

★ 什么是柔性数组？

在C99中，结构中的**最后一个元素**允许是未知大小的数组，这就叫做『柔性数组』成员。

0长度的数组在ISO C和C++的规格说明书中是不允许的。这也就是为什么在VC++2012下编译你会得到一个警告：“arning C4200: 使用了非标准扩展：结构/联合中的零大小数组”。

我们在结构体里使用零长度数组，即未知的数组长度，这就是柔性数组。

★ 柔性数组特性

- 结构中的柔性数组成员前面必须至少一个其他成员。
- sizeof 返回的这种结构大小不包括柔性数组的内存。

```
//结构体里的最后一个成员 是未知大小的数组
struct S {
    int a;
    float ff;
    int arr[]; //这就是柔性数组成员
};

int main() {
    printf("%d\n", sizeof(struct S));
    return 0;
}
```

```
//结构体里的最后一个成员 是未知大小的数组
struct S {
    int a;
    float ff;
    int arr[]; //这就是柔性数组成员
};

int main() {
    printf("%d\n", sizeof(struct S));
    return 0;
}
```

CSDN @期邈云汉

```
int main() {
    printf("%d\n", sizeof(struct S));
    return 0;
}
```

Microsoft Visual Studio 调试控制台

8

E:\Github_Gitee\C\Dynamic_memory\
按任意键关闭此窗口 CSDN @期邈云汉

- 包含柔性数组成员的结构用malloc ()函数进行内存的动态分配，并且分配的内存应该大于结构的大小，以适应柔性数组的预期大小。

```
//结构体里的最后一个成员 是未知大小的数组
struct S {
    int a;
    float ff;
    int arr[]; //这就是柔性数组成员
};

int main() {
    //printf("%d\n", sizeof(struct S)); // 8
    struct S* s=(struct S*)malloc(sizeof(struct S) + sizeof(int) * 4); //希望在前面空间的基础上再加上4个整形的空间
    //这4个整形就存放在arr中
    if (s == NULL) {
        return 1;
    }

    s->arr[0] = 10;
    s->arr[1] = 60;
    s->arr[2] = 50;
    s->arr[3] = 100;

    for (int i = 0; i < 4; i++) {
        printf("%d \n", s->arr[i]);
    }

    return 0;
}
```

```
int main() {
    //printf("%d\n", sizeof(struct S)); // 8
    struct S* s=(struct S*)malloc(sizeof(struct S) + sizeof(int) * 4); //希望在前面空间的基础上再加上4个整形的空间
    //这4个整形就存放在arr中
    if (s == NULL) {
        return 1;
    }

    s->arr[0] = 10;
    s->arr[1] = 60;
    s->arr[2] = 50;
    s->arr[3] = 100;

    for (int i = 0; i < 4; i++) {
        printf("%d \n", s->arr[i]);
    }

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
10
60
50
100

E:\Github_Gitee\C\Dynamic_memory\Debug\Dynamic_memory.exe (进程 7612) 已退出，代码为 0。
按任意键关闭此窗口。 . . .
CSDN @期趣云汉
```

★ 柔性数组的优势

考虑未知大小的数组arr，当我们malloc开辟空间后，还可以再进行realloc对arr数组进行扩容，这就是柔性数组最大优势，数组可以动态扩容。

```
int main() {
    //printf("%d\n", sizeof(struct S)); // 8
```

```

    struct S* s=(struct S*)malloc(sizeof(struct S) + sizeof(int) * 4); //希望在前
面空间的基础上再加上4个整形的空间
    //这4个整形就存放在arr中
    if (s == NULL) {
        return 1;
    }

    s->arr[0] = 10;
    s->arr[1] = 60;
    s->arr[2] = 50;
    s->arr[3] = 100;

    for (int i = 0; i < 4; i++) {
        printf("%d \n", s->arr[i]);
    }


    struct S* ptr = (struct S*)realloc(s, sizeof(struct S) + sizeof(int) * 10);
    //对数组arr扩容到10个int

    for (int i = 0; i < 10;i++) {
        ptr->arr[i] = i;
    }

    for (int i = 0; i < 10; i++) {
        printf("%d ", ptr->arr[i]);
    }

    return 0;
}

```

 Microsoft Visual Studio 调试控制台

```

10
60
50
100
0 1 2 3 4 5 6 7 8 9
E:\Github_Gitee\C\Dynamic_memory\Debug\Dynamic_
按任意键关闭此窗口. . . _
CSDN @期邈云汉

```

★ 新的写法

在结构体里我们可以使用一个指针，来代表我们的数组指针，在后续单独为该数组指针申请空间并进行赋值。

```

//对柔性数组还可以重新设计
struct S {
    int n;
    float f;
    int* arr;
};

int main() {

```

```

struct S* ps = (struct S*)malloc(sizeof(struct S)); //先为int 和 float申请空间
if (ps == NULL) {
    return 1;
}
ps->n = 10;
ps->f = 10.10f;

int* ptr = (int *)malloc(5 * sizeof(int));
if (ptr == NULL) {
    return 1;
}
else {
    ps->arr = ptr;
}

for (int i = 0; i < 5; i++) {
    scanf("%d", &ps->arr[i]);
}

printf("%d\n%f\n", ps->n, ps->f);
for (int i = 0; i < 5; i++) {
    printf("%d ", ps->arr[i]);
}
return 0;
}

```

Microsoft Visual Studio 调试控制台

```

9 8 5 2 7
10
10.100000
9 8 5 2 7
E:\Github_Gitee\C\Dynamic_memory\Debug\
按任意键关闭此窗口. . . _CSDN @期邈云汉

```