程序设计基础

第 10 章: 结构化程序设计

刘新国

浙江大学计算机学院
浙江大学 CAD&CG 国家重点实验室

December 14, 2021

内容提要

内容提要

内容提要

基本宏定义

什么是宏定义 (#define)

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能
- 定义一些符号常量，方便编写和维护程序，例如：

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能
- 定义一些符号常量，方便编写和维护程序，例如：

```
#define PI 3.1415926
```

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能
- 定义一些符号常量，方便编写和维护程序，例如：

```
#define PI 3.1415926
```

- 以后可以在程序中用 PI 代替 3.1415926，方便美观

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能
- 定义一些符号常量，方便编写和维护程序，例如：

```
#define PI 3.1415926
```

- 以后可以在程序中用 PI 代替 3.1415926，方便美观
- 宏定义的作用范围（有效范围）：

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能
- 定义一些符号常量，方便编写和维护程序，例如：

```
#define PI 3.1415926
```

- 以后可以在程序中用 PI 代替 3.1415926，方便美观
- 宏定义的作用范围（有效范围）：
  - 从定义开始，到文件结束

基本宏定义

什么是宏定义 (#define)

- 宏定义是 C 语言的常用功能
- 定义一些符号常量，方便编写和维护程序，例如：

```
#define PI 3.1415926
```

- 以后可以在程序中用 PI 代替 3.1415926，方便美观
- 宏定义的作用范围（有效范围）：
  - 从定义开始，到文件结束
  - 类似于全局变量的作用范围

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 例如:

```
#define PI                  3.1415926
#define Mile2Meter          1609
#define Foot2Centimeter     30.48
#define Inch2Centimeter     2.54
```

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 例如：

```
#define PI                  3.1415926
#define Mile2Meter          1609
#define Foot2Centimeter     30.48
#define Inch2Centimeter     2.54
```

- 我们可以在程序中用这些宏（PI, Mile2Meter, ...）代替这些魔鬼数字，方便美观

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 例如：

```
#define PI                  3.1415926
#define Mile2Meter          1609
#define Foot2Centimeter     30.48
#define Inch2Centimeter     2.54
```

- 我们可以在程序中用这些宏（PI, Mile2Meter, ...）代替这些魔鬼数字，方便美观
- 编译器在编译程序的时候会自动的将这些宏（PI, Mile2Meter, ...）的还原为它们相应的宏体字符串

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 例如：

```
#define PI                  3.1415926
#define Mile2Meter          1609
#define Foot2Centimeter     30.48
#define Inch2Centimeter     2.54
```

- 我们可以在程序中用这些宏（PI, Mile2Meter, ……）代替这些魔鬼数字，方便美观
- 编译器在编译程序的时候会自动的将这些宏（PI, Mile2Meter, ……）的还原为它们相应的宏体字符串
- 宏定义的作用范围（有效范围）：

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 例如：

```
#define PI                  3.1415926
#define Mile2Meter          1609
#define Foot2Centimeter     30.48
#define Inch2Centimeter     2.54
```

- 我们可以在程序中用这些宏（PI, Mile2Meter, ……）代替这些魔鬼数字，方便美观
- 编译器在编译程序的时候会自动的将这些宏（PI, Mile2Meter, ……）的还原为它们相应的宏体字符串
- 宏定义的作用范围（有效范围）：
  - 从定义开始，到文件结束

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 例如：

```
#define PI                  3.1415926
#define Mile2Meter          1609
#define Foot2Centimeter     30.48
#define Inch2Centimeter     2.54
```

- 我们可以在程序中用这些宏（PI, Mile2Meter, ……）代替这些魔鬼数字，方便美观
- 编译器在编译程序的时候会自动的将这些宏（PI, Mile2Meter, ……）的还原为它们相应的宏体字符串
- 宏定义的作用范围（有效范围）：
  - 从定义开始，到文件结束
  - 类似于全局变量的作用范围

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

1 英尺 = 30.48 厘米

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

1 英尺 = 30.48 厘米　　厘米数 = 英尺数 * 30.48

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

1 英尺 = 30.48 厘米　　厘米数 = 英尺数 * 30.48

1 英寸 = 2.54 厘米

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米	米数 = 英里数 * 1609
1 英尺 = 30.48 厘米	厘米数 = 英尺数 * 30.48
1 英寸 = 2.54 厘米	厘米数 = 英寸数 * 2.54

例程 10-7，单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

1 英尺 = 30.48 厘米　　厘米数 = 英尺数 * 30.48

1 英寸 = 2.54 厘米　　厘米数 = 英寸数 * 2.54

```
float mile, foot, inch;
scanf("%f%f%f", &mile, &foot, &inch);
printf("%f英里=%f米\n", mile, mile*1609);
printf("%f英寸=%f厘米\n", foot, foot*30.48);
printf("%f英尺=%f厘米\n", inch, inch*2.54);
```

例程 10-7, 单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

1 英尺 = 30.48 厘米　　厘米数 = 英尺数 * 30.48

1 英寸 = 2.54 厘米　　厘米数 = 英寸数 * 2.54

```
float mile, foot, inch;
scanf("%f%f%f", &mile, &foot, &inch);
printf("%f英里=%f米\n", mile, mile*1609);
printf("%f英寸=%f厘米\n", foot, foot*30.48);
printf("%f英尺=%f厘米\n", inch, inch*2.54);
```

1609, 30.48, 2.54 都是魔鬼数字

例程 10-7, 单位换算

编写程序，将输入的英里数转换为米数，英尺数转换为厘米数，英寸数转换为厘米数。

1 英里 = 1609 米　　米数 = 英里数 * 1609

1 英尺 = 30.48 厘米　　厘米数 = 英尺数 * 30.48

1 英寸 = 2.54 厘米　　厘米数 = 英寸数 * 2.54

```
float mile, foot, inch;
scanf("%f%f%f", &mile, &foot, &inch);
printf("%f英里=%f米\n", mile, mile*1609);
printf("%f英寸=%f厘米\n", foot, foot*30.48);
printf("%f英尺=%f厘米\n", inch, inch*2.54);
```

1609, 30.48, 2.54 都是魔鬼数字

要避免在程序中出现魔鬼数字

例程 10-7, 单位换算

利用 C 语言的宏定义消除魔鬼数字

```
#define Mile2Meter 1609
#define Foot2Centimeter 30.48
#define Inch2Centimeter 2.54
float mile, foot, inch;
scanf("%f%f%f", &mile, &foot, &inch);
printf("%f英里=%f米\n", mile,
                        mile*Mile2Meter);
printf("%f英寸=%f厘米\n",foot,
                        foot*Foot2Centimeter);
printf("%f英尺=%f厘米\n",inch,
                        inch*Inch2Centimeter);
```

例程 10-7, 单位换算

利用 C 语言的宏定义消除魔鬼数字

```
#define Mile2Meter 1609
#define Foot2Centimeter 30.48
#define Inch2Centimeter 2.54
float mile, foot, inch;
scanf("%f%f%f", &mile, &foot, &inch);
printf("%f英里=%f米\n", mile,
                        mile*Mile2Meter);
printf("%f英寸=%f厘米\n",foot,
                        foot*Foot2Centimeter);
printf("%f英尺=%f厘米\n",inch,
                        inch*Inch2Centimeter);
```

这些宏所定义的数值是常量，不可改变

例程 10-7, 单位换算

利用 C 语言的宏定义消除魔鬼数字

```
#define Mile2Meter 1609
#define Foot2Centimeter 30.48
#define Inch2Centimeter 2.54
float mile, foot, inch;
scanf("%f%f%f", &mile, &foot, &inch);
printf("%f英里=%f米\n", mile,
                        mile*Mile2Meter);
printf("%f英寸=%f厘米\n",foot,
                        foot*Foot2Centimeter);
printf("%f英尺=%f厘米\n",inch,
                        inch*Inch2Centimeter);
```

这些宏所定义的数值是常量，不可改变

增加了程序的可读性

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符
  - 单词（字母、下划线、数字组成，不能数字开头）

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符
  - 单词（字母、下划线、数字组成，不能数字开头）
- 宏体可以是任意的字符串

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符
  - 单词（字母、下划线、数字组成，不能数字开头）
- 宏体可以是任意的字符串
- 程序编译时，所有宏名出现的地方被替换为宏体字符串

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符
  - 单词（字母、下划线、数字组成，不能数字开头）
- 宏体可以是任意的字符串
- 程序编译时，所有宏名出现的地方被替换为宏体字符串

  注意：所有作为单词出现的宏名

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符
  - 单词（字母、下划线、数字组成，不能数字开头）
- 宏体可以是任意的字符串
- 程序编译时，所有宏名出现的地方被替换为宏体字符串

  注意：所有作为单词出现的宏名

```
下面哪一个宏名会被替换？
printf("Mile2Meter = %f\n", Mile2Meter);
```

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏名必须是合法的 C 语言的标识符
  - 单词（字母、下划线、数字组成，不能数字开头）
- 宏体可以是任意的字符串
- 程序编译时，所有宏名出现的地方被替换为宏体字符串

  注意：所有作为单词出现的宏名

```
下面哪一个宏名会被替换？
printf("Mile2Meter = %f\n", Mile2Meter);
```

- 第一个宏名在字符串里，不是单词，不会被替换为宏体

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏定义语句后面不需要以分号（;）结尾

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏定义语句后面不需要以分号（;）结尾
- 如果有，也会被当作宏体的一部分，给使用宏带来错误

宏定义语句的语法格式

```
#define 宏名 宏体（宏定义字符串）
```

- 宏定义语句后面不需要以分号（;）结尾
- 如果有，也会被当作宏体的一部分，给使用宏带来错误
- 宏定义语句必须在一行内完成

内容提要

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

- 方便程序设计和维护

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

- 方便程序设计和维护

```
#define SDK_INFO "This is the name of this
    software development kit"
```

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

- 方便程序设计和维护

```
#define SDK_INFO "This is the name of this
    software development kit"
```

使用宏 SDK_INFO

```
printf(SDK_INFO); //输出SDK信息
```

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

- 方便程序设计和维护

```
#define SDK_INFO "This is the name of this
    software development kit"
```

使用宏 SDK_INFO

```
printf(SDK_INFO); //输出SDK信息
```

编译器首先进行宏替换（将 SDK_INFO 替换为宏体字符串），生成下面的代码

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

- 方便程序设计和维护

```
#define SDK_INFO "This is the name of this
    software development kit"
```

使用宏 SDK_INFO

```
printf(SDK_INFO); //输出SDK信息
```

编译器首先进行宏替换（将 SDK_INFO 替换为宏体字符串），生成下面的代码

```
printf("This is the name of this software
    development kit");
```

宏定义主要用途

- 定义符号常量，例如 PI，数组长度

```
#define PI 3.1415926
```

- 方便程序设计和维护

```
#define SDK_INFO "This is the name of this
    software development kit"
```

使用宏 SDK_INFO

```
printf(SDK_INFO); //输出SDK信息
```

编译器首先进行宏替换（将 SDK_INFO 替换为宏体字符串），生成下面的代码

```
printf("This is the name of this software
    development kit");
```

然后进行编译

宏定义主要用途

- 运算表达式，或语句组合

宏定义主要用途

- 运算表达式，或语句组合

定义一个面积计算的宏

```
#define AREA  PI * r * r
```

宏定义主要用途

- 运算表达式，或语句组合

定义一个面积计算的宏

```
#define AREA  PI * r * r
```

使用宏 AREA

```
float r = 3, a;
a = AREA;
```

宏定义主要用途

- 运算表达式，或语句组合

定义一个面积计算的宏

```
#define AREA   PI * r * r
```

使用宏 AREA

```
float r = 3, a;
a = AREA;
```

编译器首先进行宏替换（将 AREA 替换为宏体字符串），生成下面的代码

```
a = PI * r * r;
```

宏定义主要用途

- 运算表达式，或语句组合

定义一个面积计算的宏

```
#define AREA   PI * r * r
```

使用宏 AREA

```
float r = 3, a;
a = AREA;
```

编译器首先进行宏替换（将 AREA 替换为宏体字符串），生成下面的代码

```
a = PI * r * r;
```

然后进行编译

宏定义主要用途

定义一个变量交换宏

```
#define DoSWAP temp=a,a=b,b=temp
```

宏定义主要用途

定义一个变量交换宏

```
#define DoSWAP temp=a,a=b,b=temp
```

应用这个变量交换宏

```
DoSWAP;
```

宏定义主要用途

定义一个变量交换宏

```
#define DoSWAP temp=a,a=b,b=temp
```

应用这个变量交换宏

```
DoSWAP;
```

编译器首先进行宏替换（将 DoSWAP 替换为宏体字符串），生成下面的代码

```
temp=a,a=b,b=temp;
```

宏定义主要用途

定义一个变量交换宏

```
#define DoSWAP temp=a,a=b,b=temp
```

应用这个变量交换宏

```
DoSWAP;
```

编译器首先进行宏替换（将 DoSWAP 替换为宏体字符串），生成下面的代码

```
temp=a,a=b,b=temp;
```

然后进行编译。

宏定义主要用途

定义一个变量交换宏

```
#define DoSWAP temp=a,a=b,b=temp
```

应用这个变量交换宏

```
DoSWAP;
```

编译器首先进行宏替换（将 DoSWAP 替换为宏体字符串），生成下面的代码

```
temp=a,a=b,b=temp;
```

然后进行编译。

要想编译正确，必须事先定义了变量 float a, b, temp;

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b
- 不能用于交换其他变量

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b
- 不能用于交换其他变量
- 限制太大，太死

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b
- 不能用于交换其他变量
- 限制太大，太死

带参数的 SWAP 宏

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b
- 不能用于交换其他变量
- 限制太大，太死

带参数的 SWAP 宏

```
#define SWAP(x,y,t) t=x,x=y,y=t
...
float a,b,c;
...
SWAP(a,b,c);
```

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b
- 不能用于交换其他变量
- 限制太大，太死

带参数的 SWAP 宏

```
#define SWAP(x,y,t) t=x,x=y,y=t
...
float a,b,c;
...
SWAP(a,b,c);
```

把交换的对象以参数的形式告诉 SWAP

宏定义主要用途

DoSWAP 宏的缺点

```
#define DoSWAP temp=a,a=b,b=temp
```

- 必须以 temp 作为中间临时变量
- 只能适用于交换变量 a,b
- 不能用于交换其他变量
- 限制太大，太死

带参数的 SWAP 宏

```
#define SWAP(x,y,t) t=x,x=y,y=t
...
float a,b,c;
...
SWAP(a,b,c);
```

把交换的对象以参数的形式告诉 SWAP

增强了 SWAP 功能

内容提要

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

- 在普通的宏定义的基础上增加了参数列表

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

- 在普通的宏定义的基础上增加了参数列表

参数1,...,参数N

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

- 在普通的宏定义的基础上增加了参数列表

参数1,...,参数N

- 参数列表只需给出形式参数的名字

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

- 在普通的宏定义的基础上增加了参数列表

```
参数1,...,参数N
```

- 参数列表只需给出形式参数的名字
  - 不需要类型（大家想想为什么不需要类型名）

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

- 在普通的宏定义的基础上增加了参数列表

```
参数1,...,参数N
```

- 参数列表只需给出形式参数的名字
  - 不需要类型（大家想想为什么不需要类型名）

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

应用带参数宏

```
float a,b,c;
int u,v,w;
SWAP(a,b,c);
SWAP(u,v,w);
```

带参数的宏

```
#define 宏名(参数1,...,参数N) 宏体字符串
```

- 在普通的宏定义的基础上增加了参数列表

```
参数1,...,参数N
```

- 参数列表只需给出形式参数的名字
  - 不需要类型（大家想想为什么不需要类型名）

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

应用带参数宏

```
float a,b,c;
int u,v,w;
SWAP(a,b,c);
SWAP(u,v,w);
```

看起来很像函数调用

带参数的宏

理解参数宏的参数传递

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

带参数的宏

理解参数宏的参数传递

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

- x,y,t 都是宏 SWAP 的形式参数

带参数的宏

理解参数宏的参数传递

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

- x,y,t 都是宏 SWAP 的形式参数
- 使用宏时，需要提供实际参数，代替形式参数们，例如：

```
float a,b,c;
SWAP(a,b,c);
```

带参数的宏

理解参数宏的参数传递

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

- x,y,t 都是宏 SWAP 的形式参数
- 使用宏时，需要提供实际参数，代替形式参数们，例如：

```
float a,b,c;
SWAP(a,b,c);
```

- 编译器首先进行宏替换，生成下面的代码

```
c=a,a=b,b=c;
```

带参数的宏

理解参数宏的参数传递

```
#define SWAP(x,y,t) t=x,x=y,y=t
```

- x,y,t 都是宏 SWAP 的形式参数
- 使用宏时，需要提供实际参数，代替形式参数们，例如：

```
float a,b,c;
SWAP(a,b,c);
```

- 编译器首先进行宏替换，生成下面的代码

```
c=a,a=b,b=c;
```

- 宏定义的形式参数没有类型限制，所以使用时的实际参数可以是任意类型，只要替换后的语句合法（符合语法规则，能够编译）

带参数的宏

```c
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
int main(void)
{
    int x, y;
    scanf("%d%d", &x, &y);
    x = MAX(x,y);
    y = SQR(x);
    printf("x = %d\ny = %d\n", x, y);
    return 0;
}
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
int main(void)
{
    int x, y;
    scanf("%d%d", &x, &y);
    x = MAX(x,y); 宏展开==> x = x>y? x : y;
    y = SQR(x);   宏展开==> y = x*x;
    printf("x = %d\ny = %d\n", x, y);
    return 0;
}
```

带参数的宏

```c
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
int main(void)
{
    int x, y;
    scanf("%d%d", &x, &y);
    x = MAX(x,y); 宏展开==> x = x>y? x : y;
    y = SQR(x);   宏展开==> y = x*x;
    printf("x = %d\ny = %d\n", x, y);
    return 0;
}
```

运行结果

```
3 6
x = 6
y = 36
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

printf("%d", SQR(3+5)); 输出结果是什么？

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

`printf("%d", SQR(3+5));` 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

`printf("%d", SQR(3+5));` 输出结果是什么？

- **我们期望的正确结果应该是** 8*8=64。

但是，实验输出：23
因为宏展开是：`printf("%d", 3+5*3+5);`

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

`printf("%d", SQR(3+5));` 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：`printf("%d", 3+5*3+5);`

- 错误的根源是运算优先级导致的，需要添加括号，先做加法

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

`printf("%d", SQR(3+5));` 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：`printf("%d", 3+5*3+5);`

- 错误的根源是**运算优先级**导致的，需要添加括号，先做加法

```
printf("%d", SQR((3+5)));
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

printf("%d", SQR(3+5)); 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：printf("%d", 3+5*3+5);

- 错误的根源是运算优先级导致的，需要添加括号，先做加法

printf("%d", SQR((3+5)));

- 宏的实际参数改为：(3+5)

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

printf("%d", SQR(3+5)); 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：printf("%d", 3+5*3+5);

- 错误的根源是运算优先级导致的，需要添加括号，先做加法

printf("%d", SQR((3+5)));

- 宏的实际参数改为：(3+5)
- SQR((3+5)) 展开为：(3+5)*(3+5)

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

`printf("%d", SQR(3+5));` 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：`printf("%d", 3+5*3+5);`

- 错误的根源是运算优先级导致的，需要添加括号，先做加法

`printf("%d", SQR((3+5)));`

- 宏的实际参数改为：(3+5)
- SQR((3+5)) 展开为：(3+5)*(3+5)
- 现在运算结果正确.

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

printf("%d", SQR(3+5)); 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：printf("%d", 3+5*3+5);

- 错误的根源是运算优先级导致的，需要添加括号，先做加法

printf("%d", SQR((3+5)));

- 宏的实际参数改为：(3+5)
- SQR((3+5)) 展开为：(3+5)*(3+5)
- 现在运算结果正确. 但每次都要添加括号，很麻烦，容易忘

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

printf("%d", SQR(3+5)); 输出结果是什么？

- 我们期望的正确结果应该是 8*8=64。

但是，实验输出：23
因为宏展开是：printf("%d", 3+5*3+5);

- 错误的根源是运算优先级导致的，需要添加括号，先做加法

printf("%d", SQR((3+5)));

- 宏的实际参数改为：(3+5)
- SQR((3+5)) 展开为：(3+5)*(3+5)
- 现在运算结果正确. 但每次都要添加括号，很麻烦，容易忘
- 有更好的方法吗？

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是:

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是：

  在宏定义中添加括号，强制将参数计算提升为最高优先级

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是：

  在宏定义中添加括号，强制将参数计算提升为最高优先级

```
#define MAX(a,b) ((a)>(b) ? (a) : (b))
#define SQR(x)   ((x)*(x))
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是：

  在宏定义中添加括号，强制将参数计算提升为最高优先级

```
#define MAX(a,b) ((a)>(b) ? (a) : (b))
#define SQR(x)   ((x)*(x))
```

- 注意：我们还对宏定义字符串加了括号

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是：
  在宏定义中添加括号，强制将参数计算提升为最高优先级

```
#define MAX(a,b) ((a)>(b) ? (a) : (b))
#define SQR(x)   ((x)*(x))
```

- 注意：我们还对宏定义字符串加了括号
  目的是：还是改变优先级。

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是：
  在宏定义中添加括号，强制将参数计算提升为最高优先级

```
#define MAX(a,b) ((a)>(b) ? (a) : (b))
#define SQR(x)   ((x)*(x))
```

- 注意：我们还对宏定义字符串加了括号
  目的是：还是改变优先级。 看下面的例子

```
int x = 3, y = 4;
int z = MAX(x,y)*MAX(x,y);
```

带参数的宏

```
#define MAX(a,b) a>b ? a : b
#define SQR(x)   x*x
```

- 更好的方法是：
  在宏定义中添加括号，强制将参数计算提升为最高优先级

```
#define MAX(a,b) ((a)>(b) ? (a) : (b))
#define SQR(x)   ((x)*(x))
```

- 注意：我们还对宏定义字符串加了括号
  目的是：还是改变优先级。 看下面的例子

```
int x = 3, y = 4;
int z = MAX(x,y)*MAX(x,y);
```

- 如果不加括号，展开的是：

```
int z = x>y ? x : y*x>y ? x : y; //先计算y*x
```

总结

- 宏定义的语法
- 宏定义替换
- 带参数的宏定义
- 宏定义的运算

今天到此为止