



第2章 基本数据类型 与表达式



主要内容

2.1 数据类型的概念

2.2 C++基本数据类型

2.3 常量与变量

2.4 操作符

2.5 表达式



主要内容

2.1 数据类型的概念

2.2 C++基本数据类型

2.3 常量与变量

2.4 操作符

2.5 表达式



2.1 数据类型

- 一种数据类型包含两个集合
 - 值集：描述该数据类型包含哪些值
 - 操作集：描述对值集中的值能实施哪些运算
- 两种类型
 - 简单数据类型：值集是不可再分解的简单数据
 - 复合数据类型：值集由其它类型的数据按照一定方式组织而成



2.1 数据类型

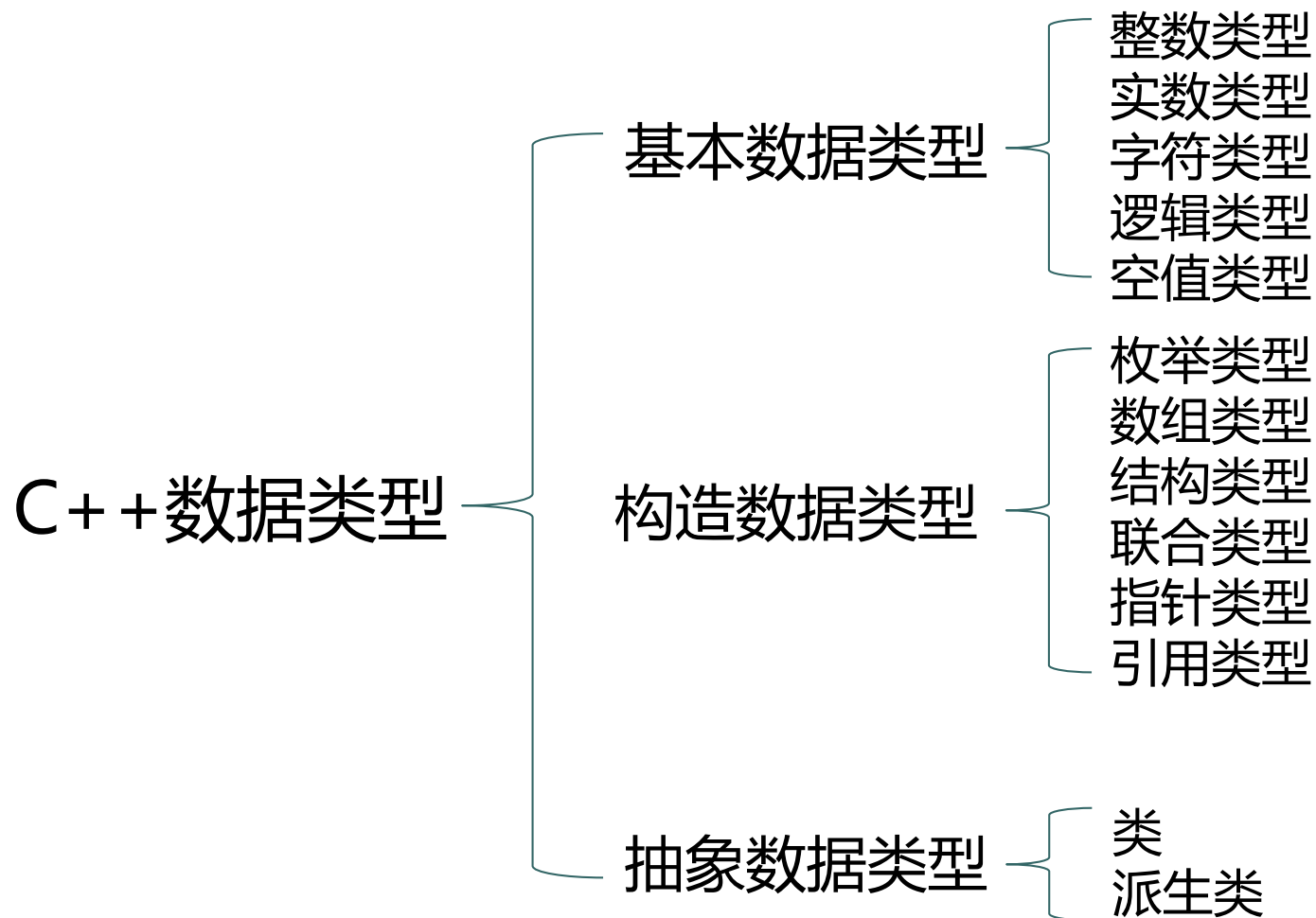
- 语言对类型的支持

- 静态类型与动态类型

- 静态类型：在静态程序中区分类型
 - 动态类型：在程序运行中区分类型

- ✓ C++是静态类型语言

2.1 数据类型





主要内容

2.1 数据类型的概念

2.2 C++基本数据类型

2.3 常量与变量

2.4 操作符

2.5 表达式



2.2 C++基本数据类型

- C++数据类型根据**提供方式**分为：
 - **基本数据类型**：**C++语言**预先定义的数据类型。
 - **构造数据类型**：**程序员**利用语言提供的类型构造机制从其它类型构造出来的数据类型。
 - **抽象数据类型**：**程序员**利用数据抽象机制把**数据与相应的操作**作为一个**整体**来描述的数据类型。



2.2 C++基本数据类型

○ 基本数据类型：

- 整数类型
- 实数类型
- 字符类型
- 逻辑类型
- 空值类型



2.2.1 整数类型

- **都以32位机器为例：**
 - int (4字节)
 - short int 或 short (2字节)
 - long int 或 long (4字节)
- **short类型容易越界，很少用；long范围大，但是计算代价高。**
 - unsigned int
 - unsigned short int 或 unsigned short
 - unsigned long int 或 unsigned long

2.2.2 实数类型

- float占用内存少，double精度高。
 - float (4字节)
 - double (8字节)
 - long double (8、12字节等等)
- 计算机内部：尾数和指数均采用二进制表示

$a \times 2^b$ a: 尾数; b: 指数

1b	a (8b)	b (23b)
----	--------	---------



2.2.3 字符类型

- 在计算机中存储的是字符的编码
 - ASCII码、Unicode码、GB2312码

- 类比：

整数类型 -> 二进制转换 -> 二进制数

字符类型 -> 字符编码器 -> 各种字符编码（二进制）



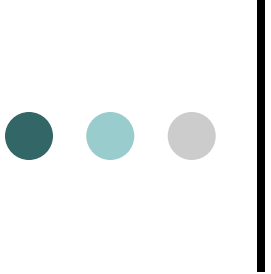
2.2.3 字符类型

- C++的字符类型：
 - char (1字节) : ASCII码
 - wchar_t (2~4字节) : Unicode码
- C++允许把字符类型的数据进行算术运算；进而提供了以下类型：
 - signed char, unsigned char

2.2.4 逻辑类型

- bool类型（1字节）：true, false
- 逻辑值与整数类型之间的转换：
 - true对应1、false对应0
 - 0 --> false、非0 --> true

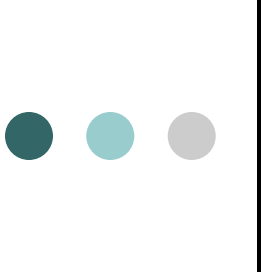
例如：`int x = 10, y = 20; bool z = x;`
`cout << z << endl; // 1`
`cout << y+z << endl; // 21`



2.2.n 空值类型

- void类型

- 没有返回值的函数的返回类型
- 通用指针类型 (void *)



2.2.n+1 其他

○ 统称：

- 各种int、char、wchar_t、bool类型统称为整型 (**integral type**)
- 整型和实数类型统称为算术型 (**arithmetic type**)

○ 取别名：typedef <已有类型> <别名>

- 例子：typedef unsigned int Unit;



主要内容

2.1 数据类型的概念

2.2 C++基本数据类型

2.3 常量与变量

2.4 操作符

2.5 表达式



2.3.1 常量

○ 两种形式

- 常量：用于表示在程序执行过程中不变
- 变量：用于表示在程序执行过程中值可变的数据

○ 常量可以用两种形式表示

- 字面常量：通过**直接写出常量值**来使用的常量。
 - 整数类型字面常量
 - 实数类型字面常量
 - 字符类型字面常量
 - 字符串类型字面常量
- 符号常量：通过常量定义给常量**取一个名字并指定一个类型**，在通过**常量名**来使用这些常量。

整数类型字面常量

- 可以采用下列进制形式来书写：
 - 十进制形式
 - 八进制形式：数字0打头，0-7数字组成
如：073, 0200, -0110为八进制表示；
 - 十六进制形式：0x或0X打头，0-9数字和A-F（或a-f）字母组成
如：0x3B, 0x80, -0x48为十六进制表示
- 在整型常量后面加上l或L，表示long int类型的常量，也可在整型常量后面加上u或U，表示unsigned int类型的常量



实数类型字面常量

- 采用**十进制**书写，内部采用**二进制**存储
- 实数型常量有两种表示法：
 - **小数表示法**：由整数部分、小数点“.”和小数部分构成，例如 456.78, -0.0057。
 - **科学表示法**：在小数表示法后加上指数部分，指数部分由E(e)和一个整数构成，表示基数为10的指数，例如 4.5678E2, -5.7e-3等。
- **实数类型常量为double型**：
 - 在实数型常量后面加上F(f)、或者L(l)表示float型、或者long double类型，例如 5.6F和5.6L。

字符类型字面常量

- 两个单引号 (') 括起来的一个字符：
 - 字符本身，如：'A'
 - 也可以用转义序列表示（由\打头的一串符号）
 - 例如：八进制数 '\ddd'，十六进制数 '\xhh'
 - 'A'的另外两种形式： '\x41'和 '\101'
 - 特殊符号：'\n'（换行符）、'\r'（回车符）等
 - 注意
 - 反斜杠 (\) 应写成：'\''
 - 单引号 (') 应写成：'\''
 - 双引号 (") 可写成：'\"'或 '\"'



字符串类型字面常量

- 两个双引号 (") 括起来的字符序列，其中字符的写法与字符类型常量基本相同。例如：
 - "This is a string."
 - "Please enter \"Y\" or \"N\":"
- 存储字符串时，要在最后一个字符后存储一个字符'\0'，表示字符串结束。

字符常量与字符串常量的区别

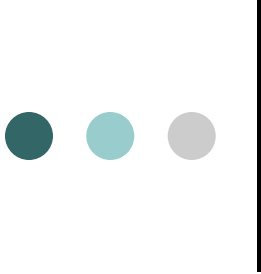
- 字符常量表示单个字符，类型为char；而字符串常量表示多个字符，类型为常量字符数组，对字符串常量的操作按字符数组的规定。
- 字符常量在内存中占1个字节；字符串常量占多个字节：串中字符个数加上1。
- string类：不是C++基本数据类型；标准C++支持



符号常量

- 通过常量定义给常量取一个名字并指定一个类型；在程序中通过常量名来使用这些常量。
- 符号常量的定义格式为：
 - `const <类型名> <常量名>=<值>;`
 - `#define <常量名> <值>`

例如：`const double PI=3.1415926;`
或 `#define PI 3.1415926`
- 优点：易读性；一致性；易维护性



2.3.2 变量（基本特性）

- **变量名**：用标识符表示。
- **类型**：指定变量能取何种值、对其能进行何种运算以及所需内存空间的大小等。
- **值**：在类型的值集范围内可变。
- **内存地址**



2.3.2 变量 (定义)

- $\langle \text{类型名} \rangle \langle \text{变量名} \rangle ;$
- $\langle \text{类型名} \rangle \langle \text{变量名} \rangle = \langle \text{初值} \rangle ;$
- $\langle \text{类型名} \rangle \langle \text{变量名} \rangle (\langle \text{初值} \rangle) ;$

- 例如: `int a=1, b=2, c=a+b;`

2.3.2 变量（输入、输出）

- 可以通过iostream中的cout和cin来完成
- 输入时，空白符作为输入数据之间的分隔符，输入数据的格式应与相应变量的类型相符。

例如：int i; double d; cin >> i >> d;

输入：12 3.4 ☒，则i=12, d=3.4

输入：012 3.4 ☒，则i=12, d=3.4

输入：12.3.4 ☒，则i=12, d值无意义



主要内容

2.1 数据类型的概念

2.2 C++基本数据类型

2.3 常量与变量

2.4 操作符

2.5 表达式



2.4.1 C++ 操作符概述

○ 按功能划分

- 算术操作符
- 关系操作符
- 逻辑操作符
- 位操作符
- 赋值操作符
- 其它操作符

○ 按操作数的个数划分

- 单目操作符
- 双目操作符
- 三目操作符

2.4.2 算术操作符

○ 操作数类型一般为算术类型（整型或者实数）：

- 取负 “-” 与取正 “+”
- 加 “+”、减 “-”、乘 “*”、除 “/” 和取余数 “%”
- 自减 “--”和自增 “++”：
 - 在操作数左侧时，先加后用
 - 在操作数右侧时，先用后加

```
int x=1, y;
```

```
y = (++x); //运行此句，则x=2, y=2
```

```
y = (x++); //运行此句，则x=2, y=1
```

=> 副作用（**side effect**）：得要结果的同时，修改操作数



2.4.3 关系操作符

- 对数据进行大小比较

- $>$ (大于), $<$ (小于), \geq (不小于), \leq (不大于),
 $==$ (相等), $!=$ (不等)

- 关系操作的结果为true或false。

例如: 'A' < 'B'的结果为true

false < true的结果为true



2.4.3 逻辑操作符

- 实现逻辑运算，用于复杂条件的表示中：
 - `!`（逻辑非）、`&&`（逻辑与）、`||`（逻辑或）
- 短路求值
 - 如果第一个操作数已能确定运算结果，则不再计算第二个操作数的值。
 - `true || x` 或者 `false && x`
 - 能够提高逻辑运算的效率，也能为逻辑运算式中的其它运算提供一个“卫士”（guard）

2.4.4 位操作 (1)

- 对操作数的各个二进制位分别进行运算的操作，包括：逻辑位运算和移位运算。

- 逻辑位操作

~ (按位取反), & (按位与), | (按位或), ^ (按位异或)

- 运算规则

$$\sim 0 \rightarrow 1, \sim 1 \rightarrow 0$$

$$0 \& 0 \rightarrow 0$$

$$0 | 0 \rightarrow 0$$

$$0 \wedge 0 \rightarrow 0$$

$$0 \& 1 \rightarrow 0$$

$$0 | 1 \rightarrow 1$$

$$0 \wedge 1 \rightarrow 1$$

$$1 \& 0 \rightarrow 0$$

$$1 | 0 \rightarrow 1$$

$$1 \wedge 0 \rightarrow 1$$

$$1 \& 1 \rightarrow 1$$

$$1 | 1 \rightarrow 1$$

$$1 \wedge 1 \rightarrow 0$$



2.4.4 位操作 (2)

○ 移位操作

- << (左移) : 高位舍弃, 低位补0
- >> (右移) : 低位舍弃, 高位按下面规则处理:
 - 对于无符号数或有符号的非负数, 高位补0
 - 对于有符号数的负数, 高位与原来的最高位相同
(有符号数的二进制补码, 高位1表示负数)

2.4.5 赋值操作

○ 简单赋值操作符

$a = b$

- 当赋值操作的两个操作数类型不同时，将进行隐式类型转换，即把右边操作数转换成左边的操作数类型。

○ 复合赋值操作符

$+=, -=, *=, /=, \%, \&=, |=, ^=, \ll=, \gg=$

- $a \# = b$ 功能上等价于： $a = a \# (b)$
- b 可以是表达式： $(x=y)*z$

2.4.6 其它操作符

- 条件操作符 (?:) $d1 ? d2 : d3$
 - 如果d1值为true或非零，则结果为d2，否则为d3。
- 逗号操作符 $d1, d2, d3, \dots$
 - 从**左至右**依次进行各个运算，操作结果为**最后一个运算**的结果，逗号操作表示的计算更加清晰。
例如： $x = a + b, y = c + d, z = x + y;$
- `sizeof(<类型名>)` 或 `sizeof(<表达式>)`
 - 计算某类型的数据占用的字节数
- 还有其他的操作符，因为：
 - 以上是针对基本数据类型的操作符，当用于抽象数据类型时，带有新含义，例如： `cin >>`



2.4.7 操作数的类型转换

- **运算前**要把操作数转换成相同类型。
 - 算术运算的结果类型与转换后的操作数类型相同。
- 两种类型转换
 - 隐式转换：由**编译程序**按照某种预定的规则进行自动转换，**基本原则**：**低精度 - > 高精度**；
 - 显式转换：由**程序员**在程序中用类型转换操作符明确地指出转换。



(1) 算术操作类型转换

- ① 如果其中一个操作数类型为long double，则另一个转换成**long double**。
- ② 如果其中一个操作数类型为double，则另一个转换成**double**。
- ③ 如果其中一个操作数类型为float，则另一个转换成**float**。
- ④ 先对操作数进行**整型提升转换（见下页）**，如果转换后操作数的类型不一样，则按5)以后的规则再进行转换。



(1) 算术操作类型转换

- 对于char、signed char、unsigned char、short int、unsigned short int类型，如果int型能够表示它们的值，则这些类型转换成int，否则，这些类型转换成**unsigned int**。
- bool型转换成int型，false为0；true为1。
- wchar_t和枚举类型转换成下列类型中第一个能表示其所有值的类型：int、unsigned int、long int、unsigned long int。



(1) 算术操作类型转换

- ⑤ 如果其中一个操作数类型为unsigned long int, 则另一个转换成**unsigned long int**。
- ⑥ 如果一个操作数类型为long int, 另一个操作数类型为unsigned int, 那么, 如果long int能表示unsigned int的所有值, 则unsigned int转换成**long int**, 否则, 两个操作数都转化成**unsigned long int**。
- ⑦ 如果一个操作数类型为long int, 则另一个操作数转换成**long int**。
- ⑧ 如果一个操作数类型为unsigned int, 则另一个操作数转换成**unsigned int**



(1) 算术操作类型转换

○ 举例：

double d;

int i;

unsigned int j;

char ch;

d + i; // i的值转为double型

ch + i; // ch的值转为int型

i + j; // i的值转为unsigned int型

(2) 关系操作的类型转换

- 操作数是算术类型时，按照常规算术转换规则转换。
- 举例：

```
int i; double d; if (d < i) ... // 都转为double
```

```
int a1 = 1, b1 = 1, c1 = 1; if (a1==b1==c1) ... // true
```

```
int a2 = 2, b2 = 2, c2 = 2; if (a2==b2==c2) ...// false
```



(3) 逻辑操作的类型转换

- 操作数是算术类型时，先转换为逻辑类型：
 - 零转为false，非零转为true
 - 不易发现的语法错误：
 - `if (i & j)` 和 `if (i && j)`
 - `if (k = 1)` 和 `if (k == 1)`



(4-6)其它操作符的类型转换

- 位操作符的类型转换
 - 逻辑位操作：算术转换规则
 - 移位操作：整数提升规则
- 赋值操作符的类型转换
 - 右侧类型转换为左侧类型
- 条件操作符的类型转换
 - d1：算术型to逻辑型
 - d2和d3：转换成一致的类型



(n) 隐式转换的问题

- 隐式转换有时不能满足要求

- 例如：

- ```
int i = -10;
```

- ```
unsigned int j = 3;
```

- ```
i+j
```

 将得到错误的结果：4294967289

- 再例如：（溢出）

- ```
int i = 2147483647; //最大的int正整数
```

- ```
int j = 10;
```

- ```
i+j
```

 将得到错误的结果：-2147483639

(n+1) 显示类型转换

- 格式为：

 <类型名> (<操作数>) 或 (<类型名>) <操作数>

- 例如：

 int i = -10;

 unsigned int j = 3;

 i+(int)j 将得到正确的结果： -7

- 再例如：

 int i = 2147483647; //最大的int正整数

 int j = 10;

 (double)i+j 将得到正确的结果： 2147483657.0



主要内容

2.1 数据类型的概念

2.2 C++基本数据类型

2.3 常量与变量

2.4 操作符

2.5 表达式

2.5.1 表达式的构成和分类

- 表达式 = 操作符 + 操作数 + 圆括号
 - 例子: $(a+b)*c/12-\max(a,b)$
 - 基本表达式: 单独的常量或变量
- 表达式的分类
 - 分类1: 算术表达式、关系/逻辑表达式、地址表达式
 - 分类2: 常量表达式、变量表达式
 - 分类3: 左值表达式、右值表达式



2.5.2 表达式的优先级和结合性

- 优先级与结合性

- 基本原则：

- ✓ 单目、双目、三目依次降低

- ✓ 算术、移位、关系、逻辑位、逻辑依次降低

- 具体的优先级与结合性（见P43）

- 不相邻的操作符，C++ 一般没有规定计算次序

2.5.3 表达式中操作数的类型转换

- 根据优先级、结合性，**逐个操作符**进行类型转换

- short int a = 2;

- int b=2147483647;

- double c=2.0;

- 表达式a*b/c得到错误的结果：-1.0。

- 解决办法：(double)a*b/c 或者 a*(double)b/c