int i, float j or another variable definition . At the end of the structure definition, before the semicolon, You can define one or more structural variables (optional). Here's how to declare the Book structure variable:

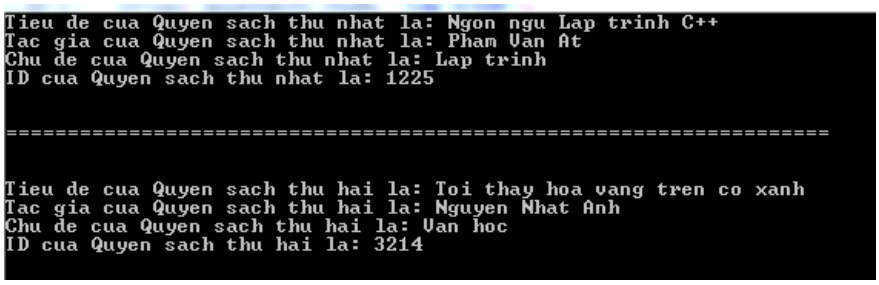
```
struct Books { char  tieude [ 50 ]; char  tacgia [ 50 ]; char  chude [ 100 ]; in
```

Access members of the structure in C ++

To access any member of the structure, use the **element access operator (.)** . The encrypted member access operator is the dot between the structure variable name and the structure member you want to access. You will use **struct** keyword to define variables of structure type. Here is an example of how to use the structure in C ++:

```
#include #include using namespace  std ; struct Books { char  tieude [ 50 ]; cha
```

Compiling and running the above C ++ program will produce the following results:

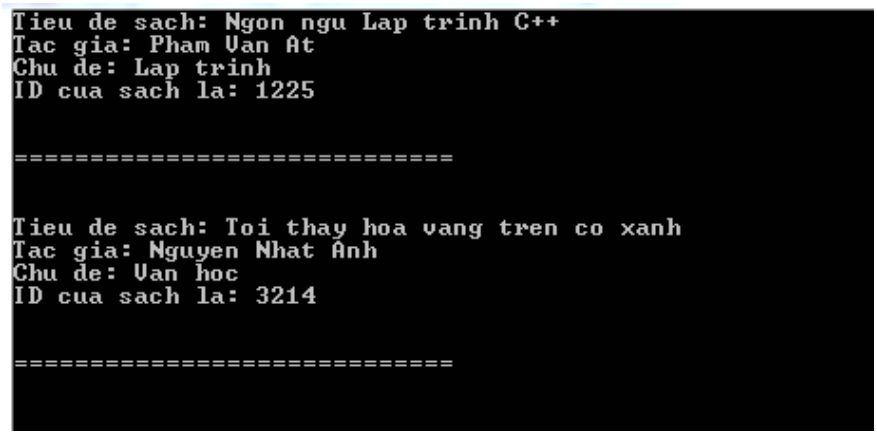


Structure as a function parameter in C ++

You can pass a structure as a function parameter in the same way as when you pass any other variable or pointer. You will access the structure variable in the same way you have access in the above example:

```
#include #include using namespace  std ; void  inthongtin ( struct Books  book
```

Compiling and running the above C ++ program will produce the following results:



Cursor to structure in C ++

You can define pointer structure in the same way you define the pointer to any other variable as follows:

```
struct Books * contro_struct ;
```

You can now store the address of the structure variable in the pointer variable defined above. To find the address of a structure variable, set the operator & before the structure name as follows:

```
contro_struct  = & QuyenSach1 ;
```

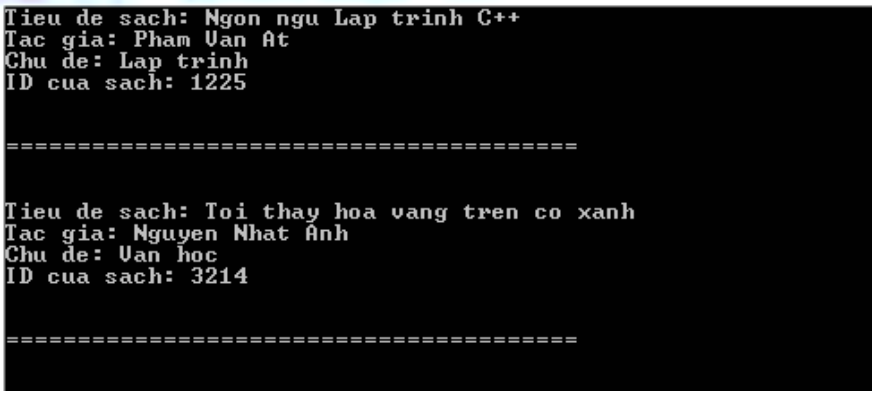
To access a member of a structure using pointers to that structure, you must use the -> operator as follows:

```
contro_struct -> tieude ;
```

Now we rewrite the example above using structured pointers, hoping this will be easy for you to understand this concept:

```
#include #include using namespace  std ; void  inthongtin ( struct Books * book
```

Compiling and running the above C ++ program will produce the following results:



Typedef keyword in C ++

There is an easier way to define structures or you can "alias" the types you create. For example:

```
typedef struct { char  tieude [ 50 ]; char  tacgia [ 50 ]; char  chude [ 100 ];
```

Now, you can use **Books** directly to define the variables of the **Books** structure type without using the **struct** keyword. Here is an example:

```
Books QuyenSach1 , QuyenSach2 ;
```

You can use the **typedef** keyword in C ++ for non-structured forms, as follows:

```
typedef long int * pint32 ;  pint32 x ,  y ,  z ;
```

For x, y and z are all pointers to long int.

According to Tutorialspoint

Previous article: Input / Output in C ++

Next lesson: Class (class) and Object in C ++