第1章　编程基础进阶

1.1　流程图与程序设计思维

1.1.1　学习任务

（1）掌握流程图的基本符号及其含义，能够正确识别和使用各类图形元素。

（2）理解程序设计的 3 种基本结构（顺序、分支和循环），并能用流程图准确表示。

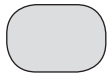
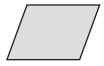
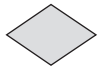
（3）能够将流程图转换为 C++ 代码，实现从算法设计到程序实现的转换。

1.1.2　知识讲解

1. 流程图的概念

流程图是由一系列标准化的图形符号和线条组成，用于表示算法或程序的逻辑流程。其基本构成部分如表 1-1 所示。

表 1-1　流程图基本构成部分

符　号	名　称	图 形 名	功　能
	起止框	椭圆形	表示流程的开始或结束
	输入 / 输出框	平行四边形	表示数据的输入或输出
	数据处理框	矩形	表示赋值、计算等处理步骤
	判断框	菱形	表示条件判断，引出分支
	流程线	带有箭头的线条	指示程序的执行方向

2. 顺序结构的流程图

顺序结构是最基本的程序结构，程序按照代码书写的顺序，从上到下依次执行每一条语句，不发生任何跳转。

流程图特征：代码从第 1 行执行到最后一行，中间不会跳过任何语句，如图 1-1 所示。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;                    // 操作 1：输入
    int sum = a + b;                  // 操作 2：计算
    cout << sum << endl;              // 操作 3：输出
    return 0;
}
```

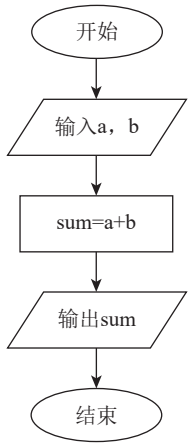


图 1-1　顺序结构流程图

3. 分支结构的流程图

分支结构根据条件的真假，选择不同的执行路径。程序会先判断条件，然后决定执行哪一段代码。

（1）单分支结构（if）

流程图特征：判断框只有“是”的一边有操作，“否”的一边直接跳过，如图 1-2 所示。

```
int score;
cin >> score;
if (score >= 60) {
    cout << "及格" << endl;
}
// 无论是否及格，都会执行到这里
```

（2）双分支结构（if-else）

流程图特征：判断框的两个出口都有操作，“是”走一条路，“否”走另一条路，最后汇合，如图 1-3 所示。

```
int age;
cin >> age;
if (age >= 18) {
    cout << "成年人" << endl;
} else {
    cout << "未成年人" << endl;
}
```

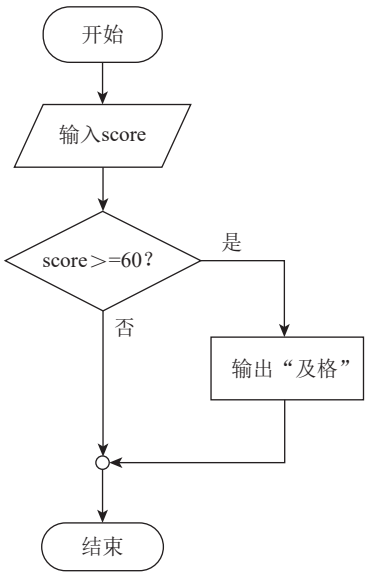


图 1-2　if 结构流程图

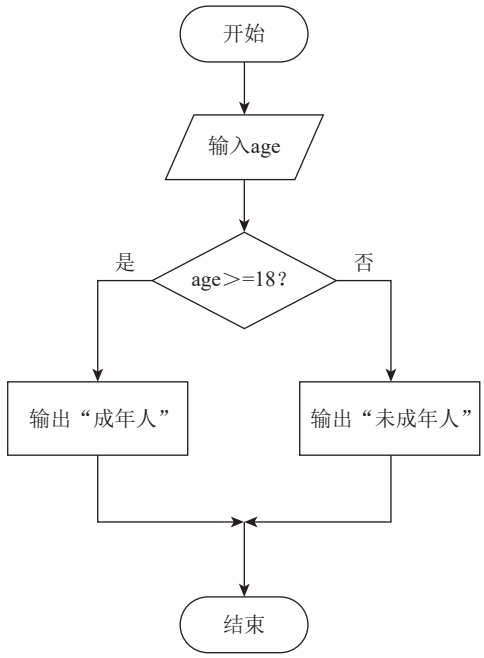


图 1-3　if-else 结构流程图

（3）多分支结构（if-else if-else）

流程图特征：多个判断框串联，依次判断多个条件，一旦某个条件为真，就执行对应操作并跳出整个结构，如图 1-4 所示。

```
int score;
cin >> score;
if (score >= 90) {
    cout << "优秀" << endl;
} else if (score >= 80) {
    cout << "良好" << endl;
} else if (score >= 60) {
    cout << "及格" << endl;
} else {
```

```
    cout << "不及格" << endl;
}
```

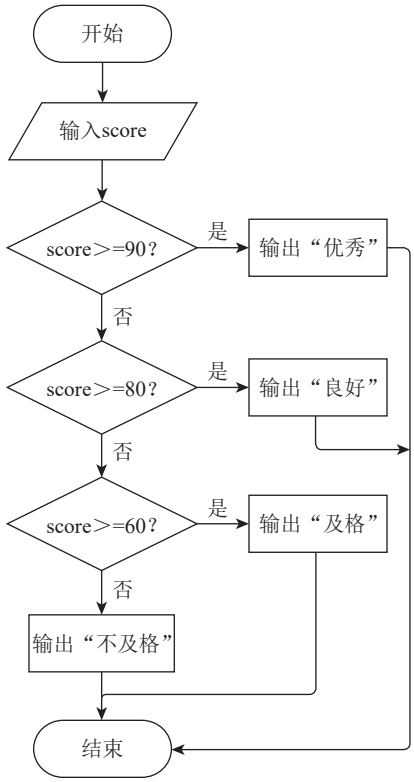


图 1-4　if-else if-else 结构流程图

4. 循环结构的流程图

循环结构用于重复执行一段代码，直到满足退出条件为止。它通过一条“回头”的箭头线，让程序能够反复执行某些操作。

（1）while 循环（先判断后执行）

流程图特征：判断框在循环体之前，先检查条件，条件为真才进入循环体，如图 1-5 所示。

```
int i = 1;
while (i <= 5) {
    cout << i << " ";
    n++;  // 改变循环变量，避免死循环
}
// 输出：1 2 3 4 5
```

（2）do-while 循环（先执行后判断）

流程图特征：判断框在循环体之后，先无条件执行一次循环体，再检查条件，如图 1-6 所示。

```
int i = 10;
```

```
do {
    cout << i << " ";
    n++;
} while (i <= 5);
// 输出：10（虽然条件不满足，但至少执行了一次）
```

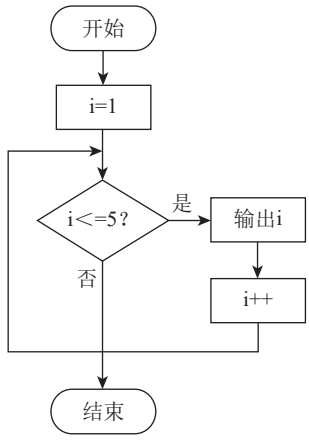


图 1-5　while 循环流程图

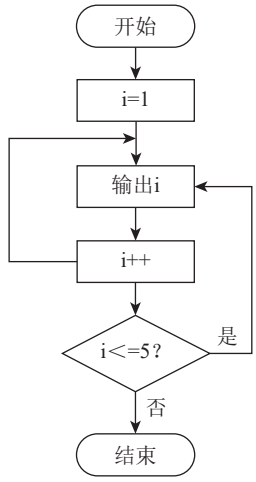


图 1-6　do-while 循环流程图

（3）for 循环（计数型循环）

流程图特征：将“初始化→判断条件→执行循环体→更新变量”4 个步骤集中管理，如图 1-7 所示。

```
for (int i = 1; i <= 5; i++) {
    cout << i << " ";
}
// 输出：1 2 3 4 5
```

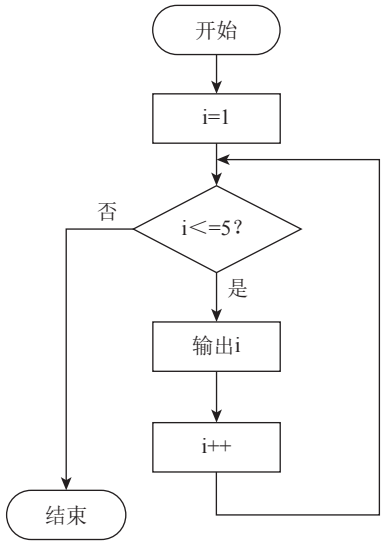


图 1-7　for 循环流程图

1.1.3 案例演示

1. 计算两数之和

题目描述：输入 2 个整数 a 和 b，输出它们的和。并用流程图描述求解的算法。

输入描述：1 行 2 个整数 a 和 b，用空格分隔。

输出描述：1 行 1 个整数，表示 a + b 的结果。

输入样例：

```
3 5
```

输出样例：

```
8
```

2. 判断奇偶性

题目描述：输入 1 个整数 n，判断它是奇数还是偶数。

输入描述：1 行 1 个整数 n。

输出描述：如果 n 是偶数，输出 even；如果是奇数，输出 odd。

输入样例：

```
4
```

输出样例：

```
even
```

3. 输出 1 到 n 的所有整数

题目描述：输入 1 个正整数 n，输出从 1 到 n 的所有整数，用空格分隔。

输入描述：1 行 1 个正整数 n（$1 \leqslant n \leqslant 100$）。

输出描述：1 行 n 个整数，用空格分隔。

输入样例：

```
5
```

输出样例：

```
1 2 3 4 5
```

4. 统计正数个数

题目描述：输入 n 个整数，统计其中正数的个数。

输入描述：第 1 行 1 个整数 n（$1 \leqslant n \leqslant 100$）。 第 2 行 n 个整数，用空格分隔。

输出描述：1 行 1 个整数，表示正数的个数。

输入样例：

```
5
3 -2 0 5 -1
```

输出样例：

```
2
```

5. 求最大值

题目描述：输入 n 个整数，找出其中的最大值。

输入描述：第 1 行 1 个整数 n（1 ≤ n ≤ 1000）。 第 2 行 n 个整数，用空格分隔。

输出描述：1 行 1 个整数，表示最大值。

输入样例：

```
5
3 7 2 9 5
```

输出样例：

```
9
```

1.1.4　AI 辅助学习

1. 流程图绘制辅助

向 AI 提问“请帮我画出求 1 ～ 100 之和的流程图”，让 AI 生成标准的流程图描述，对照学习。

2. 代码转流程图

将自己写的代码发给 AI，提问“请将这段代码转换为流程图”，检验自己对程序逻辑的理解是否正确。

1.1.5　活学活用

1. 下列关于流程图的说法，错误的是（　）。

A. 流程图必须有且仅有一个“开始”　　B. 流程图可以有多个“结束”

C. 判断框必须有两个出口　　D. 流程线可以不标注箭头方向

2. 在流程图中，用于表示输入、输出操作的图形是（　）。

A. 矩形　　B. 菱形

C. 平行四边形　　D. 椭圆形

3. 流程图 1（图 1-8）属于计算机的（　）。

A. 顺序结构　　B. 循环结构

C. 分支结构　　D. 数据结构

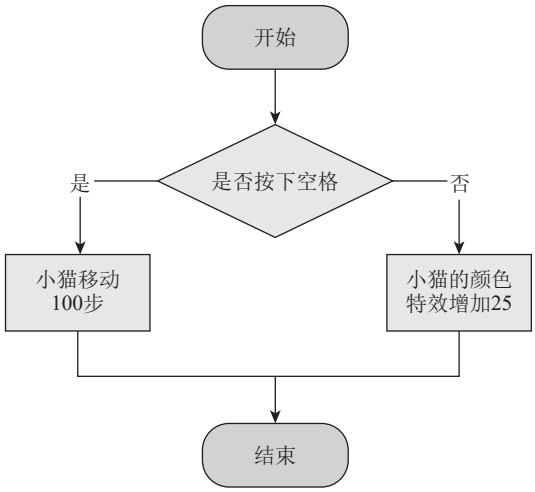


图 1-8　流程图 1

4. 流程图 2（图 1-9）的输出结果是（　）。

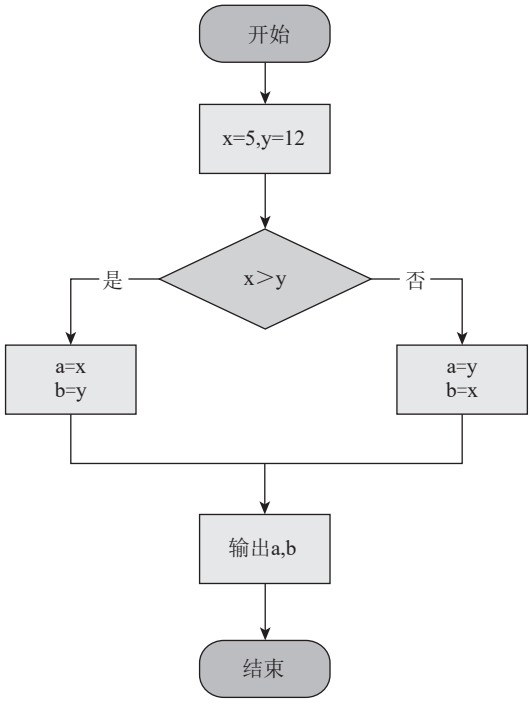


图 1-9　流程图 2

A. 5, 12　　B. 12, 5　　C. 5, 5　　D. 12, 12

5. 当流程图 3（图 1-10）中的 yr=2024 时，可以判断 yr 为闰年，并输出 2 月是 29 天，则图中菱形

框中应该填入（　）。

A. (yr%400==0) || (yr%4==0)

B. (yr%400==0) || (yr%4==0 && yr%100!=0)

C. (yr%400==0) && (yr%4==0)

D. (yr%400==0) && (yr%4==0 && yr%100!=0)

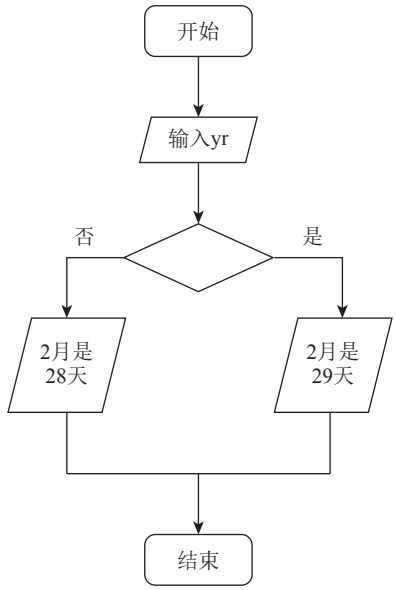


图 1-10　流程图 3

6. 观察流程图 4（图 1-11），下列（　）段 C++ 代码片段能最准确地实现该流程图的逻辑。

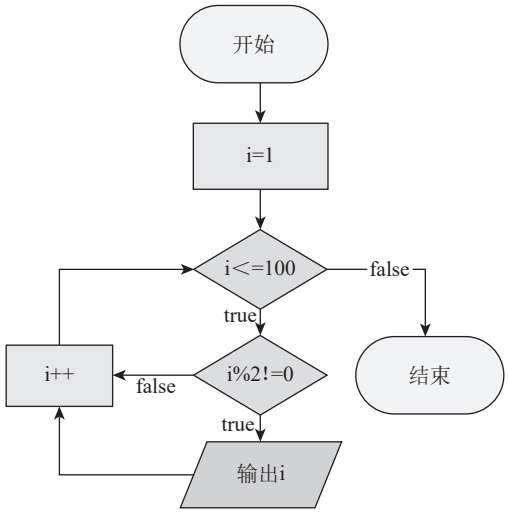


图 1-11　流程图 4

A.

```
int i = 1;
while(i <= 100) {
    if(i % 2 != 0) {
        cout << i << endl;
    }
}
```

B.

```
for(int i = 1; i <= 100; i += 2) {
    cout << i << endl;
}
```

C.

```
for(int i = 1; i <= 100; i++) {
    if(i % 2 != 0) {
        cout << i << end1;
    }
}
```

D.

```
for(int i = 1; i <= 100; i++) {
    cout << i << end1; }
```

7. 在流程图 5（图 1-12）中，当这个循环结束时，变量 i 的最终值是（　）。

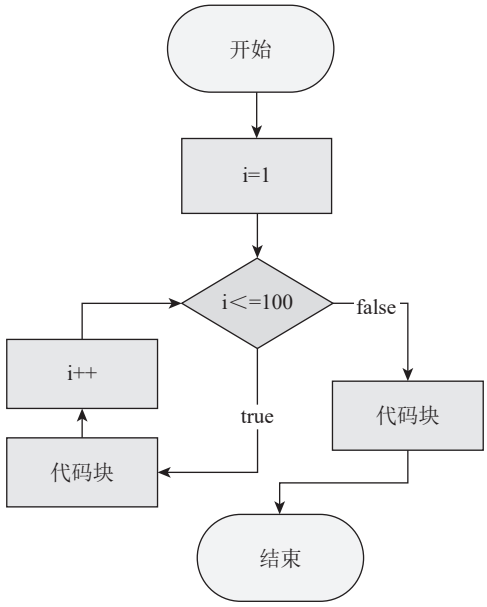


图 1-12　流程图 5

A. 99　　B. 100　　C. 101　　D. 102