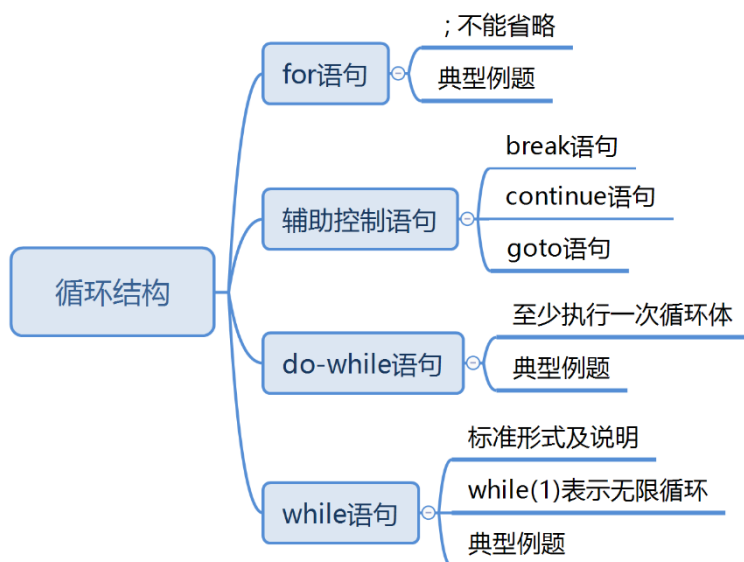




专题四 循环结构

【内容预览】



【知识清单】

4.1、while 语句

4.1.1、while 语句的标准形式

循环结构是指在给定条件时，反复执行某个程序段。

它的一般形式为：

```
while (表达式)
{
    循环体语句;
}
```

1.while 循环的表达式是循环进行的条件。用作循环条件的表达式一般是一个能够改变表达式值的变量，这个值称为循环体变量。

2.当条件表达式的值为真（即非 0）时，执行循环体；为假（即 0）时循环体结束；**while (1)** 表示无限循环。

3.当循环体不需要进行任何功能时，可以用空语句作为循环体。

该语句形式多用于后面指针章节里的通过 while 语句，使得指针由指向首地址到指向尾地址。

如：**while(*(p++2));**

4.对于循环体变量的初始化应该在 while () 语句之前，**但是往往在考试中会有在 while () 语句里进行赋值的题型**，这一点较为灵活，在后面会进行讲解。

4.1.2、while 语句的典型例题

1. 用 while 循环语句编写求 $\sum_{i=1}^{100} i$ 值的循环

```
#include<stdio.h>
void main()
{
    int i;
    int sum;
    sum=0;
    i=1;           //定义变量 sum 的初值为 0, i 的初值为 1
    while(i<=100)  //当 i>大于 100, 条件表达式 i<=100 的值为假, 不执行循环体
    {
        sum=sum+i;
        i++;       //加完后, i 的值加 1
    }
    printf("sum=%d\n", sum);
}
```

2. 用 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \dots\dots\dots$ 公式求 π 的近似值, 直到发现某一项的绝对值小于 10^{-6} 为止。

```
#include<stdio.h>
#include<math.h>           //程序中用到数学函数 fabs, 应包含头文件 math.h
int main()
{
    int sign=1;             //sign 用来表示数值的符号

    double pi=0.0,n=1.0,term=1.0;//pi 开始表达多项式的值, 最后代表  $\pi$  的值, n 代表分母。

                                term 代表当前项的值
    while(fabs(term)>=1e-6) //检查当前项 term 是否大于或等于  $10^{-6}$ 
    {
        pi=pi+term;         //把当前项累加到 pi 中
        n+=2;               //n+2 是下一项的分母
        sign=-sign;         //sign 代表符号, 下一项的符号与上一项的符号相反
        term=sign/n;        //求出下一项的值 term
    }

    pi=pi*4;                //多项式的和 pi 乘以 4, 才是  $\pi$  的近似值

    printf("pi=%10.8f\n",pi);
    return 0;
}
```

3. **译密码**。为使电文保密, 往往按一定的规律将其转换成密码, 收报人再按约定的规律将其译回原文。例如, 可以按以下规律将电文转化变成密码:

将字母 A 变成字母 E, a 变成 e, 即变成其后的第 4 个字母, W 变成 A, X 变成 B, Y 变成 C, Z 变成 D。

例如

“China!”转换为“Glmre!”。

```
#include<stdio.h>
void main()
{
    char c;
    c=getchar();           //输入一个字符给字符常量 c，参见第五章的 5.4.2 小节
    while(c!= '\n')        //检查 c 的值是否是换行符'\n'
    {
        if((c>='a'&& c<='z')||(c>='A'&& c<='Z')) //c 如果是字母
        {
            if((c>='W'&& c<='Z')||(c>='w'&& c<='z'))
                c=c-22;           //如果是 26 个字母中最后 4 个字母之一就使 c-22
            else
                c=c+4;           //如果是前面 22 个字母之一，就使 c 加 4，
                                //即变成其后第 4 个字母
        }
        printf("%c",c);          //输出已改变的字符
        c=getchar();            //再输入下一个字符给字符变量 c
    }
    printf("\n");
}
```

4.2、do-while 语句

4.2.1、do-while 语句的标准形式

do-while 语句的一般形式为：

```
do
{
    循环体;
} while (表达式);
```

例如：下面两个语句

```
int i=0;
do
{
    printf("%3d",i++);
} while(i<5);
```

//do-while 结束时的分号不能省略

显示： 0 1 2 3 4

do-while 循环类似于 while 循环。不同之处在于，它们执行循环体与计算表达式的先后顺序不同。从流程图可以看出：**do-while 至少需要执行一次循环体**。一般来说，do-while 无论在工程上，还是在大学考试中，出现的频率都远不及 while 语句和 for 语句。所以，不用深入地去掌握 do-while 语句的写法，只需要能看懂 do-while 语句就足够了。

4.2.2、do-while 语句的典型例题

1. 用 do-while 循环语句编写求 $\sum_{i=1}^{100} i$ 值的循环

```
#include<stdio.h>
void main()
{
    int i;
    int sum;
    sum=0;
    i=1;          //与 while 语句相同，定义 i 初值 1，sum 初值 0。
    do
    {
        sum=sum+i; //执行循环体，进行累加。
        i++;       //每次加完，i 的值加 1。
    }while(i<=100);
    printf("sum=%d\n",sum);
}
```

由上面一段程序可以看出，**无论是否满足执行循环体的条件，对 do-while 语句程序都会执行一次循环体。**

4.3、for 语句

4.3.1、for 语句的标准形式

for 循环是功能上比 while 循环更强的一种循环结构形式。for 循环通常用于构造“初值、终值、步长”型循环，其一般形式为：

```
for (表达式 1; 表达式 2; 表达式 3)
{
    循环体;
}
```

三个表达式起着不同的作用：表达式 1 用于进入表达式前给某些变量赋初值；表达式 2 表明循环的条件；表达式 3 用于循环中修改某些变量的值。

1. **表达式 1、2、3 可以全部或者部分省略，但是“；”永远不能省略。**

- (1) 表达式 1 省略，表示该 for 语句没有赋初值部分，或前面的程序段已为有关变量赋了初值，或确实没有特别的初值；
- (2) 表达式 2 省略，表示循环条件永远为真，可能循环体内有控制转移语句转出 for 语句；
- (3) 表达式 3 省略，表示没有对循环变量的更新，对变量的更新已在循环体内一起完成。

如：for(;;);

2. 空语句也可以作为 for 循环的循环体。例如：for (int i=0;i<10000;i++); 只是起到了延迟的作用。

3. 表达式 1，表达式 2，表达式 3 **都可由包含逗号运算符的多个表达式组成。**

4.3.2、for 语句典型例题

1. 用 for 循环语句编写求 $\sum_{i=1}^{100} i$ 值的程序

```
#include<stdio.h>
void main()
{
    int i;
    int sum;
    for(i=1,sum=0;i<=100;i++)    //sum 可以不在这里赋初值，在 for 语句前赋值也是可以的，
                                   i 初值为 1，每次执行完循环体就加 1。
    { sum+=i; }                  //每次执行循环，sum 的值就加上当前 i 的值。
    printf("sum=%d\n",sum);
}
```

2. 求序列 1!, 2!, 3!.....的前 8 项之和。

分析项与项之间的关系，可以得出规律：第 i+1 项=第 i 项* (i+1);

```
#include<stdio.h>
void main()
{
    int i;
    long sum,t;
    sum=0;
    t=1;                //初始化 sum 为 0，t 为 1
    for(i=1;i<=8;i++)  //初始化 i=1，只要 i<=8，就执行循环体
    {
        sum+=t;
        t*=(i+1);      //每次循环 sum 都加上当前项数的阶乘，然后 t 乘上 i+1
    }
    printf("sum=%ld",sum);
}
```

3. 用 $\sin x = x - x^3/3! + x^5/5! - x^7/7! + \dots$ 计算 $\sin x$ 的近似值,直到最后一项的绝对值小于 $1e-6$ 为止。

```
#include<stdio.h>
#include<math.h>
void main()
{
    int i;
    float x,sum,a,b;    //sum 代表和，a 为分子，b 为分母
    char s;
    printf("please input x: ");
    scanf("%f",&x);
    s=1;
    sum=0;
    a=x;                //分子赋初值
    b=1;                //分母赋初值
    for(i=1;a/b>=1e-6;i++)
    { sum=sum+s*a/b;    //累加一项
      a=a*x*x;          //求下一项的分子
```

```

        b=b*(2*i)*(2*i+1);//求下一项的分母
        s*=-1;
    }
    printf("sum=%f\n",sum);
}

```

4. 求 100~200 间的全部素数(素数: 质数(prime number)又称素数, 有无限个。一个大于 1 的自然数, 除了 1 和它本身外, 不能被其他自然数整除, 换句话说就是该数除了 1 和它本身以外不再其他的因数;否则称为合数。

```

#include<stdio.h>
#include<math.h>
int main()
{
    int n,k,i,m=0;    //定义变量 n, k, i, m

    for(n=101;n<=200;n=n+2) //只需要在 100~200 之间所有的奇数中找就行

    {
        for(i=2;i<n;i++)
            if(n%i==0)        //只要找到能被整除的数就跳出循环
                break;
        if(i>=n)
            printf("%d\t",n);    //输出无法被整除的数, 即素数。
    }
}

```

以上的程序是一个典型的找素数的模型, 任何找素数的编程题目都可以套用这个模型。

5. 求两个整数的最大公约数

```

#include<stdio.h>
void main()
{
    int i;
    int a,b;
    scanf("%d%d",&a,&b);
    for(i=a<b?a:b;i>0;i--)    //初值为 a, b 中最小值
    {
        if(a%i==0&&b%i==0)    //如果 a, b 都能被 i 整除, 那么 i 就是 a, b 的公约数
        {
            printf("%d",i);
            break;            //这里一定要用 break 跳出循环, 否则就会持续输出值
        }
    }
}

```

6. 求 100~999 之间所有的水仙花数 (所谓水仙花数就是说数的百位十位和个位数的立方和等于它本身)

```

#include<stdio.h>
void main()
{
    int i,a,b,c;
    for(i=100;i<=999;i++)

```

```

    {
        a=i/100;           //求百位数
        b=(i-a*100)/10;    //求十位数
        c=i%10;           //求个位数
        if(a*a*a+b*b*b+c*c*c==i) //判断百位、十位和个位数的立方和是否等于它本身
            printf("%6d",i);
    }
}

```

4.4、辅助控制语句

4.4.1、break 语句

1. 语句一般形式： `break;`

功能：（1）跳出所在的多分支 `switch` 语句；

（2）跳出所在的 `while`、`do-while`、`for` 循环语句（提前结束循环）

2. `break` 语句一个作用是使流程跳出 `switch` 结构，继续执行下面的语句。还有就是从循环体内跳出循环体，如果是这种用法，**break 只能使程序跳出一层循环。break 语句不能用于循环体和 switch 语句之外的任何其他语句**

下面，举一个简单的例子，如上一节的例 5：求两个整数的最大公约数

```

#include<stdio.h>
void main()
{
    int i;
    int a,b;
    scanf("%d%d",&a,&b);
    for(i=a<b?a:b;i>0;i--)
    {
        if(a%i==0&&b%i==0)
        {
            printf("%d",i);
            break;
        }
    }
}

```

由题意，只需要输出两个整数的最大公约数，而不需要所有的公约数，因此，最后的结果应该是一个数，而不是若干个数字。所以，如果程序没有 `break` 来跳出循环，那么将会继续循环，直到 `i=0` 为止，而在这个过程中，输出的肯定不会是只有最大公约数一个数了。`break` 对循环体的重要性就很明显了。

4.4.2、continue 语句

`continue` 语句用来结束本次循环，即跳过循环体尚未执行的语句，接着进行下一次执行循环的判定。

`continue` 语句和 `break` 语句的区别是：**continue 语句只是结束本次循环，而不是终止整个循环**，而 `break` 是结束整个循环过程。

例如：把 0~100 之间能被 5 整除的数输出。程序如下：

```
void main()
```

```
void main()
```

```

{
    int n;
    for(n=0;n<=100;n++)
    {
        if(n%5!=0)
            continue;
        printf("%d\t",n);
    }
}

{
    int n;
    for(n=0;n<=100;n++)
    {
        if(n%5==0)

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

上面两段程序都能完成所需要的功能，但是却是从正反两个方面来考虑的，所以，在适当的场合合理地使用 continue，可以提升整个程序的效率。

4.4.3、goto 语句

goto 语句的功能是把程序控制转移到标号指定的语句处。因此，**goto 语句常用于退出多重循环**，其一般形式为

goto 标号;

.....

标号: 语句;

例如:

```

goto state;
string s1,s2;    //被 goto 跳过
int sum=2;       //被 goto 跳过
...
state:
...
    
```

goto 语句一直是 C/C++中最有争议的语句，因此在考试中基本不会出现，作为编者，个人觉得 **goto 语句要慎用**。

【精选习题】

答案 P113

基础篇

1. C 语言中，循环控制语句 while 和 do-while 的主要区别是()

- A) do-while 的循环体至少无条件执行一次
- B)while 的循环控制条件比 do-while 的循环控制条件更严格
- C)do-while 允许从外部转到循环体内
- D)do-while 的循环体不能是复合语句

2. 假设 a 和 b 都是整型变量，执行以下语句后，a 和 b 的值是()

```
for (a=b=0;a<4&&++b;a++) b--;
```

- A)3 和 0
- B)3 和 1
- C)4 和 1
- D)4 和 0

3. 下面程序段中，内循环体的总执行次数是()

```
for(i=5;i;i--)
```

```
for(j=0;j<4;j++)
```

```
{    ... }
```

A) 20 次 B) 24 次 C) 25 次 D) 30 次

4. 下面程序段的运行结果是()

```
int x,y;
```

```
x=y=0;
```

```
while(x<15)
```

```
    y++,x+=++y;
```

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

A) 20,7 B) 6,12 C)20,8 D)8,20

5. 若设 m, n, k 均为 int 型变量, 执行完 for(m=0,n=10;m<=n;m++,n--)k=m+n;循环后, k 的值为()

A)7 B)8 C)9 D)10

6. 语句 while(!e==0);什么时候将会陷入死循环: ()

A)e==0 B)e!=0 C)e!=1 D)e==1

7. 下列程序段运行后 i 的值是: ()

```
int i=11;
```

```
switch(i++)
```

```
{    case 10:i++; break;
```

```
    case 11: i++;
```

```
    case 12:i++; break;
```

```
    default: i++; }
```

A)11 B)13 C)12 D)14

8. 写出下面程序执行后的结果:

```
#include <stdio.h>
```

```
void main()
```

```
{    int i,j;
```

```
    for(i=0;i<5;i++)
```

```
    { for(j=1;j<10;j++)
```

```
        if(j==5)
```

```
            break;
```

```
        if(i<2)
```

```
            continue;
```

```
        if(i>2)
```

```
            break;
```

```
        printf("j=%d",j);
```

```
    }
```

```
    printf("i=%d\n",i);
```

```
}
```

结果为: _____

9. 设变量 x,y 为 int 类型, 则循环语句 for (x=4,y=0;(y!=10) && (x>0);x--,y+=2);执行的循环次数为

()

- A) 无限多次 B) 4 次 C) 3 次 D) 2 次

提高篇

1. 若 i 和 k 都是 int 类型变量, 有以下 for 语句 for (i=0,k=-1;k=1;k++) printf("*****\n");下面关于语句执行情况的叙述中正确的是 ()

- A) 循环体执行两次 B) 循环体执行一次 C) 循环体一次也不执行 D) 构成无限循环

2. 下列描述中, 正确的一条是 ()

- A) if(表达式)语句中, 表达式的类型只限于逻辑表达式
B) 语句“goto 12;”是合法的
C) for (;;) 语句相当于 while(1)语句
D) break 语句可用于程序的任何地方, 以终止程序的执行

3. 下面程序的运行结果是 ()

```
#include<stdio.h>
```

```
void main()
```

```
{ int n=1;  
  while(n--);  
  printf("n=%d\n",n); }
```

- A)n=0 B)while 构成无限循环 C) n=1 D)n=-1

4. 以下程序段中的变量已正确定义:

```
for(i=0;i<4;i++)  
    for(k=1;k<3;k++);  
    printf("*");
```

程序的运行结果是 ()

- A)** B)**** C)* D)*****

5. 有以下程序:

```
#include<stdio.h>
```

```
main()
```

```
{ int x=8;  
  for(;x>0;x--)  
  { if(x%3)  
    { printf("%d",x--);  
      continue; }  
    printf("%d",--x);  
  }  
}
```

程序的运行结果是 ()

- A) 7,4,2, B) 8,7,5,2, C) 9,7,6,4, D) 8,5,4,2,

6. 古代《张丘建算经》中有一道百鸡问题: 鸡翁一, 钱值五; 母鸡一, 钱值三; 鸡雏三, 钱值一。问何以百钱买百鸡? (编一道程序用以解决这个问题)。

7. 编一道程序, 要求对输入进行确认, 输入不是数字就要重新输入, 直到输入量是 1 个数字为止。

8. 编一程序实现一个最简单计算器的功能。如果输入 3+5 回车, 则显示 3+5=8; 输入错误就退出(输入的不是加减乘除运算就算错)。