在 C++ 中，运算符和函数是构建程序逻辑和控制流的基础。以下是有关 C++ 运算符和函数的相关知识点：

一、运算符（Operators）

运算符是用来对操作数进行操作的符号，C++ 提供了丰富的运算符。运算符可以分为以下几类：

1. 算术运算符（Arithmetic Operators）

用于执行基本的数学运算。

- `+`：加法
- `-`：减法
- `*`：乘法
- `/`：除法
- `%`：取余

```
int a = 10, b = 3;
int sum = a + b;  // 结果是13
int mod = a % b;  // 结果是1
```

2. 关系运算符（Relational Operators）

用于比较两个值。

- `==`：等于
- `!=`：不等于
- `>`：大于
- `<`：小于
- `>=`：大于等于
- `<=`：小于等于

```
int a = 5, b = 10;
bool result = a < b;  // 结果是true
```

3. 逻辑运算符（Logical Operators）

用于处理布尔值逻辑。

- `&&`：逻辑与（AND）
- `||`：逻辑或（OR）
- `!`：逻辑非（NOT）

```
bool x = true, y = false;
bool result = x && y;  // 结果是false
```

4. 赋值运算符（Assignment Operators）

用于将值赋给变量。

- `=`：赋值
- `+=`：加法赋值
- `-=`：减法赋值
- `*=`：乘法赋值
- `/=`：除法赋值
- `%=`：取余赋值

```
int a = 5;
a += 3;  // a 变成 8
```

5. 自增自减运算符（Increment/Decrement Operators）

- `++`：自增运算符
- `--`：自减运算符

```
int a = 5;
a++;  // a变为6
--a;  // a变为5
```

6. 条件运算符（三目运算符）

`?:` 是一种简洁的条件判断运算符。

```
int a = 5, b = 10;
int max = (a > b) ? a : b;  // 结果是 10
```

7. 类型转换运算符（Type Cast Operators）

- `static_cast`：静态类型转换
- `dynamic_cast`：动态类型转换
- `const_cast`：常量类型转换
- `reinterpret_cast`：重新解释类型

```
float f = 3.5;
int i = static_cast<int>(f);  // 将 float 转换为 int
```

二、函数（Functions）

函数是执行特定任务的代码块，C++ 支持各种类型的函数。下面是 C++ 中函数的一些相关知识点。

1. 函数的定义与声明

函数定义包括返回类型、函数名、参数列表和函数体。函数声明是函数的原型，告诉编译器函数的存在。

```
// 函数声明
int add(int, int);

// 函数定义
int add(int a, int b) {
    return a + b;
}
```

2. 返回类型

函数可以有一个返回值，也可以没有返回值（`void`）。返回值的类型必须与声明时指定的类型一致。

```
int add(int a, int b) {
    return a + b;
}
```

3. 参数传递

函数的参数可以通过两种方式传递：

- **传值传递**：将参数的副本传递给函数，函数内部修改不影响外部实参。
- **传引用传递**：将参数的引用传递给函数，函数内部修改会影响外部实参。

```
void byValue(int x) {
    x = 10;  // 不会影响原始数据
}

void byReference(int &x) {
    x = 10;  // 会改变原始数据
}
```

4. 函数重载

C++ 支持函数重载，同名函数根据参数类型和参数数量的不同，定义不同的功能。

```
int add(int a, int b) {
    return a + b;
}

double add(double a, double b) {
    return a + b;
}
```

5. 默认参数

函数可以为参数提供默认值，当调用函数时，如果没有传递某个参数，函数将使用默认值。

```
void printMessage(string msg = "Hello, World!") {
    cout << msg << endl;
}
```

6. 递归函数

递归函数是调用自身的函数，通常用于解决分治问题（如计算阶乘、斐波那契数列等）。

```
int factorial(int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
```

7. 内联函数（Inline Function）

内联函数通过在调用点展开函数体来减少函数调用的开销。可以使用 `inline` 关键字声明。

```
inline int square(int x) {
    return x * x;
}
```

8. Lambda 表达式

C++11 引入了 Lambda 表达式，使得可以定义匿名函数并传递给其他函数。

```
auto add = [](int a, int b) { return a + b; };
cout << add(3, 4);  // 输出 7
```

9. 函数指针

函数指针是指向函数的指针，可以用来传递函数作为参数。

```
int add(int a, int b) {
    return a + b;
}

int main() {
    int (*funcPtr)(int, int) = add;  // 指向函数
    cout << funcPtr(2, 3);  // 调用add函数，输出5
}
```

10. 虚函数与多态

虚函数用于实现多态性，通过基类指针调用虚函数时，会根据实际对象的类型决定调用哪个版本的函数。

```
class Base {
public:
    virtual void show() {
        cout << "Base class" << endl;
    }
};

class Derived : public Base {
public:
    void show() override {
        cout << "Derived class" << endl;
    }
};
```

总结

C++ 中的运算符和函数是编写程序的基础。通过掌握各种运算符的使用，以及函数的不同特点（如函数重载、递归、内联函数等），可以有效地解决复杂的编程问题，并提高程序的可读性和可维护性。