一、程序基本语句

【大纲内容】

【难度 2】cin 语句、scanf 语句、cout 语句、printf 语句、赋值语句、复合语句

【难度 2】if 语句、switch 语句、多层条件语句

【难度 2】for 语句、while 语句、do while 语句

【难度 3】多层循环语句

（一）输入、输出、赋值语句

1、cin 语句（C++的输入语句）

cin 是 C++ 中的输入语句，表示**从键盘输入数据，把它存进变量**。

格式：cin >> 变量;

就像是你告诉电脑“我的年龄是 10 岁”，电脑会用漏斗把你的答案“倒进变量盒子”。

☑示例：

```
#include <iostream>
using namespace std;
int age;
cin >> age;
```

- 说明：当你在运行程序时，在键盘上输入一个数字，比如 10，它就会存进 age。

2、scanf 语句（C 语言的输入语句）

scanf 是 C 语言的输入方式，用**格式控制符**接收输入，如 %d 表示输入整数。

格式：scanf("格式", &变量);

就像你告诉老师“请填空”，而格式 %d 就是“我填的是整数”。

☑示例：

```
#include <stdio.h>
int age;
scanf("%d", &age);
```

- 注意：C 语言要加 **&** 表示“给出地址”，告诉电脑存到哪里。

3、cout 语句（C++的输出语句）

cout 是 C++ 中的输出语句，用来**显示变量或文字到屏幕上**。

格式： cout << 内容;

就像你大声告诉别人：“我叫小明！”程序里的 cout 就是帮你“说出来”。

☑示例：

```
cout << "你好，世界！" << endl;
```

- endl 表示“换行”（end of line）。

4、printf 语句（C 语言的输出语句）

printf 是 C 语言的输出方式，通过格式控制符来显示变量内容。

格式： printf("格式"，变量);

就像写作文填空：“我的年龄是__”，程序用 %d 来占位。

☑示例：

```
int age = 10;
printf("我的年龄是 %d 岁\n", age);
```

- %d 表示输出整数，\n 表示换行。

5、赋值语句（Assignment Statement）

赋值语句是把一个值**存进变量**里，用等号 = 来完成。

格式： 变量 = 值;

就像你拿一个空盒子（变量），把苹果（值）放进去。

☑ **示例：**

```
int x;
x = 5;
```

你也可以在定义时直接赋值：

```
int x = 5;
```

- 注意：程序里的 = 不是“等于”的意思，而是“放进去”的意思。

6、复合语句（Compound Statement）

复合语句就是用**大括号 {} 把好几行语句包起来**，让它们一起执行，组成一个“**代码块**”。

就像妈妈给你一张“任务清单”，上面有三件事：起床、洗脸、吃饭。你要一起做完，才算完成。

格式：

```
{
  语句 1;
  语句 2;
  语句 3;
}
```

☑ **示例：**

```
{
    int a = 3;
    int b = 4;
    cout << "和是：" << a + b << endl;
}
```

这三行语句包在 {} 中，会**作为一个整体执行**。

总结表格

名称（中英文）	功能	用法示例
cin（输入）	从键盘输入数据	cin >> age;
scanf（输入）	用格式读取输入	scanf("%d", &age);
cout（输出）	显示内容到屏幕	cout << "你好";
printf（输出）	按格式输出	printf("年龄是%d", age);
赋值语句（Assignment）	给变量“装上值”	x = 5;
复合语句（Compound）	多条语句打包一起执行	{ int a=1; int b=2; cout<<a+b; }

C++ scanf 和 printf 格式控制符大全

格式控制符	用途说明	适用数据类型	示例（printf）	示例（scanf）
%d	十进制整数	int	printf("%d", 42);	scanf("%d", &num);
%u	无符号十进制整数	unsigned int	printf("%u", 100);	scanf("%u", &unum);
%ld	长整型十进制	long	printf("%ld", 1000L);	scanf("%ld", &l);
%lld	长长整型十进制	long long	printf("%lld", 1000000LL);	scanf("%lld", &ll);
%f	单精度浮点数	float	printf("%f", 3.14f);	scanf("%f", &f);
%lf	双精度浮点数	double	printf("%lf", 3.14);	scanf("%lf", &d);
%c	单个字符	char	printf("%c", 'A');	scanf("%c", &ch);
%s	字符串	char[], char*	printf("%s", "Hello");	scanf("%s", str);
%%	输出%字符	-	printf("%%");	-

（二）条件语句（Conditional Statements）

条件语句是程序中用来**“做判断”和“选路径”**的语句，像一个聪明的大脑，能根据不同情况选择不同做法。

就像你每天早上起床，会判断：“今天下雨吗？下雨就带伞；不下雨就骑车”，这就是在生活中的条件语句。

1、if 语句（if Statement）

if 语句用来**判断某个条件是否为真（true），如果为真，就执行后面的语句**。

语法结构

```
if (条件) {
    // 条件为真时做的事
}
```

就像妈妈说：“如果你考了 100 分，我就带你去吃冰淇淋”。

☑ **示例代码**：

```
int score = 95;
if (score >= 90) {
    cout << "你真棒，得了优！" << endl;
}
```

说明：

- score >= 90 是判断条件；
- 只有这个条件“为真”，程序才会执行 {} 中的内容。

☑ **扩展：if...else**

```
if (score >= 90) {
    cout << "你得了优！" << endl;
```

```
} else {
    cout << "继续加油哦！" << endl;
}
```

2、switch 语句（switch Statement）

switch 是根据变量的值，**从多个选项中选择一个执行**。每个选项称为一个 case。

语法结构：

```
switch (变量) {
    case 值 1:
        // 做某事
        break;
    case 值 2:
        // 做其他事
        break;
    default:
        // 如果都不对，就做这个
}
```

类比说明：

像你去食堂点餐：

- 输入 1，吃汉堡；
- 输入 2，吃披萨；
- 输入 3，吃面条；
- 其他数字，吃家常便饭。

☑ 示例代码：

```
int choice = 2;
```

```
switch (choice) {
    case 1:
        cout << "你选择了汉堡。" << endl;
        break;
    case 2:
        cout << "你选择了披萨。" << endl;
        break;
    case 3:
        cout << "你选择了面条。" << endl;
        break;
    default:
        cout << "今天就吃家常便饭吧。" << endl;
}
```

- break 表示**跳出**这个判断块，防止继续往下执行。

3、多层条件语句（多重 if-else）

多层条件语句是指**连续判断多个条件**，每个条件都不同，根据不同条件做不同事。

语法结构：

```
if (条件 1) {
    // 做 A 事
} else if (条件 2) {
    // 做 B 事
} else if (条件 3) {
    // 做 C 事
} else {
    // 都不是，就做 D 事
```

```
}
```

类比说明：

就像根据成绩打等级：

- 90 分以上是“A”
- 80~89 分是“B”
- 70~79 分是“C”
- 否则就是“D”

☑ 示例代码：

```
int score = 82;
if (score >= 90) {
    cout << "等级：A" << endl;
} else if (score >= 80) {
    cout << "等级：B" << endl;
} else if (score >= 70) {
    cout << "等级：C" << endl;
} else {
    cout << "等级：D" << endl;
}
```

- 程序从上往下判断，只执行第一个为真的条件。

总结表格

语句	功能	用法关键字	使用示例
if 语句	判断条件是否成立	if (条件)	if (x > 5)
if...else 语句	条件为假时执行其他语句	if ... else	if (x>5){...} else {...}

switch 语句	多选一判断（**值固定**）	switch, case, break	switch(num) { case 1: ...}
多层条件语句	连续判断多个条件	else if	if (...) else if (...) ...

（三）循环语句

什么是循环语句（Loop Statement）？

循环语句是程序中用来**重复**执行某一段代码的工具。

当我们希望同样的事情做很多遍（比如做 10 次数学题），就用“循环”。

就像你妈妈说：“把这张试卷做 3 遍”，你就得重复写 3 遍，程序也一样，它可以自动“做很多遍事情”。

1、for 语句（for Statement）

for 是循环次数确定时最常用的循环结构。你可以设置：从哪里开始、到哪里结束、每次怎么走。

语法结构：

```
for (起点; 条件; 步长) {
    // 重复执行的内容
}
```

类比说明：

就像老师说：“请写 5 遍‘我爱学习’”。你从第 1 遍写到第 5 遍，每次写完自动+1 次。

☑ 示例代码：

```
for (int i = 1; i <= 5; i++) {
    cout << "第" << i << "次说：我爱学习！" << endl;
}
```

说明：

- int i = 1; → 从 1 开始;
- i <= 5; → 当 i 小于等于 5 就继续;
- i++ → 每次加 1;
- 共执行 5 次。

2、while 语句（while Statement）

while 是**先判断条件，再决定是否执行**。只要条件满足，就一直做。

语法结构：

```
while (条件) {
    // 条件为真时重复做的事
}
```

类比说明：

- 你妈妈说：“只要你作业没写完，就不许玩”。
- 你就一直写写写……直到写完为止。

☑示例代码：

```
int i = 1;
while (i <= 3) {
    cout << "正在写第" << i << "题作业" << endl;
    i++;
}
```

执行过程：

- i = 1，满足条件 → 执行;
- i = 2，继续;
- i = 4，不满足，停止。

3、do while 语句（do while Statement）

do while 是先做一遍，再判断条件。哪怕条件一开始就不成立，也会至少执行一次。

语法结构：

```
do {
    // 一定会先做一次
} while (条件);
```

类比说明：

老师说："至少写一篇作文，如果还有时间就再写一篇"，那你一定会先写一篇，条件再判断。

☑ 示例代码：

```
int i = 1;
do {
    cout << "练习第" << i << "遍舞蹈" << endl;
    i++;
} while (i <= 3);
```

会输出：

- 练习第 1 遍舞蹈
- 练习第 2 遍舞蹈
- 练习第 3 遍舞蹈

总结对比表格

类型	中文名	判断方式	是否至少执行 1 次	示例用途
for 循环	计数型循环	**先给定次数**	不一定	做 10 道题，跑 5 圈
while 循环	条件型循环	**先判断，再做**	不一定	直到写完作业才休息

do while 循环	至少一次循环	**先做，再判断**	至少做一次	至少唱一次才结束

☑综合例子：三种循环说“我在学习”

用 for：

```
for (int i = 1; i <= 3; i++) {
    cout << "第" << i << "次说：我在学习！" << endl;
}
```

用 while：

```
int i = 1;
while (i <= 3) {
    cout << "第" << i << "次说：我在学习！" << endl;
    i++;
}
```

用 do while：

```
int i = 1;
do {
    cout << "第" << i << "次说：我在学习！" << endl;
    i++;
} while (i <= 3);
```

4、什么是多层循环语句？（Nested Loop，多重循环）

多层循环语句是指**一个循环里面又嵌套另一个循环**，就像在循环中再进行循环。

想象你在画“九宫格乘法表”：

- 第一层循环控制行数（1 到 9）

- 第二层循环控制列数（每行的 1 到 9）
- 每一行里面，都要写很多列 → 这就是多层循环！

就像你妈妈说：

- “做 5 页作业，每页写 10 道题”
- → 外面 5 页是外层循环，每页的 10 道题是内层循环！

基本结构：

```
for (int i = 1; i <= 外层次数; i++) {
    for (int j = 1; j <= 内层次数; j++) {
        // 重复的动作
    }
}
```

☑示例 1：输出 3 行 2 列的“我在学习”

```
for (int i = 1; i <= 3; i++) { // 外层循环控制 3 行
    for (int j = 1; j <= 2; j++) { // 内层循环控制 2 次
        cout << "第" << i << "行，第" << j << "次：我在学习！" << endl;
    }
}
```

输出结果：

```
第 1 行，第 1 次：我在学习！
第 1 行，第 2 次：我在学习！
第 2 行，第 1 次：我在学习！
第 2 行，第 2 次：我在学习！
第 3 行，第 1 次：我在学习！
```

```
第 3 行，第 2 次：我在学习！
```

第 1 行，第 1 列	第 1 行，第 2 列
第 2 行，第 1 列	第 2 行，第 2 列
第 3 行，第 1 列	第 3 行，第 2 列

☑示例 2：输出九九乘法表

```
for (int i = 1; i <= 9; i++) {
    for (int j = 1; j <= i; j++) {
        cout << j << "×" << i << "=" << i * j << "\t";
    }
    cout << endl;
}
```

说明：

- i 是每一行的数字（1 到 9）
- j 是每一行的乘数
- i * j 是答案

结果就是我们熟悉的九九乘法表！

☑示例 3：输出小星星（矩阵）

```
for (int i = 1; i <= 4; i++) {          // 控制 4 行
    for (int j = 1; j <= 5; j++) {     // 控制每行 5 个星星
        cout << "* ";
    }
    cout << endl; // 每行结束换行
}
```

输出：

```
* * * * *
* * * * *
* * * * *
* * * * *
```

多层循环能做什么？

应用场景	示例
打印乘法表	九九乘法表
打印表格	例如成绩表、矩阵
画图形/星星	小正方形、三角形
批量操作任务	每位学生做多个任务
模拟比赛轮次	每个队和每个队都比一场

多层循环流程图说明

- 外层循环 Outer Loop：每一行

- 内层循环 Inner Loop：每一行中的每一项

外层第 1 次：

内层 1、2、3……

外层第 2 次：

内层 1、2、3……

……