

一、结构体与联合体 (Struct & Union)

(一) 结构体 (Struct)

☑ 什么是结构体?

结构体 (Struct, 全称 Structure) 是**把不同类型的数据打包在一起**的容器。

就像一个“文具盒”，里面可以放铅笔 (int)、橡皮 (char)、尺子 (float)，这些都是不同种类的“变量”。

举个例子:

假设你是老师，想记录一个学生的:

- 姓名 (name) → 字符串 (String)
- 年龄 (age) → 整数 (int)
- 成绩 (score) → 小数 (float)

你可以用结构体来把这些信息装在一起:



```
#include <iostream>

using namespace std;

struct Student {

    string name;    // 姓名

    int age;        // 年龄

    float score;    // 成绩

};

int main() {

    Student s1;    // 定义一个学生

    s1.name = "小明";

    s1.age = 10;

    s1.score = 92.5;

    cout << s1.name << ", " << s1.age << "岁, 成绩是:" << s1.score
```



```
<< endl;

}
```

结构体的考试易错点

易错点

正确方式说明

忘记 struct 关键字

必须以 struct 开头定义结构体

用点号 (.) 访问错字段

写成 s1.name, 不要漏掉 .

没有初始化结构体

使用前最好先赋值

(二) 联合体 (Union)

☑ 什么是联合体?

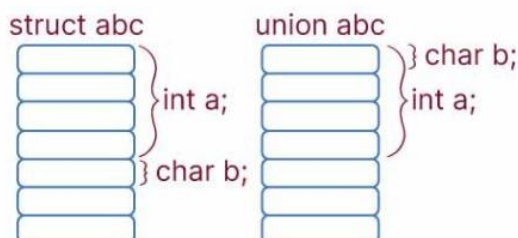
联合体 (Union) 也是一种“打包变量”的工具, 但它的特点是: **所有成员共用同一块内存。**

就像一本“共享日记”, 你和朋友轮流写, 但只能一个人写, 之前的就会被覆盖。

举个简单例子:

我们想表示一段数据, 可能是:

- 一个整数 int,
- 一个字符 char,
- 一个小数 float。



但它们不会同时出现, 就可以用联合体节省空间!

```
#include <iostream>

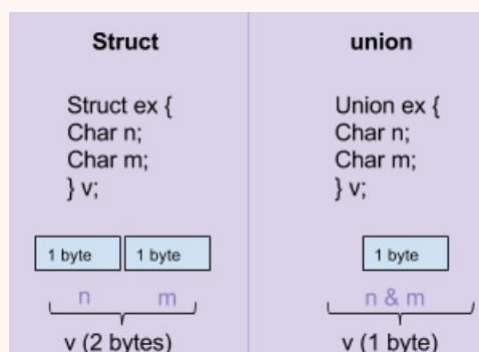
using namespace std;

union Data {

    int i;

    char ch;

    float f;
```



```

};

int main() {

    Data d;

    d.i = 100;

    cout << "整数: " << d.i << endl;

    d.ch = 'A'; // 再写字符, 覆盖了整数

    cout << "字符: " << d.ch << endl;

    cout << "此时整数已被覆盖: " << d.i << endl;

}

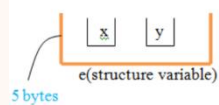
```

Structure

```

struct Emp
{
    char x;    //size 1 byte
    float y;   //size 4 byte
};

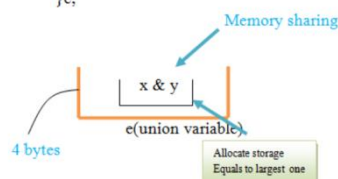
```

Unions

```

union Emp
{
    char x;
    float y;
};

```

**类比说明:**

- 结构体 (Struct) = 三个人有自己的本子, 互不影响。
- 联合体 (Union) = 三个人共用一本本子, 你写的时候我就不能写。

联合体的考试易错点:**易错点**

不清楚为什么值被覆盖

用错数据类型查看值

访问多个成员同时

正确方式说明

联合体所有成员共用内存, 后写的会覆盖前面的

写了小数再用整数取, 值可能“乱码”

联合体不能同时取多个字段的值

