Syllabus for Intermediate Programming

Course name：**Intermediate Programming** Course code：312172040

Credits： Total class hours: 68

I. The nature and purpose of the course

Intermediate programming mainly explains the C language program development methods. This course is an important foundation for computer science and technology major. It is a basic course for students to establish their way of thinking in computers and cultivate their practical ability by quickly mastering a programming language. Through the study of intermediate programming course, students are required to be proficient in programming development using C language, familiar with the process of program development, and master the general methods of program debugging, so as to lay a good foundation for the study of subsequent courses.

II. Course training goal

1．Master the basic syntax and program structure of C language, the basic types and operations of C language, the common statements and three control structures of C language, and the basic usage of common library functions of C language.

2．Establish the thinking mode of program development, master the complete process of program development, be able to carry out program design and code implementation according to the actual needs, and can skillfully use array, pointer and other complex data types to complete the modeling of practical problems.

3．Master some common algorithms, can be implemented with code and applied to complex problems. Able to read the developed source code of a given function, and understand the intent of the author, able to achieve error debugging and modification。

III. Basic content of course teaching

Chapters 1-10 cover the basic features of C, chapters 11-20 cover the advanced features, and chapters 20-27 cover the C standard library. In the teaching content, there are no requirements for the marks ** in chapters 1-18, 20, 22 and chapters 19, 21, 23-27. Students are advised to study by themselves according to their own situation.

Chapter 1 INTRODUCING C

1.1 History of C

1.1.1 Origins

1.1.2 Standardization

1.1.3 C-Based Languages

1.2 Strengths and Weaknesses of C

1.2.1 Strengths

1.2.2 Weaknesses

1.2.3 Effective Use of C

Chapter 2 C FUNDAMENTALS

2.1 Writing a Simple Program

2.1.1 Compiling and Linking

2.1.2 Integrated Development Environments

2.2 The General Form of a Simple Program

2.2.1 Directives

2.2.2 Functions

2.2.3 Statements

2.2.4 Printing Strings

2.3 Comments

2.4 Variables and Assignment

2.4.1 Types

Chapter 14 THE PREPROCESSOR (Support "II. Course training goal " in the second article)

14.1 How the Preprocessor Works

14.2 Preprocessing Directives

14.3 Macro Definitions

14.3.1 Simple Macros

14.3.2 Parameterized Macros

14.3.3 The # Operator

14.3.4 The ## Operator **

14.3.5 General Properties of Macros

14.3.6 Parentheses in Macro Definitions

14.3.7 Creating Longer Macros **

14.3.8 Predefined Macros

14.3.9 Additional Predefined Macros in C99 **

14.3.10 Empty Macro Arguments **

14.3.11 Macros with a Variable Number of Arguments **

14.3.12 The __func__ Identifiler **

14.4 Conditional Compilation

14.4.1 The #if and #endif Directives

14.4.2 The defined Operator

14.4.3 The #ifdef and #ifndef Directives

14.4.4 The #elif and #else Directives

14.4.5 Uses of Conditional Compilation

14.5 Miscellaneous Directives

14.5.1 The #error Directive **

14.5.2 The #line Directive **

14.5.3 The #pragma Directive **

14.5.4 The _Pragma Directive **

Chapter 15 RITING LARGE PROGRAMS (Support "II. Course training goal " in the third article)

15.1 Source Files

15.2 Headers Files

15.2.1 The #includeDirective

15.2.2 Sharing Macro Definitions and Type Definitions

15.2.3 Sharing Function Prototypes

15.2.4 Sharing Variable Declarations

15.2.5 Nested Includes

15.2.6 Protecting Header Files

15.2.7 #error Directives in Headers Files **

15.3 Dividing a Program into Files

15.4 Building a Multiple-File Program

15.4.1 Makefiles

15.4.2 Errors During Linking

15.4.3 Rebuilding a Program

15.4.4 Defining Macros Outside a Program **

Chapter 16 STRUCTURES, UNIONS, AND ENUMERATIONS (Support "II. Course training goal " in the second article)

16.1 Structure Variable

16.1.1 Declaring Structure Variables

16.1.2 Initializing Structure Variables

16.1.3 Designed Initializers

16.1.4 Operations on Structures

16.2 Structure Types

16.2.1 Declaring a Structure Tag

16.2.2 Defining a Structure Type

16.2.3 Structure as Arguments and Return Values

16.2.4 Compound Literals **

16.3 Nested Arrays and Structures

16.3.1 Nested Structures

16.3.2 Arrays of Structures

16.3.3 Initializing an Array of Structures

16.4 Unions

16.4.1 Using Unions to Save Space

16.4.2 Using Unions to Build Mixed Data Structures

16.4.3 Adding a “Tag Field” to a Union

16.5 Enumerations

16.5.1 Enumerations Tags and Type Names

16.5.2 Enumerations as Integers

16.5.3 Using Enumerations to Declare “Tag Fields”

Chapter 17 ADVANCED USES OF POINTERS (Support "II. Course training goal " in the third article)

17.1 Dynamic Storage Allocation

17.1.1 Memory Allocation Functions

17.1.2 NULL Pointers

17.2 Dynamically Allocated Strings

17.2.1 Using malloc to Allocate Memory for a String

17.2.2 Using Dynamic Storage Allocation in String Functions

17.2.3 Arrays of Dynamically Allocated Strings

17.3 Dynamically Allocated Arrays

17.3.1 Using malloc to Allocate Storage for an Array

17.3.2 The calloc Function **

17.3.3 The realloc Function **

17.4 Deallocating Storage

17.4.1 The free Function

17.4.2 The “Dangling Pointer” Problem

17.5 Linked Lists

17.5.1 Declaring a Node Type

17.5.2 Creating a Node

17.5.3 The –> Operator

17.5.4 Inserting a Node at the Beginning of a Linked List

17.5.5 Searching a Linked List

17.5.6 Deleting a Node from a Linked List

17.5.7 Ordered Lists

17.6 Pointers to Pointers

17.7 Pointers to Functions

17.7.1 Function Pointers as Arguments **

17.7.2 The qsort Function **

17.7.3 Other Uses of Function Pointers **

17.8 Restricted Pointers (C99) **

17.9 Flexible Array Members (C99) **

Chapter 18 DECLARATIONS（Support "II. Course training goal " in the third article）

18.1 Declaration Syntax

18.2 Storage Classes

18.2.1 Properties of Variables

18.2.2 The auto Storage Class

18.2.3 The static Storage Class

18.2.4 The extern Storage Class

18.2.5 The register Storage Class

18.2.6 The Storage Class of a Function

18.2.7 Summary

18.3 Type Qualifiers

18.4 Declarators

18.4.1 Deciphering Complex Declarations

18.4.2 Using Type Definitions to Simplify Declarations

18.5 Initializers

18.6 Inline Function

18.6.1 Inline Definitions**

18.6.2 Restrictions on Inline Functions **

18.6.3 Using Inline Functions with GCC**

Chapter 20 LOW-LEVEL PROGRAMMING (Support "II. Course training goal " in the third article)

20.1 Bitwise Operators

20.1.1 Bitwise Shift Operators

20.1.2 Bitwise Complement, And, Exclusive Or, and Inclusive Or

20.1.3 Using the Bitwise Operators to Access Bits

20.1.4 Using the Bitwise Operators to Access Bit-Fields

20.2 Bit-Fields in Structures **

20.3 Other Low-Level Techniques **

20.3.1 Defining Machine-Dependent Types

20.3.2 Using Unions to Provides Multiple Views of Data

20.3.3 Using Pointers as Addresses

20.3.4 The volatile Type Qualifier

Chapter 22 INPUT/OUTPUT (Support "II. Course training goal " in the second article)

22.1 Streams

22.1.1 File Pointers

22.1.2 Standard Streams and Redirection

22.8 String I/O

22.8.1 Output Functions

22.8.2 Input Functions

IV. Course Assessment Method

This course is an examination course. The ratio of the ordinary grade, the ordinary test and the final exam in the total score is (3:2:5). The grades take into account students' performance, including attendance, questions, homework, etc. The usual test includes 2 staged class tests。

Class hours allocation table

The teaching content	Lecture hours	Practice periods
Chapter 1	1	
Chapter 2	2	
Chapter 3	1	
Chapter 4	4	2
Chapter 5	3	2
Chapter 6	4	4
Chapter 7	1	
Chapter 8	3	2
Chapter 9	3	2
Chapter 10	2	
Chapter 11	3	2
Chapter 12	3	2
Chapter 13	3	
Chapter 14	2	
Chapter 15	2	2
Chapter 16	2	

Chapter 17	3	
Chapter 18	1	
Chapter 20	2	
Chapter 22	3	2
Total	48	20

V. References

C Programming: A modern approach (second edition), K. N. King, ISBN-10: 0-393-97950-4, ISBN-13: 978-0-393-97950-3

《C 语言程序设计_现代方法 第 2 版》 作者：(美)K. N. King 译者：吕秀锋 黄倩. 人民邮电出版社 2010