**共用体、枚举类型、typedef**

1. 共用体
   1. 定义
      1. 几个变量占用同一块内存
   2. 结构与结构体差不多，只是把关键字改成 union
   3. 共用体内存等于等于定义的变量中最长的内存
      1. 例如

union Student{

int num; //4个

char name[100]; //100个字节

};

int main(int argc, char \*argv[])

{

printf("%d", sizeof(union Student)); //100

return 0;

}

* 1. 共用体变量的特点
     1. 同一内存存放多中内存，每一时刻只有一种类型起作用 ！！！
     2. 例如
        1. union Student a；
        2. a.num = 100； //现在存放得是num
        3. Strcpy(a.name, “weierLin”); //现在存放的是name， 并且把num冲掉了
     3. 共同体首地址与变量的地址是一样的且不能进行成员的初始，但是只给1个变量赋值是可以的
     4. 共用体不能作为函数参数

1. 枚举类型
   1. 现在我们有四种颜色
   2. 结构

enum Color{

Red, //默认是 0，

Green,

Black = 17, 那就Yellow 就变成 18

Yellow

};

* 1. 所以 Red、Green、Blac、Yellow 是枚举常量
  2. enum color c1； // c1 是枚举变量
  3. 那我们不能用 c1.Red //枚举变量只能接收枚举常量的赋值
  4. C1 = Red; // 就是这样
  5. 那enum 有什么用吗
     1. 比如，我们充值会员，有黄砖，蓝砖 ……
     2. 如果我们总不能用0， 1， 2来操作
     3. 所以我们的枚举常量就派上用场了

1. Typedef 来定义类型
   1. 来定义新的类型名
      1. typedef int Int; // 我们用Int 来 替换 int 来整整型的定义
      2. 定义结构体类型
         1. Dypedef struct date{

Int year;

Int month;

}Date; //注意这不是之前讲的定义结构体名哦

* + - 1. 那么现在我们定义结构体变量就可以用下面的方式了
         1. Date data1； // 现在这样就可以了，是不是很棒的呀
      2. 那有什么用处呀
         1. 例如我们之前定义了很多的 int 类型的 变量，但是现在我们的int类型的存不下去了，当然沃恩可以利用 IDE 的全局替换模式
         2. 如果我们在程序中用 typedef int INT
         3. INT a, b, c, ……
         4. 那我们在后续的修改中就只修改一行就够了 typedef double INT

struct birth{

int year;

int month;

int day;

};

struct Student{

int num;

char name[10];

char ads[100];

struct birth birthday;

};

1. 结构体的引用
   1. 用 . 符号
      1. S1.num= 1001;
      2. S1.birthday.year = 1996
2. 结构体初始化

struct Student s = {007, "王五", "北京",2019, 2, 2};