面向对象软件开发技术 –人工智能方向基础核心课程

李长河

自动化学院，信息楼 710

lichanghe@cug.edu.cn

教材

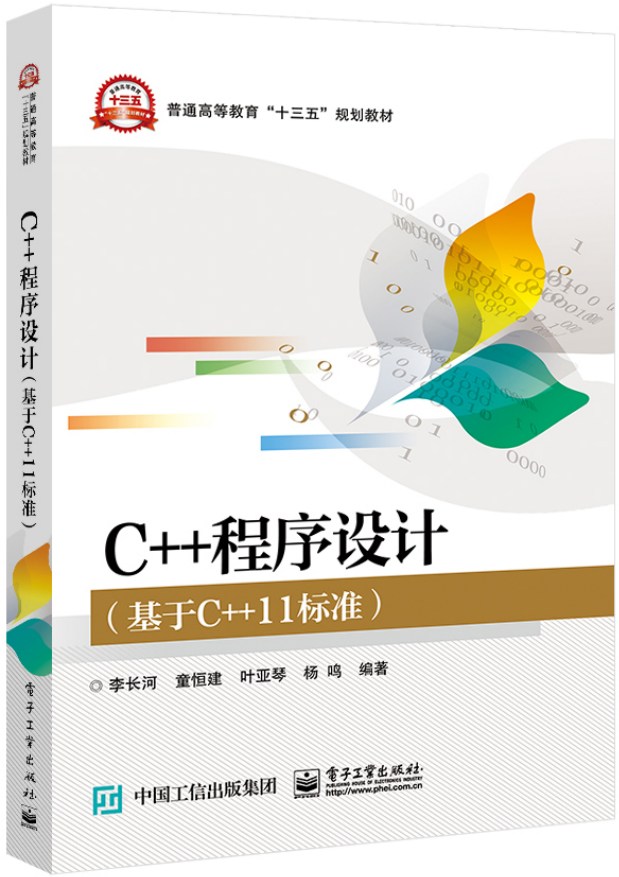
李长河, 童恒建, 叶亚琴等, C++ 程序设计（基于 C++11 标准）. 电子工业出版社, 2019 年 1 月第 2 次印刷

讲义和代码

`https://github.com/Changhe160/book-cplusplus`

参考书

Stanley B. Lippman，Josée Lajoie，Barbara E. Moo. C++ Primer（第五版）. 王刚等译. 北京：电子工业出版社，2013.



学习目标

1. 掌握面向对象程序设计方法；
2. 运用常见数据结构和算法解决实际问题；
3. 熟悉可视化程序设计开发工具（Qt）。

学习目标

1. 掌握面向对象程序设计方法；
2. 运用常见数据结构和算法解决实际问题；
3. 熟悉可视化程序设计开发工具（Qt）。

课时安排

讲课学时：28，实验学时：4（上机考试）

学习目标

1. 掌握面向对象程序设计方法；
2. 运用常见数据结构和算法解决实际问题；
3. 熟悉可视化程序设计开发工具（Qt）。

课时安排

讲课学时：28，实验学时：4（上机考试）

课程纪律

1. 课上严禁看手机；
2. 课下作业和上机考试严禁抄袭，一经发现，均记 0 分处理。

学习目标

1. 掌握面向对象程序设计方法；
2. 运用常见数据结构和算法解决实际问题；
3. 熟悉可视化程序设计开发工具（Qt）。

课时安排

讲课学时：28，实验学时：4（上机考试）

课程纪律

1. 课上严禁看手机；
2. 课下作业和上机考试严禁抄袭，一经发现，均记 0 分处理。

课程考核

课程总成绩 = 考勤 *5%+ 四次作业 *40%+ 上机考试 *40%+ 课程报告 *15%

统计联系方式

利用Excel按照以下格式统计所有学生信息，班长负责在下次上课前发给我

姓名	电子邮件
李长河	lichanghe@cug.edu.cn

助教

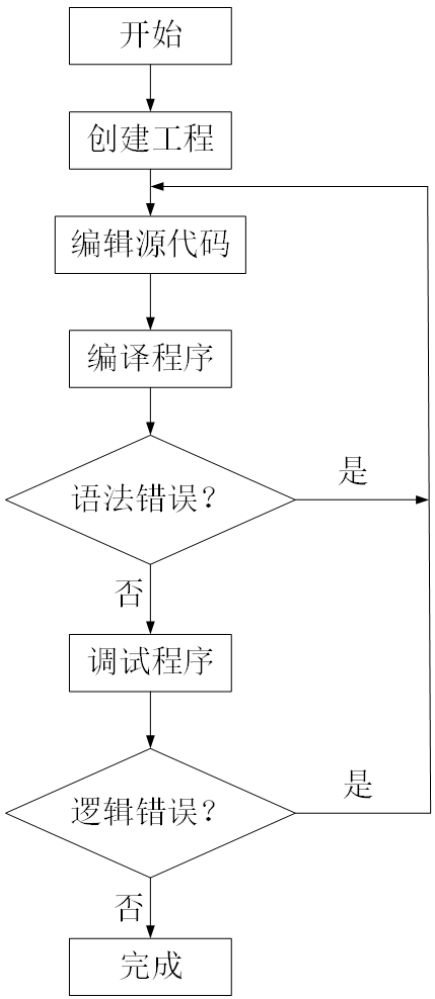
杨瑞	QQ:2280920465
陈宝剑	QQ:2545415252
王宽	QQ:872376093

C++ 基础回顾

目录

1. 编译与调试程序
2. const 修饰符
3. auto 类型推导
4. 引用

1.3 编译与调试程序



说明

1. 建立工程、编写源代码
2. 编译，编译器会指出具体的语法错误
3. 改正语法错误
4. 调试程序，找出逻辑错误（监视窗口观察对象内容的变化）

1.3 编译与调试程序

Visual Studio[a]几个常用快捷键

[a]下载地址 https://visualstudio.microsoft.com/vs/community/

F5	执行程序
F7	编译源文件
F9	添加断点
F10	单步执行一行代码
F11	进入函数体

1.3 编译与调试程序

Visual Studio[a]几个常用快捷键

[a]下载地址 https://visualstudio.microsoft.com/vs/community/

F5	执行程序
F7	编译源文件
F9	添加断点
F10	单步执行一行代码
F11	进入函数体

建议

- 遵循“编辑–编译–调试”的原则
- 养成调试程序的好习惯

2.4 常量修饰符和类型推导 — const 修饰符

对于存储圆周率 π、自然对数 e 等常量的对象，我们不希望其内容发生变化。C++ 提供了关键字const对对象的类型加以限制。

例如

```
//圆周率用pi表示，即有时给常量取个名字更方便
const double pi = 3.14159;
int i = 100;
//利用对象 i 的值初始化 ci
const int ci = i;
```

2.4 常量修饰符和类型推导 — const 修饰符

对于存储圆周率 π、自然对数 e 等常量的对象，我们不希望其内容发生变化。C++ 提供了关键字const对对象的类型加以限制。

例如

```
//圆周率用pi表示，即有时给常量取个名字更方便
const double pi = 3.14159;
int i = 100;
//利用对象 i 的值初始化 ci
const int ci = i;
```

说明

- const 修饰的对象不能改变其内容，即不能对其进行写操作
- const 对象创建时必须初始化

2.4 常量修饰符和类型推导 — const 修饰符

对于存储圆周率 π、自然对数 e 等常量的对象，我们不希望其内容发生变化。C++ 提供了关键字const对对象的类型加以限制。

例如

```
//圆周率用pi表示，即有时给常量取个名字更方便
const double pi = 3.14159;
int i = 100;
//利用对象 i 的值初始化 ci
const int ci = i;
```

说明

- const 修饰的对象不能改变其内容，即不能对其进行写操作
- const 对象创建时必须初始化

问题

判断如下代码是否正确?

```
const int numStudent = 30;
numStudent = 50;
const double pi;
```

2.4 常量修饰符和类型推导 — const 修饰符

对于存储圆周率 π、自然对数 e 等常量的对象，我们不希望其内容发生变化。C++ 提供了关键字const对对象的类型加以限制。

例如

```
//圆周率用pi表示，即有时给常量取个名字更方便
const double pi = 3.14159;
int i = 100;
//利用对象 i 的值初始化 ci
const int ci = i;
```

说明

- const 修饰的对象不能改变其内容，即不能对其进行写操作
- const 对象创建时必须初始化

问题

判断如下代码是否正确?

```
const int numStudent = 30;
numStudent = 50;
const double pi;
```

答案

- 第二行代码错误：不能对 numStudent 进行写值操作
- 第三行代码错误：const 对象必须初始化

4.2 指针—const 和指针

const 和指针

- 可以用 const 修饰符，使其不能修改所指向对象的值，即指向 const 对象的指针。例如：

```
const int ci = 10, cj = 1;
const int *ptrc = &ci; //ptrc 指向常量ci
```

练习：

下面语句有错误吗？若有，则错在哪里？

```
const int a = 30;
const int *c = &a;
*c = 100;
```

4.2 指针—const 和指针

const 和指针

- 可以用 const 修饰符，使其不能修改所指向对象的值，即指向 const 对象的指针。例如：

```
const int ci = 10, cj = 1;
const int *ptrc = &ci; //ptrc 指向常量ci
```

练习：

下面语句有错误吗？若有，则错在哪里？

```
const int a = 30;
const int *c = &a;
*c = 100;
```

答案：语句 *c = 100; 错误，不能修改所指向对象的值

4.2 指针—const 和指针

const 指针

不允许改变指向的指针，语法格式：

```
int j = 0, i = 0;
int *const cptr = &i; //定义时初始化，cptr 只能指向对象i
cptr = &j;             //错误：不能改变cptr 的指向
*cptr = 10;           //正确：可以通过*cptr 修改其指向的对象i 的值
```

4.2 指针—const 和指针

指向 const 对象的 const 指针

```
const int *const cptrc = &ci;//cptrc 是一个指向常量ci 的常量指针
```

第一个 const 修饰符表明 cptrc 为一个指向 const 对象的指针，第二个 const 修饰符表明 cptrc 不能改变指向

2.4 常量修饰符和类型推导 — 类型推导

```
int i = 0;
```

对于上面这条代码，能否根据操作数 0 的类型（int）自动推断出 i 的类型?

2.4 常量修饰符和类型推导 — 类型推导

```
int i = 0;
```

对于上面这条代码，能否根据操作数 0 的类型（int）自动推断出 i 的类型?

答

利用 auto，编译器可以根据初始值的类型自动推导出所需数据类型

```
auto pi=3.14159, rad=1.0; // pi 和 rad 都为 double 类型
auto area=pi*rad*rad;   // area 为 double 类型
auto i=0, pi=3.14159;   // 错误: i 和 pi 的类型不一致
```

2.4 常量修饰符和类型推导 — 类型推导

```
int i = 0;
```

对于上面这条代码，能否根据操作数 0 的类型（int）自动推断出 i 的类型?

答

利用 auto，编译器可以根据初始值的类型自动推导出所需数据类型

```
auto pi=3.14159, rad=1.0; // pi 和 rad 都为 double 类型
auto area=pi*rad*rad;   // area 为 double 类型
auto i=0, pi=3.14159;   // 错误: i 和 pi 的类型不一致
```

注意

- 当用 auto 定义多个对象时，其显示类型必须一致，否则会报错
- auto 不能肆意使用，否则会造成代码的可读性和可维护性下降，使用时需要权衡利弊

4.1 引用

引用

- 为已创建的对象取一个别名
- 只将别名绑定到所引用的对象，对象的内容不会复制给引用
- 函数间共享局部对象的重要途径，对于提高程序的效率有重要作用

4.1 引用

引用

- 为已创建的对象取一个别名
- 只将别名绑定到所引用的对象，对象的内容不会复制给引用
- 函数间共享局部对象的重要途径，对于提高程序的效率有重要作用

语法

```
int counter = 0;
int &refCnt = counter; //refCnt引用counter对象的内容
int &refCnt2;          //错误: 定义引用时必须和一个对象绑定

refCnt = 2;     //修改了counter所在的内存空间的内容
int i = refCnt;//通过引用读取counter对象的内容，并初始化对象i
```

4.1 引用

定义引用时，除了需要初始化外，还需要注意以下几点：

1. 定义多个引用时，每个引用必须用 & 标明:

```
int i = 0;
int &r1 = i, j = 0, &r2 = r1; //r1 和 r2 都是 i 的引用，而 j 是 int 类型
```

2. 只能引用同类型的对象:

```
double d = 0;
int &r3 = d; //错误: r3 只能引用 int 类型对象
```

3. 引用的对象必须是非 const 左值:

```
int i = 0; const int ci = 0;
int &r4 = 100, &r5 = i+1, &r6 = ci; //错误: 只能引用非 const 左值
```

5.3 参数传递 — 引用传递

引用传递

形参是实参的引用

- 引用类型;
- 通过形参改变实参的值。

5.3 参数传递 — 引用传递

引用传递

形参是实参的引用

- 引用类型;
- 通过形参改变实参的值。

引用传递的例子

```
void Swap(int &x, int &y) {//x和y分别是实参i和j的别名
    int z(x);
    x = y;
    y = z;
} //交换x和y所绑定的对象的值
int main() {
    int i(4), j(5);
    Swap(i, j);
    cout << "i=" << i << ",j=" << j << endl;//输出i=4,j=5
    return 0;
}
```