C / C++ 语言程序设计

教案

张薇

计算机学院

教学概况：

课程：C/C++语言程序设计

英文名称：C/C++ Language Programming

课程代码：B2085020

先修课程：计算机文化基础

授课班级：材料科学与工程1401，1402，1403，1404

学　期：2015－2016（1）

课程学时：48（理论32＋实验16）

教　材：黄尚福，贾澎涛编《C/C++语言程序设计》

西安电子科技大学出版社

参考书：

1. 谭浩强编著《C程序设计》第四版．清华大学出版社，2010
2. 郑莉编著《C++语言程序设计》．清华大学出版社．2010
3. 李红红编著《C++程序设计实验指导书》，西安科技大学出版社，2010
4. 李学民编著《C/C++语言程序设计》同步进阶与实训100例题与习题指导．西电出版社．2012

课程简介．性质．目的及任务：

C/C++语言可以作为工科各专业的一门专业基础课．是一门面向过程的计算机高级程序设计语言．C++是结构化和面向对象兼容的语言．既可以编写兼容C的结构化程序．又可以编写面向对象的大型程序．

熟练掌握C/C++语言的编程是为学习面向对象的语言以及数据结构和VC++等专业基础课打下良好基础．是相关专业重要的专业基础课．

该课程是理论和技能培养两者并重．相互结合，通过本课程的学习．使学生能够编写和调试C/C++语言的

语法规则，算法与基本结构，程序设计与数制，初步积累编程经验；同时，培养学生良好的程序设计风格及团队协作精神。

★ 授课对象和标：

本课程的授课对象为我校材料学科学与工程学院的学生开设学期为第三学期，面向大二学生，对于非计算机专业的学生，初次接触此程序设计课程，普遍感觉到抽象，难度大，由于2个人手一台电脑，学两节理论课，对应一节实验课，不能及时吸收消化理论课的内容，为了便于学习和结先对于各个课程的掌握，此面向问题为导向的授课方式，结合直观的程序演示来讲解，多讲例题，让学生多动手，将学生易错之处，通过例题和程序演示加深印象，并鼓励学生自分利用上机实践课多动手编程，培养兴趣。

单元教学：　　第1次课

授课课题：第一章 概述
授课时间：
授课类型：理论课
授课学时：2学时

一、教学目标、要求：

(1) 了解C/C++语言的特点；
(2) 理解面向过程和面向对象的程序设计方法和特点；
(3) 掌握C/C++程序的基本结构；
(4) 熟练掌握Microsoft Visual C++ 6.0集成环境编译、链接、运行和调试方法。

二、★知识点：

计算机语言、算法、流程、程序设计
结构化程序设计方法、面向对象程序设计方法。
C/C++程序特点、结构、VC++6.0编译环境

三、★★教学重点：

(1). C程序的基本结构

通过一个简单的C程序实例，掌握C程序的基本结构。通过分析这个简单的C程序，帮助学生理解如何通过程序来控制计算机的操作。

此块内容要着重讲解，因为对学生初次接触计算机语言，没有任何基础，对程序设计都无异非常模糊，甚至对其的理解，教授此部分内容可以用一些简单的

喻. 比如：计算机程序设计语言可以理解为将自然语言翻译成计算机能够理解并接受的语言. 程序的编写就是让计算机以它能理解的方式来解决人们需要解决的问题.

(2). 程序的开发环境和开发过程.

要求学生熟练掌握 C/C++ 程序上机步骤. 这是这门课程实践环节非常重要的. 因为所有程序都要在计算机上进行运行与编译.

让学生掌握在 C 与 C++ 程序开发步骤. 熟悉 Microsoft Visual C++ 6.0 集成环境. 在课堂上当堂演示开发步骤. 让学生一遍又一遍加深印象.

★★ 需要强调的重点：

(1). 学生对于程序设计的陌生感如何一步步消除.

方法：C/C++ 程序设计这门课程不同于其它课程. 它具有很强的抽象性. 要求学生理解为什么需要 C/C++ 语言. 这是因为计算机无法理解人类自然语言. 而人类需求要计算机来帮助我们解决问题. 这就需要一种类似于人类自然语言和机器语言来让计算机能够理解. 这就需要引入 C 或 C++ 类似的程序设计语言.

打比方：人和人之间的交流需要通过语言. 中国人之间用中国话. 英国人用英语. 等等. 人和计算机交流信息. 也需要解决语言问题. 需要创造一种计算机和人都能识别的语言. 这就是计算机语言.

(2). 学生如何能够熟练掌握 VC++6.0 开发环境与开发过程.

方法: 亲自在课堂上用电脑演示 VC++6.0 开发环境, 每一步都讲解清楚. 并编写几个最简单的C语言程序并进行编译. 让学生对程序产生兴趣取.

三. 学生能力分析:

本课程在第三学期开设. 面向大二学生, 对非计算机专业学生. 初次接触程序设计课程. 普遍感觉到抽象. 对编程兴趣不大. 由于不止人手一台电脑. 学而不理论课. 对于一些实验课. 不能及时消化吸收课论课上的讲的内容. 为了便于学生理解. 以面向问题为指导方式. 多举些例子. 多打些比方. 多在课堂上当堂演示来进行讲解. 对于一些易错之处通过演示来让学生加深印象. 鼓励学生多动手编写程序. 逐步产生对于程序设计的兴趣取.

六. 单元教学过程

1. 新课引入:

语言: 人一人语言交流
聋哑人一手势语言交流
人和一计算机语言交流
} 交流方式

计算机语言是人与计算机交流的工具. 所以要学好C语言. 就必须先弄清楚让人与计算机相互理解的方法.

对于非计算机专业的学生, 也许今后工作岗位并不会是一名软件设计工程师或编写员. 但作为一名工科学生. 在今后的工作与科研过程当中. 计算机语言与程序设计是所有工作与科研活动的基础. 举例. 对比等试验结

单元教学模式和手段:

① 运用多媒体辅助教学.
② 适当板书突出重点;
③ 讲授与提问相结合
以互动式教学方式
④ 面向实际例子. 突出
教学内容实践性和
应用性.
⑤ 对重点内容采取提问.

如何学好这门课程：四步法

Step 1：掌握数据类型、控制结构、语法规则（识记、造句、课堂为主）

Step 2：掌握程序分析、算法、编程（阅读、布局、多看多练）

Step 3：循序渐进，先模仿，后分析，最后自己写程序

Step 4：重视上机，有效利用宝贵的上机时间，掌握调试技能。

学生思考，激发学习兴趣。引发思考。⑥当堂编写案例程序，逐步执行程序，分析过程，按照初学者思路编写程序。⑦递进式完善思路，达到教学目标。

2. 新课讲授：

第一章 概述 内容包括：

了解 (1) 计算机语言及其发展
理解 (2) 算法与流程
掌握 (3) 程序设计方法
掌握 (4) C/C++的特点
掌握 (5) C与C++程序实例
掌握 (6) C/C++程序上机步骤

} 板书或PPT演示

1.1 计算机语言及其发展 <讲解>

三个阶段：机器语言 —— 汇编语言 —— 高级语言

各自特点、语法举例（板书或PPT演示）

1.2 算法与流程 <讲解>

(1) 算法的相关概念：所谓算法，指为解决一个问题而采取的方法和步骤，或者说是解题步骤的精确描述。

程序设计的灵魂是算法，而语言只是形式，语言只是一种工具，用来描述处理问题的方法和步骤，但只要有正确的算法，可以利用任何一种语言编写程序。

算法应具备有穷性，确定性，有效性，有零个或多个输

入：有一个或多个输出。

(2) 算法的表示形式：自然语言、传统流程图、结构化流程图（N-S流程图）、伪代码、计算机语言等。

① 传统流程图 —— 板书或PPT演示常用流程图符号

<问题> [例1.1] 求5！ 用传统流程图表现

<思路思考> 首先分析算法，其实 5！= 1×2×3×4×5

还原写出简单算法步骤：（板书或PPT演示）

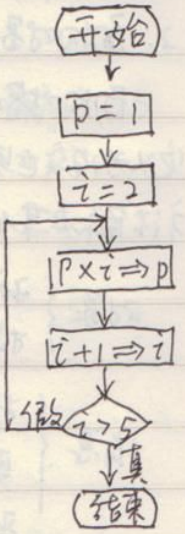
Step 1：设变量 p，相乘结果，p=1；

Step 2：设变量 i，代表乘数，i=2；

Step 3：使 p×i，乘积放在被乘数变量 p 中，可表示为：p×i⇒p

Step 4：使 i 的值加 1，即 i+1 ⇒ i；

Step 5：如果 i 不大于 5，返回重新执行步骤3以后的步骤4，否则，算法结束，最后得到 p 就是 5！的值



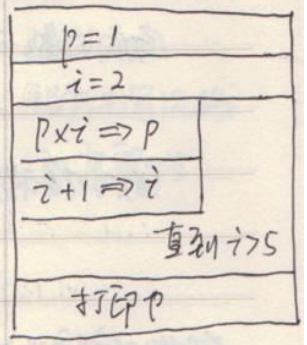
② N-S流程图（盒图）

去掉传统流程图中的流程线

将全部算法写在一个矩形框内

↓

N-S结构化流程图



1.3 程序设计方法

<讲例题>

1. 结构化程序设计方法

任何程序逻辑都可以用顺序、选择和循环三种基本结构来表示。（避免使用goto语句）

使用“自顶向下，逐步求精”的方式，对问题进行分解。

板书三种逻辑：

PPT演示一个结构化程序符合标准。

<PPT或板书>

结构化程序设计方法遵循的原则：

①自顶向下，逐步求精；②模块化设计；③程序结构化

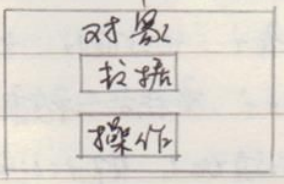
结构化程序设计过程

①分析问题(Q)；②设计算法(A)；③编码程序(P) → QAP方法.

2. 面向对象的程序设计方法

面向对象的程序设计方法的思考方式是面向问题的结构。它认为现实世界是由一个个对象组成的。面向对象的程序设计方法解决某个问题时，要确定这个问题是由哪些对象组成

对象 { 数据 / 操作



<相似>

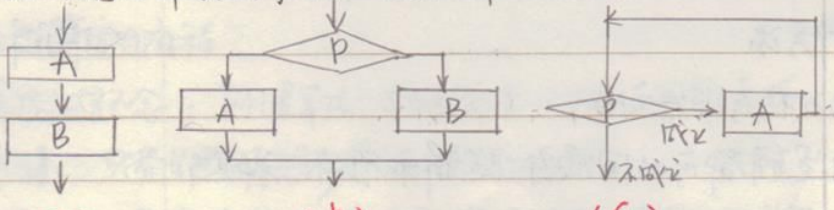
消息 { 接收消息的对象 / 要执行的操作的名字 / 必须需要的参数.

面向对象的特点 { 封装 / 继承 / 多态.

面向对象程序设计过程 { 面向对象的分析 (OOA) / 面向对象的设计 (OOD) / 面向对象的实现 (OOI)

结构化程序设计与面向对象程序设计的比较：

<区别>



1.4. C/C++的特点

<讲解>

1. C语言特点

①语言结构化 ②语言简洁 ③功能强大 ④数据

结构丰富 ⑤运算符丰富 ⑥系统可移植性好 ⑦可移植性好
⑧与C语言兼容（支持面向对象，也支持结构化）等.

2. C++语言特点、PPT源文（编译）

1.5. C与C++程序实例

1. C语言程序实例

<程序演示>

[例1.3] 简单的C语言程序.

```
#include <stdio.h>          /*预处理命令*/
main()                      /*主函数*/
{
    printf("My first C program!\n");
                            /*输出双引号中的内容*/
}
```

[例1.4] 求两个整数之和

此处引入程序书写风格设置要求

```
#include <stdio.h>
main()                      /*主函数*/
{
    int a, b, sum;          /*说明变量为整型类型*/
    a=1;                    /*给变量赋初值*/
    b=2;
    sum=a+b;                /*加法运算*/
    printf("sum=%d\n", sum);
}
```

重点：① C程序由若干函数组成，函数是完成某个整体功能的最小单位. ② C函数从左花括号开始，到对应的右花括号结束. ③ main()可以无论写在何处

何位置上，但C程序执行时，总是从main()函数开始。

2. C++程序实例

[例1.5] 简单C++程序

```
#include <iostream.h>
Void main()
{
    cout << "Hello! My first C++ program!\n";
}
```

讲述一些例题，PPT或板书演示，分析例题，提问 课堂互动

1.6. C/C++程序上机步骤：

创建一个项目；
编辑项目中的源代码；
编译项目中的文件；
纠正编译中出现的错误；
运行可执行的文件

此部分内容为学上机演示详细的上机步骤。
结合PPT进行具体说明

<板书>
约定名词说明：为何要使用VC++6.0作为开发环境；
①VC++6.0基本的全功能
②全国大学生计算机等级考试二级程序设计为VC++

3. 单元教学总结：（课堂总结）

(1) 回顾知识点：（学生学习与教师共同回忆本堂课所讲的知识点）

(2) 简短介绍下次课：一些预习内容。

4. 作业布置：

课后习题一：1～8题

(1) 写作业本上
(2) 上机调试课后例题
(3) 预习内容

单元教学　　第2次课

授课课题：第二章 数据类型和表达式（一）
授课时间：
授课类型：理论课
授课学时：2学时

一. 教学目标、要求：
(1) 理解C语言的数据类型的概念
(2) 理解标识符、常量和变量的概念.
(3) 掌握C语言简单的数据类型
(4) 掌握指针的概念和指针变量的应用.
(5) 理解运算符和表达式的概念.
(6) 掌握算术运算符和算术值表达式
(7) 掌握自增自减运算

二. 知识点：
数据类型. 标识符. 常量与变量. 指针与指针变量
运算符. 表达式. 算术运算符. 算术表达式
自增自减运算

三. 教学重点：
1. C语言中的数据类型
现实世界的客观事物，人类区分性别一样. 在程序中使用的数据也要区分类型. 数据区分类型的目的是便于对它们按不同方式和要求进行处理.
2. C语言的常量和变量
C语言处理的数据包括常量和变量两类. 对于常量来说，它属性由其取值形式表明. 而变量的属性则必须在使用前明确地加以说明.

3. 变量的三个要素：变量名、数据类型、变量的值。

4. 变量的定义方法：[类型修饰符] 数据类型 变量名;

四. 教学难点：

1. 指针类型 —— 相关概念、变量的定义

C语言中的一种重要的数据类型就是指针，指针也是区别其它语言和C语言的特点之一。然而指针的概念对于初次接触C语言的学生而言是较为抽象与难以理解的。

解决方法：结合生活比方，将抽象的概念联系到日常生活中，用一些形象的比喻来让学生尽快的理解。

例如：找人（找地址）

2. 运算符与表达式

C语言语法当中大量的运算符的组织注意。

课本P76 表3.6 运算符的优先级和结合性。

注意区分运算符的优先顺序与数学的区别：

解决方法：① 让学生结合原先数学上的运算符来理解与注意。

② 课堂上当堂多举些常用运算符的例子让学生加深印象。

五. 能力培养：

本章内容较为枯燥，但它是确定C语言语法学习的基础，也是程序设计的基础，本章内容要求学生边学习边实践，可能一下子记不住实践不了会部分内容，可在今后的章节学习中不断强化。

六、单元教学过程

1. 回顾上节课内容：（第1章内容回顾）

计算机语言、程序设计的相关概念、算法的相关概念、流程（传统流程图、N-S流程图）

结构化程序设计方法、面向对象的程序设计方法

C/C++程序结点结构、VC++6.0编译环境

2. 新课引入

数据处理程序加工、处理的对象、也加工的结果。所以在描述程序设计中所要涉及和描述的主要内容，例如：求解决如微分方程求解、学生考试成绩计算和档案信息管理等实际问题中，程序应如何描述相关的数据？程序可能会处理的基本数据对象按照类别分成一些组，或说成一些集合，属于同一集合的各数据对象都具有同样的性质，例如对它们能够做的同样的操作，它们都引用同样的编码方式等。

我们把程序语言中具有这样性质的数据集合称为数据类型。

3. 新课讲授：

第2章 数据类型和表达式

(1) 词法构成 理解
(2) 数据类型 掌握
(3) 常量与变量 掌握
(4) 指针类型 掌握
(5) 运算符和表达式 理解

〈相书〉

★ 2.1. 词法构成
- 字符集
- 标识符
- 关键字
- 注释符

〈相书〉

(1) 字符集

字符是组成词汇和程序的最基本元素。

C语言的字符集是ASCII字符集的一个子集，由字母、数字、标点符号和特殊字符组成。

«1» 英文字母：a~z，A~Z

«2» 数字：0~9

«3» 空白符：空格符、制表符、操作符等。

«4» 特殊字符：①标点符号 ②转义字符（表2-1）→<即下演示>

(2) 标识符

在程序设计中有许多需要命名的对象，以便在程序中其它地方便用。如何表示在不同的地方便用同一个对象？ <提问>

解决方法：为对象命名，通过名字在程序中建立起它与使用的关系。 <互动> <问题解决>

C语言规定：标识符只能由字母（A~Z，a~z）、数字（0~9），下划线（_）组成的字符串，并且第一个字符必须是字母或下划线。

注意4点：(1) C语言中标识符严格区分大小写。 <讲解>

(2) ANSI C标准规定标识符的长度可达31个字符

(3)、标识符命名应"见名知义"

(4) 变量名都要"先定义，后使用"

(3) 关键字（保留字）

ANSI C定义的关键字共32个，根据关键字的作用，可将其分为数据类型关键字、控制语句关键字、存储类型关键字和其它关键字四类。

打展关键字：19个

(4) 注释符 /* */

★ 2.2. 数据类型

<打比方讲解>

现实物品要分类，人要区分性别一样，在程序中使用的数据也要区分类型，...区分类型的目的是便于对它们按不同方式和要求进行处理。在C程序中，每个数据都属于一个确定的基本数据类型。

<树形图>
- 数据类型
  - 基本类型
    - 整型
      - 基本整型
      - 长整型
      - 短整型
      - 无符号整型
    - 实型(浮点型)
      - 单精度实型
      - 双精度实型
    - 字符型
    - 枚举类型
  - 构造类型
    - 数组类型
    - 结构体类型
    - 共用体类型
  - 指针类型
  - 空类型

<书或PPT>

(1) 整型

<讲解>

★ 整型数据是通常使用的整数，分为带符号整数和无符号整数两大类。 类型说明符：int 例如：int a,b;

★ 整型数据在内存中的取值范围

如课本P23，表2.2，VC++6.0环境中各数据类型占字节数。

★ 整型数据的表示形式：

十进制整数：举例：254, -127, 0 ✓

0291, 23D ✗

<书中举例>

八进制整数：举例：021，-017 ✓

256，03A2 ✗

十六进制整数：举例：0x12，-0x1F ✓

(2) 实型

★ 基本类型定义：float，double float，long double

★ 取值范围 表2.3 → <PPT演示>

★ 表示形式 { 十进制形式 / 指数形式

(3). 字符类型

★ 基本类型定义：char

★ 存储与取值范围：对应字符的ASCII值 表2.4 → <PPT演示>

★ 表示方法：字符型数据和整型数据可以通用

2.3. 常量与变量

C语言处理的数据包括常量和变量两类。对于常量来说，属性由其取值形式表明，而变量的属性则必须在使用前明确加以说明。 <讲解>

(1) 常量

★ 数据类型及实例：表2.5

★ 表示方法：(1) 数值常量

(2). 字符常量

(3). 字符串常量

(4) 常量的长度，值域根据类型不同而不同

(5) 字类型无常量，无设计类型

(2) 变量.

有关概念：

(1) 在程序运行过程中，其值可以改变的量，称为变量.

(2) 先定义后使用

(3) 系统为变量分配存储单元

(4) 使用时通过变量名来存取变量值

x — 变量名

98 — 变量值

为变量分配的存储单元

★ 变量如何定义：

[类型修饰符] 数据类型 变量名;

★ 允许在说明变量时对变量进行初始化：

例如：int a=5, b=10+2;

<举例子>:

int i, j;

long k, m;

float x, y;

char ch1, ch2;

★ 变量的初始化 课本[例2.1] — 学生程序演示

实型数据与舍入误差 [例2.2]

字符型数据与整型数据可通用 [例2.3]

转义字符的使用 [例2.4]

2.4 指针类型

重点：知识点引入：指针类型是C语言的特点之一，也是本课程的重点之一. 但是对于这种抽象的概念不好讲好理解. 此处举例子便于学生快速的理解下指针概念.

<对重点着重讲解>

举例：例如找人，我们现在需要找张茹这个教师. <举例>

有很多种方法. ① 直接找 (盲目性太强)

② 间接找 (比如可以先查到张茹老师所在的院系，然后去此院系找到其联系方式，然后打电话找到张茹老师) (快速，精确)

后者就体现了指针的特点

(1) 指针和指针变量的概念.

(1). 变量的地址与变量的内容.

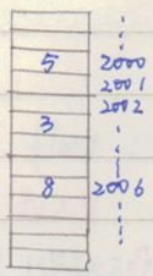
(2). 直接访问(寻址)与间接访问(寻址) <PPT演示>

直接寻址:

利用变量名

存取数据的方式

称为"直接寻址"



间接寻址:

(2) 指针变量的定义:

形式: [类型修饰符] 数据类型 *变量名列表 <课本>

例如: int i, j; /*定义整型变量*/

int *p1, *p2; /*定义指针变量p1,p2*/

指针变量的引用:

(1) * : 指针运算符

(2) & : 取址运算符

2.5. 运算符和表达式

运算符的优先级和结合性 表2.6 → <PPT演示>

运算符也称操作符. 是一种对数据进行某种运算处理的符号. C语言编译器通过识别不同的运算符, 完成各种算术运算. 逻辑运算. 位运算等运算.

★ 类型: 按运算对象分: 单目. 双目. 三目

按功能分: 算术. 赋值. 关系. 逻辑

条件. 逗号. 位. 其他.

优先级：指各种运算符号运算优先顺序 （表2.6）

结合性：指运算符号和运算对象的结合方向

表达式的有关概念：

问题1：什么是表达式？

问题2：表达式如何进行书写？ ——〈举例〉——→ b=(++a)-2

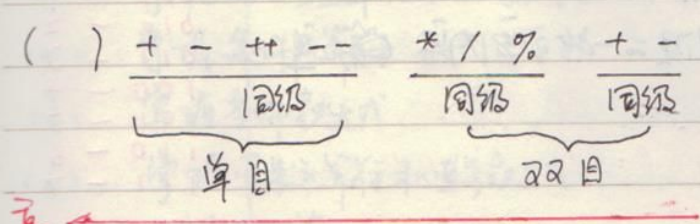
a/=a*=(a=2)

f=a>b>c

--a||++b&&c++

算术运算：

(1) 算术运算符的优先级：



〈板书〉

三个注意：(1) 没有乘方运算符 → a^3 ×

(2) "/" 可为各种数据类型，两数相除，结果也为该型

(3) "%" 要求运算对象必须是整型数据。

(2) 自增自减运算：++是单目运算符。 ★★重点内容讲解★★

两种形式 { 前缀形式 ++a ⟺ a=a+1
后缀形式 a++ ⟺ a=a+1 〈板书〉

〈问题〉例：当a=5时 〈学生思考〉

① ++a a=6 表达式值为6 〈互动〉

② a++ a=6 表达式值为5

③ b=++a ⟺ a=a+1; b=a

a=6, b=6 表达式值为6

④ b=a++ a=6, b=5 表达式值为5

注意：自增、自减运算只能用于变量，而不能用于常量或表达式！

例如：5++，(a+2)++ 不合法

自增自减运算符具有结合性，方向为右→左。

课本例题2.6 讲解 —— 观场编辑 → 〈程序演示〉

4. 单元教学总结：（课堂总结）

(1). 回顾知识点：（启发学生与老师一起回顾本章课堂的讲内容）

(2). 简短介绍下次课一些主要内容（第二章下半部分）

5. 作业布置：

八进制对应表：

000 — 0
001 — 1
010 — 2
011 — 3
100 — 4
101 — 5
110 — 6
111 — 7

十六进制对应表：

0000 — 0
0001 — 1
0010 — 2
0011 — 3
0100 — 4
0101 — 5
0110 — 6
0111 — 7
1000 — 8
1001 — 9
1010 — A
1011 — B
1100 — C
1101 — D
1110 — E
1111 — F

补充知识：1. 各种进制之间的转换（重点）

每一位所具有的值 ↑

①二进制、八进制、十六进制 转换 十进制：方法：按权相加

②十进制转换成二进制、八进制、十六进制

方法：连续除以基，从低到高记录余数，直到商为0为止。

③二进制与八进制之间的转换

二 → 八：从右向左，每3位一组（不够3位左补0）转换成八进制

八 → 二：用3位二进制表示每一位八进制数

例：(1101001)2 = (001, 101, 001)2 = (151)8

(246)8 = (010, 100, 110)2 = (10100110)2

④ 二进制与十六进制之间的转换

二 → 十六：从右向左，每4位一组（不够4位左补0）转成十六进制

十六 → 二：用4位二进制表示每一位十六进制数

例：(1101010111111101)2 = (0011, 0101, 0111, 1101)2 = (357D)16

单元教学　　第3次课

授课课题：第二章 数据类型和表达式（二）
授课时间：
授课类型：理论课
授课学时：2学时

一. 教学目标、要求：
(1) 掌握关系运算符和关系表达式
(2). 掌握逻辑运算符和逻辑表达式
(3). 掌握赋值运算符和赋值表达式
(4). 掌握条件运算符
(5) 掌握条件表达式
(6) 掌握逗号运算符和逗号表达式
(7) 了解位运算
(8) 了解各种数据类型间的转换

二. 知识点：
关系运算符. 关系表达式. 逻辑运算符和逻辑表达式
赋值运算符. 赋值表达式. 条件运算符.
条件表达式. 逗号运算符. 逗号表达式. 位运算
数据类型间的转换

三. 教学重点：
1. 几种运算符和表达式的相关概念及应用：本章中涉及了很多运算符和表达式，区分它们的相关概念，符号用法是本章一个重点内容. 多举例子进行反复的讲解

2. 自增自减运算的理解：（上次课简要）自增自减运

算是在程序设计当中经常使用的一个运算，理解并熟练掌握它的用法和计算是本章的难点之一。

四．教学难点：

1．各种运算符和表达式优先级和结合性是难以记忆是本章的一个难点。

解决方法：当堂举例，让学生多做例题。

另外告诉学生要配合一些记忆，并且在今后的程序设计当中不断的强化。

2．本章内容较为繁琐和枯燥，这是本章教学当中的一个难点。

解决方法：本章内容设在这门课程的第二章，对于初次接触这门课程的非计算机专业学生来讲比较枯燥的，不利调动起学生的兴趣，可以考虑"从例子到概念"的这种方式来进行讲述，让学生不要一下子面对繁杂的概念，通过例子便于学生接受。

五．能力培养

本章内容较为枯燥，但它确是C语言语法学习的基础，是程序设计、编码的基础，本章内容要求学生边学习边掌握，可能一下子记忆不了全部的内容，教师应多通过例题进行讲解，并在今后的章节学习中不断强化本章的基础内容。

六、单元教学过程：

1. 回顾上节课内容：（第一章前半部分内容回顾）

数据类型，标识符，常量与变量，指针与指针变量，运算符，表达式，算术运算符，算术表达式，自增自减运算。

2. 新课引入：

重点回顾下上次课中2.5.2节算术运算符和算术表达式，其中的自增自减运算这节课的重点内容，需要再次讲述，着重强调，再给出一个习题，请同学上台板书并解答。

题目（课后习题2.(2)）：已知i=0，j=1，k=2，则逻辑表达式 ++i||--j&&++k 的值为_____。

学生解答后进行点评并讲解此题。

3. 新课讲授：

2.5.3 关系运算符和关系表达式 〈讲解〉

[引]：关系运算实际上是比较运算，即将两个值进行比较，根据两个值比较运算的结果给出一个逻辑值（即真假值）。C语言没有专门提供逻辑类型，所以借用整型、字符型和实型来描述逻辑值。逻辑真为“1”，逻辑假为“0”。

“0”代表“假”，非“0”代表“真”。

★ 关系运算符（左结合） 〈板书〉

> >= < <=　　== !=

较高　　较低

★ 关系表达式：用关系运算符将运算对象连接成的式子。

例如：12<'c'+1（字符型按照比较ASCII值

a == b >= c 等价于 a==(b>=c)

与 (a==b)>=c 不等价

注意：(1) 关系运算符等价于数值，低于算术

(2) 关系运算的结果应该是逻辑值，C语言用数值

用1表示逻辑真，0表示逻辑假。

例如：7>5的值为1，5>7的值为0 <让学生思考>

'a'>'b'的值为0，'0'<'b'的值为1

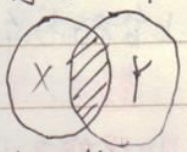
所有关系表达的值：0 or 1

(3) 实数型可进行大于或小于比较，但通常不进行

== 或!= 的关系运算

<互动>

与 X&&Y



或 X||Y

非 !X

2.5.4. 逻辑运算符和逻辑表达式

▲ 三种逻辑运算符：&& || ! →

▲ 逻辑运算符的运算规则：<PPT演示>

▲ 逻辑表达式：用逻辑运算符将运算对象连接成的式子

例如：0&&'b'

a&&b||c&&d

a||b-5||c/4

!x+y>=z

<讲解>

▲ 逻辑运算符的优先级和结合性：

(1) ! 是单目运算符，右结合，高于算术

(2) && 和 || 是双目运算符，左结合，高于赋值运算符，低于关系运算符。

▲ 从左到右依次进行逻辑计算

(1) 运算对象为非0表示逻辑真

(2) 运算对象为0表示逻辑假

▲ 逻辑运算的结果为 0 或 1

例如：设 a=15，b=0，c=-2 <简单>

a && b && c 结果为0 <思考>

a || b || c 结果为1 <互动>

(a+c) || b && c 结果为1

▲ 运算按照从左至右的顺序进行，一旦能够确定逻辑表达式的值，就立即结束运算

——逻辑运算的短路特性

例如：设 a=1，b=0，c=-2

a && b && c <稍作思考>

└ 为0，运算终止，表达式值为0 <互动>

(a++) || ++b && --c

└ 为非0，运算终止，表达式值为1

且a为2，b为0，c为-2（bc保持原值）

▲ 关系与逻辑运算符的应用

(1) 表示数学公式 a>b>c ：a>b && b>c

(2) 判断a,b,c三条线段能否组成一个三角形：

a+b>c && a+c>b && b+c>a

(3) a,b不同时为负

a>=0 || b>=0

!(a<0 && b<0)

(a<0 && b>=0) || (a>=0 && b<0) || (a>=0 && b>=0)

2.5.5 条件运算符和条件表达式

▲ 条件运算符：? : <相当>

条件表达式一般形式：表达式1 ? 表达式2 : 表达式3

例如：m<n ? x : a+3

a++>=10 && b-->20 ? a : b

x=3+a>5 ? 100 : 200

注意：C语言中唯一一个三目运算符，要正确区分用？和：分隔的表达式

★ 条件运算符的优先级

条件运算符优先级高于赋值、逗号运算符，低于其他运算符。

例子：(1) m<n ? x : a+3 <思考>

<==> (m<n) ? (x) : (a+3)

(2) a++>=10 && b-->20 ? a : b

<==> (a++>=10 && b-->20) ? a : b

(3) x=3+a>5 ? 100 : 200

<==> x=((3+a>5) ? 100 : 200)

★ 条件运算符的结合性： <讲解>

条件运算符具有右结合性

当一个表达式中出现多个条件运算符时，应该将位于最右边的问号与离它最近的冒号配对，并按照一般规则正确区分各条件运算符的运算对象。

例如：w<x ? x+w : x<y ? x : y

<==> w<x ? x+w : (x<y ? x : y)

<==> (w<x ? x+w : x<y) ? x : y

★ 条件运算符的结合性：

课本[例2.7]判断奇数、正数。 <程序演示>

（当堂上机程序演示）

2.5.6. 逗号运算符和逗号表达式 <书本>

★ 表达式1，表达式2，……，表达式n

● 逗号表达式之值：从左向右依次计算出各表达式的值，并将最后一个表达式的值作为逗号表达式之值。

例如：(1) a=5，a++，a*3，表达式值为18，且a=6 <思考>

(2) t=1，t+5，t++ 表达式值为1，且t=2 <互动>

(3) x=(a=3*5，a*4)

表达式值为60，且x=60，a=15

2.5.7. 赋值运算符和赋值表达式

★ 赋值运算符（右结合）

= += -= *= /= %=

&= |= ^= >>= <<=

★ 赋值表达式

- 将表达式之值存入变量对应的内存单元中

例如：m=12 <思考>

b=(++a)-2 <互动>

m%=3+n ⟺ m=m%(3+n)

x*=(x=5)

说明：(1) 赋值号左边必须是变量，右边可以是C语言任意合法之表达式。例：n=t+2<5 √；a+b=15 X <讲解>

(2) 赋值运算符的优先于“，”，且具有右结合性

例如：a=b=b*c>10

⟺ a=(b=((b*c)>10))

(3) 赋值号与数学中等号含义不同。

例如：数学中 a=b ⟺ b=a

C语言中 a=b <=> b=a

★ 赋值运算应用要分别

课本例2.8了 当堂演示程序. <程序演示>

2.5.8. 位运算符和位运算表达式 (引出内容) <引出>

引言: 数据在计算机里以二进制形式表示. 在某些问题中, 有一些数据对象的操作比较简单, 只需要一个或几个二进制位表示的编码表示. 如果去管这种数据用基本数据类型表示, 对计算机资源是一种浪费. 另外, 许多系统程序需要对二进制位表示的数据直接进行操作. 因此C语言提供了对二进制位的操作功能, 称为位运算.

位运算仅用于整型数据

~ & ^ | << >>

位逻辑运算符 移位运算符

位运算对象一律按二进制补码参加运算, 并按位进行运算.

(1). 位逻辑运算

按位取反: ~ 将"1"变为"0", "0"变为"1"

按位与: &

按位或: |

按位异或: ^

移位: 左移位: << ; 右移位: >>

课本例2.9了 演示程序. 编译. 成功结果

2.5.9. 其他运算表达式

(1). 取地址运算符 &

运算结果必须是已存储地址.

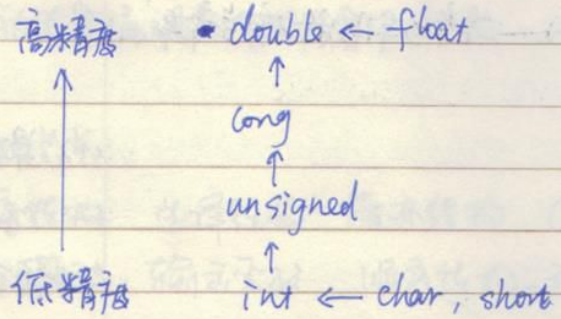
(2) 长度运算符 sizeof

运算结果只能是常量或该数据类型的长度.

2.5.10. 表达式的类型转换 (3简单内容) <3简单>

(1). 数据类型的自动转换

转换原则: 自动将精度低、表示范围小的运算对象类型向精度高、表示范围大的运算对象类型转换.



(2) 数据类型的强制转换.

一般形式: (类型名)表达式

(3). 赋值表达式中的类型转换

- 将实型数据(包括单、双精度)赋给整型变量时, 舍弃实数的小数部分
- 将整型数据赋给单、双精度变量时, 数值不变, 但以浮点形式存储到变量中.
- 将一个double型数据赋给float变量时, 截取其前7位有效数字, 存放到float变量的存储单元(32位)中, 但不可溢出.
- 字符型数据赋给整型时, 因字符只占1字节, 而整型

变量为2个字节，因此将字符数据（8位）放到1个低8位中。

例题：[例2.10] 混合运算表达式中的类型转换 <PPT演示程序>

4. 单元教学总结：（课堂总结）

（1） 回顾知识点：对本章知识点进行总结回顾。（启发学生与老师一起回忆本章的内容）

（2） 简短介绍下次课的一些学习内容（第3章 程序设计基础）——本章为本课程的重点章节。

5. 作业布置

（1）课后习题

（2）上机内容

（3）预习下节的内容。

单元教学　　第本次课

授课课题：第3章 程序设计基础（一）
授课时间：
授课类型：理论课
授课学时：2学时

一. 教学目标、要求
(1). 理解C语言程序的三种基本结构
(2) 掌握格式输入和输出函数
(3). 掌握字符型数据输入输出常用函数
(4) 初step掌握顺序结构程序设计方法.

二. 知识点：
顺序结构. 选择结构. 循环结构. C语句
输入函数. 输出函数. 顺序结构程序设计

三. 教学重点：
1. C语言程序的三种基本结构的相关含义区别
顺序结构. 选择结构和循环结构是C语言程序的三种基本结构. 本章课件3.1.1节当中对学生学习理解这三种结构.
2. printf() 函数与 scanf() 函数.
让学生初step掌握输入输出函数的内容与写法. 因为输出输入函数是每个程序设计当中应用都非常多的两个函数类型. 多举例让学生加强印象.

四. 教学难点：

1. C语句之初步掌握也本章之一个难点之一。

解决方法：C语句之编写也程序设计之基本。本章要求学生开始自己编写一些程序语句。老师在课堂上之程序演示让学生对一条条加深对于程序之印象。并能够自己编写。

2. scanf()与printf()函数使用之格式符的注意事项。

解决方法：结合课本为中之数3.1及数3.2的举例说明。例题尽量能覆盖两个函数中之内容。课后让学生上机多运行程序。加深学生对于两个函数之印象。

五. 能力培养

本章也会课程之重点章节。从本章之学习开始，要让学生开始着手自己上机进行程序编写了。课堂上要多进行程序演示为引导。课下让学生多做练习。对于程序设计开始有一定之认识。这也之本章内容对于学生能力培养上之基本要求。

六. 单元教学过程：

1. 回顾上节课内容（回顾第二章内容的问题）〈互动〉

数据类型和表达式. 词法. 数据类型（存型. 实型. 字符类型）常量与变量. 指针与相关. 指针变量. 运算符和表达式（算术运算. 关系运算. 逻辑运算. 条件运算. 逗号运算. 赋值运算. 位运算等）

（可以适当提问）

2. 新课引入

一个程序由若干篇章. 也由各种语句组成. 不同之处是章中之语句是用自然语言写成. 而程序中之语句是用某种计算机程序设计语言编写而成之.

C语言属于面向过程语言. 在这种语言中. 围绕一个程序目标所要采取之每步行动都必须由语句体现出来.

一个程序 { 数据 / 对数据之操作 } ⇒ 通过语句来实现.

本章是C语言程序设计之基础. 通过对语句之学习能让大家掌握一个简单之程序是如何编写之.

3. 新课讲授

3.1. 程序结构和语句 〈讲解〉

C语言是结构化程序设计语言. 最大之特点是以模块和结构为单位. 每个单位只有一个入口和一个出口. 程序之结构清晰明了. 可读性强. 从而提高程序设计之效率和可靠性.

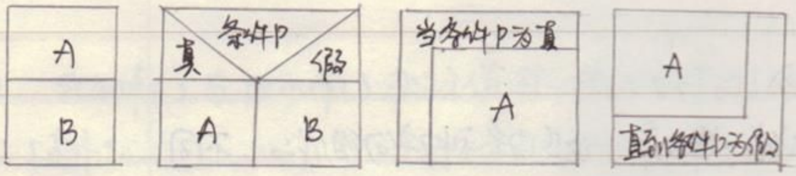
结构化程序由三种基本结构组成：{ 顺序结构 / 选择结构 / 循环结构 } ——〈板书〉

★ 顺序结构：程序按照语句的先后顺序依次执行

★ 选择结构：程序经过条件判断后，再确定执行哪一段代码段。

★ 循环结构：程序反复执行某一段代码，直到某种条件不满足时才结束执行该段程序结构。

<PPT演示>



顺序结构　　选择结构　　当型循环结构　　直到型循环结构

<问题提出> 什么是C语句？根据前面所学内容能否总结出C语句的概念？

<讲解> 一个程序
- 数据描述 — 定义数据结构
- 数据操作 — 对已提供的数据进行加工

<板书>

C语句：C程序对数据的处理和加工通过C语句的执行来实现。

C语言的语句分类：

C语言数据操作语句分类
- 简单语句
  - 表达式语句（表达式;）
  - 函数调用语句（函数名(参数表);）
- 特殊语句
  - 空语句（;）
  - 复合语句（{一条或多条语句}）
  - 注释语句（/*注释文本*/）
- 流程控制语句
  - 结构化
    - 条件语句（if、switch）
    - 循环语句（while、do-while、for）
  - 非结构化
    - 限定转向语句（break、continue、return）
    - 非限定转向语句（goto）

<PPT演示>

★ 说明性语句：

C程序由函数组成。函数的格式：

函数名（参数）

{ 说明部分 —— 按语法规则说明语句

执行部分 —— 可执行语句

}

举例：int a, b;

float function(int, int);

<讲解>

★ 简单语句：C程序中使用最频繁，语句。来自于一个表达式或者函数调用。结尾用分号就构成一个语句。

（注）：表达式后面加分号便构成了表达式语句

举例：x=3; y=y+5; ⟩ 赋值语句

x=a-b && c || d;

printf("x=%d, y=%d\n", x, y);

sort(a, 10); —— 函数调用语句

赋值语句三种形式：

<讲解>

① 简单赋值：变量=表达式 例：x=2*y+1; s=sqrt(s);

② 多重赋值：变量1=变量2=……=变量n=表达式

例：a=b=c=2; i=j=k=m+1;

③ 复合赋值：变量双目运算符=表达式

例：sum+=i; ⟺ sum=sum+i;

特殊语句：空语句、复合语句都属于特殊语句

↳ 语句只有一个分号，函数空调用

复合语句：用一对花括号"{ }"括起来的若干条语句

例3：(1) if(a>b){max=a; min=b;}
(2) for(n=1; n<10; n++)
{ p=n+p;
if(p>=100)
{ printf("%d\n", p);
break;
}
}

<举例说明>

复合语句中如有说明性语句，应该写在可执行语句前面。

例如：main( )
{ int a, b;
a=b=100;
{ float c=10.23;
printf("%f\n", c);
}
printf("%d %d\n", a, b);
}

<此处讲解花括号的使用。在括号内定义变量和C语言学习的建立->

控制语句 { 选择结构语句 → if()…else … -
循环结构语句 → for(); while(); --
其他语句 → break; continue; goto …

<讲解>

程序设计的步骤：
(1) 分析问题
(2) 确立算法
(3) 编写程序
(4) 调试运行

<讲解>

3.2. 标准输入与输出.

标准输入/输出是程序的基本功能. 它为程序运行中与用户交互提供基础. C语言没有自己的输入/输出语句, 要实现标准输入/输出功能, 必须调用标准输入/输出库函数, 这些函数在头文件"stdio.h"中定义. 因此, 在使用标准输入/输出函数之前, 要用预编译命令"#include"将该文件包含到源文件中.

格式: #include <stdio.h> <标准>

★ 格式输出函数 ★★重点内容★★

printf(格式控制字符串, 输出项表) <标准>

功能: 将各输出项的值按指定的格式输出到标准输出设备上. <讲解>

例3: int a=123, b=100; <例题>

```
printf("%d %d %d\n", a, b, a+b);
printf("c=%d+%d=%d\n", a, b, a+b);
```

★ 格式控制字符串

用双引号括起的字符串. 用于指定输出的格式类型. 格式个数. 包括: 普通字符 和 格式说明符 <教材或PPT>

```
printf("c=%d + %d = %d\n", a, b, a+b);
```

格式说明符: 指定输出3个十进制整型数. 分别为变量a, b, a+b的值.

printf(): 使用格式字符及其说明 (表3.1) <PPT演示>

例题: (不同数据类型的输出) <指导演示>

课本例3.2. (P57)

注意：(1) printf 函数格式控制中，格式说明符与输出参数个数和类型必须一一对应。 <讲解>

(2). 格式说明符必须以 "%" 开头，"%" 和后面的描述符之间不能有空格，除 %x、%E、%G 外其他描述符必须为小写字母。

(3). 格式控制字符串中，可包含转义字符。

(4). 不同的系统在实现格式输出时，输出结果可能会有一些小小差别。

★ 格式输入函数 ★★重点内容★★

scanf（格式控制字符串，输入项表）； <标书>

功能：按格式控制指定的格式，从标准输入设备（主输入缓冲区）将数据输入到对应的变量中（即将输入值赋给变量） <讲解>

例如：scanf（%d%f，&a，&f）；

scanf（%o，%f，&b，&x）； <标书>

scanf（"a=%d，b=%d"，&a，&b）；

★用双引号括起的字符串，用于指定输入数据的类型格式，个数为输入的形式

包括：普通字符 和 格式说明符 <标书改编>

↓ ↘ 照这样输入

scanf（"a=%d，b=%d"，&a，&b）；

↑ ↑

格式说明符：指定输入2个十进制类型数据赋给变量 a 和 b。

scanf( )函数使用的格式字符及其说明 (表3.2) <PPT演示>

例1：scanf("%d %d %d", &a, &b, &c); <讲解思考问的含义>

scanf("%d, %o, %f", &a, &b, &x);

scanf("a=%d, b=%d", &a, &b);

程序举例 <PPT：程序演示>

注意：(1) 要求在程序运行中输入数据，输入的数据个数和类型必须与格式说明符一一对应。 <讲解>

(2) 地址参数形式：&变量名(除数组或指针变量)

(3) 格式控制中有普通字符时，必须照原样输入

(4) 格式控制中无普通字符时，输入的数值型数据和字符串用空白符分隔，字符型数据不必分隔。

(5) double型数据输入时，必须用%lf或%le格式

(6) 实型数输入时域宽不能用m.n形式的附加说明

(7) 为了减少不必要的输入量，除了逗号、分号、空格符以外，格式控制中尽量不要出现普通字符，也不要使用'\n'、'\t'等转义字符。

(8) 格式符之间若无普通字符，则：

— 输入的数值型数据用空白符分隔

— 输入的char型数据不必分隔。

例：scanf("%d %d", &m, &n)

scanf("%c", &ch);

正确输入：32␣28a ↙

★ 字符输出函数 <讲述>

putchar(ch)

功能：在标准输出设备上输出一个字符.

例如：
```
putchar('b');
putchar('\n');
putchar('\101');
putchar(st);
```

说明：putchar是C语言为标准字函数. 使用时必须自加编译预处理命令：

#include "stdio.h" 或 #include <stdio.h>

[例] 3.3 利用putchar函数输出字符. <程序演示>

★ 字符输入函数

getchar( )

功能：从标准输入设备上交互输入一个字符.

例如：
```
getchar( );
c=getchar( );
printf("%c\n", getchar( ));
```

[例] 3.4 输入一个字符. 并输出该字符 <程序演示>

说明：getchar是C语言为标准字函数. 使用时必须加编译预处理命令：

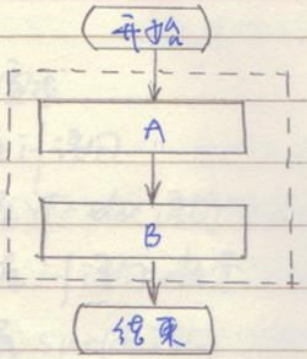
#include "stdio.h" 或 #include <stdio.h>

3.4. 顺序结构程序设计

顺序结构是程序设计语言最基本. 最简单的 <引子>

构. 也称线性结构. 程序中包含语句按照书写
次序依次执行.

特点: 一个操作执行完成后就紧接着执行紧 <讲解>
随其后的下一个操作.



[例3.5] 从键盘输入一个小写字母, 用大写字母形式输出 <板书
该字母. 演示>

[例3.6] 输入一个三位正整数, 然后逆序输出.

例1. 输入456. 输出654

4. 单元教学总结. (课堂总结)

(1). 回顾本节课知识点.

(启发学生本节课一起互动回忆本节课内容)

(2) 简短介绍下次课的主要内容.

下次课主要讲述3.4节选择结构程序设计.

这个课程的一个重点章节. 让学生预习

5. 作业布置.

(1) 课后练习

(2) 上机内容

(3) 预习

单元教学 第5次课

授课课题：第3章 程序设计基础（二）选择结构
授课时间：
授课类型：理论课
授课学时：2学时

一. 教学目标、要求
(1) 熟练掌握if语句
(2) 熟练掌握if-else语句
(3) 熟练掌握if语句的嵌套
(4) 熟练掌握switch语句
(5) 熟练掌握选择结构程序设计方法.

二. 知识点：
if语句. 单分支if语句. 双分支if语句.
switch语句. 选择结构的嵌套.
选择结构程序设计

三. 教学重点：
1. 单分支if语句与双分支if语句.
这两种分支if语句也是本章课内容的重点. 也是C程序设计的基础语句结构. 要让学生熟练掌握此部分的内容. 多举例并配合上机程序演示加以讲解. 另外实践上机课也应该涉及到这部分的习题.

2. 对于switch语句的理解和掌握
switch语句是用于解决多分支问题. switch

语句也是C语言程序设计中较常使用的语句结构，要求学生熟练掌握其语句结构，并能自己写出一些简单的程序。例如经典的"输出星期"问题，"超市购物打折扣"等简单问题。

四、教学难点：

选择结构的嵌套是本节课的难点。

解决方法：选择结构的嵌套包含了"if语句的嵌套"、"if与switch语句的混合嵌套"及以"switch"语句的嵌套三种嵌套方式。对于每一种嵌套方式，不要一上来就讲述概念和语法上总结出来的嵌套格式，而是先以一个相关例子来让学生自己总结这种嵌套方式的形式写法。这样有助于学生对于这三种嵌套方式的记忆与区分。

五、能力培养

(1)、分析问题，找出选择结构并设计算法，利用选择语句，解决实际程序问题。

(2)、学习如何使用调试器调试，初步调试程序，能够分析程序错误并找出解决方法。

六、单元教学过程

1. 回顾上节课的内容

上节课主要学习了选择结构程序设计，其中学习了if选择结构（单分支if语句与双分支if语句），switch选择结构，以及选择结构的嵌套。 〈互动〉 〈适当提问〉

2. 新课引入

在解决实际问题时，经常遇到选择的问题：当实现现实事物满足不同条件时，会有不同的结果出现。

举例：(1)某一门课程考试成绩大于等于60分，该课程考试视为通过；如果考试成绩小于60分，视为不通过。

(2)解一元二次方程的根时，如果$b^2-4ac>0$，方程有两个不相等的实根；如果$b^2-4ac=0$时，方程有两个相等实根；如果$b^2-4ac<0$时，方程有两个共轭复根。

显然，要解决选择的问题，利用顺序结构程序设无法实现的。解决这类问题需要用选择结构程序来实现

选择结构是构成我们的三种基本结构之一，其可用来根据指定的条件是否满足，决定从给定的两个或多个操作中选择其中的一种来执行。

3. 新课讲授。

选择结构的程序设计 〈板书〉

- if语句
  - if语句的一般形式
  - if语句的嵌套
- switch语句
  - switch语句的一般形式
  - break语句

★ if 语句（if 选择结构）：

简单的if语句：if（表达式）语句 → 赋值语句 / 函数调用语句 / 输入输出语句 / 复合语句 / 空语句

↓

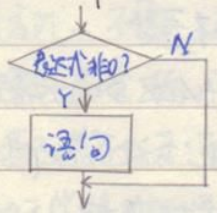
可以为算术、关系、逻辑、赋值等表达式

<讲解>

<重点内容>

功能：首先计算圆括号中表达式的值，如果表达式的值为真（非零值）时，则执行圆括号其后的语句，然后执行该语句后面的下一个语句。如果表达式的值为假（"0"），则跳过圆括号后面的语句，直接执行if语句后面的下一个语句。

简单的if语句的算法：



例如：(1) if (x>0) m++;

(2) if (a>b) {c=a; a=b; b=c;} → 计算机中实现变量a和b的值互换的正确写法。

[例3.7] [例3.8] } 程序源永远PPT讲解

双分支语句：

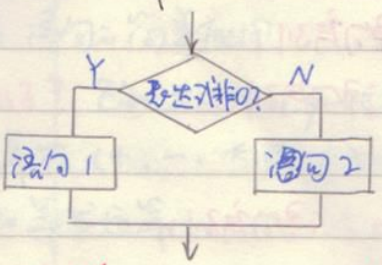
<重点内容>

if（表达式）语句1 else 语句2

功能：首先计算小括号中的表达式，如果表达式的值为真（非"0"），则执行紧跟其后的语句1，执行完语句1后，执行if-else结构后面的下条语句。如果表达式的值为假（"0"），则执行else

关键字后面的语句2.

双分支语句的算法：



例如：(1) if (x>0) m++; else m--;

(2) if (ch>='a' && ch<='z')
{ch=ch-32; printf("%c\n", ch);}
else printf("%c\n", ch);

[例3.9] 程序演示或ppt讲解

多分支语句：

if (表达式1) 语句1 <课本>
else if (表达式2) 语句2
...
else if (表达式m) 语句m
else 语句n

功能：依次计算并判断表达式i，为非0时执行后面的语句，都为0时，执行语句n。
无论执行完哪个语句分支，都转到后续语句。

[例3.10]. 成绩录入

关于if语句的几点说明（ppt）

★ switch 语句（switch 选择结构）

switch（表达式）　　<讲解>
{ case 常量表达式1：语句序列1
　case 常量表达式2：语句序列2　　<重点内容>
　……
　case 常量表达式n：语句序列n
　default：语句序列n+1
}

功能：计算表达式的值，与常量表达式的值比较，
等于第i个值时，顺序执行语句序列i，i+1，…
n+1

(2) 若与所有常量表达式值都不相等，执行语句序列
[练习3、11]"星期输出"　　<程序演示>

"case 常量表达式i："等价于语句标号，计算出表达式值等于哪个语句标号，就从哪个位置开始顺序往下执行语句序列

故，语句位置影响语句结果。

switch 与 break 语句结合才能实现程序分到。

```
switch (a)
{    case 1: printf("Monday."); break;
     case 2: printf("Tuesday."); break;
       ⋮
     case 7: printf("Sunday."); break;
     default: printf("error,");
}
```

关于switch语句的几点说明（ppt）

★ switch语句的简单应用

[例8] 已知x=100，y=15，要求输入一个算术运算符（+、-、*或/），并对x和y进行指定的算术运算。

思路：

(1) 设x和y为float型变量并且赋初值；

(2) 输入的运算符op为char型变量；

(3) 根据op的值（为'+'、'-'、'*'、'/'）进行x和y的相加、相减、相乘、相除运算（选择合适）；

(4) 还要考虑到输入字符不为+、-、*或/时的异常。

<由思路进行程序演示>

选择结构的嵌套

★ if（表达式1）

<对应伪码>

```
    if（表达式1_1）语句1_1
    else 语句1_2
else
    if（表达式2_1）语句2_1
    else 语句2_2
```

简单if语句的嵌套形式

```
if（表达式）
    if 语句 ——> 可以是各种形式的if语句
```

嵌套(或多重)合法if语句的一般形式

```
if(表达式)
    if 语句   → 若为简单if语句，必须用"{}"括起
else
    if 语句   → 可以这样的形式为if语句
```

[例3.1] (1).
```
if (c<=100)
if (c>=50) printf("50<=c<=100\n");
```
(2)
```
if (c<=100)
if (c>=50) printf("50<=c<=100\n");
else printf("c<50\n");
else
if (c<=150) printf("100<c<=150\n");
else printf("c>150\n");
```
(3)
```
if (c<=100)
if (c>=50) printf("50<=c<=100\n");
else printf("c<50\n")
```
↳ 思考：与哪个if配对.

<这里直接强调：提倡缩格书写有利于阅读程序>

[例3.1]：
```
if(a>b)
    if(a>c)
    if(a>d)  m=1;
    else     m=2;
    else     m=3;
```

<立新分>：
问题：哪一个else和哪一个if相匹配？
规则：在嵌套的if~else语句中，else总与上面最近的尚未配对的if相匹配.

[例3.14] 比较两个整数的关系。 <我会讲啊>

学习if语句的重点：

(1) if ~ else 语句的用法

(2) 正确使用表达式描述条件

(3) 正确判断执行的语句。

我总结常用的if表达式形式

[例如]：有定义 int a,b=0;

a等于什么值时，执行 b=2; 语句？

```
if (a==0) b=2;
if (a==1) b=2;
if (a!=0) b=2;
if (a=1) b=2;
if (a=0) b=2;
if (a) b=2;      当a为非0
if (!a) b=2;     当a为0
```

选择结构程序举例：

[例3.18]

[例3.19]

补充[例1]输入年份，判断该年是否为闰年。 <不会讲啊>

4. 单元教学总结 (课堂总结)

(1) 本节课知识点回顾.

(互动学习回忆知识点)

本节课主要学习了选择结构的程序结构

(2) 简单介绍下次课的内容.

下次课将学习课本3.3节 循环结构程序设计. 本节会给出重点小节. 也会给难点小节. 让学生预习内容.

5. 作业布置

(1) 课后习题

(2) 上机布置

(3) 预习内容

单元教学　　第6次课

授课课题：第3章 程序设计基础(三) 循环结构

授课时间：

授课类型：理论课

授课学时：2学时

一. 教学目标、要求：

(1). 熟练掌握 while 语句

(2) 熟练掌握 do-while 语句

(3) 熟练掌握 for 语句

(4). 熟练掌握 break, continue 语句

(5). 熟练掌握循环语句的嵌套.

(6) 熟练掌握循环结构程序设计方法.

二. 知识点：

while 语句. do-while 语句. for 语句.

break 语句. continue 语句.

循环语句的嵌套.

循环结构程序设计方法.

三. 教学重点：

1. 循环结构组成要素：分析问题如何进行程序设计的第一步. 在实际编写循环语句中, 如何正确找出循环规律. 这是此章重点掌握的内容.

2. while 语句：while 语句适于编写未知循环次数的循环, 这适用场合最多的循环语句, 这是此章重点掌握.

3. for语句：for语句是编写已知循环次数的最常用语句，学生应重点掌握。循环结构与上一节讲的选择结构如何结合解决实际问题，也是一个重点问题。

四. 教学难点：

1. 给定问题，脑子里有想法，不知道怎么按计算机的思路来解决，写出的程序达不到自己的目的。

解决方法：讲课重点放在思路上，针对问题进行分析，构建数学模型，理出算法并编程实现，结合可能出现的错误，演示学生"想当然编"程序的运行结果，分析解决。课后学习要强化实践。

2. 学会了单重循环，不会写多重循环

解决方法：首先分析多重循环问题，找出每重循环的规律和要素；其次，确定内外层循环之间的关联，最后用单步执行方式运行程序，先让学生看一下单重循环如何一步步执行的，然后再加上内层循环，再次单步执行程序，让学生看到执行的每一个步骤，以及变量的变化。

五. 能力培养

(1) 分析问题，找出循环规律并设计算法，利用合适的循环语句，解决实际问题。

(2) 学习强化如何使用调试器调试程序，能够找出语句中的bug并改正。

六. 单元教学过程.

1. 回顾上节课的内容.

上一节课学习了选择结构程序设计. 我们学习并实现了几个经典的程序案例. 如如说"考试成绩等级输出"还有"解一元二次方程的根"等. 这些问题我们都可以使用选择结构程序设计来实现. <讲解导>

但是有些问题. 比方说"重复循环"之类的问题. 仅使用选择结构方法. 编程实现起来就过于麻烦了.

那么针对这种需要有限次重复执行的程序段.

本节课我们引入"循环结构程序设计".

2. 新课引入

循环: 需要有限次重复执行的程序段. <讲解>

大自然中的循环: 日夜更迭、四季交替

生活中的循环: 循环播放mp3歌曲. 操场上一圈一圈跑步.

本节主要内容:

- 循环结构
  - 循环语句
    - while语句 ✓<重点>
    - do-while语句
    - for语句 ✓<重点>
    - continue与break
    - go to
  - 循环嵌套

<板书>

<讲重点>

举例

3. 新课讲授.

★ 循环结构的组成要素. <提示问题>

[问题]: 在操场上跑10圈 <学生思考>

分析问题: 这个问题很容易写出跑1圈的语句. 但

跑10圈也不是需要写10遍跑步吧呢？

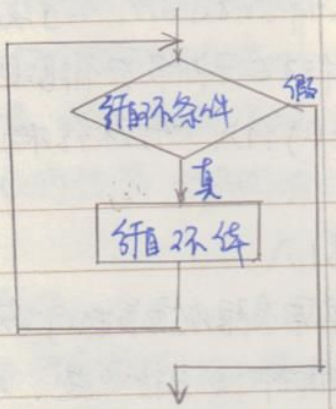
对于循环问题的关键也找规律.

有操场跑10圈 → 问题 <本书>

循环结构：
- 循环条件 — 是否跑了10圈
- 循环体 — 跑1圈
- 循环变量 — i从1→10

★ While 语句

```
while (表达式)
{
    循环体语句
}
```



<本书> <讲解>

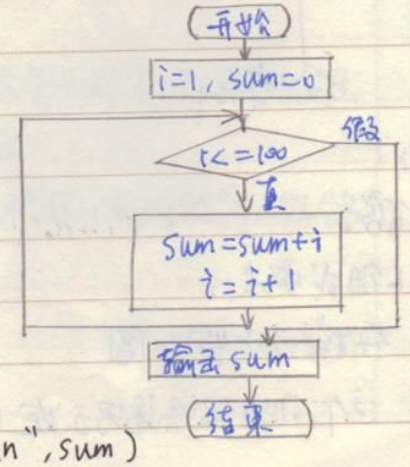
只要表达式的值为真（非0），就会执行循环体语句.

直到表达式值为假(0)时止.

（编写循环语句类似作填空题，要把循环要素填到相应的位置）

[程序示例7] [例3.20] 用while语句求1~100之累加和 <PPT>

```
#include "stdio.h"
void main( )
{ int i=1, sum=0;
  while (i<=100)
  { sum+=i;
    i++;
  }
  printf("sum=%d\n",sum)
}
```



相书 while 语句常见形式：

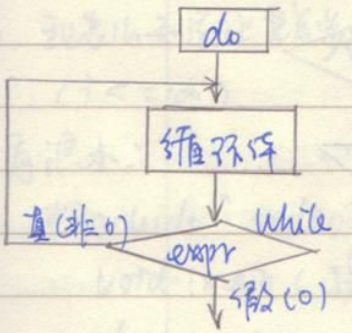
循环初始化：

```
while（循环条件）
{
    循环体；
    修改循环变量；
}
```

注意点：见PPT或课本为一准。

★ do-while 语句 （当型循环结构）


```
do
{ 循环体语句；
} while（总表式）；——分号在这里
```



特点：先循环，后判断

先执行循环体，后判断总表式

至少执行一次循环体

对比分析 while 和 do-while

（1）一对孪兄弟

（2）哥≤while 平淡，弟≤do-while 门事先试一下再说。

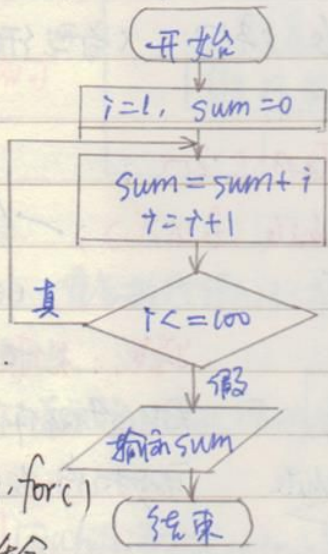
互动问题：输入密码，登录某系统，使用do-while还是while合适？

key：因为至少输入密码一次，所以更适合选用do-while

do-while 课堂练习：

用 do-while 语句求解 1～100 的累计和

```
#include "stdio.h"
void main( )
{ int i=1, sum=0;
    do
    { sum+=i;
      i++;
    }while(i<=100);
    printf("sum=%d\n", sum)
}
```



使用do-while语句：

说明：(1) 括号不能少

(2) 多个语句用大括号括起，形成复合语句

(3) 要有改变循环不变量的语句，以便结束循环

(4) do-while 没有while( ), for( ) 简洁，但有些场合较为适合。

★ for 语句 (for 循环结构)

```
for (exp1; exp2; exp3)
{
    循环体语句;
}
```

注意此处是“;”号。

格式：

for (初始化；循环条件；修改循环变量)
{
    循环体
}

<格式>
<执行顺序>

同样用for语句实现例3.22

用for语句求1~100之累计和

提问：

(1). 输入什么？（没有输入）

输出什么？（1~100之累加和）

(2). 有什么变量吗？变量什么语句？（有，sum=sum+i;）

(3). 预先知道变量多少次吗？循环变量i的初值、终值、步长？（预先知道变量100次，1，100，+1）

(4). 我怎么知道让变量步骤（什么时候）终了下来呢？循环条件？

（i<=100）

<互动>
引导学生分析问题，找到循环的规律和要点。

程序演示：

```
#include "stdio.h"
void main()
{ int i, sum=0;
  for(i=1; i<=100; i++) sum+=i;
  printf("sum=%d\n", sum)
}
```

<程序演示>

常见for语句编写错误：

[错误1]：分号改为逗号

<错误程序演示>

[错误3]：循环体内添加 i++ ；（多余i++）

for 语句的变化形式

形式一：
```
i=1;
for(;i<=100;i++)
{
    sum=sum+i;
}
```
<板书>
<讲解>

形式二：
```
i=1;
for(;i<=100;)
{
    sum=sum+i;
    i++;
}
```

★ break 与 continue 语句

提问：你在跑步上跑步，从不坐下圈一圈地跑下去，直到跑完10公里？能不能跑5公里累了就不跑了？或者中途休息下，喝点水，继续跑？

程序可以在循环体中退出或中断吗？

<互动><提问>
<引入break continue讲解>

break ；退出循环

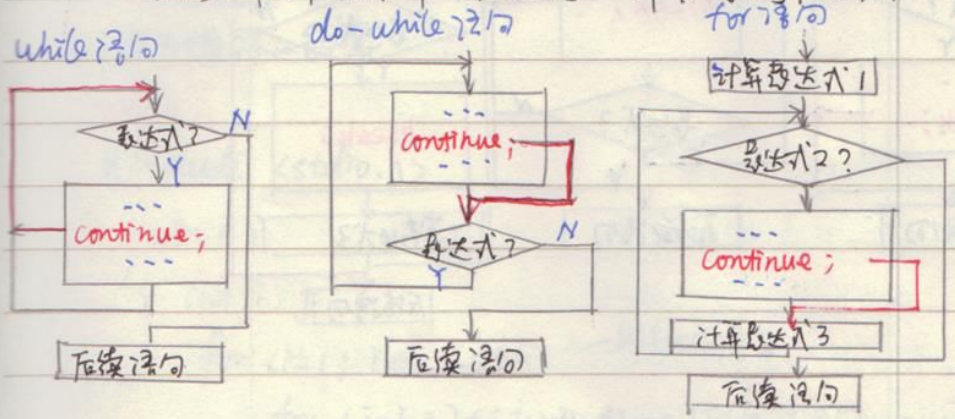
— 跑累了，完全休息

Continue ；中断此次循环体，执行下次

— 休息一圈再跑

★ continue 语句

功能：中断循环体本次执行，立即开始执行下一次循环



例如：(1) int x, n=0, s=0;

```
while (n<10)
{ scanf("%d", &x);
  if (x<0) continue;
  s+=x; n++;
};
```

(2) int x, n=0, s=0;

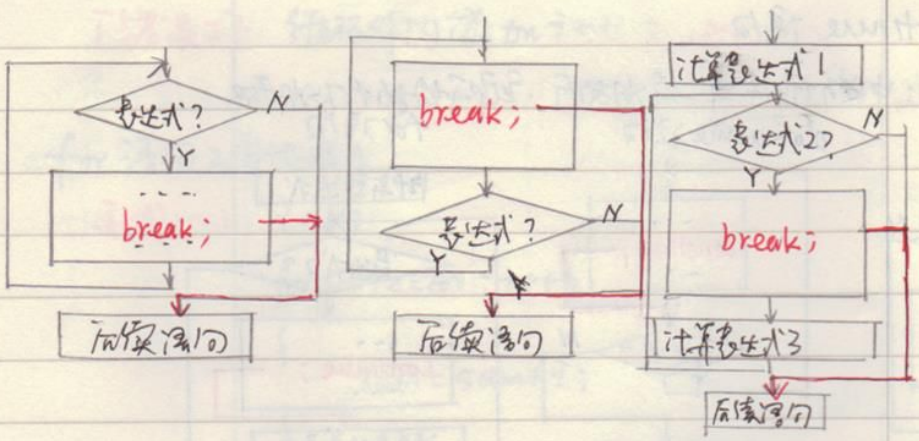
```
do
{ scanf("%d", &x);
  if (x<0) continue;
  s+=x; n++;
} while (n<10);
```

(3).

```
for(n=0, s=0; n<10; n++)
{ scanf("%d", &x);
  if(x<0) continue;
  s+=x;
}
```

★ break 语句

功能：利用break语句能够强迫终止本层循环，转到后续语句执行。



例:

(1)
```
int x, n=0, s=0;
while (n<10)
{scanf("%d", &x);
  if(x<0) break;
  s+=x; n++;
};
```

(2)
```
int x, n=0, s=0;
do
{scanf("%d", &x);
  if(x<0) break;
  s+=x; n++
} while(n<10);
```

(3)
```
for(n=0, s=0; n<10; n++)
{ scanf("%d", &x);
  if(x<0) break;
  s+=x;
}
```

★ goto 语句

(1) goto 语句的实现程序无条件转移. 与 if 语句一起构成循环结构, 为循环提供了便利. 但也无限制地使用 goto 会破坏程序的结构性和可读性.

(2). goto 语句可以随意从循环体中跳到循环体外, 或从循环体外跳至循环体内. 使用时要慎重, 考虑程序的逻辑结构. 建议只跳到循环体外!!!

★ 循环的嵌套.

如果循环语句之循环体内又包含了另一条循环语句.
则称为循环的嵌套.

例如:

```
#include <stdio.h>
main()
{ int i,j;
  for(i=1; i<10; i++)        一外循环语句
    for(j=1; j<=i; j++)      一内循环语句
      printf((j==i)?"%4d\n":"%4d",i*j);
}
```

注意: while. do-while. for循环语句可以并列. 也可以相互嵌套. 但要层次清楚, 不能出现交叉.

(2) 不同层次一般要采用不同之循环控制变量. 多重循环一般执行时. 外层循环每执行一次, 内层循环都需要循环执行多次.

例如:
```
for(a=1; a<=10; a++)
  { for(b=0; b<=5; b++)
      ------ }
```

外循环共执行了10次. 内循环共执行6次.

循环正常结束时. 内循环共执行了10×6=60次

[例1] 编程序. 输出以下图形.

```
* * * * * * *
  * * * * *
    * * *
      *
```

〈程序清单〉

〈上机程序调试〉

★ 指导设计的风格 <详细>

PPT演示

4. 单元教学总结：

(1) 本节课知识点回顾

(互动学习回忆知识点)

主要学习了循环结构与顺序结构，以重点章节

(2) 简单介绍下次课的内容

下次课将学习课本第四章数组。

5. 布置作业：

单元教学　　第7次课

授课课题：第4章 数组（一）
授课时间：
授课类型：理论课
授课学时：2学时

一. 教学目标、要求
(1) 熟练掌握一维数组的定义、初始化及引用
(2) 掌握二维数组的定义、初始化及引用
(3) 熟练掌握字符数组的定义、初始化及引用
(4) 熟练掌握字符串和字符串结束标志及字符串输入输出
(5) 了解字符串处理函数

二. 知识点
数组、一维数组、一维数组的引用、初始化
二维数组、引用、初始化、字符数组与字符串

三. 教学重点
1. 一维、二维数组的定义、初始化及引用。学生第一次接触数组的概念，下业对数组的概念定义。一维数组、二维数组的引用及初始化给予着重讲解，以便为下次课介绍数组的理解打基础。

2. 字符数组，由于C语言中没有相应的字符串变量存储，所以在C语言中，用字符数组存储字符串

对于字符数组与字符串，学生也应重点掌握。

四、教学难点：

不同的排序方法，字符串与一般字符数组的特征和使用方法之间的区别是本节课的难点所在。

解决方法：举例分别讲述一维数组、二维数组及字符数组的特征，用程序显示他们之间的使用方法与区别，让学生总结它们的区别，并用程序强化记忆。

五、能力培养

(1)

六. 第六章学习过程

1. 回顾上节课的内容.(回顾上章内容)

前一章我们学习了程序设计基础. ①学习了选择结构程序设计. 包括if语句. if-else语句. switch语句. ②学习了循环结构程序语句. 包括while. do-while. for. break. continue语句. 以及循环语句的嵌套. ③学习了利用以上程序选择. 循环结构进行程序设计.

2. 新课引入

★ 一个从N门课的成绩怎样存储和处理? <举例1:伴随>

★ 一个班N门课的成绩怎样存储和处理? ...

这些数据的特点. 具有相同的数据类型

为了方便的使用这些数据. C语言提供了一种构造数据类型: 数组

3. 新课讲授.

数组是个多值变量. 由一组同名但不同下标的元素构成.

一个数组的以同名存储在一块连续的内存中. 用数组名和下标就可以唯一的确定某个数组元素. <伴随>

只有一个下标的数组称为一维数组. 有两个下标的数组称为二维数组. 以此类推.

★ 一维数组的定义:

数据类型 数组名[整型常量表达式];

数组类型：定义数组表示数组类型

数组名：遵循C语言标识符规则

各维常量表达式：表示数组中有多少个元素，即数组的长度。

例如：float x[10];

数组定义中应注意以下几个常见错误：

(1) "int x[];" → 不能为空

(2) "int x(5);" → 应为方括号

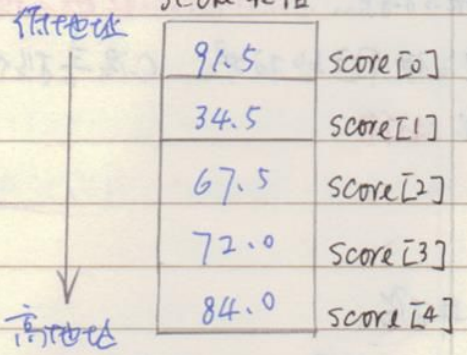
(3) "int n=4; int x[n];" → 方括号中给定为常量。

数组在内存中存放：

① 下标从0开始

② 顺序连续存放

③ score：值为score[0]之值。



★一维数组元素引用

格式：数组名[下标表达式]

例：输入学生成绩。 <教材>

```
for(i=0; i<5; i++)
    scanf("%f", &score[i]);
```

几个重点说明（见PPT） <PPT>

一维数组的初始化

初始化：在定义数组时，给数组元素赋以初值。

数据类型 数组名[常量表达式]={常量表达式,……} <书>

几种方式的初始化（见PPT） <PPT>

一维数组应用举例

[例4.2] 对数组赋值与输出

[例4.3] 输入5个数，找出最大与最小，对调，输出

[例4.4] 冒泡法排序

[例4.5] 选择法排序

<PPT演示>
<程序演示>

二维数组

二维数组的定义

数据类型 数组名[常量表达式][常量表达式]

规定了二维数组中一维数组的个数

规定了一维数组中元素的个数

例如：float x[3][3];

二维数组元素在内存中的排列顺序：按行存放。

可以把x数组看作是包含三个元素的一维数组，每个元素又是一个含有三个元素的一维数组：

x[0] — x[0][0], x[0][1], x[0][2]

↘ x[0]是数组名，是元素x[0][0]的地址。

x[1] — x[1][0], x[1][1], x[1][2]

↘ x[1]是数组名，是元素x[1][0]的地址。

二维数组的引用

数组元素表示形式：

数组名[行下标表达式][列下标表达式]

n行n列：（PPT）

二维数组的初始化

1. 按行给初值，将每一行元素的初值用一对花括号括起来。
2. 根据元素个数，由二维数组按行存储的规则，给出数组所有元素。
3. 给部分元素赋初值，例：int a[3][3]={{1},{4}}　1 0 0
4. 行数可省，列不能省，例 int a[][3]={1,2,3,4,5,6,7}　4 0 0

二维数组应用举例：

[例4.6]
[例4.7]
} <PPT>

字符数组与字符串

一维字符数组的定义：

char 数组名[整型常量表达式];

例如：char s[10];

可存放10个字符或一个长度不大于9的字符串。

二维字符数组的定义格式：

char 数组名[整型常量表达式][整型常量表达式]

例如：char a[3][5];

字符数组的初始化

1. 允许在定义时作初始化时赋值，逐个字符赋给数组中各元素。

   例：`char c[5]={'c','h','i','n','a'};`

   → 成为字符串

2. 若赋初值个数小于元素数据，则只将这些字符依次给数组中前面那些元素，其余元素自动被赋值为空字符（'\0'）。

   若大于，语法错误

3. 在初始化时省略元素个数。 例：`char s2[]={'s','t','r','i','n','g'};`

字符数组的引用

对一个数组元素的引用就是对一个字符变量的引用。

定义时可以直接赋初值。其他地方只能对逐个元素的引用。

字符串

1. 字符串常量。

   用双引号括起来的一串字符就是字符串常量。末尾将由系统自动加一个字符串结束标志'\0'。

   C语言中，没有专门的字符串变量。通常用一个字符数组来存放一个字符串常量。

2. 字符数组与字符串的区别

   · 字符数组中每个元素都可存放任意字符，不必非要放入一个字符'\0'。

   而字符串必须以'\0'为结束。

3. 在定义初值为字符数组赋字符串.

4. 输入和输出.

两种形：① 采用"%c"格式符.

② 采用"%s"格式符 { scanf() gets() } 区别

★ 字符串处理函数 { 说明① 说明② } <PPT> <PPT>

1. 字符串拷贝函数 strcpy() / strcpy(str1, str2)

2. 字符串连接函数 strcat() ↖ copy.

strcat(str1, str2)

3. 字符串比较函数 strcmp(str1, str2)

4. 求字符串长度函数 strlen(str)

5. 大写字母转换成小写字母函数 strlwr(str)

6. 小写字母转换成大写字母函数 strupr(str)

★ 字符串数组.

4. 单元教学总结

(1) 本节课知识点回顾

(2) 介绍下次课内容.

5. 布置作业.

单元教学　　第8次课

授课课题：第4章 数组（二）
授课时间：
授课类型：理论课
授课学时：2学时

一、教学目标、要求：
（1）熟练掌握指针与一维和二维数组的关系。
（2）熟练掌握指针运算
（3）熟练掌握指针与字符串
（4）了解指针数组
（5）理解多级指针

二、知识点：
指针运算　指针与一维数组关联
指向二维数组的指针　指针与字符串
指针数组　多级指针

三、教学重点
指针与一维和二维数组关联是本次课的教学重点，也是全课程的一个教学重点之一，应学生予以足够的关注，熟练掌握，是考试的重点内容。

四、教学难点
指向二维数组的指针变量的定义和理解是难点

解决方法：学日常生活中的例子来不断加深学生对指针及指针和相关概念的理解。除此以外，多练习编写。上机课让学生多上手做例题，找着感觉后，理解就容易了。

五. 能力培养。

六. 单元教学过程。

1. 回顾上节课的内容。

上节课我们学习了数组这一章前半部分内容。学习了数组的概念，一维数组，二维数组的定义，相关引用及初始化的概念。学习了字符串及字符数组。

2. 新课引入

C语言中数组与指针有着十分密切的联系。

3. 新课讲授：

C语言中数组与指针有十分密切的联系。数组名表示数组在内存的首地址。其类型为数组元素、类型*。指针可以用指针变量指向数组或数组元素，也可以把数组首地址或某一数组元素地址放到一个指针变量中。因为数组也由多个同种类型元素组成，并在内存中连续存放。所以任何能由数组下标实现的操作都可由指针来实现。运算通过指针访问对数组元素进行操作。

<讲述>

★ 指针运算 <板书>
- 算术运算
  - 指针变量加减运算
  - 自增自减运算
  - 两个指针相减运算
- 关系运算
  - < or >
  - == or !=

★ 指向一维数组的指针 <讲述>

① 一维数组地址：int a[10], *p;

a[0], a[1], ···, a[i], ···, a[9]

*(a+0), *(a+1), ···, *(a+i), ···, *(a+9)

② 一维数组指针定义 <板书>

int a[10];

int *p=a; /*指针为数组*/

③ 利用指针引用一维数组元素 <重点内容>

如果指针变量p指向数组a的第1个元素（即0号元素），即 p=a，则可以用p表示数组元素

下标法：p[0], p[1], ···· p[9]

指针法：*(p+0), *(p+1), ···, *(p+9)

注意：1. 通过指针访问数组元素时，必须首先让该指针指向当前数组，还需要使用下标或指针在数组中移动到相应元素（表4.2）

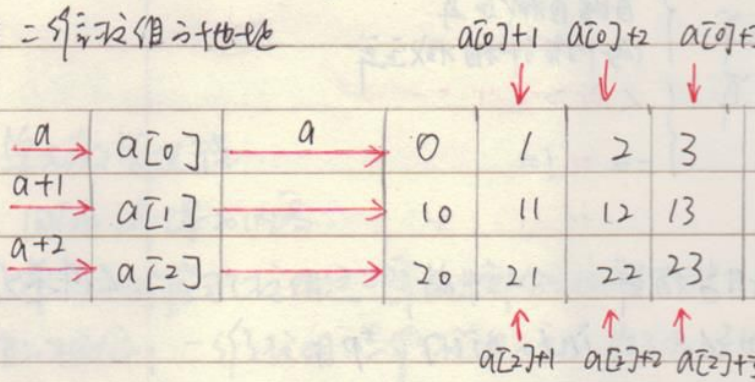
2. 使用指针引用数组元素时下标不能越界。

[错误] 4.10

[错误] 4.11

★ 指向二维数组的指针

① 二维数组元素地址



<重点难点>

<重点讲述>

例如：int a[3][4]

(1) a = &a[0]

(2) 二维数组a包含三个行元素：a[0]、a[1]、a[2]

地址分别为：a、a+1、a+2

即 &a[0]、&a[1]、&a[2] a+i = &a[i]

② 二维数组元素地址

a[i]+j

&a[i][j] = a[i]+j = *(a+i)+j

★ 二维数组元素的表示法

a[i][j]、*(a[i]+j)、*(*(a+i)+j)

二维数组a的各种表示形式的含义（表4.3）

★ 指向二维数组元素的指针变量

[例4.12]

[例4.13]

<指向源未>

★ 指向由n个元素组成一维数组的指针变量

数据类型 (*变量名)[整型常量]

(三个注意) (p可变未)

[例4.14]

<指向源未>

★ 指针与字符串.

char *变量名;

(1) 将一个字符串的首地址赋值给指针变量

```
char *p;
char s[] = "abc";
p = s;
```

(2) 将一个字符串常量赋值给指针变量

```
char *p;
p = "abc";
```

字符数组与字符指针的区别 {(1) (2)}

[5.1 | 4.15]

<指针演示>

★ 指针数组：数组中每个元素均为指针类型

数组类型 *数组名[整型常量表达式];

例如：char *a[3] = {"abc", "bcde", "fg"};

[5.1 | 4.16]

<指针演示>

★ 指向指针的指针

<指针内容>

数据类型标识符 **指针变量名;

4. 单元教学总结

(1) 本节课知识点回顾

(2) 介绍下节课内容

5. 作业布置.

学云教学 第9次课

授课课题：第5章 函数（一）（5.1～5.4节）
授课时间：
授课类型：理论课
授课学时：2学时

一. 教学目标、要求：
(1) 掌握函数的定义与分类
(2). 熟练掌握函数的调用
(3) 掌握函数的嵌套调用
(4). 掌握函数的递归调用

二. 知识点：
C程序结构 函数定义、分类、函数调用、
函数的嵌套调用、 函数的递归调用

三. 教学重点
1. 函数的声明、定义和调用
在C程序结构当中，函数是如何声明的、定义和
调用方式也是本节课的教学重点之一。

2. 函数的调用和例
函数的调用形式、调用方法也重点、在结合大量
的程序源示学生能够理解并掌握。

四. 教学重难点：

函数之间调用与递归调用

解决方法：函数之间调用与递归调用这节课我主要讲授（两个课）之教学难点。这是C语言设计中经常要使用的，也是一个非常重点的内容。必须让学生熟练掌握，深刻的理解。故先在课堂上多进行指导演示，配合上机实践。在这个知识点上多分配些上机实践之时间。多做例题，进行实践。

五. 能力培养

本模块化设计思路：用函数实现模块化程序设计

用"组装"的方法来简化程序设计之过程。这节课内容对于学生能力培养层面一个新要求。

六、单元教学过程

1. 回顾上节课内容

上节课我们学习了第四章数组的内容。学习了一维数组，二维数组，字符数组及指针数组。

2. 新课引入

<思考：为什么要用函数？ <互动>

通过前几章的学习，我们已经可以编写一个简单的C程序了。但是如果程序的功能比较多，规模庞大，把所有的程序代码均写在一个主函数中（main函数中），就会使主函数变得很长，头绪不清，使阅读和维护变得困难。另外，有时程序中要多次实现某一功能，就需要多次重复编写实现此功能的代码，使程序冗长，不精练。

因此我们想到利用“组装”的方法来简化程序设计的思路，如同组装计算机一样，事先准备各种部件（如电源、主机箱、硬盘等），在最后组装计算机时，用到什么就从仓库中取出什么，直接装上就可以了。

这就是“模块化程序设计”的思路。事先编好一些常用的函数来实现各种不同的功能，例如用sin函数实现求一个数的正弦函数值，用abs函数实现求一个整数的绝对值，把它们保存在函数库中，需要时，直接在程序中写上sin(a)或abs(a)就可以调用函数库中的函数代码，就不用再临时编写程序了。

3. 新得出程序

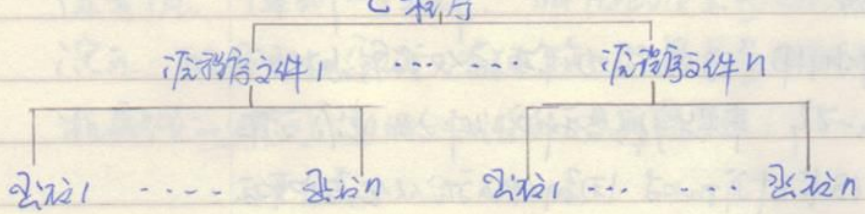
在程序设计过程中，为了方便组织人力共同完成一个复杂的任务，通常总是将任务划分成多个较小的子任务，每一个子任务都具有一定独立的功能，可以分别由不同的人员来编写和调试，在C语言中，实现相对独立的子任务的功能是通过函数实现的。

★ C程序结构

<详解>

逐步分解法（自顶向下逐步法）：把一个大问题分解成比较容易解决的小问题的过程。当然小问题还得由程序中若干个功能模块来实现。

★ C程序结构：



★ C程序源程序函数分类

(1). 从用户使用角度分类 { 标准函数 / 用户自定义函数

(2). 从函数参数有无分类 { 有参函数 / 无参函数

(3). 从函数有无返回值分类 { 有返回值 / 无返回值

(4). 从函数调用关系 { 主调函数 / 被调函数

★ 函数定义

无参函数定义格式：

<相对>

```
返回值类型 函数名( )
{
    说明部分
    语句部分
}
```

[例5.1] } 〈程序演示及说明课堂〉
[例5.2]

★ 有参函数的定义的格式

返回值类型 函数名（参数列表） 〈板书〉
{
说明部分
语句部分
}

[例5.3]

★ 函数返回值和return语句.

形式： return（表达式）；
return 表达式；
return；

return语句的几个重要的规定：
① 使函数立即退出程序去执行返回给调用者
② 可以向调用者返回值

[例5.4] 〈程序演示〉

几个重要的说明 〈见PPT〉

★ 函数调用和函数说明

无参函数调用的一般形式

函数名（ ）

有参函数调用一般形式：

函数名（参数表）

三种调用方式 ①
②
③ [例5.5]

函数原型说明一般形式：

返回值类型 函数名（参数类型表）；[例5.6]

几个重要说明 ①
②
③ [例5.7]

★ 函数调用的执行过程 <重点内容>

打比方：查字典过程 <重点讲述>

函数调用过程：

① 将实参的值赋给形参；

② 将程序执行流程从主调函数的调用语句转到被调函数函数体的开始部分，执行被调函数函数体的语句；

③ 当执行到被调函数函数体中的一个return或最后一个花括号时，将程序执行流程返回到主调函数的调用语句，如果调用语句包含表达式组成一部分，则将函数体的返回值参与表达式运算之后继续往下执行；如调用语句独立出一条语句，则直接继续往下执行。

④ 返回主调函数，带回返回值。

函数的嵌套调用和递归调用

函数的嵌套调用

<讲解>

函数定义不能嵌套，各个函数定义是相互独立的，但在一个函数内部都可以调用另外一个函数（不包含main()函数），这样一个函数调用另一个函数，即每一个函数又可以调用其他函数过程。这种形成了函数的嵌套调用。

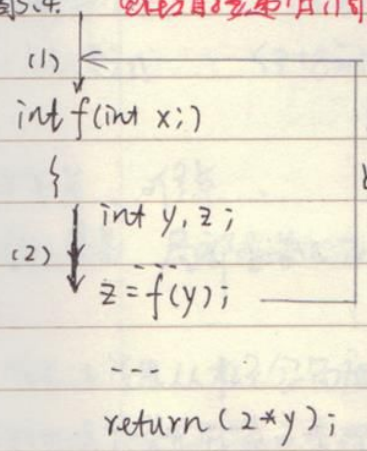
[例 5.8]

<程序演示>

函数的递归调用

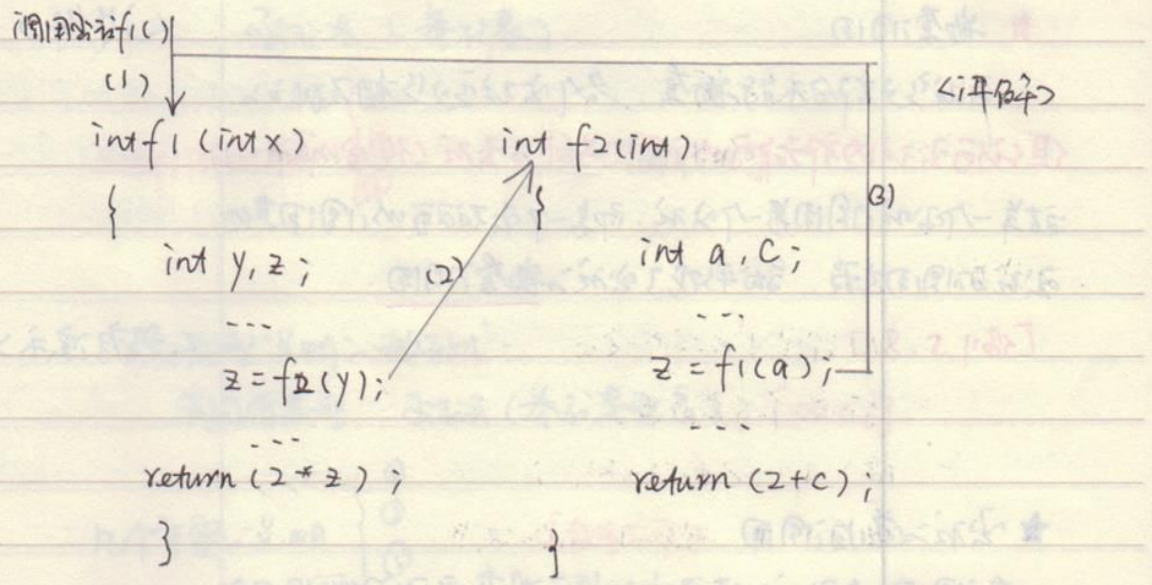
在调用一个函数的过程中如果出现直接或间接调用函数自身（除了函数main()外）的过程，称为函数的递归调用。

图5.4 函数的直接递归调用过程：



从图5.4函数执行过程中可以看出，在调用函数f()过程中，又调用函数f()，这种调用过程为直接递归调用。

图 5.5 两个间接递归调用过程



4. 算法的特点

(1) 有穷性和正确性

(2) 简要介绍下次课内容

5. 作业布置.

单元教学　　第10次课

授课课题：第5章公社（二）　5.5 - 5.6

授课时间：

授课类型：理论课

授课学时：　2学时

一. 教学目标（要求）：

（1）掌握局部变量和全局变量

（2）理解变量的存储类别

（3）理解内部公社和外部公社

（4）理解掌握公共体公社之间的传递

二. 教学知识点：

局部变量　全局变量　动态存储　静态存储

值传递　值地方式

三. 教学重点、难点：

全局变量　局部变量之区别？

解决方法：

首先让学生从概念上理解局部变量和全局变量，之后对比方法学生形象去去理解，并列举些例子进行程序演示，让学生的理解变量之作用域范围。

四. 能力培养

五. 单元教学过程.

1. 上节课内容回顾: 上节课学习了第5章函数和部分内容. 学习了C程序结构. 函数定义. 分类. 函数的参数调用和递归调用

2. 引课导入: <讲台车>

我们在学习本章之前见到的程序大多数是一个程序只包含一个main函数. 变量是在函数内部或外部定义的. 它的变量在本函数范围内是有效的. 在本章中见到了一些程序, 包含两个或多个函数, 那么各函数中定义变量, 有的同学自然会提出一个问题: 在一个函数中定义的变量, 在其他函数中能否被引用? 在不同位置定义的变量, 在什么范围内有效? 这就是变量的作用域问题. 每个变量都有一个作用域问题. 即在[哪]一些范围内有效.

3. 新课讲述:

5.5. 变量的作用域与存储方式

变量完整定义语句格式:

<存储类别> <数据类型> 变量名 [=初始值]; <板书>

- 变量的作用域
  - 局部变量: 在函数内部定义的变量, 形参及复合语句块中定义的变量都称为局部变量.

n个重要的范例 (PPT)

[5.11] <程序演示>

▲ 全局变量：又称作外部变量，全程变量，是在函数外部定义的变量，其有效范围为：从该变量的位置开始到本源文件的结束。

[例3.13] <程序演示>

★ 动态存储方式与静态存储方式

局部变量的存储方式 { 自动局部变量 auto
静态局部变量 static
寄存器变量 register

[例5.14] } <程序演示>
[例5.15]

全局变量的存储方式 { 外部存储全局变量 extern
静态全局变量 static

[例5.16] <程序演示>

★ 函数间的数据传递

在函数调用时，将由实际参数的值传送给对应的形式参数，或者由被调用函数向调用函数返回函数值。这些参数称为函数间的数据传递。

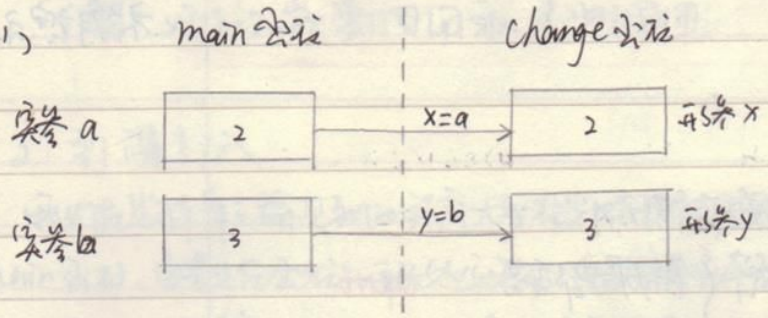
{ 值传递方式
地址方式

★ 传值方式传递特点.

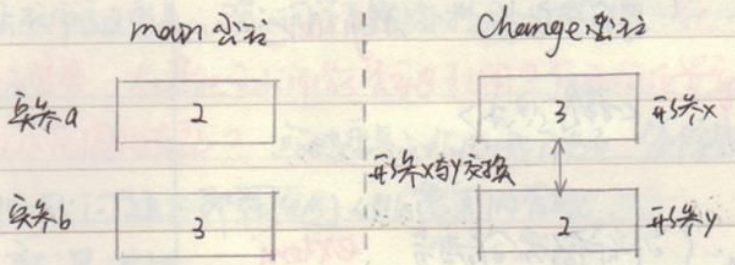
[例5.18]

程序执行过程中实参与形参的变化情况: <PPT或书本>

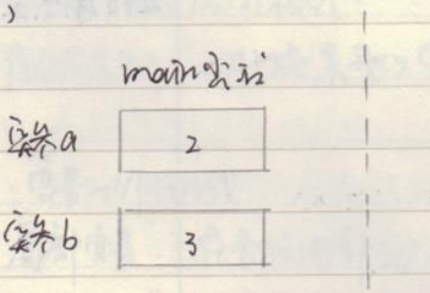
(1)



(2)



(3)



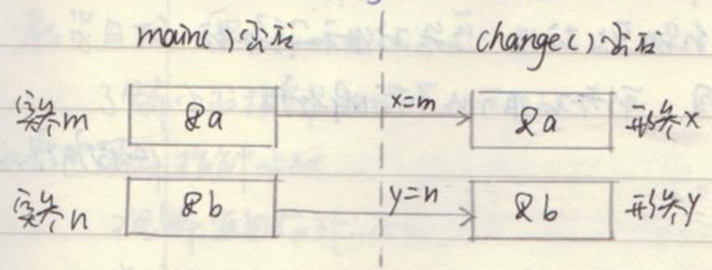
★ 传地址方式传递数据

1. 指针作为实参传递.

[例15.19]

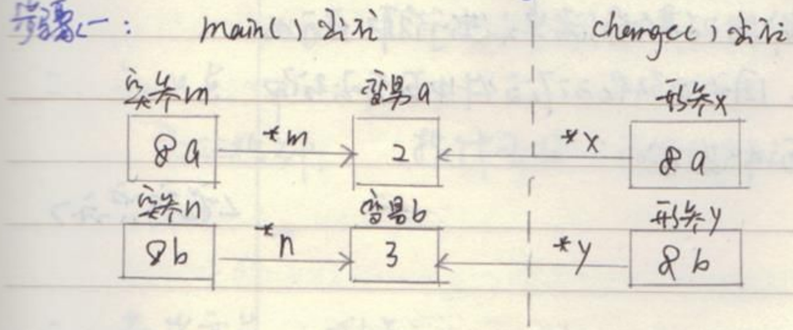
程序执行过程中，实参与形参的变化过程 <PPT或教材书>

(1) 主函数调用函数change()的过程

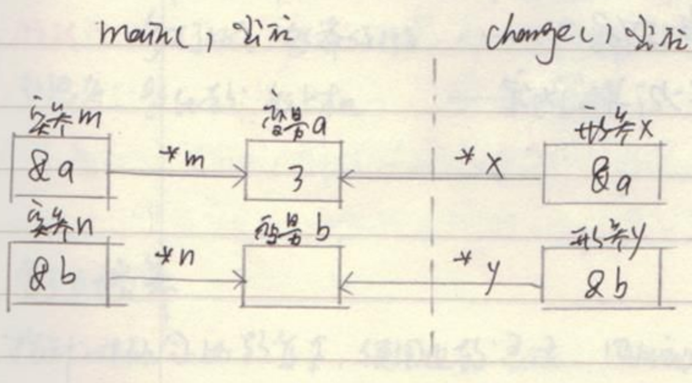


(2) 程序流程转到函数change()执行过程

步骤一:



步骤二:



2. 指针作为实参传递．

在C语言中．指针名代表了该指针在内存中的起始地址．当指针名作为函数参数时．实参与形参之间传递的就是指针起始地址．

说明：当指针作为函数参数时．在调用函数和被调用函数中需分别定义指针．实参指针和形参指针必须类型相同．形参指针可以不指明长度．

[例5.21]

<程序演示>

★ 利用全局变量传递参数．

若想让多个函数都能对某个存储单元进行存取．还可以采用全局变量的方式．因为对所在的源文件中所有函数而言．全局变量都是可以使用的．

[例5.22]

<程序演示>

4. 课堂小结

(1) 本节课知识点回顾

(2) 简要介绍下次课的内容

5. 作业布置．