Chap05-
Condition and Loop

College of Computer Science
Nankai University

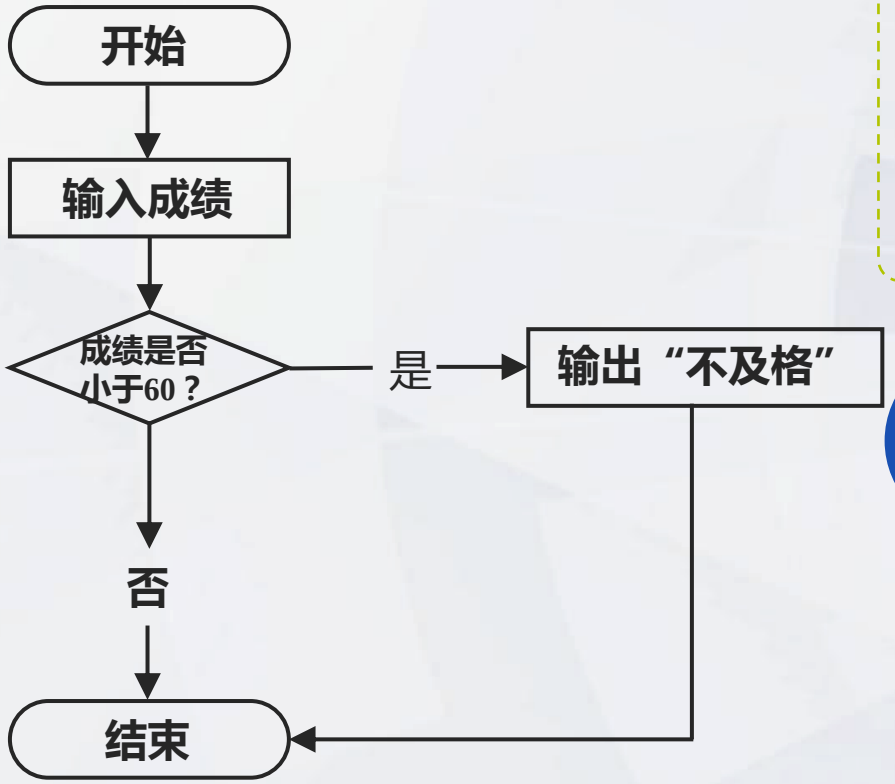
Outlook

5.1 条件语句概述
5.2 条件语句实现和pass
5.3 循环语句概述和for循环
5.4 while循环和索引
5.5 break, continue, else

5.1 条件语句概述

条件语句概述

通过设置条件，可以使得某些语句在条件满足时才会执行。



1 输入成绩并保存到变量score中

2 如果score小于60

3 输出“不及格”

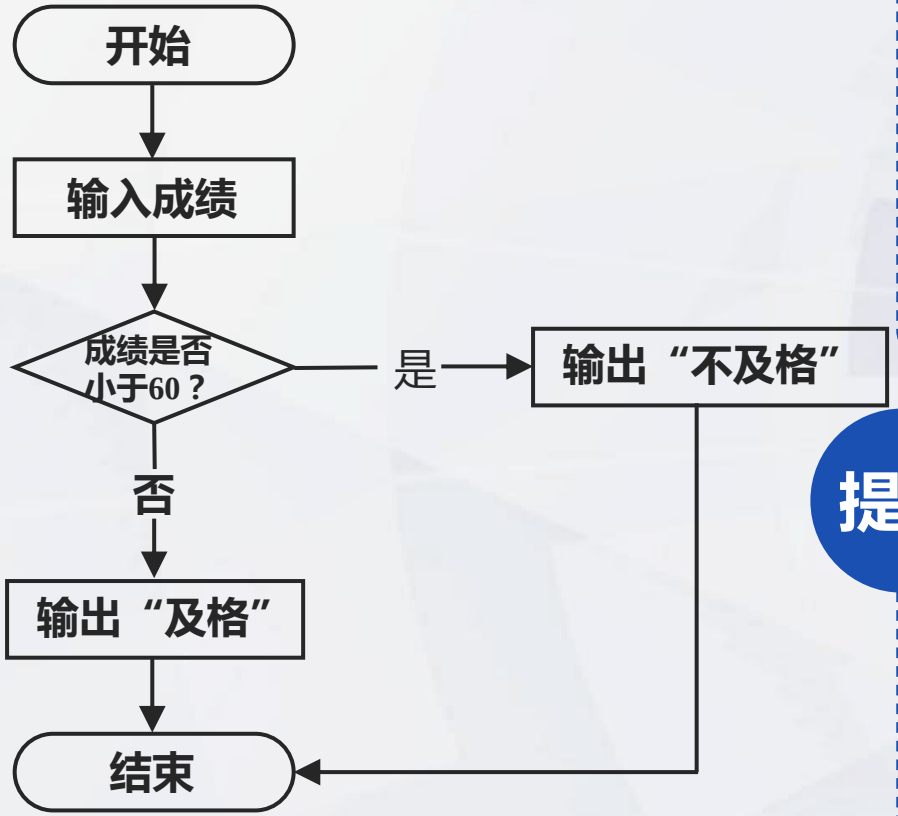
提示

在解决一个实际问题时，可以先使用流程图、自然语言或伪代码等形式描述数据处理流程（即算法设计），再按照设计好的流程（即算法）编写程序。

这样，在设计算法时可以忽略具体代码实现、而专注于如何解决问题，有利于避免程序的逻辑错误。

条件语句概述

通过设置条件，可以使得某些语句在条件满足时才会执行。

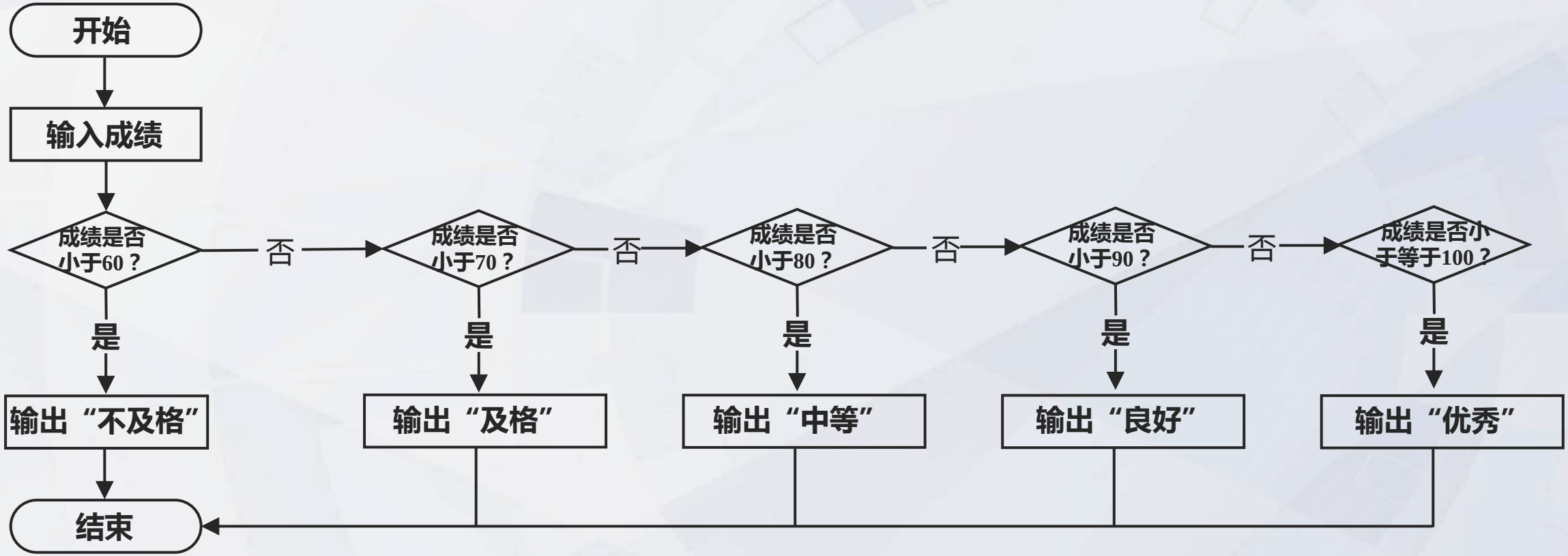


1 输入成绩并保存到变量score中
2 如果score小于60
3 　输出“不及格”
4 否则
5 　输出“及格”

提示

在绘制流程图时，要求必须从“开始”出发，经过任何处理后必然能到达“结束”。另外，流程图中使用的图形符号有着严格规定，“开始”和“结束”一般放在圆角矩形或圆中，数据处理放在矩形框中，而条件判断放在菱形框中。

条件语句概述


开始
输入成绩
成绩是否小于60？
否
成绩是否小于70？
否
成绩是否小于80？
否
成绩是否小于90？
否
成绩是否小于等于100？
是
是
是
是
是
输出“不及格”
输出“及格”
输出“中等”
输出“良好”
输出“优秀”
结束

条件语句概述

例如：

```
1   输入成绩并保存到变量score中
2   如果score小于60
3       输出“不及格”
4   否则，如果score小于70
5       输出“及格”
6   否则，如果score小于80
7       输出“中等”
8   否则，如果score小于90
9       输出“良好”
10  否则，如果score小于等于100 #显然，可以将条件去掉，直接改为“否则”
11      输出“优秀”
```

5.2 条件语句实现和pass

条件语句语法格式

if、elif、else

```
if 条件1:
    语句序列1
[elif 条件2:
    语句序列2
……
elif 条件K:
    语句序列K]
[else:
    语句序列K+1]
```

提示：

if表示“如果”，elif表示“否则如果”，else表示“否则”。最简单的条件语句只有if，elif和else都是可选项，根据需要决定是否使用。

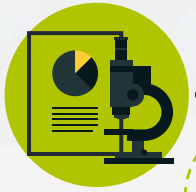
条件语句示例

例如：

```
1    score=eval(input('请输入成绩（0~100之间的整数）：'))
2    if score<60: #注意要写上“:”
3        print('不及格')
```

```
1    score=eval(input('请输入成绩（0~100之间的整数）：'))
2    if score<60:
3        print('不及格')
4    else: #注意else后也要写上“:”
5        print('及格')
```

条件语句示例



例如：

```
1       score=eval(input('请输入成绩（0~100之间的整数）：'))
2       if score<60:
3           print('不及格')
4       elif score<70: #注意elif后也要写上“:”
5           print('及格')
6       elif score<80:
7           print('中等')
8       elif score<90:
9           print('良好')
10      elif score<=100: #也可以改为“else:”
11          print('优秀')
```

条件语句示例

每一个语句序列中可以包含一条或多条语句。例如：

```
1       score=eval(input('请输入成绩（0~100之间的整数）：'))
2       if score<60:
3           print('你的成绩是%d'%score)
4           print('不及格')
```

```
1       score=eval(input('请输入成绩（0~100之间的整数）：'))
2       if score<60:
3           print('你的成绩是%d'%score)
4       print('不及格') #缺少缩进
```

pass

pass表示一个空操作，只起到一个占位作用，执行时什么都不做。

```
1  score=eval(input('请输入成绩（0~100之间的整数）：')
2  if score>=60:
3      pass #什么都不做
4  else:
5      print('不及格')
```

pass

pass表示一个空操作，只起到一个占位作用，执行时什么都不做。

提示：

在某些必要的语句（如条件语句中的各语句序列）还没有编写的情况下，如果要运行程序，则可以先在这些必要语句处写上“pass”，使得程序不存在语法错误、能够正常运行。

实际上，pass与条件语句并没有直接关系，在程序中所有需要的地方都可以使用pass作为占位符。比如，在后面将要学习的循环语句中，也可以使用pass作为占位符。

5.3 循环语句概述和for循环

概述

通过循环，可以使得某些语句重复执行多次。

例如

我们要计算从1到n的和，可以使用一个变量sum=0保存求和结果，并设置一个变量i、让其遍历1到n这n个整数；
对于i的每一个取值，执行sum+=i的运算；遍历结束后，sum中即保存了求和结果。

概述

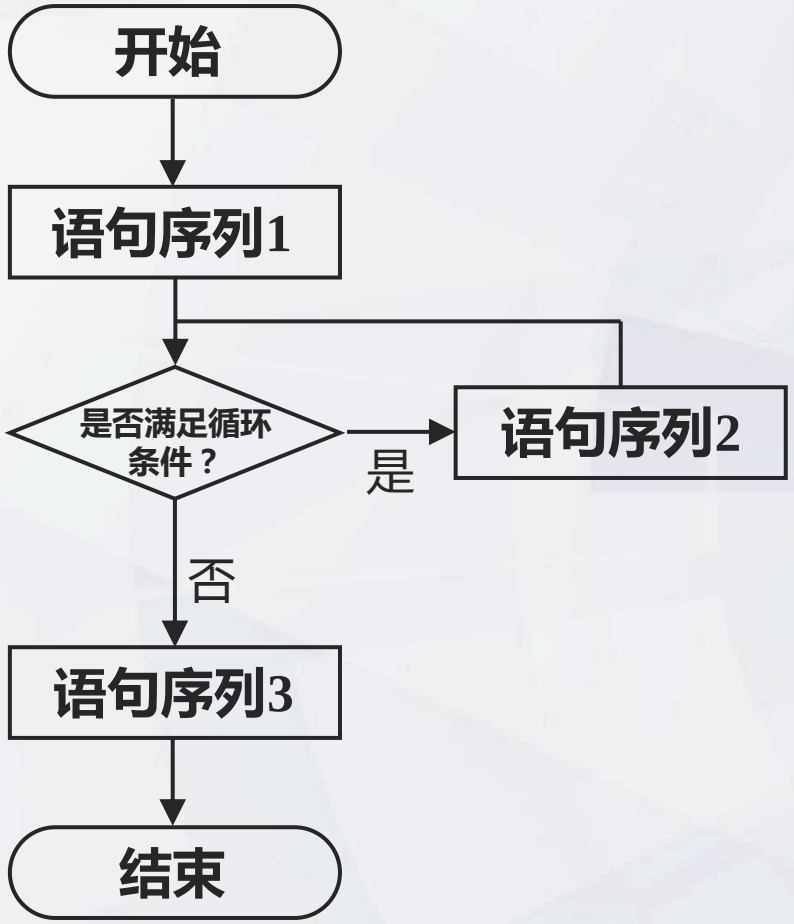
提示

“遍历”这个词在计算机程序设计中经常会用到，其表示对某一个数据中的数据元素按照某种顺序进行访问，使得每个数据元素访问且仅访问一次。

例如

对于列表ls=[1，'Python'，True]中的3个元素，如果按照某种规则（如从前向后或从后向前）依次访问了1、'Python'、True这3个元素，且每个元素仅访问了一次，则可以说对列表ls完成了一次遍历。

循环语句执行过程



提示

循环条件判断和语句序列2构成了循环语句：只要满足循环条件，就会执行语句序列2；执行语句序列2后，会再次判断是否满足循环条件。

for循环

用于遍历可迭代对象中的每一个元素，并根据当前访问的元素做数据处理，其语法格式为：

```
for 变量名 in 可迭代对象:
    语句序列
```

例如

```
1  ls=['Python','C++','Java']
2  for k in ls:
3      print(k)
```

```
Python
C++
Java
```

for循环

再如

```
1  d={'Python':1,'C++':2,'Java':3}
2  for k in d: #注意for后要写上“:”
3      print('%s:%d'%(k,d[k]))
```

```
Python:1
C++:2
Java:3
```

提示

使用for遍历字典中的元素时，每次获取到的是元素的键，通过键可以再获取到元素的值。

for循环

使用for循环时，如果需要遍历一个数列中的所有数字，则通常利用range函数生成一个可迭代对象。

range函数的语法格式如下：

```
range([beg, ]end[, step])
```

for循环

例如

```
1  print(list(range(1,5,2))) #输出“[1, 3]”
2  print(list(range(5,-1,-2))) #输出“[5, 3, 1]”
3  print(list(range(1,5))) #输出“[1, 2, 3, 4]”
4  print(list(range(5))) #输出“[0, 1, 2, 3, 4]”
```

提示

range函数返回的是一个可迭代对象，通过list函数可将该对象转换为列表。

for循环

例 ▶ 使用for循环实现1到n的求和。

```
1  n=eval(input('请输入一个大于0的整数：'))
2  sum=0
3  for i in range(1,n+1): #range函数将生成由1到n这n个整数组成的可迭代对象
4      sum+=i
5  print(sum) #输出求和结果
```

for循环

例 ▶ 使用for循环实现1到n之间所有奇数的和。

```
1  n=eval(input('请输入一个大于0的整数：'))
2  sum=0
3  for i in range(1,n+1,2): #步长2，因此会生成1、3、5、…等奇数
4      sum+=i
5  print(sum) #输出求和结果
```

5.4 while循环和索引

while循环

语法格式

```
while 循环条件:
    语句序列
```

例 ▶ 使用while循环实现1到n的求和。

```
1  n=eval(input('请输入一个大于0的整数：'))
2  i,sum=1,0 #i和sum分别赋值为1和0
3  while i<=n: #当i<=n成立时则继续循环，否则退出循环
4      sum+=i
5      i+=1 #注意该行也是while循环语句序列中的代码，与第4行代码应有相同缩进
6  print(sum) #输出求和结果
```

while循环

例 ▶ 使用while循环实现1到n之间所有奇数的和。

```
1  n=eval(input('请输入一个大于0的整数：'))
2  i,sum=1,0
3  while i<=n:
4      sum+=i
5      i+=2
6  print(sum) #输出求和结果
```

索引

如果希望不仅获取到每一个元素的值，而且能获取到每一个元素的索引，则可以通过len函数获取可迭代对象中的元素数量，再通过range函数生成由所有元素索引组成的可迭代对象。

例 ▶ 同时访问索引和元素值。

```
1  ls=['Python','C++','Java']
2  for k in range(len(ls)): #k为每一个元素的索引
3      print(k,ls[k]) #通过ls[k]可访问索引为k的元素
```

```
0 Python
1 C++
2 Java
```

索引

也可以利用enumerate函数返回的索引序列对象同时获得每个元素的索引和值。

例 ▶ 利用enumerate函数访问索引和元素值。

```
1  ls=['Python','C++','Java']
2  for k,v in enumerate(ls): #k保存当前元素索引，v保存当前元素值
3      print(k,v)
```

```
0 Python
1 C++
2 Java
```

```
1  ls=['Python','C++','Java']
2  for k,v in enumerate(ls,1): #索引从1开始（默认为0）
3      print(k,v)
```

```
1 Python
2 C++
3 Java
```

5.5 break, continue, else

break

用于跳出for循环或while循环。对于多重循环情况，跳出最近的那重循环。

例 ▶ 求1~100之间的素数。

```
1  for n in range(2,101): #n在2~100之间取值
2      m=int(n**0.5) #m等于根号n取整
3      i=2
4      while i<=m:
5          if n%i==0: #如果n能够被i整除
6              break #跳出while循环
7          i+=1
8      if i>m: #如果i>m，则说明对于i从2到m上的取值、都不能整除n，所以n是素数
9          print(n,end=' ') #输出n
```

break

输出结果

```
2  3  5  7  11  13  17  19  23  29  31  37  41  43  47  53
59  61  67  71  73  79  83  89  97
```

continue

用于结束本次循环并开始下一次循环。对于多重循环情况，作用于最近的那重循环。

例 ▶ 3的倍数的整数求和。

```
1  sum=0
2  while True: #因为循环条件设置为True，所以无法通过条件不成立退出循环
                #（永真循环）
3      n=eval(input('请输入一个整数（输入0结束程序）：'))
4      if n==0: #如果输入的整数是0，则通过break跳出循环
5          break
6      if n%3!=0: #如果n不是3的倍数，则不做求和运算
7          continue #通过continue结束本次循环、开始下一次循环，即转到第2
                    #行代码
8      sum+=n #将n加到sum中
9  print('所有是3的倍数的整数之和为：%d'%sum)
```

continue

执行程序时

依次输入10、15、20、25、30、0，则最后输出45（即15+30的结果）。

else

在for循环和while循环后面可以跟着else分支，当for循环已经遍历完列表中所有元素或while循环的条件为False时，就会执行else分支。

例 ▶ 素数判断

```
1  n=eval(input('请输入一个大于1的整数：'))
2  m=int(n**0.5) #m等于根号n取整
3  for i in range(2,m+1): #i在2至m之间取值
4      if n%i==0: #如果n能够被i整除
5          break #跳出while循环
6  else: #注意这个else与第3行的for具有相同的缩进，所以它们是同一层次的语句
7      print('%d是素数'%n)
```

else

执行程序时

如果输入5，则会输出“5是素数”；

如果输入10，则不会输出任何信息。