

# 控制流

胡船长

初航我带你，远航靠自己

# 一、本期内容

- 1. 第一节：条件表达式
- 2. 第二节：分支结构
- 3. 第三节：循环结构
- 4. 第四节：被封印的 goto 语句
- 5. 第五节：重新认识 C 语言中的一条语句
- 6. 第六节：分支-实战练习
- 7. 第七节：循环-实战练习

# 一. 条件表达式

# C 语言关系运算符

运算符	说明	例子
<code>==</code>	相等比较	<code>a == b</code>
<code>!=</code>	不等比较	<code>a != b</code>
<code>&lt;、&gt;</code>	小于和大于比较	<code>a &gt; b、 a &lt; b</code>
<code>&lt;=(&lt;=)、=&gt;(&gt;=)</code>	小于等于和大于等于比较	<code>a &lt;= b、 a &gt;= b</code>
<code>!</code>	条件非	<code>!(0)、!(NULL)</code>
<code>&amp;&amp;</code>	逻辑与(并且)	<code>a==b &amp;&amp; a&lt;c</code>
<code>  </code>	逻辑或(或者)	<code>a&gt;=b    !c</code>

# 聊聊：条件表达式

1. 条件表达式的值只有『真』和『假』
2. 其中『真』为 1, 『假』为 0

## 短路原则：

通过前面的表达式就已经可以确定整体表达式的值，就不再计算后面的表达式

# 实现程序：

读入  $a, b$  两个值， $a$  小于  $b$ ，输出  
YES，否则输出 NO

# 聊聊：『真』与『假』

# 聊聊：『真』与『假』

非0即为『真』

## 二. 分支结构

# if-else 语句

## if 语句

有条件地执行代码。

用于仅若某条件为真才执行代码的场合。

### 语法

---

属性说明符序列(可选) **if** ( 表达式 ) 语句真 (1)

---

属性说明符序列(可选) **if** ( 表达式 ) 语句真 **else** 语句假 (2)

# if-else 语句

```
if (表达式) {  
    代码段;  
}
```

```
if (表达式) {  
    代码段1;  
} else {  
    代码段2;  
}
```

```
if (表达式1) {  
    代码段1;  
} else if (表达式2) {  
    代码段2;  
} else {  
    代码段3;  
}
```

# 随堂练习题-1

程序读入一个正整数  $n$ , 代表学生的成绩, 根据分数输出分数档位

$n = 0$ , HEHE

$0 < n < 60$ , FAIL

$60 \leq n < 75$ , MEDIUM

$75 \leq n \leq 100$ , GOOD

# switch-case 语句

## switch 语句

按照整数参数的值执行代码。

在需要按照整数值执行多个分支中的一个或数个之处使用。

### 语法

属性说明符序列(可选) **switch** ( 表达式 ) 语句

属性说明符序列 - (C23) 可选的属性列表，应用到 switch 语句

表达式 - 任何整数类型 ( char 、有符号或无符号整数，或枚举) 表达式

语句 - 任何语句 (典型为复合语句)。允许在 语句 中有 **case:** 和 **default:** 标号，而 **break;** 语句拥有特殊含义。

**case** 常量表达式 : 语句 (1) (C23 前)

属性说明符序列(可选) **case** 常量表达式 : 语句(可选) (1) (C23 起)

**default :** 语句 (2) (C23 前)

属性说明符序列(可选) **default :** 语句(可选) (2) (C23 起)

# switch 语句

```
switch (a) {  
    case v1: 代码块1;  
    case v2: 代码块2;  
    case v3: 代码块3;  
}
```

case 为条件入口，程序进入 case 所对应的代码段，依次执行后续代码，直到遇到 break，或者 switch 结构末尾

# 随堂练习题-2

请使用 switch 结构完成如下任务，程序读入一个整数 n：

如果  $n = 1$ : 输出 one

如果  $n = 2$ : 输出 two

如果  $n = 3$ : 输出 three

否则输出 error

# 随堂练习题-3

请使用 switch 结构完成如下任务，程序读入一个整数 n：

如果  $n = 1$ : 输出 one two three

如果  $n = 2$ : 输出 two three

如果  $n = 3$ : 输出 three

否则输出 error



# #113. 一个月有多少天

描述

提交

自定义测试

管理

题解视频

上一题

下一题

统计

## 题目描述

给出一个年份  $y$  和月份  $m$ , 求  $y$  年  $m$  月有多少天。

## 输入

输入两个整数  $y, m$  表示年份和月份 ( $1000 \leq y \leq 3000, 1 \leq m \leq 12$ )

## 输出

输出  $y$  年  $m$  月的天数。

## 样例输入

```
2000 2
```

## 样例输出

```
29
```

# 附录1：分支预测

# 附录1：回文整数

## 9. 回文数

提示



简单



2.6K



相关企业

给你一个整数 `x`，如果 `x` 是一个回文整数，返回 `true`；否则，返回 `false`。

回文数是指正序（从左向右）和倒序（从右向左）读都是一样的整数。

- 例如，`121` 是回文，而 `123` 不是。

# 附录1：回文整数

```
1 bool isPalindrome(int x) {  
2     if (__builtin_expect(!(x < 0), 0)) return false;  
3     int y = 0, z = x;  
4     while (x) {  
5         y = y * 10 + x % 10;  
6         x /= 10;  
7     }  
8     return z == y;  
9 }
```

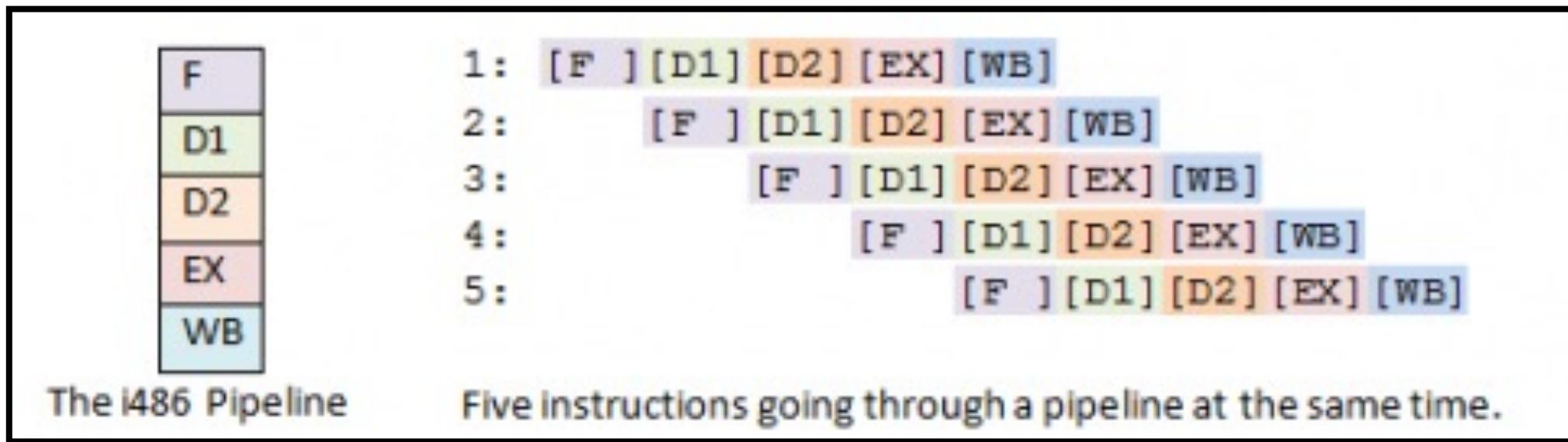
# 附录1：回文整数

```
#define likely(x)      __builtin_expect(!!(x), 1)
#define unlikely(x)     __builtin_expect(!!(x), 0)
```

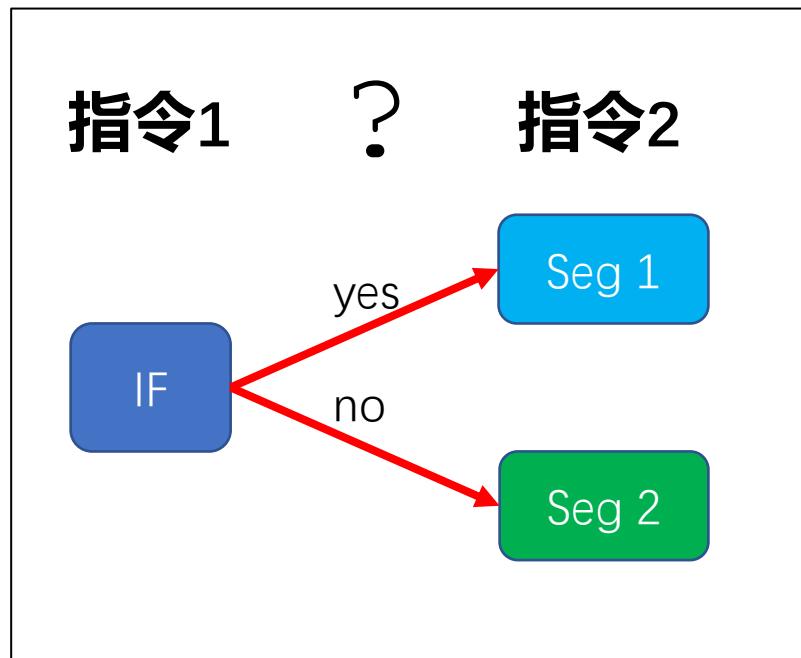
// likely 代表 x 经常成立

// unlikely 代表 x 不经常成立

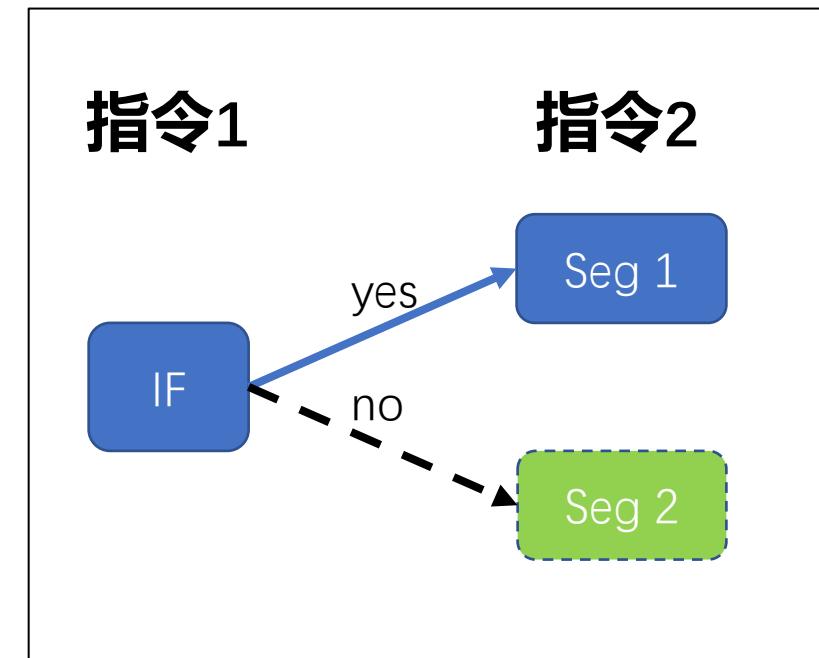
# 附录1：回文整数



# 附录1：回文整数



分支预测



# 附录1：回文整数

```
#define likely(x)      __builtin_expect(!!(x), 1)
#define unlikely(x)     __builtin_expect(!!(x), 0)
```

// likely 代表 x 经常成立，加载条件分支内部的代码  
// unlikely 代表 x 不经常成立，加载条件分支外部的代码

# 附录1：回文整数

`__builtin_ffs(x)`: 返回x中最后一个为1的位是从后向前的第几位

`__builtin_popcount(x)`: x中1的个数

`__builtin_ctz(x)`: x末尾0的个数。x=0时结果未定义

`__builtin_clz(x)`: x前导0的个数。x=0时结果未定义

`__builtin_prefetch (const void *addr, ...)`: 对数据手工预取的方法

`__builtin_types_compatible_p(type1, type2)`: 判断type1和type2是否是相同的数据类型

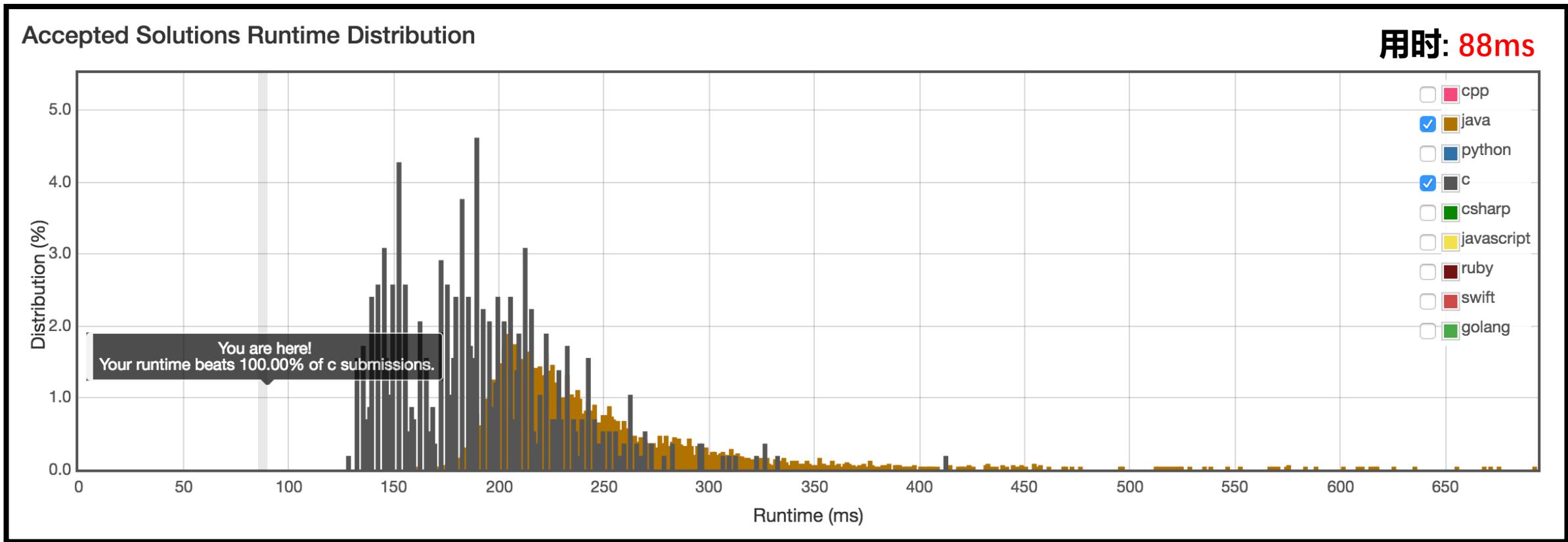
`__builtin_expect (long exp, long c)`: 用来引导gcc进行条件分支预测

`__builtin_constant_p (exp)`: 判断exp是否在编译时就可以确定其为常量

`__builtin_parity(x)`: x中1的奇偶性

`__builtin_return_address(n)`: 当前函数的第n级调用者的地址

# 附录1：回文整数



# 附录1：回文整数



# 三. 循环结构

# while 语句

## while 循环

重复执行 语句，直到 表达式 的值变得比较等于零。此测试在每次迭代前发生。

### 语法

属性说明符序列(可选) **while** ( 表达式 ) 语句

**表达式** - 任何标量类型表达式。在每次迭代前求值此表达式，而且若它与零比较相等，则退出循环。

**语句** - 任何语句，常为一条复合语句，它被用作循环体

**属性说明符序列** - (C23)可选的属性列表，应用到循环语句

# do-while 语句

## do-while 循环

重复执行 语句，直到条件 表达式 的值变为假。此测试在每次迭代后发生。

### 语法

属性说明符序列(可选) **do** 语句 **while** ( 表达式 ) ;

表达式 - 任何标量类型的表达式。此表达式在每次迭代后求值，而且若它与零比较相等，则退出循环。

语句 - 任何语句，常为一条复合语句，作为循环体

属性说明符序列 - (C23)可选的属性列表，应用到循环语句

# while 语句

```
while (表达式) {  
    代码块;  
}
```

每当表达式为真时，代码块就会被执行1次。

```
do {  
    代码块;  
} while (表达式);
```

每当代码段执行1次，就会判断一次表达式是否为真。

# 随堂练习题-4

请使用 `while` 循环实现程序，反复读入整数  $x$ ，并且输出  $2^x$  的值，直到文件结束 (EOF)。

# for 语句

## for 循环

执行循环。

用作 [while 循环](#) 的简短等价版本。

### 语法

---

属性说明符序列(C23 起)(可选) **for** ( 初始子句 ; 条件表达式 ; 迭代表达式 ) 循环语句

---

# for 语句

```
for (初始化; 循环条件; 执行后操作) {  
    代码块;  
}
```

- Step1: **初始化**
- Step2: **循环条件判断**
- Step3: **执行代码块**
- Step4: **执行后操作**
- Step5: 跳转到 Step2

# 随堂练习题-5

请使用 for 循环实现程序，输出斐波那契数列前20项的值。

$$F[n] = F[n-1] + F[n-2], \quad F[0]=1, F[1]=1$$

# 『break』与『continue』

## break 语句

1. 跳出 switch-case 结构
2. 跳出最近的一层循环

## continue 语句

1. 结束本次循环，继续下一次循环

# 四. 被封印的 goto 语句

# goto 语句

## goto 语句

将控制无条件转移到所欲位置。

在无法用约定的构造将控制转移到所欲位置时使用。

### 语法

---

属性说明符序列(可选) **goto** 标号 ;

---

标号 - goto 语句的目标**标号**

属性说明符序列 - (C23)可选的**属性**列表，应用到 goto 语句

# goto 语句

```
goto lab_1;  
    代码段1;  
lab_1:  
    代码段2;
```

跳转到 **lab\_1** 标记初，**代码段1**  
被跳过，直接执行**代码段2**

# 随堂练习题-6

用 goto 语句模拟 if-else 语句。

# 随堂练习题-7

用 goto 语句模拟 while 语句。

# 随堂练习题-8

用 goto 语句模拟 for 语句。

# 五. 重新认识 C 语言中的一条语句

# 五. 重新认识 C 语言中的一条语句

语句有五种类型：

1. 复合语句
2. 表达式语句
3. 选择(分支)语句
4. 循环语句
5. 跳转语句

# 五. 重新认识 C 语言中的一条语句

语句有五种类型：

1. 复合语句

{ statement; }

2. 表达式语句

expression;

3. 选择(分支)语句

if-else

4. 循环语句

while/do-while/for

5. 跳转语句

break/continue/return/goto

# 六、分支-课后实战题

- 1.HZOJ-103: 整除问题
- 2.HZOJ-107: 七的奇倍数
- 3.HZOJ-108: 求面积的问题
- 4.HZOJ-114: 他的名字
- 5.HZOJ-118: 生肖
- 6.HZOJ-120: 日期合法性

# 七、循环-课后实战题

1.HZOJ-128: n 个数的平均数

2.HZOJ-130: 计算复利2

3.HZOJ-136: N 以内7的倍数

4.HZOJ-137: 字母三角形

5.HZOJ-139: 输出 A 字菱形

6.HZOJ-140: 输出字母菱形

7.HZOJ-141: 输出字母沙漏

8.HZOJ-142: 五位质数回文数

不要考虑太多，坚持看完，  
你就已经超过了95%的人。

5. 整型数据类型

 | 3.58万次播放

54. 主函数参数

 | 2892次播放