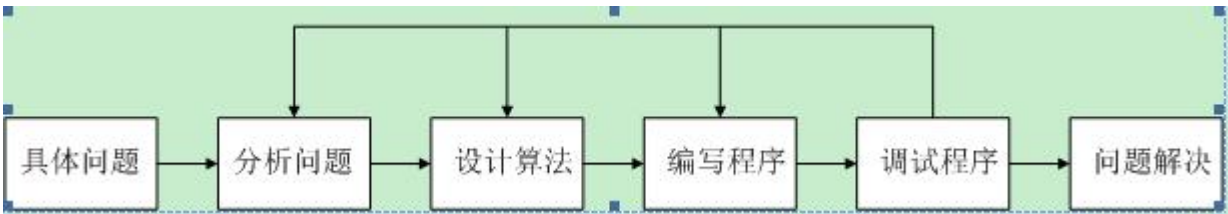


算法与程序基础

第一节 利用计算机解决问题的基本过程

一、计算机解决问题的基本过程



一般来说，用计算机解决一个具体问题时，大致经过以下几个步骤：首先要从具体问题抽象出一个适当的数学模型，然后设计一个解此数学模型的算法，最后编出程序进行测试调整直到最终解答。寻求数学模型的实质就是分析问题，从中提取操作的对象，并找出这些操作对象之间含有的关系，然后用数学的语言加以描述。

二、问题分析与算法设计之间的关系

任何一个问题必须弄清楚其内容、性质、规模，才能找到解决问题的方法，所以分析问题就是要确定用计算机做什么，接下来，就解决怎么做的问题，也就是算法。

算法就是解决问题的方法与步骤。有了算法才能转化成指令代码，计算机才能按照指令代码一步一步去执行，直到得到问题的解。

算法是程序设计的灵魂，算法独立于任何一种程序设计语言，一个算法可以用多种程序设计语言来实现。

一个问题，可能有多种算法，应该通过分析、比较、挑选一种最优的算法。一个好算法必须用到科学的方法，应该好好学习各学科处理问题的科学方法。

三、算法的基本特征

特征	说明
有穷性	一个算法必须保证它的执行步骤是有限的，即它是能终止的。
确定性	算法中的每一个步骤必须有确切的含义，而不应当是模糊的，模棱两可的。
可行性	算法的每一步原则上都能精确运行。
有输入	输入是指算法在执行时 needed 从外界获得数据，其目的是为算法建立某些初始状态。如果建立初始状态所需的数据已经包含在算法中了，那就不再需要输入了。
有输出	算法的目的是用来求解问题的，问题求解的结果应以一定的形式输出。

四、算法分析

（一）基本概念

算法分析：通常用计算机执行时在时间（时间复杂度）和空间（空间复杂度）两方面的消耗多少作为评价该算法优劣的标准，一般时间复杂度考虑的较多。

类型	概念
时间复杂度	方法分为事后统计和事前分析估算。频度：指一条语句重复执行的次数，记作： $F(n)$ 。时间复杂度： $T(n)=O(F(n))$ 它以算法中频度最大的语句来度量的
空间复杂度	空间复杂度是指在算法中所需的辅助空间单元而不包括问题的原始数据占用的空间

（二）时间复杂度举例

<pre>for (i=1; i<=n; i++) for (j=1;j<=n;j++) x++;</pre>	
找到执行次数最多的语句	$x++$
计算语句执行次数的数量级	$T(n) = n^2$ $f(n) = n^2$
用大 O 来表示结果	$O(n^2)$

五、算法的描述

一个算法可以用多种不同的方法来描述。一般用自然语言、流程图、伪代码描述。

（一）自然语言

自然语言是人们日常所用的语言，如汉语、英语、德语等。用自然语言描述算法符合我们的表达习惯，并且容易理解。缺点：书写较烦、不确定性、对复杂的问题难以表达准确、不能被计算机识别和执行。

用自然语言描述一下解决以下问题的算法：借助一个空杯将一杯橙汁和一杯可乐互换所盛放的杯子。

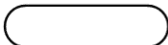


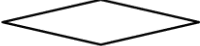


- 1.橙汁倒入空杯；
- 2.可乐倒入刚空出的杯子；
- 3.橙汁倒入刚倒出可乐的杯子。

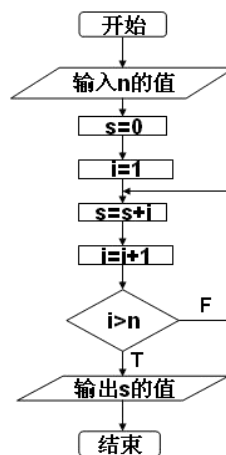
（二）流程图

流程图是由一些图框和流程线组成的，其中图框表示各种操作的类型，图框中的文字和符号表示操作的内容，流程线表示操作的先后次序。也称为程序框图，它是算法的一种图形化表示方法。

优点：形象、直观、容易理解。

由键盘输入一个任意值作为 n ，求 1 到 n 的累加值，算法流程图如下：

程序框	名称	功能
	开始/结束	算法的开始和结束
	输入/输出	输入和输出信息
	处理	计算与赋值
	判断	条件判断
	流程线	算法中的流向
	连接圈	表示算法流向出口或入口连接点

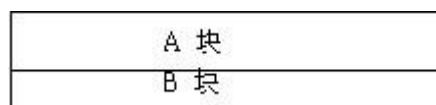


(三) N-S 流程图

N-S 图也被称为盒图或 CHAPIN 图。1973 年，美国学者 I.Nassi 和 B.Shneiderman 提出了一种在流程图中完全去掉流程线，全部算法写在一个矩形阵内，在框内还可以包含其他框的流程图形式。即由一些基本的框组成一个大的框，这种流程图又称为 N-S 结构流程图（以两个人的名字的头一个字母组成）。N-S 图包括顺序、选择和循环三种基本结构。

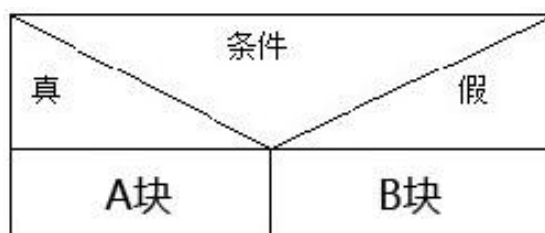
3 种基本程序结构的 N-S 图。

1. 顺序结构 N-S 图



顺序结构

2. 选择结构 N-S 图



选择结构

3. 循环结构 N-S 图



(三) 伪代码

定义	伪代码是介于自然语言和计算机程序语言之间的一种算法描述	举例： IF 九点以前 THEN
优点	简洁、易懂、修改容易	

缺点	不直观、错误不容易排查	do 私人事务; ELSE 9 点到 18 点 THEN 工作; ELSE 下班; END IF
----	-------------	---

六、常见的算法

（一）解析法

1.解析法介绍

定义	解析法（Analysis Algorithm）是指用解析的方法找出表示问题的前提条件与结果之间关系的数学表达式，并通过表达式的计算来实现问题求解。用解析法解决问题的关键就是找到求解问题的解析表达式。
步骤	1.分析问题，列出表示问题的前提条件与结果之间关系的数学表达式 2.编写程序计算表达式，实现问题求解

2.解析法举例

某城市的出租车计费标准为起步价 10 元（3 公里内），此外，在 3 公里到 10 公里之间每公里 2.1 元，超过 10 公里部分每公里 3 元，输入行车距离 x，输出车费 y。

编写数学表达式：设车程数为 X 公里，车费为 Y 元，则

车费：Y	车程：X
$Y = 10$	$x \leq 3$
$Y = 10 + 2.1 * (x - 3)$	$3 < x \leq 10$
$Y = 10 + 2.1 * 7 + 3 * (x - 10)$	$x > 10$

```

工程1 - Form1 (Code)
Form
Private Sub Form_Load()
    Form1.Show
    Dim x, y As Single
    x = Val(InputBox("请输入行车距离"))
    If x <= 3 Then
        y = 10
    ElseIf x > 3 And x <= 10 Then
        y = 10 + 2.1 * (x - 3)
    Else
        y = 10 + 2.1 * 7 + 3 * (x - 10)
    End If
    Print y
End Sub

```

运行结果如下：



（二）穷举算法

1. 穷举法介绍

这种方法一般在计算机中运用，因为计算机计算速度快，可以很快验证答案是否正确。

定义	穷举法也叫枚举法或列举法。在研究对象是由有限个元素构成的集合时，把所有对象一一列举出来，再对其一一进行研究；带入实际，一个个检验是否符合，穷举完所有对象问题将最终得以解决。
思路	罗列所有情况 → 逐个判断 → 选出答案。
步骤	1.分析问题，找出条件与未知数，确定变量； 2.列举出变量所有可能的情况，用循环或循环的嵌套实现； 3.写出符合条件的判断语句，用选择语句实现； 4.输出符合条件的情况； 5.优化程序。

2. 穷举法举例

百钱买百鸡问题。用 100 元钱买 100 只鸡，公鸡每只 5 元，母鸡每只 3 元，小鸡 3 只 1 元，要求每种都买，问能买公鸡、母鸡、小鸡各买几只？

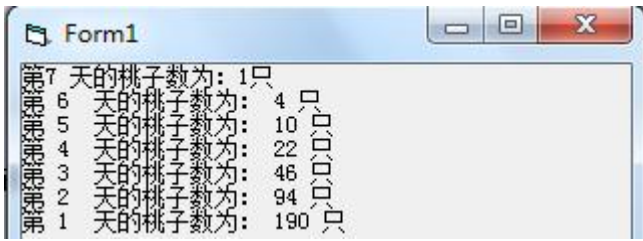
第一步：分析问题，找出条件与未知数，确定变量	
变量	取值范围
公鸡数量：x	1-20
母鸡数量：y	1-33
小鸡数量：z	
第二步：确定判断语句和表达式	
判断：5*x+3*y+z/3 是否等于 100，等于 100，输出 X，Y，Z 的值。	
第三步：列举出变量所有可能的情况，用循环或循环的嵌套实现；	
<pre>Private Sub Form_Load() Form1.Show Dim x, y, z As Integer For x = 1 To 20 Step 1 For y = 1 To 33 Step 1 z = 100 - x - y If 5 * x + 3 * y + z / 3 = 100 Then Print x, y, z End If Next y Next x End Sub</pre>	
运行结果如下：	

（三）递归法

1.递归法定义

定义	“递归法”又称为“迭代法”，其基本思想是把一个复杂的计算过程转化为简单过程的多次重复。每次重复都从旧值的基础上递推出新值，并由新值代替旧值。
步骤	1.分析问题，列出递推公式； 2.确定边界条件； 3.编写、优化程序。

2.递归法举例

问题	猴子吃桃子。小猴在一天摘了若干个桃子，当天吃掉一半多一个；第二天接着吃了剩下的桃子的一半多一个；以后每天都吃尚存桃子的一半零一个，到第7天早上要吃时只剩下一个了，问小猴那天共摘下了多少个桃子？
分析	先从最后一天推出倒数第二天的桃子，再从倒数第二天的桃子推出倒数第三天的桃子……。 设第n天的桃子为 x_n ，那么它前一天的桃子数是 x_{n-1} 。 即： $x_n = \frac{1}{2}x_{n-1} - 1$ 也就是： $x_{n-1} = (x_n + 1) \times 2$ 已知：当N=7第7天的桃子数为1，则第6天的桃子数由公式得4个，依次类推，可求得第1天的桃子数。
代码	<pre>Private Sub Form_Load() Form1.Show Dim I As Integer x = 1 Print "第7 天的桃子数为: 1只" For I = 6 To 1 Step -1 x = (x + 1) * 2 Print "第"; I; " 天的桃子数为: "; x; "只" Next I End Sub</pre> <p>运行结果如下：</p> 

七、常见的查找算法

（一）顺序查找

1.定义

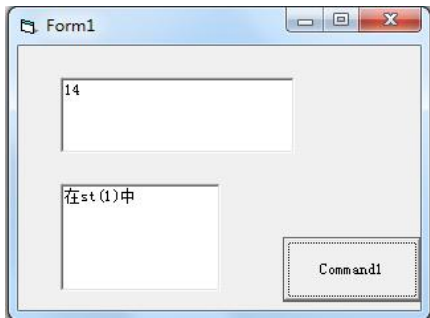
所谓顺序查找就是从数组的第一个数开始，逐个与要查找的数进行比较，如果相同则输出找到的信息。

数组就是相同数据类型的元素按一定顺序排列的集合，就是把有限个类型相同的变量用一个名字命名，然后用编号区分他们的变量的集合，这个名字称为数组名，编号称为下标。

2.举例

```
Private Sub Command1_Click()  
Dim st As Variant  
st = Array(12, 14, 15, 16, 17, 19, 20)  
Dim find As Boolean  
Dim i As Integer  
Key = Val(Text1.Text)  
p = 0  
pos = 0  
find = False  
i = 0  
Do While i <= 6 And find = False  
    p = p + 1  
    If Key = st(i) Then  
        pos = i  
        find = True  
        Exit Do  
    End If  
    i = i + 1  
Loop  
  
If find = True Then  
    Text2.Text = "在st(" + CStr(pos) + ")中"  
Else  
    Text2.Text = "未找到"  
End If  
End Sub
```

运行结果如下：



（二）二分查找

1.二分查找介绍

定义	又称折半查找，它是一种效率较高的查找方法。
要求	必须采用顺序存储结构；必须按关键字大小有序排列。
优点	比较次数少，查找速度快，平均性能好。
缺点	要求待查表为有序表，且插入删除困难。
适用范围	不经常变动而查找频繁的有序列表。
算法思想	首先，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

2.二分查找举例

以下程序是在有序的数组 arr 中进行查找操作，Text1 中输入在查找的数据，，如果找到就在 Text2 中输出该数的下标，否则输出要查找的数不在数组中。

```

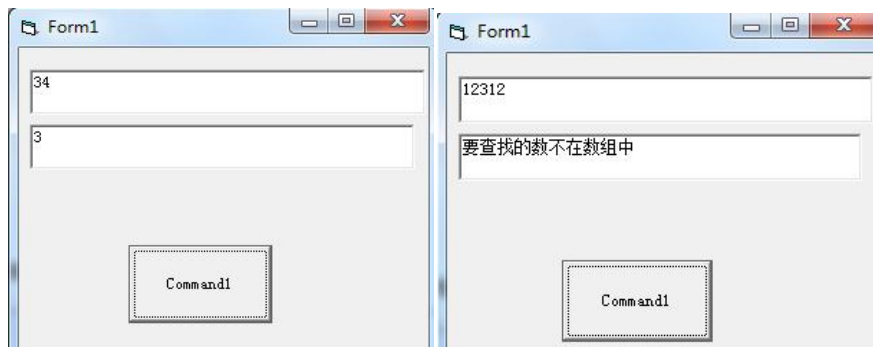
Dim arr

Private Sub Command1_Click()
    low = 0
    high = 9
    x = -1
    a = Val(Text1.Text)
    Do While low <= high
        n = (low + high) \ 2
        If a = arr(n) Then
            x = n
            Exit Do
        ElseIf a > arr(n) Then
            low = n + 1
        Else
            high = n - 1
        End If
    Loop
    If x >= 0 Then Text2.Text = x Else Text2.Text = "要查找的数不在数组中"
End Sub

Private Sub Form_Load()
    arr = Array(1, 22, 33, 34, 45, 56, 57, 58, 79, 88)
End Sub

```

运行结果如下：



（三）哈希表查找

1. 基本原理

我们使用一个下标范围比较大的数组来存储元素。可以设计一个函数（哈希函数，也叫做散列函数），使得每个元素的关键字都与一个函数值（即数组下标）相对应，于是用这个数组单元来存储这个元素；也可以简单的理解为，按照关键字为每一个元素“分类”，然后将这个元素存储在相应“类”所对应的地方。

但是，不能够保证每个元素的关键字与函数值是一一对应的，因此极有可能出现对于不同的元素，却计算出了相同的函数值，这样就产生了“冲突”，换句话说，就是把不同的元素分在了相同的“类”之中。后面我们将看到一种解决“冲突”的简便做法。

总的来说，“直接定址”与“解决冲突”是哈希表的两大特点。

2. 函数构造

构造函数的常用方法（下面为了叙述简洁，设 $h(k)$ 表示关键字为 k 的元素所对应的函数值）：

（1）除余法

选择一个适当的正整数 p ，令 $h(k)=k \bmod p$ ，这里， p 如果选取的是比较大的素数，效果比较好。而且此法非常容易实现，因此是最常用的方法。

（2）数字选择法

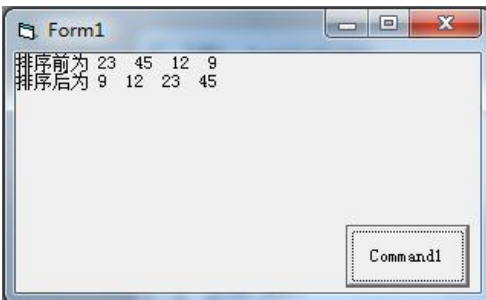
如果关键字的位数比较多，超过长整型范围而无法直接运算，可以选择其中数字分布比较均匀的若干位，所组成的新的值作为关键字或者直接作为函数值。

3.冲突处理

线性重新散列技术易于实现且可以较好的达到目的。令数组元素个数为 S，则当 $h(k)$ 已经存储了元素的时候，依次探查 $(h(k)+i) \bmod S, i=1,2,3,\dots$ ，直到找到空的存储单元为止（或者从头到尾扫描一圈仍未发现空单元，这就是哈希表已经满了，发生了错误。当然这是可以通过扩大数组范围避免的）。

八、常见的排序算法

（一）冒泡排序

定义	依次比较相邻的两个数，将小数放在前面，大数放在后面。由于在排序过程中总是小数往前放，大数往后放，相当于气泡往上升，所以称作冒泡排序																																													
排序过程	<table><tr><td>原始数组</td><td>23</td><td>45</td><td>12</td><td>9</td></tr><tr><td rowspan="4">第 1 趟</td><td>23</td><td>45</td><td>12</td><td>9</td></tr><tr><td>23</td><td>45</td><td>12</td><td>9</td></tr><tr><td>23</td><td>12</td><td>45</td><td>9</td></tr><tr><td>23</td><td>12</td><td>9</td><td>45</td></tr><tr><td rowspan="3">第 2 趟</td><td>23</td><td>12</td><td>9</td><td>45</td></tr><tr><td>12</td><td>23</td><td>9</td><td>45</td></tr><tr><td>12</td><td>9</td><td>23</td><td>45</td></tr><tr><td rowspan="2">第 3 趟</td><td>12</td><td>9</td><td>23</td><td>45</td></tr><tr><td>9</td><td>12</td><td>23</td><td>45</td></tr></table>	原始数组	23	45	12	9	第 1 趟	23	45	12	9	23	45	12	9	23	12	45	9	23	12	9	45	第 2 趟	23	12	9	45	12	23	9	45	12	9	23	45	第 3 趟	12	9	23	45	9	12	23	45	<p>第一趟：第 1 个和第 2 个数，将小数放前，大数放后。然后比较第 2 个数和第 3 个数，直至比较最后两个数。</p> <p>第二趟：仍从第一对数开始比较，将小数放前，大数放后，一直比较到倒数第二个数。</p> <p>第三趟：从前两个开始，比较到倒数第三个数。</p> <p>.....</p> <p>重复以上过程，直至最终完成排序。</p>
原始数组	23	45	12	9																																										
第 1 趟	23	45	12	9																																										
	23	45	12	9																																										
	23	12	45	9																																										
	23	12	9	45																																										
第 2 趟	23	12	9	45																																										
	12	23	9	45																																										
	12	9	23	45																																										
第 3 趟	12	9	23	45																																										
	9	12	23	45																																										
举例	<p>使用冒泡排序方法将数组 a 中的数从小到大排好序</p> <pre>Private Sub Command1_Click() Dim a Dim b As Integer Dim i As Integer Dim j As Integer a = Array(23, 45, 12, 9) Print "排序前为"; For i = 0 To 3 Print a(i); Next i For i = 0 To 2 For j = 0 To 2 - i If a(j) > a(j + 1) Then b = a(j + 1) a(j + 1) = a(j) a(j) = b End If Next j Next i Print Print "排序后为"; For i = 0 To 3 Print a(i); Next i End Sub</pre> 																																													

（二）选择排序

	<pre>Private Sub Command1_Click() Dim a '数组定义 Dim i, j, k, temp As Integer '变量定义 a = Array(20, 6, 15, 7, 3, 6) '数组元素赋值 For i = 1 To 5 'i 是循环变量, j是指针变量, k是暂时变量 k = a(i) '将a(i)作为比较与插入的目标 For j = 0 To i - 1 '将a(i)与其左边的元素a(j)比较 If k < a(j) Then '若a(i) < a(j) For temp = i To j + 1 Step -1 'temp 是移动元素的循环变量 a(temp) = a(temp - 1) '将a(i)左边各元素往右移一位 Next a(j) = k '将较小的元素a(i)插入到第j个位置 Exit For '结束一轮比较与插入, 跳出j循环 End If Next 'j 循环继续, 准备将a(i)与右边剩余的元数作比较 Next 'i 循环继续, 准备将a(i+1)作为比较与插入目标 For i = 0 To 5 Print a(i); Next i End Sub</pre> 
算法分析	<p>直接插入排序的比较次数取决于原记录序列的有序程度。如果原始记录的关键字正好为递增顺序时, 比较次数最少为 $n-1$ 次; 如果为递减顺序时, 比较次数最多, 为 $(n-1)(n+2)/2$ 次, 因此, 直接插入排序的时间复杂度为 $O(n^2)$。</p>

(四) 排序算法的比较

类别	排序方法	时间复杂度			空间复杂度	稳定性
		平均情况	最好情况	最坏情况	辅助存储	
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	不稳定

第二节 程序设计基础

一、程序设计语言产生与发展过程

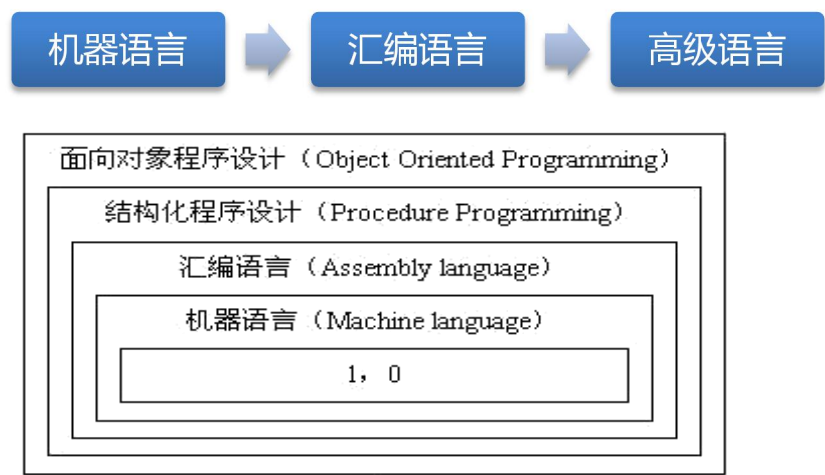
（一）基本概念

概念	定义
程序	可以被连续执行的指令的集合。
程序设计	用计算机语言（程序语言）对所要解决的问题中的数据以及处理问题的方法和步骤进行描述的过程。
程序设计语言	指一切用于书写计算机程序的语言。

（二）程序设计语言介绍

1.发展历程

计算机语言的发展经历了从机器语言、汇编语言到高级语言的历程，如图所示。



2.不同程序设计语言的比较

语言	定义	优点	缺点	举例									
机器语言	是由二进制数组成的指令集，是计算机唯一能识别并直接执行的语言。	不用编译直接执行，执行效率高。	可读性差，可移植性差，编写困难，维护困难。	0000：代表 加载（LOAD） 0001：代表暂存器 B 0000000000001：代表地址为 1 的内存 0000,0000,000000000001 ： 代 表 LOAD B, 1									
汇编语言	用助记符表示计算机指令，需要编译才能执行。	容易读懂，维护方便，执行效率高。	编写效率不高。	<div>例如，要计算 $c=7+8$，可以用如下几条汇编命令：</div> <table><thead><tr><th>标号</th><th>指令</th><th>说明</th></tr></thead><tbody><tr><td>START</td><td>GET 7； ADD 8； PUT C；</td><td>把 7 送进累加器 ACC 中 累加器 ACC+8 送进累加器 ACC 中 把累加器 ACC 送进 C 中</td></tr><tr><td>END</td><td>STOP；</td><td>停机</td></tr></tbody></table>	标号	指令	说明	START	GET 7； ADD 8； PUT C；	把 7 送进累加器 ACC 中 累加器 ACC+8 送进累加器 ACC 中 把累加器 ACC 送进 C 中	END	STOP；	停机
标号	指令	说明											
START	GET 7； ADD 8； PUT C；	把 7 送进累加器 ACC 中 累加器 ACC+8 送进累加器 ACC 中 把累加器 ACC 送进 C 中											
END	STOP；	停机											

语言	定义	优点	缺点	举例
高级语言 (分面向过程, 面向对象)	是一种用能表达各种意义的“词”和“数学公式”按一定的“语法规则”编写程序的语言。需要编译才能执行。	不依赖于计算机型号, 通用性较好, 编写效率高, 可读性好。		C 语言的范例: 求 $1 \times 2 \times 3 \cdots \times 100$ <pre> #include <stdio.h> main() {int i ,sum; i=1, sum=1; while(i<101) {i++; sum=sum*i; } printf("sum=%d\n",sum); }</pre>

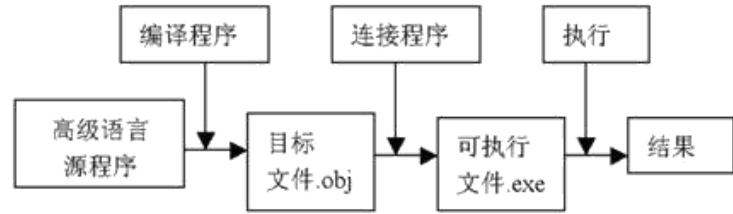
(三) 高级语言源程序翻译成机器语言程序的方法

高级语言源程序翻译成机器语言程序的方法有“解释”和“编译”两种。

解释方法采用边解释边执行的方法, 如早期的 BASIC 语言即采用解释方法, 在执行 BASIC 源程序时, 解释一条 BASIC 语句, 执行一条语句。

编译方法采用相应语言的编译程序, 先把源程序编译成指定机型的机器语言目标程序, 然后再把目标程序和各种标准库函数连接装配成完整的目标程序, 在相应的机型上执行。如 C、C++、Visual C++ 及 Visual Basic 等均采用编译的方法。

编译方法比解释方法更有效率。



二、程序设计的特点

程序设计分两大类, 面向过程和面向对象。

(一) 面向对象简介

面向对象是尽可能模拟人类习惯的思维方式, 使开发软件的方法与过程尽可能接近人类认识世界解决问题的方法与过程。

(二) 面向过程简介

面向过程也就是结构化程序设计是一种自顶向下逐步求精的设计方法, 和单入口单出口的程序结构。以设计五子棋游戏为例, 面向过程和面向对象处理方式对比如下:

类别	面向对象	面向过程
----	------	------

游戏设计思路 和步骤	1.黑白双方，这两方的行为是一模一样的； 2.棋盘系统，负责绘制画面； 3.规则系统，负责判定诸如犯规、输赢等。 第一类对象(玩家对象)负责接受用户输入，并告知 第二类对象(棋盘对象)棋子布局的变化，棋盘对象 接收到了棋子的变化就要负责在屏幕上面显示出 这种变化，同时利用第三类对象(规则系统)来对棋 局进行判定。	1.开始游戏； 2.黑子先走； 3.绘制画面； 4.判断输赢； 5.轮到白子； 6.绘制画面； 7.判断输赢； 8.返回步骤 2； 9.输出最后结果。 把上面每个步骤用分别的函数来实现，问题就解决了。
加入悔棋功能	棋盘系统保存了黑白双方的棋谱，改动棋盘对象，使其回溯即可，其他不变。可扩充性好。	从输入到判断到显示这一连串的步骤都要改动，甚至步骤之间的循序都要进行大规模调整。可扩充性差。
划分问题的依据	以功能划分。	以步骤划分。

（三）结构化程序设计（面向过程）

1.设计原则

自顶向下、逐步求精、模块化的组织方式、结构化的语句结构。

2.结构化程序设计的优点

- （1）算法描述准确。
- （2）对每一子过程模块容易进行程序正确性证明。

3.结构化程序设计的缺点

- （1）本质上是面向“过程”的，不能直接反映人类求解问题的思路。
- （2）程序代码可重用性差。
- （3）维护程序的一致性困难。

（四）面向对象程序设计

1.面向对象程序设计方法

面向对象程序设计强调的是数据对象，建立层次化的对象体系。它是基于解决问题的业务逻辑，而不是基于具体的程序步骤。

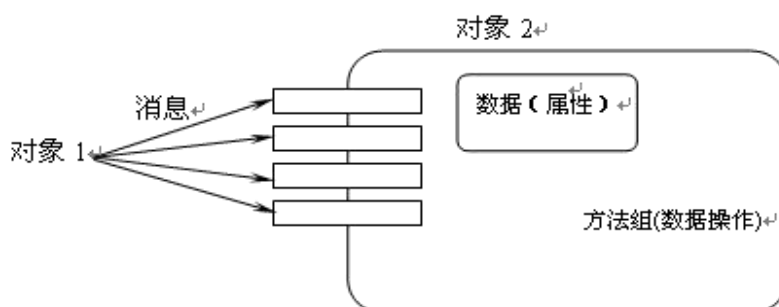
面向对象的程序设计方法可以表示为：

面向对象=对象+类+继承+消息通信



2.面向对象程序的工作原理

从面向对象的角度看，程序是对象的集合；对象之间的相互作用构成了一个软件系统。对象参与的交互动作称为事件。通过事件，消息在对象之间发送，接收消息的对象调用相应的方法进行响应。面向对象程序的工作原理如图所示。

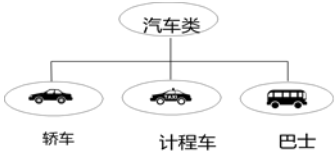



3.面向对象程序设计的优点

- (1) 抽象模型更接近现实世界
- (2) 模型与程序易于理解
- (3) 程序具有重用性和模块化
- (4) 软件易于维护

4.面向对象技术的主要概念

概念	定义	示意图	特点
对象	描述客观事物的一个实体，它是构成系统的一个基本单元，由属性和方法组成。	<div>对象名 学生王强</div> <div>属性 学号、姓名、专业、性别、年龄、身份证号、电话等等</div> <div>方法 报到注册、交学费、课程选修、考试、毕业</div>	1.一体性：对象名、属性、行为三者一体。 2.属性是静态描述。 3.行为是动态描述。 4.外部是通过调用服务与对象取得联系。
类	具有相似属性和行为的一组对象。	水果——>类 苹果——>对象	类是对象的抽象。 实例是类的实现。
封装	把对象的属性和方法结合成一个独立的单元，并尽可能隐蔽对象的内部细节。封装使数据具有独立性、隐藏性、安全性。		1.数据独立。 2.隐藏性、安全性。 易于维护。 3.但需要更多的输入输出函数。

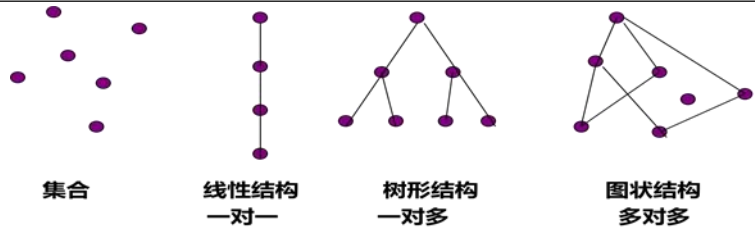
继承	在面向对象的程序设计中，允许在已有类的基础上通过增加新特征而派生出新的类，这称为继承。		1.共享性。 2.区别性：派生类不等于基类。 3.层次性：表示了类之间的关系。
多态性	在通过继承而派生出一系列类中，可能存在一些名称相同，但实现过程和功能不同的方法。		用户不必知道某个对象所属的类就可以执行多态行为，从而为程序设计带来更大方便。
动态联编	编译程序在编译阶段并不能确切知道将要调用的函数，只有在程序运行时才能确定将要调用的函数，这种在程序运行时进行联编工作被称为动态联编。		

（五）面向对象程序设计语言介绍

语言	特点
Java	Java 是一种可以撰写跨平台应用软件的面向对象的程序设计语言。 Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于个人 PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群。在全球云计算和移动互联网的产业环境下， Java 更具备了显著优势和广阔前景。
C++	C++是一种静态数据类型检查的、支持多重编程范式的通用程序设计语言。它支持过程化程序设计、数据抽象、面向对象程序设计、泛型程序设计等多种程序设计风格。
C#	是微软公司发布的一种面向对象的、运行于.NET Framework 之上的高级程序设计语言，它是第一个面向组件的编程语言。
VB.NET	Visual Basic.NET 是基于微软.NET Framework 之上的面向对象的中间解释性语言，可以看作是 Visual Basic 在。Visual Basic.NET 的应用范围包括 Windows 桌面、Web 以及当下突然觉醒的正在奋力追赶的第三大移动平台 Windows Phone。
VB	Visual Basic 是一种由微软公司开发的包含协助开发环境的事件驱动编程语言，“面向对象”的特性，VB 的程序是一种基于窗体的可视化组件安排的联合，并且增加代码来指定组件的属性和方法。但不支持继承，无原生支持多线程、异常处理不完善等三项明显缺点，是入门级语言。

第三节 数据结构

一、数据结构简介

定义	是相互之间存在一种或多种特定关系的数据元素的集合。	
意义	计算机内的数值运算依靠方程式，而非数值运算（如表、树、图等）则要依靠数据结构。	
四种结构	 <div>集合 线性结构 一对一 树形结构 一对多 图状结构 多对多</div>	
逻辑结构	定义	逻辑结构指数据元素之间的逻辑关系。即从逻辑关系上描述数据，它与数据的存储无关，是独立于计算机的。
	分类	<div><div>数据结构</div><div><div>线性结构</div><div>非 线 性 结 构</div></div><div><div>1. 线性表</div><div>2. 栈</div><div>3. 队列, 双队列</div><div>4. 数组</div><div>5. 字符串</div><div>1. 树, 二叉树</div><div>2. 图</div></div></div>
物理结构	定义	物理结构亦称存储结构，是数据的逻辑结构在计算机存储器内的表示（或映像）。它依赖于计算机。
	分类	<p>顺序存储结构：用一组地址连续的存储单元依次存储线性表的各个数据元素。</p> <p>链式存储结构：用一组任意的存储单元存储线性表的数据元素(这组存储单元可以是连续的，也可以是不连续的)。</p> <p>索引存储结构：通过建立附加的索引表来标识结点的地址的存储结构。</p> <p>散列存储结构：将关键字与存储位置建立关联的存储结构。</p>

二、线性表

（一）基本概念

定义	线性表是 n ($n \geq 0$) 个数据元素的有限序列
格式	(a_1, a_2, \dots, a_n) ，数据元素 a_i ($1 \leq i \leq n$) 只是一个抽象的符号
举例	26 个英文字母组成的字母表可表示为： (A, B, C, \dots, Z)
特性	<p>同一性：线性表由同类数据元素组成，每一个 a_i 必须属于同一数据对象。</p> <p>有穷性：线性表由有限个数据元素组成，表长度就是表中数据元素的个数。</p> <p>有序性：相邻数据元素之间存在着序偶关系 $\langle a_i, a_{i+1} \rangle$。均匀性，有序性。</p>

类型	顺序存储	链式存储
特点	逻辑结构中相邻的数据元素在存储结构中仍然相邻；线性表的顺序存储结构是一种随机存取的存储结构。	用一组任意的存储单元存储线性表的元素，不要求逻辑上相邻的元素在物理位置上也相邻。
实现方式	方法简单，各种高级语言中都有数组类型，容易实现。	基于指针的，相对来讲复杂些。
存储空间和分配	存储空间是静态分配的，在程序执行之前必须明确规定它的存储规模。	动态分配存储空间的，不用事先估计存储规模。
运算实现	按序号访问数据元素操作方便。	插入删除操作方便。

（二）链表

链表采用链式存储实现的线性表。根据其存储结构的不同，可以将它分为单链表、双链表和循环链表三类。

分类	单链表	
	双链表	
	循环链表	

单链表	示意图	说明
定义		每个结点包括两个域：数据、指针（指向下一个结点），最后一个结点的指针为 null，head 表示头指针
删除		<ol style="list-style-type: none"> 1.找到要删除的结点 q 和前驱 p; 2.将 p 的指针指向 q 所指向的后继结点 <p>$p \rightarrow next = q \rightarrow next$;</p>

		该操作的算法平均时间复杂度 $O(n)$ 。
插入		1. 找到插入点的前驱 p ; 2. 将要插入的结点 q 的指针指向 p 的后继结点: $q \rightarrow next = p \rightarrow next$; 3. 将 p 的指针指向 q : $p \rightarrow next = q$ 。

三、栈

定义	限定只在表的一端（表尾）进行插入和删除操作的线性表	
特性	后进先出	
示意图		
分类	顺序栈，链栈。	

四、队列

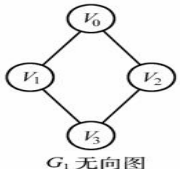
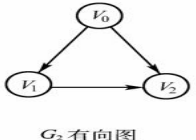
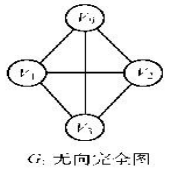
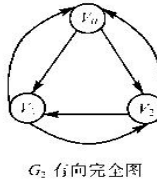
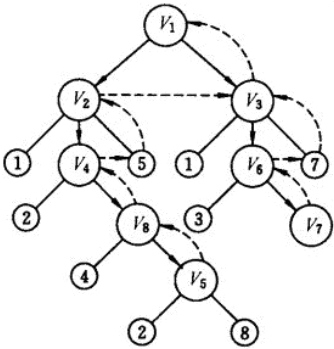
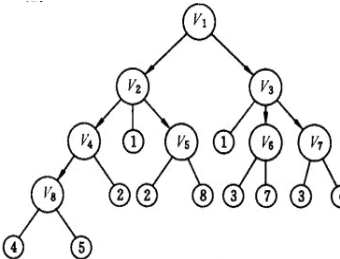
定义	队列是限定在表的一端进行删除，在表的另一端进行插入操作的线性表。允许删除的一端叫做队头（front），允许插入的一端叫做队尾（rear）。	
特性	先进先出	
示意图		
分类	链队列	
	循环队列	

五、树和二叉树

树的定义	<p>是 n 个结点的有限集 T，T 为空时称为空树，否则它满足如下两个条件：</p> <p>1.有且仅有一个特定的称为根（Root）的结点；</p> <p>2.其余的结点可分为 m ($m \geq 0$) 个互不相交的子集 $T_1, T_2, T_3, \dots, T_m$，其中每个子集又是一棵树，并称其为子树（Subtree）。</p>	
树的应用 文件系统		
二叉树	是每个结点至多只有两个子结点的树。	
二叉树的度	度就是二叉树中结点的子分支数。	
二叉树示意图		
二叉树的遍历	按某条搜索路径巡访树中的每一个结点，使得每一个结点均被访问一次，而且仅被访问一次。以上图满二叉树为例：	
	先序遍历：根结点→左子树→右子树	1—2—4—5—3—6—7
	中序遍历：左子树→根结点→右子树	4—2—5—1—6—3—7
	后序遍历：左子树→右子树→根结点	4—5—2—6—7—3—1
二叉树的性质	性质 1：在二叉树的第 i 层上至多有 2^{i-1} 个结点	
	性质 2：深度为 k 的二叉树至多有 $2^k - 1$ 个结点 ($k \geq 1$)	
	性质 3：具有 n 个结点的完全二叉树的深度为： $\lfloor \log_2 n \rfloor + 1$	
	性质 4： $n_0 = n_2 + 1$ (n_0 表示叶子结点的个数， n_2 表示度为 2 的非叶子结点的个数)	
满二叉树	除最后一层无任何子结点外，每一层上的所有结点都有两个子结点的二叉树。	

六、图

图的定义	<p>图是一种数据结构，图中的数据元素通常用顶点来表示，而数据元素间的关系用边来表示，故图可定义为:图 G 由两个集合 $V(G)$ 和 $E(G)$ 所组成，记作 $G = (V, E)$ 其中 $V(G)$ 是图中顶点的非空有限集合，$E(G)$ 是边的有限集合（边是 $V(G)$ 中顶点的无序对或有序对集合）。</p>
------	--

分类	<p>无向图:</p> <p>如果图中每条边都是顶点的无序对，即每条边在图示时都没有箭头，则称此图为无向图。</p>	 <p>G_1 无向图</p>
	<p>有向图:</p> <p>如果图中每条边都是顶点的有序对，即每条边在图示时都用箭头表示方向，则称此图为有向图。</p>	 <p>G_2 有向图</p>
	<p>无向完全图:</p> <p>若一个无向图有 n 个顶点，且每一个顶点与其他 $n-1$ 个顶点之间都有边，这样的图称为无向完全图。</p> <p>有 $n(n-1)/2$ 条边</p>	 <p>G_3 无向完全图</p>
	<p>有向完全图:</p> <p>若一个有向图有 n 个顶点，且每一个顶点与其他 $n-1$ 个顶点之间都有一条以该顶点为弧尾的弧，这样的图称为有向完全图。</p> <p>有 $n(n-1)$ 条边</p>	 <p>G_4 有向完全图</p>
图的遍历	<p>从图的任一顶点出发访问图中的各个顶点，并且使每个顶点仅被访问一次。这一过程叫做图的遍历。对图的遍历通常采用两种遍历次序，即深度优先搜索（DFS）和广度优先搜索（BFS）。</p>	
	<p>深度优先搜索:</p> <p>深度优先搜索就是在搜索树的每一层始终先只扩展一个子节点，不断地向纵深前进直到不能再前进（到达叶子节点或受到深度限制）时，才从当前节点返回到上一级节点，沿另一方向又继续前进。这种方法的搜索树是从树根开始一枝一枝逐渐形成的。</p>	
	<p>广度优先搜索:</p> <p>广度优先搜索属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。</p>	

第四节 LOGO 语言

一、LOGO 语言概述

（一）来源

LOGO 源自希腊文，原意即为思想。

（二）概念

一种早期的编程语言，与自然语言非常接近的编程语言，它通过“绘图”的方式来学习编程。

（三）语言特点

1.针对性

对于儿童来说，“画画”比“文字处理”更具有活力。

2.易操作

通过向前，后退、向左转、向右转、回家等儿童易于理解的语言和命令，这非常适合儿童的知识水平。

二、LOGO 语言的启动

单击“开始”按钮，指向“程序”，单击“PC Logo”命令，出现“PC Logo for Windows”窗口。



屏幕的上方是视图窗口，中间的“小海龟”就是 Logo 语言的绘图工具，通过小海龟上、下、左、右移动来实现画图。屏幕中间出现的小海龟的位置就是小海龟的“家”，通常称它的为“母位”。

屏幕的下方是命令窗口，在命令窗口中输入 Logo 语言命令可以指挥小海龟运动。在命令窗口中，“？”是 Logo 语言的提示符，“|”是光标。

logo 语言常用命令

命令	含义	命令	含义
----	----	----	----

DRAW	清屏、海龟回母位	CLEAN	清除基本命令
CS	清屏并复位	CT	清除先前所有文本
BK	后退	RT	右转
LT	左转	FD	前进
PU	抬笔	PD	落笔
HT	藏龟	ST	显龟
HOME	回家	REPEAT	重复
SS	图文混合屏	FS	全图形屏
TS	全文字屏	ND	全文字屏、清屏
WRAP	环绕状态	WINDOW	窗口状态
FENCE	围栏状态	SHOW	屏幕输出
EDIT ALL	进入编辑部	STOP	停止
IF THEN STOP	如果 那么 停止	PAUSE	正确
TT	在画图区显示文字	TO END	过程头、尾
SETPC	设置画笔颜色	SETBG	设置屏幕底色

（一）基本命令

1. “HT”——藏龟命令（hideturtle）

图形画好后，屏幕上还留有海龟标记，若要去掉这个标记，可以输入“HT”命令，并按回车键。

2. “ST”——显龟命令（showturtle）

要使海龟重新出现在屏幕上，可以输入“ST”命令，并按回车键。

3. “CS”——清屏复位命令（clearscreen）

如果想在屏幕上画新的图形，可以使用“CS”命令将屏幕“擦”干净，同时使海龟回到母位，且方向朝上。

4. “FD”——前进命令

“FD”（forward）是前进命令，它必须与数字配合输入。因此一个完整的“FD”命令分两部分：前半部分为命令“FD”，表示海龟向前爬行；后半部分为具体数字，表示海龟向前爬行的具体长度，一般称为步数。注意两部分之间一定要用空格分开。例如，“FD 20”表示海龟向前爬行 20 步。

5. “BK”——后退命令

“BK”（back）是后退命令，它的作用与“FD”命令相反，表示海龟向后走。

6. “RT”——右转命令

“RT”（right）是右转命令，“RT 90”表示海龟向右转 90°，也就是向右转一直角。

7. “LT”——左转命令

“LT”（left）是左转命令，“LT 90”表示海龟向左转 90°，也就是向左转一个直角。

（二）实例学习

1.画虚线



```
rt 90 pd fd 40 pu fd 60 pd fd 40
```

```
pu fd 60 pd fd 40 pu fd 60
```

使用“PU”（penup）命令后，小海龟前进或者后退时都不留痕迹，屏幕上不显示任何线条，直至使用“PD”（pendown）命令后，才能画出线条。

2.画正多边形



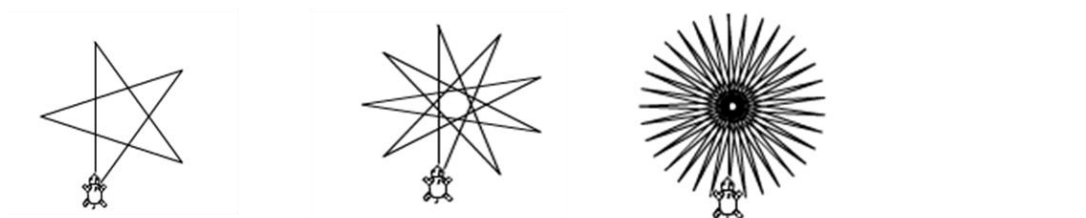
等边三角形：REPEAT 3[FD 50 RT 120]

正方形：REPEAT 4[FD 50 RT 90]

八边形：REPEAT 8[FD 40 RT $360^\circ / 8$]

圆：REPEAT 36[FD 8 RT 10]

3.画星星



五角星：REPEAT 5[FD 100 RT 144]

九角星：REPEAT 9[FD 100 RT 160]

33角星：REPEAT 33[FD 100 RT $180^\circ - 180^\circ / 33$]

4.彩色世界



紫色五角星：

```
SETPC 4 REPEAT 5[FD 70 RT 144]
```

三角红旗：

```
SETPC 12
```

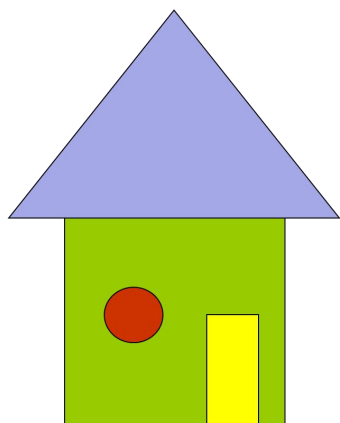
```
REPEAT 3[FD 100 RT 120]
```

```
PU RT 30 FD 10
```

```
PD FILL
```


颜色	代码	颜色	代码	颜色	代码
黑色	0	棕色	6	浅红色	12
蓝色	1	浅灰色	7	浅紫色	13
绿色	2	深灰色	8	浅黄色	14
青色	3	浅蓝色	9	白色	15
红色	4	浅绿色	10		
紫色	5	浅青色	11		

5.组合图形



RT 30（确定画房顶的起始角度）

REPEAT 3[FD 100 RT 120]（画三角形的房顶）

RT 60 FD 10（将小海龟移动到画房身的起始位置及角度）

REPEAT 4[FD 80 RT 90]（画正方形的房身）

FD 80 RT 90 FD 80 RT 90 FD 10 RT 90（将小海龟移动到画门的起始位置及角度）

REPEAT 2[FD 55 LT 90 FD 25 LT 90]（画长方形的房门）

PU FD 50 LT 90 FD 60 RT 90 PD（将小海龟移动到画窗的起始位置及角度）

REPEAT 36[FD 2 RT 10]（画圆形的窗）

HT（隐藏小海龟）

6.定义过程

过程的一般格式为：

TO 过程名（过程头）

过程体（过程体）

END（过程尾）

案例：定义画一个边长为 50 步的正方形的过程。

TO ZFX

REPEAT 4[FD 50 RT 90]

END

此后，只要在命令窗口中输入“ZFX”后按回车键，便可画出一个边长为 50 步的正方形。定义好

的过程在退出 Logo 前可以重复使用。

7.带参数的过程

过程里有参数了，称为带参数的过程。

TO 过程名 参数名 或 TO 过程名 参数名 参数名 参数名……

过程体

过程体

END

END

例如：

TO YUAN :R (:R 为圆的半径)

REPEAT 36 [FD :R*0.174 RT 10]

END

TO A :N (:N 为重复次数)

CS REPEAT :N [YUAN 60 RT 360/:N]

END

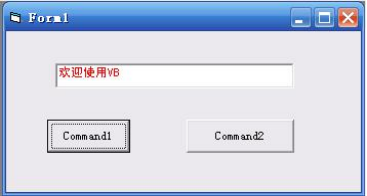
第五节 VB 语言

一、Visual Basic 中的对象

(一) 对象分类

对象分类	说明
预定义对象	由系统设计好提供给用户使用，如窗体、控件、打印机、剪贴板、屏幕等。
用户建立对象	就是新建窗体以及在窗体上绘制控件的过程。对象也可通过程序来建立，但必须由用户在程序中为对象命名，如窗体 Form1，按钮(控件)button 等。

(二) 对象的属性、方法和事件

要素	说明	设置或调用方法
属性	VB 对象常见属性有标题 (Caption)、控件名称 (Name)、颜色 (Color)、字体大小 (Font size)、是否加粗 (Font Bold)、是否可见 (Visible) 等。	设置对象属性语法格式： [对象名.]属性名=属性值 Text1.Text="欢迎光临"
方法	能执行的动作或功能，如显示或打印、绘图、移动。	调用对象的语法格式： [对象名.]方法名 [参数列表] Txtname.SetFocus Form1.Print "欢迎使用 VB"
事件	事件是 VB 预先设置好的、能够被对象识别的动作，如 Click (单击)、DbClick (双击)、MouseMove (移动鼠标)、Load (装入) 等。	<pre>Private sub Command1_Click() Text1.FontName="黑体" Text1.ForeColor=vbRed Text1.Text="欢迎使用 VB" End Sub</pre> 

(三) VB 程序结构与事件驱动编程机制

1.VB 程序的结构

VB 应用程序以“对象”为中心来设计模块，它通常由窗体模块、标准模块和类模块三种模块组成。



程序结构	说明
窗体模块	文件扩展名为.frm，其中包含窗体的图形描述；其控件以及控件的属性设置；事件过程和通用过程等。
标准模块	扩展名为.BAS，它完全由代码组成，代码不与具体的窗体或控件相关联，但可以被窗体模块中的任何事件过程调用。
类模块	扩展名为.CLS，可以在窗体模块中创建类的对象，从而调用类模块中的过程。

2.事件驱动编程机制

事件驱动编程机制具有如下几个要点：

- (1) 应用程序基于对象组成。
- (2) 每个对象都有预定义的事件集。
- (3) 每个事件的发生都依赖于一定的条件（用户或系统驱动）。
- (4) 每个事件发生后的响应取决于事件过程中的程序代码事件驱动程序的核心机制是由用户控制事件的发生，即用户发出什么动作（事件），事件驱动应用程序（相关联的事件过程）执行程序代码，做出响应。

例如，当用户单击窗体，则执行窗体的单击事件过程中代码；如果单击命令按钮，则执行命令按钮的单击事件过程中代码。

（四）窗体与控件

1.窗体

常用属性	常用方法
------	------

<p>Name: 窗体的名字, 默认值 Form1</p> <p>Caption: 窗体标题</p> <p>BackColor: 窗体背景颜色</p> <p>ForeColor: 前景颜色</p> <p>BorderStyle: 窗体的边框样式, 0~5 的整数</p> <p>Height,Width: 指定窗体的高度和宽度</p>	<p>Form_Click(): 单击窗体触发</p> <p>Form_DblClick(): 双击窗体触发</p> <p>Load: 载入窗体触发</p> <p>Unload: 卸载窗体触发</p> <p>Activate: 窗体变为活动触发</p> <p>Paint: 窗体移动或缩放触发</p> <p>Print: 输出显示</p> <p>Cls: 清除 Print 显示内容</p> <p>Move: 移动, 改变大小</p> <p>Show 或 Hide: 显示隐藏窗体</p>
<p>举例: 窗体单击事件过程 Form1_Click()</p>	
<pre>Private Sub Form1_Click() Form1.Print "how are you";"?" Form1.Print "123456","ABCDEF" Picture.Print "计算机世界" Picture.Print "教材书";5+20;"本" End Sub</pre>	
<pre>Private Sub Form1_Click() Move 500,500,3800,2500 Text1.Move 200,200,1500,1000 End Sub</pre>	<p>该事件过程先把窗体移到距屏幕左边界 500, 上边界 500 的位置处, 并将其大小设置为宽度 3800 和高度 2500; 然后把文本框移到窗体的 (200,200) 处, 把大小设置为宽 1500, 高 1000</p>

2. 控件的使用

(1) 控件分类

在设计用户界面时, 需要在窗体上画出各种所需的控件, 控件是构成用户界面的基本元素。控件类型如下:

控件分类	说明
标准控件 (也称内部控件)	内部控件以图标形式在工具箱中列出, 如标签、文本框、图片框、命令按钮、列表框等。
ActiveX 控件	各种版本 VB 提供的控件, 仅在专业版和企业版中提供的控件, 以及第三方提供的 ActiveX 控件。
可插入对象	该对象能添加到工具箱中, 可以被当作控件。

(2) 常用控件属性

常用控件属性		说明
公共属性	Name	控件的名字
	Caption	控件上显示的文字内容

	Visible	控件的是否可见，取值 True 或 False
	FontName	字体
	FontSize	字号
	FontBold	粗体字
	FontItalic	斜体字
	FontUnderline	下划线
	Enabled	逻辑型，确定对象是否可用，取值 True 或 False
	Alignment	文字对齐方式，0-左对齐、1-右对齐、2-居中
标签 特有属性	AutoSize	逻辑型，根据文字自动调整标签大小
	WordWrap	逻辑型，是否自动换行
文本框 特有属性	Text	设置文本框中显示的内容
	MaxLenght	允许在文本框中输入的最大字符数
	Multiline	是否允许多行，取值 True-多行，取值 False-单行
	ScrollBars	确定文本框中是否有滚动条
	PasswordChar	密码占位符，主要用来输入口令
	Locked	逻辑型，确定文本框是否可被编辑
按钮 特有属性	Cancel	设置为 True 时，单击该命令按钮与按 Esc 键的作用相同
	Default	设置为 True 时，按回车键与单击该命令按钮的作用相同

SetFocus 方法：用于将焦点移到指定的命令按钮。即该按钮被激活或获得焦点，该按钮在其内侧有一个虚线框。

Click 事件：单击命令按钮时发生。它不支持 DblClick（双击）事件。

命令按钮的方法和事件举例：

例如：在一窗体上有三个命令按钮，Caption 属性分别设置为“红色”、“绿色”、“还原”字样，字体统一设置为宋体、粗体、四号。程序一启动，单击“红色”按钮，窗体背景色被设置为红色，“还原”按钮获得焦点；此时按下回车键，窗体背景色还原为默认颜色；单击“绿色”按钮，窗体背景色变为绿色；再次单击“还原”，窗体背景色又一次还原。



各命令按钮对应单击（Click）事件过程如下：

```

Dim s                                '定义全局变量
Private Sub Command1_Click()
s=BackColor                        '保存窗体默认背景色到 s
BackColor=vbRed                    '设置窗体背景色为红色
Command3.SetFocus                  '还原按钮获得焦点
End Sub
Private Sub Command2_Click()
BackColor=vbGreen                  '设置窗体背景色为绿色

```

```
End Sub
Private Sub Command3_Click()
BackColor=s      '设置窗体背景色为默认颜色
End Sub
```

二、数据类型

类型	表示方法	表示范围
整型	Integer	-32768～32767 范围内的任何整数
长整型	Long	-2147483648～2147483647 范围内的任何整数
单精度 实数型	Single	绝对值在 1.401298E-45～3.402823E+38 内的任何实数，有效数字约 6~7 位
双精度 实数型	Double	绝对值在 10E-324～1.79E308 内的任何实数，有效数字约 6~7 位
逻辑型	Boolean	True 或 False
字节型	Byte	0～255
货币型	Currency	-922337203685477.5808～+922337203685477.5807
日期型	Date	100 年 1 月 1 日～9999 年 12 月 31 日
字符串	String	0～约 20 亿
变体类型	Variant	变体类型能支持所有简单的数据类型，如整型、浮点、字符串、布尔型、日期时间、货币及 OLE 自动化对象等

三、变量

变量的命名规则如下：

- （一）首字符必须是字母；
- （二）只能由字母、数字和下划线组成；
- （三）不能使用 VB 中的关键字（保留字）；
- （四）在同一个作用域内必须唯一；
- （五）变量名长度不超过 255。

变量的作用域如下表所示：

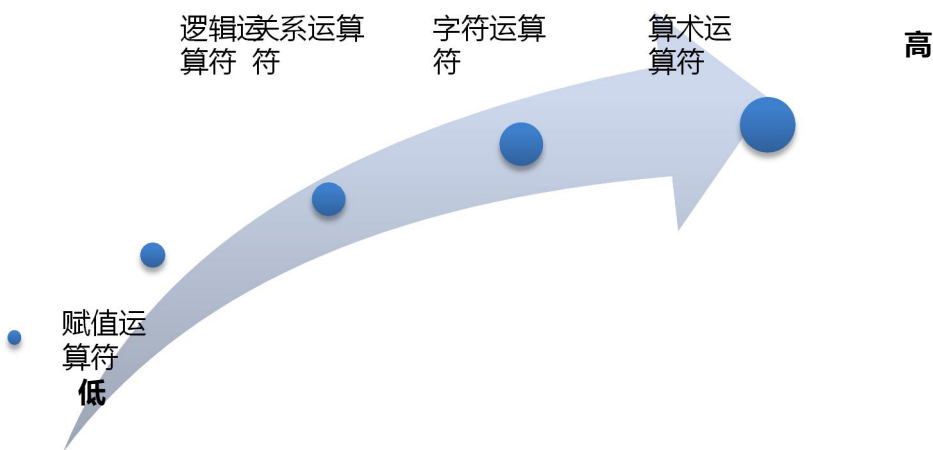
等级	声明方式	声明位置	作用范围	举例
局部	Dim, Static	过程内部	仅在过程中有效	Dim A(3) As Integer Static bring As String
窗体/模块	Dim, Private	窗体或模块的声明 段	在窗体或模块的过程中有 效	Dim a As Integer,b As Long
全局	Public	窗体或模块的声明 段	在工程的所有过程内有效	Public abc(2,4) As Integer

四、运算符与表达式

类别	运算	运算符	优先级	表达式
算术运算	乘方	^	9	2^8 表示 2 ⁸
	负号	-	8	-3 表示负 3
	乘法	*	7	a*b 表示 a 乘以 b
	除号	/	7	5.2/2 计算结果为 2.6

	整除	\	6	5\2 计算结果为 2
	求余数	Mod	5	17 mod 3 计算结果为 2
	加法	+	4	a+b 表示 a 加 b 的和
	减法	-	4	a-b 表示 a 减去 b 的差
字符运算	连接	+, &	4	"中国" & "江苏" "中国" + "江苏" 结果均为 "中国江苏"
关系运算	大于	>	3	100>99 结果为 True
	小于	<	3	1.2<-23 结果为 False
	大于等于	>=	3	Sin(1)>=0 结果为 True
	小于等于	<=	3	Sqr(3)<=0 结果为 False
	等于	=	3	12=13 结果为 False
逻辑运算	非（求反）	Not	2	Not(12<>13)结果为 False
	与（并且）	And	1	(23>10) and (30<23)结果为 False
	或（或者）	Or	0	(23>10) or (30<23)结果为 True
赋值运算	等于	=	0	A=3

优先级数字越大，优先级别越高，在进行运算时越先计算。运算符的优先级为：算术运算符→字符运算符→关系运算符→逻辑运算符→赋值运算符。



五、VB 中的函数

函数名称	功能
Abs(x)	以相同的数据类型返回一个数字的绝对值：abs(-3.5)=3.5
Asc(x)	返回指定字符串中第一个字符的 ASCII 码值：asc("A")=65
Atn(x)	返回指定数字的反正切值
Chr(x)	返回指定的 ASCII 代码所对应的字符：chr(65)= "A"
Sqr(x)	以双精度浮点数的形式返回一个数（不小于 0）的平方根：sqr(4)=2.0
Exp(x)	以双精度浮点数的形式返回以 e（自然对数的底）为底的指数
Fix(x)	返回一个数字的整数部分:fix(3.9)=3， fix(3.1)=3， fix(-3.9)=-3
Int(x)	返回一个不大于 x 的最大整数：int(3.9)=3， int(-3.9)=-4
Round(x)	返回一个按指定小数位数的四舍五入的数值
mod(nExp1,nExp2)	即是两个数值表达式做除法运算后的余数
Log(x)	以双精度浮点数的形式返回一个数字的自然对数
Rnd()	以单精度浮点数的形式返回一个随机数

Sin(x)	以双精度浮点数的形式返回一个角度的正弦值
Cos(x)	以双精度浮点数的形式返回一个角度的余弦值
Tan(x)	返回一个角度的正切值
Left(x,m)	返回指定字符串 x 中最左边的 m 个字符
Len(x)	返回指定字符串的字符个数或返回存储某个变量所需要的字节数
Mid(x,m,n)	返回指定字符串 X 中从 m 开始的 n 个字符
Right(x,m)	返回指定字符串 x 中最右边的 m 个字符
Space(x)	返回 x 个空格的字符串
Second()	返回秒数（0~59 之间）
Str(x)	将一个数字转换成对应的数字字符串，并返回该字符串
Val(x)	将一个数字字符串转换成对应的数值
String(x)	返回由若干个同一个字符组成的字符串
Date()	返回当前系统日期
Day()	返回一个 1~31 之间的整数，用来表示一月中的某一天
Minute()	返回一个 0~59 之间的整数，用来代表一小时中的某一分钟
Mouth()	返回一个 1~12 之间的整数，用来代表一年中的某个月份
Now()	返回当前系统的日期和时间
Hour()	返回一个 0~23 之间的整数，用来代表一天中的某个小时
Time()	返回当前的系统时间
Timer()	返回从午夜 0 时开始到现在经过的秒数
Weekday()	返回一个用来表示一星期中某一天的整数
Year()	返回一个用来表示年份的整数

六、VB 中的基本语句

VB 中的程序结构分顺序结构、分支结构和循环结构，不同结构所用的语句如下：

程序结构	分类	语句	说明
分支结构	单分支	If endif	条件为真时执行
	双选条件	If else endif	在两种可能的操作中按条件选取一个执行
	多选条件	If elseif elseif...else endif	在多种可能的操作中按条件选取一个执行
		Select case	
循环结构	当型循环	Do While...Loop 语句	条件为真时循环执行，为假时退出
		Do Until...Loop 语句	条件为假时循环执行，为真时退出
	直到型循环	Do ...Loop While	先执行循环语句，再判断条件，为真继续循环，为假退出
		Do... Loop Until	先执行循环语句，再判断条件，为假继续循环，为真退出
	已知次数的循环	For...Next	判断循环次数，次数不到，执行，循环次数加 1；循环次数到了退出

（一）IF 分支语句

If 语句分类	流程图	举例
If <条件> Then <语句> End If		If wether="晴天" then Action="登山"; End if
If <条件> Then <语句 1> Else <语句 2> End If		If deng="green" then Action="go"; Else Action="stop"; End if
if <条件> Then <语句 1> ElseIf <条件 2> Then <语句 2> Elseif <条件 n-1> Then <语句 n-1> Else <语句 n> End If		If score>85 then Grade="优秀"; ElseIf score>70 then Grade="良好"; ElseIf score>=60 then Grade="合格"; Else Grade="不合格"; End if

(二) Select Case 语句

当程序中分支较多，尤其需要多重嵌套的时候，使用 If 分支语句比较冗长，而且结构也不清晰，为此 VB 提供了一种更加简洁的分支语句 Select Case。该语句对一个结果的多种情况进行判断，Case 表达式的几种形式如下：

CASE 表达式形式	举例
一个固定值	Case 1, Case 3*a+7
几个固定值，逗号分隔	Case 1,3,5
上、下限范围，to 表示	Case 90 To 100
单向范围，is 表示	Case is<60

语法格式和流程图如下：

CASE 语法格式	流程图	代码
-----------	-----	----

Select Case <条件表达式> Case <表达式 1> <语句 1> Case <表达式 2> <语句 2> Case <表达式 n-1> <语句 n-1> Case Else <语句 n> End Select		Select Case mark Case is>89 Label1.Text="优秀" Case 80 To 89 Label1.Text="良好" Case 60 To 79 Label1.Text="合格" Case else Label1.Text="不及格" End Select
---	--	--

(三) Do...Loop 语句

例 1.求 1+2+3+...+n 的和值 (n 值由用户指定), 分别用下面四种循环结构实现。

语句分类	流程图	代码
Do While <条件表达式> 语句 Loop		Dim sum as long,n as integer sum=0 n=1 Do while n<=val(text1.text) sum=sum+n n=n+1 Loop Label10.caption=sum
Do Until <条件> 语句 Loop		Dim sum as long,n as integer sum=0 n=1 Do until n>val(text1.text) sum=sum+n n=n+1 Loop Label10.caption=sum
Do 语句 Loop While<条件>		Dim sum as long,n as integer sum=0 n=1 Do sum=sum+n n=n+1 Loop while n<=val(text1.text) Label10.caption=sum

Do 语句 Loop Until<条件>		Dim sum as long,n as integer sum=0 n=1 Do sum=sum+n n=n+1 Loop until n>val(text1.text) Label10.caption=sum
----------------------------	--	---

(四) For next 语句

语句结构	流程图
For 循环变量=初值 To 终值[Step 步长] 循环体 循环变量=循环变量+步长 [Exit For] Next [循环变量]	<pre> graph TD Start([Start]) --> Init[循环变量=初值] Init --> Decision{循环变量不超过终值} Decision -- 是 --> Body[循环体] Body --> Increment[循环变量增加步长] Increment --> Decision Decision -- 否 --> End([End]) </pre>

举例 1.编程求 [1, 100] 内的奇数和。

```

Private Sub Command1_Click()
    Dim x, s As Long

    s = 0
    ' s为累加求和的变量，没有累加前，s的初始值应该为0

    For x = 1 To 99 Step 2

        s = s + x
        ' 将变量x的值进行累加
    Next

    Print "[1, 100] 内的奇数和为": s
    Print "循环结束后循环变量x的值为": x
End Sub

```

运行结果如下：



举例 2.求素数。

算法说明：

素数（质数）：就是一个大于等于 2 的整数，并且只能被 1 和本身整除，而不能被其他整数整除的数。
判别某数 m 是否是素数的经典算法是：

对于 m, 从 I=2, 3, 4,, m-1 依次判别能否被 I 整除, 只要有一个能整除, m 就不是素数, 否则 m 是素数。

```
Private Function sushu(ByVal n As Long) As Boolean
    Dim i As Long
    For i = 2 To n - 1
        If (n Mod i) = 0 Then Exit For
    Next i
    If I=n then sushu=True
End Function
```

以上判断是否为素数的代码务必识记!

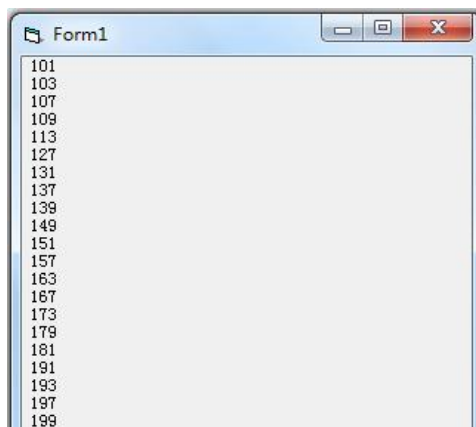
应用举例:

求 100—200 之内素数。

```
Private Function sushu(ByVal n As Long) As Boolean
    Dim i As Long
    For i = 2 To n - 1
        If (n Mod i) = 0 Then Exit For
    Next i
    If i = n Then sushu = True
End Function
```

```
Private Sub Form_Load()
    Form1.Show
    Dim j As Integer
    For j = 100 To 200
        If sushu(j) = True Then
            Print j
        End If
    Next j
End Sub
```

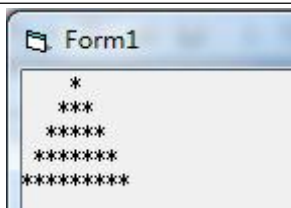
运行结果如下:



举例 3.用 “*” 输出三角形。

```
Private Sub Form_Load()
    Form1.Show
    n = 5
    For i = 1 To n
        Print Space(n - i);
        For j = 1 To 2 * i - 1
            Print "*";
        Next j
        Print
    Next i
End Sub
```

运行结果如下:



实例 4：利用辗转相除法求最大公约数。

辗转相除法，又名欧几里德算法 (Euclidean algorithm) 乃求两个正整数之最大公因子的算法。

算法：

1. $a \div b$ ，令 r 为所得余数 ($0 \leq r < b$)

若 $r = 0$ ，算法结束： b 即为答案。

2. 否则，互换；置 $a \leftarrow b$ ， $b \leftarrow r$ ，并返回第一步

```
Private Sub Command1_Click()
    m = Val (InputBox("第一个数: ")): n = Val (InputBox("第二个数: "))

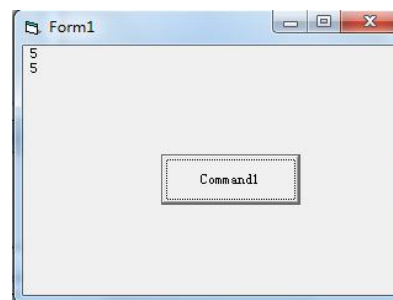
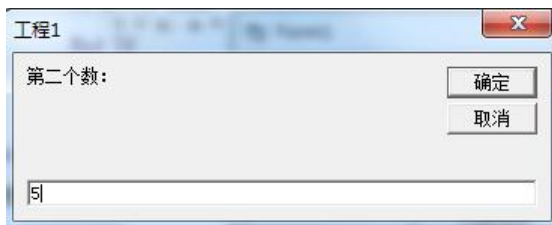
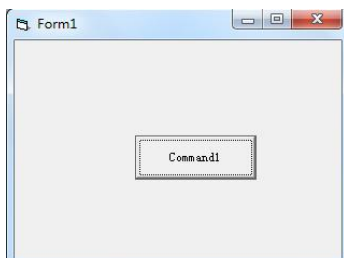
    If n > m Then
        t = m: m = n: n = t
    End If

    r = m - n

    Do While r > 0
        If n < r Then
            t = n: n = r: r = t
        End If

        m = n: n = r: r = m - n
    Loop
    Print n
End Sub
```

运行结果如下：



任意给定一年及其月份判断该年是否为闰年？并根据给出的月份判断该月有多少天？

闰年的条件是：年号能被 4 整除，但是不能被 100 整除或者能被 400 整除。

要求用 Visual Basic 编写程序完成以上问题，其中用 InputBox 输入数据，用 MsgBox 输出结果。

第六节 C 语言

一、程序语言

（一）概念

程序设计语言是人们为了描述计算过程而设计的一种具有语法语义描述的记号。

（二）发展历程

程序设计语言的发展经历了机器语言、汇编语言和高级语言的过程。

1. 机器语言

机器语言是一种指令集的体系，计算机 CPU 可直接识别，由 0、1 序列构成。

2. 汇编语言

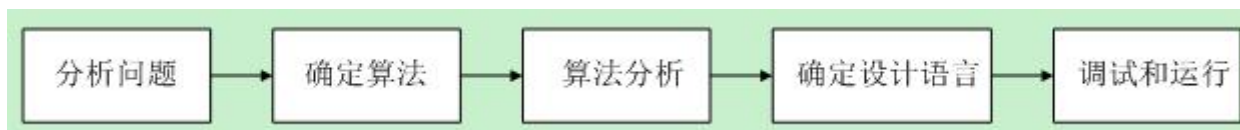
汇编语言是用助记符号描述的指令系统，如 ADD A,B。

3. 高级语言

由于汇编语言依赖于硬件体系，且助记符量大难记，所以发明了高级语言，它是较接近自然语言和数学公式的编程，高级语言并不是特指的某一种具体的语言，而是包括很多编程语言，如目前流行的 Java, C, C++, C#, Pascal, FoxPro, VC, VB 等。分为面向过程的高级语言与面向对象的高级语言。

高级语言编译生成的程序代码一般比用汇编程序语言设计的程序代码要长，执行的速度也慢。所以汇编语言适合编写一些对速度和代码长度要求高的程序和直接控制硬件的程序。高级语言、汇编语言和机器语言都是用于编写计算机程序的语言。

4. 程序设计步骤



（三）C 语言的特点

1. 语言简洁、紧凑、灵活

C 语言一共有 32 个关键字，9 种控制语句，程序书写形式自由，编写的程序短小精练，输入程序的工作量少。

2. 运算符和数据类型丰富

C 语言提供了 34 种运算符，这些运算符的灵活使用可实现其他高级语言中难以实现的运算。数据类型除了包含其他高级语言所具有的数据类型外，还具有现代化语言的数据结构，如数组、指针、结构体、共用体等。

3. 程序设计结构化、模块化

C 语言提供了一整套循环、条件判断和转移语句，实现了对程序逻辑流程的有效控制，有利于结构化程序设计，符合现代编程风格要求。

4. 生成目标代码质量高

C 语言的大多数运算符与一般机器指令相一致，可直接翻译成机器指令，因此，用它编写程序生成

的代码质量高。实践表明，C 语言的代码效率只比汇编语言低 10%~20%。但是 C 语言在描述问题时编程迅速、可读性好、表达能力强等优点是汇编语言无法相比的。

5.可移植性好

C 语言中的输入/输出不依赖于计算机硬件实现。用 C 语言写的程序基本上不做修改就能用于各种型号的计算机和各种操作系统。

（四）C 语言程序的结构特点

1.每个源文件可由一个或多个函数组成。

2.一个源程序不论由多少个文件组成，都有一个且只能有一个 main 函数，即主函数。

3.源程序中可以有预处理命令(include 命令仅为其中的一种)，预处理命令通常应放在源文件或源程序的最前面。

4.每一个说明，每一个语句都必须以分号结尾。

5.标识符、关键字之间必须至少加一个空格以示间隔。若已有明显的间隔符，也可不再加空格来间隔。

（五）简单的 C 语言程序

1.程序

```
/* example1.1    calculate the sum of a and b*/
#include <stdio.h> /* This is the main program */
main()
{   int a,b,sum;
    a=10;
    b=24;
    sum=add(a,b);
    printf("sum= %d\n",sum);
}

/* This function calculates the sum of x and y */
int add(int x,int y)
{   int   z;
    z=x+y;
    return(z);
}
```

2.说明

（1）C 语言的基本单位是函数，函数的基本单位是语句。

（2）其中 main 为函数名，该函数称为主函数，有且仅有一个，（）里面用来存放参数。

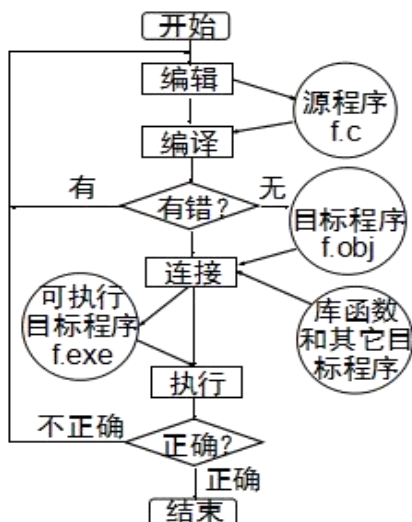
（3）{} 里面的部分称为函数体，也是程序员重点设计的内容。

（4）/* */ 中为注释语句，不被程序执行，可以放在程序的任何位置，但不能嵌套使用。

（5）#include <stdio.h> 为编译预处理命令。

（六）C 语言开发步骤

1.流程图



2.说明

- (1) C 语言源程序名的后缀是.c
- (2) 通过编译后的源程序的后缀为.obj
- (3) 形成的系统可执行程序为.exe

（七）输入输出函数

1.函数 printf()

- (1) 功能：通过标准输出设备输出数据。
- (2) 格式：printf(格式控制,输出表列)。

① “格式控制”是用双引号括起来的部分，由要输出的文字和数据格式说明组成。

②数据格式说明由“%”开头，形式为%<数据输出宽度说明><格式符>。

- (3) 数据宽度说明。

- ①如果实际数据小于宽度，左补空格。
- ②如果实际数据大于宽度，实际位数输出。
- ③如果缺省宽度说明，则按实际宽度输出。

格式符	功能
d	以带符号的十进制形式输出整数（整数不输出正号）
o	以不带符号的八进制形式输出整数
x	以不带符号的十六进制形式输出整数
u	以不带符号的十进制形式输出整数
c	以字符形式输出一个字符
s	输出一个或多个字符
f	以小数形式输出单、双精度数，默认输出 6 位小数
e	以标准指数形式输出单、双精度数，数字部分小数位数为 6 位

举例：

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    printf("a=%4d,b=%6.2f,c=%c,d=%s",12,3.456,'A',"hello" );
}
```

运行结果如下：



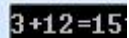
```
a= 12,b= 3.46,c=A,d=hello
```

(4) 输出表列可以是变量、表达式或者是数值。

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int a=3;

    printf("%d+%d=%d",a,12,a+12);
}
```

运行结果如下图所示：



```
3+12=15
```

2.函数 scanf()

(1) 功能：通过标准输入设备输入数据。

(2) 格式：scanf(格式控制,地址表列)。

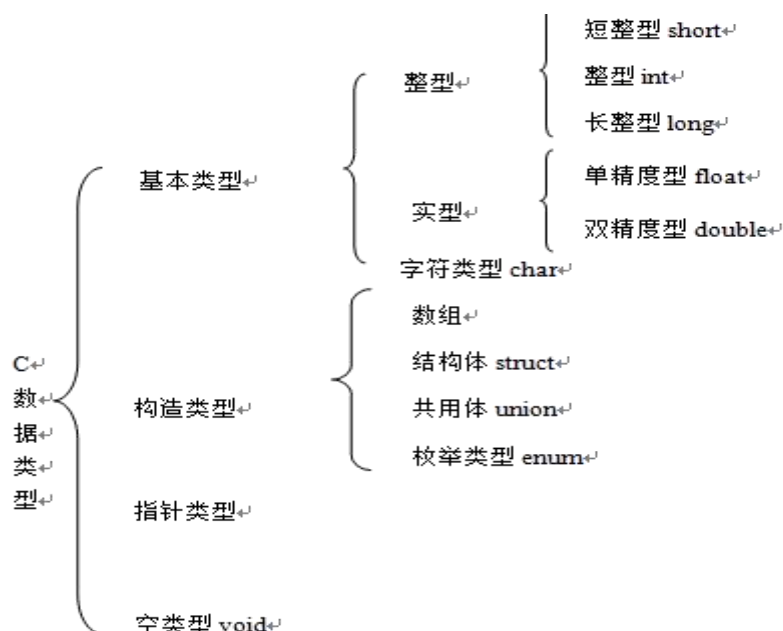
(3) 使用规则：

①基本上 printf()函数相同。

②若格式符中无其他字符间隔，输入时用空格、回车或跳格键 Tab 为分隔。

③若格式中有特定的分割符，输入时也必须一致。

(八) C 语言的数据类型



(九) 标识符

1. 定义

用来标识变量、常量、函数等的字符序列。

2. 分类

(1) 关键字

关键字就是已被 C 语言本身使用，不能作其他用途使用的标识符。

(2) 预定义标识符

系统预先定义的标识符，如系统类库名、常量名、函数名。预定义标识符具有见字明义的特点，如 printf、scanf、sin 等。预定义标识符可作为用户标识符，但会失去系统规定的原意。

(3) 用户标识符

用户根据需要定义的标识符。用来给变量、函数、数组等命名。只能由字母、数字、下划线组成；第一个字母必须是字母或下划线。

(十) 常量

1. 整型常量

(1) 十进制整数：由数字 0~9 和正负号表示。如 123，-456，0。

(2) 八进制整数：由数字 0 开头，后跟数字 0~7 表示。如 0123，011。

(3) 十六进制整数：由 0x 开头，后跟 0~9，a~f，A~F 表示。如 0x123，0Xff。

2. 实型常量

(1) 十进制数形式：如 0.12，.15，0.0，123。

(2) 指数形式：(e 前 e 后必有数且 e 后为整数) 如 12.3e3，123E2，1.23e4，e-5。

3. 字符常量

(1) 用单引号括起来的单个普通字符或转义字符。

(2) 每一个字符常量都有一个与其对应的 ASCII 码值。

(3) 转义字符。

转义字符	含义	转义字符	含义
\n	换行	\t	水平制表
\v	垂直制表	\b	退格
\r	回车	\f	换页
\a	响铃	\\	反斜线
\'	单引号	\''	双引号
\ddd	3 位 8 进制数代表的字符	\xhh	2 位 16 进制数代表的字符

4.字符串常量

- (1) 定义：用双引号("")括起来的字符序列。
- (2) 存储：每个字符串尾自动加一个'\0'作为字符串结束标志。

5.符号常量

- (1) 用标识符表示的常量称为符号常量。习惯上，符号常量名用大写，变量用小写。

```
#include <stdio.h>
#include <stdlib.h>
#define PRICE 30
int main()
{
    int num,total;

    num=10;

    total=num*PRICE;
    printf("total=%d",total);
}
```

运行结果如下图所示：

total=300

- (2) 符号常量的值在其所在的函数内不变。
- (3) 使用符号常量可以使程序清晰，修改容易。

(十一) 变量

1.基础知识

- (1) 概念：其值可以改变的量，一般格式为：数据类型 变量 1[变量 2，…，变量 n]。
- (2) 变量初始化：定义时赋初值。
- (3) 使用规则：先定义，后使用。
- (4) 分类：整型变量(int)、实型变量(float 与 double)、字符型变量(char)。

2.存储范围

类型	符号	关键字	所占位数	数的表示范围
整型	有	(signed)int	16	-32768-32767
		(signed)short	16	-32768-32767

	无	(signed) long	32	-2147483648-2147483647
		unsigned int	16	0-65535
		unsigned short	16	0-65535
		unsigned long	32	0-4294967295
实型	有	float	32	3.4e-38-3.4e38
	有	double	64	1.7e-308-1.7e308
字符型	有	char	8	-128-127
	无	unsigned char	8	0-255

3. 字符数据的存储方式

- (1) 字符数据是以 ASCII 码存储的，它的存储形式与整数的存储形式相类似 `c1='a'` 和 `c1=97`。
- (2) 没有字符串变量，字符串变量用数组存储。
- (3) 几个主要的字符的 ASCII 码，0 为 48，A 为 65，a 为 97。

4. 变量赋初值

- (1) 初始化

`int a=3; /*指定 a 为整型变量，初值为 3*/`

`float f=3.56;`

`char c='a';`

- (2) 其他方式

给被定义变量部分赋初值：`int a,b,c=5;`

对几个变量赋以同一个值：错误：`int a=b=c=3;` 正确：`int a=3,b=3,c=3;`

(十二) 运算符

C 运 算 符	算术运算符：(+ - * / % ++ --) ↵
	关系运算符：(< <= == > >= !=) ↵
	逻辑运算符：(! &&) ↵
	位运算符：(<< >> ~ ^ &) ↵
	赋值运算符：(= 及其扩展) ↵
	条件运算符：(?:) ↵
	逗号运算符：(,) ↵
	指针运算符：(* &) ↵
	求字节数：(sizeof) ↵
	强制类型转换：(类型) ↵
	分量运算符：(. ->) ↵
	下标运算符：([]) ↵
	其它：(() -) ↵

(十三) 表达式

1. 自增、自减运算符

- (1) `++i, --i` (在使用 i 之前, 先使 i 的值加 (减) 1)。

- (2) `i++`, `i--` (在使用 `i` 之后, 先使 `i` 的值加(减)1)。
- (3) 自增、自减运算符只能用于变量, 而不能用于常量或表达式。
- (4) `++`和`--`的结合方向是“自右至左”。

2. 赋值表达式

- (1) 概念: 由赋值运算符将一个变量和一个表达式连接起来的式子。
- (2) 形式: `<变量><赋值运算符><表达式>`
- (3) 举例分析

① `a=5+(c=6)`

② `a=(b=4)+(c=6)`

③ `a=(b=10)/(c=2)`

④ `a` 初值为 6; `a+=a-=a*a`

3. 逗号运算符

- (1) 形式: 表达式 1, 表达式 2, 表达式 3。
- (2) 规则: 从左至右分别计算表达式 1、2、3, 那么表达式 3 的值是该逗号表达式的值。
- (3) 举例分析

① `a=3*5, a*4` `a` 的值为 15, 表达式的值为 60

② `x=(a=3, 6*3)` 赋值表达式

③ `x=a=3, 6*a` 逗号表达式

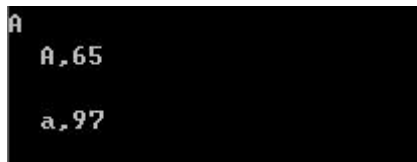
举例: 从键盘输入一个大写字母, 要求改用小写字母输出。

```
main()
{
    char c1, c2;

    c1=getchar();

    printf(" %c, %d\n\n", c1, c1);
    c2=c1+32;
    printf(" %c, %d\n\n", c2, c2);
}
```

运行结果如下图所示:



```
A
A, 65

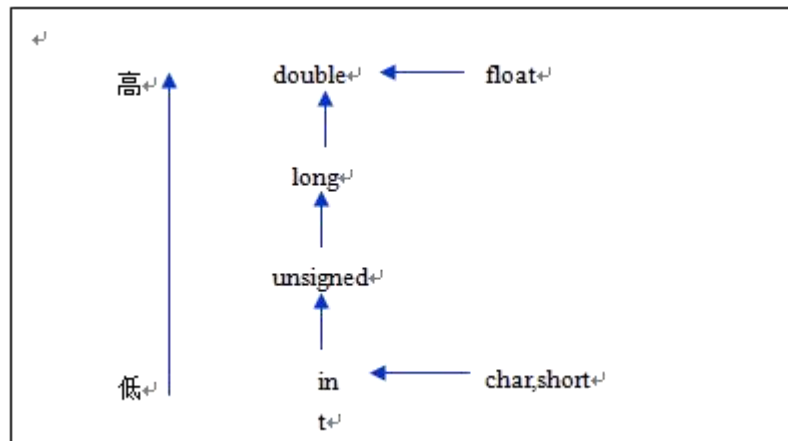
a, 97
```

4. 不同类型数据间的转换

- (1) 隐式转换

- ① 运算转换——不同类型数据混合运算时。
- ② 赋值转换——把一个值赋给与其类型不同的变量时。
- ③ 输出转换——输出时转换成指定的输出格式。
- ④ 函数调用转换——实参与形参类型不一致时转换。

⑤运算转换规则：不同类型数据运算时先自动转换成同一类型。



(2) 强制类型转换

①概念：利用强制类型转换运算符将一个表达式转换成所需类型。

②形式：（类型名）（表达式）。

③得到的类型是中间结果，并不改变变量原先的类型。

④强制转换的是舍去法而不是四舍五入。

⑤举例分析

(int)(x+y)

(int)x+y

(double)(3/2)

(int)3.6

5. 数学函数

在使用数学函数的时候需要加上：include <math.h>或#include "math"。

函数名	作用	举例
pow(a,b)	计算 a 的 b 次方	Pow(2,3)的值为： 8
sqrt(a)	计算根号 a	Sqrt(100)的值为： 10
log(x)	计算以 e 为底 x 的对数	Log(100)的值为： 4
log10(x)	计算以 10 为底 x 的对数	Log10(100)的值为： 2
abs(x)	求整数 x 的绝对值	Abs(-5)的值为： 5
floor(x)	返回的是小于或等于 x 的最大整数	floor(2.5)=2 floor(-2.5)=-3
ceil	返回的是大于 x 的最小整数	ceil(2.5) = 3 ceil(-2.5) = -2

二、选择结构

（一）关系运算符和关系表达式

1. 关系运算符和优先次序图

<	(小于)	} 优先级相同 (高)
<=	(小于或等于)	
>	(大于)	
>=	(大于或等于)	
==	(等于)	} 优先级相 (低)
!=	(不等于)	

(1) 关系运算符的优先级低于算术运算符。

(2) 关系运算符的优先级高于赋值运算符。

2. 关系表达式

(1) 用关系运算符将两个表达式连接起来的式子，称关系表达式。

(2) 表达式包括算术表达式、关系表达式，逻辑表达式，赋值表达式，字符表达式。

(3) 关系表达式的值是一个逻辑值，即“真”或“假”，C语言中1代表真，0代表假。

例：a>b, a+b>b+c, (a=3)>(b=5), 'a'<'b', (a>b)>(b<c)

3. 逻辑运算符和逻辑表达式

(1) 逻辑运算符

&&：（逻辑与）相当于其他语言中的 AND

||：（逻辑或）相当于其他语言中的 OR

!：（逻辑非）相当于其他语言中的 NOT

举例分析：

① **a&&b**：若 a, b 为真，则 a&&b 为真。

② **a||b**：若 a, b 之一为真，则 a||b 为真。

③ **!a**：若 a 为真，则 !a 为假。

优先次序：

① **!** (非) -> **&&()** -> **||()**

② “&&” 和 “||” 低于关系运算符，“!” 高于算术运算符

(2) 逻辑表达式

用逻辑运算符将关系表达式或逻辑量连接起来的式子就是逻辑表达式。

任何非零的数值被认作“真”，逻辑表达式的值应该是一个逻辑“真”或“假”。

举例分析：

5>3&&8<4-!0，该表达式逻辑值为假。

(3) 逻辑表达式运算规则

在逻辑表达式的求解中，并不是所有的逻辑运算符都要被执行。

a&&b&&c：

① 只有 a 为真时，才执行第一个&&号后的 b 表达式。

② 只有 a 和 b 都为真时，才执行 C 表达式。

a||b||c：

① 只要 a 为真，b、c 表达式不被执行。

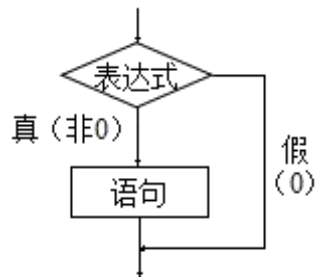
② 若 a 为假，判断 b 与 c。

③若 a 和 b 都为假才判断 c。

(二) IF 语句

1.if (表达式) 语句

if(x>y) printf("%d",x);



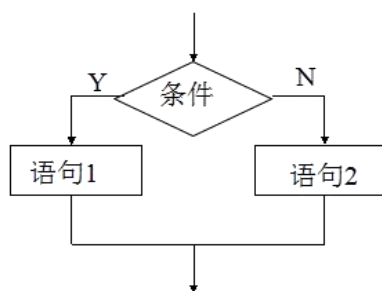
2.if(表达式)语句 1

else 语句 2

例子:

if(x>y) printf("%d",x);

else printf("%d",y);



3.if (表达式 1) 语句 1

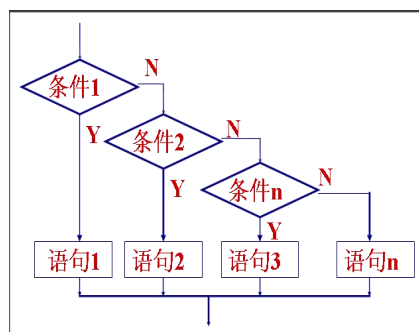
else if(表达式 2)语句 2

else if(表达式 3)语句 3

.....

else if(表达式 n)语句 n-1

else 语句 n



例题:

```

main()
{
    int a=1,b=2,c=3,d=4,r=0;

    if(a!=1); else r=1;
    if(b==2)    r+=2;
    else;       if(c!=3) r+=3;
    else;       if(d==4) r+=4;
    printf("%d\n",r);
}

```

运行结果如下图所示：

?

（三）条件运算符

1.格式

表达式 1?表达式 2:表达式 3

2.功能

判断表达式 1 的值，若为真执行表达式 2 且表达式 2 为该条件表达式的值，若为假，执行表达式 3，表达式 3 为该条件表达式的值。

3.特点

代替 if 语句，节省语句空间。

$k=(a<b)?((a<c)?a:c):((b<c)?b:c)$

if(a<b)

if(a<c)

k=a;

else k=c;

else

if(b<c) k=b;

else k=c;

举例：

输入一个字符，判别它是否大写字母，如果是，将它转换成小写字母；如果不是，不转换，然后输出最后得到的字符。

```

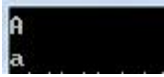
main()
{
    char ch;

    scanf("%c",& ch);

    ch=(ch>='A' && ch<='Z')?(ch+32):ch;
    printf("%c\n",ch);
}

```

运行结果如下所示：



(四) SWITCH 语句

switch 语句的格式:

```
switch(表达式)
{
    case 常量表达式 1:语句 1;break;
    case 常量表达式 2:语句 2;break;
    .....
    .....
    case 常量表达式 n:语句 n;break;
    default:语句 n+1;
}
```

例如: 要求按照考试成绩的等级输出百分制分数段, 用 switch 语句实现

```
switch(grade)
{
    case 'A':printf("85-100\n");break;
    case 'B':printf("70-84\n");break;
    case 'C':printf("60-69\n");break;
    case 'D':printf("<60\n");break;
    default:printf("error\n");
}
```

三、循环结构

(一) While 循环

1.while 的一般形式

while(表达式)

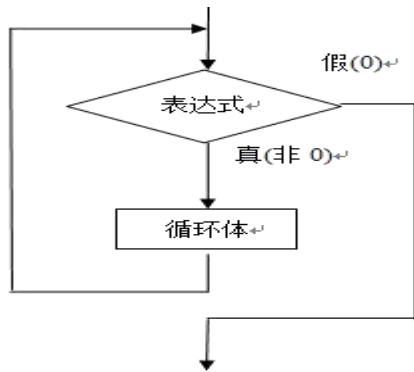
```
{
    循环体语句
}
```

说明:

- (1) while 是 C 语言的关键字, 必须小写。
- (2) 表达式是任意合法的表达式 (常量、算术、关系、逻辑、赋值表达式)。
- (3) 循环体语句只能是一条语句或是一个复合语句。
- (4) 表达式不能省略, 循环体语句省略时表示不做任何的操作。

2.执行过程

先判断表达式, 后执行语句



(二) Do-While 循环

1.do-while 语句的一般形式

do

{

 循环体语句

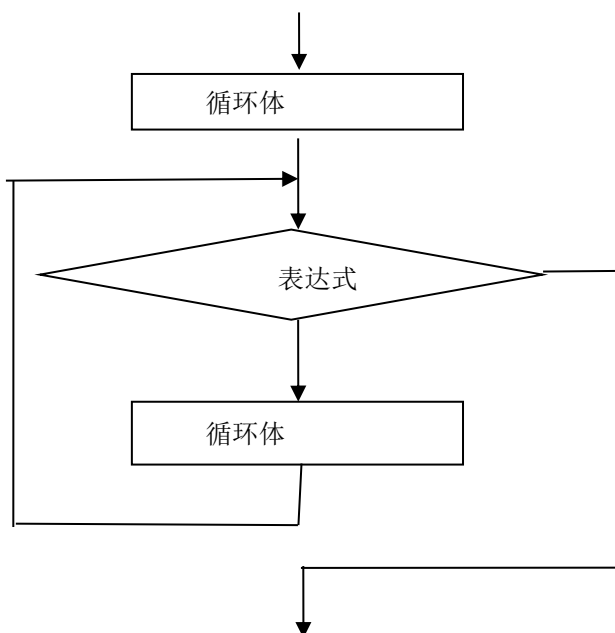
}while(表达式);

说明:

- (1) do 是关键字和 while 搭配使用，do-while 语句至少执行一次循环体。
- (2) 表达式可以是任意合法的表达式，循环体语句是一条语句或一条复合语句。
- (3) 表达式不能省略，循环体语句省略时表示不做任何的操作。
- (4) while 后面的分号不能省略。

2.执行过程

先执行循环体，后判断表达式。



例 1.求 1 到 100 之间的奇数之和，偶数之积。


```

#include <stdio.h>
#include <stdlib.h>
main()
{
    int a,b,num1,num2,temp;

    printf("please input two numbers:\n");
    scanf("%d,%d",&num1,&num2);
    if(num1<num2)
    {
        temp=num1;
        num1=num2;
        num2=temp;
    }
    a=num1;b=num2;
    while(b!=0)/*利用辗除法，直到b为0为止*/
    {
        temp=a%b;
        a=b;
        b=temp;
    }

    printf("gongyueshu:%d\n",a);
    printf("gongbeishu:%d\n",num1*num2/a);
}

```

程序运行结果如下所示:

```

please input two numbers:
2,12
gongyueshu:2
gongbeishu:12

```

(三) For 循环

1.for 语句的一般形式

for(表达式 1;表达式 2;表达式 3)

```

{
    循环体语句
}

```

说明:

for 是 C 语言的关键字。

for 里面的三个表达式必须使用分号隔开。

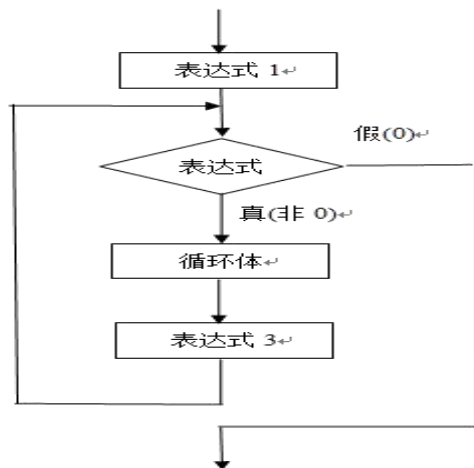
(1) 表达式 1 表示赋初值。

(2) 表达式 2 表示终值。

(3) 表达式 3 表示步长。

循环体语句只能是一条语句或是一个复合语句。

2.执行过程



注意:

- (1) 表达式 1 可省, 但分号不能省。
- (2) 若表达式 2 省略, 循环条件永远为真。
- (3) 表达式 3 也可省略, 但应设法保证循环正常结束。
- (4) 可只给循环条件, 这时与 while 语句等同。
- (5) 三个表达式都可省。

for (;;) 相当于 while (1)

例 4. 打印出所有的“水仙花数”, 所谓“水仙花数”是指一个三位数, 其各位数字立方和等于该数本身。

```

main()
{
    int i,j,k,n;

    printf("'water flower'number is:\n");
    for(n=100;n<1000;n++)
    {
        i=n/100; /*分解出百位*/
        j=n%100/10; /*分解出十位*/
        k=n%10; /*分解出个位*/

        if(i*100+j*10+k==i*i*i+j*j*j+k*k*k)
        {
            printf("%-5d\n",n);
        }
    }
    printf("\n");
}
  
```

程序运行结果如下所示:


```
'water flower'number is:
153
370
371
407
```

（四）循环的嵌套

1.概念

一个循环体内又包含另一个完整的循环结构。

2.说明

（1）嵌套可以是多层的。

（2）一个循环体必须完整嵌套在另一个循环体内，不能出现交叉。

（3）三种循环可以互相嵌套。

例 5.两个乒乓球队进行比赛，各出三人。甲队为 a, b, c 三人，乙队为 x, y, z 三人。已抽签决定比赛名单。有人向队员打听比赛的名单。a 说他不和 x 比，c 说他不和 x, z 比，请编程找出三队赛手的名单。

```
main()
{
    char i,j,k;
    /*i是a的对手, j是b的对手, k是c的对手*/
    for(i='x';i<='z';i++)
        for(j='x';j<='z';j++)
            for(k='x';k<='z';k++)
                if(i!=k && j!=k && i!=j && i!='x'&& k!='x'&& k!='z')
                    printf("order is a--%c\tb--%c\tc--%c\n",i,j,k);
}
```

程序运行结果如下所示：

```
order is a--z   b--x   c--y
```

（五）break 和 continue 语句

1.break 语句

（1）形式：break;

（2）作用：跳出循环体，终止循环，执行后面的语句；

（3）注意：break 语句只能用于循环语句和 switch 语句。

例 6.从键盘上任意输入一个正整数，判断其是否为素数。

```

main()
{
    int m,i;

    scanf("%d",&m);
    for(i=2;i<=m-1;i++)
        if(m%i==0) break;
        if(i>m-1) printf("%d is a prime",m);
        else printf("%d is not a prime",m);
}

```

程序运行结果如下所示:

```

11
11 is a prime
12
12 is not a prime

```

2.continue 语句

- (1) 形式: continue;
- (2) 作用: 结束本次循环, 执行下一次是否执行循环的判定。

例 7.把 100-200 之间的不能被 3 整除的数输出。(要求使用 continue)。

```

main()
{
    int i;
    int j;
    for(i=100;i<=130;i++)
    {
        if (i%3==0) continue;
        printf("  %d",i);
    }
}

```

程序运行结果如下图所示:

```

100 101 103 104 106 107 109 110 112 113 115 116 118 119 121 122
124 125 127 128 130请按任意键继续. . .

```

3.break 和 continue 语句的区别

- (1) continue 语句只结束本次循环, 不终止整个循环的执行;
- (2) break 语句是终止整个循环的执行, 不再进行条件判断。

四、数组与字符串

(一) 一维数组的定义和引用

1.定义

类型说明符 数组名 [整型常量表达式];

2.说明

- (1) 数组名的定名规则和变量名相同, 遵循标识符定名规则。
- (2) 常量表达式表示元素的个数, 即数组长度。

(3) 常量表达式中包括常量和符号常量，不能包含变量，即定义时必须确定数组的大小。

3. 引用形式

数组名[下标] 下标表达式的下限为 0，上限为常量表达式—1

4. 赋值

(1) 单个赋值

```
int a[3]; a[0]=10;
```

(2) 整体赋值

①一维数组赋值之数组长度等于赋值个数的情况

```
int a[3]={10,20,30};
```

②一维数组赋值之数组长度大于赋值个数的情况

```
int a[5]={10,20,30};
```

③一维数组赋值之数组长度小于赋值个数的情况

```
int a[5]={10,20,30,40,50,60}; 此时显示错误
```

④一维数组赋值之特殊情况

```
int a[5]={};表示给数组赋值不确定值，而不是 0
```

⑤一维数组赋值之不指明数组长度

```
int a[]={1,2,3,4,5}; /*数组长度为 4*/
```

```
int a[]={0}; /*数组长度为 1*/
```

```
int a[]={}; /*数组长度为 0*/
```

5. 一维数组的经典应用

举例 1：冒泡排序算的思想。

①比较相邻的元素，如果第一个比第二个大，交换。

②对每一对相邻元素作同样的工作。

③针对所有的元素重复以上的步骤，直到没有任何一对数字需要比较。

```

main()
{
    int i, j, tmp, number[10] = {95,45, 15, 78, 84, 51,24, 12, 34, 50};
    for (i = 0; i < 10; i++)
    {
        for (j = 10 - 1; j > i; j--)
        {
            if (number[j] < number[j-1])
            {
                tmp = number[j-1];
                number[j-1] = number[j];
                number[j] = tmp;
            }
        }
    }
    for (i = 0; i < 10; i++)
    {
        printf("%d ", number[i]);
        printf("\n");
    }
}

```

程序运行结果如下所示:

```
12 15 24 34 45 50 51 78 84 95
```

举例 2: 使用 for 循环为一个数组赋值, 并将数组倒序输出。

```

main()
{
    int i,a[10];

    for(i=0;i<=9;i++)
        a[i]=i;
    for(i=9;i>=0;i--)
        printf("%d ",a[i]);
}

```

程序运行结果如下:

```
9 8 7 6 5 4 3 2 1 0
```

举例 3: 输入 10 个数字并输出最大值。

```

main()
{
    int i,max,a[10];

    printf("input 10 numbers:\n");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    max=a[0];
    for(i=1;i<10;i++)
        if(a[i]>max) max=a[i];
    printf("maximum=%d\n",max);
}

```

程序运行结果如下所示：

```
input 10 numbers:
1 23 43 576 789 123 32 45 65 12
maxnum=789
```

本例程序中第一个 for 语句逐个输入 10 个数到数组 a 中。然后把 a[0]送入 max 中。在第二个 for 语句中，从 a[1]到 a[9]逐个与 max 中的内容比较，若比 max 的值大，则把该下标变量送入 max 中，因此 max 总是在已比较过的下标变量中为最大者。比较结束，输出 max 的值。

（二）字符串的概念和操作

1.定义

以双引号括起来的，由若干字符组成的序列。

2.字符串结构："字符+\0"

'\0'是字符串结束的标志，是一个转义字符，ASCII 为 0，也称为“空值”。

3.字符串长度

第一个'\0'以前的字符个数。

4.字符串的总长度

所有字符个数和所有'\0'个数。

5.用字符串常量初始化

(1) char c[]={"I am happy"};

(2) char c[]= "I am happy";

6.二维字符数组存放字符串数组

(1) 二维数组的行，使用一行存放一个字符串。

(2) 二维数组的列，限制了字符串的长度。

(3) 字符串的长度小于数组的列，那么使用'\0'进行补充剩余的个数。

7.字符数组的输入输出

(1) 逐个字符输入输出。用格式符“%c”输入或输出一个字符。

(2) 将整个字符串一次输入或输出。用“%s”格式符，意思是输出字符串。

举例 1：输入一串字符，求这个字符串的长度，并输出这个字符串。

```
main()
{
    char ch[20];
    int i,k=0;

    scanf("%s",ch);
    for(i=0;ch[i]!='\0';i++)
        k++;
    printf("%d\n",k);
    for(i=0;ch[i]!='\0';i++);
    printf("%c\n",ch[i]);
}
```

程序运行结果如下所示：

```
eqrwefasdf
10
```

举例 2：输入一串字符，将这个字符串复制到另一个字符串中并输出。

```
main()
{
    char c1[20],c2[20];
    int i;

    scanf("%s",c1);
    for(i=0;c1[i]!='\0';i++)
        c2[i]=c1[i];
    c2[i]='\0';
    printf("%s\n",c2);
}
```

程序运行结果如下所示：

```
qlekj lk lak jdf
qlekj
```

举例 3：输入一行字符，分别统计出其中英文字母、空格、数字和其他字符的个数。

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int i=0,zm,sz,kg,other;
    char c[20],a;
    zm=sz=kg=other=0;
    scanf("%s",c);
    while(c[i]!='\0')
    {
        if (c[i]>='a' && c[i]<='z' || c[i]>='A' && c[i]<='Z')
            zm++;
        else if (c[i]>='0' && c[i]<='9')
            sz++;
        else if (c[i]==' ')
            kg++;
        else
            other++;
        i++;
    }
    printf("%d,%d,%d,%d ",zm,sz,kg,other);
}
```

程序运行结果如下所示：

```
123asd#@
3,3,0,2
```

（三）字符串操作常用函数

1.puts(字符数组名)

（1）作用

将一个字符串（以'\0'结束的字符序列）输出到终端。

（2）功能

设 str 是一个字符数组名，且其数组已被初始化为“China”，若 puts(str);则输出 China 到终端。

2.gets(字符数组名)

（1）作用

从终端输入一个字符串到字符数组中，并得到一个函数值。该函数值是字符数组的起始地址。

（2）功能

gets(str)从键盘输入：offcn（回车）将输入的字符串“offcn”送给字符数组 str。

3.字符串连接函数

（1）strcat 格式

strcat (字符数组名 1,字符数组名 2)

（2）功能

删除字符数组 1 字符串“\0”把字符数组 2 的字符串链接到其后。

例题：

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    static char st1[30]="My school is";
    int st2[10];

    printf("input your name:\n");

    gets(st2);
    strcat(st1,st2);
    puts(st1);
}
```

程序运行结果如下：



```
input your name:
zhong gong jiao yu
My school iszhong gong jiao yu
```

注意：字符数组 1 足够的长度，否则不能全部装入被连接的字符串。

4.strcpy(字符数组 1,字符串 2)

（1）格式

strcpy (字符数组名 1, 字符数组名 2)

（2）功能

把字符数组 2 中的字符串拷贝到字符数组 1 中。

（3）字符数组 1 足够的长度，否则不能全部装入被连接的字符串。

5.字符串比较函数 strcmp

（1）格式

`strcmp`(字符数组名 1,字符数组名 2)

(2) 功能

按照 ASCII 码顺序比较两个数组中的字符串，并由函数返回值返回比较结果。

①字符串 1==字符串 2，函数值为 0

②字符串 1>字符串 2，函数值为一正整数

③字符串 1<字符串 2，函数值为一负整数

(3) 本函数也可以用于比较两个字符串常量，或比较数组和字符串常量。

6.`strlen`(字符数组)

测试字符串长度的函数，函数的值为字符串的实际长度，不包括'\0'在内。

7.`strlwr`(字符串)

将字符串中大写字母换成小写字母。

8.`strupr`(字符串)

作用：将字符串中小写字母换成大写字母。

第七节 Java 简介

一、Java 主要特性

Java 语言的语法与 C 语言和 C++ 语言很接近,使得大多数程序员很容易学习和使用。另一方面,Java 丢弃了 C++ 中很少使用的、很难理解的、令人迷惑的那些特性,如操作符重载、多继承、自动的强制类型转换。特别地,Java 语言不使用指针,而是引用,并提供了自动的废料收集,使得程序员不必为内存管理而担忧。

二、Java 基础语法

一个 Java 程序可以认为是一系列对象的集合,而这些对象通过调用彼此的方法来协同工作。下面简要介绍下类、对象、方法和实例变量的概念。

对象:对象是类的一个实例,有状态和行为。例如,一条狗是一个对象,它的状态有:颜色、名字、品种;行为有:摇尾巴、叫、吃等。

类:类是一个模板,它描述一类对象的行为和状态。

方法:方法就是行为,一个类可以有很多方法。逻辑运算、数据修改以及所有动作都是在方法中完成的。

实例变量:每个对象都有独特的实例变量,对象的状态由这些实例变量的值决定。

(一) 第一个 Java 程序

下面看一个简单的 Java 程序,它将打印字符串 Hello World

```
public class MyFirstJavaProgram {  
/*第一个 Java 程序,它将打印字符串 Hello World*/  
public static void main(String []args)  
{  
System.out.println("Hello World");//打印 Hello World  
}  
}
```

(二) 基本语法

编写 Java 程序时,应注意以下几点:

大小写敏感:Java 是大小写敏感的,这就意味着标识符 Hello 与 hello 是不同的。

类名:对于所有的类来说,类名的首字母应该大写。如果类名由若干单词组成,那么每个单词的首字母应该大写,例如 MyFirstJavaClass。

方法名:所有的方法名都应该以小写字母开头。如果方法名含有若干单词,则后面的每个单词首字母大写。

源文件名:源文件名必须和类名相同。当保存文件的时候,你应该使用类名作为文件名保存(切记 Java 是大小写敏感的),文件名的后缀为.java。(如果文件名和类名不相同则会导致编译错误)。

主方法入口:所有的 Java 程序由 public static void main(String args[])方法开始执行。

(三) Java 标识符

Java 所有的组成部分都需要名字。类名、变量名以及方法名都被称为标识符。

关于 Java 标识符，有以下几点需要注意：

- (1) 所有的标识符都应该以字母 (A-Z 或者 a-z) ,美元符 (\$)、或者下划线 (_) 开始。
- (2) 首字符之后可以是任何字符的组合。
- (3) 关键字不能用作标识符。
- (4) 标识符是大小写敏感的。

合法标识符举例：age、\$salary、_value、__1_value

非法标识符举例：123abc、-salary

(四) Java 修饰符

像其他语言一样，Java 可以使用修饰符来修饰类中方法和属性。主要有两类修饰符：

可访问修饰符：default，public，protected，private

不可访问修饰符：final，abstract，strictfp

(五) Java 变量

Java 中主要有如下几种类型的变量：

- (1) 局部变量；
- (2) 类变量（静态变量）；
- (3) 成员变量（非静态变量）。

(六) Java 数组

数组是储存在堆上的对象，可以保存多个同类型变量。在后面的章节中，我们将会学到如何声明、构造以及初始化一个数组。

(七) Java 常用关键字

下面列出了 Java 保留字。这些保留字不能用于常量、变量、和任何标识符的名称。

关键字	描述
abstract	抽象方法，抽象类的修饰符
boolean	布尔数据类型
byte	8-bit 有符号数据类型
case	switch 语句的一个条件
catch	和 try 搭配捕捉异常信息
char	16-bit Unicode 字符数据类型
class	定义类
continue	不执行循环体剩余部分
default	switch 语句中的默认分支
do	循环语句，循环体至少会执行一次
double	64-bit 双精度浮点数
enum	枚举类型
extends	表示一个类是另一个类的子类
final	表示一个值在初始化之后就不能再改变了，表示方法不能被重写，或者一个类不能有子类

finally	为了完成执行的代码而设计的，主要是为了程序的健壮性和完整性，无论有没有异常发生都执行代码。
float	32-bit 单精度浮点数
for	for 循环语句
goto	未使用
if	条件语句
implements	表示一个类实现了接口
import	导入类
int	32 位整型数
interface	接口，一种抽象的类型，仅有方法和常量的定义
long	64 位整型数
new	分配新的类实例
package	一系列相关类组成一个包
private	表示私有字段，或者方法等，只能从类内部访问
protected	表示字段只能通过类或者其子类访问 子类或者在同一个包内的其他类
public	表示共有属性或者方法
return	方法返回值
short	16 位数字
static	表示在类级别定义，所有实例共享的
super	表示基类
switch	选择语句
synchronized	表示同一时间只能由一个线程访问的代码块
this	表示调用当前实例或者调用另一个构造函数
throw	抛出异常
try	表示代码块要做异常处理或者和 finally 配合表示是否抛出异常都执行 finally 中的代码
void	标记方法不返回任何值
while	while 循环

三、Java 对象和类

（一）Java 中的类

类：类是一个模板，它描述一类对象的行为和状态。

通过下面一个简单的类来理解下 Java 中类的定义：

```
public class Dog{
    String breed;
    int age;
```

```

    String color;
    void barking(){
    }
    void hungry(){
    }
    void sleeping(){
    }
}

```

一个类可以包含以下类型变量。

局部变量：在方法、构造方法或者语句块中定义的变量被称为局部变量。变量声明和初始化都是在方法中，方法结束后，变量就会自动销毁。

成员变量：成员变量是定义在类中，方法体之外的变量。这种变量在创建对象的时候实例化。成员变量可以被类中方法、构造方法和特定类的语句块访问。

类变量：类变量也声明在类中，方法体之外，但必须声明为 `static` 类型。

一个类可以拥有多个方法，在上面的例子中：`barking()`、`hungry()`和 `sleeping()`都是 `Dog` 类的方法。

（二）Java 中的对象

对象：对象是类的一个实例，有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。

对比现实对象和软件对象，它们之间十分相似。

软件对象也有状态和行为。软件对象的状态就是属性，行为通过方法体现。

在软件开发中，方法操作对象内部状态的改变，对象的相互调用也是通过方法来完成。

（三）构造方法

每个类都有构造方法。如果没有显式地为类定义构造方法，`Java` 编译器将会为该提供一个默认构造方法。

在创建一个对象的时候，至少要调用一个构造方法。构造方法的名称必须与类同名，一个类可以有多个构造方法。

下面是一个构造方法示例：

```

public class Puppy{
    public Puppy(){
    }
    public Puppy(String name){
// 这个构造器仅有一个参数：name
    }
}

```

（四）创建对象

对象是根据类创建的。在 `Java` 中，使用关键字 `new` 来创建一个新的对象。创建对象需要以下三步。

声明：声明一个对象，包括对象名称和对象类型。

实例化：使用关键字 `new` 来创建一个对象。

初始化：使用 `new` 创建对象时，会调用构造方法初始化对象。

下面是一个创建对象的例子：

```

public class Puppy{
    public Puppy(String name){
        //这个构造器仅有一个参数： name
        System.out.println("Passed Name is :" + name );
    }
    public static void main(String []args){
//下面的语句将创建一个 Puppy 对象
        Puppy myPuppy = new Puppy( "tommy");
    }
}

```

编译并运行上面的程序，会打印出下面的结果：

Passed Name is:tommy

（五）访问实例变量和方法

通过已创建的对象来访问成员变量和成员方法，如下所示：

```

/*实例化对象*/
ObjectReference = new Constructor();
/*访问其中的变量*/
ObjectReference.variableName;
/*访问类中的方法*/
ObjectReference.MethodName();

```

实例：下面的例子展示如何访问实例变量和调用成员方法。

```

public class Puppy{
int puppyAge;
public Puppy(String name){
    //这个构造器仅有一个参数： name
    System.out.println("Passed Name is:" + name );
}
    public void setAge( int age ){
        puppyAge = age;
    }
    public int getAge( ){
        System.out.println("Puppy's age is " + puppyAge);
        return puppyAge;
    }
    public static void main(String []args){
/*创建对象*/
        Puppy myPuppy = new Puppy( "tommy" );
/*通过方法来设定 age*/
        myPuppy.setAge( 2 );
/*调用另一个方法获取 age*/
        myPuppy.getAge( );
/*你也可以像下面这样访问成员变量*/

```

```
        System.out.println("Variable Value:" + myPuppy.puppyAge );  
    }  
}
```

编译并运行上面的程序，产生如下结果：

Passed Name is:tommy

Puppy's age is:2

Variable Value:2

（六）Import 语句

在 Java 中，如果给出一个完整的限定名，包括包名、类名，那么 Java 编译器就可以很容易地定位到源代码或者类。Import 语句就是用来提供一个合理的路径，使得编译器可以找到某个类。

例如，下面的命令行将会命令编译器载入 java_installation/java/io 路径下的所有类 `import java.io.*;`