第 1 讲 数据类型的转换和数位分离

■ 自动类型转换（隐式类型转换）

基本概念

定义：编译器自动进行的类型转换，无需程序员显式指定

时机：不同类型数据混合运算、赋值、函数传参时

转换规则

低精度类型 → 高精度类型

char → bool → int → long long→ double

算术运算中的自动转换

```
int a = 10;
double b = 3.14;
cout << a + b;  // int自动转为double
// a转换为double(10.0)，结果为13.14
```

赋值时的自动转换

```
int x = 5;
double y = x;          // int自动转为double，y=5.0
double pi = 3.14159;
int approx = pi;       // double自动转为int，approx=3（截断小数）
```

字符与整数的转换

```
char c = 'A';
int ascii = c;         // char自动转为int，ascii=65

int num = 66;
char ch = num;         // int自动转为char，ch='B'
```

■ 强制类型转换（显式类型转换）

仅临时改变当前数据类型，不改变原变量的类型 / 存储值

```
double pi = 3.14159;
int intPi = (int)pi;              // 3（截断小数部分）C风格

char c = 'A';
int code = int(c);                // 65（字符转ASCII码）C++风格
```

■ 运算符优先级总览（从高到低）

优先级	运算符
1	!
2	* / %
3	+-
4	> >= < <=
5	== !=
6	&&
7	||

逻辑运算符的取反规则：

“拆括号，分负号，且变或，或变且”

1）拆掉整个表达式外面的否定号 (!) 和括号 ()。

2）将否定号 (!) 分配给括号内的每一个变量。

3）同时将括号内的逻辑运算符翻转：&& 变成 ||，|| 变成 &&。

■ 变量（标识符）命名规则

1）首字符不能以数字开头。

2）只能包含字母、数字或下划线。

3）区分大小写

4）不能使用关键字

关键词	关键词	关键词	关键词	关键词
asm	do	if	return	typedef
auto	double	inline	short	typeid
bool	dynamic_cast	int	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	export	new	struct	virtual
class	extern	operator	switch	void
const	FALSE	private	template	volatile
const_cast	float	protected	this	wchar_t
continue	for	public	throw	while
default	friend	register	TRUE	
delete	goto	reinterpret_cast	try	

■ ASCII 编码

定义：美国标准信息交换码
范围：128(大小为 0 ~ 127)个字符（7 位二进制）
包含：英文字母、数字、控制字符
核心规则：char 参与运算时，自动转为对应 ASCII 码的 int 值

字符常用操作：

操作需求	实现方法	示例
数字字符转整型	数字字符 - '0'	int a = '9' - '0'；(a=9)
大写字母转小写	大写字母 + 32	char b = 'A' + 32；(b='a')
小写字母转大写	小写字母 - 32	char c = 'b' - 32；(c='B')
字母循环移位（n 位后）	小写：(ch1-'a'+n)%26 + 'a' 大写：(ch2-'A'+n)%26 + 'A'	char d = ('a'-'a'+27)%26 + 'a'；(d='b')

■ 常见数据类型范围

数据类型	存储空间	数据范围
int	4Byte	$-2^{31} \sim 2^{31}-1$（约±21 亿）
long long	8Byte	$-2^{63} \sim 2^{63}-1$（更大整数范围）
double	8Byte	约 $2.22507e-308 \sim 1.79769e+308$（浮点数）
char	1Byte	$-126 \sim 125$
bool	1Byte	false(0)、true(1)

数据溢出：大数值赋值给小类型变量（超出范围），高位被截断，导致数据丢失（如 int a=4294967296;超出 int 范围）

■ 数位分离（整数）

基础：数位分离（逐位提取）
功能：从末位开始，依次提取整数的每一位数字

```
int main() {
    int num;
    cin >> num;
    // 核心循环：分离每一位
    while (num != 0) {
        int d = num % 10; // 1. 取当前末位数字
        cout << d << " "; // 2. 使用当前位（可替换为其他操作）
        num = num / 10; // 3. 去掉已处理的末位
    }
    return 0;
}
// 输入：12345 → 输出：5 4 3 2 1
// 输入：678 → 输出：8 7 6
```

常用拓展 1：求数字的位数
功能：计算整数包含的数字个数（0 的位数为 1）

```
int main() {
    int num;
    cin >> num;
    if (num == 0) {
        cout << 1;
        return 0; // 特殊情况：0的位数是1
    }
    int count = 0;
    while (num != 0) {
        count++; // 每循环一次，位数+1
        num = num / 10; // 去末位
    }
    cout << "位数：" << count << endl;
    return 0;
}
// 输入：12345 → 输出：5
// 输入：0 → 输出：1
// 输入：987 → 输出：3
```

常用拓展 2：求数字的倒序数

功能：将整数的数字顺序反转（如 12345→54321）

```
int main() {
    int num;
    cin >> num;
    int reverse = 0;
    while (num != 0) {
        int d = num % 10; // 取末位
        reverse = reverse * 10 + d; // 拼接末位到反转数
        num = num / 10; // 去末位
    }
    cout << "倒序数: " << reverse << endl;
    return 0;
}
// 输入: 12345 → 输出: 54321
// 输入: 678 → 输出: 876
// 输入: 100 → 输出: 1 (末尾0反转后省略)
```

常用拓展 3：求数位之和

功能：计算整数所有数字的累加和（如 123→1+2+3=6）

```
int main() {
    int num;
    cin >> num;
    int sum = 0;
    while (num != 0) {
        sum += num % 10; // 累加末位数字
        num = num / 10; // 去末位
    }
    cout << "数位之和: " << sum << endl;
    return 0;
}
// 输入: 12345 → 输出: 15
// 输入: 987 → 输出: 24 (9+8+7)
// 输入: 0 → 输出: 0
```