← C++ 基本的输入输出　　C++ 类 & 对象 →

C++ 数据结构

C/C++ 数组允许定义可存储相同类型数据项的变量，但是**结构**是 C++ 中另一种用户自定义的可用的数据类型，它允许您存储不同类型的数据项。

结构用于表示一条记录，假设您想要跟踪图书馆中书本的动态，您可能需要跟踪每本书的下列属性：

Title： 标题

Author： 作者

Subject： 类目

Book ID： 书的 ID

定义结构

为了定义结构，您必须使用 **struct** 语句。struct 语句定义了一个包含多个成员的新的数据类型，struct 语句的格式如下：

```
struct type_name {
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
.
.
} object_names;
```

type_name 是结构体类型的名称，**member_type1 member_name1** 是标准的变量定义，比如 **int i;** 或者 **float f;** 或者其他有效的变量定义。在结构定义的末尾，最后一个分号之前，您可以指定一个或多个结构变量，这是可选的。下面是声明一个结构体类型 **Books**，变量为 **book**：

```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

访问结构成员

为了访问结构的成员，我们使用**成员访问运算符（.）**。成员访问运算符是结构变量名称和我们要访问的结构成员之间的一个句号。

分类导航

HTML / CSS
JavaScript
服务端
数据库
移动端
XML 教程
ASP.NET
Web Service
开发工具
网站建设

下面的实例演示了结构的用法：

实例

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

using namespace std;

// 声明一个结构体类型 Books
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( )
{
   Books Book1;        // 定义结构体类型 Books 的变量 Book1
   Books Book2;        // 定义结构体类型 Books 的变量 Book2

   // Book1 详述
   strcpy( Book1.title, "C++ 教程");
   strcpy( Book1.author, "Runoob");
   strcpy( Book1.subject, "编程语言");
   Book1.book_id = 12345;

   // Book2 详述
   strcpy( Book2.title, "CSS 教程");
   strcpy( Book2.author, "Runoob");
   strcpy( Book2.subject, "前端技术");
   Book2.book_id = 12346;

   // 输出 Book1 信息
   cout << "第一本书标题 : " << Book1.title <<endl;
   cout << "第一本书作者 : " << Book1.author <<endl;
   cout << "第一本书类目 : " << Book1.subject <<endl;
   cout << "第一本书 ID : " << Book1.book_id <<endl;

   // 输出 Book2 信息
   cout << "第二本书标题 : " << Book2.title <<endl;
   cout << "第二本书作者 : " << Book2.author <<endl;
   cout << "第二本书类目 : " << Book2.subject <<endl;
   cout << "第二本书 ID : " << Book2.book_id <<endl;

   return 0;
}
```

实例中定义了结构体类似 Books 及其两个变量 Book1 和 Book2。当上面的代码被编译和执行时，它会产生下列结果：

```
第一本书标题 : C++ 教程
第一本书作者 : Runoob
第一本书类目 : 编程语言
第一本书 ID : 12345
第二本书标题 : CSS 教程
第二本书作者 : Runoob
```

```
第二本书类目 : 前端技术
第二本书 ID : 12346
```

结构作为函数参数

您可以把结构作为函数参数，传参方式与其他类型的变量或指针类似。您可以使用上面实例中的方式来访问结构变量：

实例

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

using namespace std;
void printBook( struct Books book );

// 声明一个结构体类型 Books
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( )
{
   Books Book1;        // 定义结构体类型 Books 的变量 Book1
   Books Book2;        // 定义结构体类型 Books 的变量 Book2

    // Book1 详述
   strcpy( Book1.title, "C++ 教程");
   strcpy( Book1.author, "Runoob");
   strcpy( Book1.subject, "编程语言");
   Book1.book_id = 12345;

   // Book2 详述
   strcpy( Book2.title, "CSS 教程");
   strcpy( Book2.author, "Runoob");
   strcpy( Book2.subject, "前端技术");
   Book2.book_id = 12346;

   // 输出 Book1 信息
   printBook( Book1 );

   // 输出 Book2 信息
   printBook( Book2 );

   return 0;
}
void printBook( struct Books book )
{
   cout << "书标题 : " << book.title <<endl;
   cout << "书作者 : " << book.author <<endl;
   cout << "书类目 : " << book.subject <<endl;
   cout << "书 ID : " << book.book_id <<endl;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
书标题 : C++ 教程
书作者 : Runoob
书类目 : 编程语言
书 ID : 12345
书标题 : CSS 教程
书作者 : Runoob
书类目 : 前端技术
书 ID : 12346
```

指向结构的指针

您可以定义指向结构的指针，方式与定义指向其他类型变量的指针相似，如下所示：

```
struct Books *struct_pointer;
```

现在，您可以在上述定义的指针变量中存储结构变量的地址。为了查找结构变量的地址，请把 & 运算符放在结构名称的前面，如下所示：

```
struct_pointer = &Book1;
```

为了使用指向该结构的指针访问结构的成员，您必须使用 -> 运算符，如下所示：

```
struct_pointer->title;
```

让我们使用结构指针来重写上面的实例，这将有助于您理解结构指针的概念：

实例

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

using namespace std;
void printBook( struct Books *book );

struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( )
{
   Books Book1;        // 定义结构体类型 Books 的变量 Book1
   Books Book2;        // 定义结构体类型 Books 的变量 Book2

    // Book1 详述
   strcpy( Book1.title, "C++ 教程");
   strcpy( Book1.author, "Runoob");
   strcpy( Book1.subject, "编程语言");
   Book1.book_id = 12345;

   // Book2 详述
   strcpy( Book2.title, "CSS 教程");
```

```
   strcpy( Book2.author, "Runoob");
   strcpy( Book2.subject, "前端技术");
   Book2.book_id = 12346;

   // 通过传 Book1 的地址来输出 Book1 信息
   printBook( &Book1 );

   // 通过传 Book2 的地址来输出 Book2 信息
   printBook( &Book2 );

   return 0;
}
// 该函数以结构指针作为参数
void printBook( struct Books *book )
{
   cout << "书标题  : " << book->title <<endl;
   cout << "书作者 : " << book->author <<endl;
   cout << "书类目 : " << book->subject <<endl;
   cout << "书 ID : " << book->book_id <<endl;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
书标题  : C++ 教程
书作者 : Runoob
书类目 : 编程语言
书 ID : 12345
书标题  : CSS 教程
书作者 : Runoob
书类目 : 前端技术
书 ID : 12346
```

typedef 关键字

下面是一种更简单的定义结构的方式，您可以为创建的类型取一个"别名"。例如：

```
typedef struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
}Books;
```

现在，您可以直接使用 *Books* 来定义 *Books* 类型的变量，而不需要使用 struct 关键字。下面是实例：

```
Books Book1, Book2;
```

您可以使用 **typedef** 关键字来定义非结构类型，如下所示：

```
typedef long int *pint32;

pint32 x, y, z;
```

x, y 和 z 都是指向长整型 long int 的指针。

← C++ 基本的输入输出

C++ 类 & 对象 →

7 篇笔记 **写笔记**

反馈/建议