
课程安排，请关注微信公众平台或者官方微博

编程语言： **Golang** 与 **html5**

编程工具： **Goland** 和 **HBuilder**

预计平均一周左右更新一或二节课程

授人以鱼，不如授人以渔。

大家好，我是彬哥，

欢迎大家来到 **Golang** 语言社区云课堂课程的学习。Slice

社区论坛网址： www.golang.ltd

技术交流群 ： **221273219**

微信公众号 ： **Golang 语言社区**

微信服务号 ： **Golang 技术社区**

第一季 Go 语言基础、进阶、提高课程

第十三节 Go 语言 结构体

Go 语言中数组可以存储同一类型的数据，但在结构体中我们可以为不同项定义不同的数据类型。

结构体是由一系列具有相同类型或不同类型的数据构成的数据集合。

结构体表示一项记录，比如保存图书馆的书籍记录，每本书有以下属性：

- Title : 标题
- Author : 作者
- Subject: 学科
- ID: 书籍 ID

定义结构体

结构体定义需要使用 **type** 和 **struct** 语句。**struct** 语句定义一个新的数据类型，结构体中有有一个或多个成员。**type**

语句设定了结构体的名称。结构体的格式如下：

```
type struct_variable_type struct {  
    member definition;  
    member definition;  
    ...  
    member definition;  
}
```

一旦定义了结构体类型，它就能用于变量的声明，语法格式如下：

```
variable_name := structure_variable_type {value1, value2...valuen}  
或  
variable_name := structure_variable_type { key1: value1, key2: value2..., keyn: valuen}
```

实例如下：

```
package main  
  
import "fmt"  
  
type Books struct {  
    title string  
    author string  
    subject string  
    book_id int  
}  
  
func main() {  
  
    // 创建一个新的结构体  
    fmt.Println(Books{"Go 语言", "www.Golang.Ltd", "Go 语言教程", 6495407})  
  
    // 也可以使用 key => value 格式  
    fmt.Println(Books{title: "Go 语言", author: " www.Golang.Ltd ", subject: "Go 语言教程", book_id: 64  
95407})  
  
    // 忽略的字段为 0 或 空  
    fmt.Println(Books{title: "Go 语言", author: " www.Golang.Ltd "})  
}
```

输出结果为：

```
{Go 语言 www.Golang.Ltd Go 语言教程 6495407}  
{Go 语言 www.Golang.Ltd Go 语言教程 6495407}  
{Go 语言 www.Golang.Ltd 0}
```

访问结构体成员

如果要访问结构体成员，需要使用点号 `.` 操作符，格式为：

```
结构体.成员名"
```

结构体类型变量使用 `struct` 关键字定义，实例如下：

```
package main

import "fmt"

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

func main() {
    var Book1 Books      /* 声明 Book1 为 Books 类型 */
    var Book2 Books      /* 声明 Book2 为 Books 类型 */

    /* book 1 描述 */
    Book1.title = "Go 语言"
    Book1.author = " www.Golang.Ltd "
    Book1.subject = "Go 语言教程"
    Book1.book_id = 6495407

    /* book 2 描述 */
    Book2.title = "Python 教程"
    Book2.author = " www.Golang.Ltd "
    Book2.subject = "Python 语言教程"
    Book2.book_id = 6495700

    /* 打印 Book1 信息 */
    fmt.Printf( "Book 1 title : %s\n", Book1.title)
    fmt.Printf( "Book 1 author : %s\n", Book1.author)
    fmt.Printf( "Book 1 subject : %s\n", Book1.subject)
    fmt.Printf( "Book 1 book_id : %d\n", Book1.book_id)

    /* 打印 Book2 信息 */
    fmt.Printf( "Book 2 title : %s\n", Book2.title)
    fmt.Printf( "Book 2 author : %s\n", Book2.author)
    fmt.Printf( "Book 2 subject : %s\n", Book2.subject)
```

```
    fmt.Printf( "Book 2 book_id : %d\n", Book2.book_id)
}
```

以上实例执行运行结果为:

```
Book 1 title : Go 语言
Book 1 author : www.Golang.Ltd
Book 1 subject : Go 语言教程
Book 1 book_id : 6495407
Book 2 title : Python 教程
Book 2 author : www.Golang.Ltd
Book 2 subject : Python 语言教程
Book 2 book_id : 6495700
```

结构体作为函数参数

你可以像其他数据类型一样将结构体类型作为参数传递给函数。并以以上实例的方式访问结构体变量:

```
package main

import "fmt"

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

func main() {
    var Book1 Books    /* 声明 Book1 为 Books 类型 */
    var Book2 Books    /* 声明 Book2 为 Books 类型 */

    /* book 1 描述 */
    Book1.title = "Go 语言"
    Book1.author = " www.Golang.Ltd "
    Book1.subject = "Go 语言教程"
    Book1.book_id = 6495407

    /* book 2 描述 */
    Book2.title = "Python 教程"
    Book2.author = " www.Golang.Ltd "
    Book2.subject = "Python 语言教程"
    Book2.book_id = 6495700
}
```

```
/* 打印 Book1 信息 */
printBook(Book1)

/* 打印 Book2 信息 */
printBook(Book2)
}

func printBook( book Books ) {
    fmt.Printf( "Book title : %s\n", book.title);
    fmt.Printf( "Book author : %s\n", book.author);
    fmt.Printf( "Book subject : %s\n", book.subject);
    fmt.Printf( "Book book_id : %d\n", book.book_id);
}
```

以上实例执行运行结果为：

```
Book title : Go 语言
Book author : www.Golang.Ltd
Book subject : Go 语言教程
Book book_id : 6495407
Book title : Python 教程
Book author : www.Golang.Ltd
Book subject : Python 语言教程
Book book_id : 6495700
```

结构体指针

你可以定义指向结构体的指针类似于其他指针变量，格式如下：

```
var struct_pointer *Books
```

以上定义的指针变量可以存储结构体变量的地址。查看结构体变量地址，可以将 `&` 符号放置于结构体变量前：

```
struct_pointer = &Book1;
```

使用结构体指针访问结构体成员，使用 `."` 操作符：

```
struct_pointer.title;
```

接下来让我们使用结构体指针重写以上实例，代码如下：

```
package main

import "fmt"
```

```
type Books struct {
    title string
    author string
    subject string
    book_id int
}

func main() {
    var Book1 Books      /* Declare Book1 of type Book */
    var Book2 Books      /* Declare Book2 of type Book */

    /* book 1 描述 */
    Book1.title = "Go 语言"
    Book1.author = " www.Golang.Ltd "
    Book1.subject = "Go 语言教程"
    Book1.book_id = 6495407

    /* book 2 描述 */
    Book2.title = "Python 教程"
    Book2.author = " www.Golang.Ltd "
    Book2.subject = "Python 语言教程"
    Book2.book_id = 6495700

    /* 打印 Book1 信息 */
    printBook(&Book1)

    /* 打印 Book2 信息 */
    printBook(&Book2)
}

func printBook( book *Books ) {
    fmt.Printf( "Book title : %s\n", book.title);
    fmt.Printf( "Book author : %s\n", book.author);
    fmt.Printf( "Book subject : %s\n", book.subject);
    fmt.Printf( "Book book_id : %d\n", book.book_id);
}
```

以上实例执行运行结果为:

```
Book title : Go 语言
Book author : www.Golang.Ltd
Book subject : Go 语言教程
Book book_id : 6495407
Book title : Python 教程
Book author : www.Golang.Ltd
Book subject : Python 语言教程
```

Book book_id : 6495700



极客学院