

C语言：循环

循环中的while, for, do-while

while循环

语法形式和与if的区别

上一篇文章我们谈到了if语句：

```
if(表达式)
    语句
```

而while语句：

```
while(表达式)
    语句//如果有多条语句需要用{}括起来
```

就是说如果if后面的表达式为真那么就会执行后面的语句，while后面的表达式为真的话也会执行语句，当执行完语句后会返回再一次判断表达式的真假。它们的区别是while语句是可以进行循环的操作的。

```
int main
{
    while(1)
    {
        printf("肉末茄子报吃\n");
    }
    return 0;
}
```

也就是说如果while后面的表达式一直为真（==1）就会一直执行语句“肉末茄子报吃”

练习

下面如果我想在屏幕上打印数字1~10就可以用while：

```
int main()
{
    int i = 1;
    while (i <= 10)
    {
        printf("%d ", i);
        ++i;
    }
    return 0;
}
```

for循环

语法形式

```
for(表达式1;表达式2;表达式3)
    语句//如果有多条语句可以用{}括起来
```

表达式1：用于循环变量的初始化

表达式2：用于循环结束条件的判断

表达式3：用于循环变量的调整

在for语句中最先执行表达式1，之后是表达式2。如果表达式2为假（==0）则循环结束，如果表达式2为真（==1）就执行下面的语句，执行完后，再执行表达式3（调整循环变量）之后判断表达式2的真假决定是否再进行下一次循环。

在整个过程中，表达式1初始化部分只被执行1次，剩下的就是表达式2、循环语句、表达式3在循环。

练习

还是打印1~10的数字，用for循环就应该这样操作：

```
int main()
{
    for (int i = 1; i <= 10; i++)
    {
        printf("%d ", i);
    }
}
```

```
    return 0;  
}
```

对比while与for

```
int main()  
{  
    int i = 1; 1.初始化  
    while (i <= 10) 2.判断  
    {  
        printf("%d ", i); 3.执行语句  
        ++i; 4.调整  
    }  
    return 0;  
}
```

CSDN @数某

编辑

```
int main()  
{  
    1.初始化    2.判断  
    for (int i = 1; i <= 10; i++) 4.调整  
    {  
        printf("%d ", i); 3.执行语句  
    }  
  
    return 0;  
}
```

CSDN @数某

编辑

由于for循环的初始化，判断，调整相对与while来说较为集中，便于维护所以在形式上for要优于while。

do-while循环

语法形式

```
do  
    语句  
while(表达式);
```

for和while是先判断再进循环，而do-while是先进入循环体执行语句后再进行判断，表达式为真(==1)则进行下一次循环，为假则结束循环。(注意分号)

do-while循环的一个特殊的点是它至少要执行一次循环语句。

练习

那么还是打印数字1~10。

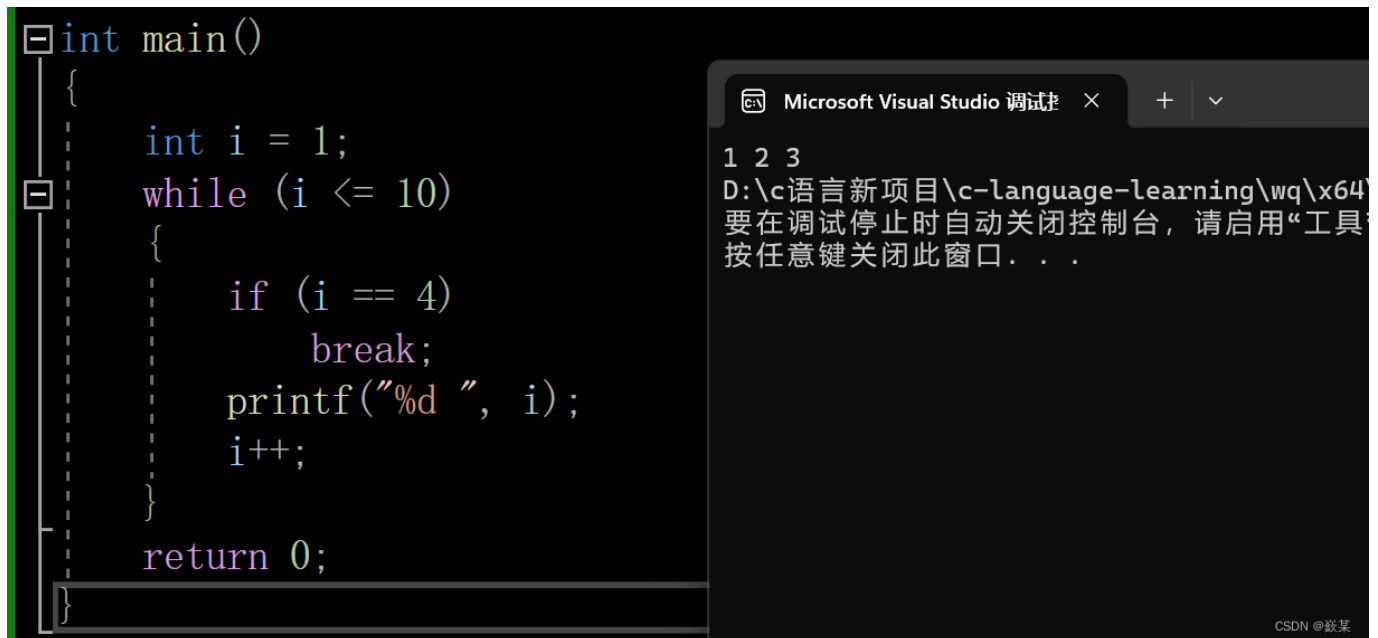
```
int main()
{
    int i = 1;
    do
    {
        printf("%d ", i);
        i += 1;
    } while (i<=10);

    return 0;
}
```

break和continue

while循环中的break和continue

```
int main()
{
    int i = 1;
    while (i <= 10)
    {
        if (i == 4)
            break; //当i等于4时，执行break，跳出循环
        printf("%d ", i);
        i++;
    }
    return 0;
}
```



The screenshot shows the Microsoft Visual Studio IDE. On the left, the 'int main()' function is being edited. The code is as follows:

```
int main()
{
    int i = 1;
    while (i <= 10)
    {
        if (i == 4)
            break;
        printf("%d ", i);
        i++;
    }
    return 0;
}
```

On the right, the 'Microsoft Visual Studio 调试' (Debug) window is open, showing the output of the program:

```
1 2 3
D:\c语言新项目\c-language-learning\wq\x64
要在调试停止时自动关闭控制台，请启用“工具
按任意键关闭此窗口...
```

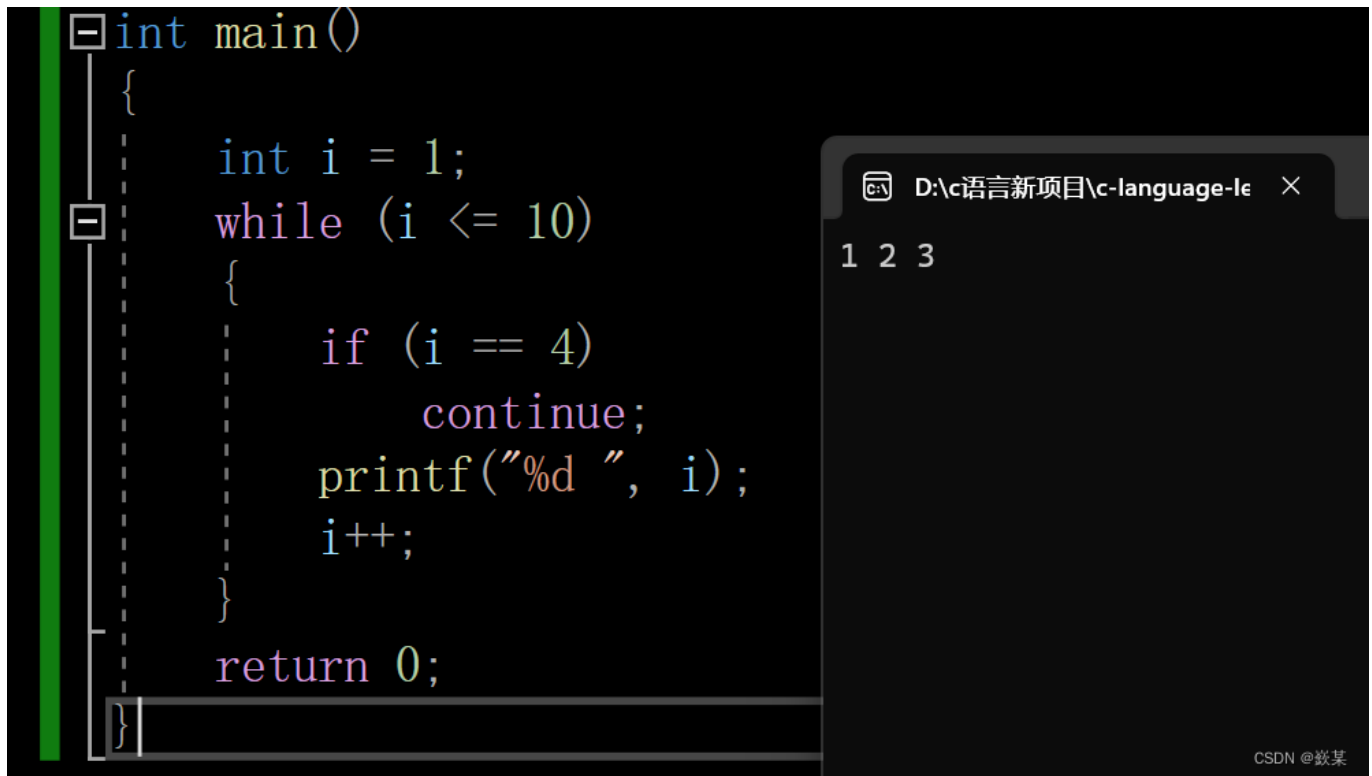
The output shows the numbers 1, 2, and 3 printed on the first line, followed by a new line and the file path. The second line is a message about closing the console window.

编辑

代码的执行结果是在屏幕上打印1 2 3

下面是continue

```
int main()
{
    int i = 1;
    while (i <= 10)
    {
        if (i == 4)
            continue; //当i等于4时执行continue跳过continue后面的语句直接进入下一次循
环。
        //由于跳过了i的自加，i一直为4，程序进入死循环。
        printf("%d ", i);
        i++;
    }
    return 0;
}
```



```
int main()
{
    int i = 1;
    while (i <= 10)
    {
        if (i == 4)
            continue;
        printf("%d ", i);
        i++;
    }
    return 0;
}
```

D:\c语言新项目\c-language-le ×

1 2 3

CSDN @数某

编辑

虽然这个代码的运行结果也是1 2 3 但是程序一直没有跳出循环，大家可以自行比较这两段代码的运行结果的差异。

for循环中的break和continue

```
int main()
{
    for (int i = 1; i <= 10; i++)
    {
        if (i == 4)
            break;
        printf("肉末茄子报吃\n");
    }
    return 0;
} //结果是打印三次“肉末茄子报吃”
```

```
int main()
{
    for (int i = 1; i <= 10; i++)
    {
        if (i == 4)
            continue;
        printf("%d ", i);
    }
}
```

```
    }  
    return 0;  
}
```

而换成continue的打印结果是 1 2 3 5 6 7 8 9 10，因为这里直接跳过了printf语句，直接到了自加环节。

do-while中的break和continue和while循环中的几乎一模一样所以就不过多叙述。

循环的嵌套

这里有一个题目：

找出100~200之间的素数，并打印在屏幕上。

思路：

1. 要从100~200之间找出素数，首先得有100~200之间的数，这里可以使用循环解决。
2. 假设要判断i是否为素数，需要拿2~i-1之间的数字去试除i，需要产生2~i-1之间的数字，也可以使用循环解决。
3. 如果2~i-1之间有数字能整除i，则i不是素数，如果都不能整除，则i是素数。

请自行思考，答案将在下一篇文章中给出。

goto语句

goto语句语句可以实现在**同一个函数**内跳转到设置好的标号处。

例如：

```
int main()  
{  
    printf("haha\n");  
    goto next;  
    printf("hehe\n");  
next:  
    printf("跳过了hehe的打印\n");  
}
```

要注意的是如果在一个函数中大量的使用goto语句会让逻辑非常的混乱，但是在多层循环中想要快速跳出就可以使用goto：

```
for()  
{  
    for()  
}
```

```
{
    for()
    {
        for()
        {
            goto next;
        }
    }
}
next:
```

非常好用！