

01 - 函数 I

C++ 程序设计进阶

SOJ 信息学竞赛教练组

2024 年 7 月 21 日

目录

1 引入函数

2 函数的定义与调用

3 函数的参数

4 函数的返回值

5 函数调用时的执行过程

6 常用的函数

7 总结

什么是函数？

- 数学中的函数：一个变量 y 随另一个变量 x 的变化而变化
 - 例如： $y = 2x + 1$ 中 x 为自变量， y 为因变量

- 数学中的函数：一个变量 y 随另一个变量 x 的变化而变化
 - 例如： $y = 2x + 1$ 中 x 为自变量， y 为因变量
- 编程中的函数：一系列代码，用于实现某个特定的功能
 - 接收给定的自变量，通过该函数处理，得到特定的结果（因变量或效果）

目录

1 引入函数

2 函数的定义与调用

3 函数的参数

4 函数的返回值

5 函数调用时的执行过程

6 常用的函数

7 总结

函数的定义

- 函数的定义形式如下：

```
1 void 函数名() {  
2     函数体  
3 }
```

函数的定义

- 函数的定义形式如下：

```
1 void 函数名() {  
2     函数体  
3 }
```

- 函数的定义需要写在 main 函数外面

函数的定义

- 函数的定义形式如下：

```
1 void 函数名() {  
2     函数体  
3 }
```

- 函数的定义需要写在 main 函数外面
- 确定函数的**功能**，把功能的实现写在函数体中

函数的定义

- 函数的定义形式如下：

```
1 void 函数名() {  
2     函数体  
3 }
```

- 函数的定义需要写在 main 函数外面
- 确定函数的**功能**，把功能的实现写在函数体中

```
1 void print() {  
2     cout << "hello world" << endl;  
3 }
```

例 1.1：输出一行 hello world

编程题

- 请定义一个 `print` 函数，功能：输出一行 `hello world`。
在主函数调用该函数，实现输出。
- 样例输入
无
- 样例输出
`hello world`

- 函数名的命名规则与变量名的命名规则相同

函数的实现

- 函数名的命名规则与变量名的命名规则相同
- 函数名后面的小括号不能省略

函数的实现

- 函数名的命名规则与变量名的命名规则相同
- 函数名后面的小括号不能省略
- 函数体是函数功能的具体实现：输出一行 hello world

```
1 void print() {  
2     cout << "hello world" << endl;  
3 }
```

- 定义函数后，需要用调用语句来执行函数

函数的调用

- 定义函数后，需要用调用语句来执行函数
- 函数的调用语句为：**函数名 ()**

函数的调用

- 定义函数后，需要用调用语句来执行函数
- 函数的调用语句为：**函数名 ()**
 - 调用函数时不能省略小括号

函数的调用

- 定义函数后，需要用调用语句来执行函数
- 函数的调用语句为：**函数名 ()**
 - 调用函数时不能省略小括号
- 函数体中可以调用任何前面已经定义的函数

函数的调用

- 定义函数后，需要用调用语句来执行函数
- 函数的调用语句为：**函数名 ()**
 - 调用函数时不能省略小括号
- 函数体中可以调用任何前面已经定义的函数
 - `print` 函数定义在 `main` 函数之前，故可在 `main` 函数中调用

函数的调用

- 定义函数后，需要用调用语句来执行函数
- 函数的调用语句为：**函数名 ()**
 - 调用函数时不能省略小括号
- 函数体中可以调用任何前面已经定义的函数
 - `print` 函数定义在 `main` 函数之前，故可在 `main` 函数中调用
 - 特殊地，在程序启动时会自动调用 `main` 函数

例 1.1：输出一行 hello world

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print() {
6     cout << "hello world" << endl;
7 }
8
9 int main() {
10     print();
11     return 0;
12 }
```

如何让函数输出 n 行 hello world 呢?

函数的多次调用

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print() {
6     cout << "hello world" << endl;
7 }
8
9 int main() {
10     int n;
11     cin >> n;
12     for (int i = 1; i <= n; i++) {
13         print();
14     }
15     return 0;
16 }
```

函数的多次调用

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print() {
6     cout << "hello world" << endl;
7 }
8
9 int main() {
10     int n;
11     cin >> n;
12     for (int i = 1; i <= n; i++) {
13         print();
14     }
15     return 0;
16 }
```

- 一次函数调用能否输出多行？

目录

1 引入函数

2 函数的定义与调用

3 函数的参数

4 函数的返回值

5 函数调用时的执行过程

6 常用的函数

7 总结

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 void 函数名(参数列表) {  
2     函数体  
3 }
```

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 void 函数名(参数列表) {  
2     函数体  
3 }
```

- 参数列表**相当于“给定的自变量”，接收实现函数功能所需的数据

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 void 函数名(参数列表) {  
2     函数体  
3 }
```

- 参数列表**相当于“给定的自变量”，接收实现函数功能所需的数据
- 之前写的函数都是参数列表为空的情况

参数列表

- 参数列表可以为空
- 参数列表可以不为空
 - 要说明每个参数的**参数类型**、**参数名**
 - 参数类型可以是 `int`, `long long`, `double`, `char`, ...

例 1.2：输出 5 行 hello world

编程题

- 请定义一个 `print` 函数，功能：输出 n 行 `hello world`。
在主函数调用该函数，实现输出 5 行 `hello world`。
- 样例输入
无
- 样例输出
`hello world`
`hello world`
`hello world`
`hello world`
`hello world`

有参数的函数实现

- 函数体是函数功能的具体实现：输出 n 行 hello world

有参数的函数实现

- 函数体是函数功能的具体实现：输出 n 行 hello world
- 其中， n 是实现函数功能所需接收的数据，故为函数的参数，其类型为整数

```
1 void print(int n) {  
2     for (int i = 1; i <= n; i++) {  
3         cout << "hello world" << endl;  
4     }  
5 }
```


有参数函数的调用

- 有参数函数的调用语句为：**函数名 (参数)**

有参数函数的调用

- 有参数函数的调用语句为：**函数名 (参数)**
- 调用有参数的函数时，参数应为具体的数值或变量

有参数函数的调用

- 有参数函数的调用语句为：**函数名 (参数)**
- 调用有参数的函数时，参数应为具体的数值或变量
 - `print(5);` 表示调用函数，输出 5 行 hello world

有参数函数的调用

- 有参数函数的调用语句为：**函数名 (参数)**
- 调用有参数的函数时，参数应为具体的数值或变量
 - `print(5);` 表示调用函数，输出 5 行 hello world
 - `int x = 10;`
`print(x);` 表示调用函数，输出 $x(10)$ 行 hello world

例 1.2: 输出 5 行 hello world

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print(int n) {
6     for (int i = 1; i <= n; i++) {
7         cout << "hello world" << endl;
8     }
9 }
10
11 int main() {
12     print(5);
13     return 0;
14 }
```

找错题

1. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 void print() {
4     cout << "hello world" << endl;
5 }
6
7 print();
8 int main() {
9     return 0;
10 }
```

找错题

1. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 void print() {
4     cout << "hello world" << endl;
5 }
6
7 print(); 该语句不会被执行
8 int main() {
9     return 0;
10 }
```

找错题

2. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 void print() {
4     cout << "hello world" << endl;
5 }
6
7 int main() {
8     print;
9     return 0;
10 }
```


找错题

2. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 void print() {
4     cout << "hello world" << endl;
5 }
6
7 int main() {
8     print; 不能省略小括号
9     return 0;
10 }
```

找错题

3. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 void print(int n) {
4     for (int i = 1; i <= n; i++) {
5         cout << "hello world" << endl;
6     }
7 }
8
9 int main() {
10     print();
11     return 0;
12 }
```

找错题

3. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 void print(int n) {
4     for (int i = 1; i <= n; i++) {
5         cout << "hello world" << endl;
6     }
7 }
8
9 int main() {
10     print(); 需要传参
11     return 0;
12 }
```

如果函数需要返回某个结果，该如何实现呢？

目录

- 1 引入函数
- 2 函数的定义与调用
- 3 函数的参数
- 4 函数的返回值**
- 5 函数调用时的执行过程
- 6 常用的函数
- 7 总结

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 没有返回值时，其返回值类型为 `void`，这样的函数称为“无返回值的函数”

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 没有返回值时，其返回值类型为 `void`，这样的函数称为“无返回值的函数”
- 返回值类型为其他数据类型时，这样的函数称为“有返回值的函数”

函数的进阶写法

- 函数的定义形式可以调整为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 没有返回值时，其返回值类型为 `void`，这样的函数称为“无返回值的函数”
- 返回值类型为其他数据类型时，这样的函数称为“有返回值的函数”
- 之前写的函数都是无返回值的函数

- 返回值相当于函数执行后得到的“**因变量（结果）**”
- 返回值可以为空
- 返回值可以不为空
 - 根据结果的类型确定返回值类型
 - 返回值类型可以是 `int`, `long long`, `double`, `char`, ...
 - 返回值由 `return` 语句返回

例 1.3: 输出从 1 到 n 的整数之和

编程题

- 请定义一个 `getSum` 函数，功能：计算 $1 \sim n$ 的和。
输入一个正整数 $n(1 \leq n \leq 1000)$ ，输出 $1 \sim n$ 的和。
- 样例输入
100
- 样例输出
5050

有返回值函数的实现

- 函数体是函数功能的具体实现：计算 $1 \sim n$ 的和

有返回值函数的实现

- 函数体是函数功能的具体实现：计算 $1 \sim n$ 的和
- 其中， n 是实现函数功能所需接收的数据，故为函数的参数，其类型为整数

有返回值函数的实现

- 函数体是函数功能的具体实现：计算 $1 \sim n$ 的和
- 其中， n 是实现函数功能所需接收的数据，故为函数的参数，其类型为整数
- 其中，“和”是函数执行后需要得到的结果，故为函数的返回值，其类型为整数，使用 `return` 语句返回

```
1 int getSum(int n) {  
2     int sum = 0;  
3     for (int i = 1; i <= n; i++) {  
4         sum += i;  
5     }  
6     return sum;  
7 }
```

有返回值函数的调用

- 有返回值函数的调用语句与无返回值函数的调用语句一致

有返回值函数的调用

- 有返回值函数的调用语句与无返回值函数的调用语句一致
- 有返回值函数调用时，可将其视作具体的数值进行运算或输出

有返回值函数的调用

- 有返回值函数的调用语句与无返回值函数的调用语句一致
- 有返回值函数调用时，可将其视作具体的数值进行运算或输出
- `int sum = getSum(5);` 表示调用函数，计算 1 ~ 5 的和，并将和存入 `sum` 中

有返回值函数的调用

- 有返回值函数的调用语句与无返回值函数的调用语句一致
- 有返回值函数调用时，可将其视作具体的数值进行运算或输出
 - `int sum = getSum(5);` 表示调用函数，计算 1 ~ 5 的和，并将和存入 `sum` 中
 - `cout << getSum(5) << endl;` 表示调用函数，计算 1 ~ 5 的和，并将和输出

例 1.3：输出从 1 到 n 的整数之和

```
1 #include <iostream>
2
3 using namespace std;
4
5 int getSum(int n) {
6     int sum = 0;
7     for (int i = 1; i <= n; i++) {
8         sum += i;
9     }
10    return sum;
11 }
12
13 int main() {
14     int n;
15     cin >> n;
16     cout << getSum(n) << endl;
17     return 0;
18 }
```

找错题

1. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     getSum(4);
13     return 0;
14 }
```

找错题

1. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     getSum(4);
13     return 0;
14 }
```

运行结果无作用，程序无输出

找错题

2. 阅读程序，找出代码中的错误

```
1 // #include ...
2
3 int getSum(int n) {
4     long long sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     cout << getSum(1000000) << endl;
13     return 0;
14 }
```

找错题

2. 阅读程序，找出代码中的错误

```
1 // #include ...
2 类型不正确，类型转换后数据溢出
3 int getSum(int n) {
4     long long sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     cout << getSum(1000000) << endl;
13     return 0;
14 }
```

小结：函数的定义与调用

- 函数的完整框架为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```


小结：函数的定义与调用

- 函数的完整框架为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 函数是否有参数，取决于实现函数功能时是否需要接收数据

小结：函数的定义与调用

- 函数的完整框架为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 函数是否有参数，取决于实现函数功能时是否需要接收数据
- 有参数的函数，调用时需要传入具体的数值

小结：函数的定义与调用

- 函数的完整框架为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 函数是否有参数，取决于实现函数功能时是否需要接收数据
- 有参数的函数，调用时需要传入具体的数值
- 函数是否有返回值，取决于函数是否需要得到一个计算结果

小结：函数的定义与调用

- 函数的完整框架为：

```
1 返回值类型 函数名(参数列表) {  
2    函数体  
3    return 返回值; //无返回值时留空  
4 }
```

- 函数是否有参数，取决于实现函数功能时是否需要接收数据
- 有参数的函数，调用时需要传入具体的数值
- 函数是否有返回值，取决于函数是否需要得到一个计算结果
- 有返回值的函数，调用时需要注意处理返回的结果，如进行运算或输出

目录

- 1 引入函数
- 2 函数的定义与调用
- 3 函数的参数
- 4 函数的返回值
- 5 函数调用时的执行过程**
- 6 常用的函数
- 7 总结

函数的调用

- 函数体中可以调用任何前面已定义的函数
- 当执行到函数的调用语句时，程序将跳转到该函数中，执行其函数体的语句，直到执行完函数体或执行到 `return` 语句。待函数执行完，程序会跳转回原调用处继续往下执行。

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
- 输出

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
- 输出

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
主函数执行暂停，等待 function1 函数执行
- 输出

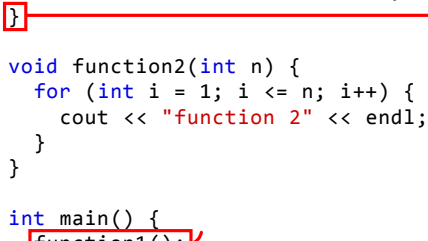
示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
主函数执行暂停，等待 function1 函数执行
- 输出
function 1

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```



- 执行过程
function1 函数执行完毕，回到主函数继续执行
- 输出
function 1

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
function1 函数执行完毕，回到主函数继续执行
- 输出
function 1

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
主函数执行暂停，把 3 赋值给 n ，等待 function2 函数执行
- 输出
function 1

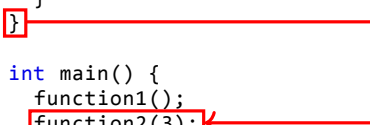
示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
主函数执行暂停，把 3 赋值给 n ，等待 function2 函数执行
- 输出
function 1
function 2
function 2
function 2

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

A red box highlights the closing brace of function2 in line 11. A red arrow points from this box to the function2(3) call in line 15 of the main function. Another red box highlights the function2(3) call in line 15.

- 执行过程
function2 函数执行完毕，回到主函数继续执行
- 输出
function 1
function 2
function 2
function 2

示例：无返回值函数的执行过程

```
1 // #include ...
2
3 void function1() {
4     cout << "function 1" << endl;
5 }
6
7 void function2(int n) {
8     for (int i = 1; i <= n; i++) {
9         cout << "function 2" << endl;
10    }
11 }
12
13 int main() {
14     function1();
15     function2(3);
16     return 0;
17 }
```

- 执行过程
程序结束
- 输出
function 1
function 2
function 2
function 2

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
- 输出

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
- 输出

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
输入 x 的值，值为 3
- 输出

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
主函数执行暂停，把 x 赋值给 n ，等待 getSum 函数执行
- 输出

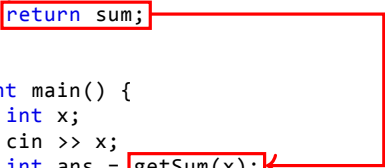
示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
计算得到 $sum = 6$
- 输出

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```



A red rectangular box highlights the `return sum;` statement on line 8 of the `getSum` function. A red line extends from this box to the right, then turns downwards and then left, ending with an arrow pointing to the `getSum(x);` call on line 14 of the `main` function. Both the `return sum;` statement and the `getSum(x);` call are also enclosed in red rectangular boxes.

- 输入
3
- 执行过程
返回值为 6，回到主函数继续执行
- 输出

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
把 6 赋值给 *ans*
- 输出

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
输出 *ans*
- 输出
6

示例：有返回值函数的执行过程

```
1 // #include ...
2
3 int getSum(int n) {
4     int sum = 0;
5     for (int i = 1; i <= n; i++) {
6         sum += i;
7     }
8     return sum;
9 }
10
11 int main() {
12     int x;
13     cin >> x;
14     int ans = getSum(x);
15     cout << ans << endl;
16     return 0;
17 }
```

- 输入
3
- 执行过程
程序结束
- 输出
6

- 调用函数需要使用函数名，并把实际使用的参数放在括号中传递进去
- **形式参数**：函数定义时在括号里定义的变量，它只作为**指代**
- **实际参数**：调用函数时传给形参的**具体数值**

示例：有返回值函数的执行过程

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print(int n) {
6     for (int i = 1; i <= n; i++) {
7         cout << "hello world" << endl;
8     }
9 }
10
11 int main() {
12     print(5);
13     return 0;
14 }
```

示例：有返回值函数的执行过程

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print(int n) { ←
6     for (int i = 1; i <= n; i++) {
7         cout << "hello world" << endl;
8     }
9 }
10
11 int main() {
12     print(5);
13     return 0;
14 }
```

示例：有返回值函数的执行过程

```
1 #include <iostream>
2
3 using namespace std;
4
5 void print(int n) {
6     for (int i = 1; i <= n; i++) {
7         cout << "hello world" << endl;
8     }
9 }
10
11 int main() {
12     print(5);
13     return 0;
14 }
```

形式参数

n = 5

实际参数

参数列表

- 函数的参数列表也可以有多个参数
- **每个参数都要说明其数据类型和参数名**，不能省略
- **参数之间使用逗号**隔开
- 调用函数时，函数的**实际参数**需要和**形式参数的顺序一致**

例 1.4：求梯形的面积

编程题

- 已知梯形的面积公式为 $S = \frac{(a+b)h}{2}$ ，其中 a, b, h 分别为上底、下底、高。
请定义一个 `ladder_shaped` 函数，功能：用梯形的上底、下底和高，计算梯形的面积。
输入三个实数，表示梯形的上底、下底、高，输出梯形的面积，输出保留两位小数。
- 样例输入
1.5 2.5 3
- 样例输出
6.00

例 1.4：求梯形的面积

```
1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5
6 double ladder_shaped(double a, double b, double h) {
7     return (a + b) * h / 2;
8 }
9
10 int main() {
11     double x, y, z;
12     cin >> x >> y >> z;
13     double size = ladder_shaped(x, y, z);
14     cout << fixed << setprecision(2) << size << endl;
15     return 0;
16 }
```


例 1.4：求梯形的面积

```
1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5
6 double ladder_shaped(double a, double b, double h) {
7     return (a + b) * h / 2;
8 }
9
10 int main() {
11     double x, y, z;
12     cin >> x >> y >> z;
13     double size = ladder_shaped(x, y, z);
14     cout << fixed << setprecision(2) << size << endl;
15     return 0;
16 }
```

a = x
b = y
h = z

- 函数的定义和调用：有无参数、有无返回值
- 函数用于实现某个特定功能，可以理解为将“功能”包装到函数中，方便使用

目录

- 1 引入函数
- 2 函数的定义与调用
- 3 函数的参数
- 4 函数的返回值
- 5 函数调用时的执行过程
- 6 常用的函数**
- 7 总结

常用的函数

函数名	功能	参数类型	返回值类型	头文件
<code>sqrt(a)</code>	平方根	浮点数	浮点数	<code>cmath</code>
<code>pow(a, n)</code>	幂	浮点数	浮点数	<code>cmath</code>
<code>abs(a)</code>	绝对值	整数或浮点数	同参数类型	<code>algorithm</code>
<code>max(a, b)</code>	最大值	同类型参数	同参数类型	<code>algorithm</code>
<code>min(a, b)</code>	最小值	同类型参数	同参数类型	<code>algorithm</code>
<code>swap(a, b)</code>	交换变量	同类型参数	无	<code>utility</code>

- 示例

常用的函数

函数名	功能	参数类型	返回值类型	头文件
<code>sqrt(a)</code>	平方根	浮点数	浮点数	<code>cmath</code>
<code>pow(a, n)</code>	幂	浮点数	浮点数	<code>cmath</code>
<code>abs(a)</code>	绝对值	整数或浮点数	同参数类型	<code>algorithm</code>
<code>max(a, b)</code>	最大值	同类型参数	同参数类型	<code>algorithm</code>
<code>min(a, b)</code>	最小值	同类型参数	同参数类型	<code>algorithm</code>
<code>swap(a, b)</code>	交换变量	同类型参数	无	<code>utility</code>

- 示例
 - 可用 `sqrt(2)` 求 $\sqrt{2}$

常用的函数

函数名	功能	参数类型	返回值类型	头文件
<code>sqrt(a)</code>	平方根	浮点数	浮点数	<code>cmath</code>
<code>pow(a, n)</code>	幂	浮点数	浮点数	<code>cmath</code>
<code>abs(a)</code>	绝对值	整数或浮点数	同参数类型	<code>algorithm</code>
<code>max(a, b)</code>	最大值	同类型参数	同参数类型	<code>algorithm</code>
<code>min(a, b)</code>	最小值	同类型参数	同参数类型	<code>algorithm</code>
<code>swap(a, b)</code>	交换变量	同类型参数	无	<code>utility</code>

- 示例

- 可用 `sqrt(2)` 求 $\sqrt{2}$
- 可用 `pow(2, 5)` 求 2^5

常用的函数

函数名	功能	参数类型	返回值类型	头文件
<code>sqrt(a)</code>	平方根	浮点数	浮点数	<code>cmath</code>
<code>pow(a, n)</code>	幂	浮点数	浮点数	<code>cmath</code>
<code>abs(a)</code>	绝对值	整数或浮点数	同参数类型	<code>algorithm</code>
<code>max(a, b)</code>	最大值	同类型参数	同参数类型	<code>algorithm</code>
<code>min(a, b)</code>	最小值	同类型参数	同参数类型	<code>algorithm</code>
<code>swap(a, b)</code>	交换变量	同类型参数	无	<code>utility</code>

• 示例

- 可用 `sqrt(2)` 求 $\sqrt{2}$
- 可用 `pow(2, 5)` 求 2^5
- 可用 `abs(-2)` 求 $|-2|$

目录

- 1 引入函数
- 2 函数的定义与调用
- 3 函数的参数
- 4 函数的返回值
- 5 函数调用时的执行过程
- 6 常用的函数
- 7 总结**

- 函数的定义
 - 返回值类型、函数名、参数列表、函数体

- 函数的定义
 - 返回值类型、函数名、参数列表、函数体
- 函数的调用
 - 函数调用方式与跳转过程
 - 形参、实参
 - 多参数、有返回值函数调用注意要点

- 函数的定义
 - 返回值类型、函数名、参数列表、函数体
- 函数的调用
 - 函数调用方式与跳转过程
 - 形参、实参
 - 多参数、有返回值函数调用注意要点
- 常用的 C++ 库函数

Thank you!