**《面向对象程序设计(基于C++)》教学大纲**

（中文版）

|  |  |  |  |
| --- | --- | --- | --- |
| **课程代码** |  | **课程性质** | **专业选修** |
| **课程名称：** | **面向对象程序设计(基于C++)** | | |
| **英文名称** | Object Oriented Programming & C++ | | |
| **学时/学分** | **72/3** | **其中实验/实践学时** | **36** |
| **开课单位** | **计算机科学与技术学院** | **适用专业：** | **计算机科学与技术** |
| **先修课程** | **程序设计原理与C语言** | | |
| **大纲撰写人** | **陆幼利** | **大纲审核人** |  |
| **课程网址** | **大夏学堂课程主页** | **授课语言** | **中英文双语** |

一、课程说明

本课程是计算机科学和技术专业的专业选修课。

通过本课程的课堂学习和上机实践，使学生掌握面向对象方法的基本概念和C++程序设计的基本原理与基本方法，培养学生使用C++语言进行程序设计的能力和系统开发的能力，为后继课程的学习和从事应用开发工作打下坚实基础，特别为进行面向对象的系统开发打下基础。本课程是一门实践性较强的课程，特别注重学生程序设计上机能力的培养。

二、教学目标

目标1：掌握面向对象程序设计的基础知识、原理和方法(支撑毕业要求1-1)

目标2：能利用面向对象程序设计知识分析计算机编程方面的需求与解决方案(支撑毕业要求1-3)

目标3：掌握面向对象程序设计基本原理，用C++语言编写与调试程序的方法和技巧(支撑毕业要求2-1)

目标4：能应用面向对象程序设计基本原理识别求解问题的关键算法和关键代码，能正确表述面向对象编程问题，通过调试得到正确程序(支撑毕业要求2-2)

三、课程目标与毕业要求的对应关系

|  |  |  |
| --- | --- | --- |
| **毕业要求** | **指标点** | **课程目标** |
| 1工程知识：能够将数学、自然科学、工程基础和专业知识用于解决复杂工程问题。 | 1-1掌握数学、自然科学、工程科学和计算机专业的基础知识、原理和方法。 | 目标1 |
| 1-3 能利用相关知识和数学模型方法推演、分析计算机领域的复杂工程问题 | 目标2 |
| 2问题分析：能够应用数学、自然科学和工程科学的基本原理，识别、表达、并通过文献研究分析复杂工程问题，以获得有效结论。 | 2-1掌握数学、自然科学和工程科学的基本原理，掌握文献检索与研究的方法和技巧。 | 目标3 |
| 2-2 能应用数学、自然科学和工程科学的基本原理识别计算机领域复杂工程问题的关键环节，能正确表述复杂的计算机工程问题 | 目标4 |

四、教学内容

|  |  |  |  |
| --- | --- | --- | --- |
| **章标题** | **知识点** | **支持课程目标** | **要求学生** |
| PART I Introduction to OOP and C++ | 1.1 What is Object Oriented Programming | **目标1** | 1.了解面向对象的基本概念  2.了解C++的基本情况  3.了解C++的各种编程风格  4.了解C++的标准与现状  5.熟悉上机环境的用法  6.完成简单程序的上机操作 |
| 1.2 What is C++ | **目标1** |
| 1.3 C++ Programming Paradigms🞛 | **目标2** |
| 1.4 C++ Standard🞛 | **目标2** |
| 1.5 Programming Environment | **目标3** |
| PART II Introduction to Standard Template Library | 2.1 Introduction to STL | **目标1** | 1. 了解STL概况 2. 理解简单程序的描述 3. 理解名字空间的意义 4. 学习名字空间的表达 5. 学会基本的标准I/O 6. 理解和熟练使用字符串类型 7. 理解容器的概念 8. 理解迭代器的概念 9. 学习常用算法 10. 掌握使用STL的程序设计 |
| 2.2 Hello Program | **目标2** |
| 2.3 The Standard Library Namespace | **目标1** |
| 2.4 Output | **目标2** |
| 2.5 Strings🞛 | **目标3** |
| 2.6 Input | **目标2** |
| 2.7 Containers (vector, list, map, stack, and other standard containers) 🞛 | **目标4** |
| 2.8 Iterators and Algorithms🞛 | **目标4** |
| 2.9 Program Design with STL🞛 | **目标4** |
| PART III Extension to C Language | 3.1 Boolean Type | **目标1** | 1. 学习C++的新增数据类型 2. 学习C++的新增语句 3. 学习C++的新增运算符 4. 了解名字空间和作用范围 5. 理解三种类型的引用 6. 掌握引用和常量引用的使用 7. 理解引用参数传递方式 8. 掌握指针与常量的混合使用 9. 掌握文件输入的基本用法 10. 理解自由存储区的概念与作用 11. 掌握自由存储区的申请与释放 12. 掌握类型转换模板的使用 13. 理解函数名重载的意义与内部处理 14. 掌握重载函数解析规则 15. 编写使用各种参数传递方式的函数 16. 编写使用自由内存区的程序 17. 编写调试稍复杂程序 18. 学会检测内存泄漏 |
| 3.2 Declaration statement | **目标1** |
| 3.3 Scope | **目标1** |
| 3.4 Pointers and Constants🞛 | **目标2** |
| 3.5 Reference, Constant References and Rvalue References🞛 | **目标2** |
| 3.6 More about Input | **目标3** |
| 3.7 Operators | **目标1** |
| 3.8 Free Store🞛 | **目标3** |
| 3.9 Explicit Type Conversion🞛 | **目标2** |
| 3.10 Range For Statement | **目标1** |
| 3.11 Overloaded Function Names🞛 | **目标1** |
| 3.12 Default Arguments | **目标1** |
| 3.13 Programming with call-by-reference🞛 | **目标3** |
| 3.14 programming with free store🞛 | **目标4** |
| 3.15 Debugging C++ Programs🞛 | **目标4** |
| PART IV Namespaces and Exceptions | 4.1 Modularization and Interfaces | **目标1** | 1. 理解模块化设计的意义 2. 理解界面设计的重要性 3. 掌握名字空间的定义与使用 4. 了解异常处理的意义与基本处理方法 5. 理解独立编译的意义 6. 理解连接相关问题 7. 编写调试中等复杂多文件程序 |
| 4.2 Namespaces | **目标1** |
| 4.3 Introduction to Exceptions | **目标1** |
| 4.4 Separate Compilation🞛 | **目标2** |
| 4.5 Linkage🞛 | **目标3** |
| 4.6 Design Program with multiple files🞛 | **目标4** |
| PART V Classes and Objects | 5.1 Introduction to Data Abstraction | **目标1** | 1. 了解数据抽象的意义 2. 掌握抽象数据类型的描述方法 3. 掌握类的定义 4. 掌握封装的意义 5. 理解类定义的各种机制 6. 理解具体类的设计原则 7. 熟练掌握对象定义和销毁时的各种操作细节 8. 熟练掌握巨头概念实现的抽象设计 9. 深入理解说明和实现分离的意义 10. 掌握复制操作定义的一般方法 11. 编写调试类定义与类测试的复杂程序 |
| 5.2 Classes🞛 | **目标1, 目标2** |
| 5.3 Efficient User-Defined Types🞛 | **目标2** |
| 5.4 Objects🞛 | **目标1, 目标2** |
| 5.5 Design data types🞛 | **目标3** |
| 5.6 Seperation of Specification and Implementation🞛 | **目标4** |
| 5.7 User-define Copy🞛 | **目标4** |
| 5.8 Design Programs to define and test classes🞛 | **目标4** |
| PART VI Operator Overloading | 6.1 Introduction to Operator Overloading | **目标1** | 1. 理解设计运算符的原因 2. 理解运算符与函数的关系 3. 掌握运算符定义的一般原则 4. 掌握类型转换的定义与使用 5. 学习友元函数的意义与定义方法 6. 深入理解Move与Copy语意的不同 7. 熟练掌握Move语意的定义 8. 掌握一些主要运算符的一般定义方法 9. 理解下标运算符的意义与定义方法 10. 理解函数调用运算符的意义与定义方法 11. 了解自增自减前后缀的定义区分 12. 设计使用运算符的程序 13. 编写调试高效率运行的复杂程序 |
| 6.2 Operator Functions🞛 | **目标1** |
| 6.3 A Complex Number Type as an Example | **目标2** |
| 6.4 Conversion Operators🞛 | **目标2** |
| 6.5 Friends | **目标1** |
| 6.6 Large Objects and Move Semantics🞛 | **目标2** |
| 6.7 Essential Operators | **目标2** |
| 6.8 Subscripting🞛 | **目标2** |
| 6.9 Function Call🞛 | **目标2** |
| 6.10 Increment and Decrement | **目标1** |
| 6.11 Design classes with operators🞛 | **目标3** |
| 6.12 Debugging Programs with Copy and Move Operations🞛 | **目标4** |
| PART VII Derived Classes | 7.1 Introduction to Inheritance | **目标1** | 1. 了解继承的基本概念 2. 熟练掌握类继承的定义 3. 了解抽象类的意义 4. 理解和掌握虚函数的概念 5. 深入了解多态性的概念与机制 6. 了解使用抽象机制的设计模式 7. 学会继承类的基本编程方法 8. 理解和设计程序框架 9. 掌握使用框架设计程序 10. 编写调试使用框架的复杂程序 |
| 7.2 Derived Classes🞛 | **目标1** |
| 7.3 Abstract Classes🞛 | **目标2** |
| 7.4 Virtual Functions🞛 | **目标2** |
| 7.5 Design of Class Hierarchies🞛 | **目标3** |
| 7.6 Class Hierarchies and Abstract Classes | **目标4** |
| 7.7 Design classes by Inheriting🞛 | **目标3** |
| 7.8 Design a program framework🞛 | **目标4** |
| 7.9 Design programs using framework🞛 | **目标4** |
| PART VIII Templates | 8.1 Introduction to Templates | **目标1** | 1. 了解参数化类型的基本概念 2. 掌握类模板的定义方法 3. 理解类模板的实例化意义 4. 掌握函数模板的定义方法 5. 理解模板特例的概念 6. 掌握模板特例的简单定义 7. 掌握包含模板使用的源代码组织 8. 编写设计函数模板的程序 9. 编写调试复杂的模板相关程序 |
| 8.2 Class Templates Definition and Instantiation🞛 | **目标1** |
| 8.3 Function Templates🞛 | **目标2** |
| 8.4 Specialization | **目标1** |
| 8.5 Source Code Organization Related to Templates | **目标2** |
| 8.6 Design Template Functions🞛 | **目标3** |
| 8.7 Design Template Classes🞛 | **目标1** |
| 8.8 Design programs using Templates🞛 | **目标3** |
| PART IX Exception Handling | 9.1 Error Handling | **目标1** | 1. 了解异常的基本概念 2. 了解异常处理的基本概念 3. 掌握异常类型的设计方法 4. 理解和设计异常处理机制 5. 深入理解使用异常处理机制实现资源管理的意义与具体做法 6. 了解非捕获异常的系统处理方法 7. 理解STL中的异常处理 8. 掌握设计异常安全程序的一般方法 9. 编写调试异常安全的复杂程序 |
| 9.2 Grouping of Exceptions | **目标2** |
| 9.3 Catching Exceptions | **目标2** |
| 9.4 Resource Management & RAII🞛 | **目标3** |
| 9.5 Uncaught Exceptions | **目标1** |
| 9.6 Standard Exceptions | **目标2** |
| 9.7 Design Exception Types🞛 | **目标3** |
| 9.8 Design Exception-safety Programs🞛 | **目标3** |
| 9.9 Debugging Programs with Exception Handling🞛 | **目标4** |
| PART X Streams | 10.1 Standard Streams | **目标1** | 1. 掌握标准流的使用 2. 理解基本格式的用法 3. 掌握文件流的使用 4. 掌握字符串流的使用 5. 理解缓冲区的概念 6. 编写调试使用各种流的程序 |
| 10.2 Formatting | **目标1** |
| 10.3 File Streams🞛 | **目标2** |
| 10.4 String Streams🞛 | **目标2** |
| 10.5 Buffering | **目标2** |
| 10.6 Design Programs with Streams🞛 | **目标4** |

注：🞛表示重点难点内容

五、教学进度

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **周次** | **讲课内容**  **分章和分节的名称** | **课时数** | **习题课、课堂讨论**  **或课堂练习内容** | **课时数** | **实验、实习或**  **科学研究名称** | **课时数** |
| 第1周 | 1.1 What is Object Oriented Programming  1.2 What is C++  1.3 C++ Programming Paradigms  1.4 C++ Standard | 2 |  |  | 1.5 Programming Environment | 2 |
| 第2周 | 2.1 Introduction to STL  2.2 Hello Program  2.3 The Standard Library Namespace  2.4 Output  2.5 Strings  2.6 Input | 2 |  |  | 2.9 Program Design with STL(1) | 2 |
| 第3周 | 2.7 Containers (vector, list, map, stack, and other standard containers)   2.8 Iterators and Algorithms  3.1 Boolean Type  3.2 Declaration statement  3.3 Scope  3.4 Pointers and Constants | 2 |  |  | 2.9 Program Design with STL(2) | 2 |
| 第4周 | 3.5 Reference, Constant References and Rvalue References  3.6 More about Input  3.7 Operators  3.8 Free Store | 2 |  |  | 3.13 Programming with call-by-reference | 2 |
| 第5周 | 3.9 Explicit Type Conversion  3.10 Range For Statement  3.11 Overloaded Function Names  3.12 Default Arguments | 2 |  |  | 3.14 programming with free store | 2 |
| 第6周 | 4.1 Modularization and Interfaces  4.2 Namespaces  4.3 Introduction to Exceptions | 2 |  |  | 3.15 Debugging C++ Programs | 2 |
| 第7周 | 4.4 Separate Compilation  4.5 Linkage  5.1 Introduction to Data Abstraction | 2 |  |  | 4.6 Design Program with multiple files | 2 |
| 第8周 | 5.2 Classes | 2 |  |  | 5.5 Design data types | 2 |
| 第9周 | 5.3 Efficient User-Defined Types | 2 |  |  | 5.6 Seperation of Specification and Implementation | 2 |
| 第10周 | 5.4 Objects  5.7 User-define Copy | 2 |  |  | 5.8 Design Programs to define and test classes | 2 |
| 第11周 | 6.1 Introduction to Operator Overloading  6.2 Operator Functions  6.3 A Complex Number Type as an Example  6.4 Conversion Operators | 2 |  |  | 6.11 Design classes with operators | 2 |
| 第12周 | 6.5 Friends  6.6 Large Objects and Move Semantics  6.7 Essential Operators | 2 |  |  | 6.12 Debugging Programs with Copy and Move Operations(1) | 2 |
| 第13周 | 6.8 Subscripting  6.9 Function Call  6.10 Increment and Decrement | 2 |  |  | 6.12 Debugging Programs with Copy and Move Operations(2) | 2 |
| 第14周 | 7.1 Introduction to Inheritance  7.2 Derived Classes  7.3 Abstract Classes | 2 |  |  | 7.7 Design classes by Inheriting | 2 |
| 第15周 | 7.4 Virtual Functions  7.5 Design of Class Hierarchies  7.6 Class Hierarchies and Abstract Classes | 2 |  |  | 7.8 Design a program framework  7.9 Design programs using framework | 2 |
| 第16周 | 8.1 Introduction to Templates  8.2 Class Templates Definition and Instantiation  8.3 Function Templates  8.4 Specialization  8.5 Source Code Organization Related to Templates | 2 |  |  | 8.6 Design Template Functions  8.7 Design Template Classes  8.8 Design programs using Templates | 2 |
| 第17周 | 9.1 Error Handling  9.2 Grouping of Exceptions  9.3 Catching Exceptions  9.4 Resource Management & RAII  9.5 Uncaught Exceptions  9.6 Standard Exceptions | 2 |  |  | 9.7 Design Exception Types  9.8 Design Exception-safety Programs  9.9 Debugging Programs with Exception Handling | 2 |
| 第18周 | 10.1 Standard Streams  10.2 Formatting  10.3 File Streams  10.4 String Streams  10.5 Buffering | **2** |  |  | 10.6 Design Programs with Streams | **2** |

六、教学方法

课程包括上课学习和上机两部分。教师应力求讲清基本概念、基本思想方法，通过程序实例，介绍程序设计的技能与技巧，特别注意讲解内存管理和其它资源管理等课程难点内容。不布置书面作业，布置具有难度系数稍大的上机作业（建议按小组布置上机任务），以培养学生的编程实践能力和团队合作能力，特别是调试程序的能力。

课程采用双语教学方式。结合学生的实际情况，使用多媒体英文幻灯片和中英文结合的讲课方式。建立支持课程的英文网页和电子资源。

七、考核方式

|  |  |  |  |
| --- | --- | --- | --- |
| **考核方式** | | **具体考核内容** | **支撑课程目标** |
| 过程性 | 出勤情况与课堂表现 | 不无故缺席，积极参与课堂讨论 | 目标1-目标2 |
| 上机作业 | 上机作业的完成情况(每周1次) | 目标3-目标4 |
| 小测 | 共5次左右 | 目标1-目标4 |
| 总结性 | 期末闭卷考试 | 组织卷面考试和上机考试 | 目标1-目标4 |

八、成绩评定

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 出勤表现(5%) | 上机作业(20%) | 小测  (15%) | 期末考试(60%) | 课程目标  达成评价方法 |
| 目标1 | 70 |  | 40 | 40 | 课程目标达成度=  5\*出勤表现平均成绩/100+20%上机作业平均成绩/100  +15%\*小测成绩/100+60%\*期末考试平均成绩/100  即：考核各个部分的加权平均之和  注：子目标达成度另行计算 |
| 目标2 | 30 |  | 40 | 30 |
| 目标3 |  | 60 | 10 | 15 |
| 目标4 |  | 40 | 10 | 15 |

九、推荐教材和教学参考书目与文献

**推荐教材：**

The C++ Programming Language(4th Edition), by Bjarne Stroustrup, Addison Wesley Longman, 2013.

**参考教材：**

1. **Programming Abstraction in C++**, by Eric S. Roberts, Julie Zelenski. Stanford CS106B Course Material
2. **Programming – Principles and Practice Using C++** by Bjarne Stroustrup, Pearson Education Inc. 2009.
3. 上机所用编译系统CodeBlocks的联机文档及G++11文档

**其他学习资源：自建大夏学堂资源**

十、评分标准

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程目标** | **评分标准** | | | | |
| **90-100** | **80-89** | **70-79** | **60-69** | **0-59** |
| **目标1** | 完整掌握面向对象程序设计的基础知识、原理和方法 | 较好掌握面向对象程序设计的基础知识、原理和方法 | 掌握面向对象程序设计的基础知识、原理和方法 | 基本掌握面向对象程序设计的基础知识、原理和方法 | 掌握面向对象程序设计的基础知识、原理和方法达不到基本要求 |
| **目标2** | 完全能利用面向对象程序设计知识分析计算机编程方面的需求与解决方案 | 能较好利用面向对象程序设计知识分析计算机编程方面的需求与解决方案 | 能利用面向对象程序设计知识分析计算机编程方面的需求与解决方案 | 基本能利用面向对象程序设计知识分析计算机编程方面的需求与解决方案 | 利用面向对象程序设计知识分析计算机编程方面的需求与解决方案达不到基本要求 |
| **目标3** | 完全能掌握面向对象程序设计基本原理，用C++语言编写与调试程序的方法和技巧 | 能较好掌握面向对象程序设计基本原理，用C++语言编写与调试程序的方法和技巧 | 能掌握面向对象程序设计基本原理，用C++语言编写与调试程序的方法和技巧 | 基本能面向对象程序设计基本原理，用C++语言编写与调试程序的方法和技巧 | 掌握面向对象程序设计基本原理，用C++语言编写与调试程序的方法和技巧达不到基本要求 |
| **目标4** | 完全能应用面向对象程序设计基本原理识别求解问题的关键算法和关键代码，能正确表述面向对象编程问题，通过调试得到正确程序 | 能较好应用面向对象程序设计基本原理识别求解问题的关键算法和关键代码，能正确表述面向对象编程问题，通过调试得到正确程序 | 能应用面向对象程序设计基本原理识别求解问题的关键算法和关键代码，能正确表述面向对象编程问题，通过调试得到正确程序 | 基本能应用面向对象程序设计基本原理识别求解问题的关键算法和关键代码，能正确表述面向对象编程问题，通过调试得到正确程序 | 应用面向对象程序设计基本原理识别求解问题的关键算法和关键代码，正确表述面向对象编程问题，通过调试得到正确程序达不到基本要求 |