C++ 带默认参数函数学习文档

目录

---

1. 默认参数的基本概念

- **定义**：默认参数是在函数声明或定义中，为形参赋予一个默认值。当调用函数时，若没有为这些形参提供实参，则使用默认值。
- **作用**：
  - 减少函数的调用复杂度。
  - 提供更加灵活的函数调用方式。

示例：

```
int sum(int a = 10, int b = 20) {
    return a + b;
}
```

- 调用 `sum()` 使用默认值 `a=10, b=20`，返回 `30`。
- 调用 `sum(5)` 传入 `a=5`，使用默认值 `b=20`，返回 `25`。
- 调用 `sum(5, 15)` 传入 `a=5, b=15`，返回 `20`。

---

2. 默认参数的定义与规则

2.1 默认参数的定义

- 默认参数可以在函数声明或定义中提供：

```
// 在声明中指定默认值
int sum(int a = 10, int b = 20);

// 在定义中无需重复默认值
int sum(int a, int b) {
    return a + b;
}
```

2.2 默认参数的规则

1. **从右向左赋值**：
   - 默认参数必须从右侧开始连续设置，不能在中间跳过。

```
int sum(int a, int b = 20, int c = 30); // 合法
int sum(int a = 10, int b, int c = 30); // 错误：b 不能跳过
```

2. **只能声明一次默认值**：
   - 默认参数只能在声明或定义中设置一次。

```
int sum(int a = 10, int b = 20); // 声明
int sum(int a, int b = 20) {     // 错误：重复默认值
    return a + b;
}
```

3. **默认值必须是编译时可用的常量**：
   - 默认值可以是常量、全局变量或函数的返回值，但不能是运行时变量。

```
const int x = 10;
int sum(int a = x); // 合法
int sum(int a = foo()); // 合法，foo 是函数
```

---

3. 示例代码与编译器处理解析

3.1 简单调用示例

```
int sum(int a = 10, int b = 20) {
    return a + b;
}

int main() {
    int result1 = sum();        // 使用默认值 a=10, b=20
    int result2 = sum(5);       // a=5, b 使用默认值 20
    int result3 = sum(5, 15);   // a=5, b=15
    return 0;
}
```

输出：

```
result1 = 30
result2 = 25
result3 = 20
```

---

4. 常见错误与注意事项

4.1 跳过中间参数

```
int sum(int a, int b = 20, int c); // 错误：中间参数 b 设置了默认值，但 c 没有默认值
```

解决方法：默认参数必须从右向左连续设置：

```
int sum(int a, int b = 20, int c = 30); // 正确
```

4.2 重复默认值

错误代码：

```
int sum(int a = 10, int b = 20);
int sum(int a, int b = 20) { // 错误：重复定义默认值
    return a + b;
}
```

解决方法：仅在声明或定义中设置一次默认值：

```
int sum(int a = 10, int b = 20); // 在声明中指定默认值
int sum(int a, int b) {          // 在定义中省略默认值
    return a + b;
}
```

4.3 默认值的类型

默认值必须是编译期常量，不能是运行时变量：

```
int x = 10;
int sum(int a = x); // 错误：x 是运行时变量
```

---

5. 编译器的指令翻译分析（详细版）

以下以代码实例进行编译器翻译分析：

代码示例

```
int sum(int a = 10, int b = 20) {
    return a + b;
}

int main() {
    int ret1 = sum();          // 使用默认值调用
    int ret2 = sum(5);         // 部分参数调用
    int ret3 = sum(5, 15);     // 全参数调用
    return 0;
}
```

5.1 调用 `sum()` 使用默认值

代码：

```
int ret1 = sum();
```

编译器生成的指令：

```
push 20           ; 将默认参数 b=20 压栈
push 10           ; 将默认参数 a=10 压栈
call sum          ; 调用 sum 函数
mov [ret1], eax   ; 将返回值存入 ret1
```

5.2 调用 `sum(5)` 部分使用默认值

代码：

```
int ret2 = sum(5);
```

编译器生成的指令：

```
push 20           ; 将默认参数 b=20 压栈
push 5            ; 将显式参数 a=5 压栈
```

```
call sum          ; 调用 sum 函数
mov [ret2], eax   ; 将返回值存入 ret2
```

5.3 调用 `sum(5, 15)` 使用显式参数

代码：

```
int ret3 = sum(5, 15);
```

编译器生成的指令：

```
push 15           ; 显式参数 b=15 压栈
push 5            ; 显式参数 a=5 压栈
call sum          ; 调用 sum 函数
mov [ret3], eax   ; 将返回值存入 ret3
```

---

6. 总结

6.1 总结

1. 默认参数的值由编译器在调用时自动插入。
2. 默认参数从右向左设置，不能跳过中间参数。
3. 默认参数只能在声明或定义中指定一次。
4. 默认参数由编译器在调用处展开，不影响运行效率。