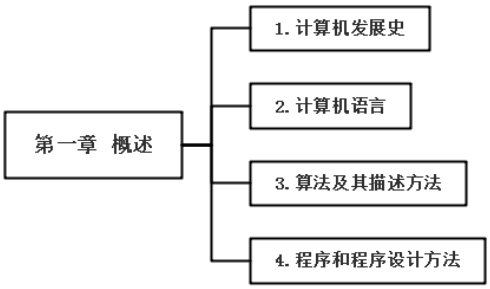
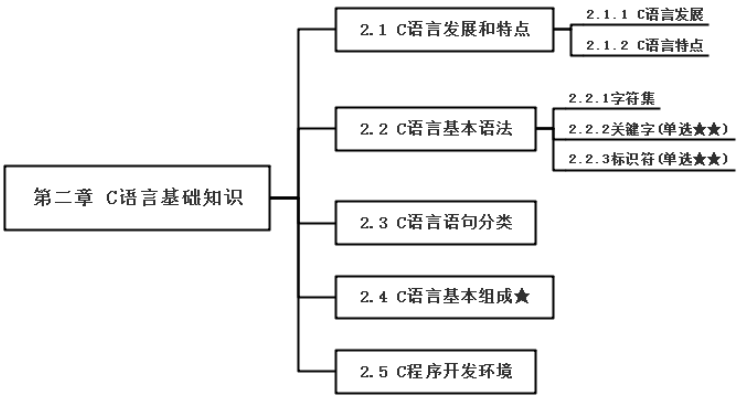
第一章 概述



第一节 计算机发展	1946年世界上第一台电子计算机-电子数字积分计算机诞生
	四个时代：1.电子管计算机时代 2.晶体管计算机时代 3.集成电路计算机时代 4.大规模集成电路时代
第二节 计算机语言	机器语言是用二进制代码表示的能被计算机识别和执行的指令集合
	机器语言和汇编语言都是面向机器的语言，也称为"低级语言"
	源程序是指高级语言编写的程序，它必须翻译成二进制程序才能够执行
	源程序全部翻译成二进制程序之后再运行，则完成该翻译工作的工作程序称为编译程序，编译得到的二进制程序称为目标程序
	在翻译过程中，翻译程序翻译一句执行一句，该翻译程序称为解释程序
第三节 算法及其描述方法	解决问题的步骤序列就是算法
第四节 程序和程序设计方法	计算机程序就是指根据算法描述，用计算机语言表示的能被计算机识别和执行的指令集合

第二章 C 语言基础知识



2.1 C语言发展和特点

2.1.1 C语言发展	C语言源于ALGOL60语言，于20世纪60年代初提出。
2.1.2 C语言特点	（1）结构化语言（2）运算能力强大（3）数据类型丰富（4）具有预处理能力（5）可移植性好（6）程序执行效率高（7）程序设计自由度大

2.2 C语言基本词法

2.2.1字符集	字符是组成语言最基本的元素。C语言的字符集由字母、数字、空格、标点和特殊字符组成。
2.2.2关键字(单选★★)	表 2-2 C 语言关键字的含义 关键字 含义 关键字 含义 关键字 含义 auto 自动 extern 外部 sizeof 计算字节数 break 中止 float 浮点 static 静态 case 情况 for 对于 struct 结构 char 字符 goto 转向 switch 开关 continue 继续 int 整 typedef 类型定义 const 常量 if 如果 union 共用 default 默认 long 长 unsigned 无符号 do 做 register 寄存器 void 空 double 双 return 返回 volatile 可变的 else 否则 short 短 while 当 enum 枚举 signed 有符号 C语言共有32个关键字，所有的关键字均由小写字母组成。
2.2.3标识符(单选★★)	C语言规定，用户标识符仅由大小写英文字母、数字和下画线组成，且第一个字符不能是数字。

	用户自定义的标识符既不能是C语言的关键字，也不能与用户已编写的函数或C语言的库函数重名。

2.3 C语言基本语句分类

2.3 C语言基本语句分类	按照语句功能或构成的不同，可将C语言的语句分为数据定义语句、赋值语句、函数调用语句、表达式语句、流程控制语句、复合语句和空语句等等，C语言的每个语句都以分号“;”结束。

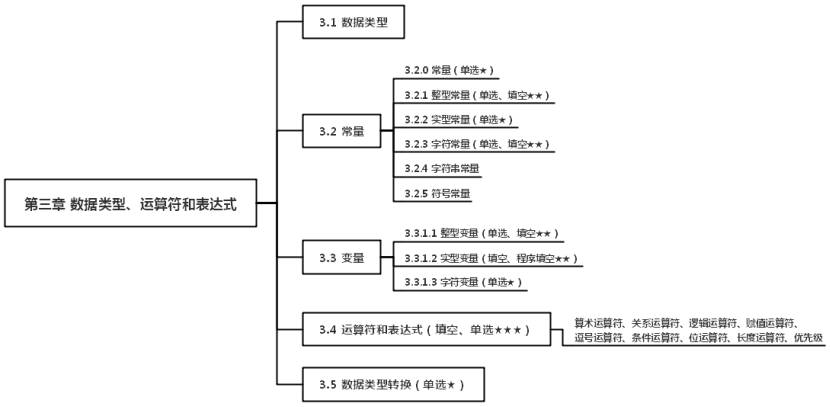
2.4 C程序基本组成

2.4 C程序基本组成(单选★)	C程序是由语句组成的，通常包括一个或多个函数（函数是C程序的基本单位），其中有且只有一个函数称为主函数（位置任意，C程序总是从主函数开始，并在其结束），其函数名为main

2.5 C程序开发环境

2.5 C程序开发环境	C程序的运行步骤：1.编辑 建立源程序文件扩展名为"c"或"cpp" 2.编译 编译程序将源程序转换为二进制形式的目标程序，扩展名为"obj" 3.链接 生成可执行文件，扩展名为"exe" 4.运行
	利用Microsoft Visual C++ 6.0调试C程序时，"Step Into"和"Step Over"都是单步调试程序，前者进入函数内部逐步执行，后者并不进入函数内部执行

第三章 数据类型、运算符和表达式



.1 数据类型

3.1 数据类型	数据类型包括基本类型（整型、实型（又称浮点型）、字符型和枚举型）、构造类型、指针类型和空类型

3.2 常量

3.2.0 常量（单选★）	常量又称为常数，是在程序运行过程中其值不能改变的数据。C 语言规定常量包括整型常量、实型常量、字符型常量、字符串常量和符号常量
3.2.1 整型常量（单选、填空★★）	整型常量有三种书写形式：八进制整型常量（必须以 0 开头，0-7 组成）、十进制整型常量（通常的整数）、十六进制整型常量（以 0X 或 0x 开头，0-9、a-f 或 A-F 组成）
3.2.2 实型常量（单选★）	实型常量又称浮点型常量或实数，只使用十进制。两种书写形式：1.十进制小数形式（必须有小数点） 2.指数形式（aEb 或 aeb，a 为尾数部分，b 为指数部分且必须为整数，均有正负之分）
3.2.3 字符常量（单选、填空★★）	字符常量包括普通字符常量和转义符常量两种，使用一对单引号括起来的单个字符
3.2.4 字符串常量	字符串常量又称"字符串"，使用一对双引号括起来
3.2.5 符号常量	符合常量定义格式为 #define 符号常量 常量

3.3 变量

3.3.0 变量	变量是在程序运行过程中，其值可以改变的数据。变量具有变量名、变量值和变量类型三种属性。
3.3.1.1 整型变量（单选、填空★★）	1、基本整型，类型关键字为 int。2、短整型，类型关键字为 short[int]。3、长整型，类型关键字为 long[int]。4、无符号整型，又称为无符号基本整型（unsigned [int]）、无符号短整型（unsigned short）和无符号长整型（unsigned long）三种。在 16 位编译器中，一般一个 int 型变量占用 2 字节，且 long 型（4 字节）≥int 型（2 字节）≥short 型（2 字节）；而在 32 位编译器中，一个 int 型变量占用 4 字节，long 型变量占用 4 字节，short 型变量占用 2 字节。
3.3.1.2 实型变量（填空、程序填空★★）	1、单精度实型，类型关键字为 float，一般占用内存 4 字节，保留 6~7 位有效数字。 2、双精度实型，类型关键字为 double，一般占用内存 8 字节，保留 15~16 位有效数字。
3.3.1.3 字符变量（单选★）	字符型变量的类型关键字是 char，存放的是字符的 ASCⅡ码值（无符号整数），占用内存 1 字节。

3.4 运算符和表达式（填空、单选★★★）

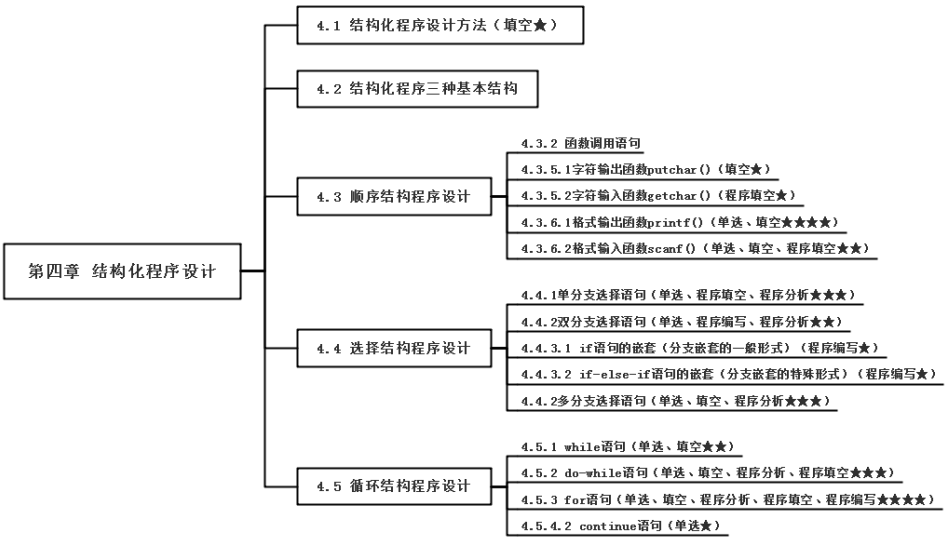
算术运算符	算术运算符包括+、-、*、/、+(取正)、-（取负）,自增运算符++、自减运算符--(单目前缀后引用，单目后缀先引用),算术表达式(由运算对象、圆括号和算法运算符组成)
关系运算符	关系运算符包括>、>=、<、<=、==、!=，逻辑结果值为真（1）或假（0）

逻辑运算符	逻辑运算符包括&&、||和！三种，用于对关系式或逻辑值（真、假）进行运算，运算结果是逻辑值
赋值运算符	赋值运算符分为基本赋值运算符（=）、复合赋值运算符（+-、-=、*=、/=、%=、&=、^=、|=、<<=、>>=）
逗号运算符	逗号运算符又称顺序求值运算符、它是双目运算符、运算对象是表达式，常用于 for 循环语句中
条件运算符	条件运算符格式：表达式 1？表达式 2：表达式 3，执行过程表达式 1 为真，返回表达式 2，若为假返回表达式 3
位运算符	位运算符分为位逻辑运算符（按位非~、按位与&、按位或|、按位异或^）和位移位运算符(左移<<,右移>>)
长度运算符	长度运算符:sizeof(数据类型符)或 sizeof(变量)
优先级	表 3-4 C 语言运算符的优先级和结合性 优先级 / 运算符 / 结合性 1 / () [] -> . / 从左至右 2 / ! ~ ++ -- *（取内容） &（取地址） +（取正） -（取负） sizeof / 从右至左 3 / *（乘） / % / 从左至右 4 / +（加） -（减） 5 / << >> 6 / < <= > >= 7 / == != 8 / &（位逻辑与） 9 / ^ 10 / | 11 / && 12 / || 13 / ?: 14 / = += -= *= /= %= &= ^= |= >>= <<= / 从右至左 15 / , / 从左至右

3.5 数据类型转换

3.5 数据类型转换（单选★）	C 语言的数据类型转换可归纳为三种，即自动类型转换、赋值类型转换和强制类型转换（格式：(数据类型符)（表达式））

第四章 结构化程序设计



4.1 结构化程序设计方法

4.1 结构化程序设计方法(填空★)	结构化程序设计是以模块功能和处理过程设计为主的详细设计的基本原则。1、自顶向下，逐步求精 2、模块化设计 3、结构化编码：经模块化设计后进入编码阶段，任何程序都由顺序、选择和循环三种基本结构组成。

4.3 顺序结构程序设计

4.3.2 函数调用语句	库函数分类存放在扩展名为"h"的文件中，称为头函数或头文件，若需要在 C 源程序中使用某个或某些库函数，则必须在源程序的开头包含：#include<头文件名.h>
4.3.5.1 字符输出函数 putchar()（填空★）	【调用格式】putchar(ch) 【功能】把一个字符输出到标准输出设备（显示器）。
4.3.5.2 字符输入函数 getchar()（程序填空★）	【调用格式】getchar()【功能】从标准输入设备（键盘）输入一个字符。
4.3.6.1 格式输出函数 printf()（单选、填空★★★★）	【调用格式】printf(格式控制字符串,输出项表) 【功能】按照用户指定的格式把指定的数据输出到标准输出设备（显示器）。 表 4-1 printf( )函数常用的格式字符及其含义 格式字符 / 含 义 d / 以十进制有符号形式输出整型数据（正整数不输出符号+） o / 以八进制无符号形式输出整型数据（不输出前缀 0） X、x / 以十六进制无符号形式输出整型数据（不输出前缀 0x），若 x 则输出小写十六进制数符 a~f，若 X 则输出大写十六进制数符 A~F u / 以十进制无符号形式输出整型数据 f / 以小数形式输出实型数据（包括单精度实型和双精度实型），小数部分为 6 位 G、g / 以 f 或 e 格式中宽度较短的一种格式输出实型数据，不输出无意义的 0。在用 G 时，若以指数形式输出，则指数以大写 E 表示 E、e / 以指数形式输出实型数，小数部分为 6 位 c / 以字符形式输出，只输出一个字符 s / 输出字符串

4.3.6.2 格式输入函数 scanf()（单选、填空、程序填空★★）	【调用格式】scanf(格式控制字符串,输入项首地址表) 【功能】从键盘按照“格式控制字符串”中规定的格式读取若干个数据，然后按照“输入项首地址表”中的顺序，依次将数据存入相应的变量对应的地址。

4.4 选择结构程序设计

4.4.1 单分支选择语句（单选、程序填空、程序分析★★★）		【格式】if（表达式）语句; 【功能】计算表达式的值，若为“真” （非 0），则执行语句；否则不执行语句。
4.4.2 双分支选择语句（单选、程序设计、程序分析★★）		【格式】if（表达式）语句 1; else 语句 2; 【功能】计算表达式的值，若为“真” （非 0），则执行语句 1；否则执行语句 2.

4.4.3 分支的嵌套	4.4.3.1 if 语句的嵌套（分支嵌套的一般形式）（程序设计★）	if（表达式 1）
		if(表达式 2) 语句 1;
		else 语句 2;
		else
		if(表达式 3) 语句 3;
		else 语句 4;

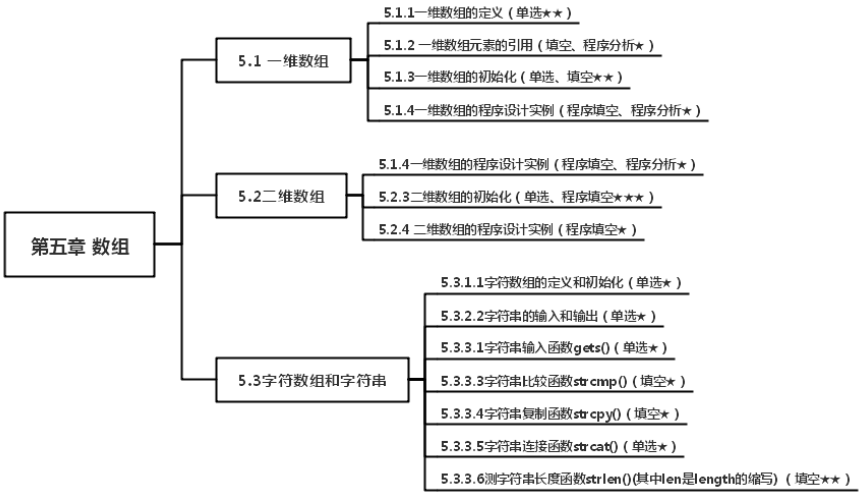
	4.4.3.2 if-else-if 语句的嵌套(分支嵌套的特殊形式)(程序设计★)	if（表达式 1） 语句 1 else if（表达式 2） 语句 2 else if（表达式 3） 语句 3 … else if （表达式 n） 语句 n else 语句 n+1

4.4.2 多分支选择语句（单选、填空、程序分析★★★）		switch(表达式)
		{
		case 常量表达式 1:语句组 1;break;
		case 常量表达式 2:语句组 2;break;
		…
		case 常量表达式 n:语句组 n;break;
		[default:语句组 n+1;[break;]]
		}

4.5 循环结构程序设计

4.5.1 while 语句（单选、填空★★）	while 语句主要用来实现当型循环结构。格式：while(表达式) 语句；功能：计算表达式值，为真则执行语句，重复执行上述操作直至表达式值为假时止
4.5.2 do-while 语句（单选、填空、程序分析、程序填空★★★）	do-while 语句主要用来实现直到型循环结构。格式： do 语句；while(表达式)；功能：执行语句，计算表达式的值，为假退出循环，为真继续循环
4.5.3 for 语句（单选、填空、程序分析、程序填空、程序设计★★★★）	for 语句主要用来实现次数型循环结构。格式：for(表达式 1；表达式 2；表达式 3)语句；功能：第一步，计算表达式 1，第二步计算表达式 2，若为假退出循环，否则执行语句，再计算表达式 3，转第二步
4.5.4.2 continue 语句（单选★）	break 语句，强制结束当前的循环，不再执行循环体中 break 后面的语句，continue 语句，跳过 continue 之后的语句，重新判断循环控制条件，决定是否继续循环

第五章 数组



5.1 一维数组

5.1.1 一维数组的定义（单选★★）	一维数组的定义：数据类型符 数组名 1[长度 1]，数组名 2[长度 2]，...；例：int a[4]；（数组同变量一样，必须先定义后使用）

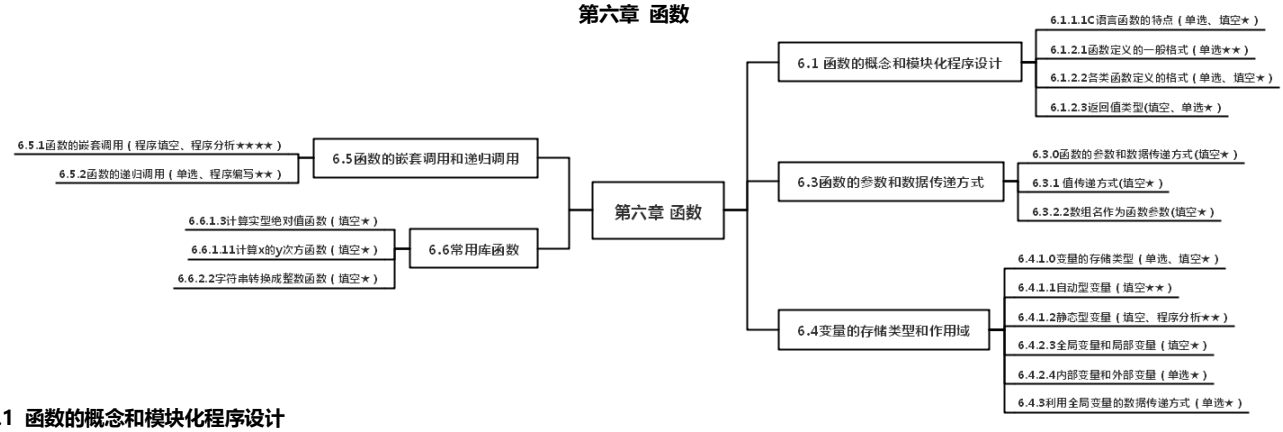
5.1.2 一维数组元素的引用（填空、程序分析★）	一维数组元素的引用：数组名[下标] "下标"可以使整型常量、整型变量或整型表达式，其取值范围是 0~长度-1 例：a[0]=1；
5.1.3 一维数组的初始化（单选、填空★★）	一维数组的初始化：数据类型符 数组名[长度]={初值表}，...； 注意：数组可全部或部分赋值，赋值格式不得超过长度，全部赋值可省略长度
5.1.4 一维数组的程序设计实例（程序填空、程序分析★）	初始化示例：1.int a[4]={1,2,3,4}； 2.int a[4]={1,2}；3.int a[]={1,2,3,4}；

5.2 二维数组

5.2.2 二维数组元素的引用（单选、填空★）	二维数组的定义：数据类型符 数组名[行长度][列长度]，...； 例：int a[3][4]；二维数组的引用：数组名[行下标][列下标]
5.2.3 二维数组的初始化（单选、程序填空★★★）	二维数组的初始化一般形式：数据类型符 数组名[行长度][列长度]={{第 0 行初值表}，{第 1 行初值表}，...{最后 1 行初值表}}；
5.2.4 二维数组的程序设计实例（程序填空★）	二维数组的初始化不分行形式为：数据类型符 数组名[][]={初值表}；

5.3 字符数组和字符串

5.3.1.1 字符数组的定义和初始化（单选★）	一维字符数组的定义形式：char 数组名[数组长度]={初值表}; 二维字符数组定义形式：char 数组名[行长度][列长度]={{初值表},{初值表},...,{初值表}};
5.3.2.2 字符串的输入和输出（单选★）	一维字符数组的输入和输出 （1）使用 scanf()和 printf()函数 逐个字符输入和输出。在 scanf()函数或 printf()函数中用“%c”格式说明符每次输入或输出一个字符，重复该过程完成整个字符串的输入和输出。 字符串整个输入和输出。在 scanf()函数或 printf()函数中用“%s”格式说明符一次性输入或输出整个字符串。 （2）使用 gets()和 puts()函数。使用字符串输入函数 gets()或字符串输出函数 puts()将字符串整个输入或输出。
5.3.3.1 字符串输入函数 gets()（单选★）	【调用格式】gets(字符数组名) 【功能】从标准输入设备即键盘上输入一个字符串（可以包含空格，仅以回车作为结束标志），并将其存储到指定的字符数组中。
5.3.3.3 字符串比较函数 strcmp()（填空★）	【调用格式】strcmp(字符串 1,字符串 2) 【功能】比较两个字符串的大小，即两个字符串从左到右逐个字符进行比较（按照 ASCⅡ码值的大小进行比较），直到出现不同字符或遇到 ‘\0’ 为止。
5.3.3.4 字符串复制函数 strcpy()（填空★）	【调用格式】strcpy(字符数组名,字符串[,整型表达式]) 【功能】将“字符串”的前“整型表达式”个字符组成新的字符串复制到“字符数组”中，若省略“整型表达式”，则将“字符串”完整地复制到“字符数组中”，字符数组的原有内容被覆盖。
5.3.3.5 字符串连接函数 strcat()（单选★）	【调用格式】strcat(字符数组名,字符串) 【功能】把“字符串”连接到“字符数组”中字符串的尾端（最后一个有效字符的后面），组成新的字符串并存储到“字符数组”。“字符数组”中原来的结束标志，被“字符串”的第一个字符覆盖，而“字符串”在操作中未被修改。
5.3.3.6 测字符串长度函数 strlen()(其中 len 是 length 的缩写) （填空★★）	【调用格式】strlen(字符串) 【功能】测字符串（字符串常量或字符数组）的实际长度（不包含字符串结束标志符 ‘\0’ ）。



第六章 函数

6.1 函数的概念和模块化程序设计

6.1.1.1C 语言函数的特点（单选、填空★）	**特点**：1、一个 C 程序由一个或多个函数组成，其中必须有且只能有一个 main 函数（称为主函数）。2、C 程序的执行从主函数开始，如果在主函数中调用其他函数，调用后返回到主函数，并在主函数中结束整个程序的运行。3、主函数可以调用其他函数，但其他函数不能调用主函数。主函数由操作系统调用，其他函数之间可以互相调用。
6.1.2.1 函数定义的一般格式（单选★★）	函数定义的一般形式： 数据类型符 函数名(形式参数表) { 数据定义语句部分;

	执行语句部分; } 需要注意的是，函数首部末尾不能加分号。“函数名”是一个标识符，在同一个编译单元中的函数不能重名。“形式参数表”是使用逗号分隔的若干个形式参数及其数据类型的说明。
6.1.2.2 各类函数定义的格式（单选、填空★）	有返回值函数和无返回值函数的主要区别体现在两个方面。一是数据类型符，有返回值函数的数据类型符可以选取任意的数据类型符，而无返回值函数的数据类型符只能是 void；二是有返回值函数的函数体中必须有语句 return(表达式)；而无返回值函数的函数体中的语句 return；可以有也可以省略。
6.1.2.3 返回值类型(填空、单选★)	函数返回值的类型即函数类型，是指返回给主调函数的结果的类型，应根据具体函数的功能确定。如果函数不返回任何值，则函数返回值类型定义为 void，称为“无类型”。

6.3 函数的参数和数据传递方式

6.3.0 函数的参数和数据传递方式(填空★)	C 语言规定在函数之间传递数据包括值传递、地址传递、返回值和全局变量传递四种方式。其中，值传递和地址传递是利用定义函数时设置的形参和调用函数时给出的实参来传递数据的。
6.3.1 值传递方式(填空★)	值传递方式是在形参和实参之间传递数据的一种方式，值传递方式传递的是参数值。判断是否是值传递方式的唯一方法是看函数定义时的形参是不是变量形式。如果形参是变量，则是值传递方式。
6.3.2.2 数组名作为函数参数(填空★)	数组元素只能用做函数实参，和变量作为函数实参一样。在调用函数时，把数组元素的值传递给形参，实现单向的值传递方式。需要注意的是：1、数组元素作为实参时，只要数组的类型和函数形参的类型一致即可，并不要求函数的形参也是下标变量。2、普通变量或下标变量作为函数参数时，形参变量和实参变量由编译程序分配不同的内存单元。

6.4 变量的存储类型和作用域

6.4.1.0 变量的存储类型 (单选、填空★)	C 语言中变量的存储类型分为自动型（auto）、寄存器型（register）、外部型（extern）和静态型（static）四种。变量的存储方式分为静态存储方式和动态存储方式两大类。自动型变量和寄存器型变量属于动态存储方式；外部型变量和静态型变量属于静态存储方式。
6.4.1.1 自动型变量（填空★★）	【格式】[auto] 数据类型 变量表; 关键字 auto 可以省略。当省略 auto 时，C 语言默认 auto 型，即定义变量时不特别声明存储类型的都默认自动型变量。
6.4.1.2 静态型变量（填空、程序分析★★）	【格式】static 数据类型 变量表; 【说明】1、静态型变量存放在内存的静态存储区，属于静态存储方式。静态变量的的生存期与程序的运行期相同。因为函数中的静态变量在函数调用结束时不释放占用的存储空间，因此其值能保留下来，供下一次调用该函数时使用。2、定义静态型变量时如果没有初始化，则编译系统自动赋值 0（整型和实型）或‘\0’（字符型）。3、静态型变量如果初始化，只有第一次执行定义语句时随着分配内存赋予初值，且只赋一次初值。4、静态型变量在程序运行期间始终存在，但在其作用域以外不能使用。
6.4.2.3 全局变量和局部变量（填空★）	生存期覆盖了定义点到整个程序结束的变量称为“全局变量”；生存期只覆盖了某个函数（或复合语句）的变量称为“局部变量”。全局变量存放在静态存储区，属于静态存储方式。全局变量的生存期与程序运行期相同。定义全局变量时如果没有初始化，则自动赋值 0（整型和实型）或‘\0’（字符型）。
6.4.2.4 内部变量和外部变量（单选★）	变量 | 允许的存储类型 | 生存期 | 作用域 外部变量：省略（称无存储类型）/ static → 全局变量 → 定义点到源程序结束 内部变量：省略（默认为 auto）/ auto / register → 局部变量；static → 全局变量 → 定义该变量的函数或复合语句内部
6.4.3 利用全局变量的数据传递方式（单选★）	全局变量有两种，一是在任何函数之外定义的全局变量，其作用域覆盖了定义点到整个源程序结束的所有函数，这种全局变量叫作“外部变量”；二是在函数体内部声明为 static 型的变量。

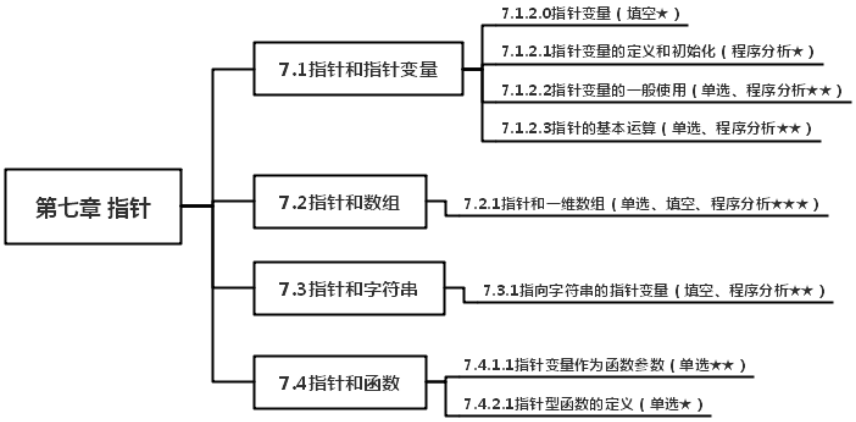
6.5 函数的嵌套调用和递归调用

6.5.1 函数的嵌套调用（程序填空、程序分析★★★★）	数的嵌套调用是指在调用函数时，被调函数又调用了其他函数或自身。当被调函数是函数自身时，称为函数的递归调用，递归调用是嵌套调用的一种特例。
6.5.2 函数的递归调用（单选、程序设计★★）	在调用一个函数的过程中又直接或间接地调用其自身，称为递归调用。递归调用又称为自调用函数。

6.6 常用库函数

6.6.1.3 计算实型绝对值函数（填空★）	【函数首部】int abs(int x)【返回值】返回整数 x 的绝对值。
6.6.1.11 计算 x 的 y 次方函数（填空★）	【函数首部】double pow(double x,double y)【返回值】返回 x 的 y 次方的值。 【说明】不能出现 x、y 均<0；或 x≤0，而 y 不是整数的情况。
6.6.2.2 字符串转换成整数函数（填空★）	【函数首部】int atoi(char *x) 【返回值】返回 x 所指向的字符串转换成的整型数。 【说明】x 所指向的字符串中存放的应当是一个整数形式。

第七章 指针



7.1 指针和指针变量

7.1.2.0 指针变量（填空★）	指针变量是专门用于存储其他变量地址的变量。
7.1.2.1 指针变量的定义和初始化（程序分析★）	【格式】数据类型符　*指针变量名[=初始地址值],...; 【功能】定义指向“数据类型符”的变量或数组的指针变量，同时为其赋初值。
7.1.2.2 指针变量的一般使用（单选、程序分析★★）	1、给指针变量赋值 【格式】指针变量=地址型表达式 2、直接使用指针变量 【格式】指针变量名 3、通过指针变量引用所指向的变量 【格式】*指针变量名
7.1.2.3 指针的基本运算（单选、程序分析★★）	表 7-1　取地址运算符和指针运算符的说明 名称 | 运算符 | 运算对象个数 | 运算规则 | 运算对象类型 | 结果类型 | 结合性 取地址 | & | 单目前缀 | 获取运算对象的地址 | 变量或数组元素 | 运算对象的地址 | 从右至左 指针 | * | 单目前缀 | 获取运算对象对应的变量或数组元素 | 地址（指针变量，变量地址，数组元素的地址） | 运算对象对应的数据 | 从右至左

7.2 指针和数组

7.2.1 指针和一维数组（单选、填空、程序分析★★★）	数组的指针是指向数组在内存的起始地址，数组元素的指针是指向数组元素在内存的起始地址。 若将指针变量指向一维数组，可以采用以下两种方法。 1、在数据定义语句中用赋初值的方式，即*指针变量=数组名。 2、在程序中用赋值的方式，即指针变量=数组名;。 若将指针变量指向一维数组元素，可以采用以下两种方法。 1、在数据定义语句中用赋初值的方式，即*指针变量=&数组名[下标]。 2、在程序中用赋值的方式，即指针变量=&数组名[下标];。 当指针变量指向一维数组，利用指针变量引用一维数组元素的方法如下： 1、引用下标为 0 的数组元素　*(指针变量+0)或 *指针变量 或 指针变量[0] 2、引用下标为 i 的数组元素　*(指针变量+i) 或 指针变量[i] 当指针变量指向下标为 i 的一维数组元素时，利用指针变量引用数组元素的方法如下： 1、引用下标为 i 的数组元素　*(指针变量+0) 或 *指针变量 2、引用下标为 i-k 的数组元素　*(指针变量-k) 3、引用下标为 i+k 的数组元素　*(指针变量+k) 当指针变量指向一维数组时，引用下标为 i 的一维数组元素可以采用四种方法。 1、*(指针变量+i)2、*(数组名+i)3、指针变量[i]4、数组名[i]

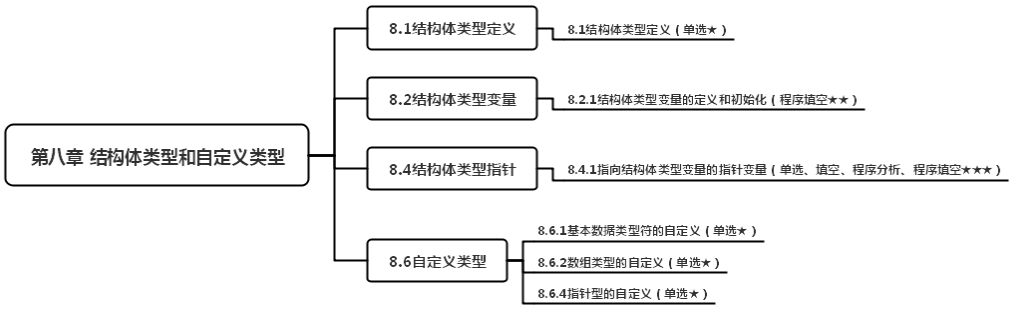
7.3 指针和字符串

7.3.1 指向字符串的指针变量（填空、程序分析★★）	将指针变量指向字符串的方法如下： 1、在数据定义语句中用赋初值的方式　*指针变量=字符串 2、在程序中用赋值的方式　指针变量=字符串; 需要注意的是，这两种方术并不是将字符串赋予指针变量，而是将存放字符串的连续内存单元的首地址赋予指针变量。 当指针变量指向字符串时，则可以利用指针变量处理字符串。处理方式主要有： 处理整个字符串： 输出整个字符串　printf("%s",指针变量); 输入整个字符串　scanf("%s",指针变量);

7.4 指针和函数

7.4.1.1 指针变量作为函数参数（单选★★）	指针变量既可以作为函数的形参，也可以作为函数的实参。指针变量作为函数参数，形参和实参之间的数据传递方式本质上是值传递，只是在调用函数时传递的内容是地址，这样使得形参变量和实参变量指向同一个变量。若被调函数中有对形参所指变量内存的改变，实际上是改变了实参所指变量的内容。
7.4.2.1 指针型函数的定义（单选★）	C 语言中，函数可以返回整型、实型、字符型数据，也可以返回指针类型数据，即返回一个地址。指针型函数是指函数的返回值是指针型，即这类函数的返回值必须是地址值，调用该类函数时，接收返回值的必须是指针变量、指针数组元素等能够存放地址值的对象。定义指针型函数的格式和有返回值的函数的定义格式基本相同，唯一的区别是在函数名前面加一个“*”，表示函数的返回值是指针型数据。

第八章 结构体类型和自定义类型



8.1 结构体类型定义

8.1 结构体类型定义（单选★）	结构体类型定义的一般形式： struct　结构体类型名　　/*struct 是结构体类型的关键字*/ { 　　数据类型符　成员名 1; 　　数据类型符　成员名 2; 　　… 　　数据类型符　成员名 n; };　　/*此行分号不能缺少*/

8.2 结构体类型变量

8.2.1 结构体类型变量的定义和初始化（程序填空★★）	定义结构体类型变量的三种方法：1.先定义结构体类型、后定义结构体类型的变量（间接定义法）2.在定义结构体类型的同时，定义结构体类型变量（直接定义法）3.省略结构体类型名，直接定义结构体类型变量

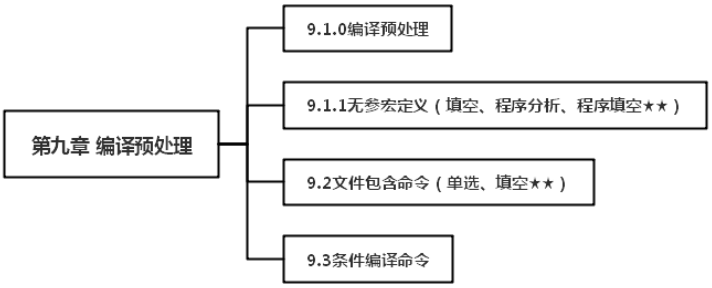
8.4 结构体类型指针

8.4.1 指向结构体类型变量的指针变量（单选、填空、程序分析、程序填空★★★）	定义指向结构体类型变量的指针变量和定义结构体类型变量的方法基本相同，区别是在结构体类型指针变量名的前面加一个“*”。 一般地，如果指针变量 pointer 已指向结构体类型变量 var，则以下三种形式等价： 方式一：var.成员 方式二：pointer->成员 方式三：(*pointer).成员　/*“*pointer”外面的一对圆括号不能省略*/

8.6 自定义类型

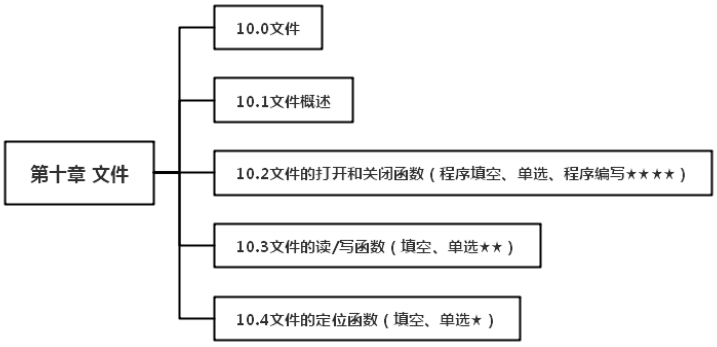
8.6.1 基本数据类型符的自定义（单选★）	【格式】typedef　基本数据类型符　用户类型符; 【功能】将“基本数据类型符”定义为“用户类型符”。
8.6.2 数组类型的自定义（单选★）	【格式】typedef　基本数据类型符　用户类型符[数组长度]; 【功能】以后可以使用“用户类型符”定义“基本数据类型符”的数组，其长度为定义时确定的数组长度。
8.6.4 指针型的自定义 (单选★)	【格式】typedef　struct 　　{ 　　数据类型符　成员名 1; 　　数据类型符　成员名 2; 　　… 　　数据类型符　成员名 n; 　　}用户类型符; 【功能】以后可以使用“用户类型符”定义含有 n 个成员的结构体类型变量、数组和指针变量等。

第九章 编译预处理



9.1.0 编译预处理	编译预处理是指对源程序进行编译之前，先对源程序中的各种预处理命令进行处理，然后再将处理结果和源程序一起进行编译，以获得目标代码
9.1.1 无参宏定义（填空、程序分析、程序填空★★）	通常，宏定义用于定义程序中的符号常量、类型别名、运算式代换和语句代换等。 在程序中使用已经定义的宏名，称为"宏调用";C 程序编译之前将所有的宏名替换为对应的"一串字符"，这一过程称为"宏替换"。由于宏替换是在编译之前完成的，所以宏定义命令属于 C 语言的编译预处理命令。 1.无参宏定义：#define 宏名 字符序列 2.有参宏定义：#define 宏名(形参表) 带参数的字符序列
9.2 文件包含命令（单选、填空★★）	文件包含是指一个源文件可以将另一个源文件的全部内容包含进来 格式：#include "文件名" #include<文件名>
9.3 条件编译命令	条件编译命令格式： #ifdef 标识符 程序段1 [#else 程序段2] #endif #ifndef 标识符 程序段1 [#else 程序段2] #endif #if 表达式 程序段1 [#else 程度2] #endif

第十章 文件



10.0 文件	对 C 语言的文件操作之前需要打开文件，操作结束后需要关闭文件。文件操作凡是主要包括读和写，可以把输入/输出设备当作文件处理，从输入设备读取数据或将数据写到输出设备
10.1 文件概述	C 语言将文件看作是一个一个的字符(ASCII 码文件)或字节(二进制文件)组成的，这种文件称为流式文件。而在其他高级语言中，组成文件的基本单位是记录，对文件操作的基本单位也是记录
	分类：按照文件内容划分可分为程序文件(分为源文件、目标文件和可执行文件)和数据文件，按照文件组织方式可分为存取文件和随机存取文件，按照文件存储形式为文本文件和二进制文件，按照文件的存储介质可分为普通文件(存储介质文件)、设备文件（非存储介质文件） 存取方式：顺序存取和随机存取
	格式：FILE *文件型指针名 1，*文件型指针名 2，...;
10.2 文件的打开和关闭函数（程序填空、单选、程序设计★★★★）	文件打开函数：FILE *fopen(char * filefilename,char *mode) 符号含义：r 读 w 写 a 追加 t 文本文件(可省略) b 二进制 +读和写 文件关闭函数：int fclose(FILE *fp) stdin 指标准输入（即键盘） stdout 指标准输出（即显示器） stderr 指标准输入错误（即显示器）
10.3 文件的读/写函数（填空、单选★★）	文件尾测试函数：int feof(FILE *fp) 功能：若遇到文件尾，则返回值是非 0；否则，返回值是 0 字符读(int fgetc(FILE *fp))/写函数（int fputc(char ch,FILE *fp)） 字符串读(char *fgets(char *str,int length,FILE *fp))/

	写函数（int fputs(char *str，FILE *fp)）数据块读(fread())/写函数(fwrite()) 格式化读（fscanf()）/写函数(fprintf())
10.4 文件的定位函数（填空、单选★）	文件指针复位函数：int rewind(FILE *fp) 功能:将 fp 所指向文件的位置指针重置到文件头，即把文件的位置指针重新定位到 fp 所指向文件的起始位置 文件随机定位函数：int fseek(FILE *fp,long offset,int origin)