本章知识点小结

内容	基本描述	备注

for 语句	**for** (表达式 1; 表达式 2; 表达式 3)
	{
	循环体语句
	}
		用于实现当型循环控制结构。在循环顶部进行循环条件测试，如果循环条件第一次测试就为假，则循环体一次也不执行。适合于循环次数已知、计数控制的循环。

while 语句	**while** (表达式)
	{
	循环体语句
	}
		用于实现当型循环控制结构。适合于循环次数未知、条件控制的循环。

do-while 语句	**do**{ 循环体语句 }**while**(表达式);	用于实现直到型循环控制结构。在循环底部进行循环条件测试，循环至少执行一次。适合于循环次数未知、条件控制的循环。尤其适合于构造菜单子程序，因为菜单子程序至少要执行一次，用户输入有效响应时，菜单子程序采取相应动作；输入无效响应时，则提示重新输入。
break 语句	用于退出 switch 或一层循环结构。	用于流程控制。
continue 语句	用于结束本次循环、继续执行下一次循环。	用于流程控制。
goto 语句	无条件转移到标号所标识的语句处去执行。	用于流程控制。当程序需要退出多重循环时，用 goto 语句比用 break 语句更直接方便
标准函数 exit()	**exit(code);**	作用是终止整个程序的执行。当 code 值为 0 时，表示程序正常退出；当 code 值为非 0 值时，表示程序出现某种错误后退出。
逗号运算符和逗号表达式	表达式 1, 表达式 2, …, 表达式 n	优先级最低，具有左结合性。通常，使用逗号表达式的目的并非要得到和使用整个逗号表达式的值，而仅仅是顺序计算各个表达式的值。

累加算法设计	for (sum=0,i=0; i<n; i++)
	{
	sum = sum + 通项;
	}
		累加和变量的初值通常设为 0。当累加的前后项无关时，需单独计算通项。而当累加的前后项之间有关时，可以根据后项与前项之间的关系，利用前项计算后项。
		与累加算法不同的是，累乘积变量的初值通常设为 1。

本章常见错误小结

常见错误实例	常见错误描述	错误类型

while (i <= n)
{
sum = sum + i;
i++;
}
	在循环开始前，未将计数器变量、累加求和变量或者累乘求积变量初始化，导致运行结果出现乱码。	运行时错误

while (i <= n)
sum = sum + i;
i++;
	在界定 while 和 for 语句后面的复合语句时，忘记了花括号。	运行时错误

for (i=1; i<=n; i++) ;
{
sum = sum + i;
}
	在紧跟 for 语句表达式圆括号外之后写了一个分号。位于 for 语句后面的分号使循环体变成了空语句，即循环体不执行任何操作。	运行时错误

while (i <= n) ;
{
sum = sum + i;
i++;
}
	在紧跟 while 语句条件表达式的圆括号外之后写了一个分号。位于 while 语句后面的分号使循环体变成了空语句，在第一次执行循环循环控制条件为真时，将引起死循环。	运行时错误

while (n < 100)
{
printf("n = %d", n);
}
	在 while 循环语句的循环体中，没有改变循环控制条件的操作，在第一次执行循环循环控制条件为真时，将导致死循环。	运行时错误

do{
  sum = sum + i;
  i++;
}while (i <= n)
	do-while 语句的 while 后面忘记加分号。	编译错误

for (i=1, i<=n, i++)
{
  p = p * i;
}
	用逗号分隔 for 语句圆括号中的三个表达式。	编译错误

	嵌套循环中的左花括号{与右花括号}不配对。	编译错误
	嵌套循环的内层和外层的循环控制变量同名。	运行时错误