1：c语言中的宏是预处理命令，它的原理只是简单的字符串替换。

该命令有两种格式：一种是简单的宏定义，另一种是带参数的宏定义。

(1) 简单的宏定义：

#define <宏名> <字符串>

例： #define PI 3.1415926

(2) 带参数的宏定义

#define <宏名> (<参数表>) <宏体>

例： #define A(x) x

一个标识符被宏定义后，该标识符便是一个宏名。这时，在程序中出现的是宏名，在该程序被编译前（由编译器来进行文本替换，编译后预处理命令就不存在了，所以预处理命令不占内存空间），先将宏名用被定义的字符串替换，这称为宏替换，替换后才进行编译，宏替换是简单的替换。

2：define的多行定义

define可以替代多行的代码，例如MFC中的宏定义（非常的经典）

```
#define MACRO(arg1, arg2)  \
/* declarations */  \
stmt1;  \
stmt2;  \
.....  \
```

tip:最后一条语句没有分号；。

3：GCC新增的功能

```
#define xxx() {}
```

标准C支持的

```
#define xxx() ({})
```

GCC新增的功能，主要为了防止宏展开出现问题，默认展开时是要加上一个;的，容易出问题， 最后一个式子的返回值作为宏函数的返回值。

```
#define A(a,b,c) ({a=1;b+=1;c=3;a+b+c;})
#include <stdio.h>
int main()
{
    int a;
    int b=1;
```

```
    int c;
    int d;
    d=A(a,b,c);
    printf("%d,%d,%d,%d/n",a,b,c,d);
    return 0;
}
```

表示该宏函数还有返回值，最后一个式子的返回值作为宏函数的返回值。

运行结果：

1,2,3,6

4：特殊的用法 ## ， #@， #

```
#define Conn(x,y) x##y
#define ToChar(x) #@x
#define ToString(x) #x
```

x##y表示什么？表示x连接y，举例说：

int n = Conn(123,456)； 结果就是n=123456;

char* str = Conn("asdf", "adf")结果就是 str = "asdfadf";

再来看#@x，其实就是给x加上单引号，结果返回是一个const char。举例说：

char a = ToChar(1);结果就是a='1';

做个越界试验char a = ToChar(123);结果是a='3';

但是如果你的参数超过四个字符，编译器就给给你报错了！ error C2015: too many characters in constant ： P

最后看看#x,估计你也明白了，他是给x加双引号

char* str = ToString(123132);就成了str="123132";