# 第五周（2024-12-6）

---

## 一、If ... Else 条件语句

### 条件和 If 语句

C语言 支持数学中的常见逻辑条件:

- 小于:a < b
- 小于或等于:a <= b
- 大于:a > b
- 大于或等于:a >= b
- 等于 a == b
- 不等于:a != b

C 有以下条件语句:

- 使用`if`指定要执行的代码块，如果指定条件为真（True）
- 使用`else`指定要执行的代码块，如果相同条件为假(False)
- 如果第一个条件为假，则使用 `else if` 指定要测试的新条件
- 使用 `switch` 指定要执行的许多替代代码块

---

### if 语句

使用 `if` 语句指定在条件为 `true` 时要执行的 C 代码块 .

### 语法

```
if (*condition*) {
 *// 条件为 true 时执行的代码块*
}
```

注意，`if` 是小写字母。 大写字母（If 或 IF）会产生错误。

在下面的例子中，我们测试两个值来判断 20 是否大于 18。如果条件为 `true`，打印一些文本:

## 实例

```
if (20 > 18) {
 printf("20 is greater than 18");
}
```

我们也可以测试变量:

## 实例

```
int x = 20;
int y = 18;
if (x > y) {
 printf("x 大于 y");
}
```

### 示例说明

在上面的示例中，我们使用两个变量 **x** 和 **y** 来测试 x 是否大于 y（使用 `>` 运算符）。 由于 x 是 20，y 是 18，并且我们知道 20 大于 18，所以我们在屏幕上打印"x 大于 y"。

---

## else 语句

使用 `else` 语句指定在条件为 `false` 时要执行的代码块。

## 语法

```
if (*condition*) {
 *// 条件为 true 时执行的代码块*
} else {
 *// 条件为 false 时要执行的代码块*
}
```

## 实例

```
int time = 20;
if (time < 18) {
 printf("Good day.");
} else {
 printf("Good evening.");
}
// 输出 "Good evening."
```

### 示例说明

在上面的例子中，时间（20）大于18，所以条件是`false`。 因此，我们转到 `else` 条件并打印到屏幕"Good evening"。 如果时间小于 18，程序将打印"Good day"。

---

## else if 语句

如果第一个条件为 `false`，则使用 `else if` 语句指定新条件。。 p>

## 语法

```
if (*condition1*) {
 *// 如果条件 1 为 true，则要执行的代码块*
} else if (*condition2*) {
 *// 如果条件 1 为 false 且条件 2 为 true，则要执行的代码块*
} else {
 *// 如果条件1为 false 且条件2为 false 时要执行的代码块*
}
```

### 实例

```
int time = 22;
if (time < 10) {
 printf("Good morning.");
} else if (time < 20) {
 printf("Good day.");
} else {
 printf("Good evening.");
}
// 输出 "Good evening."
```

### 示例说明

在上面的例子中，时间 (22) 大于 10，所以 **第一个条件** 是 `false`。 `else if` 语句中的下一个条件也是 `false`，所以我们继续进行 `else` 条件，因为`condition1` 和 `condition2` 都是 `false` - 并打印到屏幕上"Good evening"。

但是，如果时间是 14 点，我们的程序会打印"Good day"。

---

## 二、Switch 语句

`switch` 语句选择要执行的许多代码块之一：

### 语法

```
switch(*expression*) {
 case x:
  *// code block*
  break;
 case y:
  *// code block*
  break;
 default:
  *// code block*
}
```

它的工作原理：

- `switch` 表达式只计算一次
- 将表达式的值与每个`case`的值进行比较
- 如果匹配，则执行关联的代码块
- `break` 语句跳出 switch 块并停止执行
- `default` 语句是可选的，指定在没有大小写匹配时运行的一些代码

以下示例使用工作日编号来计算工作日名称：

## 实例

```
int day = 4;

switch (day) {
 case 1:
  printf("Monday");
  break;
 case 2:
  printf("Tuesday");
  break;
 case 3:
  printf("Wednesday");
  break;
 case 4:
  printf("Thursday");
  break;
 case 5:
  printf("Friday");
  break;
 case 6:
  printf("Saturday");
  break;
 case 7:
  printf("Sunday");
  break;
}

// 输出 "Thursday" (day 4)
```

## break 关键字

当 C 到达 `break` 关键字时，它会跳出 switch 块。

这将停止块内更多代码和案例测试的执行。

找到匹配项并完成工作后，就该休息一下了。 无需进行更多测试。

中断可以节省大量执行时间，因为它"忽略" 执行 switch 块中的所有其余代码。

---

## default 默认关键字

`default` 关键字指定在没有大小写匹配时要运行的一些代码:

### 实例

```
int day = 4;

switch (day) {
 case 6:
  printf("Today is Saturday");
  break;
 case 7:
  printf("Today is Sunday");
  break;
 default:
  printf("Looking forward to the Weekend");
}

// 输出 "Looking forward to the Weekend"
```

注意: **default** 关键字必须作为 **switch** 中的最后一条语句，并且不需要 **break**。

---

# 三、While 循环

## 循环

只要达到指定条件，循环就可以执行一段代码。

循环很方便，因为它们可以节省时间、减少错误并且使代码更具可读性。

---

## While 循环

只要指定条件为 `true`，`while` 循环就会遍历一段代码:

## 语法

```
while (*condition*) {
 *// 要执行的代码块*
}
```

在下面的示例中，只要变量 (`i`) 小于 5，循环中的代码就会反复运行:

## 实例

```
int i = 0;

while (i < 5) {
 printf("%d\n", i);
 i++;
}
```

**注意:**不要忘记增加条件中使用的变量（`i++`），否则循环永远不会结束！

---

## Do...While 循环

`do/while` 循环是 `while` 循环的变体。 该循环将执行代码块一次，在检查条件是否为真之前，只要条件为真，它将重复循环。

### 语法

```
do {
 *// 要执行的代码块
*}
while (*condition*);
```

下面的示例使用 `do/while` 循环。 循环总是会至少执行一次，即使条件为假，因为代码块是在条件被测试之前执行的:

### 实例

```
int i = 0;

do {
 printf("%d\n", i);
 i++;
}
while (i < 5);
```

**注意：不要忘记增加条件中使用的变量，否则循环永远不会结束！**

---

## 四、For循环

当我们确切知道要循环一段代码的次数时，请使用 `for` 循环而不是 `while` 循环:

## 语法

```
for (*statement 1*; *statement 2*; *statement 3*) {
 *// 要执行的代码块*
}
```

**语句 1** 在代码块执行之前执行（一次）。

**语句2** 定义了执行代码块的条件。

**语句 3** 在代码块执行后（每次）执行。

下面的例子将打印数字 0 到 4:

## 实例

```
int i;

for (i = 0; i < 5; i++) {
 printf("%d\n", i);
}
```

### 示例说明

语句 1 在循环开始之前设置一个变量 (int i = 0)。

语句 2 定义了循环运行的条件（i 必须小于 5）。 如果条件为真，则循环重新开始，如果为假，则循环结束。

每次执行循环中的代码块时，语句 3 都会增加一个值 (i++)。

---

## 另一个例子

此示例将仅打印 0 到 10 之间的偶数:

### 实例

```
for (i = 0; i <= 10; i = i + 2) {
 printf("%d\n", i);
}
```

---

## 五、Break 和 Continue 语句

### Break

`break` 语句， 它用于"跳出"`switch` 语句。

`break` 语句也可用于跳出循环。

本例在`i`等于4时跳出循环:

### 实例

```
int i;

for (i = 0; i < 10; i++) {
 if (i == 4) {
  break;
 }
 printf("%d\n", i);
}
```

---

### Continue

如果指定条件发生，`continue` 语句会中断一次迭代（循环中），并继续循环中的下一次迭代。

这个例子跳过了 4 的值:

## 实例

```
int i;

for (i = 0; i < 10; i++) {
 if (i == 4) {
  continue;
 }
 printf("%d\n", i);
}
```

---

## 在 While 循环中中断并继续

还可以在 while 循环中使用 `break` 和 `continue`:

## Break 实例

```
int i = 0;

while (i < 10) {
 if (i == 4) {
  break;
 }
 printf("%d\n", i);
 i++;
}
```

---

Continue 实例

```
int i = 0;

while (i < 10) {
 i++;
 if (i == 4) {
  continue;
 }
 printf("%d\n", i);
}
```

---

## 六、案例

if-else语句：

```
#include <stdio.h>

int main() {
    int num;
    printf("请输入一个整数: ");
    scanf("%d", &num);

    if (num > 0) {
        printf("你输入的是一个正数。\n");
    } else if (num < 0) {
        printf("你输入的是一个负数。\n");
    } else {
        printf("你输入的是零。\n");
    }

    return 0;
}
```

switch语句：

```c
#include <stdio.h>

int main() {
    int day;
    printf("请输入1到7之间的数字以表示星期几: ");
    scanf("%d", &day);

    switch (day) {
        case 1:
            printf("星期一\n");
            break;
        case 2:
            printf("星期二\n");
            break;
        case 3:
            printf("星期三\n");
            break;
        case 4:
            printf("星期四\n");
            break;
        case 5:
            printf("星期五\n");
            break;
        case 6:
            printf("星期六\n");
            break;
        case 7:
            printf("星期日\n");
            break;
        default:
            printf("无效输入\n");
    }

    return 0;
}
```

while循环：

```c
#include <stdio.h>

int main() {
    int sum = 0, i = 1;

    while (i <= 10) {
        sum += i;
        ++i;
    }

    printf("1到10的累加和是: %d\n", sum);
    return 0;
}
```

do...while循环：

```c
#include <stdio.h>

int main() {
    int num;

    do {
        printf("请输入一个正整数: ");
        scanf("%d", &num);
        if (num <= 0)
            printf("这不是一个正整数，请重试。\n");
    } while (num <= 0);

    printf("你输入了一个正整数: %d\n", num);
    return 0;
}
```

for循环：

```c
#include <stdio.h>

int main() {
    for (int i = 1; i <= 10; ++i) {
        printf("%d 的平方是 %d\n", i, i * i);
    }
    return 0;
}
```

---

## 七、实践作业

### （一）判断某年是否是闰年（闰年：能被4整除且不能被100整除的年份、能被400整除的年份）

```c
#include <stdio.h>

// 定义一个函数来检查给定的年份是否为闰年
int isLeapYear(int year) {
    // 判断是否为闰年的逻辑
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        return 1; // 是闰年返回1
    } else {
        return 0; // 不是闰年返回0
    }
}

int main() {
    int year;

    // 提示用户输入年份
    printf("请输入一个年份: ");
    scanf("%d", &year);

    // 调用isLeapYear函数并根据返回值输出结果
    if (isLeapYear(year)) {
        printf("%d年是闰年。\n", year);
```

```
    } else {
        printf("%d年不是闰年。\n", year);
    }

    return 0;
}
```

## （二）判断某年某月有多少天（注意：闰年二月）

```
#include <stdio.h>

// 判断是否为闰年的函数
int isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

// 获取某年某月的天数的函数
int getDaysInMonth(int year, int month) {
    // 数组存储各个月份的天数（非闰年）
    int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31};

    // 如果是闰年并且询问的是2月，则返回29天
    if (month == 2 && isLeapYear(year)) {
        return 29;
    } else {
        // 否则返回对应月份的天数
        return daysInMonth[month - 1];
    }
}

int main() {
    int year, month;

    // 提示用户输入年份和月份
    printf("请输入年份: ");
    scanf("%d", &year);
    printf("请输入月份: ");
    scanf("%d", &month);
```

```
    // 检查月份的有效性
    if (month < 1 || month > 12) {
        printf("无效的月份。\n");
        return 1; // 返回错误码
    }

    // 调用getDaysInMonth函数并输出结果
    printf("%d年%d月有%d天。\n", year, month, getDaysInMonth(year,
month));

    return 0;
}
```

## （三）计算1+3+5+…+999=?

```
#include <stdio.h>

int main() {
    int sum = 0;
    for (int i = 1; i <= 999; i += 2) {
        sum += i;
    }
    printf("1+3+5+...+999 的和是: %d\n", sum);
    return 0;
}
```