c语言基础知识学习

常见问题记录

一、数据溢出问题解决

1.使用宏定义限制

> #define MAX_SIZE N

二、常用占位符

> %d——int 接受整数值并将它表示为有符号的十进制整数
> %f——float 单精度浮点数
> %c——char 字符型，可以把输入的数字按照ASCII码相应转换为对应的字符
> %s——char * 字符串，输出字符串中的数字直至字符串中的空字符（字符0）
> **mac当中的特殊占位符**
> %zu——用于正确输出size_t 类型

三、类型转换

1.自动转换（隐式转换）

> 遵循一定的规则，由编译系统自动完成
> 2.强制类型转换
> 把表达式的运算结果强制转换成所需的数据类型
> ***转换原则：占用内存字节数少（值域小）的类型，向占用内存字节数多（值域大）的类型转换，以保证精度不降低。***

四、循环结构

代码中，如果while括号中一直为真，会陷入死循环
1.for 和 while 循环

> 规则：循环的条件表达式后面绝对不能有分号，除非你故意想要一个空循环体。

2.do-while 循环

> 规则：do-while 循环是一个例外，它在整个语句的末尾必须有一个分号。
> 这是因为它的语法结构是 do { ... } while (...);
> 忘记这个分号会导致编译错误。

- 除了 do-while，其他循环的行末都不要轻易写分号。
- 如果你写的循环没有用大括号 {}，那么循环体就是紧跟着循环语句的第一条语句。
- 一个单独的分号 ; 就是一条合法的语句，称为“空语句”。

3.嵌套循环

> 外层运行一次，内层运行一圈（一个周期）

4.区分while和for的运行条件

> **循环次数不确定（相比于for而言最为核心的原因）**
> 例如：字符串的编写常常用while，是因为不知道字符串具体的长度，但是知道字符串在“\0”处停止
> 此时条件ch1[i]=\0 可以很好定义循环的终止条件

五、随机数

产生随机数：

> 1.导入头文件 time.h；stdlib.h
> 2.添加随机数种子(若不添加随机数种子，则每次得到的数是一样的）
> 例如：获得一个0～9的随机数
> srand((unsigned int)time(NULL)); //每次随机数不一样
> int value = rand()%10; //0~9
> 3.获取随机数

六、数组和字符串

1.二维数组

> *arr[i][j]其中，数组名部分可以省略，其它部分不可以省略
> “二维数组大小：%d\n”,sizeof(arr)
> “二维数组一行大小：%d\n”,sizeof(arr[0])——表示第一行大小
> “二维数组元素大小：%d\n”,sizeof(arr[0][0])——表示第一行第一列的元素大小

> “二维数组行数：%d\n”,sizeof(arr)/sizeof(arr[0])
> “二维数组列数：%d\n”,sizeof(arr[0])/sizeof(arr[0][0])

七、函数

1.函数的调用

> #include<time.h>
> time_t time(time_t*t);
> 功能：获取当前系统时间
> 参数：常设置为NULL
> 返回值：当前系统时间，time_t相当于long类型，单位为毫秒

> #include<stdlib.h>
> void srand(unsigned int seed);
> 功能：用来设置rand（）产生随机数时的随机种子
> 参数：如果每次seed相等，rand（）产生随机数相等
> 返回值：无

> #include<stdlib.h>
> int rand（void）
> 功能：返回一个随机数值
> 参数：无
> 返回值：随机数

2.随机数举例应用

```
#include<stdlib.h>
#include<time.h>
#include<stdio.h>
int main()
{
srand((size_t)time NULL);
for (int i = 0; i < 100; i++)
{
printf("%d\n",rand()%51+50);—>表示取模于50～100之间的数值
}
return 0;
```

> }
> 该代码可以表示随机50～100之内的数字

3.函数的定义与使用&返回值

💡 记忆技巧

函数类型	定义方式	是否需要return	调用方式
有返回值	`int func()`	必须return值	`int x = func();`
无返回值	`void func()`	不需要return或`return;`	`func();`
main函数	`int main()`	需要`return 0;`	系统调用

函数的返回值是通过函数中的return语句获得的，return后面的值也可以是一个表达式

> A）尽量保证return语句中表达式的值和函数返回类型是同一类型
> B）如果函数返回的类型和return语句中表达式的值不一致，则以函数返回类型为准，即函数返回类型决定返回值的类型。对数值型数据，可以自动进行类型转换。
> C）return语句的另一个作用为中断return所在的执行函数，类似于break中断循环、switch语句一样。
> D）如果函数带返回值，return后面必须跟着一个值，如果函数没有返回值，函数名字的前面必须写一个void关键字，这时候，我们写代码时也可以通过return中断函数（也可以不用），只是这时，return后面不带内容（分号“；”除外）
> **最好有void关键字**
> **如果函数返回的类型和return语句中表达式的值类型不一致，而他又无法自动进行类型转换，程序则会报错**

4.无参函数调用

如果是调用无参函数，则不能加上“实参”，但括号不能省略

> 例如：
> void test（）
> {}
> int main（）
> {

> //函数的调用
> test（）；//right，圆括号（）不能省略
> test（250）；//error，函数定义时没有参数
> return 0；
> }
> **void类型不可以直接定义数据**
> **void类型可以作为函数的返回值类型 表示没有返回值**

5.函数的声明
注意：一个函数只能被定义一次，但可以声明多次

> 函数声明的三种方式：
> //extern int add01(int a, int b)
> //int add01(int a, int b)
> //int add01(int,int) 在这种声明中，int后面可以加上其他字母，仅仅表示一种样式

声明和定义的区别

> 声明变量不需要建立存储空间，如：extern int a
> 定义变量需要建立存储空间，如：int b
> 从广义的角度来讲声明中包含着定义，即定义是声明的一个特例，所以并非所有的声明都是定义
> 例如：
> int b它既是声明，同时又是定义
> 对于extern b来讲它只是声明，不是定义

一般情况下，把建立存储空间的声明称之为“定义”，而把不需要建立存储空间的声明称之为“声明”

八、指针

1.野指针和空指针

- & 取地址：升维度
  - 值 → 指针（0维 → 1维）
  - 指针 → 指向指针的指针（1维 → 2维）
- *** 解引用：降维度**
  - 指向指针的指针 → 指针（2维 → 1维）
  - 指针 → 值（1维 → 0维）

2.万能指针void

> void*指针可以指向任意变量的内存空间：
> 万能指针可以接受任意类型变量的内存地址
> 再通过万能指针修改变量的值时，需要找到变量对应的指针类型
> *void指针不能直接解引用
> *void指针不能进行算术运算——要先转换类型在解引用

3.const修饰的指针变量

> **Const修饰靠其近的东西，被其修饰的东西不可以被修改：**
> 1.const修饰指针类型（const int *p）——指针地址
> 可以修改指针变量的值，不可以修改指针指向内存空间的值
> 2.const修饰指针变量（int *const p*）——*指针常量*
> *可以修改指针指向内存空间的值，不可以修改指针变量的值*
> *3.只读指针（const int*p）
> 既不可以直接修改指针变量的值，也不可以直接修改指针指向内存空间
> 要用二级指针进行修改
> **const修饰总是有点不安全**
> *记忆方法*:
> const int *p; // const在*左边：指向的值是常量
> int *const p; // const在*右边：指针本身是常量

4.多级指针（int**pp）：

- 第一个*：pp是一个指针
- 第二个*：pp指向的内容也是一个指针
- int：最终指向的数据是int类型

5.指针和数组

数组名字是数组的首元素地址，但它是一个常量
数组名是一个常量，不允许赋值
数组名是数组首元素地址

6.值传递和地址传递

> 值传递：形参不影响实参的值
> 地址传递：形参可以改变实参的值
> **数组名作函数参数，函数的形参会退化为指针**

7.指针和字符串

> char ch[]="hello world"; //栈区字符串
> char*p="hello world"; //数据区常量区字符串—->常量不可以被修改
> **这两种表示方法中，均可以打印出来该字符串，但是只有第一种可以直接修改字母，第二种不可以**

end.....