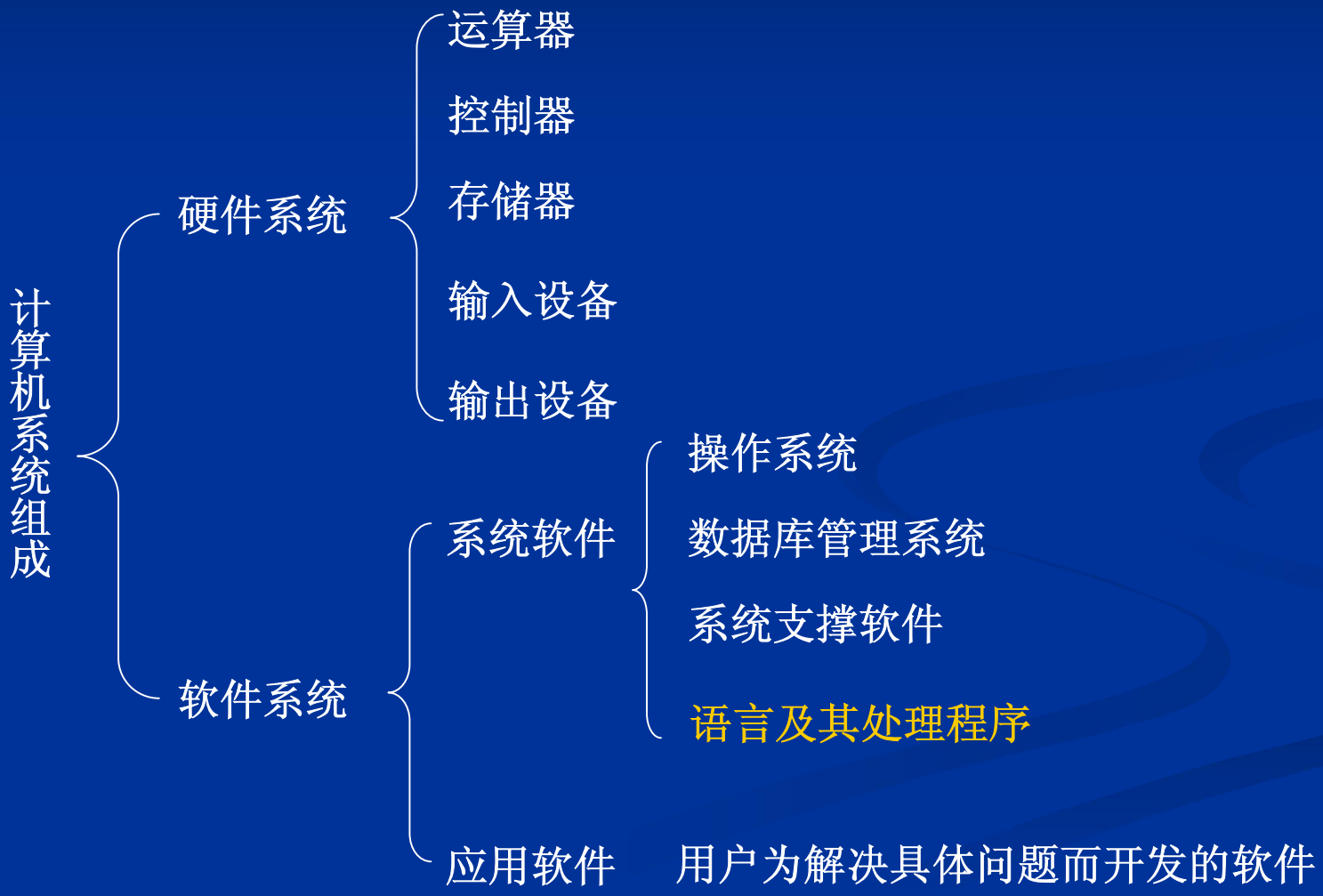
C语言程序基础

回忆计算机系统的组成


计算机系统组成
硬件系统
运算器
控制器
存储器
输入设备
输出设备
软件系统
系统软件
操作系统
数据库管理系统
系统支撑软件
语言及其处理程序
应用软件
用户为解决具体问题而开发的软件

为什么要学习程序设计？

- 计算机的本质是“程序的机器”。
- 程序和指令的思想是计算机系统中最基本的概念。
- 只有懂得程序设计，才能进一步了解计算机是怎么工作的。
- 通过程序设计，进一步了解计算机的工作原理，更好地理解和应用计算机。
- 通过掌握计算机处理问题的方法，培养分析问题和解决问题的能力，开展相关领域的应用，实现创新。

什么是计算机程序？

- 其实生活中也经常用到程序一词。常指完成某项事务的一套既定活动方式或活动过程。如：
  - 入党的组织发展程序
  - 图书馆借阅程序
- 计算机程序（通常简称程序）：是指为实现特定目标或解决特定问题而用计算机语言编写的一组计算机能识别和执行的指令序列。
  - 每种计算机都提供了一套指令，每种指令对应于计算机能执行的一个基本动作。
  - 为使计算机实现各种功能，需要若干程序。

什么是程序设计？

- 程序设计：是给出解决特定问题的程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。
- 程序设计过程应当包括分析、设计、编码、测试、排错等不同阶段。

什么是程序设计语言？

- 语言：通常指人类生活中使用的自然语言，如汉语、英语等。主要是人们交流信息的工具和媒介。
- 人和计算机交流，指挥计算机工作也需要创造一种计算机和人都能识别的语言。也就是说需要有一种适用的描述程序的语言。
- 供人编写计算机程序用的语言就是程序设计语言。

机器语言

- 二进制编码表示的计算机能直接识别和执行的一种机器指令集合。
- 优点：执行速度快
- 缺点：难学、难写、难记、开发效率低、难维护，且没有标准化

汇编语言

- 用助记符号描述的指令系统。如: ADD 代表“加”。
- 优点：比机器语言简单好记些
- 缺点：汇编语言机器不能直接执行, 必须翻译成机器语言，仍较难掌握。

高级语言

- 由于离计算机低层“很近”，所以把机器语言和汇编语言称为低级语言。
- 接近于自然语言和日常所用数学语言的计算机程序设计语言。
- 不能直接运行，需要“翻译”，转换为机器指令，通常的转换方式有解释和编译两种。
- 优点：便于理解和掌握

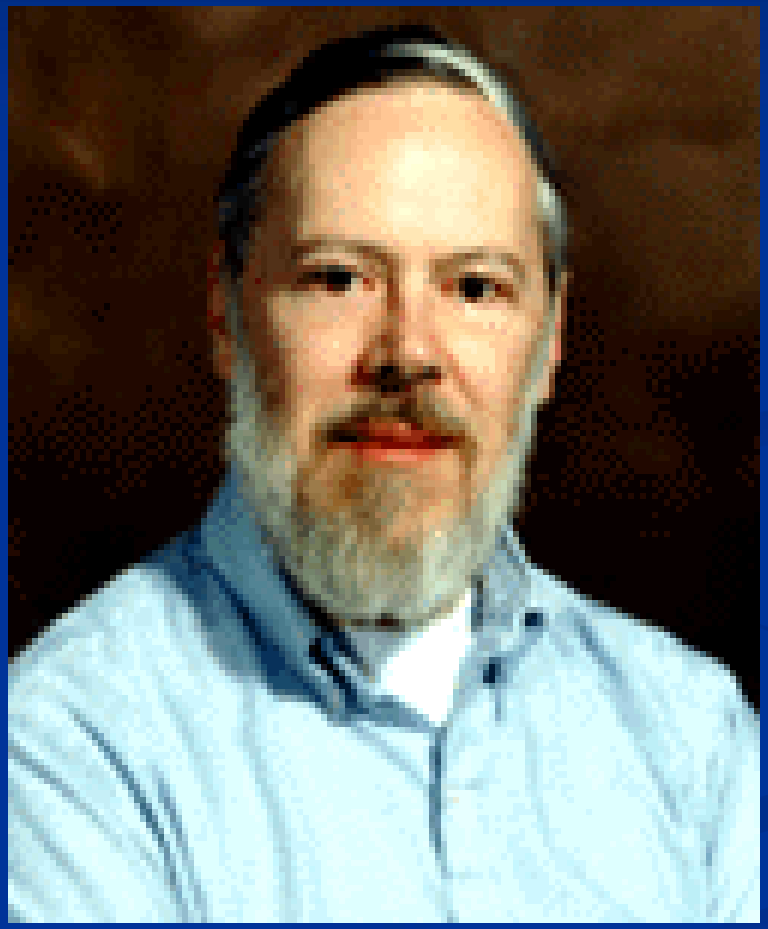
为什么选择C语言

- 进行程序设计，必须用一种计算机语言作为工具，否则只是纸上谈兵。
- C语言功能丰富、表达能力强、使用灵活方便、应用面广、目标程序效率高、可移植性好，既具有高级语言的优点，又具有低级语言的许多特点，既适于编写系统软件，又能方便地用来编写应用软件。
- 美国一专家对计算机专业学生的建议：
  - “大学生毕业前要学好C语言，C语言是当前程序员共同的语言。它使程序员互相沟通，比你在大学学到的‘现代语言’（比如Java，Python）都更加接近机器。”

C的起源

- 一切从一个叫“Space Travel”的电子游戏开始……
- 为了让他的游戏能在PDP-7上运行，Ken Thompson用汇编语言给PDP-7写了一个操作系统——UNIX
- 汇编使用起来不方便，Thompson需要高级语言
- 试验了一些高级语言，包括Fortran，都不理想
- 他在BCPL基础上，自己设计了一个B语言
- UNIX开始发展，B也不够用了
- 邀请Dennis Ritchie加入，把B改造成C
- 开始用C重写UNIX

C语言的祖师爷
Dennis M. Ritchie



Ken Thompson & Dennis Ritchie

C的发展

- C是高级语言，因其为编写UNIX操作系统而设计的，所以最初主要用于系统编程。
- 随着UNIX的流行，C逐渐成为开发系统程序和复杂软件系统的通用语言。C几乎无所不能。
- C的标准化：K&R C－C89（ANSI C）－C99－C11（2011.12.8日ISO发布）
- C的特点

观其大略（1）

```
int  main( )
{
      return 0;
}
```

观其大略（2）

```
#include    <stdio.h>
int main( )
{
        printf(“This is my first C program!”);
        return 0;
}
```

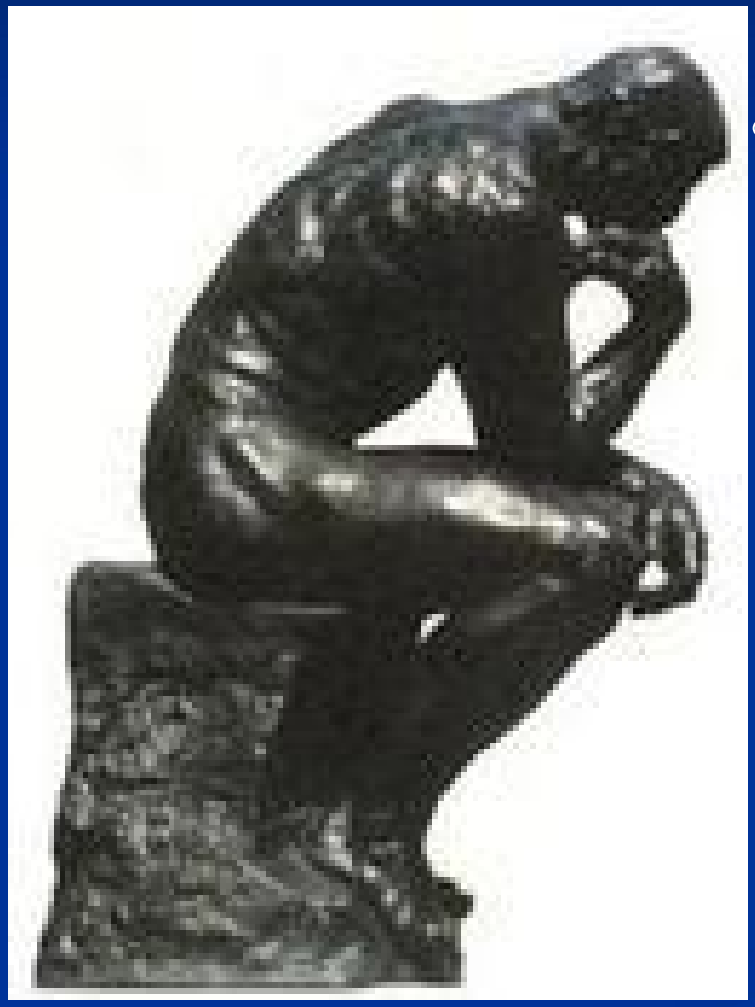
观其大略（3）

```
#include    <stdio.h>
int main( )
{
        int  value1, value2, sum;
        scanf(“%d%d”, &value1, &value2);
        sum = value1 + value2;
        printf(“%d+%d=%d\n”, value1,value2,sum);
        return 0;
}
```

观其大略（4）

```
#include   <stdio.h>
float max ( float  x,  float  y )
{      float  z;
       if (x > y)
            z = x;
       else
            z = y;
       return  z;
}
int  main( )
{       float  a, b, c;
        scanf("%f%f",  &a,  &b);
        c = max(a, b);
        printf("max=%f\n",  c);
        return 0;
}
```

思考



从前几个例子中，我能看出什么端倪？

略知一二

- 每个程序都是由一个或多个如下形式的内容组成

  符号甲　符号乙（可有/可无）

  {

  ……

  }

- 且每个程序中一定会有一个叫**main**的名字在符号乙位置出现。

一个概念：函数

- 函数是C程序的基本单位
  - 一个C源程序一般由一个main函数和零或多个其他函数构成，注意一个C程序必须包含且只能包含一个main函数。
  - C程序总是从main函数开始执行，而不论main身在何处。main函数可以调用其他函数，其他函数之间也可以互相调用，但其他函数不能调用main函数。
  - C函数分为系统提供的标准库函数和用户自定义的函数两种形式。

函数基本结构

```
float  max(float  x, float  y )
{
      float  z;

     if (x > y)
          z = x;
     else
          z = y;
     return  z;
}
```

① 函数首部

返回值类型　函数名(参数列表)

② 函数体

{

　声明部分：声明所用变量。

　执行部分：由若干语句组成。

}

进一步对比观察

- 除第一个程序外，其他程序的开始处都有如下一行：

  #include <stdio.h>

- 除第一个程序外，其他程序的{ }中都有类似如下的一行：

  printf(…….);

第二个概念：预处理指令

- 预处理指令是指C编译系统在对源程序进行“翻译”以前，先由一个“预处理器”对预处理指令进行预处理。
- 因为在程序的执行过程中，main函数会直接或间接调用其他函数。printf( )是系统提供的用于输出的函数，它在stdio.h文件中定义，因为程序中用到了它，所以就需要用

  #include <stdio.h>

  这个预处理指令把stdio.h文件的内容包含进来，便于程序正确执行。
- 预处理指令的特点：以 # 开头。
- 有点类似：“引用请注明出处，违者必究”。

第三个概念：语句

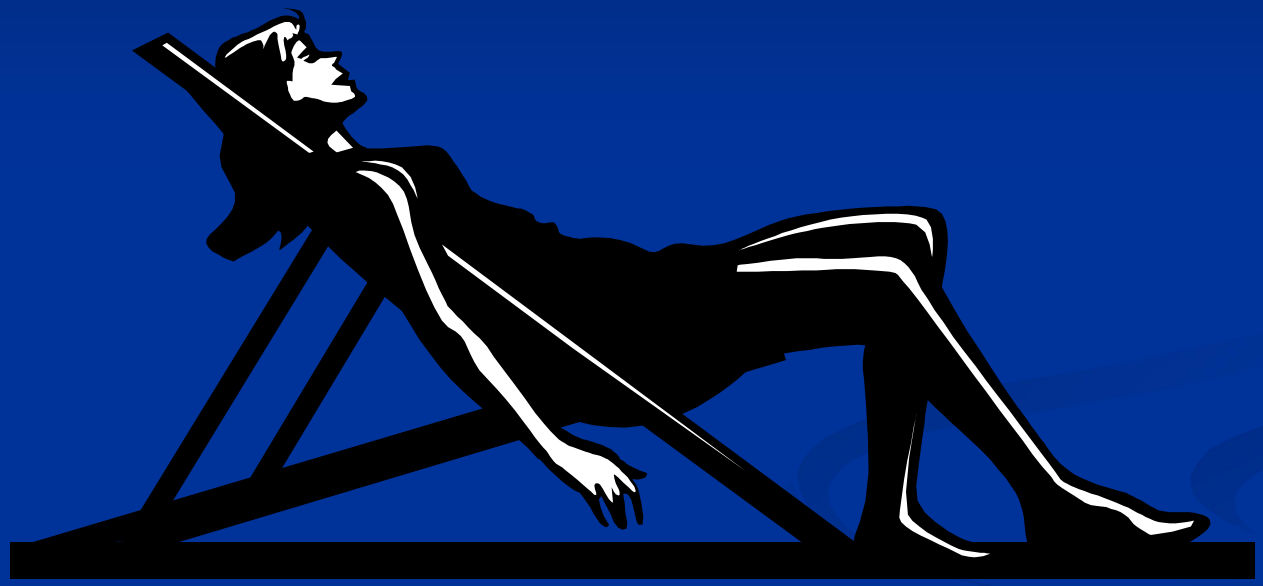
- 再进一步观察，你会发现每个程序{ }内的代码几乎每行结尾都有一个分号；
- 程序中对计算机的操作是由函数中的C语句完成的，语句的最后必须有一个分号，分号是C语句的必要组成部分，如：

```
sum = value1 + value2;
```

C程序的一般形式

预处理指令
返回类型　函数名（参数列表）
{
　　声明部分
　　可执行语句
}

休息一下脑子，想想刚才讲了什么

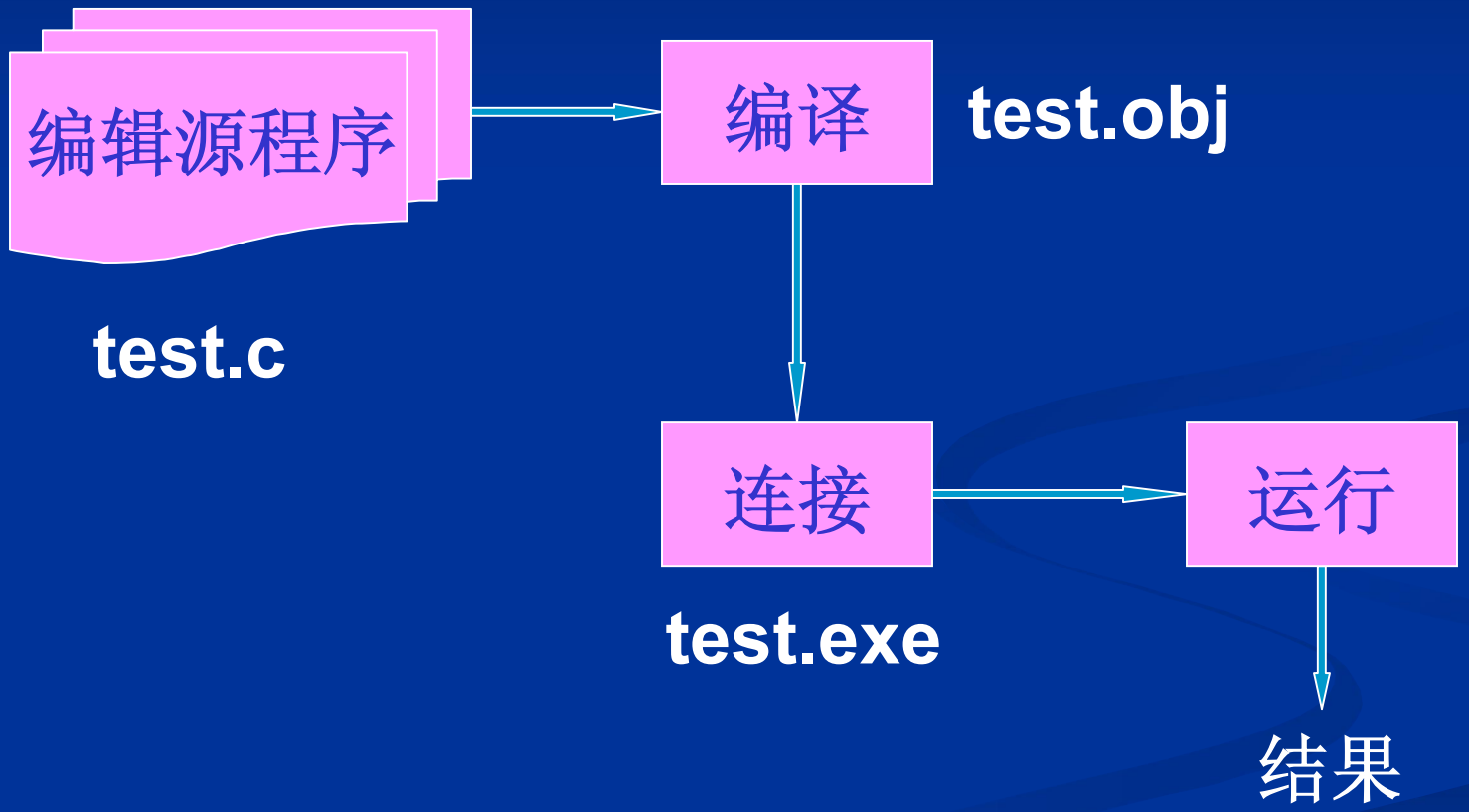


接下来你想学什么？

C程序的运行

- 源程序
  - 用C语言编写的程序称为源程序。
  - 其扩展名为“.C”。
  - 源程序不能直接在计算机上执行，需要经过“编译”将其翻译为二进制形式的代码。
- 目标程序
  - 源程序经过“编译”所得到的二进制代码为目标程序。
  - 其扩展名为“.OBJ”。
  - 目标代码尽管已经是机器指令，但是还不能运行，因为它还没有解决函数调用问题。
- 可执行程序
  - 目标程序与库函数进行连接，形成完整的可在操作系统下独立执行的程序称为可执行程序。
  - 其扩展名为“.EXE”

C程序运行过程



C语言开发环境

- 为了把写好的C源程序能运行起来，需要进行编译、连接，而这必须要有相应的编译系统。
- 目前使用的大多数C编译系统都是集成环境（IDE）的，即把程序的编辑、编译、连接和运行全部集中在一个界面上进行。
- 常用的开发环境
  - Turbo C 2.0
  - Microsoft VC++ 6.0

让我们共同了解开发环境

演示几个程序掌握如下知识

- C语言字符集
- 标识符
- 关键字
- 分隔符
- 注释符
- 运算符
- 转义字符
- 编程风格
- 程序调试

```
/*例1：一个简单的C程序*/
#include <stdio.h>

int main( )
{
    int  age;                  //声明一个变量

    age = 20;                  //给age赋一个值
    printf("This is my first C program!\n");//使用printf()函数
    printf("I am %d years old.", age);
    printf("I am learning C programming.\n");
    return 0;
}
```

- 例1说明：
  - #include和头文件： stdio.h
  - main()函数
  - 注释: /*...*/　　//
  - 花括号， 函数体: {...}
  - 声明: int age; 在{开始， C99允许任意位置
  - 赋值： age = 20;
  - printf()函数： \n
  - return语句

```
/*例2：米和尺之间的转换*/
#include <stdio.h>

int main( )
{
    int  meter, ruler;

    meter = 5;
    ruler = 3 * meter;
    printf("%d米等于%d尺.\n", meter, ruler);
    return 0;
}
```

- 例2的说明：
  - 多个声明。程序中声明了两个变量，需要用逗号把两个变量分开；如

    int meter, ruler;

    和如下等价：

    int meter;

    int ruler;
  - 用了乘法。*表示乘法
  - 输出多个值。以新的方式使用了printf( )。

```
/*例3：在一个文件中使用两个函数*/
#include <stdio.h>

void ans_age( );       //函数原型

int main( )
{
    printf("Hello, How old are you?\n");
    ans_age( );
    printf("I am very glade to see you!\n");
    return 0;
}

void ans_age( )                //函数定义的开始
{
    printf("I am 20 years old.\n");
}
```

- 例3说明：
  - ans_age( )函数在程序中出现了3次。第一次出现在原型中，通知编译器要用到该函数。原型是一种声明的形式，用于告诉编译器你正在用一个特殊的函数，它指明了该函数的属性。
  - 第二次在main( )函数中以函数调用的形式出现，通过给出ans_age( )函数的名字来实现调用。
  - 最后一次是程序给出的函数的定义，即函数本身的源代码，定义方式和main( )相同。

C语言字符集

- 字符是C语言最基本的元素，C语言字符集由字母、数字、空白符、标点符号和特殊字符组成。
- C程序是用下列字符所组成的字符集写成的：
  - 字母: A-Z，a-z
  - 数字: 0-9
  - 标点符号、特殊字符(28个):

    ! # % ^ & + - * / = ~ > < \ | . , ; : ? ‘ “ ( ) [ ] { }
  - 空白符: 空格, 制表符(Tab跳格健), 换行符的总称。空白符除了在字符、字符串中有意义外，在程序中只起间隔作用，编译系统会忽略其它位置的空白符。

标识符

- 标识符是用户给程序中的实体所起的名字
  - 这些实体指：变量, 常量, 函数, 数组, 类型等
- 标识符的命名规则
  - 由字母，数字和下划线组成，必须以字母或下划线开头
  - 标识符不能与C语言的关键字重名（如：int, float, if 等）
  - 建议用户定义标识符时尽量不用下划线开头
  - 标识符中区分大小写字母 （如: sum 和 SUM 是不同的标识符）
- 定义标识符应遵循的原则
  - 尽量做到见名知义
  - 一般习惯上变量名、函数名用小写 , 而符号常量用大写
  - 应尽量避免使用容易认错的字符　如: 数字1和小写字母l

关键字

- 关键字是C语言预先定义的、具有特定意义的标识符，也称为保留字。C89包括32个关键字:

auto	break	case	char	const	continue
default	do	double	else	enum	extern
float	for	goto	if	int	long
register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void
volatile	while				

- C99又增加了： restrict, inline, _Complex, _Imaginary, _Bool

分隔符

- 在C语言程序中，空格、逗号、回车/换行等，在不同的应用场合起着分隔符的作用。
  - 例如: int value1, value2, sum ; 中的空格和逗号都起着分隔符的作用，如果int和value1之间没有空格，程序就会出错；而在value1和value2之间少了逗号，value1value2就会被认为是一个变量。

注释符

- 两种注释形式
  - 以/*开始，以*/结尾。/*和*/之间可包含任何字符，也称多行注释，注意：不能嵌套在另一个这种注释中；当出现在“”中时，不起注释作用。
  - 以//开始的单行注释（C99新增，但在C89的很多编译器上已经被支持了）。
- 编译器对注释的处理：转换成空格，因此注释可以出现在任何空格可以出现的位置。
- 作用：
  - 起到提示和解释的作用，是个好习惯，要学会写注释。
  - 用于程序的调试，暂时屏蔽一些语句。

运算符

- 运算符是用于描述某种运算功能的符号，如=、+、-、 *、 / 等。
- 运算符可以由一个或多个字符组成。
- 根据参与运算的操作数个数的不同，可分为：单目运算符、双目运算符和三目运算符。

转义字符

- 以“\”开头，后跟一些指定字符，以表示特定的、具有控制功能的字符。例：
  - ‘\n’：表示回车换行
  - ‘\t’：表示制表符
  - ‘\”’，‘\’’，‘\\’，……
  - ‘\ddd’：八进制表示字符
  - ‘\xhh’：十六进制表示字符

编程风格

- 使程序具有可读性是一个良好的编程习惯。
- 下面是一些好的习惯：
  - 选择有意义的变量名
  - 使用注释
  - 合理的使用空行
  - 每个语句占用一行
  - 采用缩进（用TAB键，而不是空格）
- 总之，你可以选择任意合理的风格，并一直坚持即可。

调试

- 编写程序难免出错，程序的错误通常叫做bugs，而发现和修正这些错误的过程叫调试。错误一般分为：
  - 语法错误：不遵循C语言的规则就会犯语法错误
  - 语义错误：语法没错，但在意思上出错。简单说就是程序可以运行，但结果不正确。

到目前初学常见语法错误

- 语句结尾丢失分号
- 预处理命令结尾加分号
- 函数定义时函数首部右括号后加分号
- (), {}, “”等没有成对出现
- 函数名，关键字等拼写不正确
- printf()函数输出变量时，变量类型和格式说明符不匹配

语义错误调试方法

- 程序状态：是指在程序执行过程中的给定点上所有变量值的集合。它是当前计算状态的一个快照。
- 调错方法一般有三种：
  - 自己逐步执行程序，以监视程序状态。
  - 在程序的几个关键点处加入额外的printf()语句以监视所选变量的值。通过观察变量值的变化了解程序执行情况。
  - 使用调试器。一般IDE都带调试器。

数据类型

- 在程序的指示下，计算机可以做很多事情。
- 但为了完成这些事，程序需要使用各种数据，即承载信息的数字与字符。
- 程序对数据进行操作，在设计中用类型来约束数据的解释。
- 简单说，数据类型就是一组数值和对这组数值的一组操作。例：整型数值包括一些指定范围的整数和对这些数值的一组操作，包括加、减、大小比较等等 。

示例程序

- 1海里等于1.852公里。编写一个程序，要求输入海里数（可以有小数），然后显示对应的公里数。

常量和变量

- 程序中处理的数据又分常量和变量。
- 常量
  - 指数据在程序运行之前预先设定并在整个运行过程中没有变化的值。
- 变量
  - 用于存储程序的输入数据和计算结果的存储单元称为变量，即程序运行过程中存储在变量中的值可以改变。
- 例： ruler = 3 * meter; 在这个语句中3就是常量，ruler就是变量。

变量声明

- 变量声明：是用于向编译器通知程序中用到的变量名和每个变量存储的信息类型的语句。
- 格式：类型标识符　变量列表;
  - 例：int meter, ruler;
- 说明：
  - 进行变量声明后，计算机系统会为声明的变量分配存储空间，用以存放数据。
  - 变量的存储空间可能由一个或多个字节组成，内存中的每个字节都有自己的地址，变量名实际上是一个符号地址。在程序中对变量的赋值和取值操作实际上是通过变量名找到相应的内存地址，然后从对应的存储空间中读取数据。

C的基本数据类型

C的数据类型关键字

原来的K&R关键字	C89关键字	C99关键字
int	signed	_Bool
long	void	_Complex
short		_Imaginary
unsigned		
char		
float		
double		

常用基本数据类型

- 上表中的数据类型可按其在计算机中的存储方式分为如下两个系列：
  - 整数类型
  - 浮点数类型
- 区别：
  - 对人而言，整数和浮点数的区别在于它们的书写
  - 对计算机而言，整数和浮点数的区别在于它们的存储方式

整数和浮点数

- 整数就是没有小数部分的数，在C中永远不会有小数点出现在整数的书写中。它以二进制数字存储。如：2，365
- 浮点数类似数学上的实数，一般书写时都有小数点，但也其他写法。浮点数表示法将一个数分为小数部分和指数部分分别存储。如3.14，0.75
- 需要注意的是：数据在计算机中是存放在存储单元中的，且存储单元是由有限的字节构成的，每一个存储单元中存放数据的范围是有限的。所以在计算机中的数据是有取值范围的。

整数类型

- 整型家族包括：
  - short int（短整型）
  - int（整型）
  - long int（长整型）
  - long long int（C99引入）
  - 上述每种类型又分signed（有符号）和unsigned（无符号），缺省为signed。
- 标准规定整型值相互之间大小的规则：
  - 长整型至少应该和整型一样长，而整型至少应该和短整型一样长。
- 注意：
  - K&R C标准并没有规定长整型必须比短整型长，只是规定它不得比短整型短。ANSI C（C89）标准加入了一个规范，说明了各种整型值的最小允许范围，即：
  - short int至少16位，long int至少32位，int通常取决于机器字长，至少16位，但一般是32位。long long int是为了支持64位机器需求而引进，至少64位。

int类型

- int类型是有符号整数，可以是正的，负的或者是0，其取值范围依赖于计算机系统。
- ISO/ANSI C（C89）规定int类型的最小范围是-32768到32767，现代系统一般int类型占4个字节。

声明int变量

- 格式：先写“int”，后加变量名，再加一个分号。要声明多个变量，可以逐个声明每个变量；也可以在int后跟上一个变量名列表，各个变量之间用逗号分隔。
- **int age;**
- **int meter, ruler;**
- 声明变量会使编译器为变量赋予名称并安排存储空间。

初始化变量

- 初始化变量就是为变量赋一个初始值。C语言中，可以在声明语句中初始化变量，即在变量名后跟上赋值运算符（=）和要赋给变量的值，如：
- int age = 20;
- int meter = 5, ruler =15;
- int dogs, cats = 10; //该语句有效，但形式不好

int类型常量

- C把不含小数点和指数的数当作整数。
- 十进制整数常量：23，-32，0
- 八进制整数常量：以0开头，017, 023
- 十六进制整数常量：以0x(或0X)开头，0x17，0X2e
- C把大多数整数常量看作是int类型。如果整数特别大，则有不同的处理。

打印int值

- 可以使用printf( )函数打印int类型的值。
- %d（称为格式说明符）符号用于指示在一行中的什么位置打印整数。
- 程序员必须确保格式说明符的数目同待打印值的数目相同，编译器不会发现这种类型的错误。
- %o用于显示八进制整数。
- %x用于显示十六进制整数。

其他整数类型

- short int a;
- long int b;
- long long int c;
- unsigned int d;
- unsigned short int e;
- unsigned long int f;
- unsigned long long int g;

多种整数常量之间的转换

- 通常，在代码中使用356这样的数字时，它以int类型存储。当这样的数字很大时，编译器会依次按如下规则转换：

  int－long int－unsigned long int－long long int－unsigned long long int
- 有时想让一个较小的整数常量用long类型或unsigned int来存储，这时需要在该常量后面加L或者加U。
  - 356， 356U， 356L， 356UL

char类型

- 用于存储字母和标点符号之类的字符。
- 在技术实现上char是整数类型，因为char类型实际存储的是整数而不是字符。
- 为了处理字符，计算机使用一种数字编码，用特定的整数表示特定的字符，常用的编码是ASCII码。在ASCII中，整数值65代表大写字母A；因此要存储字母A，实际只需存储数65。
- char类型占用一个字节。 C把一个字节（byte）定义为char类型使用的位（bit）数。 char类型通常定义为使用8位内存单元，但有些系统上一个字节不是8位。

声明char类型变量

- char ch;
- char grade, rank;s

字符常量及其初始化

- 用一对单引号括起来的单个字符
  - 例：‘a’, ‘8’, ‘A’, ‘*’
- 转义字符：以“\”开头，后跟一些指定字符，以表示特定的、具有控制功能的字符
  - 例：‘\n’, ‘\”’, ‘\”’, ‘\\’, ‘\t’, ‘\ddd’, ‘\xhh’
- char grade = ‘A’; //单引号必不可少
- char grade = 65; /*对于ASCII，这是可以的，但这是一种不好的编程风格*/
- 注意：C将字符常量视为int类型而非char类型。

打印字符

- printf()函数使用%c说明符打印一个字符。
- 字符变量被存储为1字节长的整数值，因而，如果使用通常的%d说明符打印char变量，将得到一个整数。
- %c格式说明符告诉printf()函数打印编码值等于那个整数的字符。

字符型数据与整型数据的相互运算

```
#include  <stdio.h>
int main(void)
{
    char  ch;

    ch='a';
    printf("%c\n", ch);
    printf("%d\n", ch);
    return 0;
}
```

运行结果：a
97

```
#include  <stdio.h>
int main(void)
{
    char  ch;
    int  x;
    ch='A';
    x=ch+32;
    printf("%c,%d\n", ch, ch);
    printf("%c,%d\n", x, x);
    return 0;
}
```

运行结果：A, 65
a, 97

浮点数类型

- float（单精度浮点数）
  - C标准规定，float类型必须至少能表示6位有效数字。
  - 取值范围至少为10^{-37}到10^{37}。
  - 通常占用4个字节。
- double（双精度浮点数）
  - double必须至少能表示10位有效数字。
  - 和float类型具有相同的最小取值范围。
  - 通常占8个字节。
- long double
  - 满足比double类型更高的精度需求。不过，C只保证long double类型至少同double类型一样精确。

声明浮点变量及初始化

- float x, y;
- double result;
- long double test;
- double score = 92.5;

浮点常量

- 十进制形式：由数字和小数点组成（必有小数点）
  - 例：2.8，-3.6，3.，.75　（0可省）
- 指数形式：1.289e+3，　31.4E-1
  - 说明：e前面的数可以没有小数点，e后面的正号可以省略，e后面必须是整数；e和前面的数之间不能有空格。
- 默认情况下，编译器将浮点数常量当作double类型。如果想让该常量作为float类型，需在常量后面加F或f，如2.3F。后缀L使浮点常量成为long double类型，如4.2L。

打印浮点值

- printf()函数使用%f格式说明符打印十进制记数法的float和double数字。
- 用%e打印指数记数法的数字。
- 用%Lf、%Le打印long double类型数字。
- 注意：float和double类型的输出都使用%f、%e说明符。这是由于在向printf()传递参数时，C自动将float类型的参数转换为double类型。

字符串

- 由一对双引号(“ ”)括起来的字符序列。
  - 例："Hello"、"I love China!"、"How do you do?“
- 为了判断字符串结束，C编译器会自动在字符串的末尾加一个转义字符‘\0’，作为字符串常量的结束标志。‘\0’的ASCII值为0，是不可显字符，也不作任何操作，即“空操作字符”。因此字符串的实际长度为字符个数加1。

- 注意:
  - 'a'与"a"是不同的
  - 'a'是字符常量,在内存中占1个字节
  - "a"是字符串常量,在内存中占2个字节

数据的输入与输出

- C语言的输入/输出均由函数来实现, 在使用输入输出函数时, 应在程序开头写预编译命令： #include <stdio.h>
- 本节介绍最常用的输入输出函数
  - 格式输入输出函数： printf和 scanf
  - 字符输入输出函数： putchar和getchar

printf( )函数

- 一般形式： printf(“格式控制”, 输出表列);
  - 例：printf(“hello world\n”);

    printf(“sum is : %d\n”, sum);

    printf(“a=%d,b=%d\n”, a, b);
- 格式控制
  - 普通字符原样输出
    - 普通字母、数字及符号
    - 转义字符：\n, \t等
  - 格式说明符：由%和格式字符及附加格式说明字符组成
    - %d：输出十进制整数
    - %f： 输出十进制浮点数
    - %c：输出单个字符
    - %s: 输出字符串

- 附加格式说明符
  - 放置位置：在%和格式字符之间
  - 如下字符：
    - 字母l：输出long型数据，如：%ld等
    - m.n：m (正整数)表示数据的最小宽度；n(正整数)：对于实数，表示输出n位小数；对于字符串，表示截取的字符个数
    - 一(负号)：输出的数据或字符在域内向左对齐（缺省右对齐）
- 输出表列：输出项可以是常量, 变量, 表达式
  - 例：int x=8;　　输出结果：

    printf("x=%8.4f\n", 6.85);　x=□□6.8500

    printf("x=%d\n", x);　　x=8

scanf( )函数

- 一般形式：scanf(“格式控制”, 地址表列);
- 格式控制
  - 格式说明符：基本同printf()相似
    - %d：输入十进制int数据
    - %c：输入char数据
    - %f：输入float数据
    - %lf：输入double数据
    - %ld：输入long int数据
    - %s：输入字符串数据
  - 普通字符原样输入
  - 地址表列
    - 由变量地址组成：即在变量名前加地址运算符“&”
- 例：scanf(“%d%d”, &a, &b);

  要求输入：5 8

  scanf(“%d,%d”,&a, &b);

  要求输入：5,8

  scanf(“a=%d,b=%d”, &a, &b);

  要求输入：a=5,b=8

- 注意问题
  - 用%c输入字符时, 空格和转义字符都会作为有效字符输入

    例： scanf(“%c%c”, &ch1, &ch2) ; 如输入 : A　B

    则　ch 1为字符A ， ch2为空格
  - 输入数据时,遇到以下情况时输入结束
    - 遇到空格, 或“回车”键, “跳格”键时结束
    - 按指定输入宽度结束

    例： scanf(“%d%3d%2d”, &x , &y , &z) ; 如输入: 25　1867490

    x 为 25 ， y 为 186 ， z 为 74
    - 遇到非法输入时结束

    例： scanf(“%d%d”, &x , &y) ; 如输入: 25 , 3

    x 为25, 　y 无正确数据(因遇逗号而出错)
  - 如果%后有附加格式说明符“*”, 表示跳过它指定的列数

    例： scanf(“%3d%*3c%2d”, &a, &b); 如输入： 123456789

    则　a为123, b为78 (456被跳过)
  - 不要在格式控制中输入转义字符及指定精度

    例： scanf(“%5.2f\n”, &a); 这样是错误的

putchar( )函数

- 格式 : putchar(参数)　注意：参数只有一个
- 功能: 将参数对应的字符输出到显示器上，参数可以是字符型或整型的常量和变量。

```
#include <stdio.h>
int main(void)
 {
    char  a='B', b='O';
    putchar(a);
    putchar(b);
    putchar('Y');
    return 0;
 }
```

输出结果：BOY

等价于:
printf("%c", a);

getchar( )函数

- 格式：getchar( )　　注意：它是无参函数
- 功能：从键盘输入一个字符

```
#include <stdio.h>
int main(void)
{
    char  ch;
    ch=getchar( );
    putchar(ch);
    putchar('\n');
    return 0;
}
```

输入：A
输出：
A

等价于：
scanf("%c", &ch);

基本运算符——赋值运算符

- =称为赋值运算符（注意：不是等号）
- 例：age = 20; //将20赋给名字为age的变量
- 即：=左边是个变量名，=右边是赋给该变量的值。（注意区分变量名和变量值）
- 例：i = i + 1; 可以吗？
- 例：20 = age; 可以吗？
- 注意：赋值运算符是从右到左进行的。
  - 例：a=b=3+8;
  - 先做3+8，等于11，然后b得到值11，然后a再得到值11。

算术运算符

- + - * / %（取模）
- 说明
  - 都为双目运算符，即它们需要两个操作数。
  - 对于除法运算来说，浮点数相除得到一个浮点数结果，但整数相除则产生一个整数结果，小数被无条件截去，C99采用“向零取整”的原则。
    - 例：5/3等于1；1/3等于0
  - %运算符要求参与运算的操作数必须为整数，它的结果为两个整数相除的余数。
    - 例：5%3等于2； 6%3等于0

运算符的两个属性

- 优先级
  - 简单说，优先级是用来决定两个相邻的运算符哪个先被执行。
- 结合性
  - 简单说，结合性就是一串运算符是从左向右依次执行还是从右向左逐个执行。

运算符的优先级

- 例： test = 2.5 + 6.2 * n / m;　　执行顺序？
- C关于执行顺序的规则：两个相邻的运算符哪个先执行取决于它们的优先级，如果两者的优先级相同，它们的执行顺序由它们的结合性决定。除此之外，编译器可以自由决定使用任何顺序对表达式求值，只要不违背某些运算符（, && || ?:）所施加的限制就行。
- 算术运算符的优先级高于赋值运算符
- *、/、 %的优先级高于+、-
- 算术运算符是左结合，赋值运算符是右结合

优先级和求值顺序

- 表达式的求值顺序依据运算符的优先级规则，但并非完全由运算符的优先级决定。
- 例：test = a + b * c; 规则起作用
- 例：test = a * b + c * d + e * f; 怎么执行？
- 当两个运算符共享一个操作数时，优先级规定了求值的顺序，即：优先级只对相邻运算符的执行顺序起作用。
- 对于上例，优先级规则只能保证每个乘法运算在它相邻的加法运算之前执行即可，但并没有规则要求所有的乘法运算首先进行，也没有规则规定这几个乘法之间谁先执行。

复合的赋值运算符

- 在赋值运算符“=”之前加上+、-、*、/、%、<<、>>、&、^、|等10种双目运算符，可构成复合的赋值运算符。即构成+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=10种复合运算符。
- 例：

a += 5; ⟺ a = a+5;

y *= x-2; ⟺ y = y*(x-2);

x %= y/3; ⟺ x = x%(y/3);

自增与自减运算符

- ++、--　　作用：使操作数递增1或递减1
- 形式：
  - 前缀：++a;　--a;
  - 后缀：a++;　a--;
  - 自增、自减运算符都是单目运算符
- 例：int x, a; a= 5; x = a++; x = ++a;　x是多少？
- 注意：
  - 自增、自减运算符只能作用于变量，而不能是常量或没有确定地址的表达式，如：8++、(x*y)++　错
  - 在不需要使用任何具体值且仅需要递增变量的情况下，前缀方式和后缀方式的效果相同。

sizeof运算符

- sizeof运算符以字节为单位返回其操作数的大小。
- 操作数可以是一个具体的数据对象（如变量名），或者一个类型。如果它是一个类型（如float），操作数必须被括在圆括号里。
- 例： int a; float b;

  printf(“%d,%d\n”, sizeof a, sizeof (int));

逗号运算符

- ,
- 一般形式： 表达式1, 表达式2, ……表达式n ;
- 执行过程：从左至右依次求解表达式1，表达式2……表达式n 。逗号表达式的值就是表达式n 的值。
- 优先级：最低
- 例：int i=2, j=3, k=4;

  i+1 , j+4 , k-2; 最后表达式语句的值为2

  a=(a=6, a*3, a+3); 最后a的值为？

表达式

- 表达式：是由运算符和操作数组合构成的。
- 最简单的表达式是一个单独的操作数。
- 例：x　　4　　4+6　　a*(b+3)　　x=(3+2)%4
- 通过例子可以看出，操作数可以是常量、变量或者二者的组合。一些表达式是多个较小的表达式的组合，这些小的表达式被称为子表达式。
- 重要：每一个表达式都有一个值。
- 赋值运算符构成的表达式的值是：与=左边的变量取相同的值。例：a=3+2

语句

- 语句：是构造程序的基本成分。一个语句是一条完整的计算机指令。在C中，语句用结束处的一个分号标识。
- 例：a=4只是一个表达式，而a=4;是个语句
- 常见C语句：
  - 表达式语句：C把任何后面加有一个分号的表达式都看作是一个语句。3+4; (没用)　x=4; (有用)
  - 赋值语句：为变量分配一个值，是表达式语句的特例。
  - 函数调用语句：引起函数的执行。printf(“%d”, x); 等
  - 空语句：就一个分号，什么也不作。　;
  - 控制语句：控制程序流程。　for, if等，后面学习
  - 复合语句：括在{}里面的一句或多句。 {a=b*3; b++}　函数体
  - 声明语句：用于建立变量的名字和类型并导致为它们分配内存空间，它不是一个表达式语句，即去掉分号，得到的不是一个表达式，也不具有一个值。　int　age;

类型转换

- 语句和表达式通常应该只使用一种类型的变量和常量。但如果混合使用多种类型，就会涉及到类型转换的问题。
- 例：x=3+’a’+2.3*4.5F;
- 类型转换基本规则：
  - 类型级别从高到低的顺序：long double、double、float、unsigned long long、long long、unsigned long、long、unsigned int、int
  - 在表达式里，char和short都将自动被转换为int。
  - 在包含两种数据类型的任何运算里，两个值都被转换成两种类型里较高的级别。
  - 赋值语句中，计算的最后结果被转换成将要被赋予值的那个变量的类型。这个过程可能导致提升或降级。
  - 当作为函数的参数被传递时，char和short会被转换为int，float会被转换为double。

强制类型转换

- 上述提到的类型转换是自动完成的，程序员也可以使用强制类型转换运算符将一个表达式的值转换成所需的数据类型。
- 形式：　(类型名)(表达式)
- 例：int a, z;　double x, y;

  x = (double)a;　z = (int)(x+y);
- 说明：
  - 为避免出错，应将表达式用括号括起来。
  - 对一个变量进行强制转换后，得到一个所需类型的中间变量，原来变量的类型和值不发生变化。

编写简单C程序

- 三种不同的程序设计结构
  - 顺序结构
  - 选择结构（分支结构）
  - 循环结构
- 其中，顺序结构是最基本、最简单的程序结构。在顺序结构程序中，各语句是按照位置的先后次序，顺序执行的，且每个语句都会被执行到。

顺序结构程序示例

- 例：已知a=5，b=10，试交换a、b的值。
- 分析：交换两个变量的值，就像一个瓶装酱油，一个瓶装醋，要交换内容需要借助第三个空瓶一样。交换两个数也需要借助第三个变量。

```
#include <stdio.h>
int  main(void)
{
    int  a=5, b=10;
    int  temp;
    temp = a;
    a = b;
    b = temp;
    printf("a=%d, b=%d \n", a, b);
    return 0;
}
```

- 例：求任意三个整数的平均值。
- 分析：
  - 首先要有3个整数， 这3个数可以用a、b、c 三个整型变量存放
  - 其次平均值应该用一个变量保存，该变量应该是浮点数类型
  - 求任意3个整数的平均值，所以这3个数应该从键盘输入
  - 计算平均值：(a+b+c)/3.0， 因为C语言中两个整数相除得到的结果也是整数(去掉小数部分)，为保证结果正确必须用实数3.0

```
#include <stdio.h>
int main(void)
{
    int  a, b, c;
    float   ave;
    scanf("%d%d%d", &a, &b, &c);
    ave = (a+b+c) / 3.0;
    printf("ave=%f\n", ave);
    return 0;
}
```

算法

- 进行程序设计，要解决两个问题：
  - 要学习和掌握解决问题的思路和方法，即算法
  - 学习怎么实现算法，即用语言编写程序
- 算法的概念
  - 为解决一个问题而采取的方法和步骤
  - 例：购物，回家，上课，吃饭……
- 算法的分类
  - 数值算法：解决求数值解的问题
  - 非数值算法：解决用分析推理、逻辑推理求解的问题

算法的特性

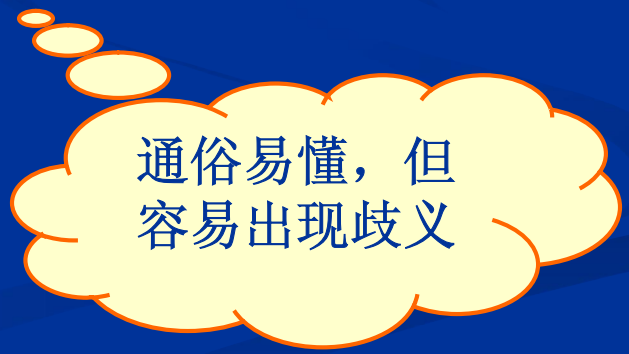
- 有穷性
  - 空间有穷：一个算法包含有限的操作步骤；
  - 时间有穷：算法可以在合理的时间内运行完。
- 确定性
  - 算法中的每一个步骤是确定的，含义是唯一的
- 有效性
  - 算法中每一个步骤是有效可行的。
- 有零个或多个输入
- 有一个或多个输出

算法的表示方法

- 自然语言表示

  例：用自然语言表示求两个数的和(如2+3)

  - Step1: 将输入的第一个数2存入x
  - Step2: 将输入的第二个数3存入y
  - Step3: 将x和y相加的结果存入z
  - Step4: 输出结果z



- 流程图表示
  - 用图框表示各种操作，用箭头表示算法流程


起止框


输入/输出框


判断框

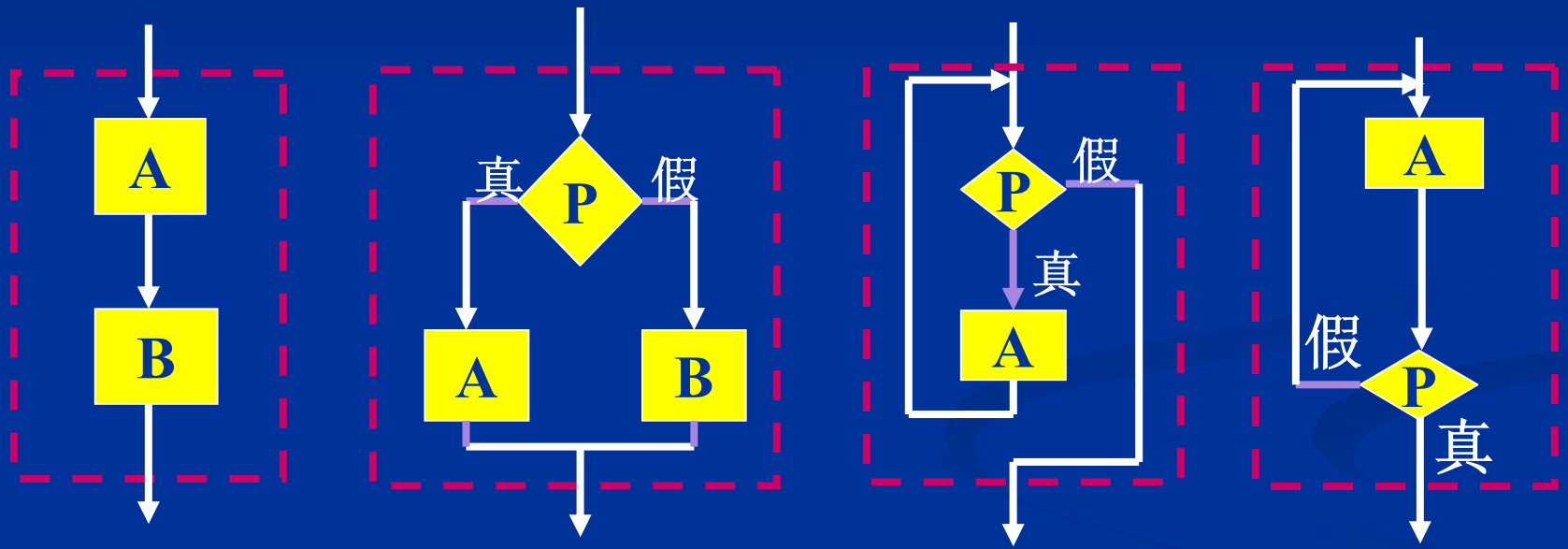

处理框

流程线


连接点

- 用流程图表示求两个数的和

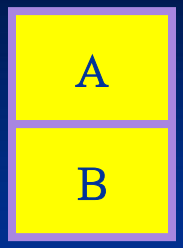
- 为了限制流程线的滥用，提出了三种基本结构，由其按一定规律组成算法结构。



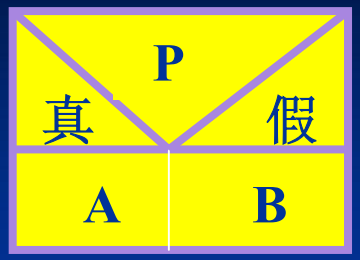
顺序结构　选择结构　当型循环　直到型循环

循环结构

N-S流程图表示



顺序结构



选择结构

当型循环

直到型循环

- 顺序结构：先执行A操作，再执行B操作，两者是顺序执行关系
- 选择结构：当P条件为真时，执行A模块，否则执行B模块
- 循环结构：
  - 当型循环：当P条件成立时，反复执行A，直到P为假。
  - 直到型循环：先执行A操作，再判断P是否为假，若P为假，再执行A，直到P为真为止。
- 优点：比文字描述更加直观、形象，易于理解；比传统的流程图紧凑易画；废除流程线，整个算法结构是由各个基本结构按顺序组成的

- 伪代码表示

例：求5!，用伪代码表示的算法如下：

```
begin
    t=1
    i=1
    while i≤5
    {
       t = t * i
       i = i + 1
    }
    print  t
end
```

C语言的产生、发展及特点

- C语言的产生及发展

  ALGOL60→CPL→BCPL→B→C→标准C→ANSI C→ISO C
- 常见的C语言版本有
  - Borland公司：
  - Turbo C
  - Turbo C++
  - Borland C++
  - C++ Builder
- Microsoft公司：
  - Microsoft C
  - Visual C++

- C语言的特点
  - C语言的语言成分简洁、紧凑、书写形式自由
  - C语言拥有丰富的数据类型
  - C语言的运算符丰富、功能更强大
  - C语言是结构化程序设计语言
  - C语言对语法限制不严格，程序设计灵活
  - C语言编写的程序具有良好的可移植性
  - C语言可以实现汇编语言的大部分功能
  - C语言编译后生成的目标代码小，质量高，程序的执行效率高


Class is over