# 复合类型

* 影响最为深远的符合类型是类；
* 指针是将数据地址告诉计算机的变量；
* 如果数组元素没有初始化，则元素的值就是之前驻留在该内存单元里头的值；
* Sizeof是个运算符，不是个函数，返回数据对象的数据长度 单位是字节byte；
* 数组只有在 刚开始定义的时候才能够初始化，只有这一次 机会，以后再初始化都是违法的；
* 如果只对字符型的数组的一部分元素进行初始化，则剩下的元素自动补为0；
* 对数组进行初始化操作的时候 不允许 数据截断 发生long A[1] = {1.0}是违法的；

# 字符串

* 字符串必须以‘\0’结尾，否则那只是一个字符数组；
* Char fish [] = “bubble”;也是一种安全的字符串初始化的方法；
* 在确定乘装字符串的数组的大小的时候 必须 算上末尾的’\0’;既是目标字符串的长度+1；
* Cin 通过空白字符（制表符，空格，enter来确定字符串结束的位置）；cin每次只读取一个单词；
* Cin.getline(字符串数组，目标字符串长度)；在读到换行符或者读到字符串长度的时候停止，**并且自动丢弃换行符**；
* Cin.get 读取一整行，但不读取换行符；
* 函数重载现象：cin.get(字符串数组，目标字符串长度).get(); 第一个get（）用于读走一个换行符之前的字符串，然后返回一个cin对象，再使用这个对象的get方法读走后面的换行符；**（注：最多也就只能是cin.get().get() 不能再叠加一层了）**
* 要使用string类 必须要在头文件加上 #include<string> string类位于命名空间std之中，string类定义隐藏了字符串数组的性质，就像处理普通变量那样处理string就行了；
* 可以将字符串数组 char ch[] 视为一组用于存储一个字符串的char存储单元，而string类变量是一个表示字符串的实体；
* stringName.length() stringName.size() 都是统计字符串长度的方法；
* 在字符型数组没有初始化的时候 因为不知道数组的每个存储单元里面有没有‘\0’或者不知道‘\0’到底在哪 所以 用 strlen()函数得到的长度肯能会 > 或者 < 字符串数组的定义长度 这是正常的；
* 字符串的输入方法：
  + 基于字符串数组：char str1[20];
    - * Cin>>str1;
      * Cin.get(str1, 20).get();
      * Cin.getline(str1,20);
      * char str1[20] = {“dfghjk”}; char str1 = {“dfghjk”}; char str1 {“dfghjk”};
  + 基于字符串类：string str2；
    - * Getline(cin,str2); 其中 cin作为参数指出去哪找输入，str2没有指明长度 因为string的字符串长度是自由变化的；
* raw字符串叫做 原始字符串，字符表示的就是自己，以R作为标识，“（ 和 ）”作为定界符，例如 cout << R “( “King” uses”\n” instead of endl.)”; “ \*( )\*”为自定义的定界符；
* 在原始字符串中 输入回车 不仅会在控制台换行 还会再原始字符串中添加一个回车字符；

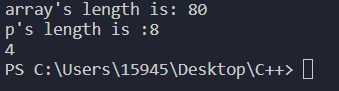
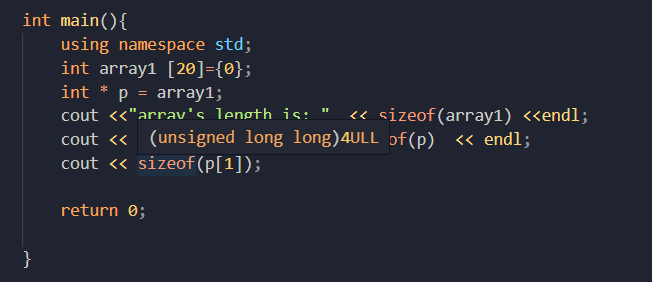
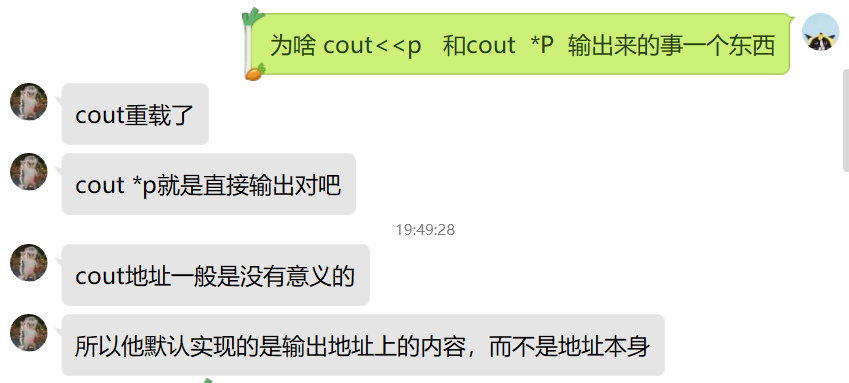
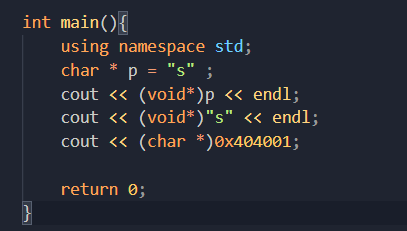
# 结构体

* C++在定义结构体变量的时候省去关键字 struct；
* 对于 struct Node node = {}; 则会导致node结构体中的所有 成员都设置为0或‘\0’

# 共用体

* 共用体的长度就是共用体成员的最大长度；

# 指针

* \*被称为间接值运算符 或者 解除引用运算符；
* 定义指针的时候必须要声明类型；
* Int\* 要放在一起，是一种复合类型，既是指向int型的指针；
* 一定要在指针运用\*之前将指针初始化为一个确定的，适当的地址，这是关于正确使用指针的金科玉律；
* 用new来分配内存语法格式：typeName \* pointerName = new typeName;（int \* p = new int；） new 运算符根据 变量的类型来确定确定需要多少字节来分配内存； 称 p 指向一个数据对象；
* 变量的值都存储在栈内存区域中，new从堆 或自由存储区的内存分配区域分配内存；
* 内存释放： delete pointName; 这将会释放指针指向的内存，但指针本身不会被删除；
* delete pointName; delete pointName; 连续两句 执行的结果将是不可预见的 不要玩火；
* New 创建动态数组： int \* Arry = new int [10];
* Delete 删除动态数组： delete [] Arry;
* **注意：如果使用new[] 来为数组分配内存，则必须使用delete[] 释放；如果用new分配实体内存 必须使用delete（没有[]）来释放内存；总结来讲就是 delete 后面[]的有无 必须和new配套；**
* arryName[i] = \*(arryName + i);
* 
* short tell[20]; &tell = Short (\*)[20]; 即&tell 表示指向包含20个元素的short型数组；
* Short \* p [20] 表示 包含20个short型**指针**元素的**数组**，即是个数组但数组成员全是short型的指针；
* Short (\*p) [20] 表示包含20个short型元素的数组；
* 在C++中：如果给cout提供一个字符的地址，它将从该字符开始打印，直到遇到空字符‘\0’为止；“asdsdf”传给cout的时候 也是一个首地址； 让一个字符指针指向一个字符串也是同样的解决办法；
* 在C++中 char数组名 char指针 以及用“”表示的字符串常量都被解释为字符串的首地址；
* 
* 
* 
* 要想输出字符串的地址则必须强制转化为另一种指针类型；
* Const char \* bird = ”wuya”； cout << bird; 等价于 cout << “wuya”;
* 
* 用new创建结构体指针： structName \* pointName = new structName;
* 变量的使用寿命：静态变量存在于程序的整个生命周期中，自动变量只存在于声明他的代码块中；
* 自动变量通常存储在栈中；
* 在栈中添加和删除的内存总是连续的；
* 实在搞不清楚变量类型的时候 可以用 auto 类型 编译器知道是什么类型；
* 声明模板类 vector #include<vector> using namespace std; vector <typeName> vectorName （vectorLength ） **其中vectorLength可以是变量 也可以是常量；**
* 声明模板类 array #include<array> using namespace std; array < typeName , arrayLength> arrayName; **其中 arrayLength只能是常量；**
* ~~vector <int> v1（4）; vector <int> v2（4）; v2 = v1; 可以整体进行赋值；~~
* array < double, 4> arr1; array < double, 4> arr2; arr2 = arr1; array 对象可以整体进行赋值；