C 语言程序设计

第五单元
循环结构程序设计

引言

循环结构是结构化程序设计的基本结构之一，它与顺序结构、选择结构一起成为构成各种复杂程序的基础。因此掌握循环结构的概念及其使用是程序设计的最基本的要求。

循环是指在一定条件下对同一个程序段重复执行若干次。循环结构又称重复结构，凡是规律性、重复性的运算等操作，都要用到循环结构。几乎所有实用的程序都包含循环。

C 语言的循环结构用循环语句来实现。共有三种类型的循环语句， while 语句、 do –while 语句和 for 语句。本章要重点的讲述这三个语句的形式、功能及其使用，能用这三个语句编写循环结构程序。

循环结构程序设计

任务 10　淘宝销售衣服价格统计

1. 任务描述
2. 知识准备
3. 知识应用
4. 任务实现

任务描述

本任务设计完成一个淘宝销售衣服价格统计程序。程序运行后，提示输入衣服价格，待用户输入后，循环提示用户输入下一件衣服价格。每次用户输入后判定输入的价格是否为 0 ，如果价格为 0 ，则结束循环并输出销售衣服的件数以及衣服总价，否则继续提示用户输入衣服价格。

知识准备

一、循环结构程序设计思想

在许多实际问题中会遇到具有规律性的重复运算或者重复操作的内容，因此就需要在程序中将某些语句重复执行，这些语句叫做循环体。每重复一次都必须做出是继续重复还是停止重复的决定，这个决定所依据的条件称为循环继续的条件。循环体与循环继续的条件一起构成了所谓的循环结构。例如，求 1~100 的累计和。根据已有的知识，虽然可以用“1+2+3+．．．．．．+100”来求解，但显然很烦琐。进一步，如果要求“1+2+3+．．．．．．+10000”的累计和呢？

现在不妨换个思路来考虑“1+2+3+．．．．．．+100”的累加和。首先设置一个累加和变量 sum ，其初值为 0 ，利用 sun=sun+n 来计算（ n 依次取 1 、2、．．．．．．、 100 ）， n 为循环控制变量，用于控制循环的次数（即构成循环继续的条件）。

从上述问题解决可以看出，循环结构程序的设计要考虑两个方面：

1 ．循环继续的条件。循环继续的条件是循环结构设计的关键，它决定着循环体重复执行的次数。通常可以利用关系表达式和逻辑表达式来构成。

2 ．循环体。循环体是需要重复执行的语句。它可以是顺序结构语句，也可以是选择结构语句，甚至也可以是循环结构语句。

循环结构程序的设计就是要正确描述循环继续的条件并针对问题分析出其规律性，利用循环控制语句进行处理。

知识准备

二、 while 语句介绍

1 ． while 语句用来实现“当型”循环结构。其一般形式为：

```
while （ 表达式 ）
{
    循环体语句
}
```

2 ． while 语句的流程图如下图 5-1 所示：

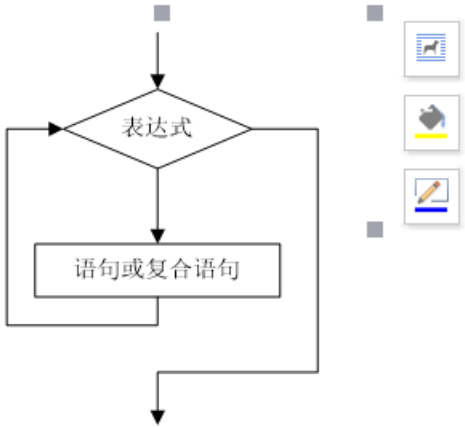


图 5-1 while 语句流程

3 ．执行过程： while 语句执行时，首先计算表达式（循环继续的条件）的值，若表达式为非 0 （真），则执行循环体语句，然后再计算表达式的值，只要不为 0 ，继续执行循环体语句，如此重复，直到表达式的值为 0 （假）为止，流程控制转到循环结构的下一条语句。

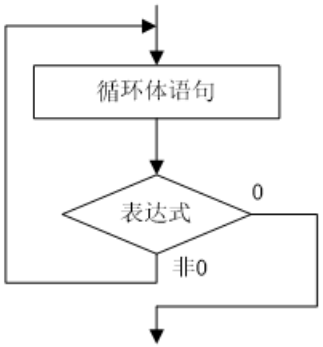
知识准备

三、 do-while 语句介绍

1 ． do-while 语句的一般形式：

```
do
{
循环体语句
}while ( 表达式 ) ;
```

2 ． do-while 语句流程如下图 5-2 所示。



3 ．执行过程： while 语句执行时，首先计算表达式（循环继续的条件）的值，若表达式为非 0 （真），则执行循环体语句，然后再计算表达式的值，只要不为 0 ，继续执行循环体语句，如此重复，直到表达式的值为 0 （假）为止，流程控制转到循环结构的下一条语句。

知识准备

四、 while 语句与 do_while 语句特点及使用注意事项：

1 ． while 语句的特点：先判断表达式，后执行语句，循环体语句至少执行 0 次；do_while 语句特点：先执行一次循环体语句，后判断表达式，循环体语句至少执行 1 次；这是 while 语句与 do_while 语句的本质区别。

2 ． while 与 do_while 语句中的循环体语句也称为循环体，它既可以是简单语句也可以是复合语句。

3 ．循环体内一定要有改变循环继续条件的语句，使循环趋向于结束，否则循环将无限进行下去，造成“死循环”。

4 ．为使循环能正确开始运行，还要做好循环前的准备工作，就是给循环变量初始化，使其有初值。

【知识应用】

1 ．用 while 语句实现从键盘依次输入学生的 C 语言成绩，当输入 -1 时，停止输入，输出学生人数，及平均成绩。

分析：

（ 1 ）“当输入 -1 时，停止输入” ，不等于 -1 要输入学生成绩（循环条件）。很容易想到“当型”循环，凡是程序题意描述中出现此类字样均可考虑用“当型”语句解决。

（ 2 ）若输出学生人数，就要设置一个变量，输入一个学生成绩，变量增 1 。

（ 3 ）平均成绩 = 总成绩 / 学生人数，因此要对输入的成绩进行累加，需设置累加和变量，其初值应清零。

知识应用

用 while 语句编写的程序如下：

```
#include<stdio.h>
main()
{
    int n;
            float sum , score,average ;
            n=0;
            sum=0;
            printf(" 请输入学生成绩： \n");
            scanf ("%f", &score);
    while ( score!= -1)/* 输入 -1 时，循环结束 */
            {
       n++;  /* 记录学生人数 */
       sum=sum+score; /* 成绩累加 */
       scanf ("%f", &score);
            }
            average=sum/n; /* 求平均成绩 */
    printf(" 学生人数是 %d , 平均成绩是： %.2f\n", n, average);
}
```

知识应用

2 ．用 do-while 实现提款机密码验证功能。

分析：

（ 1 ）在银行自动提款机取钱时，输入密码只给 3 次机会，如果输入密码正确，则提示“密码正确，请取款！”；如果 3 次输入密码后都不正确，则提示“密码错误，请取卡！”。对于密码是否争取的判断需要进行 3 次，至少执行 1 次，因此选择 do_while 循环语句实现。

知识应用

（ 2 ）本程序假设密码为 “ 123”，如果输入密码为 “ 123”，则提示 “密码正确，请取款！” ，否则提示“密码错误，请取卡！” 。

```
#include<stdio.h>
main()
{
   long pw;
              int i=0;
              int temp=0;
              do
              {
                        printf(" 请输入密码： ");
                        scanf("%ld",&pw);
                        i++;
                        if(pw==123) /* 判断密码是否为 123*
                        {
                                  temp=1; /* 密码输入正确时， temp 值变为 1*/
                                  break;
                        }
              }while(i<3);
              if(temp==1)
                        printf(" 密码正确，请取款！ \n");
              else
                        printf(" 密码错误，请取卡！ \n");
}
```

任务实施

- **一、任务流程分解**

1．流程描述：程序开始，提示用户输入第 1 件衣服价格，待用户输入后循环提示用户输入下一件衣服价格，每次用户输入后判定输入的价格是否为 0 ；如果输入为 0 ，则结束循环并输出销售衣服的件数以及衣服总价，否则继续提示用户输入衣服价格。

（ 1 ）程序初始化分析：初始化变量，提示用户输入第 1 件衣服价格

（ 2 ）数据录入分析：记录用户每次输入的衣服价格

（ 3 ）数据处理分析：判断每次用户输入的价格是否为 0 ，累加用户输入的衣服价格

（ 4 ）输出结果分析：输出衣服的件数和衣服的总价。

任务实施

二、代码实现

```
#include<stdio.h>
main ( )
{
        int count,num,total;
        count=1;  /* 用 count 记录衣服的件数 */
        total=0;   /*total 记录衣服的总价 */
        printf(" 请输入第 1 件衣服价格（元）： ");
        scanf("%d",&num);
        while (num!=0) /* 输入 0 时，循环结束 */
        {
                total =total+ num; /* 衣服价格累加 */
                count++;         /* 衣服件数增 1*/
                printf (" 请输入第 %d 件衣服价格（元）： ", count);
                scanf ("%d", &num);
        }
        printf(" 今天累计销售衣服 %d 件，总价为： %d 元 \n", count-1,total); /*  最后一件衣服的价格输入为 0 ，是循环结束条件，因此衣服的实际件数为 count-1 */
}
```

任务实施

三、结果演示

输入 5 件衣服的价格，输出累计销售衣服的件数以及总价格的结果，程序结果演示如图 5-3 （ 1 ）所示。

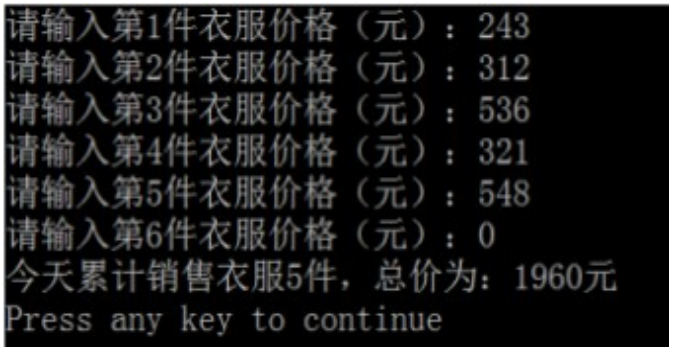
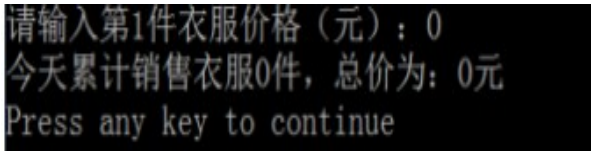


图 5-3 演示结果界面（ 1 ）

输入 0 件衣服的价格，输出累计销售衣服的件数以及总价格的结果，程序结果演示如图 5-3 （ 2 ）所示。



任务 11 警察抓逃犯

1 任务描述

2 知识准备

3 知识应用

4 任务实现

任务描述

本任务设计完成一个简单警察抓逃犯程序。一天夜里一司机撞伤行人之后逃跑，经交警调查后，有 4 名目击者，但是他们都没有看清楚号牌，只有一些模糊记忆。

甲说：“车牌号的后两位相同”；

乙说：“车牌号是 4 位数”；

丙说：“车牌号的前两位加起来为 8”；

丁说：“车牌号尾号是偶数，第一位不是 0”；

请编写程序确定车牌范围，帮助交警抓逃犯。

知识准备

C 语言中的 for 语句使用最为灵活，它不仅可以用于循环次数已经确定的情况，而且可以用于循环次数不确定而只给出循环结束条件的情况，它完全可以代替 while 语句和 do-while 语句。

知识准备

一、 for 语句介绍

1 ．for 语句来的一般形式：

for（表达式 1；表达式 2；表达式 3 ）

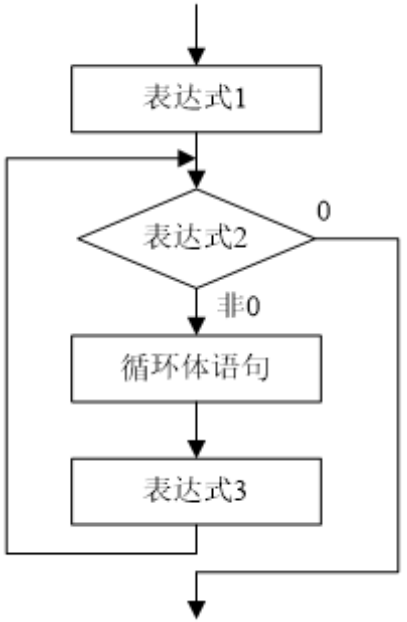
循环体语句

例如：

for （ i=1;i<=100;i++ ） /*1~100 的累加 */

sum=sum+i;

2 ．for 语句流程图及其执行过程如图 5-4 所示：



执行 for 语句时，首先执行表达式 1 ，然后计算表达式 2 的值，若表达式 2 的值非 0 ，则执行循环体语句，接着执行表达式 3 ，再计算表达式 2 ，并判断表达式 2 的值是否非 0 ，若非 0 ，接着执行循环体语句，如此循环下去，直到表达式 2 的值为 0 时，循环结束，流程控制转到循环结构的下一语句。

知识准备

3 ． for 语句使用注意事项

（ 1 ）for 语句中，表达式 1 通常为赋值表达试，也可以是逗号表达式，一般为循环变量赋初值，仅执行一次；表达式 2 通常是关系或逻辑表达式，有时也可以是数值表达式或者字符表达式，用来作为循环继续的条件。表达式 3 通常为算术表达式或赋值表达试，使循环变量改变。

（ 2 ）循环体语句可以是单语句，也可以是复合语句。

（ 3 ）for 语句中的 3 个表达式都能省略，但是分号必须保留。

① 表达试 1 省略，此时要注意应在 for 语句之前给循环变量赋初值。 例如：

```
i=1;
for(; i<=100;i++)
{
      printf("%d",i);
}
```

② 表达式 3 省略，此时要注意应循环体内有改变循环变量值的语句来结束循环。否则循环将无限进行下去。例如：

```
for(i=1; i<=100;)
{
   printf("%d",i);
   i++;
}
```

知识准备

③ 表达式 2 省略，表示无条件循环，就是认为表达式 2 始终为真，此时可以在循环体内使用 if 语句和 break 语句相配合来实现循环的结束。例如：

```
for(i=1;;i++)
{
   printf("%d",i);
   if(i>100) break;      /*break 语句用来结束循环结构，其作用后面会详细介绍*/
}
```

④ 表达式 1 和表达式 3 省略，即只给出了循环条件，这种情况下， for 语句完全等同于 while 语句。例如：

```
i=1;
for(;i<=100;  )
{
      printf("%d",i);
      i++;
}
```

```
i=1;
while(i<=100)
{
      printf("%d",i);
      i++;
}
```

⑤ 三个表达式都可省略，如：

for(;;) 语句　相当于　while（ 1 ）语句

知识应用

1 ．农场里有鸡和兔，已知鸡和兔一共有 40 只，脚一共有 100 只，请统计鸡兔各有多少只。

分析：

（ 1 ）定义两个整型变量 ji ， tu 分别表示鸡和兔的个数；

（ 2 ）鸡从 1 开始最多到 39 只，即 ji 的取值范围为 1~39 ，用 for 语句实现；鸡兔共有 40 只，那么兔的个数为： 40-ji ；

（ 3 ）每只鸡 2 只脚，每只兔 4 只脚，需要满足共有 100 只脚的条件，用 if 语句判断 ji*2+tu*4==100 。条件成立输出鸡和兔的个数。

```
#include<stdio.h>
main() {
        int ji,tu;
        for(ji=1; ji<=39; ji++) /*ji 的取值范围为 1~39*/
        {
                tu=40-ji;
                if(2*ji+4*tu==100) /* 满足脚有 100 只条件时，输出鸡兔的个数 */
                printf(" 鸡 %d 只，兔 %d 只 \n",ji,tu);
        }
}
```

知识应用

- 2 ．某校组织学生参加夏令营，共计 15 天，共计行程 414km 。已知同学们晴天日行 32km ，雨天日行 21km 。编写程序，计算整个夏令营期间，雨天、晴天各多少天。
- 分析：
- （ 1 ）定义两个整型变量 qt ， yt 分别表示晴天数和雨天数；
- （ 2 ）晴天数从 1 开始最多到 14 天，即 qt 的取值范围为 1~14 ，用 for 语句实现；夏令营共计 15 天，那么雨天数为： 15-qt ；

知识应用

（ 3 ）晴天日行 32km ，雨天日行 21km ，共计行程 414km 。用 if 语句判断 qt*32+yt*21==414 。条件成立输出晴天和雨天的个数。

```
#include<stdio.h>
main()
{
        int qt,yt;
        for(qt=1;qt<=14;qt++)/* 晴天取值范围为 1~14*/
        {
                yt=15-qt;
                if(qt*32+yt*21==414) /* 满足共计行程 414 时，输出晴天和雨天天数 */
                        printf(" 晴天： %d 天，雨天： %d 天 \n",qt,yt);
        }
}
```

任务实施

一、任务流程分解

1．流程描述：首先，车牌号是四位数，车牌范围应为 1000~9999；然后，假设车牌四位数分别为 q，b，s，g，那么车牌的后两位相同，即 s==g；车牌的前两位相加为 8，即 q+b==8；车牌尾号为偶数，即车牌可以被 2 整除。

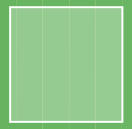
（1）程序初始化分析：设定 for 循环条件，chepai 循环条件初始值为 1000，结束值为 9999

（2）数据处理分析：q=chepai/1000; b=chepai/100%10;s=chepai/10%10;g=chepai%10;

用 if 语句进行判断，是否满足 s==g && q+b==8 && chepai%2==0 条件。

（3）输出结果分析：输出车牌的可能值

任务实施



二、代码实现

```
#include<stdio.h>
main()
{
        int chepai,q=0,b=0,s=0,g=0;
        for(chepai=1000;chepai<=9999;chepai++)
        {
                q=chepai/1000;  /* 计算 4 位车牌的千位 */
                b=chepai/100%10; /* 计算 4 位车牌的百位 */
                s=chepai/10%10; /* 计算 4 位车牌的十位 */
                g=chepai%10; /* 计算 4 位车牌的个位 */
                if(s==g && q+b==8 && chepai%2==0 ) /* 判断满足目击证人的条件
输出车牌 */
                        printf("%d\t",chepai);
        }
}
```

任务实施

三、结果演示

根据已知的条件判断所有可能，逐一输出符合条件的可能的结果，程序结果演示如图 5-5 所示。

```
可能的车牌号为:
1700    1722    1744    1766    1788
2600    2622    2644    2666    2688
3500    3522    3544    3566    3588
4400    4422    4444    4466    4488
5300    5322    5344    5366    5388
6200    6222    6244    6266    6288
7100    7122    7144    7166    7188
8000    8022    8044    8066    8088
Press any key to continue
```

图 5-5 演示结果界面

循环结构程序设计

任务 12　水仙花数

1. 任务描述
2. 知识准备
3. 知识应用
4. 任务实现

任务描述

打印出所有的“水仙花数”。所谓“水仙花数”是指一个 3 位数，其个位、十位和百位数字的立方和等于该数本身。例如，153 就是一水仙花数，因为 153=13+53+33。程序通过嵌套的 for 循环分别控制个位、十位及百位数的增长，在最内层循环体中判断是否符合水仙花数的条件，符合即打印出结果。

知识准备

循环嵌套

在实际的一些问题中，复杂的问题往往单层循环结构无法实现，需要用两层或多层循环结构才能解决。在一个循环体内包含另一个循环结构，称为循环的嵌套。三种循环不仅可以自身嵌套，还可以相互嵌套。

```
(1) while ( e1)
    {
      …
      while (e2 )
        {s}
      …
    }
```

```
(2) while (e1 )
    {
      …
      do
        {s}while (e2 );
      …
    }
```

```
(3)while (e1 )
    {
      …
      for( e2;e3;e4 )
        {s}
      …
    }
```

```
(4)do
    {
      …
        do
          {s}while(e2 );
      …
    }while (e1);
```

```
(5)do
    {
      …
      while(e2 )
        {s}
      …
    }while (e1);
```

```
(6)do
    {
      …
      for(e2;e3;e4 )
        {s}
      …
    }while (e1);
```

知识准备

(7) for(e1;e2;e3)
{
…
for(e4;e5;e6)
{s}
…
}

(8) for(e1;e2;e3)
{
…
while(e4)
{s}
…
}

(9) for(e1;e2;e3)
{
…
do
{s}while(e4);
…
}

循环嵌套的形式

知识应用

1 ．全班有 40 名学生，每名学生考 7 门课。要求分别统计出每名学生的平均成绩。

程序分析：利用 while 循环控制学生人数为 40 人，循环体中利用 for 循环控制录入每名学生的各门课成绩，每次录入完成全部 7 门成绩后计算改名同学的并输出，然后继续输入下一名同学的成绩。全部 40 名同学成绩输入完毕后，提示“成绩计算完毕。”

```
#include<stdio.h>
main ( ) {
        int i,j,score,sum;
        float aver;
        j=1;
        while(j<=40) {
                sum=0;
                for(i=1;i<=7;i++) {
                        printf(" 输入第 %d 名同学的第 %d 门成绩 :",j,i);
                        scanf("%d",&score);
                        sum=sum+score;
                } aver=sum/7;
                printf(" 第 %d 名同学的平均成绩为 ： %.2f\n",j,aver);
                j++;
        }
        printf(" 成绩计算完毕！ ");
}
```

知识应用

2 ．输出九九乘法表。

分析：定义两个整型变量 i ， j 分别表示行数和列数，用双重循环控制 i 和 j 的变化，外层 i 的变化为 1~9 ，内存 j 的变化为 1~i ；在内层循环中输出 i ， j 及 i* j ；内层循环结束时输出换行符，外层循环结束即程序结束。

```
#include<stdio.h>
main()
{
    int i,j;
    for(i=1;i<=9;i++) /* 用双重循环，控制 i ， j 的变化 */
    {
        for(j=1;j<=i;j++)
            printf("%d*%d=%2d\t",j,i,i*j);  /*  输出 i ， j ， i 与 j 的乘积  */
        printf("\n"); /* 每一行结束后，输出换行 */
    }
}
```

任务实施

一、任务流程分解

1．流程描述：要求打印出所有的“水仙花数”。所谓“水仙花数”是指一个 3 位数，其各位数字立方和等于该数本身。例如，153 是一水仙花数，因为 153=13+53+33。程序首先限定 for 循环条件：个位数从 1 到 9 循环，嵌套十位数从 0 到 9 循环，嵌套百位数从 0 到 9 循环，利用 if 语句判定个位、十位和百位立方之和是否等于这个三位数值，如果是则输出水仙花数，否则继续循环，直至循环从最内层到最外层结束。

（1）程序初始化分析：初始化 for 循环的个位、十位、百位数字

（2）数据录入分析：由外向内逐层循环

（3）数据处理分析：最内层循环体判断是否符合水仙花添加

（4）输出结果分析：输出水仙花数字。

任务实施

二、知识扩展

1．在循环嵌套时，要保证一个循环完全在另一个循环之中，不能出现循环结构交叉。

2．在循环嵌套执行过程中，要注意内部循环执行结束后，方可执行外部循环。

任务实施

三、代码实现

```
#include <stdio.h>
main()
{
    int a,b,c;
    printf(" 水仙花数是 :\n");
    for(a=1;a<10;a++)
        for(b=0;b<10;b++)
            for(c=0;c<10;c++)
            {
                if(a*a*a+b*b*b+c*c*c==a*100+b*10+c)
                    printf("%-5d\n",a*100+b*10+c);
            }
}
```

任务实施

四、结果演示

输出“水仙花数”的结果，程序结果演示如图 5-6 所示。

```
水仙花数是:
153
370
371
407
Press any key to continue
```

图 5-6 演示结果界面

循环结构程序设计

任务 13　猜数字

1. 任务描述
2. 知识准备
3. 知识应用
4. 任务实现

任务描述

本任务设计完成一个猜数字程序。程序运行后，按照提示信息，用户输入数字，该数字与系统随机产生数字进行比较，输出这两个数字的大小关系，作为用户下次输入数据的提示信息；当用户输入数字与系统随机产生的数字相等，即数字被猜对了，提示用户“恭喜您猜中了”，并输出用户猜数字所用时间。

知识准备

为了使循环控制更加灵活，C 语言提供了两种无条件转移控制语句的——— break 语句、 continue 语句。无条件转移控制语句的主要作用是改变程序的运行流程。

知识准备

一、 break 语句的使用

1 ．break 语句格式： break;

2 ．break 语句的功能：

（ 1 ）用在 switch 语句，使某个 case 子句执行后，控制立刻跳出 switch 结构。

（ 2 ）用在循环语句的循环体中，可以从循环体内跳出循环体，提前结束该层循环，继续执行后面的语句。

- 3 ．break 语句使用注意事项：
- （ 1 ）在嵌套循环中， break 语句仅能退出一层 (当前) 循环。
- （ 2 ）若在循环语句中包含了 switch 语句，那么 switch 语句中的 break 语句仅能使控制退出 switch 语句。

知识准备

二、 continue 语句的使用

1 ． continue 语句格式： continue;

2 ． continue 语句的功能：

（ 1 ）continue 语句仅能在循环语句中使用，它的作用是结束本次循环，而是开始一次新的循环。

（ 2 ）对于 for 语句，将控制转到执行表达式 3 和条件测试部分。

（ 3 ）对于 while 和 do-while 语句，将控制转到条件测试部分

3 ． continue 语句使用注意：

（ 1 ） continue 语句只能用在循环结构，不能用在 switch 结构中。

（ 2 ） continue 语句常用来和 if 语句配合使用，用来实现特定作用的循环。

知识准备

三、 break 语句与 continue 语句的比较：

1 ． continue 语句跟 break 语句有相似，它们均可以和 if 语句配合使用，用来实现特定作用的循环，但是 continue 语句使用的频率要少得多。

2 ． continue 语句与 break 语句不同， continue 语句只能结束本次循环，而不是跳出循环体，它跳出当前的这一次循环，马上进行下一次循环的判断（在 while 语句和 do-while 语句里，这就意味着马上执行条件测试部分；在 for 语句里，程序执行方向转到增量步骤，也就是表达式 3 的位置）。而 break 语句则是结束整个循环过程，不再判断执行循环的条件是否成立。

知识应用

1 ．分析程序运行结果，体会 break 语句作用

```
#include<stdio.h>
main()
{
        int i;
        for(i=0;i<=10;i++)
        {
                if(i%5==0)
                        break;
                printf("%4d",i);
        }
}
```

2 ．打印 A 到 Z 这 26 个字母 ，然后只打印出 M 。

```
#include<stdio.h>
main()
{
        int ch=0;
        for(ch=65;ch<=90;ch++)
                printf("%c\t",ch);
        printf("\n\n");
        for(ch=65;ch<=90;ch++)
        {
                if(ch==77)
                {
                        printf("%c\t",c
h);
                        break;
                }
        }
        printf("\n\n");
}
```

知识应用

3 ．分析程序运行结果，体会 continue 语句作用

```
#include<stdio.h>
main()
{
        int i;
        for(i=0;i<=10;i++)
        {
                if(i%5==0)
                        continue;
                printf("%4d",i);
        }
}
```

4 ．打印 A 到 Z 这 26 个字母 ，然后在基础上去掉 Q 。

```
#include<stdio.h>
main()
{
        int ch=0;
        for(ch=65;ch<=90;ch++)
                printf("%c\t",ch);
        printf("\n\n");
        for(ch=65;ch<=90;ch++)
        {
                if(ch==81)
                        continue;

                printf("%c\t",ch);
        }
        printf("\n\n");
}
```

任务实施

一、任务流程分解

1．游戏流程描述：开始游戏，程序自动产生要猜测的数字（范围在 1-999 之间的随机数），用户进行数字猜测，程序根据用户猜测，产生不同的应答，当猜测正确时，显示用户猜测数字所用时间。

（1）程序初始化分析：记录当前时间，产生随机数

（2）数据录入分析：用户录入猜测的数字

（3）数据处理分析：产生随机数和猜测数字比较大小关系

（4）输出结果分析：

结果 1：随机数大于猜测数字，程序提示：太大了；

结果 2：随机数小于猜测数字，程序提示：太小了；

结果 3：当随机数等于猜测数字，程序提示：猜对了，您的用时为多少秒。

任务实施

二、知识扩展

1 ．产生随机数 rand()

（ 1 ）引入头文件： stdlib.h

（ 2 ）使用方式：

```
srand((unsigned)time(NULL));  /* 设置一次随机种子 */
num = rand();/* 产生一个 0-65535 之间的随机数 */
```

（ 3 ）注意： rand() 函数使用要先设置随机种子，才可以使用。

2 ．时间记录 clock() ：

（ 1 ）引入头文件： time.h

（ 2 ）使用方式： clock();/* 产生一个毫秒为单位的数字。 */

（ 3 ）注意：从“开启这个程序进程”到“程序中调用 clock() 函数”时之间的 CPU 时钟计时单元（ clock tick ）数，单位为毫秒。

任务实施

三、代码实现

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
main(){
printf("================================\
n");
printf(" ***** 欢迎使用猜数字软件 *****\n");
printf("================================\
n");
printf(" 请问是否要进行猜数字游戏 (Y/N):");
        char ch;
        int num,ce,js;
        float fk,fe,fh;
        ch=getchar();
        if(ch=='Y'||ch=='y')
{       while(1){
        printf(" 开始计时 \n");
srand((unsigned)time(NULL));
        num = rand()%1000
        fk=clock();
while(1)
{
printf(" 请进行数字猜测 (1-100
0) : "); scanf("%d",&ce);
        if(ce>num)
    printf(" 太大了 \n");
        if(ce<num)
        printf(" 太小了  \n");
        if(ce==num)
        {
        fe=clock();

        fh=(fe-fk)/1000.0;
```

任务实施

```
printf(" 恭喜你猜对了！ \n");
            printf(" 你的用时为 %.4fs\n",fh);
            break;
                                    }
                        }
printf(" 按 1 重新进行游戏，按 2 退出游戏： ");
scanf("%d",&js);
if(js==2)               break;
            }       }
else
            if(ch=='N'||ch=='n')
                        printf(" 退出程序！ \n");
            else
                        printf(" 您的输入有误！ \n");
```

任务实施

四、结果演示

输入 Y 开始计时猜数字，随机猜测数字直到正确输出所用时间，根据提示输入 1 重新开始猜数字游戏，程序结果演示如图 5-9（ 1 ）所示。

```
==================================
 ******欢迎使用猜数字软件******
==================================
请问是否要进行猜数字游戏(Y/N):Y
开始计时
请进行数字猜测(1-1000)： 500
太大了
请进行数字猜测(1-1000)： 167
恭喜你猜对了！
你的用时为14.8090s
按1重新进行游戏， 按2退出游戏： 1
开始计时
请进行数字猜测(1-1000)：
```

输入 N 退出程序，程序结果演示如图 5-9（ 2 ）所示。

```
==============================
 *****欢迎使用猜数字软件*****
==============================
请问是否要进行猜数字游戏(Y/N):N
退出程序!
Press any key to continue
```

任务实施

输入其他内容，输出您的输入有误的结果，程序结果演示如图 5-9（ 3 ）所示。

```
==============================
 *****欢迎使用猜数字软件*****
==============================
请问是否要进行猜数字游戏(Y/N):D
您的输入有误!
Press any key to continue
```

图 5-9 演示结果界面（ 3 ）

本章小结

本章重点内容为三种循环控制语句的使用及如何运用循环嵌套来解决具有规律性重复运算或重复操作。利用三种流程控制语句（ while 、 do_while 、 for ）编写循环结构程序，其共同点是：三种循环都可以处理同一问题，一般可以互相代替；均可以用 break 语句跳出循环，用 continue 语句结束本次循环；他们可以相互嵌套构成多重循环。其不同点是： while 和 for 结构是先判定表达式，后执行循环，若条件不满足，则不执行循环体，而 do-while 结构是先执行循环语句，后判定表达式。在解决实际问题时，要根据问题的需要，选择适当的语句，达到程序结构清晰、简洁、可读性强。

本章难点内容为 break 与 continue 的灵活运用。在使用时它们通常与 if 语句配合使用出现在循环结构程序中。二者主要区别是： break 语句可以用于 switch 结构或者循环结构，而 continue 语句只能用在循环结构。在循环结构程序中 break 语句是结束整个循环过程，不再判断执行的循环条件是否成立，而 continue 语句只结束本次循环，而不是终止整个循环的执行。

练习题

一、选择题

1．下面有关 for 循环的正确描述是（　　）。

A．for 循环只能用于循环次数已经确定的情况。

B．for 循环是先执行循环体语句，后判定表达式。

C．在 for 循环中，不能用 break 语句跳出循环体。

D．for 循环体语句中，可以包含多条语句，但要用花括号括起来。

2．对于 for（表达式 1 ；；表达式 3 ）可理解为（　　）。

A．for（表达式 1 ； 0 ；表达式 3 ）

B．for（表达式 1 ； 1 ；表达式 3 ）

C．for（表达式 1 ；表达式 1 ；表达式 3 ）

D．for（表达式 1 ；表达式 3 ；表达式 3 ）

3．执行语句 for(i=1;i++<4;) ; 后变量 i 的值是（　　）。

A．3　　B．4　　C．5　　D．不定

4．语句 while(!E); 中的表达式 !E 等价于（　　）。

A．E==0　　B．E!=1　　C．E!=0　　D．E==1

5．t 为 int 类型，进人下面的循环之前， t 的值为 0 ，则以下叙述中正确的是（　　）。

while(t=l){……}

A．循环控制表达式的值为 0　　B．循环控制表达式的值为 1

C．循环控制表达式不合法　　D．以上说法都不对

二、填空题

1．以下程序的功能是：计算 1 到 10 之间的奇数之和及偶数之和。

```
  main()
{
int a,b,c,i;
a=c=0;
for(i=0;i<=10;i+=2)
{
 a+=i;
_____;
c+=b;
}
printf("The sum of even number is %d\n",_____);  /* 输出偶数之和 */
printf("The sum of odd number is %d\n",c-11); /* 输出奇数之和 */
}
```

2．下面程序的功能是输出 100 以内能被 3 整除且个位数为 6 的所有整数。

```
main( )
{
int i,j;
for(i=0;_____;i++)
{
j=i*10+6;
if (_____)
continue;
printf("%3d",j);
}
}
```

三、程序设计

1 ．求 s=12+22+32+……92 的值。

2 ．试编程求 100 以内的所有完全数。（如果一个数除自身之外，恰好等于它的所有因子之和，则该数为完全数。例如： 6=1+2+3 ，则 6 就是一个完全数。 ）

3 ．编写程序，输出从公元 1000 至 2000 年所有闰年的年号。每输出 3 个年号换一行。判断公元年是否为闰年的条件是：

（ 1 ）公元年数如能被 4 整除，而不能被 100 整除，则是闰年；

（ 2 ）公元年数能被 400 整除也是闰年。

C 语言程序设计

Thank You!