

# 结构体

- 结构体类型的声明
- 结构体初始化
- 结构体成员访问
- 结构体传参

正文开始@比特科技

## 结构体的声明

### 结构体的基础知识

结构是一些值的集合，这些值称为成员变量。结构的每个成员可以是不同类型的变量。

### 结构的声明

```
struct tag
{
    member-list;
}variable-list;
```

例如描述一个学生：

```
typedef struct Stu
{
    char name[20]; //名字
    int age; //年龄
    char sex[5]; //性别
    char id[20]; //学号
}Stu; //分号不能丢
```

### 结构成员的类型

结构的成员可以是标量、数组、指针，甚至是其他结构体。

## 结构体变量的定义和初始化

有了结构体类型，那如何定义变量，其实很简单。

```
struct Point
{
    int x;
    int y;
}p1; //声明类型的同时定义变量p1
struct Point p2; //定义结构体变量p2

//初始化：定义变量的同时赋初值。
struct Point p3 = {x, y};

struct Stu //类型声明
{
```

```

char name[15]; //名字
int age;      //年龄
};
struct Stu s = {"zhangsan", 20}; //初始化

struct Node
{
    int data;
    struct Point p;
    struct Node* next;
}n1 = {10, {4,5}, NULL}; //结构体嵌套初始化

struct Node n2 = {20, {5, 6}, NULL}; //结构体嵌套初始化

```

## 结构体成员的访问

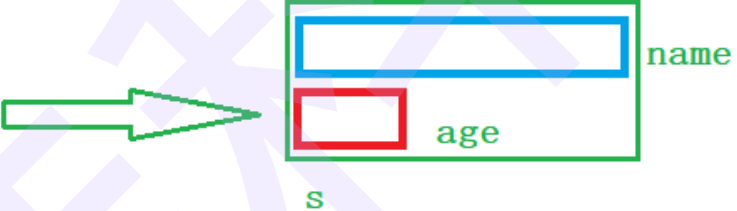
- 结构体变量访问成员 结构变量的成员是通过点操作符 (.) 访问的。点操作符接受两个操作数。例如：

```

struct Stu
{
    char name[20];
    int age;
};

struct Stu s;

```



我们可以看到 `s` 有成员 `name` 和 `age`；那我们如何访问 `s` 的成员？

```

struct S s;
strcpy(s.name, "zhangsan"); //使用. 访问name成员
s.age = 20; //使用. 访问age成员

```

- 结构体指针访问指向变量的成员 有时候我们得到的不是一个结构体变量，而是指向一个结构体的指针。

那该如何访问成员。如下：

```

struct Stu
{
    char name[20];
    int age;
};

void print(struct Stu* ps)
{
    printf("name = %s age = %d\n", (*ps).name, (*ps).age);
    //使用结构体指针访问指向对象的成员
    printf("name = %s age = %d\n", ps->name, ps->age);
}

int main()

```

```
{  
    struct Stu s = {"zhangsan", 20};  
    print(&s); //结构体地址传参  
    return 0;  
}
```

## 结构体传参

直接上代码：

```
struct S  
{  
    int data[1000];  
    int num;  
};  
  
struct S s = {{1,2,3,4}, 1000};  
//结构体传参  
void print1(struct S s)  
{  
    printf("%d\n", s.num);  
}  
//结构体地址传参  
void print2(struct S* ps)  
{  
    printf("%d\n", ps->num);  
}  
  
int main()  
{  
    print1(s); //传结构体  
    print2(&s); //传地址  
    return 0;  
}
```

上面的 print1 和 print2 函数哪个好些？

答案是：首选 print2 函数。原因：

函数传参的时候，参数是需要压栈的。如果传递一个结构体对象的时候，结构体过大，参数压栈的系统开销比较大，所以会导致性能的下降。

**结论：**结构体传参的时候，要传结构体的地址。

---

本章完



比特科技