第 3 章

分 支 结 构

在程序设计中，分支结构如同十字路口的红绿灯，指引着代码的执行方向。当面对“如果…那么…”的逻辑判断时，分支结构让程序能根据条件选择不同路径，这种分支控制能力正是结构化程序设计的精髓所在。本章将通过 5 个精心设计的任务，带领读者掌握分支结构的运用艺术。

从判断迫击炮发射时间的合理性到实现成绩等级的智能转换，从景区门票的差异化计费到 5 级分制的百分数段映射，每个任务都对应着真实场景中的逻辑判断需求。读者将运用 if-else 的层层嵌套，体验多分支结构的决策魅力；通过 switch 语句的巧妙编排，感受多条件分支的简洁高效。这些实践不仅能帮助读者掌握分支结构的语法细节，更能培养将现实问题抽象为逻辑判断的能力。

分支结构不仅是语法规则，更是编程思维的分水岭。它让程序突破线性执行的桎梏，在条件判断的分支中展现出智能决策的魅力。通过本章的系统学习，读者将掌握程序“思考”的核心机制，为构建复杂应用系统打下坚实基础。

3.1 基本分支结构

3.1.1 任务 1：合理范围内计算炮弹坐标

第 2 章任务 1 实现了炮弹在任意输入时间 t 的坐标位置计算，但没有考虑时间 t 是否合理。例如，如果输入的时间是负数或者超过炮弹落地时间，根据公式计算出的纵坐标也是负数（假设是平地），这显然是不合理的。本次任务要求对输入时间进行限制，即输入时间必须在合理的范围之内（图 3.1）。任务描述如下。

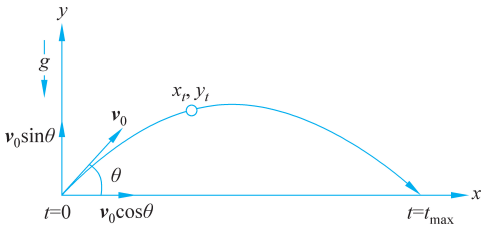


图 3.1 炮弹飞行的起始时间 0 和落地时间 t_{max}

已知迫击炮发射炮弹的初速度 $v_0=50\text{m/s}$，发射角度 $\theta=45\pi/180$，重力加速度 $g=9.8\text{m/s}^2$，不考虑空气阻力和落点有效区域，利用计算机模拟建立坐标系，编写程序，计算合理范围内某一时刻炮弹的坐标值，结果精确到小数点后 2 位。

3.1.2 if-else 分支结构

1.分支结构概述

在程序设计中，if-else 分支结构如同交通信号灯，通过条件判断引导程序执行不同路

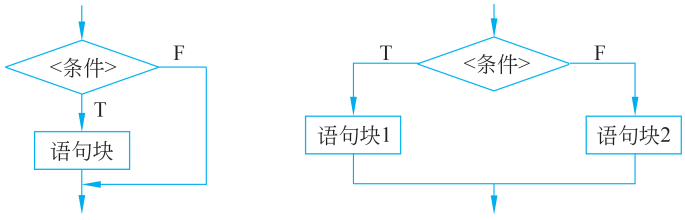
径。作为程序设计的核心控制结构之一，它根据条件表达式的真假，选择执行对应的代码块，为程序赋予决策能力。

分支结构常用于用户输入验证、状态监测、功能开关等场景，如游戏开发中根据玩家选择触发不同剧情、电商系统中根据用户行为推荐商品、工业控制中根据传感器数据调整设备参数。清晰的逻辑分支使程序意图一目了然，便于团队协作与后期维护。

掌握 if-else 分支结构，就如同为程序装上“智能开关”，使代码能根据条件变化灵活响应，为后续学习循环结构和函数奠定基础。

2. 单分支结构：编程决策的起点

如图 3.2(a)所示，单分支结构作为分支结构的基础形式，如同交通信号灯中的绿灯，仅当条件满足时允许程序执行特定代码块。这种简约而不简单的结构，通过 if 语句实现程序流程的初步控制，是编程逻辑中不可或缺的“决策单元”。



(a) 单分支结构　　(b) 双分支结构

图 3.2　单分支与双分支结构

(1) 基本语法。

C 语言单分支结构语法如下。

```
if (条件表达式) {
    //条件为真时执行的代码块
}
```

单分支结构的执行规则是仅当条件为真(非零值)时执行代码块，否则直接跳过。其中的条件表达式支持算术、关系、逻辑运算，如 x ＞ 0、a ＝＝ b 等。代码块置于花括号之内，若代码块只有一条语句，可省略花括号。代码块建议统一缩进(通常为 4 个空格或 1 个 Tab)以提高代码的可读性。

(2) 核心用途。

单分支结构专为单一条件判断场景设计，例如，本任务检查用户输入的时间 t 是否有效，只有 t 为正数且小于炮弹落地时间，才计算炮弹的坐标并输出结果。

```
if (t >0 && t<t_max) {
    //计算炮弹的坐标并输出结果;
}
```

(3) 注意事项。

由于浮点数在计算机中是不精确的表示，因此不应直接使用相等运算符(＝＝)判断是否相等。判断一个浮点数 f 是否近似为零，应检查其绝对值是否小于一个定义的精度值(如 1e-6)。

```
if (fabs(f) <1e-6)) {
    //代码块
}
```

判断两个浮点数 a 和 b 是否近似相等，则应检查它们差值的绝对值是否小于精度值。

```
if (fabs(a -b) <1e-6) {
    //代码块
}
```

3. 双分支结构

如图 3.2(b)所示，双分支结构在单分支结构的基础上增加了一条路径，如同交通信号灯中的红绿灯，不仅允许程序在条件满足时执行特定代码块，还在条件不满足时提供另一条执行路径。这种结构通过 if-else 语句实现，是编程逻辑中处理二元选择的核心工具。

(1) 基本语法。

在 C 语言中，双分支结构的标准语法如下。

```
if (条件表达式) {
    //条件为真时执行的代码块
} else {
    //条件为假时执行的代码块
}
```

双分支结构的执行规则是当条件为真(非零值)时执行第一个代码块，否则执行 else 后的代码块。

(2) 核心用途。

双分支结构专为二元条件判断场景设计。例如，任务 1 检查用户输入的时间 t 是否有效，当 t 为正数且小于炮弹落地时间时，计算炮弹的坐标并输出结果；否则输出错误提示信息。

```
if (t >0 && t<tmax) {
    //计算炮弹的坐标并输出结果;
}else{
    //输出错误提示信息
}
```

(3) 注意事项。

由于 else 必须与最近的未配对 if 配对，为避免嵌套错误，应使用{}明确代码块范围，增强可读性，即使代码块只有一条语句，也建议使用{}包裹；同时，应保持 if 和 else 代码块的缩进一致性。

掌握双分支结构，如同掌握了程序决策的“基础招式”。其简洁的语法背后，蕴含着对二元条件判断本质的深刻理解。在后续的多分支及嵌套结构学习中，这种基础思维将持续发挥作用，帮助开发者构建出既高效又健壮的程序逻辑。

3.1.3　炮弹飞行时间的分析

炮弹落地时，在竖直方向上的位移为零，即 $y_t=0$。代入式(2.1)可得式(3.1)：

$$y_t = \boldsymbol{v}_0 \sin\theta t - \frac{1}{2} g t^2 = 0 \tag{3.1}$$

求解方程，得到落地时间 t_{max} 如式(3.2)所示。

$$t_{max} = \frac{2\boldsymbol{v}_0 \sin\theta}{g} \tag{3.2}$$

接收用户输入时间 t；按式(3.2)计算 t_{max}，检查输入时间 t 是否在合理范围内，即 $0 \leqslant t \leqslant t_{max}$。如果 t 符合条件，使用式(2.1)计算并输出炮弹的坐标($x(t)$，$y(t)$)；如果 t 不合理，输出错误提示信息，告知用户输入时间不在有效范围内。通过以上步骤，可以设计出能够计算炮弹在合理时间范围内坐标值的程序。这样的程序不仅考虑了物理规律，还增加了对用户输入的有效性检查，提高了程序的健壮性。

炮弹飞行时间的分析充分地展现了逻辑思维在问题求解中的系统性应用。首先，通过物理方程建立数学模型，将炮弹运动问题转换为时间变量的函数关系，这是逻辑推理的起点。在程序实现层面，采用分支结构对输入时间进行验证，构建了 $0 \leqslant t \leqslant t_{max}$ 的约束条件，体现了逻辑思维中的分类讨论方法。条件判断的设计遵循了排中律原则，即时间 t 要么属于有效区间，要么不属于，不存在中间状态。这种非此即彼的二值逻辑判断，确保了程序行为的确定性。

3.1.4 计算合理范围炮弹坐标的程序实现

1. C 语言实现

程序 3.1 合理范围内计算炮弹的坐标点

```
00  #include <stdio.h>
01  #include <math.h>
02  #define PI 3.14159265
03  #define g 9.8
04  int main() {
05        double v0 =50.0;                                    //初速度,单位为 m/s
06        double theta =45.0 * (PI / 180.0);                  //角度弧度
07        double t;                                           //时间
08        double x, y;                                        //坐标值
09        int valid_time =1;              //时间是否合理的标志,假设一开始时间是合理的
10        double t_max =(2.0 * v0 * sin(theta)) / g;          //计算炮弹的落地时间
11        printf("输入时间(0～%.2lf): ",t_max );
12        scanf("%lf", &t);                                   //获取用户输入的时间
13        if (t >=0 && t<=t_max) {
14            x =v0 * cos(theta) * t;                         //炮弹的 x 坐标值
15            y =v0 * sin(theta) * t -0.5 * g * t * t;        //炮弹的 y 坐标值
16            printf("炮弹在时间 %.2lf 秒时的坐标为: (%.2lf, %.2lf)\n", t, x, y);
17        }else{
18          printf("输入的时间不合理!\n");
19        }
20        return 0;
21  }
```

2. 代码分析

程序 3.1 用于计算炮弹在某一特定时间的坐标，并使用分支结构判断时间是否合理。

程序第 10 行计算炮弹落地时间 t_{max}；第 11 行输出提示信息并提醒合理的时间范围 0～t_{max}；第 12 行通过 scanf()函数获取输入的时间，将其保存在变量 t 中；第 13 行判断输入时间 t 是否在合理的范围内，如果合理则执行第 14～16 行，计算炮弹在时间 t 的坐标并输出，否则执行第 18 行输出错误提示。

程序的输出结果为

```
输入时间(0～7.22): 6.2
炮弹在时间 0.20 秒时的坐标为: (7.07, 6.88)
```

3.2　多分支结构

3.2.1　任务 2：成绩等级评判

全国计算机等级考试(NCRE)是经教育部批准，由教育部考试中心主办，面向社会，用于考查应试人员计算机应用知识与技能的全国性计算机水平测评体系。NCRE 考试实行百分制计分，但以等级形式通知考生成绩。分数分为 4 等：90～100 分为“优秀”，80～89 分为“良好”，60～79 分为“及格”，0～59 分为“不及格”。

考试成绩在“及格”以上者为通过考试，由教育部考试中心颁发合格证书。请编写程序，输入一个百分制的成绩，以等级的形式显示结果。

3.2.2　if-else if-else 多分支结构

在 C 语言中，if-else if-else 链式结构是处理多分支逻辑的核心工具(图 3.3)，通过逐层判断条件实现精准分支控制。该结构以条件表达式为决策依据，特别适合处理具有优先级关系的复杂条件判断。

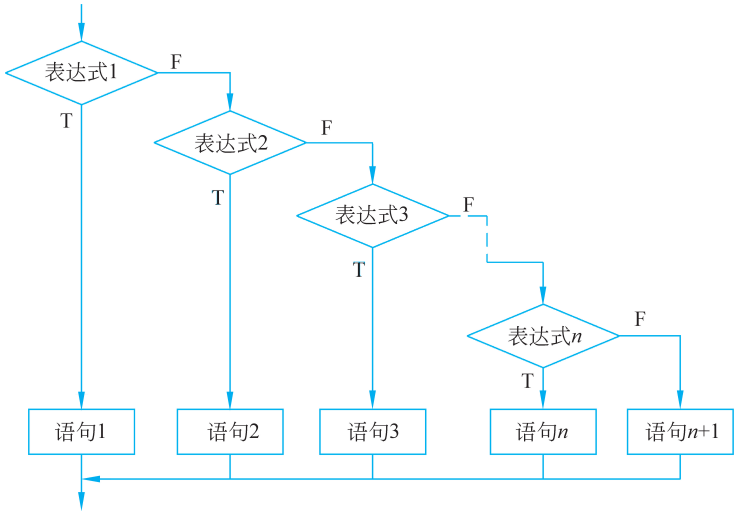


图 3.3　多分支结构

1. 基本语法

```
if(条件 1) {
    //分支 1 代码块
}
else if(条件 2) {
    //分支 2 代码块
}
else if(条件 3) {
    //分支 3 代码块
}
…
```

```
else {
    //默认分支代码块
}
```

多分支结构采用“短路求值”机制，当某个条件满足时，立即执行对应代码块并跳过剩余的判断。多分支结构通过清晰的阶梯式判断流程，能够有效处理具有优先级关系的业务逻辑。在实际开发中，建议结合具体场景设计条件顺序，并通过代码审查确保所有边界条件都得到妥善处理。

2. 核心用途

多分支结构可用于区间分级判断、权限验证、游戏状态机等场景。例如，如下代码是权限验证场景。

```
if(userLevel ==ADMIN) {
    //开放全部功能
}
else if(userLevel ==MANAGER) {
    //开放管理模块
}
else if(userLevel ==STAFF) {
    //开放基础功能
}
else {
    //拒绝访问
}
```

3. 注意事项

(1) 条件的顺序。

多分支结构对条件的顺序具有敏感性，因此，高频条件应前置以提高效率。例如，先判断“score ＜ 60”比后续分数区间判断更高效。

(2) 冗余判断。

使用该结构时要注意冗余判断，避免不必要的条件判断，例如：

```
//错误示范：条件 2 包含条件 1 的判断
if(x >0 && x <100) { … }
else if(x >0) { … }                          //冗余判断
//正确写法：
if(x >=100) { … }
else if(x >0) { … }
else { … }
```

(3) 默认分支设计。

必须处理所有未明确列出的情况，例如，对非法输入应设置错误处理机制。

(4) 避免过度复杂的逻辑嵌套。

为提高代码的可维护性，确保每个分支保持单一职责原则，同时添加必要的注释说明业务逻辑，并保持垂直对齐的代码格式。

(5) 性能优化策略。

将最可能执行的条件放在最前面以提高整体效率，合理使用 && 与 || 运算符减少计算量。

3.2.3　成绩分级方法分析

1. 百分制与等级的映射关注

根据题意，百分制分数与成绩等级的对照关系如表 3.1 所示。

表 3.1　成绩百分制与成绩等级

百　分　制	成 绩 等 级	百　分　制	成 绩 等 级
[90,100]	优秀	[60,80)	及格
[80,90)	良好	[0,60)	不及格

2. 多分支结构设计

以表 3.1 的对应关系为基础，设计 if-else if-else 多分支结构，按分数的区段输出结果。多分支结构如下（假设分数为变量 score）。

```
if (score>100 || score<0) printf("错误的输入\n");          //条件表达式 1
else if (score>=90 && score<=100) printf("优秀\n");       //条件表达式 2
else if (score>=80 && score<90) printf("良好\n");         //条件表达式 3
else if (score>=60 && score<80) printf("及格\n");         //条件表达式 4
else if (score>=0 && score<60) printf("不及格\n");        //条件表达式 5
```

上述代码的第一个条件表达式处理不在这些区间内的非法输入，后面 4 个条件表达式对应表 3.1 的各个分数区间。

3. 多分支结构的优化

由于多分支结构具有“短路求值”机制，即任何一个表达式的结果为真时，将立即执行对应代码块并跳过剩余的判断。反过来说，任何一个表达式的结果为假时，才会执行后面的表达式。因此，如果多分支结构执行到了第 k 个表达式，其前面的 $k-1$ 个表达式的结果一定都是假。因此，可以利用已经由前面的 $k-1$ 个表达式验证为假的情况，优化第 k 个表达式。

例如，只有当条件表达式 1 为假时，才会执行条件表达式 2。而条件表达式 1 为假，意味着 score＞100 是不成立的，因此不需要在条件表达式 2 中添加 score＜＝100；同理，条件表达式 3、4 也可以省略后面的那个条件。对于条件表达式 5，由于表达式 1 不成立，score＜0 不成立，因此表达式 5 可以完全省略。多分支结构可以简化为

```
if (score>100 || score<0) printf("错误的输入\n");      //条件表达式 1
else if (score>=90) printf("优秀\n");                  //条件表达式 2
else if (score>=80) printf("良好\n");                  //条件表达式 3
else if (score>=60) printf("及格\n");                  //条件表达式 4
else printf("不及格\n");
```

成绩分级求解方案展现了计算思维中的分类逻辑与优化策略。通过建立百分制与等级之间的映射关系表，将连续数值区间离散化为有限类别，体现了问题分解与模式识别的基本方法。

多分支结构的设计采用层次化条件判断，通过 if-else if-else 的分支逻辑实现区间的严格划分，反映了计算思维中精确控制流程的特性。短路求值机制的运用则展示了计算效率优化思想，即利用逻辑表达式的执行特性，通过逐步缩小判断范围，减少冗余条件检测。

优化后的分支结构去除了重复验证的条件，仅保留必要的下限判断，这种简化既保持了

逻辑完整性，又提高了执行效率。整个设计过程从初始的完整条件表达到逐步精简，呈现了计算思维从功能实现到性能优化的递进过程，最终形成的紧凑分支结构既确保了分类准确性，又体现了计算资源的经济性原则。

3.2.4 成绩等级的程序实现

1. C 语言实现

程序 3.2 百分制转换成等级

```
00  #include <stdio.h>
01  int main ( void )
02  {
03      float score;
04      printf ("请输入分数(0-100): ");
05      scanf("%f",&score);
06      if (score>100 || score<0) printf("错误的输入\n"); //条件表达式 1
07      else if (score>=90) printf("优秀\n");             //条件表达式 2
08      else if (score>=80) printf("良好\n");             //条件表达式 3
09      else if (score>=60) printf("及格\n");             //条件表达式 4
10      else printf("不及格\n");
11  }
```

2. 代码分析

程序第 5 行通过 scanf( )函数接收输入的学生成绩，将其保存在变量 score 中；程序第 6～10 行使用多分支结构根据输入将成绩划分到不同的区间，并输出相应的等级信息，该结构利用条件表达式执行的先后次序，对结构进行了优化。

程序的输出结果为

```
请输入分数(0-100): 85
良好
```

3.3 分支结构的嵌套

3.3.1 任务 3：景点门票计费

某旅游景点(图 3.4)为吸引游客，旺季和淡季的门票价格不同，旺季为每年 5～10 月，门票不打折，而淡季的门票价格是八折。无论是旺季还是淡季，65 岁及以上的老人均免票，14 岁以下的孩童均半价，其余游客全价。

假设景点全票价格为 100 元，请编写一个景点门票计费程序，输入游览月份和游客年龄，输出票价。

3.3.2 嵌套分支结构

1. 分支结构嵌套的语法

分支结构可以嵌套使用。一个分支结构的代码块中可以包含另一个分支结构。嵌套的深度没有限制。一般形式如下。

图 3.4　旅游景点

```
if( 条件表达式 1){
    if (条件表达式 2) {
          代码块 1;
    }else{
          代码块 2;
}else{
    if (条件表达式 3){
          代码块 3;
    }else{
          代码块 4;
}
```

计算条件表达式 1，如果结果为真，则执行 if 结构中的代码：计算条件表达式 2，如果结果为真，执行代码块 1；否则执行代码块 2。

如果条件表达式 1 的结果为假，则执行 else 结构中的代码：计算条件表达式 3，如果结果为真，执行代码块 3；否则执行代码块 4。

2. 嵌套的二义性

需要注意 if 与 else 的配对关系。从最内层开始，else 总是与它前面最近的未曾配对的 if 配对，除非用花括号改变其配对关系。例如，如下形式的结构：

```
if(条件表达式 1)
    if (条件表达式 2)     代码块 1;
else
    代码块 2;
```

从代码的缩排形式上看，else 是与第 1 个 if 对齐的，意图是使 else 与第 1 个 if 对应。但实际上按 else 的匹配规则，是与第 2 个 if 配对的，因为它们相距最近。为避免二义性，可以用花括号来确定配对关系。

```
if (条件表达式 1){
    if(条件表达式 2)   语句块 1;
}else
    语句块 2;
```

上面的代码使用“{ }”限定了内嵌 if 语句的范围，因此 else 与第 1 个 if 配对。

3. 分段函数问题

对于如式(3.3)所示分段函数，用嵌套分支结构进行求解。

$$y=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases} \tag{3.3}$$

下面的代码看似能够求解式(3.3)的分段函数，但按照 else 匹配最近 if 的规则，实际上 else 是与第 2 个 if 语句匹配的。

```
y=0
if(x>=0)
    if(x>0)
        y=1;
else
    y=-1;
```

因此，以上代码实际等价于如下形式。

```
y=0
if(x>=0){
    if(x>0)
        y=1;
    else
        y=-1;
}
```

当 $x>0$ 时，y 为 1；当 $x=0$ 时，y 为 -1；否则($x<0$)，y 保持原来的值 0，所代表的分段函数为式(3.4)。

$$y=\begin{cases}1, & x>0\\ -1, & x=0\\ 0, & x<0\end{cases} \tag{3.4}$$

可用“{ }”限定内嵌 if 语句的范围，将代码修改为

```
y=0
if(x>=0){
    if(x>0)
        y=1;
}else
    y=-1;
```

因此，为了避免嵌套分支结构的错误匹配问题，应该为 if-else 分支结构尽可能地添加{}，哪怕代码块中只有一条语句。

需要注意的是，对于分段函数问题，上面的代码并非最好的设计方式，仅仅是为了演示嵌套分支结构。更好的方法是使用 3.2 节介绍的多分支结构。

```
if(x>0) y=1;
else if (x==0) y=0;
else y=-1;
```

4. 嵌套分支结构的应用案例

嵌套分支结构通常用于需要多层次条件判断的场景，其核心特点是外层条件的结果决定是否需要继续执行内层条件判断。以下是典型使用场景的分析。

(1) 多阶段决策流程。

多阶段决策流程是指当问题需要分阶段处理,且后续阶段的判断依赖前一阶段的结果。例如,用户登录系统时,先验证账号是否存在,若存在则继续验证密码,若密码正确则判断账号状态(如是否冻结)。每一阶段的判断结果决定能否进入下一阶段,形成链式决策。代码结构如下。

```
if (账号存在){
      if (密码正确){
          if (账号状态正常){
              //允许登录
          }
      }
}
```

(2) 条件细化与优先级。

当某个外层条件满足后,需对内层条件进一步细分。例如,在任务 3 景点门票计费问题中,外层分支结构用来判断是否旺季,以决定是否八折;内层分支结构用来根据年龄计算票价。

```
if (当前是旺季){
    if (是 65 岁及以上老人){
        //免票
    }else if (是 14 岁以下){
        //半价
    }else{
        //全价
    }
}else{
    if (是 65 岁及以上老人){
        //免票
    }else if (是 14 岁以下){
        //8 折后半价
    }else{
        //8 折
    }
}
```

嵌套分支的核心价值在于将复杂问题分解为逐层递进的子问题,适用于逻辑依赖性强、条件间存在“先后关系”的场景。合理使用嵌套能提升代码的表达力,但需警惕过度嵌套导致的意大利面条代码。意大利面条代码指非结构化、难以维护的源代码,通常表现为代码逻辑混乱、嵌套过深、依赖复杂等问题。这种代码因其控制流像一碗意大利面条一样纠缠不清而得名。

3.3.3　门票计算方法分析

1. 票价优惠详情

根据题目提供的信息,该旅游景点的门票价格是根据季节和游客的年龄来确定优惠政策的。以下是门票价格的详细说明。

(1) 旺季(5～10 月)。

特殊人群的优惠政策:65 岁及以上的老人免票,14 岁以下的孩童是全价的一半。其他游客(即不符合上述两类优惠条件的游客)全价。

(2) 淡季(11 月到次年 4 月)。

特殊人群的优惠政策:65 岁及以上的老人免票,14 岁以下的孩童是八折的一半。其他

游客八折。

2. 设计思路

可设计嵌套分支结构，外层分支处理旺季与淡季，内层分支处理年龄，实现不同季节和不同年龄段的门票价格计算。

首先，根据输入的月份，判断是旺季还是淡季。采用 if-else 结构完成该判断。然后，根据输入的游客年龄，进一步判断游客是否符合优惠政策，即 65 岁及以上的老人免票，14 岁以下的孩童半价。

(1) 在旺季情况下，如果游客的年龄大于或等于 65 岁，则免票；如果游客的年龄在 14 岁以下，则半价；否则门票全价。采用 if-else if -else 结构完成。

(2) 在淡季情况下，如果游客的年龄大于或等于 65 岁，则免票；如果游客的年龄在 14 岁以下，在八折基础上半价；否则门票价格打八折。采用 if-else if -else 结构完成。

门票计费问题的求解思路体现了计算思维中层次化分解与结构化决策思想。通过将复杂的票价规则分解为季节和年龄两个维度，建立了清晰的决策树模型，反映了问题分解的基本方法。外层分支对时间维度进行离散化处理，将连续月份划分为旺季和淡季两个互斥区间，体现了分类逻辑的完备性。内层分支采用年龄作为决策属性，通过严格的边界条件(14 岁和 65 岁)实现人群的精确划分，展示了计算思维中的精确量化特征。

3.3.4 景点门票计费的程序实现

1. C 语言实现

程序 3.3 景点门票计费

```
01  #include<stdio.h>
02  int main ()
03  {
04      int month , age ;
05      float price=100, money ;
06      printf ("请输入游览月份: ");          //输入月份
07      scanf ("%d",& month );
08      printf ("请输入游客年龄: ");
09      scanf ("%d",&age );                   //输入游客的年龄
10      if (month>=5 && month<=10){           //判断是否旅游旺季
11          if (age>=65) money=0 ;            //判断是否 65 岁以上
12          else if(age<14) money=price/2;    //判断是否 14 岁以下
13          else money=price;
14      }else{
15          if (age>=65) money =0;
16          else if (age<14) money=price * 0.8/2;
17          else money =price * 0.8;
18      }
19      printf ("该游客应购门票价格为%.1f\n", money);
20  }
```

2. 代码分析

程序根据旅游季节和游客的年龄来计算门票费用。

程序第 7 行输入游览月份存储到变量 month 中，第 9 行输入游客年龄存储到变量 age 中，景点门票单价存储到变量 price 中。

第 10 行是外层分支语句，判断月份是否为 5～10，即 month>=5 && month<=10

是否成立，如果为真则是旅游旺季。在旅游旺季中，第 11～13 行根据游客的年龄进一步确定门票价格，如果游客年龄大于或等于 65 岁(age＞＝65)，门票价格设为 0；如果游客年龄小于 14 岁(age＜14)，门票价格为一半；否则，门票价格为原价。

如果月份不为 5～10，则是旅游淡季。在旅游淡季，第 15～17 行根据游客的年龄进一步确定门票价格；如果游客年龄大于或等于 65 岁，门票价格设为 0；如果游客年龄小于 14 岁，门票价格为八折的一半(即 price×0.8/2)；否则，门票价格为原价的八折(即 price×0.8)。

最后，第 19 行使用 printf 函数输出该游客应购买的门票价格 money，保留 1 位小数。

程序的输出结果为

```
请输入游览月份：6
请输入游客年龄：12
该游客应购门票价格为 50.0
```

3.4　switch 分支结构

3.4.1　任务 4：钢筋等级与用途

钢筋(图 3.5)等级是根据其强度、化学成分和用途进行划分的，主要用于建筑和工程结构中。表 3.2 列出的是常见的钢筋等级及其特点和用途。

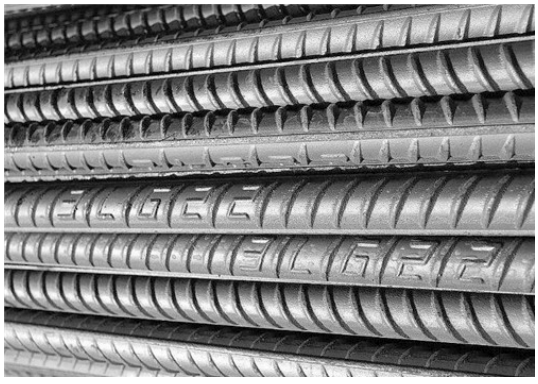


图 3.5　钢筋

表 3.2　常见钢筋等级与特点和用途

等级	强度 MPa	特　　点	用　　途
1	300	低强度，塑性好，伸长率高，便于弯折成型，容易焊接	广泛用于中、小型钢筋混凝土结构的主要受力钢筋，构件的箍筋，以及钢、木结构的拉杆等
2	335	强度较高，塑性较好，焊接性能理想，表面有纵筋和横肋以增强与混凝土的黏结	用于大、中型钢筋混凝土结构，如桥梁、水坝、港口工程和房屋建筑结构的主筋
3	400	性能与二级钢筋相似，但强度更高	适用于需要更高强度的建筑结构
4	500	强度最高，加入钒或钛等合金元素以提高强度和韧性	主要用于预应力混凝土板类构件及大型预应力建筑构件，如屋架、吊车梁等

编写程序，输入钢筋的等级，输出其对应的强度和用途。

3.4.2 switch 分支结构

1. switch 分支结构

在 C 语言中，switch 分支结构是一种基于变量或表达式的值进行多路分支选择的控制结构。与 if-else 多分支结构相比，switch 提供了一个更加简洁和易读的方法来处理多个条件。switch 的语法结构如下。

```
switch (表达式) {
    case 常量 1:
        //代码块 1
        break;
    case 常量 2:
        //代码块 2
        break;
    …
    default:
        //默认代码块
}
```

表达式的结果必须为整型或枚举类型；case 标签后跟常量表达式，用于匹配表达式的值；break 是用于退出 switch 语句的关键字，default 是可选分支，当所有 case 都不匹配时执行。

例如，学生成绩等级 A 对应 90～100、B 对应 80～89、C 对应 70～79、D 对应 60～69，E 对应 0～60，下面的代码可以根据成绩等级输出对应的分数区间。

```
switch (grade)
{
    case 'A' : printf("90～100\n"); break;
    case 'B': printf("80～89\n"); break;
    case 'C': printf("70～79\n"); break;
    case 'D': printf("60～69\n"); break;
    case 'E': printf("0～60\n"); break;
}
```

switch 分支结构首先计算表达式的值，然后按顺序匹配 case 常量，找到匹配项后执行对应代码块，若遇到 break 则退出分支结构。除非逻辑上需要进行 case 穿透（见 3.5.2 节），建议每个 case 后加 break，避免 case 穿透现象发生。

2. switch 与 if-else 的核心区别

在 C 语言中，switch 和 if-else 均为条件分支结构，但二者在设计逻辑、适用场景和性能表现上存在显著差异。

（1）结构差异。

从结构上来看，switch 基于单一整型/字符型表达式的值进行多分支匹配，通过 case 标签定义离散值分支。语法要求 case 值为编译期常量。而 if-else 通过布尔表达式逐级判断，支持任意复杂条件（如范围判断、逻辑组合），可嵌套实现多层分支，灵活性更高。

（2）适用场景差异。

switch 和 if-else 结构分别适合在不同场景下使用，switch 擅长处理固定离散值的分支选择（如枚举状态码），代码结构扁平化，可读性优于多层嵌套的 if-else。而 if-else 适合动态条件或范围判断（如 if(score>=90)），能处理浮点运算、字符串比较等非整型条件。

(3) 性能差异。

在性能方面，switch 结构可能在编译期优化为跳转表(Jump Table，一种编译优化技术，以 case 标签的值为索引直接检索出相应代码执行地址)，以提高 case 标签的匹配效率，但仅适用于标签密集分布情形，对于稀疏值分布，可能退化为链式比较。而 if-else 采用顺序判断，分支较多时效率较低，但现代编译器会优化预测分支。

(4) 使用场景差异。

与 if-else 结构相比，switch 更适合如下场景：菜单选择系统(如 1-添加，2-删除，3-修改)、状态机模式(如订单状态：待支付 → 已支付 → 已发货)、错误码处理(如服务器的状态码分类)、枚举类型处理(如文件类型 PDF、DOCX、TXT)。

if-else 则更适合这些场景：范围判断(如 if(score ＞＝90))、动态条件组合(if(a＞0 && b＜10))、需要提前返回的场景(利用 else if 的短路特性)、复杂业务逻辑判断等。

3.4.3　钢筋等级问题分析

1. 分支结构选择

钢筋等级问题任务可以用 if-else 多分支结构实现，也可以用 switch 分支结构实现。使用 if-else 多分支结构的实现框架如下。

```
if (grade==1){
    //输出等级 1 钢材的相关信息
}else if (grade==2){
    //输出等级 2 钢材的相关信息
...
```

由于钢筋的等级是连续的整数，因此使用 switch 结构效果更好，可以提高代码可读性并获得编译器的优化加成。

2. 设计思路

采用 switch 结构程序架构如下。

```
switch(grade){
    case 1:                          //处理 1 级钢筋
    case 2:                          //处理 2 级钢筋
    case 3:                          //处理 3 级钢筋
    case 4:                          //处理 4 级钢筋
    default:                         //异常处理
}
```

上述结构中，4 个标签分别处理 4 个等级的钢筋。default 分支处理非法的输入信息，用于增强程序的健壮性。这些设计体现了计算思维中的离散化分类与结构化处理思想。钢筋等级作为离散的整数枚举值，天然适合采用 switch 分支结构进行处理，这种选择反映了对问题数据特征的准确识别和匹配。通过将连续的钢筋等级离散化为有限的 case 标签，实现了从数值到类别的清晰映射，展现了计算思维中的模式抽象能力。switch 结构中 default 分支的设置体现了计算思维中的容错设计理念，为所有可能的异常输入提供了统一的处理路径，确保了程序的完备性。

3.4.4 钢筋等级处理的程序实现

1. C 语言实现

程序 3.4 钢筋等级问题

```
00  #include <stdio.h>
01  int main() {
02        int grade;
03        printf("请输入钢筋等级(1-4): ");   //输入提示
04        if(scanf("%d", &grade) !=1) {      //获取用户输入,验证是否为整数
05            printf("错误: 请输入有效的数字!\n");
06            return 1;                      //异常退出
07        }
08        switch(grade) {                    //使用 switch 语句处理不同等级
09        case 1:
10            printf("等级: 1\n");
11            printf("强度: 300 MPa\n");
12            printf("用途: 广泛用于中、小型钢筋混凝土结构的主要受力钢筋,构件的箍筋,
              以及钢、木结构的拉杆等。\n");
13            break;
14        case 2:
15            printf("等级: 2\n");
16            printf("强度: 335 MPa\n");
17            printf("用途: 用于大、中型钢筋混凝土结构,如桥梁、水坝、港口工程和房屋建筑
              结构的主筋。\n");
18            break;
19        case 3:
20            printf("等级: 3\n");
21            printf("强度: 400 MPa\n");
22            printf("用途: 适用于需要更高强度的建筑结构。\n");
23            break;
24        case 4:
25            printf("等级: 4\n");
26            printf("强度: 500 MPa\n");
27            printf("用途: 主要用于预应力混凝土板类构件及大型预应力建筑构件,如屋架、
              吊车梁等。\n");
28            break;
29        default:
30            printf("错误: 无效的钢筋等级!请输入 1-4 的整数。\n");
31     }
32     return 0;
33  }
```

2. 代码分析

程序 3.4 是根据钢筋等级来输出对应的强度和用途。第 3～7 行通过键盘输入获得钢筋等级并存储在 grade 变量中,如果输入正确,scanf 函数的返回值是 1。第 4 行利用该返回值判断输入的数据是否是整数,如果不是,在第 5 行输出提示信息并退出整个程序。

第 8～31 行是用 switch 语句实现的多分支结构,每个标签对应一种类型的钢筋,default 标签用于处理非法的输入数字。

程序的输出结果为

```
请输入钢筋等级(1-4): 3
等级: 3
强度: 400 MPa
用途: 适用于需要更高强度的建筑结构。
```

3.5　switch 结构应用

3.5.1　任务 5：成绩等级评判(switch)

和本章任务 2 一样，仍然是根据输入的成绩输出等级信息，但是要求用 switch 结构来实现。

分数分为 4 等，90～100 分为“优秀”，80～89 分为“良好”，60～79 分为“及格”，0～59 分为“不及格”。编写程序，输入一个百分制的成绩，以等级的形式显示结果。

3.5.2　case 穿透

1. case 穿透现象

在 C 语言的 switch 语句中，case 穿透指的是在一个 case 子句执行完毕后，如果没有遇到 break 语句，程序会继续执行下一个 case 子句的代码。这种特性使得多个 case 可以共享同一段代码，常用于处理具有连续性的多个值或需要统一处理的多个情况。例如，下面的代码在成绩等级为 A、B、C、D 时，会发生 case 穿透现象而执行 case 'D'的代码块输出“及格”。

```
switch (grade)
{
    case 'A' :                              //匹配'A'之后,继续执行 case 'B'的代码块
    case 'B':                               //匹配'B'之后,继续执行 case 'C'的代码块
    case 'C':                               //匹配'C'之后,继续执行 case 'D'的代码块
    case 'D': printf(">60\n"); break;       //执行 case 'D'的代码块后退出 switch
    case 'E': printf("<60\n"); break;       //执行 case 'E'的代码块后退出 switch
}
```

2. 谨慎使用 case 穿透

在使用 case 穿透时，需要特别注意控制流程，确保穿透行为符合预期。因为一旦省略了 break 语句，程序就会继续执行后续的 case 代码，这可能导致意外的逻辑错误。因此，在设计 switch 语句时，应谨慎考虑哪些 case 需要穿透，哪些需要独立处理，并在适当的位置添加 break 语句以终止 switch 语句的执行。

3.5.3　成绩等级问题分析

为了满足将百分制成绩转换为等级的需求，可以利用 switch 语句的 case 穿透特性来简化代码。观察表 3.1，将分数除以 10 之后的结果作为 switch 的条件，可以发现其与等级的对应关系如表 3.3 所示。

表 3.3　成绩/10 结果与成绩等级

成绩/分	成绩/10	成 绩 等 级
[90,100]	9,10	优秀
[80,90)	8	良好
[60,80)	6,7	及格
[0,60)	0,1,2,3,4,5	不及格

由于部分成绩等级对应多个整数，例如，优秀对应 9 和 10，及格对应 6 和 7，可以使用 case 穿透来完成。

3.5.4 成绩等级的程序实现(switch)

1. C 语言实现

程序 3.5 成绩等级判断(switch)

```
00  #include <stdio.h>
01  int main() {
02        int score;
03        printf("请输入分数(0-100): ");
04        scanf("%d", &score);
05        if (score>100 || score<0) {
06            printf("错误的输入\n");
07            return 1;
08        }
09        switch(score / 10) {                    //将分数除以 10
10            case 10://穿透到下一个 case,继续执行
11            case 9:
12                printf("优秀!\n");
13                break;
14            case 8:
15                printf("良好\n");
16                break;
17            case 7:                             //穿透到下一个 case,继续执行
18            case 6:
19                printf("及格\n");
20                break;
21            default:
22                printf("不及格\n");
23        }
24        return 0;
25  }
```

2. 代码分析

程序第 3～8 行处理输入数据，若输入数据不合法则输出错误提示并退出程序。第 9～23 行是用 switch 分支结构实现的成绩等级判断。第 10～13 行用 10 和 9 匹配“优秀”等级，注意第 10 行的 case 由于没有 break 会穿透到第 11 行的 case，因此 100 分和 90～99 分都会执行第 12 行输出“优秀”。由于“良好”对应的成绩是 80～89，除以 10 之后的结果都是 8，因此第 14～16 行的 case 模块添加了 break 语句防止发生 case 穿透。由于“及格”对应的成绩是 60～79，除以 10 之后的结果有两类，即 6 和 7，因此第 17～20 行利用 case 穿透完成 1 到多的映射。最后，不及格分数除以 10 对应多个值(0～5)，由 default 分支统一处理。

程序的输出结果为

```
请输入分数(0-100): 90
优秀!
```

小结

本章深入探讨了分支结构在编程中的核心作用，重点解析了 if-else 和 switch 两种分支控制语句，通过理论结合实例的方式，展现了它们在处理不同决策场景时的灵活性与高

效性。

if-else 语句作为双分支结构的典型代表，为程序赋予了基础的条件判断能力。它通过判断条件表达式的真假，选择执行对应的代码块，是处理二元选择问题的基本工具。本章通过判断迫击炮发射时间的合理性任务，展示了 if-else 在范围判断中的直接应用；在计算景点门票的案例中，通过嵌套 if-else 实现了多条件组合决策，体现了其在复杂逻辑处理中的扩展性。该语句的链式结构更支持多分支场景，如根据用户输入动态调整程序行为，是用户交互、状态监测等场景的首选方案。

switch 语句则为多分支选择提供了更优雅的解决方案。它通过匹配表达式值与常量，精准定位执行路径，特别适用于处理固定离散值的分支场景。本章的成绩等级转换任务中，利用 switch 配合算术运算将百分制映射到单个数字，结合 case 穿透特性，显著简化了代码量。钢筋等级查询案例则进一步验证了 switch 在处理枚举类型数据时的天然优势，每个 case 独立维护业务逻辑，配合 default 兜底处理，既保证了代码的可读性，又增强了系统的健壮性。

通过本章的学习，读者不仅可掌握分支结构的语法细节，还能培养将现实问题抽象为逻辑判断的能力，为后续构建复杂算法奠定了坚实基础。

习题

一、简答题

1. C 语言的分支语句有几种？请分别描述它们的基本形式。
2. 什么是 if 语句中的条件表达式？它的作用是什么？
3. if 语句有哪些常见的嵌套使用方式？请给出几个例子。
4. switch 语句在什么情况下会比 if-else 语句更有效？
5. 在 switch 语句中，case 和 default 标记的作用是什么？
6. 什么是 break 语句？它在 switch 语句中的作用是什么？

二、编程题

1. 猜数游戏。

设计一个猜数游戏，由计算机产生一个随机数 magic，从键盘输入一个数 guess，若输入的数 guess 等于随机数 magic，则输出“你猜对了！”，否则输出“你猜错了！”。

提示：用库函数 rand()产生随机数，为避免每次产生相同的随机数序列，需要使用 srand(unsigned seed)来设置随机数种子，一般使用 time()函数获取当前时间作为随机数种子，传入 srand 函数作为参数。

2. 整除判断。

输入一个整数，判断它能否同时被 3、5、7 整除。

提示：判断一个整数能否被另一个整数整除，可以使用模运算(%)，若余数为 0 则表示能整除，否则不能整除。

3. 求最大值。

输入三个浮点数，找出并输出其中的最大值。

提示：设三个数分别为 a、b、c，先比较 a 和 b，将较大者存入变量 max_val，然后比较 max_val 和 c，将较大者存入 max_val，max_val 的值即为最大值。

4. 三个数排序。

输入三个任意整数，然后按从小到大的顺序依次输出。

提示：输入三个任意整数分别存在变量 a、b 和 c 中。然后对这三个数进行两两比较：使用 if 语句进行条件判断，如果 a 大于 b，则借助中间变量 t 交换 a 和 b 的值，同理可以比较 a 和 c、b 和 c，最终 a、b、c 的值按从小到大排列。

5. 闰年判断。

输入 4 位数字的年份，判断是否是闰年，如果是则输出“闰年”，否则输出“平年”。

提示：若年份数字能被 4 整除但不能被 100 整除 或者能被 400 整除，则为闰年。

6. 月份的天数。

输入年份数字和月份数字，输出该月的天数。

提示：月份 1、3、5、7、8、10、12 为大月，每月 31 天；月份 4、6、9、11 为小月，每月 30 天；月份为 2 月时，闰年 29 天，否则 28 天。

7. 一元二次方程求解。

对于一元二次方程 $ax^2+bx+c=0$，输入系数 a、b、c，求方程的根。

提示：输入三个系数 a、b、c，求根公式为 $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$。令 delta$=b^2-4ac$，若 delta$>0$，方程有两个根；若 delta$=0$，方程有一个根；若 delta$<0$，方程无根。

8. 判别输入字符的类别。

根据输入字符的 ASCII 码所在范围判别其类型，分别给出不同的输出。若 ASCII 码值小于 32 输出“控制字符”，0 和 9 之间的字符输出“数字”，A 和 Z 之间的字符输出“大写字母”，a 和 z 之间的字符输出“小写字母”，其余字符输出“其他字符”。例如，输入字符 e，输出“小写字母”。

9. 年龄段的划分。

年龄段划分标准是根据联合国世界卫生组织，对全球人体素质和平均寿命进行检测，并对年龄的划分标准给出了新的规定。我国年龄划分标准是：童年$<$5 周岁；幼年 5～9 周岁；少年 10～19 周岁；青年 20～34 周岁；壮年 35～49 周岁；中年 50～64 周岁；老年$\geqslant$65 周岁。

编写程序，输入年龄，显示其所在年龄段。

10. 分段计算票价。

乘坐地铁时，一般按照乘坐的站数计算票价，表 3.4 给出了站数与票价的对应关系。

表 3.4 站数与地铁票价的关系

站数	1～6 站	7～11 站	12～16 站	17～21 站	22 站及以上
票价	2	3	4	5	6

编写程序，根据乘坐的站点数，输出对应的票价。

11. 分段函数。

对于如式(3.5)所示数学公式：