



4.1 函数的基本使用

李洁



目录

基本概念

参数

返回值

调用

库函数与自定义函数



4.1 函数的基本使用

4.1.1 基本概念

什么是函数？



4.1 函数的基本使用

4.1.1 基本概念

什么是函数？

函数：一个固定的**程序段**，包含若干条语句，用来实现某种**独立的功能**。在其他函数（包括主函数）中，可以使用**函数名**来**调用**这个程序段。

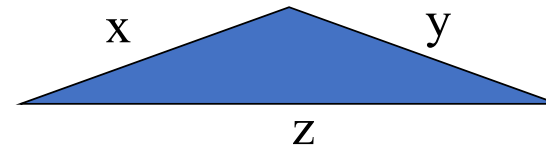
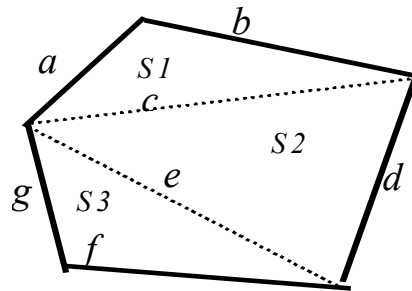


4.1 函数的基本使用

4.1.1 基本概念

例：已知五边形的各条边的长度，计算其面积

- 计算多边形面积，可将多边形分解成若干个三角形



- 计算三角形面积的公式如下：

$$area = \sqrt{c(c-x)(c-y)(c-z)} \quad c = \frac{1}{2}(x+y+z)$$



4.1 函数的基本使用

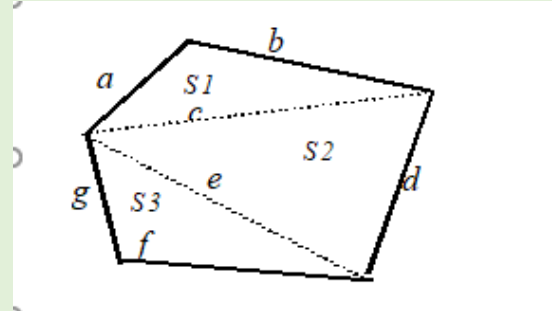
4.1.1 基本概念

使用之前所学知识实现上例



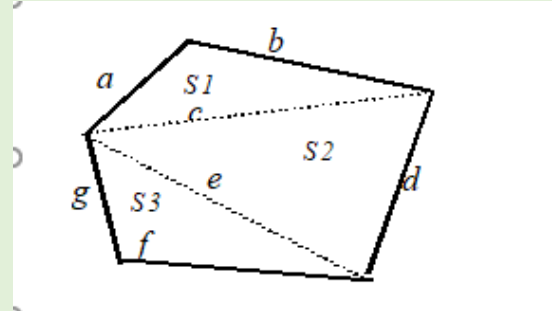
```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    float a, b, c, d, e, f, g, p1, p2, p3, s1, s2, s3, s;
    cin >> a >> b >> c >> d >> e >> f >> g;

    cout << s << endl;
    return 0;
}
```



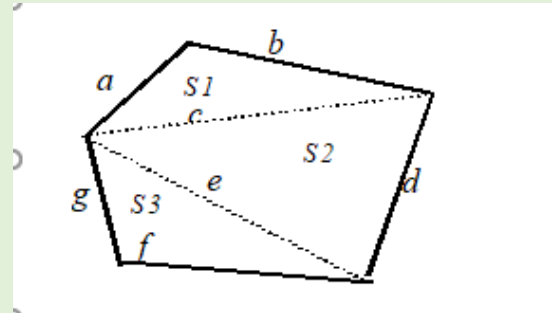


```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    float a, b, c, d, e, f, g, p1, p2, p3, s1, s2, s3, s;
    cin >> a >> b >> c >> d >> e >> f >> g;
    p1 = (a + b + c) / 2;
    s1 = sqrt(p1 * (p1 - a) * (p1 - b) * (p1 - c));
    p2 = (c + d + e) / 2;
    s2 = sqrt(p2 * (p2 - c) * (p2 - d) * (p2 - e));
    p3 = (e + f + g) / 2;
    s3 = sqrt(p3 * (p3 - e) * (p3 - f) * (p3 - g));
    s = s1 + s2 + s3;
    cout << s << endl;
    return 0;
}
```





```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    float a, b, c, d, e, f, g, p1, p2, p3, s1, s2, s3, s;
    cin >> a >> b >> c >> d >> e >> f >> g;
    p1 = (a + b + c) / 2;
    s1 = sqrt(p1 * (p1 - a) * (p1 - b) * (p1 - c));
    p2 = (c + d + e) / 2;
    s2 = sqrt(p2 * (p2 - c) * (p2 - d) * (p2 - e));
    p3 = (e + f + g) / 2;
    s3 = sqrt(p3 * (p3 - e) * (p3 - f) * (p3 - g));
    s = s1 + s2 + s3;
    cout << s << endl;
    return 0;
}
```

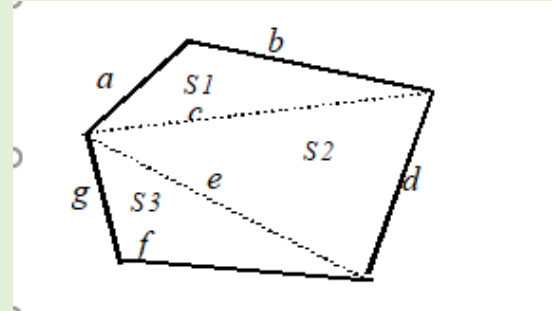


Microsoft Visual Studio 调试控制台

```
2 2 2 2 2 2 2
5.19615
```



```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    float a, b, c, d, e, f, g, p1, p2, p3, s1, s2, s3, s;
    cin >> a >> b >> c >> d >> e >> f >> g;
    p1 = (a + b + c) / 2;
    s1 = sqrt(p1 * (p1 - a) * (p1 - b) * (p1 - c));
    p2 = (c + d + e) / 2;
    s2 = sqrt(p2 * (p2 - c) * (p2 - d) * (p2 - e));
    p3 = (e + f + g) / 2;
    s3 = sqrt(p3 * (p3 - e) * (p3 - f) * (p3 - g));
    s = s1 + s2 + s3;
    cout << s << endl;
    return 0;
}
```



辨析:

★ 使用非函数的普通程序段来计算五边形面积时，需要重复编写海伦公式三次，且三次公式用到的计算变量都不相同，很容易因为复制粘贴语句而产生错误



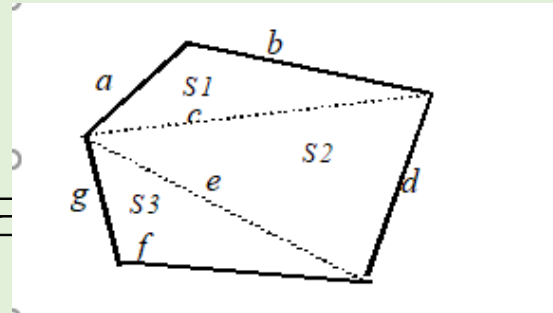
4.1 函数的基本使用

4.1.1 基本概念

用函数实现上例：

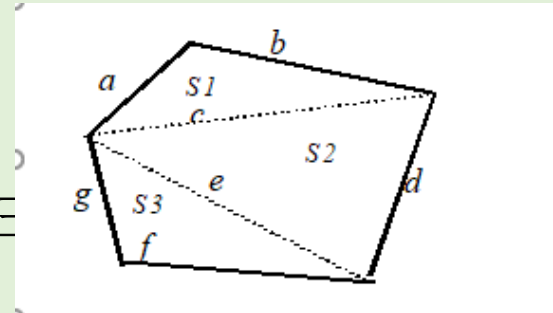


```
#include <cmath>
#include<iostream>
using namespace std;
float  area(float x, float y, float z)//求三
{
    float  c, s;
    c = (x + y + z) / 2;
    s = sqrt(c * (c - x) * (c - y) * (c - z));
    return s;
}
int main()                //主函数
{
    float  a, b, c, d, e, f, g, s;
    cin >> a >> b >> c >> d >> e >> f >> g;
    s = area(a, b, c) + area(c, d, e) + area(e, f, g);
    cout << s << endl;
    return 0;
}
```





```
#include <cmath>
#include<iostream>
using namespace std;
float  area(float x, float y, float z)//求三
{
    float  c, s;
    c = (x + y + z) / 2;
    s = sqrt(c * (c - x) * (c - y) * (c - z));
    return s;
}
int main()                //主函数
{
    float  a, b, c, d, e, f, g, s;
    cin >> a >> b >> c >> d >> e >> f >> g;
    s = area(a, b, c) + area(c, d, e) + area(e, f, g);
    cout << s << endl;
    return 0;
}
```

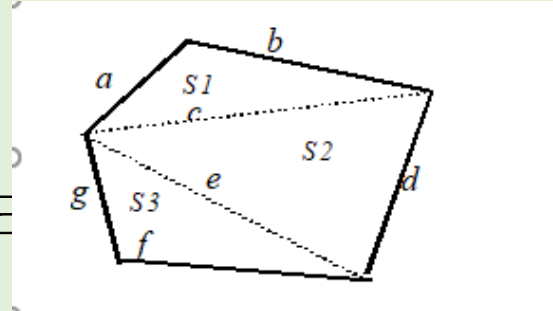


C:\ Microsoft Visual Studio 调试控制台

```
2 2 2 2 2 2 2
5.19615
```



```
#include <cmath>
#include<iostream>
using namespace std;
float  area(float x, float y, float z)//求三
{
    float  c, s;
    c = (x + y + z) / 2;
    s = sqrt(c * (c - x) * (c - y) * (c - z));
    return s;
}
int main()                //主函数
{
    float  a, b, c, d, e, f, g, s;
    cin >> a >> b >> c >> d >> e >>
    s = area(a, b, c) + area(c, d,
    cout << s << endl;
    return 0;
}
```



辨析:

★ 使用函数来计算五边形面积时，程序清晰易读，且用到海伦公式的地方公式只用写一次即可



4.1 函数的基本使用

4.1.1 基本概念

为什么要使用函数？

实现封装 (Encapsulation) 和抽象 (abstraction)：

例：我们经常使用的电子计算器，只要输入数字和计算符，就可以得出结果。我们在使用计算器时不需要考虑它是具体如何进行计算的过程的，只需要输入算式（函数的输入参数），按动计算器（调用函数），就可以得到计算结果（函数的输出）。

使用函数有利于代码重用，提高开发效率



§2.基础知识



2.3.C++程序简介

2.3.2.程序结构的基本形式

- ★ C++程序由函数组成，函数是C++程序的基本单位
- ★ 一个C++程序可由若干源程序文件（*.cpp）组成，每个源程序文件可以包含若干函数
- ★ 有且仅有一个名为main()的函数，称为主函数，程序的执行从它开始
- ★ C++提供许多库函数（已做好，可直接使用的）

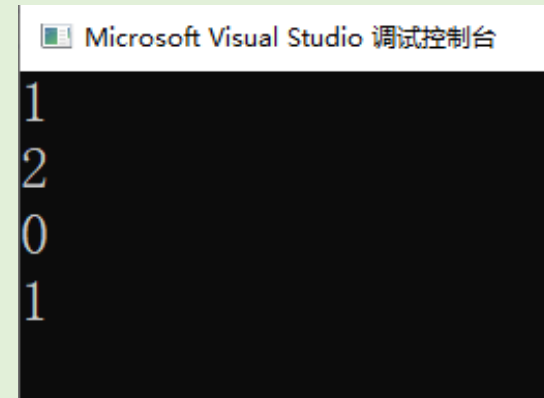
```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, World." << endl;
    return 0;
}
```




§2.基础知识

两种方法验证

```
#include <iostream>
using namespace std;
int main()
{
    cout << sizeof("") << endl;
    cout << sizeof(" ") << endl;
    cout << strlen("") << endl;
    cout << strlen(" ") << endl;
    return 0;
}
```



\0

32 \0

2.6.常

2.6.3.5

2.6.3.4

在内存

★ ""与

"" -

"" -

白 个 工 作 的 子 符 串 ， 长 度 为 1



4.1 函数的基本使用

4.1.1 基本概念

C/C++函数定义格式

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, World." << endl;
    return 0;
}
```



4.1 函数的基本使用

4.1.1 基本概念

C/C++函数定义格式

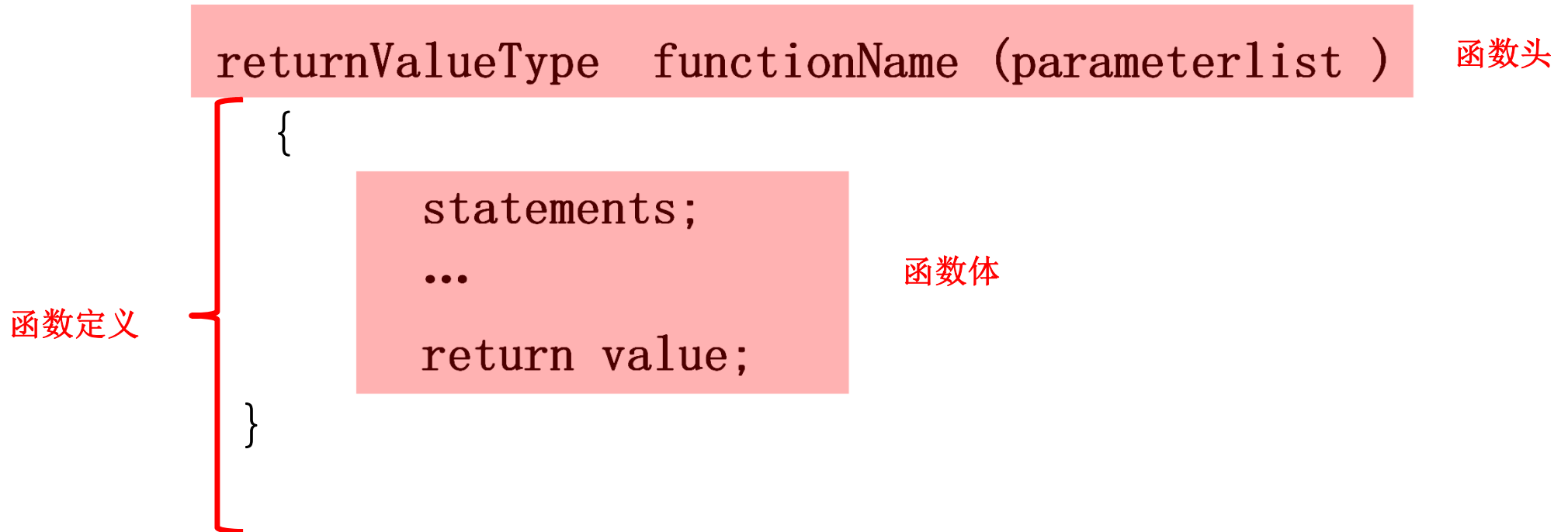
```
returnValueType  functionName (parameterlist )  
{  
    statements;  
    ...  
    return value;  
}
```



4.1 函数的基本使用

4.1.1 基本概念

C/C++函数定义格式





4.1 函数的基本使用

4.1.2 参数

```
returnValueType  functionName ( parameterlist )  
{  
    statements;  
    ...  
    return value;  // not necessary  
}
```

函数的输入部分, 可以没有输入参数/也可以有多个输入参数;
输入参数的数据类型可以是常量, 变量, 数组, 指针, 结构体,
表达式等多种数据类型。

函数没有输入参数 (即输入参数parameterlist为空), 此时函数的输入参数可以不填或者填写成void (空类型)



4.1 函数的基本使用

4.1.2 参数

对于没有输入参数的函数，在调用该函数时，输入参数不填写，
注意括号不可省略

例

```
#include <iostream>
using namespace std;

int main()
{
    char c;
    c = getchar(); //无输入参数
    //库函数，从键盘读入一个字符
    return 0;
}
```



4.1 函数的基本使用

4.1.2 参数

函数的输入参数的赋值需匹配其数据类型，否则会产生隐式的类型转换或是错误

```
returnValueType  functionName ( parameterlist )
{
    statements;
    ...
    return value;  // not necessary
}
```



```
#include<iostream>
using namespace std;
double calGrade(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()                //主函数
{
    double result;
    result = calGrade(90, 80, 40); //正确调用函数
    cout << result;
    return 0;
}
```




```
#include<iostream>
using namespace std;
double calGrade(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()           //主函数
{
    double result;
    result = calGrade(90, 80, 40); //正确调用函数
    cout << result;
    return 0;
}
```

C:\Users'

57



4.1 函数的基本使用

4.1.2 参数

错例：错误使用函数的输入参数，数据类型不匹配



```
#include<iostream>
using namespace std;
double calGrade(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()                //主函数
{
    double result;
    result = calGrade(90, 80, '0' );    //调用函数
    cout << result;
    return 0;
}
```



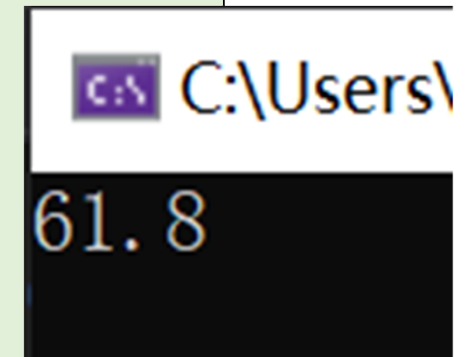
```
#include<iostream>
using namespace std;
double calGrade(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()           //主函数
{
    double result;
    result = calGrade(90, 80, '0');    //调用函数
    cout << result;
    return 0;
}
```

C:\Users\

61.8



```
#include<iostream>
using namespace std;
double calGrade(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()           //主函数
{
    double result;
    result = calGrade(90, 80, '0');
    cout << result;
    return 0;
}
```



C:\Users\
61.8

辨析:

此时产生了隐式的类型转换, 将字符 '0' 转换为它的ASCII码值即48, $90 * 0.1 + 80 * 0.3 + 48 * 0.6 = 61.8$

不会发出警告, 因为发生了符合语法的隐式的类型转换



4.1 函数的基本使用

4.1.2 参数

可以为参数列表中**后边的每一个**参数指定默认值。当调用函数时，如果未传递参数的值，则会使用默认值，如果指定了值，则会忽略默认值，使用传递的值。

```
returnValueType  functionName ( parameterlist )  
    {  
        statements;  
        ...  
        return value;  // not necessary  
    }
```

```
#include <iostream>
using namespace std;
int sum(int a, int b=20)
{
    int result;
    result = a + b;
    return (result);
}

int main ()
{
    int a = 100;
    int b = 200;
    int result;
    result = sum(a, b);
    cout << "Total value is :" << result << endl;
    result = sum(a);
    cout << "Total value is :" << result << endl;
    return 0;
}
```





```
#include <iostream>
using namespace std;
int sum(int a, int b=20)
{
    int result;
    result = a + b;
    return (result);
}

int main ()
{
    int a = 100;
    int b = 200;
    int result;
    result = sum(a, b);
    cout << "Total value is :" << result << endl;
    result = sum(a);
    cout << "Total value is :" << result << endl;
    return 0;
}
```

C:\Users\kevin\Desktop\

Total value is :300
Total value is :120



```
#include <iostream>
using namespace std;
int sum(int a, int b=20)//给出默认值
{
    int result;
    result = a + b;
    return (result);
}

int main ()
{
    int a = 100;
    int b = 200;
    int result;
    result = sum(a, b);
    cout << "Total value is :" << result
    result = sum(a);
    cout << "Total value is :" << result << endl;
    return 0;
}
```

C:\Users\kevin\Desktop\

```
Total value is :300
Total value is :120
```

辨析:

调用函数时，如果未传递参数的值，则会使用默认值，如果指定了值，则会忽略默认值，使用传递的值。



```
#include <iostream>
using namespace std;
int sum(int a=20, int b)//给出默认值
{
    int result;
    result = a + b;
    return (result);
}

int main ()
{
    int a = 100;
    int b = 200;
    int result;
    result = sum(a, b);
    cout << "Total value is :" << result << endl;
    result = sum(b);
    cout << "Total value is :" << result << endl;
    return 0;
}
```

能给a默认值，给b指定值么？



```
#include <iostream>
using namespace std;
int sum(int a=20, int b)//给出默认值
{
    int result;
    result = a + b;
    return (result);
}

int main ()
{
    int a = 100;
    int b = 200;
    int result;
    result = sum(a, b);
    cout << "Total value is :";
    result = sum(b);
    cout << "Total value is :";
    return 0;
}
```

辨析:

会报错!




只能为参数列表中**后边的**每一个参数指定默认值。

错误列表

整个解决方案

错误 3

警告 0

	代码	说明
	E0306	默认实参不在形参列表的结尾
	E0165	函数调用中的参数太少
	C2548	"sum": 缺少形参 2 的默认实参



4.1 函数的基本使用

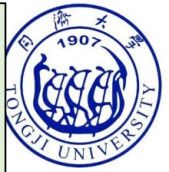
4.1.2 参数

```
returnValueType  functionName ( parameterlist )  
    {  
        statements;  
        ...  
        return value;  // not necessary  
    }
```

```
void swap(int var1, int var2)
{
    int temp;
    cout << "(before)var1:" << var1 << "\n";
    cout << "(before)var2:" << var2 << "\n";
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "(after)var1:" << var1 << "\n";
    cout << "(after)var2:" << var2 << "\n";
}

int main()                //主函数
{
    int a, b;
    cin >> a >> b;
    swap(a, b);
    return 0;
}
```





```
void swap(int var1, int var2)
{
    int temp;
    cout << "(before)var1:" << var1 << endl;
    cout << "(before)var2:" << var2 << endl;
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "(after)var1:" << var1 << endl;
    cout << "(after)var2:" << var2 << endl;
}

int main()                //主函数
{
    int a, b;
    cin >> a >> b;
    swap(a, b);
    return 0;
}
```

辨析:

a, b 称为函数的**实际参数**,

用于传递值的变量, 出现在调用者(不一定是main函数)的语句段中, 不能在swap函数中使用。

var1, var2称为函数的**形式参数**,

用于接收传递值的变量, 只能在swap函数内部使用。

在函数调用的过程中, 将**实参**a, b的**值传递给形参**var1, var2。形参和实参的**命名不需要相同**。



4.1 函数的基本使用

4.1.2 参数

函数的**形式参数**:

用于**接收传递值**的变量,
出现在**被调用函数的定义**中,
只能在该函数内部使用。在函
数被调用时创建并被分配资源,
在函数返回时删除并回收资源。

函数的**实际参数**:

用于**传递值**的变量, 出
现在**调用者**(不一定是main
函数)的语句段中, 实参**不
能**在被调用的函数中使用。

在函数调用的过程中, 程序通常将**实参的值传递给形参**。形参和
实参的**命名不需要相同**。



```
void swap(int var1, int var2)
{
    int temp;
    cout << "(before)var1:" << var1 << "\n";
    cout << "(before)var2:" << var2 << "\n";
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "(after)var1:" << var1 << "\n";
    cout << "(after)var2:" << var2 << "\n";
}

int main()                //主函数
{
    int a, b;
    cin >> a >> b;
    swap(a, b);
    return 0;
}
```

辨析:

在调用swap函数时，传递给swap函数的是参数a, b的**值**。相当于先将a, b变量的值进行**复制**，再将复制的值**传递**给一个刚创建的**新的变量**var1, var2，再对它们进行操作；



```
void swap(int var1, int var2) //因为页面缩放大小问题未一句一行
```

```
{ // var1, var2为形式参数
    int temp;
    cout << "(before)var1:" << var1 << "\n";
    cout << "(before)var2:" << var2 << "\n";
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "(after)var1:" << var1 << "\n";
    cout << "(after)var2:" << var2 << "\n"; }

int main() //主函数
```

```
{ int a, b;
  cin >> a >> b;
  cout << "(before)a:" << a << "\n";
  cout << "(before)b:" << b << "\n";
  swap(a, b); // a, b为实际参数
  cout << "(after)a:" << a << "\n";
  cout << "(after)b:" << b << "\n";
  return 0; }
```


输入 a=1, b=2, 在执行函数 swap 交换语句之前和之后, 形参 var1, var2 和实参 a, b 的值是多少?



```
void swap(int var1, int var2) //因为页面缩放大小问题未一句一行
```

```
{ // var1, var2为形式参数
    int temp;
    cout << "(before)var1:" << var1 << "\n";
    cout << "(before)var2:" << var2 << "\n";
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "(after)var1:" << var1 << "\n";
    cout << "(after)var2:" << var2 << "\n"; }
```

```
int main() //主函数
{
    int a, b;
    cin >> a >> b;
    cout << "(before)a:" << a << "\n";
    cout << "(before)b:" << b << "\n";
    swap(a, b); // a, b为实际参数
    cout << "(after)a:" << a << "\n";
    cout << "(after)b:" << b << "\n";
    return 0; }
```

 Microsoft Visual

```
1 2
(before)a:1
(before)b:2
(before)var1:1
(before)var2:2
(after)var1:2
(after)var2:1
(after)a:1
(after)b:2
```



```
void swap(int var1, int var2)
{
    int temp;
    cout << "(before)var1:" << var1 << "\n";
    cout << "(before)var2:" << var2 << "\n";
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "(after)var1:" << var1 << "\n";
    cout << "(after)var2:" << var2 << "\n";
}

int main()                //主函数
{
    int a, b;
    cin >> a >> b;
    swap(a, b);
    return 0;
}
```

辨析:

在调用swap函数时，传递给swap函数的是参数a, b的**值**。相当于先将a, b变量的值进行**复制**，再将复制的值**传递**给一个刚创建的**新的变量**var1, var2，再对它们进行操作；

swap函数体内对于参数var1, var2的操作，不会改变函数体外（调用它的程序）中的参数a, b的值。

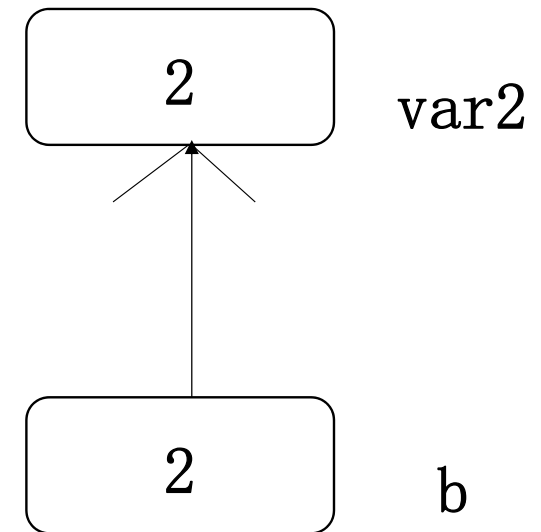
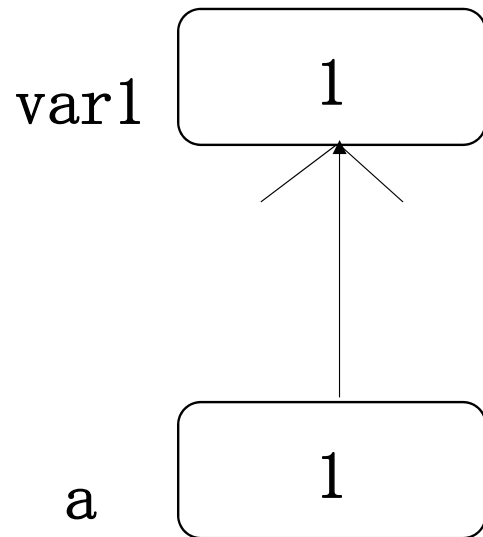


4.1 函数的基本使用

4.1.2 参数

过程分析：

调用函数swap(a, b)时, 创建新变量var1, var2, 并将a, b的值赋给它们



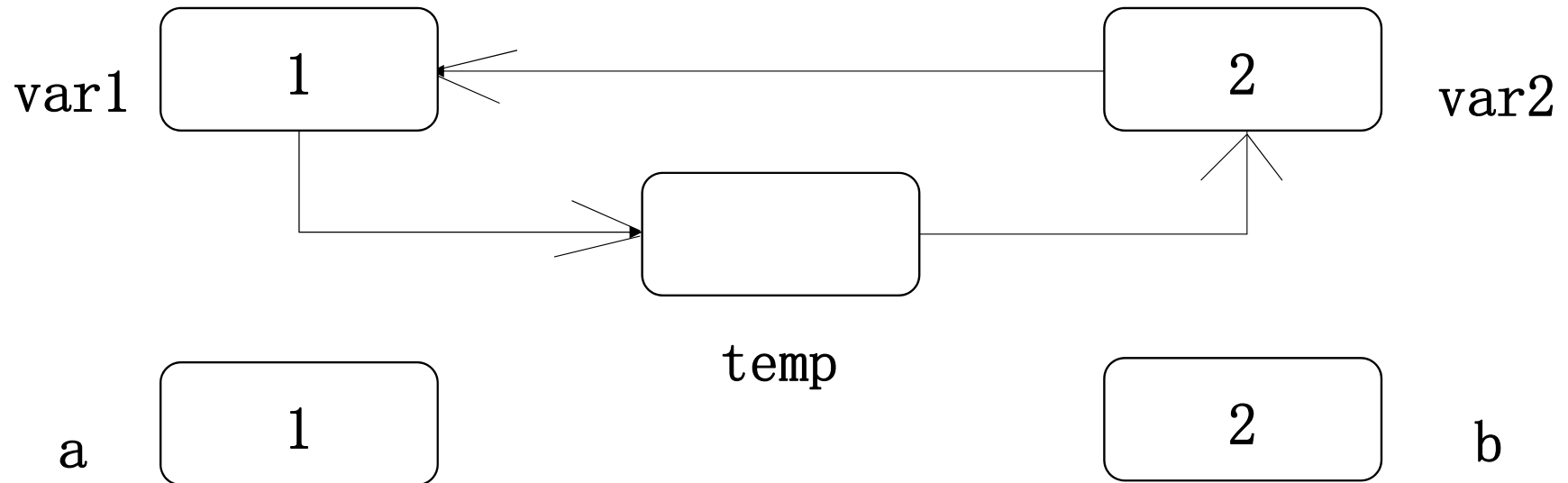


4.1 函数的基本使用

4.1.2 参数

过程分析:

运行语句 `temp = var1;` `var1 = var2;` `var2 = temp;` 时





```
void swap(int var1, int var2) //因为页面缩放大小问题未一句一行
```

```
{ // var1, var2为形式参数
```

```
    int temp;
```

```
    cout << "(before)var1:" << var1 << "\n";
```

```
    cout << "(before)var2:" << var2 << "\n";
```

```
    temp = var1;
```

```
    var1 = var2;
```

```
    var2 = temp;
```

```
    cout << "(after)var1:" << var1 << "\n";
```

```
    cout << "(after)var2:" << var2 << "\n"; }
```

```
int main() //主函数
```

```
{    int a, b;
```

```
    cin >> a >> b;
```

```
    cout << "(before)a:" << a << "\n";
```

```
    cout << "(before)b:" << b << "\n";
```

```
    swap(a, b); // a, b为实际参数
```

```
    cout << "(after)a:" << a << "\n";
```

```
    cout << "(after)b:" << b << "\n";
```

```
    return 0; }
```

C:\ Microsoft Visual

1 2

(before) a:1

(before) b:2

辨析:

在调用交换函数swap的前后，实际参数a, b的值**都不会改变**；

形式参数var1, var2的值在交换函数语句**运行前**与实际参数的值**一致**，在交换语句**运行后**两个参数的值发生了**交换**

(after) b:2



4.1 函数的基本使用

4.1.2 参数

按值传递:

在调用函数时, 传递给函数的是参数的**值**。相当于先将输入函数的变量的值进行**复制**, 再将复制的值**传递**给一个刚创建的**新的变量**, 并将新的变量放入函数的语句中操作。

单向传值: 此时函数体内对于参数的操作, 不会改变函数体外 (调用它的程序) 中的参数变量的值。



4.1 函数的基本使用

4.1.2 参数

按地址传递:

在调用函数时，传递给函数的是参数的地址。相当于直接将输入函数的变量放入函数的语句中操作。

`Void swap(int *x, int *y));` //通过指针调用。向函数传递参数的指针调用方法，把参数的地址复制给形式参数。

`void swap(int &x, int &y) ;` //通过引用传值。把引用的地址复制给形式参数。

此时修改形式参数会影响实际参数（以后课程会讲）。



4.1 函数的基本使用

4.1.3 返回值

```
returnValueType  functionName (parameterlist )  
{  
    statements;  
    ...  
    return value; // not necessary  
}
```

函数的输出部分，可以没有输出参数/也可以有多个输出参数；参数的数据类型**不可以是数组**，但可以是常量，变量，指针，结构体，表达式等其他任何类型。

返回值类型字段returnValueType为void（空类型）时，表示没有输出参数，此时不写返回语句或将返回语句写为return;



4.1 函数的基本使用

4.1.3 返回值

例

```
double calGrade(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
```

参数gRe为该函数的返回值



4.1 函数的基本使用

4.1.3 返回值

对于**没有返回值**的函数，其returnValueType字段为void, 其返回语句可以不写或是写成return;

例

```
void swap(int var1, int var2)
{
    int temp;
    temp = var1;
    var1 = var2;
    var2 = temp;
    cout << "var1:" << var1 << "\n";
    cout << "var2:" << var2 << "\n";
    return; //可省略
}
```



4.1 函数的基本使用

4.1.3 返回值

主函数`main`的返回值类型不能是`void`，主函数的返回值类型只能是`int`，其返回值`return 0`;表示该程序成功运行并结束返回(提交操作系统)

正例

```
#include <iostream>
using namespace std;
int main()
{
    cout << "hello!";
    return 0;
}
```



4.1 函数的基本使用

4.1.3 返回值

错例

```
#include <iostream>
using namespace std;
void main()
{
    cout << "hello!";
}
```



4.1 函数的基本使用

4.1.3 返回值

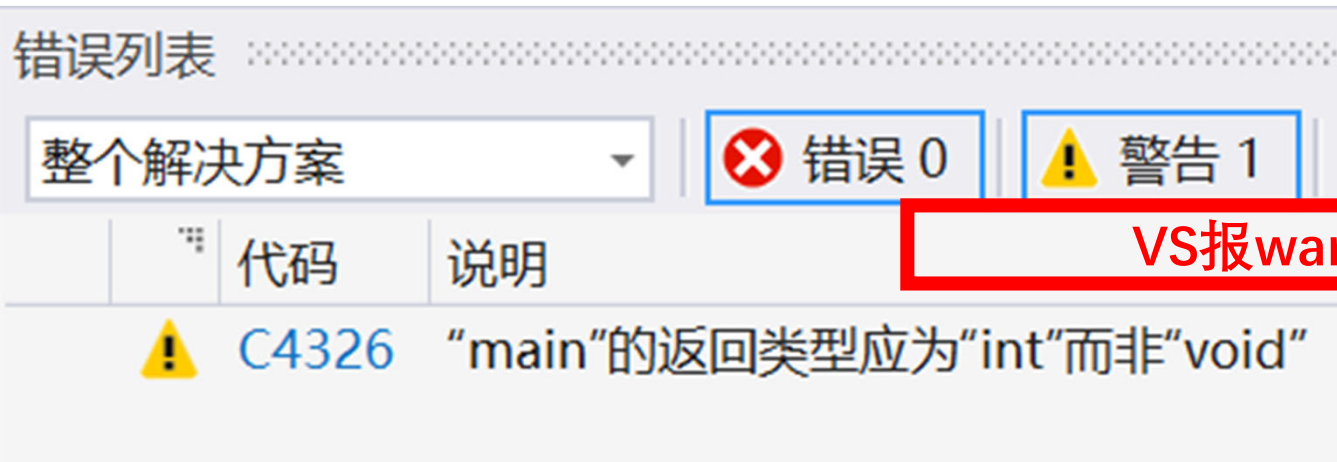
错例

```
#include <iostream>
using namespace std;
void main()
{
    cout << "hello!";
}
```

Dev报error

信息

[Error] '::~main' must return 'int'



VS报warning



4.1 函数的基本使用

4.1.3 返回值

如函数的返回值和`returnValueType`指明的数据类型不匹配，会产生隐式的类型转换或是错误。

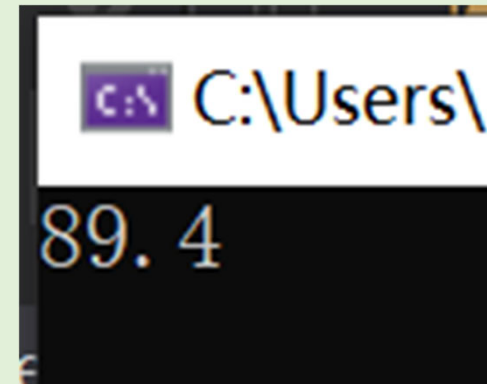
正例：



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()           //主函数
{
    double result;
    result = calGrade1(90, 80, 94); //调用函数
    cout << result;
    return 0;
}
```




```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()                //主函数
{
    double result;
    result = calGrade1(90, 80, 94);    //调用函数
    cout << result;
    return 0;
}
```





4.1 函数的基本使用

4.1.3 返回值

如函数的返回值和`returnValueType`指明的数据类型不匹配，会产生隐式的类型转换或是错误。

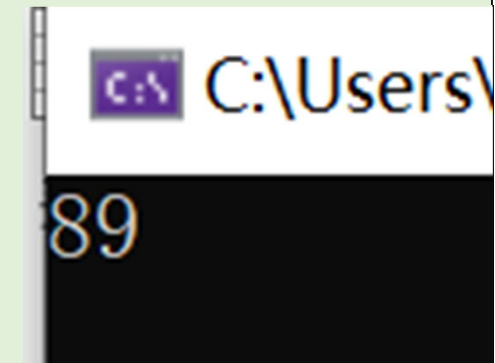
反例：



```
#include<iostream>
using namespace std;
int calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe; //函数返回值的数据类型不匹配
}
int main()                //主函数
{
    double result;
    result = calGrade1(90, 80, 94);    //调用函数
    cout << result;
    return 0;
}
```

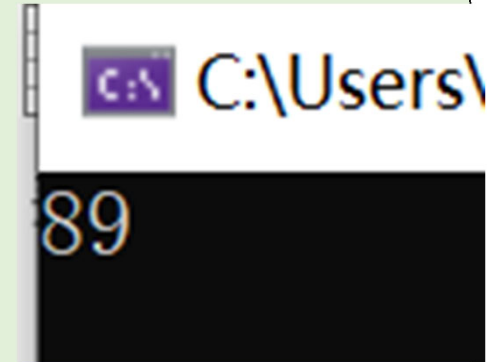


```
#include<iostream>
using namespace std;
int calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe; //函数返回值的数据类型不匹配
}
int main()           //主函数
{
    double result;
    result = calGrade1(90, 80, 94); //调用函数
    cout << result;
    return 0;
}
```





```
#include<iostream>
using namespace std;
int calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe; //函数返回值的数据类型不匹配
}
int main()           //主函数
{
    double result;
    result = calGrade1(90, 80, 94);
    cout << result;
    return 0;
}
```



辨析:

此时产生了隐式的类型转换, 将函数中计算得到的double类型返回值2.5隐式转换为int类型, 导致最后输出错误结果89



```
#include<iostream>
using namespace std;
int calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe; //函数返回值的数据类型不匹配
}

int main()
{
    double result;
    result = calGrade
    cout << result;
    return 0;
}
```

错误列表

整个解决方案

❌ 错误 0

⚠️ 警告 1

📘 消息 0



代码

说明

[C4244](#)

"return": 从"double"转换到"int", 可能丢失数据



4.1 函数的基本使用

4.1.3 返回值

- 函数的返回语句、调用函数的语句以及函数定义中的返回值**数据类型应相同**，在类型不同时，程序在运行时会发生隐式的自动类型转换（具体转换情况参考数据类型的相关章节）。其数据变量在类型转换时可能不会使程序出错中止，而只是**产生警告**，但可能会使程序**得到错误的结果**。



4.1 函数的基本使用

4.1.3 返回值

函数可以有**多条返回语句**，只要函数执行到一条返回语句，函数就会**立即结束并返回**

例：



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a > b)
        return a;
    else
        return b;
}
int main()
{
    int a, b;
    cin >> a >> b;
    cout << print_bigger (a, b);
    return 0;
}
```

输入 a=2, b=1, 输出?

输入 a=1, b=2, 输出?



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a > b)
        return a;
    else
        return b;
}
int main()
{
    int a, b;
    cin >> a >> b;
    cout << print_bigger (a, b);
    return 0;
}
```

A screenshot of a Microsoft C++ console window. The prompt 'c:\>' is followed by the text 'Microsoft'. Below this, the input '2 1' is shown on one line, and the output '2' is shown on the next line, indicating that the function correctly identified 2 as the larger number.

A screenshot of a Microsoft C++ console window. The prompt 'c:\>' is followed by the text 'Microsoft'. Below this, the input '1 2' is shown on one line, and the output '2' is shown on the next line, indicating that the function correctly identified 2 as the larger number.



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a > b)
        return a;
    //else 去掉else句 会怎么样?
        return b;
}
int main()
{
    int a, b;
    cin >> a >> b;
    cout << print_bigger (a, b);
    return 0;
}
```

输入 a=2, b=1, 输出?

输入 a=1, b=2, 输出?



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a > b)
        return a;
    //else 去掉else句 会怎么样?
        return b;
}
int main()
{
    int a, b;
    cin >> a >> b;
    cout << print_bigger (a, b);
    return 0;
}
```

输入 a=2, b=1, 输出?

输入 a=1, b=2, 输出?



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a > b)
        return a;
    //else 去掉else句 会怎么样?
        return b;
}
int main()
{
    int a, b;
    cin >> a >> b;
    cout << print_bigger (a, b);
    return 0;
}
```

A screenshot of a Microsoft Windows command prompt window. The title bar says "C:\> Microsoft". The command prompt shows the input "2 1" and the output "2".

```
C:\> Microsoft
2 1
2
```

A screenshot of a Microsoft Windows command prompt window. The title bar says "C:\> Microsoft". The command prompt shows the input "1 2" and the output "2".

```
C:\> Microsoft
1 2
2
```



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a > b)
        return a;
    //else 去掉else句 会怎么样?
        return b;
}
int main()
{
    int a, b;
    cin >> a >> b;
    cout << print_bigger (a, b);
    return 0;
}
```

辨析:

删去else句程序运行的**结果不变**，这是因为当函数的输入参数 $a > b$ 时，程序执行到语句return a; 时会**立刻结束** print_bigger函数并返回值；而当函数的输入参数 $a \leq b$ 时，程序会跳过if语句，并剩余的语句都执行完，即程序会执行到语句return b; 并将函数结束并返回值。



```
#include <iostream>
using namespace std;
int print_bigger (int a, int b)
//打印两数中更大的那个数
{
    if (a >b)
        return a;
    //else 去掉else句 结果一样; 但是加上程序可读性强
        return b;
}
int main()
{
    int a,b;
    cin >> a >>b;
    cout << print_bigger (a,b);
    return 0;
}
```



4.1 函数的基本使用

4.1.3 返回值

函数的返回值可以是表达式

例：



```
#include <iostream>
using namespace std;

double cal(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe
}

double cal(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    return be * 0.1 + midG * 0.3 + finG * 0.6;
}

//二种方式都可以
```



4.1 函数的基本使用

4.1.4 调用

★ 使用一个函数的方法是调用这个函数，在调用时需要提供该函数的输入参数，并在该函数运行结束时接收函数的返回值。

★ 函数调用时，需要给出函数定义中的所有输入参数，且输入参数与函数定义中输入参数的**数目一致（无默认值情况）、数据类型相同**。不匹配的输入参数会导致**隐式数据类型转换或是BUG**的出现。

★ 一个函数只有在**被声明过后**，才可以被调用，且对于每个函数声明，在程序中都**必须给出**其对应的**函数实现**。



4.1 函数的基本使用

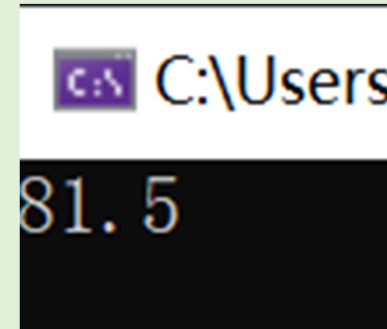
4.1.4 调用

函数调用时，需要给出函数定义中的所有输入参数，且输入参数与函数定义中输入参数的**数目一致（无默认值情况）**、**数据类型相同**。不匹配的输入参数会导致**隐式数据类型转换或是BUG**的出现。

正例：



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5, result;
    double b = 75.5, c = 84.5;
    result = calGrade1(a, b, c); // 参数的数目一致、数据类型相同
    cout << result;
    return 0;
}
```





4.1 函数的基本使用

4.1.4 调用

函数调用时，需要给出函数定义中的所有输入参数，且输入参数与函数定义中输入参数的**数目一致（无默认值情况）**、**数据类型相同**。不匹配的输入参数会导致**隐式数据类型转换或是BUG**的出现。

反例：



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b);
    cout << result;
    return 0;
}
```



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b);
    cout << result;
    return 0;
}
```

辨析:

此时程序在运行main函数时，在调用的calGrade1函数时，输入参数缺失，出现错误



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
```

输出

显示输出来源(S): 生成

1>----- 已启动生成: 项目: test, 配置: Debug Win32 -----

1>example7.cpp

1>C:\Users\kevin\Desktop\助教\program4.1\test\example7.cpp(18,36): error C2660: “calculateGrade1”: 函数不接受 2 个参数

1>C:\Users\kevin\Desktop\助教\program4.1\test\example7.cpp(5,8): message : 参见“calculateGrade1”的声明

```
    return gne;
}

int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b);
    cout << result;
    return 0;
}
```

辨析:

此时程序在运行main函数时，在调用的calGrade1函数时，输入参数缺失，出现错误



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1('a', b, c);
    cout << result;
    return 0;
}
```



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1('a', b, c);
    cout << result;
    return 0;
}
```

A screenshot of a terminal window showing the output of the program. The path 'C:\Users\' is visible, and the result '83.05' is displayed in a large font.



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1('a', b, c);
    cout << result;
    return 0;
}
```

辨析:

产生了隐式的类型转换, 将字符 'a' 转换为它的ASCII码值即97, $97 * 0.1 + 75.5 * 0.3 + 84.5 * 0.6 = 83.05$

注意到这个错误在编译运行时不会发出警告, 因为发生了符合语法的隐式的类型转换



4.1 函数的基本使用

4.1.4 调用

★ 使用一个函数的方法是调用这个函数，在调用时需要提供该函数的输入参数，并在该函数运行结束时接收函数的返回值。

★ 函数调用时，需要给出函数定义中的所有输入参数，且输入参数与函数定义中输入参数的**数目一致（无默认值情况）**、**数据类型相同**。不匹配的输入参数会导致**隐式数据类型转换或是BUG**的出现。

★ 一个函数只有在**被声明过后**，才可以被调用，且对于每个函数声明，在程序中都**必须给出**其对应的**函数实现**。



4.1 函数的基本使用

4.1.4 调用

函数声明:

指明函数的名字, 参数和返回值类型, 但不写出函数的主体语句部分, 又叫**引用说明**、**函数原型**。(告知编译器函数名称及如何调用函数)

```
returnValueType functionName (parameterlist )
```

函数实现:

给出函数完整的结构和所有语句, **包含函数头和函数的具体实现（函数定义）**两部分。

```
returnValueType  functionName (parameterlist )  
{  
    statements;  
    ...  
    return value;  
}
```



4.1 函数的基本使用

4.1.4 调用

- 函数声明注意事项:

- 函数声明中的变量名可以与函数实现中的相同
- 函数声明中的变量名也可以不同于函数实现
- 函数声明中也可以不写变量名
- 函数声明一定不能缺失的部分: 函数名, 输入参数的数据类型, 返回值的数据类型



```
#include<iostream>
using namespace std;
double rate_minus1(double item1, double item2); //函数声明
double rate_minus2(double a, double b); //函数声明
double rate_minus3(double, double); //函数声明
int main()
{
    double a, b, result1, result2, result3;
    cin >> a >> b;
    result1 = rate_minus1(a, b); // 函数调用
    result2 = rate_minus2(a, b); // 函数调用
    result3 = rate_minus3(a, b); // 函数调用
    cout << result1 << endl;
    cout << result2 << endl;
    cout << result3 << endl;
    return 0;
}
```

//函数实现部分见后页



```
double rate_minus1(double item1, double item2)
{
    double result;
    result = item1 * 0.7 - item2 * 0.6;
    return result;
}    // 函数实现

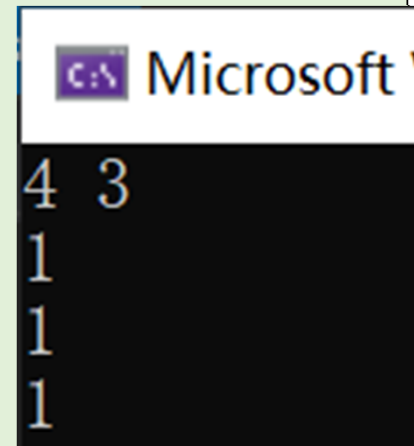
double rate_minus2(double item1, double item2)
{
    double result;
    result = item1 * 0.7 - item2 * 0.6;
    return result;
}    //函数实现

double rate_minus3(double item1, double item2)
{
    double result;
    result = item1 * 0.7 - item2 * 0.6;
    return result;
}    //函数实现
```




```
#include<iostream>
using namespace std;
double rate_minus1(double item1, double item2); //函数声明
double rate_minus2(double a, double b); //函数声明
double rate_minus3(double, double); //函数声明
int main()
{
    double a, b, result1, result2, result3;
    cin >> a >> b;
    result1 = rate_minus1(a, b); // 函数调用
    result2 = rate_minus2(a, b); // 函数调用
    result3 = rate_minus3(a, b); // 函数调用
    cout << result1 << endl;
    cout << result2 << endl;
    cout << result3 << endl;
    return 0;
}
```

//函数实现部分见后页





```
#include<iostream>
using namespace std;
double rate_minus1(double item1, double item2); //函数声明
double rate_minus2(double a, double b); //函数声明
double rate_minus3(double, double); //函数声明
int main()
{
    double a, b, result1, result2, result3;
    cin >> a >> b;
    result1 = rate_minus1(a, b); //
    result2 = rate_minus2(a, b); //
    result3 = rate_minus3(a, b); //
    cout << result1 << endl;
    cout << result2 << endl;
    cout << result3 << endl;
    return 0;
}
```

辨析:

函数的声明一定不能缺失的部分:
函数名, 输入参数的数据类型,
返回值的数据类型

//函数实现部分见后页



4.1 函数的基本使用

4.1.4 调用

一个函数的声明和实现可以写在一起，也可以分开写在程序段的
不同位置。



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)

// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}

int main()
{
    double a = 81.5, result;
    double b = 75.5, c = 84.5;
    result = calGrade1(a, b, c); // 输入参数的数据类型相同
    cout << result;
    return 0;
}
```



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG)
//函数的声明和实现写在一起
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
int main()
{
    double a = 81.5, result;
    double b = 75.5, c = 84.5;
    result = calGrade1(a, b, c); //输入参数的数据类型相同
    cout << result;
    return 0;
}
```

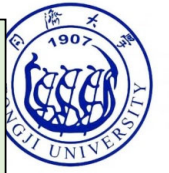


4.1 函数的基本使用

4.1.4 调用

如果在上例程序中，将calGrade1函数放在main函数之后，调用函数：
`result = calGrade1('a', b, c);`

会出现什么问题？



```
#include<iostream>
using namespace std;
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b, c);
    cout << result;
    return 0;
}

double calGrade1(double be, double midG, double finG)
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
```

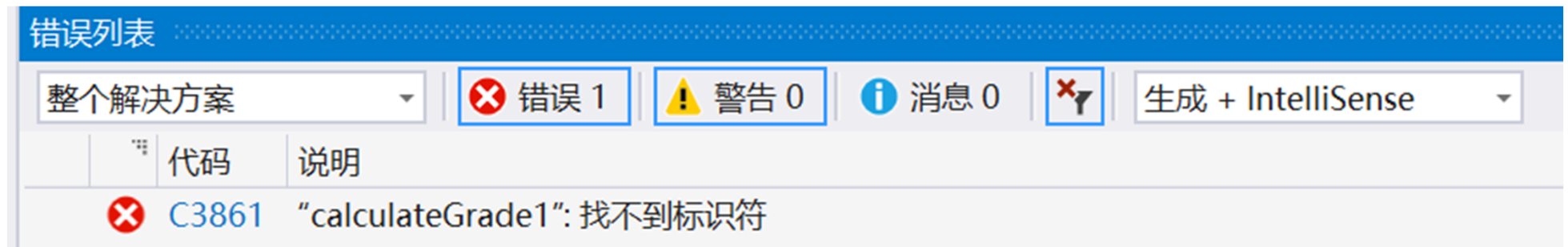
如果在上例程序中，将calGrade1函数放在main函数之后，调用函数：`result = calGrade1(a, b, c);`会出现什么问题？



4.1 函数的基本使用

4.1.4 调用

如果在上例程序中，将calGrade1函数放在main函数之后
调用函数： `result = calGrade1(a, b, c);`
会出现什么问题？



- 此时程序在运行main函数时，无法找到所调用的calGrade1函数，也就是函数调用之前没有被声明，导致出现错误，解决的方法是在main函数之前声明calGrade1函数



4.1 函数的基本使用

4.1.4 调用

解决方案：在调用main函数调用之前，加入函数的声明语句

正例：



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG); //函数声明
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b, c);
    cout << result;
    return 0;
}
double calGrade1(double be, double midG, double finG) //函数实现
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
```



4.1 函数的基本使用



4.1.4 调用 常见错误:



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG) //函数声明
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b, c);
    cout << result;
    return 0;
}
double calGrade1(double be, double midG, double finG) //函数实现
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
```



```
#include<iostream>
using namespace std;
double calGradel(double be, double midG, double finG) //函数声明
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGradel(a, b, c);
    cout << result;
    return 0;
}
do
//
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
```

	代码	说明	项目
do	 E0130	应输入“{”	test
//	 C2144	语法错误:“int”的前面应有“;”	test



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG) //函数声明
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b, c);
    cout << result;
    return 0;
}
double calGrade1(double be, double mi
// 加权计算一名学生的总成绩
{
    double gRe;
    gRe = be * 0.1 + midG * 0.3 + finG * 0.6;
    return gRe;
}
```

辨析:

函数声明是一条语句，因此必须以分号结束。



4.1 函数的基本使用

4.1.4 调用

疑问：可以只写函数的声明语句，而不实现么？

反例：



```
#include<iostream>
using namespace std;
double calGrade1(double be, double midG, double finG);
int main()
{
    double a = 81.5;
    double b = 75.5, c = 84.5, result;
    result = calGrade1(a, b, c);
    cout << result;
    return 0;
}
```




4.1 函数的基本使用

4.1.4 调用

错误列表

整个解决方案 ▾ | 错误 2 | 警告 0 | 消息 0 | | 生成 + IntelliSense ▾

代码	说明
LNK2019	无法解析的外部符号 "double __cdecl calculateGrade(double,double,double)" (?calculateGrade@@YANNNN@Z), 该符号在函数 "int __cdecl example3(void)" (?example3@@YAHXZ) 中被引用

- 此时程序在运行main函数时，无法找到所调用的calGrade1函数的函数实现，出现错误，解决的方法是在main函数之后实现calGrade1函数

推荐的顺序结构：函数声明 + main函数 + 函数实现



4.1 函数的基本使用

4.1.4 调用

函数声明存在的意义

使用函数声明一方面可以帮助编译器快速的确定所调用的函数原型，避免额外花费时间用于搜索程序文件中的对应函数；另一方面也可以帮助编程者在编写大量的函数时快速的区分和查找不同的函数，降低BUG出现的概率。



§2.基础知识

两种方法验证



2.6.常量

2.6.3.字符常量与字符串常量

2.6.3.4.字符串常量

在内存中的存放：每个字符的ASCII码+字符串结束标志'\0'
(ASCII 0、尾0)

★ ""与" "的区别

"" - 空字符串，长度为0

" " - 含一个空格的字符串

没有函数声明和实现?

```
#include <iostream>
using namespace std;
int main()
{
    cout << sizeof("") << endl;
    cout << sizeof(" ") << endl;
    cout << strlen("") << endl;
    cout << strlen(" ") << endl;
    return 0;
}
```

\0

32

\0



4.1 函数的基本使用

4.1.5 库函数与自定义函数

- 由C/C++语言(编译器) **预先定义好的**函数组成的库，这些库通过 **#include语句**被添加到程序中，编程者不用定义或声明，可以 **直接调用**库函数。
- 与库函数相对应的，编程者自己编写定义的函数称为 **用户自定义函数**。编程者也可以进一步将自己编写的函数组成一个 **自定义库**，供其他编程者或是程序使用。（具体内容见之后的课程）



```
#include <cmath>
#include<iostream>
using namespace std;
int user_abs(int item)
{
    int result;
    if (item < 0)
        result = item * (-1);
    else
        result = item;
    return result;
}

int main()
{
    int result1,result2, a;
    cin >> a;
    result1 = user_abs(a);
    result2 = abs(a);
    cout << result1 << "\n";
    cout << result2 << "\n";
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
-1
1
1
```



```
#include <cmath>
#include<iostream>
using namespace std;
int user_abs(int item)
// 自定义函数
{
    int result;
    if (item < 0)
        result = item * (-1);
    else
        result = item;
    return result;
}
```

```
int main()
{
    int result1,result2, a;
    cin >> a;
    result1 = user_abs(a);
    result2 = abs(a); //库函数
    cout << result1 << "\n";
    cout << result2 << "\n";
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
-1
1
1
```



```
#include <cmath>
#include<iostream>
using namespace std;
int user_abs(int item)
// 自定义函数
{
    int result;
    if (item < 0)
        result = item * (-1);
    else
        result = item;
    return result;
}
```

```
int main()
{
    int result1,result2, a;
    cin >> a;
    result1 = user_abs(a);
    result2 = abs(a); //库函数
    cout << result1 << "\n";
    cout << result2 << "\n";
}
```

辨析:

user_abs为用户自定义函数;

abs为库函数, 包含这个函数的库通过
#include <cmath> 语句被添加到程序中,
编程者不用定义或声明, 可以直接调用

Microsoft Visual Studio

```
-1
1
1
```



4.1 函数的基本使用

4.1.5 库函数与自定义函数

★ “math.h”为C语言的数学计算函数库头文件，其在C++环境下也可以使用，但在使用C++编程时，更推荐使用C++的数学计算函数库头文件<cmath>（参看教材P17页）

★ 需要注意的是，很多库函数在多个函数库头文件中都有被定义，此时需要根据文档和程序需求选择使用哪个函数库头文件



4.1 函数的基本使用

4.1.5 库函数与自定义函数

常用库函数举例（函数库<cmath>）：

- `extern float pow(float x, float y);` // 计算x的y次方

`pow(10, 2):` 100

- `extern float sqrt(float x);` // 计算x的开方结果

`sqrt(100):` 10

- `extern float log(float x);` // 计算x的自然对数

`log(20):` 2.99573



4.1 函数的基本使用

4.1.5 库函数与自定义函数

常用库函数举例（函数库`<string.h>`,`<iostream>`）:

- `extern int strlen(const char *str)`//计算字符串 `str` 的长度
`strlen("Hello!")` : 6



4.1 函数的基本使用

4.1.5 库函数与自定义函数

- C/C++语言(编译器)拥有众多的库函数和不同类型的函数库, 这些库函数涉及输入输出、字符串处理、数学计算、日期处理、内存分配、对象类等多个方面, 具体内容**参考文档**
<https://www.runoob.com/cprogramming/c-standard-library.html> 和 <https://www.runoob.com/cplusplus/cpp-standard-library.html>。
- 相比于自定义函数, C/C++语言的库函数**更加高效和规范化**, 可以帮助编程者提升程序的运行效率并减少程序中可能出现的BUG。



总结

基本概念 函数定义/使用函数的作用/定义格式

参数 无输入参数/输入参数类型匹配/形参实参/单向传值

返回值 无返回值/返回值类型匹配/多返回语句/返回表达式

调用 参数匹配/声明与实现/声明的位置/声明意义

库函数与自定义函数 库函数用法/自定义函数