

★什么是位段？

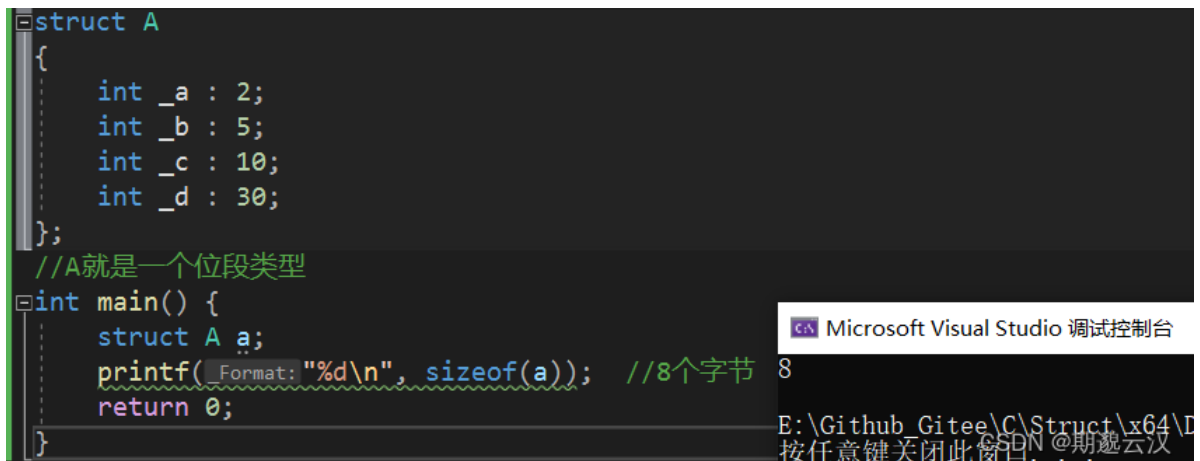
位段的声明和结构是类似的，有两个不同：

1. 位段的成员必须是 `int`、`unsigned int` 或 `signed int` 。
2. 位段的成员名后边有一个冒号和一个数字。

```
//位段
//
//1.位段的成员必须是 int、unsigned int 或signed int 。
//2.位段的成员名后边有一个冒号和一个数字。
//例如：
struct A
{
    int _a : 2;
    int _b : 5;
    int _c : 10;
    int _d : 30;
};
```

A就是一个位段类型

```
int main() {
    struct A a;
    printf("%d\n", sizeof(a)); //8个字节
    return 0;
}
```



```
struct A
{
    int _a : 2;
    int _b : 5;
    int _c : 10;
    int _d : 30;
};
//A就是一个位段类型
int main() {
    struct A a;
    printf("Format: \"%d\n\", sizeof(a)); //8个字节
    return 0;
}
```

Microsoft Visual Studio 调试控制台

8

E:\Github_Gitee\C\Struct\x64\I

按任意键关闭此窗口. . .

★考虑位段的内存分配

根据上面的例子，`每一个成员变量冒号：后面表示占用多少个bit位`

1. 位段的成员可以是 `int unsigned int signed int` 或者是 `char`（属于整形家族）类型。
2. 位段的空间上是按照需要以4个字节（`int`）或者1个字节（`char`）的方式来开辟的。
3. 位段涉及很多不确定因素，位段是不跨平台的，注重可移植的程序应该避免使用位段。

则考虑上面的结构体A

```
struct A
{
    //首先根据规则开辟4个字节，共32个比特位
    int _a : 2;    //剩余30个bit位
    int _b : 5;    //剩余25个bit位
    int _c : 10;   //剩余20个bit位
    int _d : 30;   //当d要使用时，发现不够30个bit位，那么再4个字节
};
```

那么最终就是开辟了8个字节。

考虑如下示例

```
struct S
{
    char a:3;
    char b:4;
    char c:5;
    char d:4;
};
struct S s = {0};
s.a = 10;
s.b = 12;
s.c = 3;
s.d = 4;
```

那么我们已经知道S将会占用3个字节，即24个bit位。

```
s.a = 10;    //1010  注意a存放时只能使用3个bit位
s.b = 12;    //1100
s.c = 3;     //011
s.d = 4;     //100
//来验证存放方式（总低位开始存储）
//高位 <----- 低位 | 高位 <-----低位 | 高位 <----- 低位 |
//0 b b b b a a a | 0 0 0 c c c c c | 0 0 0 0 d d d d
//0 1 1 0 0 0 1 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 1 0 0
// 四个一位数
//  6      2      0      3      0      4
```

那么在内存里s将会存放 6 2 0 3 0 4，调试查看结果如下：

```
int main() {
    //printf("%d\n", sizeof(s)); //3
    s.a = 10; //1010 注意a存放时只能使用3个bit位
    s.b = 12; //1100
    s.c = 3; //011
    s.d = 4; //100
    //来验证存放方式
    //0 b b b b a a a | 0 0 0 c c c c c | 0 0 0 0 d d d d
    //0 1 1 0 0 0 1 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 1 0 0
    // 四个一位数
    // 6 2 0 3 0 4
    return 0; 已用时间 <= 1ms
}
```

```
0x00007FF662D6C170 62 03 04 01
0x00007FF662D6C174 00 00 00 00
0x00007FF662D6C178 00 00 00 00
0x00007FF662D6C17C 00 00 00 00
0x00007FF662D6C180 78 10 d6 62
0x00007FF662D6C184 f6 7f 00 00
0x00007FF662D6C188 00 00 00 00
0x00007FF662D6C18C 00 00 00 00
0x00007FF662D6C190 02 00 00 00
0x00007FF662D6C194 00 00 00 00
0x00007FF662D6C198 00 00 00 00
0x00007FF662D6C19C 00 00 00 00
0x00007FF662D6C1A0 00 01 00 00
0x00007FF662D6C1A4 C5 00 00 00
```

位段的跨平台问题

1. int 位段被当成有符号数还是无符号数是不确定的。
2. 位段中最大位的数目不能确定。（16位机器最大16，32位机器最大32，写成27，在16位机器会出问题。
3. 位段中的成员在内存中从左向右分配，还是从右向左分配标准尚未定义。
4. 当一个结构包含两个位段，第二个位段成员比较大，无法容纳于第一个位段剩余的位时，是舍弃剩余的位还是利用，这是不确定的

跟结构相比，位段可以达到同样的效果，但是可以很好的节省空间，但是有跨平台的问题存在。