**在宏定义中常数的定义最为简单。**

比如 #define N 10 也可以这样 #define N (10)

**现在注意的就是在宏定义中对于表达式一定要加上括号!!**

比如：#define SNU (-1)

#define KUY (10-1)

typedef和#define的区别：

#define shao char\*

int main()

{

shao c,d;

char a=1;

c=&a;

d=&a;

printf("%d,%d",\*c,\*d);

return 0;

}

这样会把shao c,d;翻译成char\* c,d;也就是说d不是char\* 指针变量，而是char型变量。

而

typedef char\* shao;

int main()

{

shao c,d;

char a=1;

c=&a;

d=&a;

printf("%d,%d",\*c,\*d);

return 0;

}

这样会把shao c,d;翻译成char\* c; char\* d; 编译连接就没有错误。

**接下来就是带参数的宏**

**参数一定要用括号括起来！！**

比如：#define STRUCTURE(type) struct type

定义好了，就可以写成如下的命令：

STRUCTURE(Rectangle)

{

double length;

}

预处理之后变为：

struct Rectangle

{

double length;

}

**跨行的宏**

“反斜线+ENTER”将宏切分为数行

#define shao \

char\*

等同于 #define shao char\*

**##的用法**

type\_function中的type宏定义无法将其定位参数

type##\_function 这样表面type是参数，所以宏定义可以将其取代

**多参数宏定义**

比如 #define max(a,b) ((a)>(b)? (a):(b))

注意：传给宏的参数应避免++ --运算

**传给宏函数的参数不限定类型**

比如 #define add(x,y) ((x)+(y))

这里我们可以这样使用 int a=add(1,2); 也可以float a=add(1.5,2.4);

#define 是用来定义宏的 而#undef是用来取消宏定义

所以#define ~ #undef 说明了宏的有效范围。

**条件性编译**

两大优点：

提高查错能力

增加程序的可移植性

#if

#else

#endif

**条件性定义**

#ifdef #ifndef

#else #else

#endif #endif