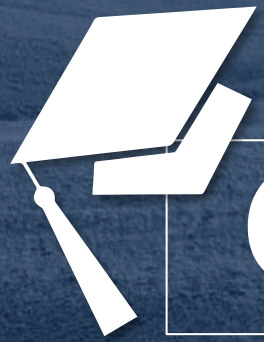
章节复习 | 引 论 | 函 数 | 变 量 | 宏定义 | 思维导图



C语言程序设计第四章

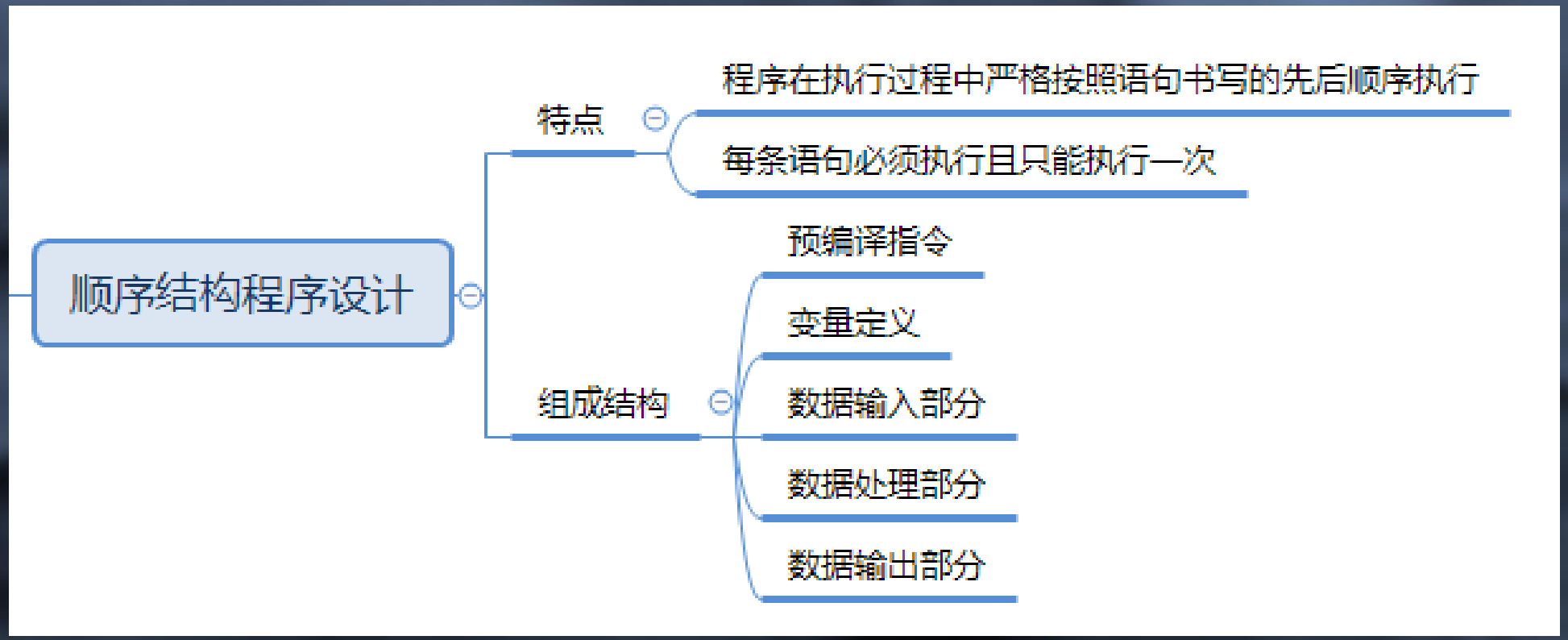
——函数与预处理

主讲人： 李伦彬

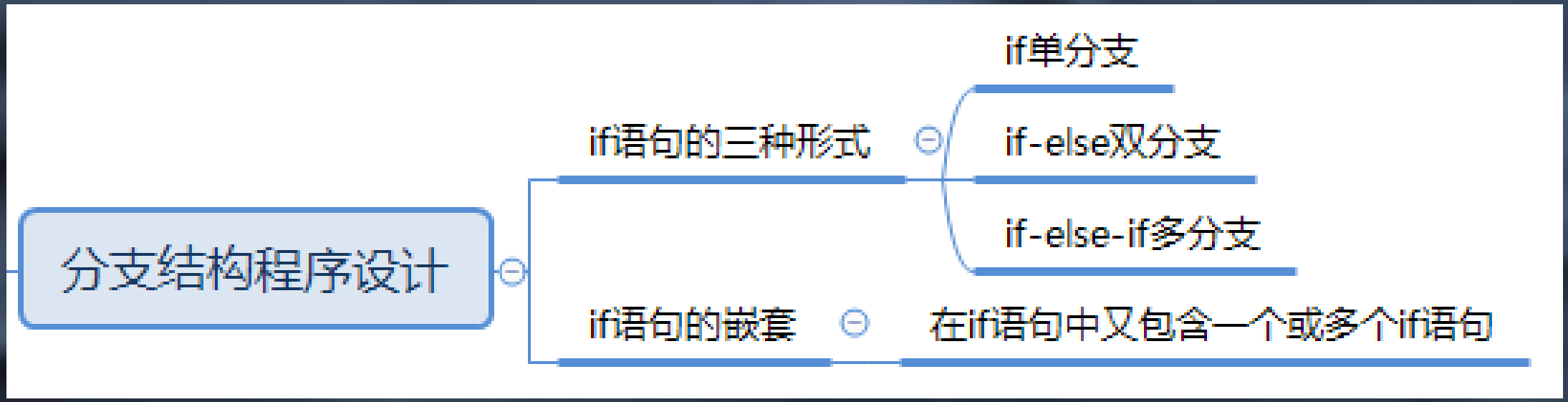
CONTENTS

章节复习

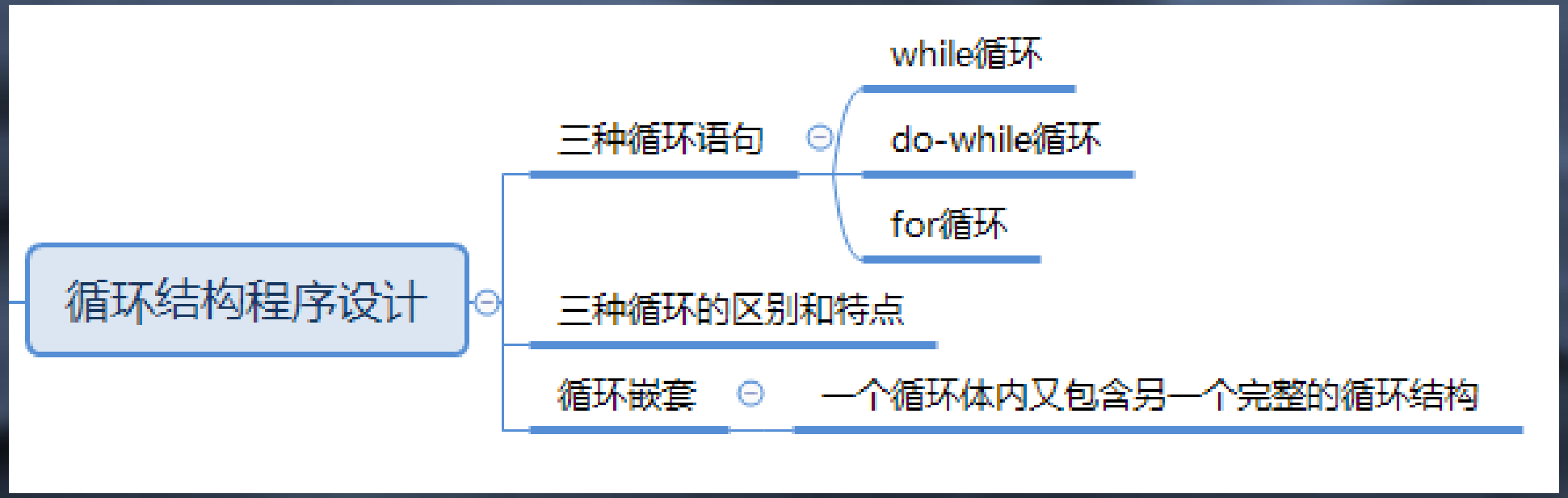
顺序结构程序设计部分思维导图

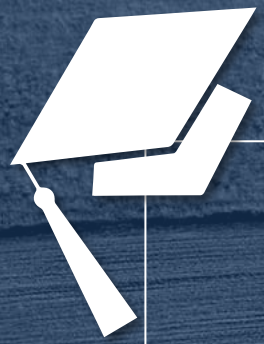


分支结构程序设计部分思维导图



循环结构程序设计部分思维导图





引 论

本节内容

程序的模块化设计思想

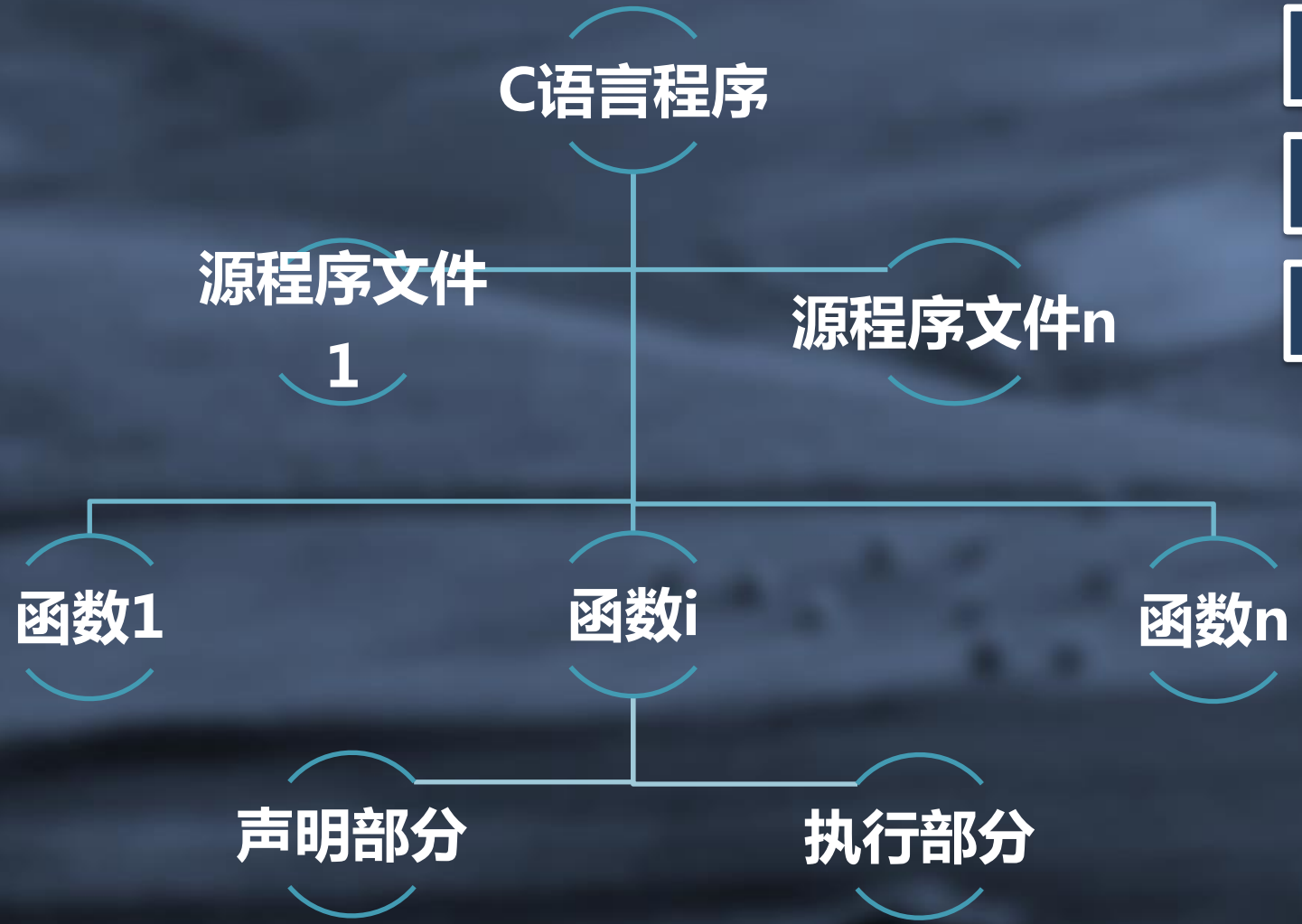
C语言程序结构

程序的模块化设计思想

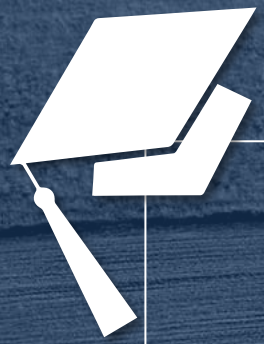
一辆汽车可分为若干个部件，由若干厂家分头生产，因此汽车厂商只是设计、组装汽车。

C语言是模块化程序设计语言

1. 模块化
2. 自顶向下
3. 逐步求精



C语言程序结构



函 数

本节内容

函数定义的一般形式

函数的参数和返回值

函数的调用

练习巩固

重难点突破

重点内容：

1. 函数的一般定义形式
2. 不同情况下返回值的处理
3. 函数的各种调用形式

难点问题：

1. 实参和形参的区别
2. 不同返回值类型函数对应返回方式的区别

函数的一般形式

```
类型标识符 函数名（arg1,arg2,...）
{
    声明部分
    语句部分
}
```

类型标识符：说明函数返回值类型

函数名：函数通过函数名实现调用

参数列表：为函数体传递参数

花括号：标志函数的开始和结束

函数体：实现函数的功能

函数按参数列表中有无参数可分为有参函数和无参函数；按函数体是否为空，可将函数分为空函数和非空函数。

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

项目管理 查看类 调试

Head01.h

```
/**
 * Author:Lee
 * Time:2018/08/16 8:31
 * Content:函数的三种类型示例
 * 1.无参函数
 * 2.有参函数
 * 3.空函数
 *
 * */

/**
 * 无参函数示例
 * 打印输出欢迎文字
 *
 * */
void print( ) {
      printf("welcome to china! \n");
}

/**
 * 有参函数示例
 * 返回两个整数中较大一个整数，若两者相等，则返回前者
 * param x 待比较的整数
 * param y 待比较的整数
```

编译器 资源 编译日志 调试 搜索结果

行: 32 列: 1 已选择: 0 总行数: 32 长度: 64 插入 在 0.015 秒内完成解析

函数参数和返回值

1 **形式参数**：定义函数时函数名后面括号中的变量名称为“形式参数”（简称“形参”）

2 **实际参数**：主调函数中调用一个函数时，函数名后面括号中的参数(可以是一个表达式)称为“实际参数”（简称“实参”）。

3 **函数返回值**：return后面的括号中的值作为函数带回的值（称函数返回值）。

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

项目管理 查看类 调试

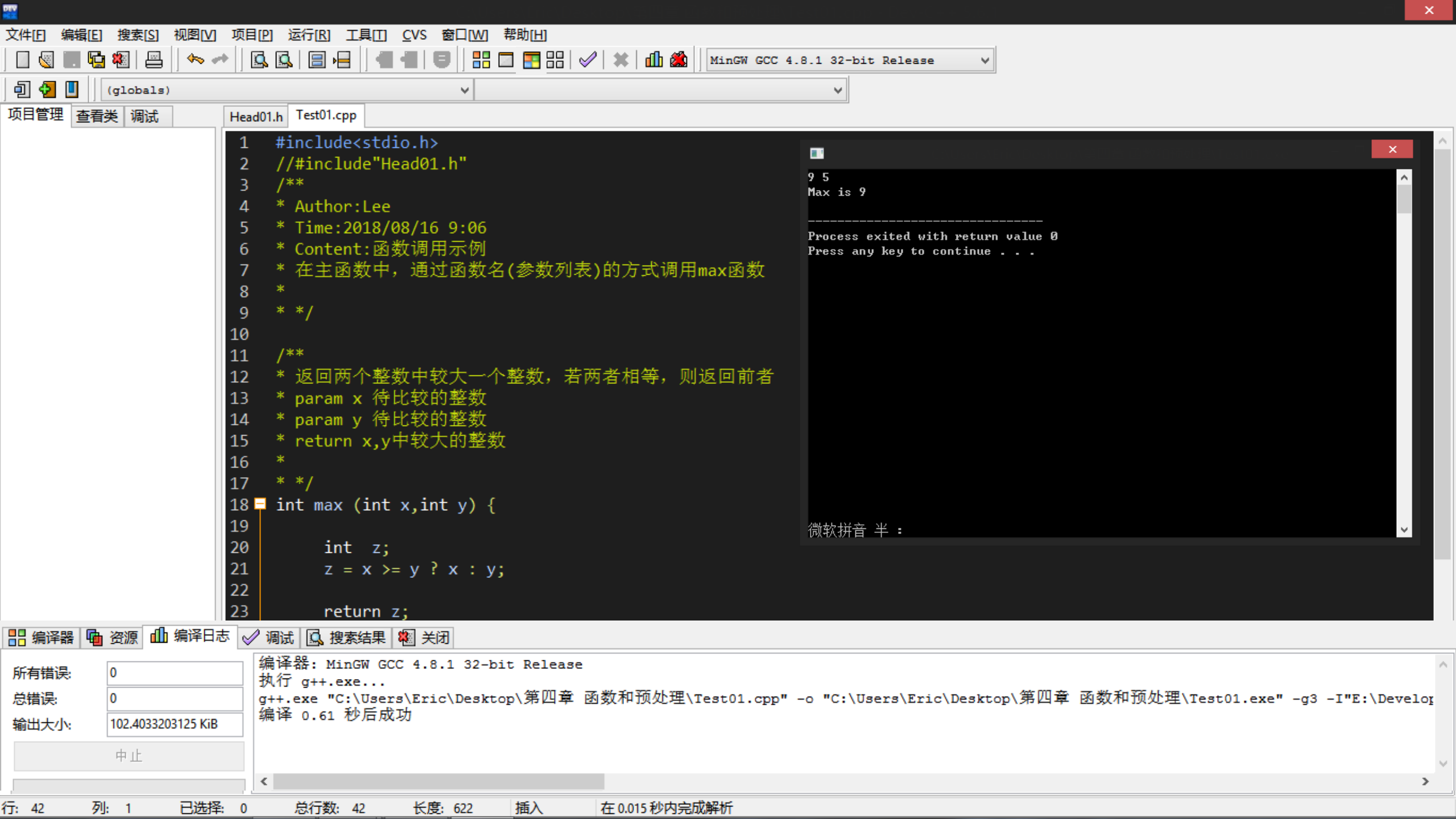
Head01.h Test01.cpp

```
1  #include<stdio.h>
2  //#include"Head01.h"
3  /**
4  * Author:Lee
5  * Time:2018/08/16 9:06
6  * Content:函数调用示例
7  * 在主函数中，通过函数名(参数列表)的方式调用max函数
8  *
9  * */
10
11 /**
12 * 返回两个整数中较大一个整数，若两者相等，则返回前者
13 * param x 待比较的整数
14 * param y 待比较的整数
15 * return x,y中较大的整数
16 *
17 * */
18 int max (int x,int y) {
19
20       int  z;
21     z = x >= y ? x : y;
22
23     return z;
24 }//max(int,int) end
```

微课视频

编译器 资源 编译日志 调试 搜索结果

行: 37 列: 2 已选择: 0 总行数: 39 长度: 613 插入 在 0.015 秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Head01.h Test01.cpp
1 #include<stdio.h>
2 //#include"Head01.h"
3 /**
4 * Author:Lee
5 * Time:2018/08/16 9:06
6 * Content:函数调用示例
7 * 在主函数中，通过函数名(参数列表)的方式调用max函数
8 *
9 * */
10
11 /**
12 * 返回两个整数中较大一个整数，若两者相等，则返回前者
13 * param x 待比较的整数
14 * param y 待比较的整数
15 * return x,y中较大的整数
16 *
17 * */
18 int max (int x,int y) {
19
20 int z;
21 z = x >= y ? x : y;
22
23 return z;
9 5
Max is 9
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 :
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 102.4033203125 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\Test01.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\Test01.exe" -g3 -I"E:\Develop
编译 0.61 秒后成功
行: 42 列: 1 已选择: 0 总行数: 42 长度: 622 插入 在 0.015 秒内完成解析

函数实参和形参的特点

- 函数定义时必须说明形参的类型，形参只能是变量（含指针变量）或数组
- 实参可以是常量、变量或表达式，必须有确定值
- 函数被调用期间形参占用内存，调用结束后，形参所占用的内存将被回收
- 实参与形参的类型、顺序、个数一致
- 实参向形参的数据传递是“值传递”，即单向传递（实参→形参）

函数返回值


Void型函数
非Void型函数
无return
return;
return 表达式;
return(表达式);
0 error,0 warning
一般用于强制退出该函数
产生警告，回传一个不确定值
一般用于强制退出该函数
0 error,0 warning
推荐使用，便于扩展维护

```
#include <stdio.h>
/**
 * Author:Lee
 * Time:2018/08/18
 * Content:两个数最大值函数
 * Describe:利用两个数最大数函数来讲解非void型函数中返回值的使用
 *
 * */

/**
 * return (表达式); 型
 * param x 整数1
 * param y 整数2
 * return 两个数中较大者
 *
 * */
int max1( int x, int y) {

      int z;
      z = (x > y) ? x:y;
      return ( z );
}

/**
```

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

项目管理 查看类 调试

Chapter04_Example02.cpp

```
1  #include <stdio.h>
2  /**
3   * Author:Lee
4   * Time:2018/08/18
5   * Content:两个数最大值函数
6   * Describe:利用两个数最大数函数来讲解非void型函数中返回
7   *
8   * */
9
10 /**
11  * return (表达式); 型
12  * param x 整数1
13  * param y 整数2
14  * return 两个数中较大者
15  *
16  * */
17 int max1( int x, int y) {
18
19     int z;
20     z = (x > y) ? x:y;
21     return ( z );
22 }
23
```

```
max1: 5
max2: 5
max3: 5
max4: 5

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

微软拼音 半 :

编译器 资源 编译日志 调试 搜索结果 关闭

所有错误: 0

总错误: 0

输出大小: 103.341796875 KiB

中止

```
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example02.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapte
编译 4.83 秒后成功
```

行: 84 列: 1 已选择: 0 总行数: 102 长度: 1440 插入 在 0.016 秒内完成解析

函数的调用

- 函数调用的一般形式为：函数名（实参表列）
- 如果是调用无参函数，则“实参表列”可以没有，但括弧不能省略。
- 如果实参表列包含多个实参，则各参数间用逗号隔开
- 实参与形参的个数应相等，类型应匹配。实参与形参按顺序对应，一一传递数据。
- 如果实参表列包括多个实参，对实参求值的顺序并不是确定的（不一定从右至左）

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

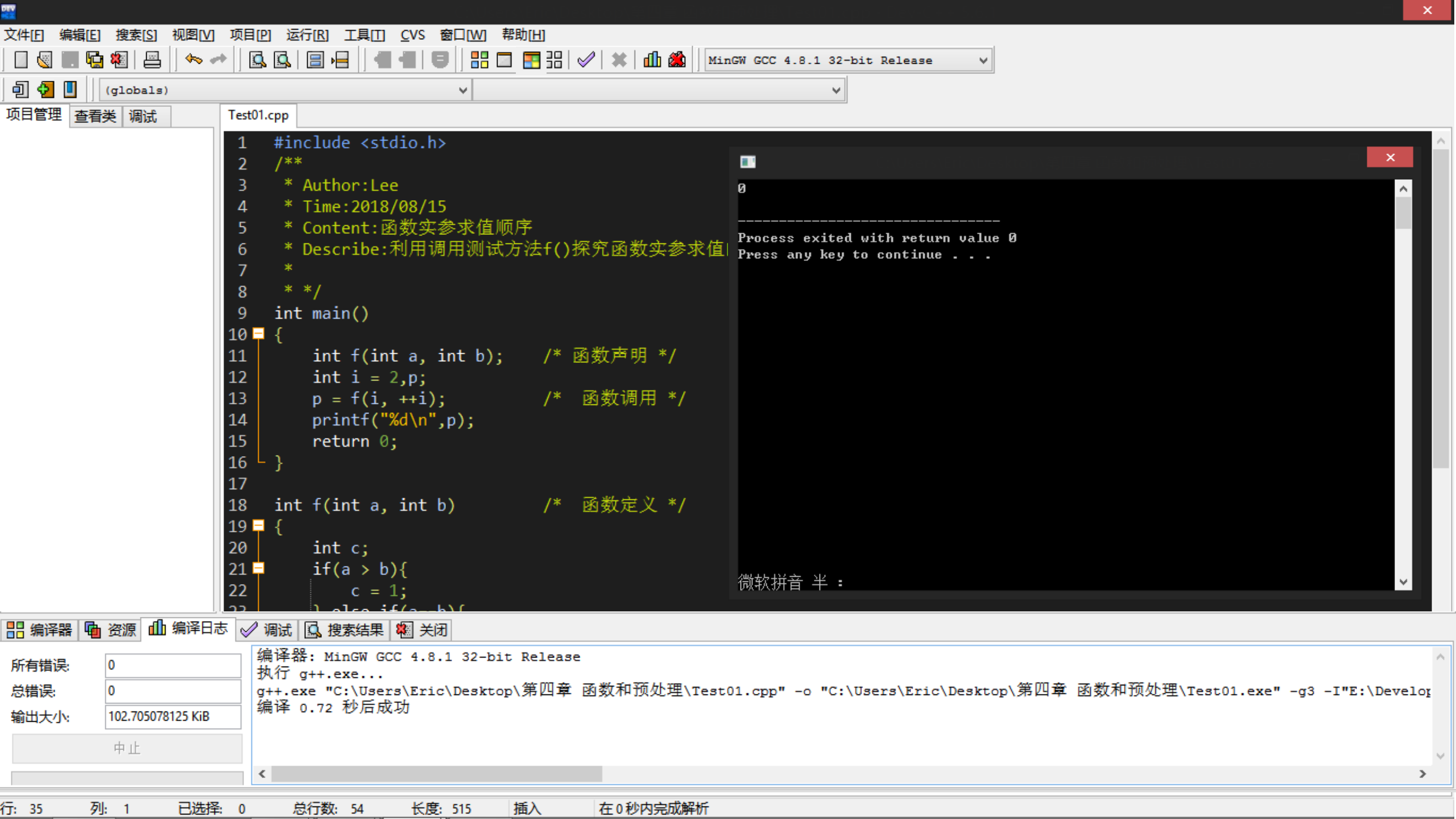
项目管理 查看类 调试

Test01.cpp

```
#include <stdio.h>
/**
 * Author:Lee
 * Time:2018/08/15
 * Content:函数实参求值顺序
 * Describe:利用调用测试方法f()探究函数实参求值的先后顺序
 *
 * */
int main()
{
	int f(int a, int b);		/* 函数声明 */
	int i = 2,p;
	p = f(i, ++i);			/*  函数调用 */
	printf("%d\n",p);
	return 0;
}

int f(int a, int b)		/*  函数定义 */
{
	int c;
	if(a > b){
		c = 1;
	} else if(a==b){
		c = 0;
```

编译器 资源 编译日志 调试 搜索结果

行: 25 列: 1 已选择: 0 总行数: 25 长度: 50 插入 在0秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Test01.cpp
#include <stdio.h>
/**
* Author:Lee
* Time:2018/08/15
* Content:函数实参求值顺序
* Describe:利用调用测试方法f()探究函数实参求值
*
* */
int main()
{
int f(int a, int b); /* 函数声明 */
int i = 2,p;
p = f(i, ++i); /* 函数调用 */
printf("%d\n",p);
return 0;
}
int f(int a, int b) /* 函数定义 */
{
int c;
if(a > b){
c = 1;
0
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 :
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 102.705078125 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\Test01.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\Test01.exe" -g3 -I"E:\Develop
编译 0.72 秒后成功
行: 35 列: 1 已选择: 0 总行数: 54 长度: 515 插入 在 0 秒内完成解析

函数调用的方式

1. **函数语句**：把函数调用作为一个语句。这时不要求函数带回值，只要求函数完成一定的操作。

2. **函数表达式**：函数出现在一个表达式中，这种表达式称为函数表达式。

   例如：c = 2 * max(a, b);

3. **函数参数**：函数调用作为一个函数的实参。

   例如：m = max (a, max(b, c));

练习巩固

- Proproblem description

  利用函数调用编写一个简单的加法计算小程序

- Input

  输入占一行。输入两个实数，中间以英文逗号分隔

- Output

  输出占一行。输出一个小数，表示两个数之和。小数点后保留六位

C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_homework.cpp - [Executing] - Dev-C++ 5.6.1

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

项目管理 查看类 调试

Chapter04_homework.cpp

```
1  #include <stdio.h>
2  /**
3   * Author:Lee
4   * Time:2018/08/18
5   * Content:练习巩固
6   * Describe:编写一个简单的加法函数
7   *
8   * */
9  # include <stdio.h>
10 int main() {
11     /*对被调用函数add的声明*/
12     float  add(float x, float y);
13     float  a,b,c;
14
15     scanf("%f,%f",&a,&b);
16     c=add(a,b);
17     printf("sum is %f \n",c);
18
19     return 0;
20 }
21
22 float add(float x,float y) {        /*函数首部*/
23     float z;                        /*  函数体 */
24     z=x+y;
25     return(z);
26 }
27
28
29
```

```
5,9
sum is 14.000000

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

编译器 资源 编译日志 调试 搜索结果

行: 32 列: 1 已选择: 0 总行数: 33 长度: 508 插入 在 0.016 秒内完成解析

本小节内容

函数嵌套调用的含义

函数嵌套调用的过程

函数嵌套调用举例

练习巩固

重难点突破

重点内容：

1. 函数嵌套调用的概念
2. 函数嵌套调用的两种典型过程

难点问题：

1. 嵌套调用和嵌套定义的区别
2. 学会使用函数嵌套调用解决实际问题

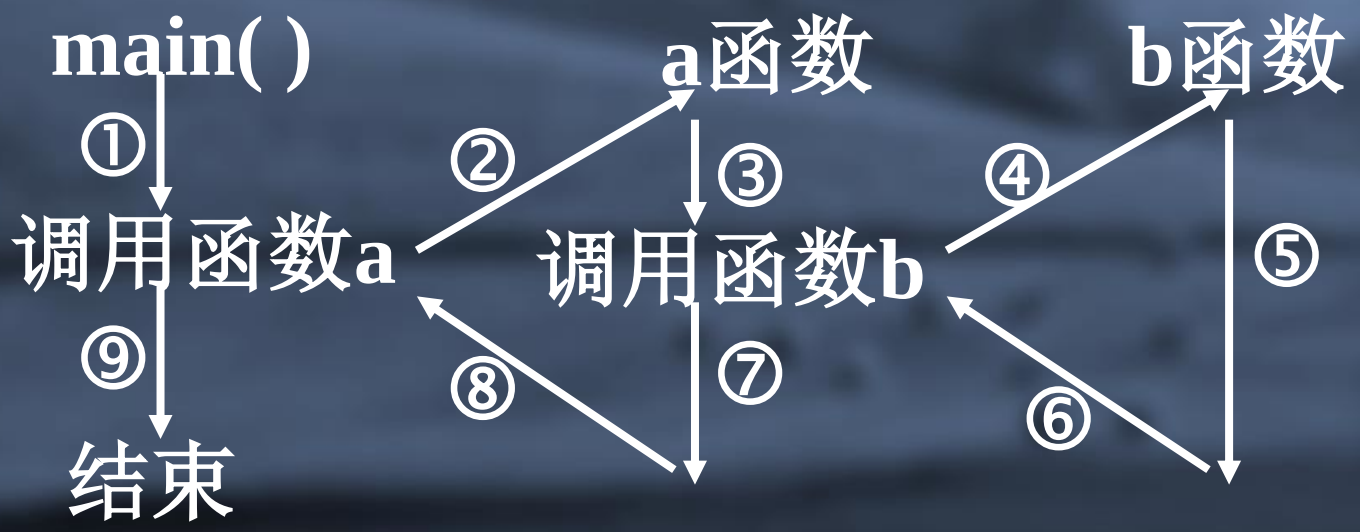
函数嵌套调用含义：

在调用一个函数的过程中，又调用另一个函数。

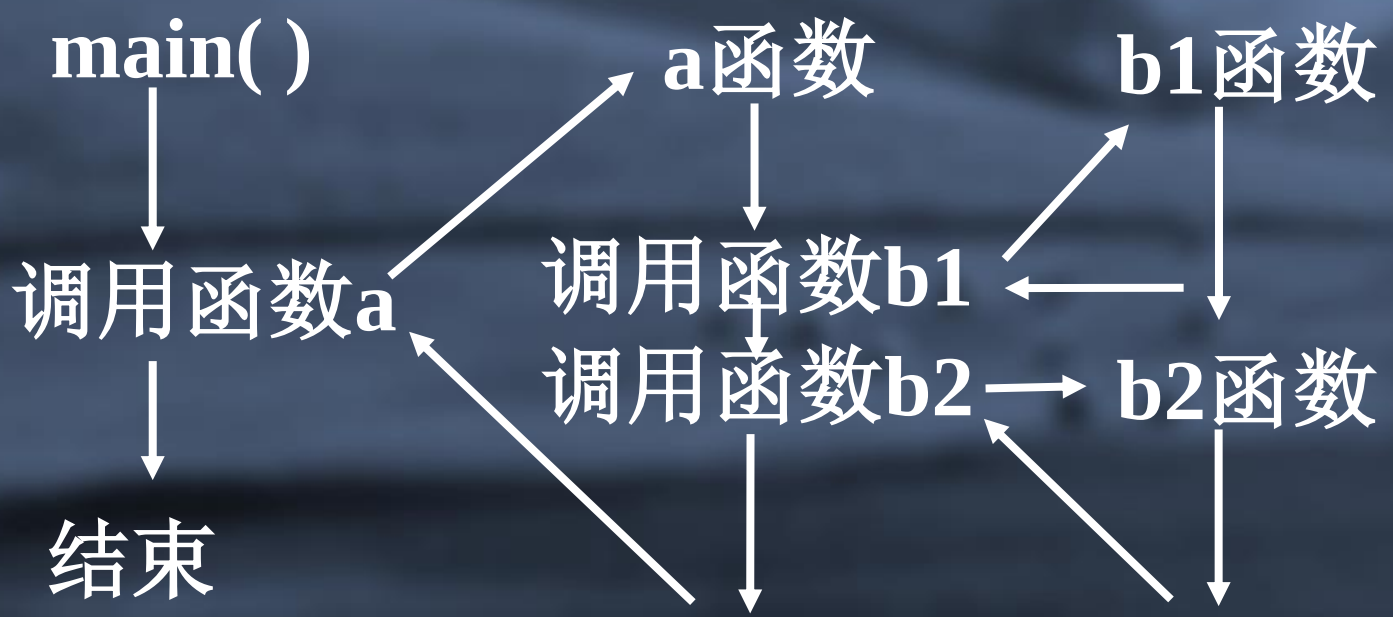
- 允许嵌套调用
  在调用某函数过程中又调用另一函数

- 不允许嵌套定义
  函数间的关系是平行的、独立的

函数嵌套调用过程1



函数嵌套调用过程2



文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

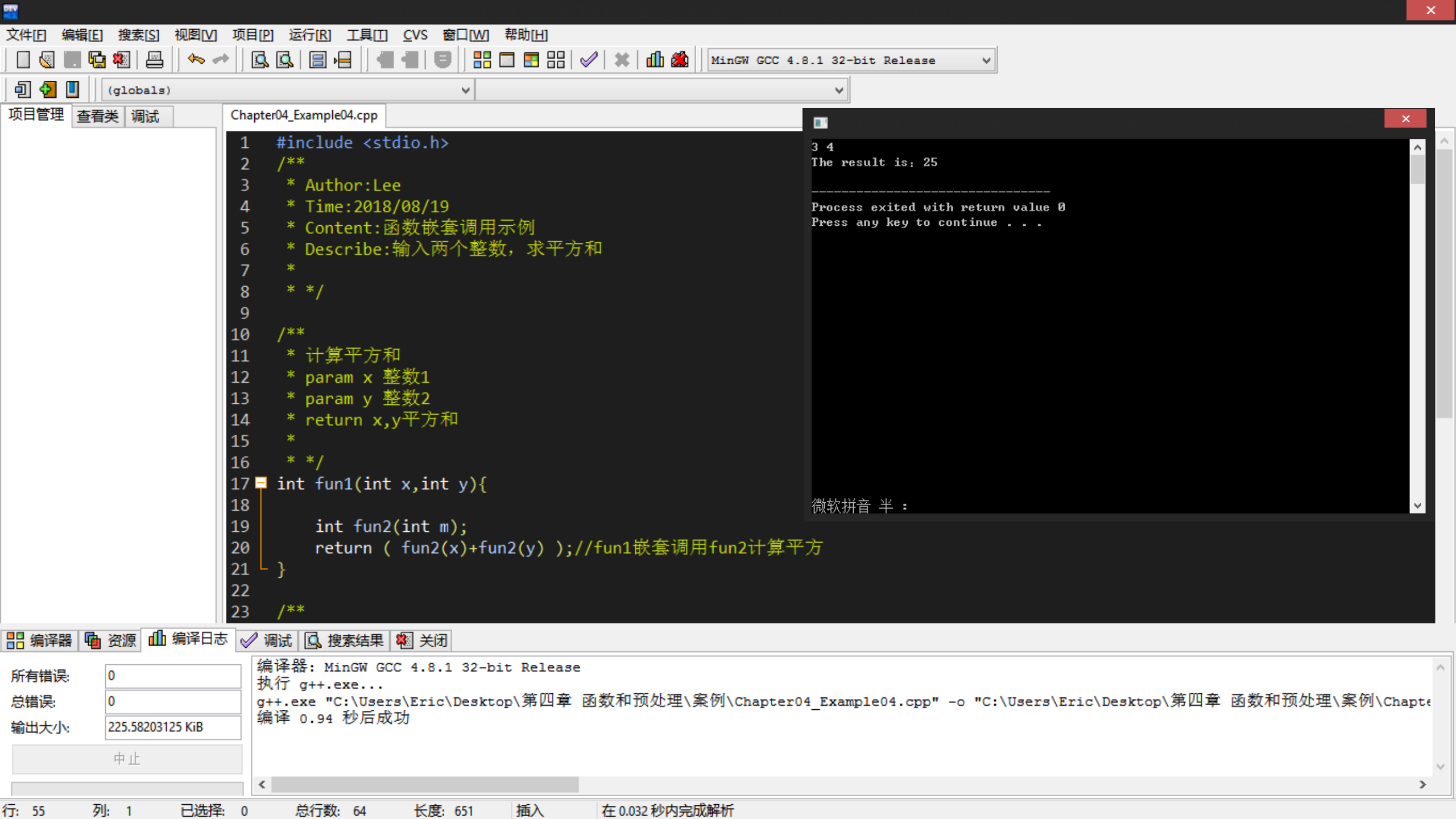
项目管理 查看类 调试

Chapter04_Example04.cpp

```
#include <stdio.h>
/**
 * Author:Lee
 * Time:2018/08/19
 * Content:函数嵌套调用示例
 * Describe:输入两个整数，求平方和
 *
 * */

/**
 * 计算平方和
 * param x 整数1
 * param y 整数2
 * return x,y平方和
 *
 * */
int fun1(int x,int y){

      int fun2(int m);
      return ( fun2(x)+fun2(y) );//fun1嵌套调用fun2计算平方
}

/**
 * 计算平方
```

编译器 (1) 资源 编译日志 调试 搜索结果

行: 55 列: 1 已选择: 0 总行数: 64 长度: 651 插入 在 0.032 秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Chapter04_Example04.cpp
1 #include <stdio.h>
2 /**
3 * Author:Lee
4 * Time:2018/08/19
5 * Content:函数嵌套调用示例
6 * Describe:输入两个整数，求平方和
7 *
8 * */
9
10 /**
11 * 计算平方和
12 * param x 整数1
13 * param y 整数2
14 * return x,y平方和
15 *
16 * */
17 int fun1(int x,int y){
18
19 int fun2(int m);
20 return (fun2(x)+fun2(y));//fun1嵌套调用fun2计算平方
21 }
22
23 /**
3 4
The result is: 25
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 ：
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 225.58203125 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example04.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapte
编译 0.94 秒后成功
行: 55 列: 1 已选择: 0 总行数: 64 长度: 651 插入 在 0.032 秒内完成解析

练习巩固

- Proproblem description

编写一个判断素数的函数，实现输入一个整数，使用判断素数的函数进行判断，然后输出是否是素数的信息。

- Input

输入占一行。即待判断的整数n

- Output

输出占一行。若n为素数，输出yes，否则输出no

C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example05.cpp - [Executing] - Dev-C++ 5.6.1

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

捕获滚动窗口 <Ctrl+Alt+PrtSc>

项目管理 查看类 调试

Chapter04_Example05.cpp

```
1   #include <stdio.h>
2   /**
3    * Author:Lee
4    * Time:2018/08/19
5    * Content:练习巩固
6    * Describe:判断素数
7    *
8    * */
9   int isPrime(int num){
10      for(int i = 2; i * i <= num; i++) {
11          if(num % i == 0){
12              return 0;
13          }
14      }
15      return 1;
16  }
17  int main() {
18
19      int n;
20      //增加输入输出提示可以提升用户体验
21      //但在竞赛中，不符合题干要求的输入输出提示反而会造成扣分
22      //printf("Please Enter a Integer") ;
23      scanf("%d",&n);
24      if(isPrime(n)) {
25          printf("yes\n");
26      } else {
27          printf("no\n");
28      }
29      return 0;
30  }
31
```

```
9
no

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

```
11
yes

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

微软拼音 半 ：

编译器 资源 编译日志 调试 搜索结果

行: 33 列: 1 已选择: 0 总行数: 56 长度: 558 插入 在 0 秒内完成解析

本小节内容

递归的定义与调用方式

递归调用的关键

递归与循环的比较

程序举例及练习巩固

重难点突破

重点内容：

1. 递归的定义方式
2. 递归的关键要素
3. 递归和循环的区别

难点问题：

1. 如何确定递归出口和递归方程式
2. 学会使用递归解决实际问题

递归的定义：

函数调用过程中，直接或间接的调用自身，称为函数的递归。

- 直接递归调用
  在函数体中，直接调用自身

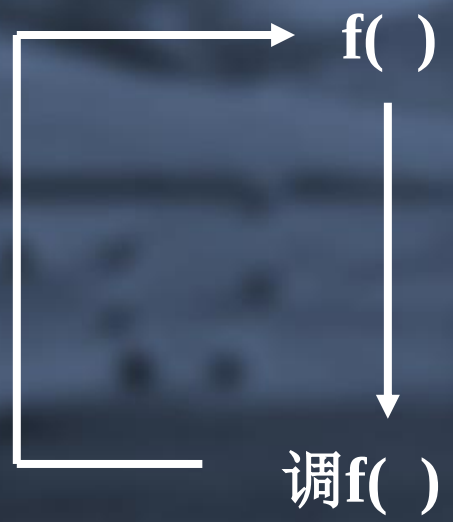
- 间接递归调用
  在函数体中，调用其它函数，在其他函数中又反过来调用自身

- 直接递归调用：在函数体内又调用自身

主调函数

被调函数

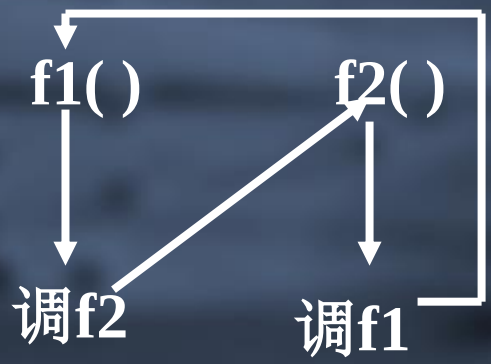
```
int  f(int x){
     int y,z;
     ……
     z=f (y);
     ……
     return(2*z);
}
```



- 间接递归调用：当函数1去调用另一函数2时，而另一函数2反过来又调用函数1。

```
int  f1(int x)
{    int y,z;
      ……
     z=f2(y);
     ……..
     return(2*z);
}
```

```
int  f2(int t)
{    int a,c;
      ……
     c=f1(a);
     ……..
     return(3+c);
}
```



函数递归调用的关键：

- 递归关系（递归方程式）
  递归运行的规律，即如何进行递归

- 递归边界（递归终止条件/递归出口）
  递归结束的条件，即如何终止递归

递归与循环的比较

A

效 率

递归和循环都可以实现函数。但循环的时间效率和空间效率优于递归

B

需求场景

当问题需要“后进先出”操作时，用递归更有效。

C

可读性和可维护性

递归使程序更简洁，但是使程序难于阅读和维护。

递归的一般形式

```
递归函数名f(参数n){
if (n==初值)
        结果=----
Else
        结果=含f(n-1)的表达式
Return (返回结果);
}
```

递归出口：递归的结束条件

递归方程式：递归进行的规则，进一步划分问题。

返回调用：返回上一层调用，合并问题求解。

递归函数，通过调用函数自身来简化原问题

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

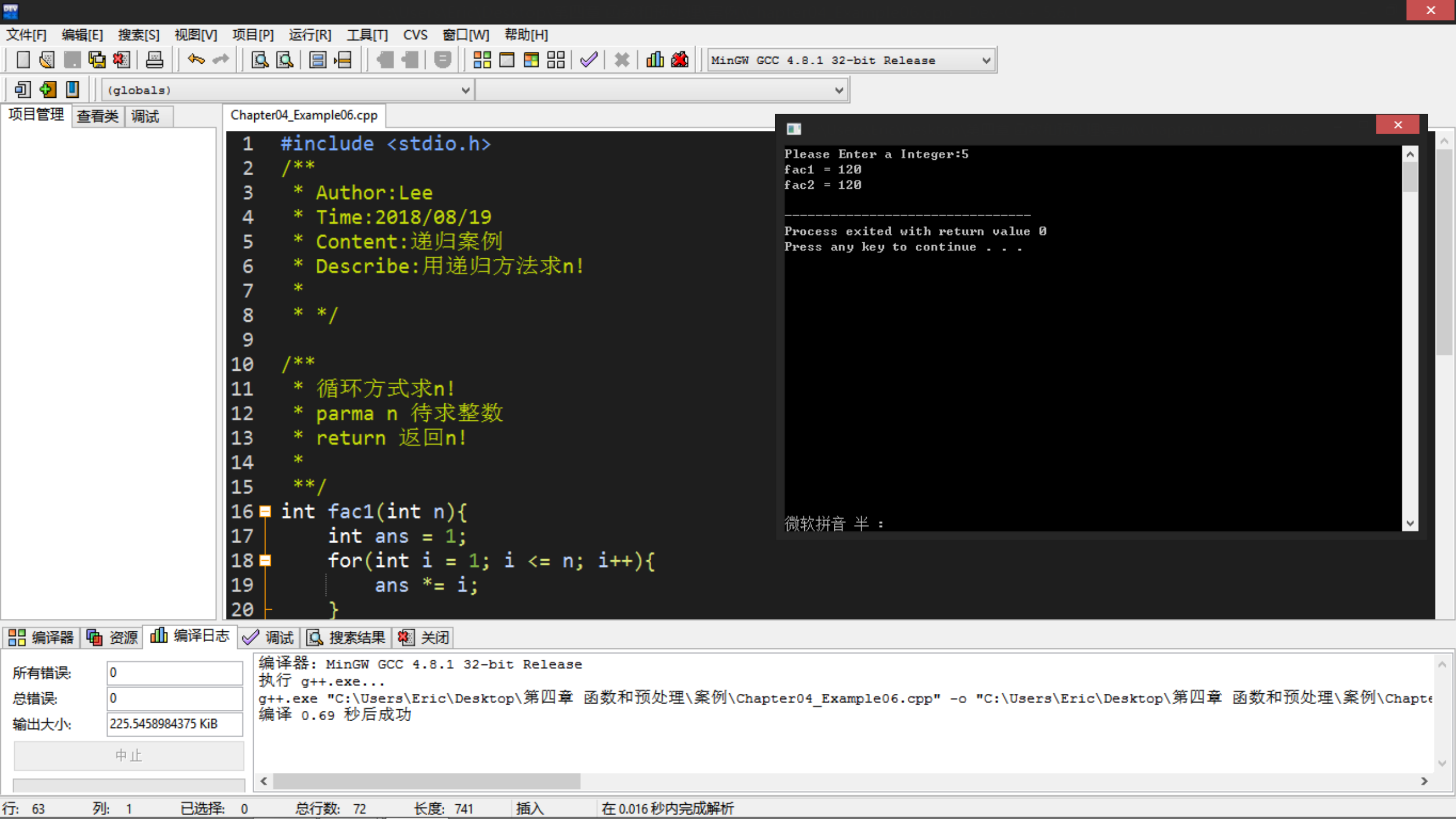
项目管理 查看类 调试

Chapter04_Example06.cpp

```
#include <stdio.h>
/**
 * Author:Lee
 * Time:2018/08/19
 * Content:递归案例
 * Describe:用递归方法求n!
 *
 * */

/**
 * 循环方式求n!
 * parma n 待求整数
 * return 返回n!
 *
 **/
int fac1(int n){
      int ans = 1;
      for(int i = 1; i <= n; i++){
            ans *= i;
      }
      return ans;
}

/**
```

编译器 资源 编译日志 调试 搜索结果

行: 63 列: 1 已选择: 0 总行数: 72 长度: 741 插入 在 0.016 秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Chapter04_Example06.cpp
1 #include <stdio.h>
2 /**
3 * Author:Lee
4 * Time:2018/08/19
5 * Content:递归案例
6 * Describe:用递归方法求n!
7 *
8 * */
9
10 /**
11 * 循环方式求n!
12 * parma n 待求整数
13 * return 返回n!
14 *
15 **/
16 int fac1(int n){
17 int ans = 1;
18 for(int i = 1; i <= n; i++){
19 ans *= i;
20 }
Please Enter a Integer:5
fac1 = 120
fac2 = 120
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 :
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 225.5458984375 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example06.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapte
编译 0.69 秒后成功
行: 63 列: 1 已选择: 0 总行数: 72 长度: 741 插入 在 0.016 秒内完成解析

练习巩固

- Proproblem description
  求Fibonacci斐波那契数列，求前n项的值
- Input
  输入占一行。即项数n
- Output
  输出占一行，包含n个元素。即为斐波那契数列前n项的数字。

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

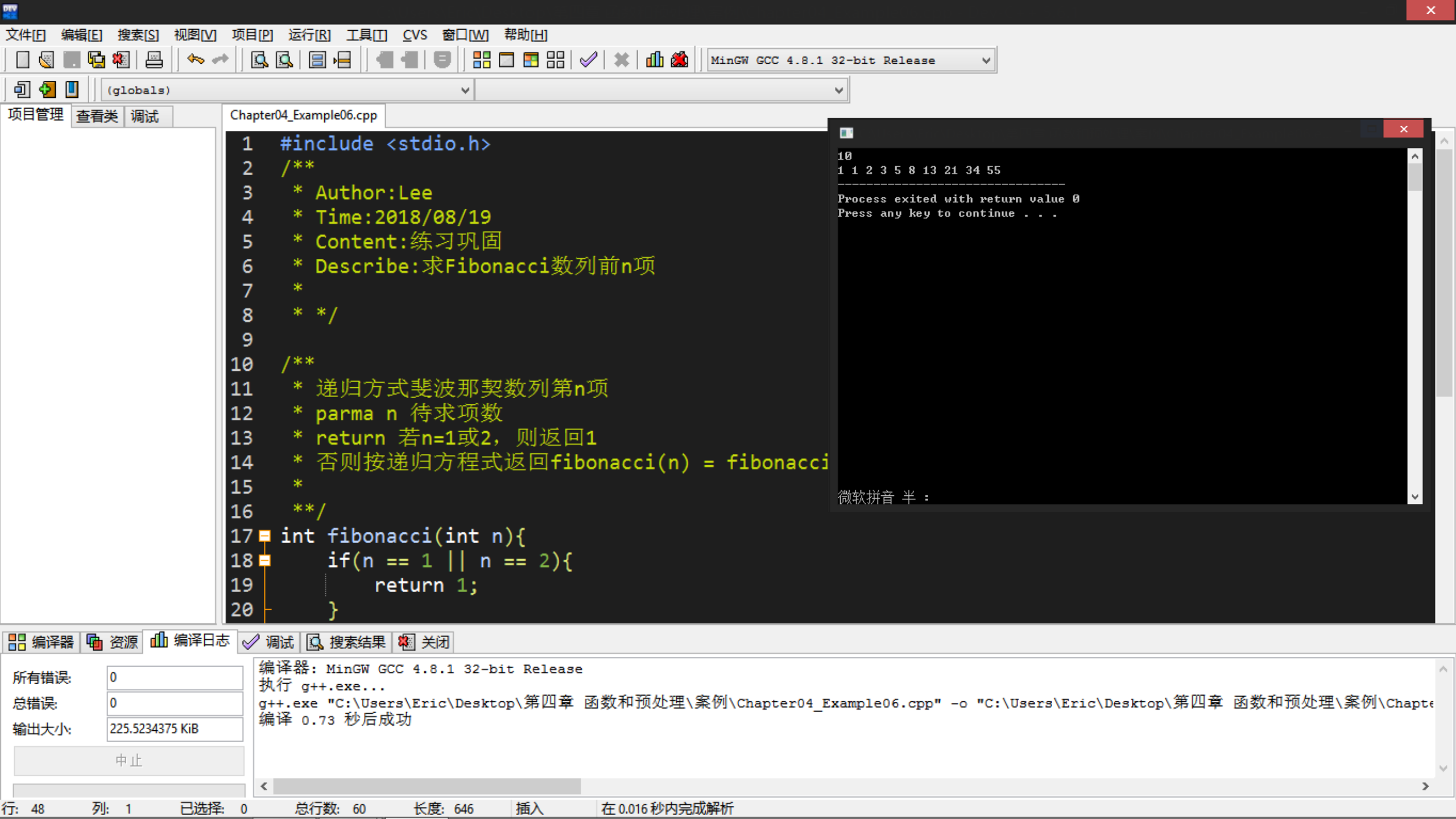
项目管理 查看类 调试

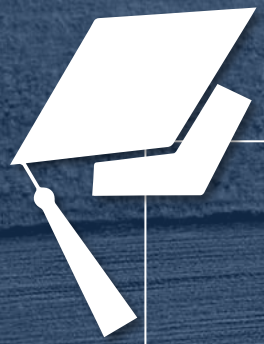
Chapter04_Example06.cpp

```
#include <stdio.h>
/**
 * Author:Lee
 * Time:2018/08/19
 * Content:练习巩固
 * Describe:求Fibonacci数列前n项
 *
 * */

/**
 * 递归方式斐波那契数列第n项
 * parma n 待求项数
 * return 若n=1或2，则返回1
 * 否则按递归方程式返回fibonacci(n) = fibonacci(n-1) + fibonacci(n-1)
 *
 **/
int fibonacci(int n){
    if(n == 1 || n == 2){
        return 1;
    }
    return (fibonacci(n-1) + fibonacci(n-2));
}

//主函数
```

编译器 资源 编译日志 调试 搜索结果

行: 48 列: 1 已选择: 0 总行数: 60 长度: 646 插入 在 0.016 秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Chapter04_Example06.cpp
#include <stdio.h>
/**
* Author:Lee
* Time:2018/08/19
* Content:练习巩固
* Describe:求Fibonacci数列前n项
*
* */
/**
* 递归方式斐波那契数列第n项
* parma n 待求项数
* return 若n=1或2，则返回1
* 否则按递归方程式返回fibonacci(n) = fibonacci
*
**/
int fibonacci(int n){
if(n == 1 || n == 2){
return 1;
}
10
1 1 2 3 5 8 13 21 34 55
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 ：
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 225.5234375 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example06.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapte
编译 0.73 秒后成功
行: 48 列: 1 已选择: 0 总行数: 60 长度: 646 插入 在 0.016 秒内完成解析



变 量

本节内容

局部变量

全局变量

练习巩固

重难点突破

重点内容：

1. 全局变量和局部变量的概念
2. 全局变量和局部变量的区别
3. 几种存储类型的作用和特点

难点问题：

1. 作用域和生存期的概念
2. static关键字的使用

变量的作用域

指变量的作用范围，即变量在程序中可以被识别的范围，分为源程序内有效、源文件内有效、函数内有效和复合语句内有效4种。

从变量的作用域（即从空间）角度来分，可以分为全局变量和局部变量。

变量的生存期

指当程序执行时变量的存在时间。即变量何时被分配空间，何时被回收空间，分为程序期、函数期和复合语句期3种。

内部变量/局部变量

在一个函数内部定义的变量称内部变量。它只在本函数范围内有效。

即：只有在本函数内才能使用这些变量，故称为局部变量 。

局部变量作用域示例

a, b, c作用域

x, y, I, j作用域

m, n作用域

```
float f1( int a) {              /*函数f1 */
    int b,c;
    …
}
char f2(int x,int y) {          /*函数f2 */
    int i,j;
}
void main( ) {                  /*主函数*/
    int m,n;
    …
}
```

局部变量使用注意事项

- 主函数中定义的变量只在主函数中有效，而不因为在主函数中定义而在整个文件或程序中有效。主函数也不能使用其他函数中定义的变量。
- 不同函数中可以使用相同名字的变量，它们代表不同的对象，互不干扰。
- 形式参数也是局部变量。
- 在一个函数内部，可以在复合语句中定义变量，这些变量只在本复合语句中有效，这种复合语句也称为“分程序”或“程序块”

外部变量/全局变量

函数之外定义的变量称为外部变量。外部变量可以为本文件中其他函数所共用。它的有效范围为从定义变量的位置开始到本源文件结束。所以也称全局变量。

全局变量的定义和初始化都是在程序编译时进行，其初始化只有一次。若没有初始化，则自动赋初值0(数值型)或’ \0’ (字符型)。

全局变量作用域示例

全局变量p, q的作用域

全局变量c1, c2的作用域

```
int p=1,q=5;                    /* 外部变量 */
float f1(int a) {               /* 定义函数f1 */
    int b,c;
    …
}

char c1,c2;                     /* 外部变量*/
char f2 (int x, int y) {        /* 定义函数f2 */
    int i,j;
    …
}

void main ( ) {                 /*主函数*/
    int m,n;
    …
}
```

全局变量使用注意事项

- 全局变量在程序的全部执行过程中都占用存储单元，而不是仅在需要时才开辟单元。
- 使用全局变量过多，会降低程序的清晰性。在各个函数执行时都可能改变外部变量的值，程序容易出错。因此，要限制使用全局变量。

变量的存储类型

格式：

[存储类别] 数据类型 变量名；

或：数据类型 [存储类别] 变量名;

如：auto int x;

或 int auto x;

变量的存储类型分类

自动变量auto

静态变量static

寄存器变量 register

外部变量extern

auto自动变量

不专门声明为static存储类别的局部变量都称为自动变量（auto）。

例如：

```
int f（int a）{              /*定义f函数， a 为形参 */
   auto int b,c = 3;         /*显式定义 b 、 c 为自动变量 */
   int c = 4;                /*c也为自动变量 */
   …
}
```

static静态变量

当函数中的局部变量的值在函数调用结束后不消失而保留原值时，就该指定该变量为静态局部变量。用关键字static进行声明

注意事项：

- 在内存中占据着永久性的存储单元
- 初始化只在编译时进行一次，每次调用时，不再重新赋值，只是保留上次调用结束时的值。

register寄存器变量

它不是存放在内存中，而是存放在CPU的寄存器中，取值速度快。

注意事项：

编译器能识别使用频繁的变量，自动将其放到寄存器中，不需程序指定，register很少用。

extern全局变量

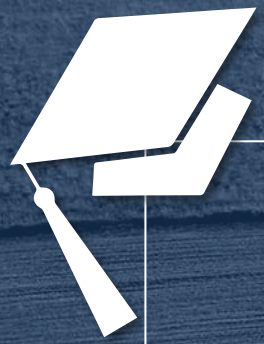
当全局变量定义后，引用它的函数在前时，需把全局变量的作用域延伸至该函数。

全局变量声明一般形式：

extern 数据类型 全局变量名；

如 extern int n;

其作用是将该全局变量的作用域从当前扩展至整个程序



宏定义

本节内容

带参数的宏定义基础知识

带参的宏与函数的区别

例题分析

练习巩固

重难点突破

重点内容：

1. 带参宏定义的一般定义方式
2. 不带参宏定义的一般定义方式
3. 两种宏定义的区别和优势

难点问题：

1. 学会运用宏定义解决实际问题
2. 学会处理宏定义的细节，保证宏展开的结果是自己需要的

宏定义的概念

- C语言中，允许用一个标识符来表示一个字符串，称为宏定义
- 被定义的标记符称为宏名
- 指定的字符串称为宏体
- 用宏体字符串替换宏名，称为宏替换，或宏展开
- 宏定义分为不带参数和带参数两种

不带参数的宏定义

- 一般形式：

#define 标识符 字符串

如：#define PI 3.1415926

宏定义声明

宏名

宏体

功能
用指定标识符(宏名)替换字符串序列(宏体)
字符串允许可包含已定义过的宏名

C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example05.cpp - [Executing] - Dev-C++ 5.6.1

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

项目管理 查看类 调试

Chapter04_Example05.cpp

```
1  #include <stdio.h>
2  /**
3   * Author:Lee
4   * Time:2018/08/19
5   * Content:不带参宏定义示例
6   * Describe:通过设计简单的无参宏定义实验，加深对宏定义的理解
7   *
8   * */
9
10 #define R  3.0
11 #define PI  3.1415926
12 #define L  2*PI*R
13 #define S  PI*R*R
14
15 int main(void){
16
17     //L宏展开: 2 * 3.1415926 * 3.0
18     //S宏展开: 3.1415926 * 3.0 * 3.0
19     printf("L=%f\nS=%f\n",L,S);
20
21     return 0;
22 }
23
24
25
26
```

```
L=18.849556
S=28.274333

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

微软拼音 半 ：

编译器 资源 编译日志 调试 搜索结果

行: 27 列: 1 已选择: 0 总行数: 27 长度: 391 插入 在 0.016 秒内完成解析

带参数的宏定义

❖一般形式：

#define 宏名(参数表) 宏体

如： #define S(a,b) a*b

宏定义声明

宏名 + 参数列表

宏体

功能
进行字符串替换，并进行参数替换

宏展开
形参用实参替换，其它字符保留
宏体及各形参外一般应加括号( )

C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example06.cpp - [Executing] - Dev-C++ 5.6.1

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

项目管理 查看类 调试

Chapter04_Example06.cpp

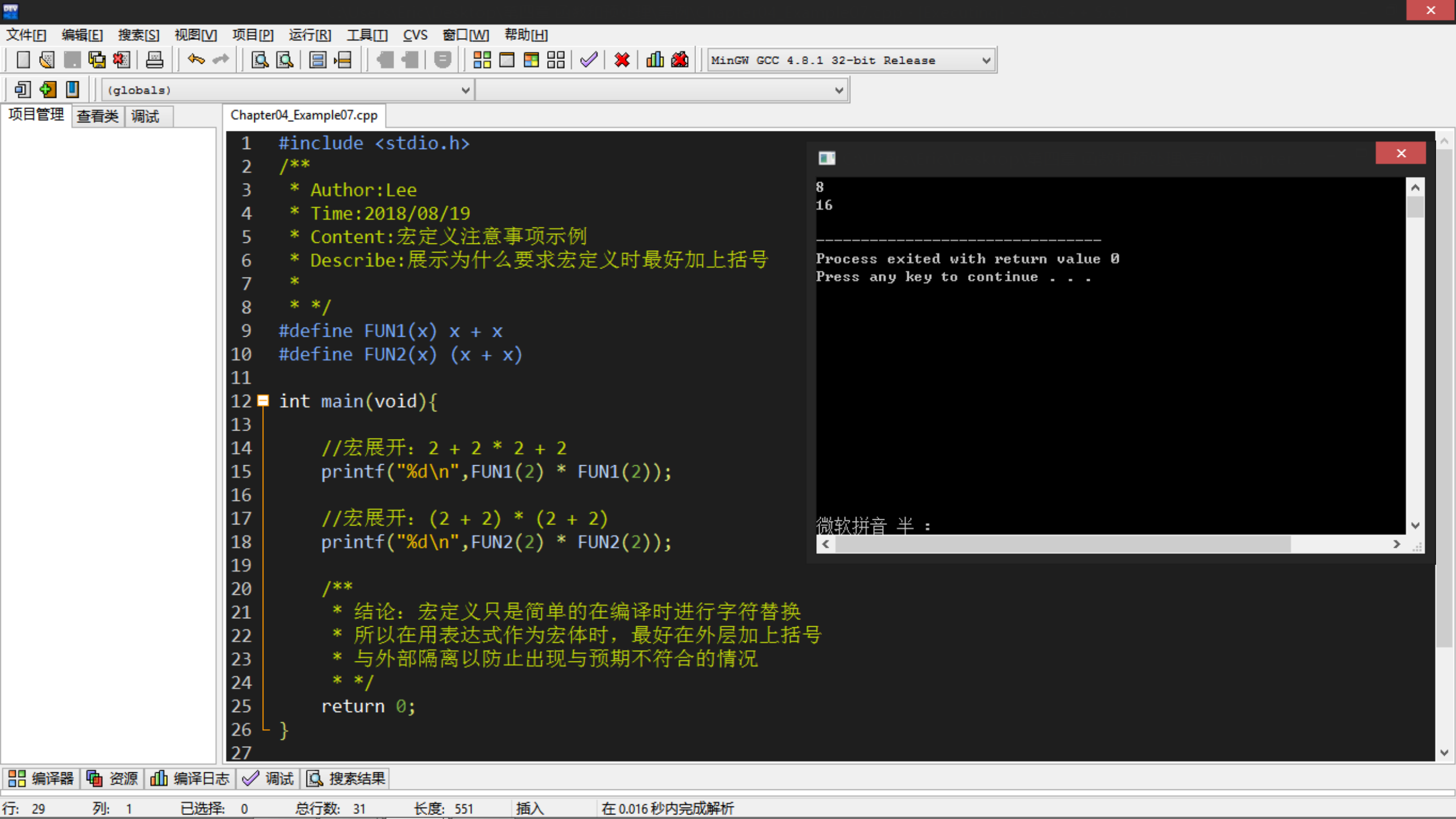
```
1  #include <stdio.h>
2  /**
3   * Author:Lee
4   * Time:2018/08/19
5   * Content:带参宏定义示例
6   * Describe:通过设计简单的带参宏定义实验，加深对宏定义的理解
7   *
8   * */
9
10 #define PI  3.1415926
11 #define S(r)  PI*r*r
12
13 int main(void){
14
15     float a,area;
16     a=3.6;
17     //S(a)宏展开: 3.1415926 * 3.6 * 3.6
18     area=S(a);
19     printf("r=%f\narea=%f\n",a,area);
20
21     return 0;
22 }
23
24
25
26
```

```
r=3.600000
area=40.715038

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

微软拼音 半 :

编译器 资源 编译日志 调试 搜索结果

行: 27 列: 1 已选择: 0 总行数: 27 长度: 380 插入 在 0.016 秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Chapter04_Example07.cpp
1 #include <stdio.h>
2 /**
3 * Author:Lee
4 * Time:2018/08/19
5 * Content:宏定义注意事项示例
6 * Describe:展示为什么要求宏定义时最好加上括号
7 *
8 * */
9 #define FUN1(x) x + x
10 #define FUN2(x) (x + x)
11
12 int main(void){
13
14 //宏展开: 2 + 2 * 2 + 2
15 printf("%d\n",FUN1(2) * FUN1(2));
16
17 //宏展开: (2 + 2) * (2 + 2)
18 printf("%d\n",FUN2(2) * FUN2(2));
19
20 /**
21 * 结论: 宏定义只是简单的在编译时进行字符替换
22 * 所以在用表达式作为宏体时, 最好在外层加上括号
23 * 与外部隔离以防止出现与预期不符合的情况
24 * */
25 return 0;
26 }
27
8
16

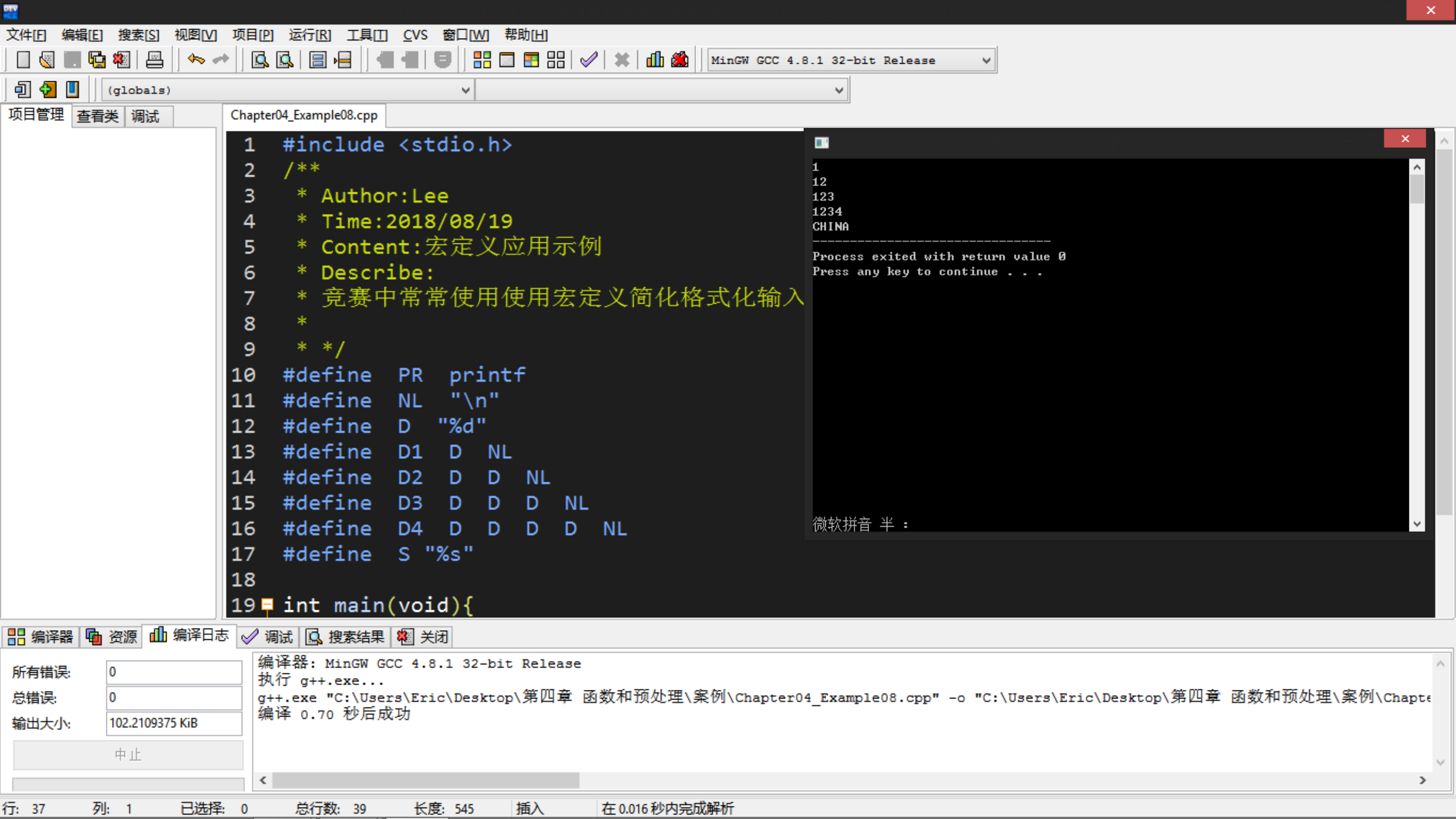
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 :
编译器 资源 编译日志 调试 搜索结果
行: 29 列: 1 已选择: 0 总行数: 31 长度: 551 插入 在 0.016 秒内完成解析

带参数的宏与函数的区别

	带参数的宏	函数
处理时间	编译时	程序运行时
参数类型	无类型问题	定义实参,形参类型
处理过程	不分配内存 简单的字符置换 无值递和返值	分配内存 先求实参值,再代入形参
程序长度	使源程序变长	不变
运行速度	不占运行时间	调用和返回占时间
值的个数	可以设法得到几个结果	只有一个返回值

Chapter04_Example08.cpp

```
#include <stdio.h>
/**
 * Author:Lee
 * Time:2018/08/19
 * Content:宏定义应用示例
 * Describe:
 * 竞赛中常常使用使用宏定义简化格式化输入输出，以提高编码速度
 *
 * */

#define  PR  printf
#define  NL  "\n"
#define  D  "%d"
#define  D1  D  NL
#define  D2  D  D  NL
#define  D3  D  D  D  NL
#define  D4  D  D  D  D  NL
#define  S "%s"

int main(void){

      int a,b,c,d;
      char string[ ]="CHINA";
      a=1;
```


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Chapter04_Example08.cpp
1 #include <stdio.h>
2 /**
3 * Author:Lee
4 * Time:2018/08/19
5 * Content:宏定义应用示例
6 * Describe:
7 * 竞赛中常常使用使用宏定义简化格式化输入
8 *
9 * */
10 #define PR printf
11 #define NL "\n"
12 #define D "%d"
13 #define D1 D NL
14 #define D2 D D NL
15 #define D3 D D D NL
16 #define D4 D D D D NL
17 #define S "%s"
18
19 int main(void){
1
12
123
1234
CHINA

Process exited with return value 0
Press any key to continue . . .
微软拼音 半 :
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 102.2109375 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_Example08.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapte
编译 0.70 秒后成功
行: 37 列: 1 已选择: 0 总行数: 39 长度: 545 插入 在 0.016 秒内完成解析

练习巩固

- Proproblem description

  分别利用宏定义和函数方式实现比较大小的功能

- Input

  无显式输入，可以在程序内直接初始化

- Output

  输出占两行。分别以函数和宏定义的方式输出，并加上适当的输出提示加以区分

文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]

MinGW GCC 4.8.1 32-bit Release

(globals)

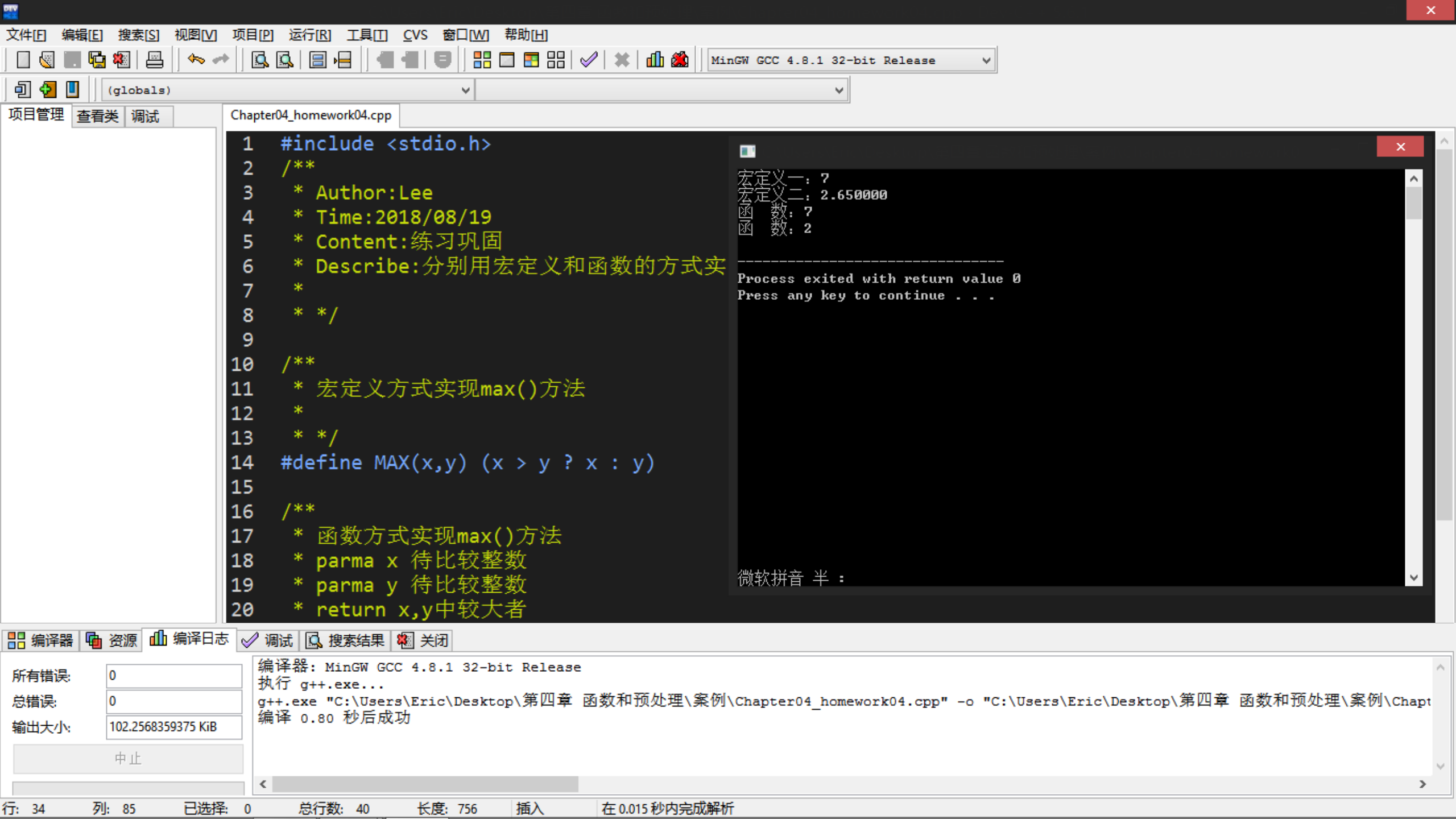
项目管理 查看类 调试

Chapter04_homework04.cpp

```
1  #include <stdio.h>
2  /**
3   * Author:Lee
4   * Time:2018/08/19
5   * Content:练习巩固
6   * Describe:分别用宏定义和函数的方式实现max()的功能
7   *
8   * */
9
10 /**
11  * 宏定义方式实现max()方法
12  *
13  * */
14 #define MAX(x,y) (x > y ? x : y)
15
16 /**
17  * 函数方式实现max()方法
18  * parma x 待比较整数
19  * parma y 待比较整数
20  * return x,y中较大者
21  *
22  * */
23 int max(int x, int y){
24      return x > y ? x : y;
25
26
```

编译器 资源 编译日志 调试 搜索结果

行: 34 列: 85 已选择: 0 总行数: 40 长度: 756 插入 在 0.015 秒内完成解析


文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] CVS 窗口[W] 帮助[H]
MinGW GCC 4.8.1 32-bit Release
(globals)
项目管理 查看类 调试
Chapter04_homework04.cpp
1 #include <stdio.h>
2 /**
3 * Author:Lee
4 * Time:2018/08/19
5 * Content:练习巩固
6 * Describe:分别用宏定义和函数的方式实
7 *
8 * */
9
10 /**
11 * 宏定义方式实现max()方法
12 *
13 * */
14 #define MAX(x,y) (x > y ? x : y)
15
16 /**
17 * 函数方式实现max()方法
18 * parma x 待比较整数
19 * parma y 待比较整数
20 * return x,y中较大者
宏定义一: 7
宏定义二: 2.650000
函 数: 7
函 数: 2
Process exited with return value 0
Press any key to continue . . .
微软拼音 半 :
编译器 资源 编译日志 调试 搜索结果 关闭
所有错误: 0
总错误: 0
输出大小: 102.2568359375 KiB
中止
编译器: MinGW GCC 4.8.1 32-bit Release
执行 g++.exe...
g++.exe "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapter04_homework04.cpp" -o "C:\Users\Eric\Desktop\第四章 函数和预处理\案例\Chapt
编译 0.80 秒后成功
行: 34 列: 85 已选择: 0 总行数: 40 长度: 756 插入 在 0.015 秒内完成解析

思维导图

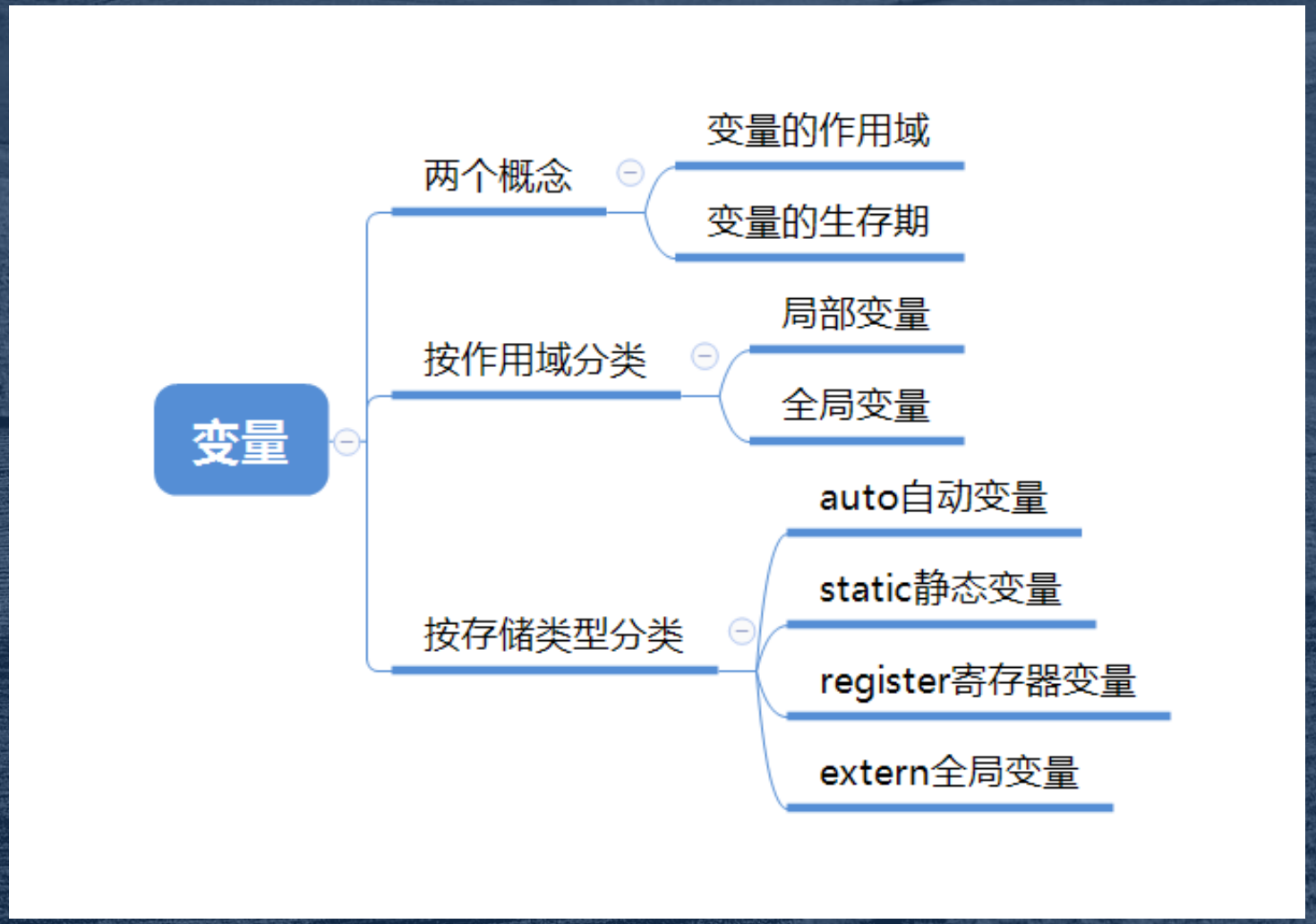
引论部分思维导图


引论
结构化程序设计思想
自顶向下
逐步求精
模块化
C语言程序结构
C语言源程序
C语言源文件1
C语言源文件i
函数1
函数i
声明部分
执行部分
函数n
C语言源文件n
中

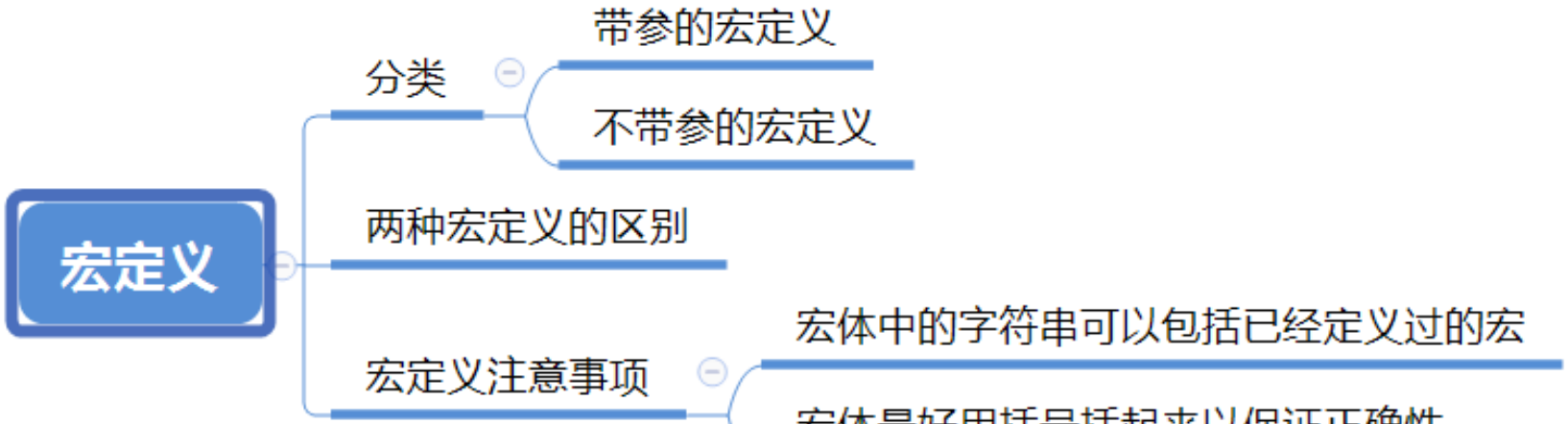
函数部分思维导图


函数
函数的一般定义形式
函数返回值类型
函数名
参数列表
函数体
声明部分
执行部分
函数参数和返回值
参数
形参
实参
返回值
无返回值
return;
return 表达式;
return (表达式);
推荐使用
函数调用基础
函数语句调用
函数表达式调用
函数参数调用
函数的嵌套调用
规则
允许嵌套调用
不允许嵌套定义
两种嵌套调用方式
递归
定义
直接或间接的调用自身
要素
递归方程式
递归出口

变量部分思维导图


变量
两个概念
变量的作用域
变量的生存期
按作用域分类
局部变量
全局变量
按存储类型分类
auto自动变量
static静态变量
register寄存器变量
extern全局变量

宏定义部分思维导图


宏定义
分类
带参的宏定义
不带参的宏定义
两种宏定义的区别
宏定义注意事项
宏体中的字符串可以包括已经定义过的宏
宏体最好用括号括起来以保证正确性

教学评价

完成本章的作业和练习题后，请在QQ群中根据自己对本章知识的掌握情况，进行投票，我将根据同学们的投票，及时了解同学们对本章的掌握情况，并做出教学调整。

练习题库-笔试

练习题库-上机

自测系统

FSCapture

Dev-Cpp编程工具

ZoomIt

Xmind-8

课件下载网址：

http://www.jsjx.com/index.aspx

推荐资料1 - C语言经典编程282例

推荐资料2 - 华为C语言编程规范

推荐资料3 - 嗨翻C

推荐资料4 - C&C++ API.chm

程序社团-技术交流群：540653133

课件下载网址：

http://jsjxy.hhhxy.cn/info/1074/2602.htm

课件下载二维码：

感谢聆听

下一章内容：数组

主讲人：李伦彬

请同学们提前预习课本相关章节