# 目录

# Part I

# C++ 入门

# Chapter 1

# 编写第一个程序

## 1.1 编译和链接源代码

对于你创建的 C++ 源代码文件，可使用扩展名 .cpp、.cxx、.cp 或 .c。大多数 C++ 都不关心源代码文件的扩展名，但使用 .cpp 有助于你识别源代码文件。

源代码是供人类阅读的 C++ 程序，必须经过编译和链接才能运行。

编译源代码时，将生成一个目标文件，链接器将把它转换为可执行的程序。

创建 C++ 程序时，将链接一个或多个目标文件以及一个或多个库。库是一系列可链接的文件，提供了有用的函数和类，可供你在程序中使用。

创建 C++ 程序的步骤如下。

1. 使用文本编辑器创建源代码。
2. 使用编译器将源代码转换为目标文件。
3. 使用链接器链接目标文件和必要的库，生成可执行的程序。
4. 输入可执行文件的名称以运行它。

GCC 编译器将编译和链接合而为一。

# Chapter 2

# 程序的组成部分

## 2.1 程序的组成部分

### 预处理器编译指令

C++ 编译器执行的第一项操作是，调用另一个被称为预处理器的工具对源代码进行检查，这是在编译器每次运行时自动进行的。

如果第一个字符是符号 #，它指出这行是一个将由预处理器处理的命令。这些命令称为预处理器编译指令。预处理器的职责是，阅读代码，查找编译指令并根据编译指令相应地修改代码。修改后的代码将提供给编译器。

预处理器相当于编译前的代码编辑，每条编译指令都是一个命令，告诉这位编辑如何做。编译指令 #include 告诉预处理器，将指定文件的全部内容加入到程序的指定位置。C++ 提供了一个标准源代码库，你可在程序中使用它们来执行有用的功能。文件 iostream 中的代码支持输入输出任务，如在屏幕上显示信息以及从用户那里接受输入。

文件名 iostream 前后的 <> 告诉预处理器，前往一组标准位置寻找该文件。由于这些尖括号，预处理器将前往为编译器存储头文件的目录中查找文件 iostream。这些文件也被称为包含文件，因为它们被包含在源代码中。

### 源代码行

函数是执行一个或多个相关操作的代码块，它执行某些操作后返回到调用它的位置。

每个 C++ 程序都包含一个 main() 函数，程序运行时将自动调用 main()。在 C++ 中，所有函数都必须在完成任务后返回一个值。函数 main() 总是返回一个整数，这是使用关键字 int 指定的。

与 C++ 程序中的其他代码块一样，函数也包含在 { 和 } 内。所有函数都以左大括号 { 开头，并以右大括号 } 结尾。

std::cout 后面是 <<，它被称为输出重定向运算符。运算符是代码行中根据某种信息执行操作的字符。运算符 << 显示它后面的信息。

通常，程序返回 0 表示它运行成功，而其他数字表示出现了某种故障。

## 2.2 注释

在 C++ 中，有两种类型的注释。单行注释以两个斜杆（//）打头，导致编译器忽略从这里开始到行尾的全部内容。

多行注释以斜杠和星号（/*）打头，并以星号和斜杆（*/）结尾。/* 和 */ 之间的所有内容都是注释，哪怕它们占据多行。如果程序中不存在与 */ 配对的 /*，编译器将视之为错误。

> 警告
>
> 关于多行注释，需要牢记的一个重点是，不能将其嵌套。如果你使用 /* 开始注释，并在几行后又使用了一个 /*，则编译器见到第一个 */ 后，将认为多行注释到此结束，这样第二个 */ 将导致编译器错误。

## 2.3 函数

main() 是独特的 C++ 函数，因为程序启动时将自动调用它。

程序从函数 main() 开头开始，逐行执行源代码。调用函数时，程序将转而执行该函数，函数执行完毕后，将返回到调用函数的代码行。函数可能返回值，也可能不返回，但函数 main() 是个例外，它总是返回一个整数。

函数由函数头和函数体组成，其中函数头包含以下三项内容。

- 函数的返回类型。
- 函数名。
- 函数接受的参数。

函数名是一个简短的标识符，描述了函数的功能。

函数不返回值时，使用返回类型void，这表示空。

参数是传递给函数的数据，控制函数做什么，函数收到的参数称为**实参**。函数可接受零个、一个或多个参数。参数放在括号内，用逗号分隔，构成参数列表。没有参数的函数包含一组空括号。

函数的名称、参数及其排列顺序被称为签名，函数的签名也唯一地标识了它。

函数名不能包含空格，一般采用驼峰命名法，除第一个单词外，其他每个单词的首字母都大写。

函数体由左大括号、零或多条语句以及右大括号组成。返回值的函数使用 return 语句。return 语句导致函数结束。如果函数不包含 return 语句，将自动在函数体末尾返回 void。在这种情况下，必须将函数的返回类型指定为 void。

# Chapter 3

# 创建变量和常量

## 3.1 变量是什么

变量是计算机内存中的一个位置，您可在这里存储和检索值。可将计算机内存视为一系列排成长队的文件架，并按顺序都为每个文件架进行了编号，而文件架的编号就相当于内存地址。

变量有地址，并赋予了描述其用途的名称。变量名相当于文件架上的标签，这样无需知道变量的实际内存地址就能访问它。

### 在内存中存储变量

在 C++ 中，当您创建变量时，必须将变量的名称和存储的信息类型（如整数、字符或浮点数）告诉编译器，这就是变量的类型，有时也称为**数据类型**。通过变量的类型，编译器将知道需要预留多少内存空间，以存储变量的值。

内存中的每个文件架为 1 字节，如果变量长 2 字节，将需要 2 字节的内存。

短整型（在 C++ 中用 short 表示）通常占用 2 字节，长整型（long）占用 4 字节，整型（int）可以为 2 或 4 字节，长长整型（long long）为 8 字节。

字符类型（在 C++ 中用 char 表示）通常为 1 字节。在 Figure 3.1 中，每个文件架表示 1 字节，因此一个短整型变量可能占据文件架 106 和 107。

布尔值用 bool 类型变量存储，这种变量只能存储值 true 或 false。

short 的长度总是不超过 int，而 int 的长度总是不超过 long。浮点数类型与此不同。

前面提及的常见变量类型的长度并不适用于所有系统，要获悉变量类型的长度，可使用函数 sizeof()，并在括号内指定类型名。函数 sizeof() 由编译器提供，不需要使用编译指令 include。

### 无符号变量和带符号变量

所有整型变量都又分两种类型，这是使用一个关键字指定的。当这些变量只存储正值时，可使用关键字 unsigned 进行声明，而当它们可存储正值或负值时，使用关键字 signed 进行声明。无符号整型变量和带符号整型变量都可存储 0。如果声明时没有指定，则默认为带符号的。带符号和无符号整型

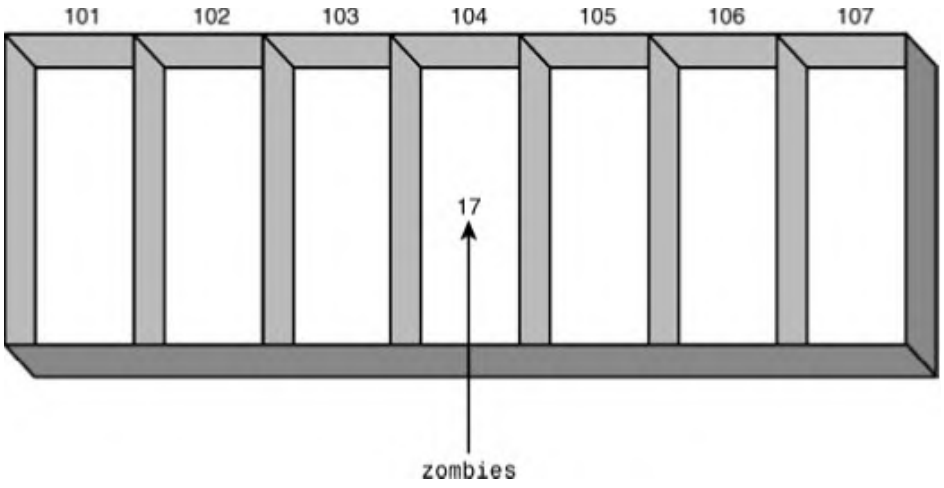


图 3.1: 图显示了 7 个文件架，它们的地址为 101-107。在文件架 104 中，变量 zombies 的值为 17，而其他文件架是空的。

变量占据的字节数相同，因此，无符号整型变量可存储的最大数是带符号整型变量可存储的最大正数的两倍。

### 变量类型

除整型变量外，C++ 还支持浮点数类型和字符。

浮点变量可存储包含小数的值，而字符变量占用 1 字节，可存储 ASCII 字符集中的 256 个字符和符号之一。

在 C++ 中，short 和 long 变量也称为 short int 和 long int，在程序中这两种名称都可以使用。

可将 char 变量用于存储很小的整数，但这是一种糟糕的编程习惯。每个字符都有相应的数值，即字符集中的 ASCII 码。

## 3.2 定义变量

在 C++ 中，变量是通过声明其类型和名称定义的，并以分号结束语句。在一条语句中可定义多个变量，只要它们的类型相同。在这种情况下，应用逗号将变量名分隔。

C++ 保留了一些单词，不能将它们用作变量名，因为它们是 C++ 使用的关键字。保留的关键字包括 if、while、for 和 main。通常，任何合理的变量名都几乎不是关键字。

## 3.3 给变量赋值

要给变量赋值，可使用运算符 =，它被称为赋值运算符。可在声明变量的同时给它赋初值。这称为初始化变量。初始化看起来像赋值，但后面使用常量时，将看到有些变量必须初始化，因为不能给它们赋值。

## 3.4 使用类型定义

当 C++ 程序包含大量变量时，不断输入原本的变量类型，比如 unsigned short 既繁琐又容易出错。要创建现有类型的简捷表示，可使用关键字 typedef，它表示类型定义（type definition）。

类型定义要求使用关键字 typedef，后面跟现有类型及其新名称。

## 3.5 常量

与变量一样，常量也是一个内存位置，可在其中存储值；不同的是，常量的值不会改变，您必须在创建常量时对其进行初始化。C++ 支持两种类型的常量：字面常量和符号常量。

字面常量是直接在需要的地方输入的值。符号常量是用名称表示的常量，与变量类型相似。声明符号常量时，需要使用关键字 const，并在后面跟类型、名称和初值。通常将常量名全部大写，以区分于变量。

### 定义常量

还有另一种定义常量的方法，这起源于早期的 C 语言（C++ 的前身）版本。可使用编译指令 #define 来创建常量，方法是在它后面指定常量的名称和值，并用空格将它们分开。

这种常量不需要指定类型，如 int 或 char。编译指令 #define 执行简单的文本替换，比如将代码中所有的 KILLBONUS 都替换为 5 000，编译器只能看到替换后的结果。

由于这种常量没有指定类型，因此编译器无法确保它们的值是合适的。

### 枚举常量

枚举常量在一条语句中创建一组常量，它们是使用关键字 enum 定义的，后面跟一组用逗号分隔的名称，这些名称放在大括号内。

枚举常量值以 0（对应于第一个常量）打头，其他常量的值依次加 1。

也可以使用赋值运算符指定枚举常量的值。这种方法的优点是，可使用符号名称，如 BLACK 和 WHITE，而不是无意义的数字，如 1 或 700。

## 3.6 自动变量

C++ 还有一个关键字 auto，可用于根据赋给变量的初值推断出变量的类型，这种工作使用编译器完成的。使用 auto 来声明变量时，必须同时对变量进行初始化。

可使用一个 auto 关键字来声明多个变量，条件是这些变量的数据类型相同。

警告

在较久的 C++ 版本中，关键字 auto 用于指出变量为程序中的局部变量–这种概念被称为作用域。C++ 标准制定者研究了数百万行代码，发现 auto 的这种用法很少见，它主要用于测试包中。因此，他们认为 auto 的这种含义是多余的，进而赋予他用于自动确定变量类型的新功能。如果代码以以前的方式使用了 auto，则在 C++14 中将行不通。

## 3.7 总结

变量用于存储在程序运行过程中可改变的值，而常量用于存储不变的值，换句话说，它们是不可变的。

使用变量时，最大的挑战在于选择合适的类型。如果处理的无符号整数可能大于 65000，就应将其存储在 long 变量而不是 short 变量中；如果它们可能大于 21 亿，对 long 类型来说，就太大了。如果数字值包含小数部分，就必须使用 float 或 double 变量来存储，这是 C++ 支持的两种浮点类型。

使用变量时，要牢记的另一点是它们占用的字节数，这因系统而异。函数 sizeof() 提供了编译器返回的变量类型占用的字节数。

# Chapter 4

# 使用表达式、语句和运算符

## 4.1 语句

所有 C++ 都由语句组成，语句是以分号结尾的命令。根据约定，每条语句占一行，但并非必须这样：可将多条语句放在一行，只要每条语句都以分号结尾即可。语句控制程序的执行流程、评估表达式甚至可以什么也不做（空语句）。

### 空白

在 C++ 程序的源代码中，空格、制表符和换行符统称为空白。空白旨在让程序员方便阅读代码，编译器通常忽略它们。

编译器忽略空白，变量名不能包含空白。

### 复合语句

可将多条语句编组，构成一条复合语句，这种语句以左大括号 { 开头，以右大括号 } 结束。可将复合语句放在任何可使用单条语句的地方。

复合语句中的每条语句都必须以分号结尾，但复合语句本身不能以分号结尾，

## 4.2 表达式

表达式是语句中任何返回一个值的部分。赋值运算符 = 导致左操作数的值变为右操作数的值。操作数是一个数学术语，指的是被运算符操作的表达式。

## 4.3 运算符

运算符是导致编译器执行操作的符号，如赋值、执行乘法运算、除法运算或其他数学运算。

### 赋值运算符

赋值表达式由赋值运算符、左操作数（也叫左值）和右操作数（也叫右值）组成。

常量可作为右值，但不能作为左值。左值和右值可能出现在编译器错误消息中。

### 数学运算符

数学运算符有五个：加法运算符（+）、减法运算符（-）、乘法运算符（*）、除法运算符（/）和求模运算符（%）。与 C 语言一样，C++ 也没有乘方运算符（将一个值相乘指定次数），这种任务由函数完成。

加法、减法和乘法运算符与您预期的一致，但除法运算符更复杂。

**整数除法与普通除法不同**。将 21 除以 4 时，结果是一个带小数的实数。但整数除法的结果为整数，余数被丢弃，因此 21 / 4 返回5。

计算模数在编程中很有用。如果要在任务每执行 10 次就显示一条声明，就可以使用表达式 Count % 10。在这里，模数的范围为 0-9，每当模数为 0，就说明任务执行的次数为 10 的整数倍。

浮点数除法与常规除法相同，表达式 21 / 4.0 的结果为 5.25。

C++ 根据操作数的类型决定执行哪种除法。**只要有一个运算符为浮点数变量或浮点数字面量，就将执行浮点数除法；否则，执行整数除法。**

### 组合运算符

经常需要将一个变量与一个值相加，并将结果赋给这个变量。自赋值加法运算符+=将右值与左值相加，然后将结果赋给左值。还有自赋值减法运算符（ =）、自赋值除法运算符（/=）、自赋值乘法运算符（*=）和自赋值求模运算符（%=）。

### 递增和递减运算符

将变量加 1 或减 1 很常见。将变量加 1 称为递增，而将变量减 1 称为递减。C++ 提供了递增运算符 ++ 和递减运算符 - -，用于完成这些任务。

### 前缀运算符和后缀运算符

递增运算符 ++ 和递减运算符 - - 可放在变量名前面，也可放在变量名后面，但效果不同。放在变量名前面称为前缀运算符，放在变量名后面称为后缀运算符。

将变量递增或递减，再将结果赋给另一个表达式时，前缀运算符和后缀运算符的差别将显现出来：**后缀运算符在赋值后执行。**

### 运算符优先级

复杂表达式的结果取决于运算符优先级，即表达式的计算顺序。每个运算符都有优先级。表 Table 4.1 列出了运算符优先级。**突然没看懂这里面的函数，是不是有很多重复的，但是优先级不同？**

表 4.1: 运算符优先级

| 优先级 | 运算符 | 运算顺序 |
|---|---|---|
| 1(最高) | () . [] → :: | 从左到右 |
| 2 | * & ! ∼ ++ - - + - | 从右到左 |
| | sizeof new delete | 从左到右 |
| 3 | .* → * | 从左到右 |
| 4 | * / | 从左到右 |
| 5 | + - | 从左到右 |
| 6 | << >> | 从左到右 |
| 7 | < <= > >= | 从左到右 |
| 8 | == != | 从左到右 |
| 9 | & | 从左到右 |
| 10 | ^ | 从左到右 |
| 11 | \| | 从左到右 |
| 12 | && | 从左到右 |
| 13 | \|\| | 从左到右 |
| 14 | ?: | 从右到左 |
| 15 | = *= /= += %= <<= >>= &=^= \| = | 从右到左 |
| 16(最低) | . | 从左到右 |

表 4.2: 关系运算符

| 名称 | 运算符 |
|---|---|
| 相等 | == |
| 不等 | != |
| 大于 | > |
| 大于等于 | >= |
| 小于 | < |
| 小于等于 | <= |

当优先级顺序不符合要求时，可使用括号来改变顺序。括号内运算符的优先级比其他任何数学运算符都高。括号可以嵌套，在这种情况下，将首先计算最内面的括号内的表达式。

### 关系运算符

关系运算符用于比较，以判断一个数是大于、等于还是小于另一个数。每个关系表达式都是要么返回 true，要么返回 false。Table 4.2 列出了关系运算符。

## 4.4 if-else 条件语句

通过使用关键字 if，可在仅满足条件时执行代码。

### else子句

程序可在 if 条件为 true 时执行一条语句，并在 if 条件为 false 时执行另一条语句。要指定 if 条件为 false 执行的语句，可使用关键字 else。

### 复合if语句

在可使用单条语句的任何地方，都可使用复合语句。if 条件和 if-else 条件后面通常是复合语句。可将任何语句与 if 条件结合使用，这包括另一个 if 条件子句，甚至另一条 if 或 else 语句。

## 4.5 逻辑运算符

通过使用逻辑运算符，可测试多个条件，这包括与运算符 && 和或运算符 ||，而逻辑运算符非 ! 检查表达式是否为 false。

### 与运算符

逻辑运算符与连接两个表达式，如果它们都为 true，那么整个表达式的结果也为 true。

### 或运算符

逻辑运算符或连接两个表达式，只要这两个表达式之一为 true，整个表达式就为 true。

### 非运算符

逻辑非运算符对表达式求反，在表达式为 false 时返回 true，而在表达式为 true 时返回 false。

### 关系运算符和逻辑运算符的优先级

逻辑运算符与和或的优先级相同，因此按从左到右的顺序计算。

## 4.6 棘手的表达式值

条件表达式的结果为 true 或 false。在 C++ 中，0 也被认为是 false，而其他值被认为是 true。

# Chapter 5

# 调用函数

## 5.1 函数是什么

函数是程序的一部分，可对数据执行操作并返回一个值。每个 C++ 程序都至少有一个函数：程序运行时自动调用的函数 main()。这个函数可包含调用其他函数的语句，而这些函数中有些可能又调用其他函数，以此类推。

每个函数都有名称，可用于调用它。函数被调用时，将首先执行其第一条语句，再不断执行，直到到达最后一条语句，然后返回到调用该函数的地方执行。

设计良好的函数执行特定任务。对于复杂的任务，应将其划分成多个函数，然后依次调用它们，这让代码更容易理解和维护。

## 5.2 声明和定义函数

编写函数的代码前，必须声明它。函数声明将函数的名称、函数返回的数据类型以及函数参数的类型告诉编译器。函数声明也叫原型，不包含任何代码。

声明将函数的工作方式告诉编译器。函数原型是单条语句，以分号结尾。

参数列表列出了所有参数及其类型，并用逗号分隔它们。

函数原型必须与函数的三个元素匹配，否则无法通过编译；唯一无需匹配的是形参名。在函数声明中，可根本不指定形参名。

函数可返回任何 C++ 数据类型。如果函数不返回值，就应将返回类型声明为 void。返回类型为 void 的函数不需要包含 return 语句。

不同于函数声明，在函数定义中，指定函数名的语句不以分号结尾。

> 如果将函数定义移到调用它的代码前面，则不需要原型。在小型程序（如本书创建的程序）中，这样可行，但在大型编程项目中，确保所有函数在使用前都进行了定义很麻烦。通过使用原型声明所有函数，就不用考虑这个问题了。

## 5.3 在函数中使用变量

函数以多种方式使用变量：调用函数时可将变量指定为参数；在函数内部可声明变量，这些变量在函数执行完毕后将消失；还可在函数和程序的其他部分之间共享变量。

### 局部变量

在函数内创建的变量为局部变量，因为它只存在于函数内，函数返回后，其所有局部变量都不能供程序使用。

局部变量的创建方式与其他变量相同，函数的形参也被视为局部变量。

在块中声明的变量的作用域为当前块，到达该代码块末尾的右大括号后，这些变量便不可用。可在任何代码块中声明变量，如 if 条件语句和函数内。

### 全局变量

在 C++ 程序中，也可在函数（包括函数 main()）外面定义 C++ 变量，这样的变量称为全局变量，因为它们在程序的任何地方都可用。

在函数外面定义的变量的作用域为全局，因此可在程序的任何函数（包括 main()）内使用。