

C++学习笔记——入门基础

赠自己：每天坚持学习一点点：

1、C++初识

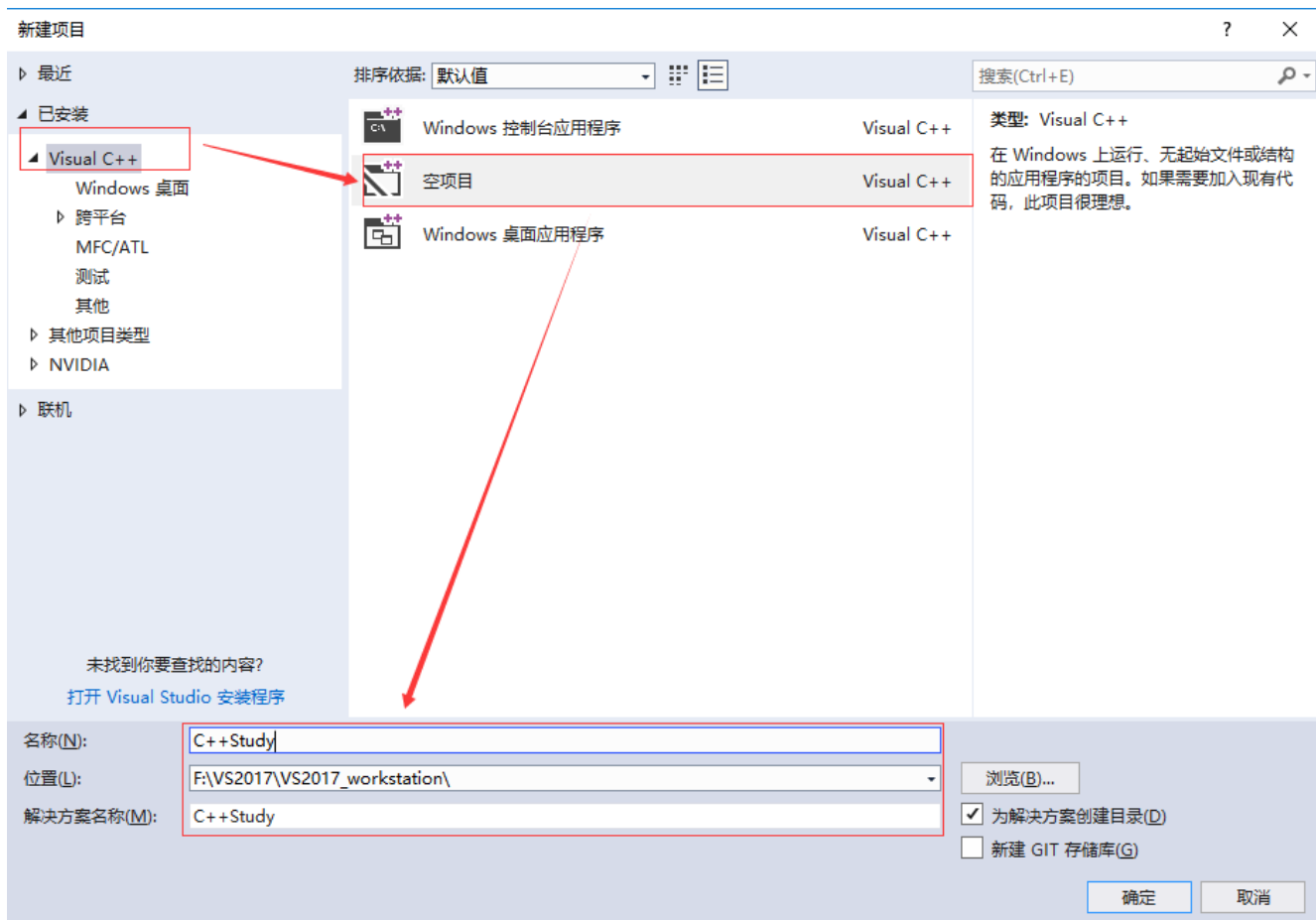
1.1 第一个c++程序

编写一个C++程序总共分为4个步骤

- 创建项目
- 创建文件
- 编写代码
- 运行程序

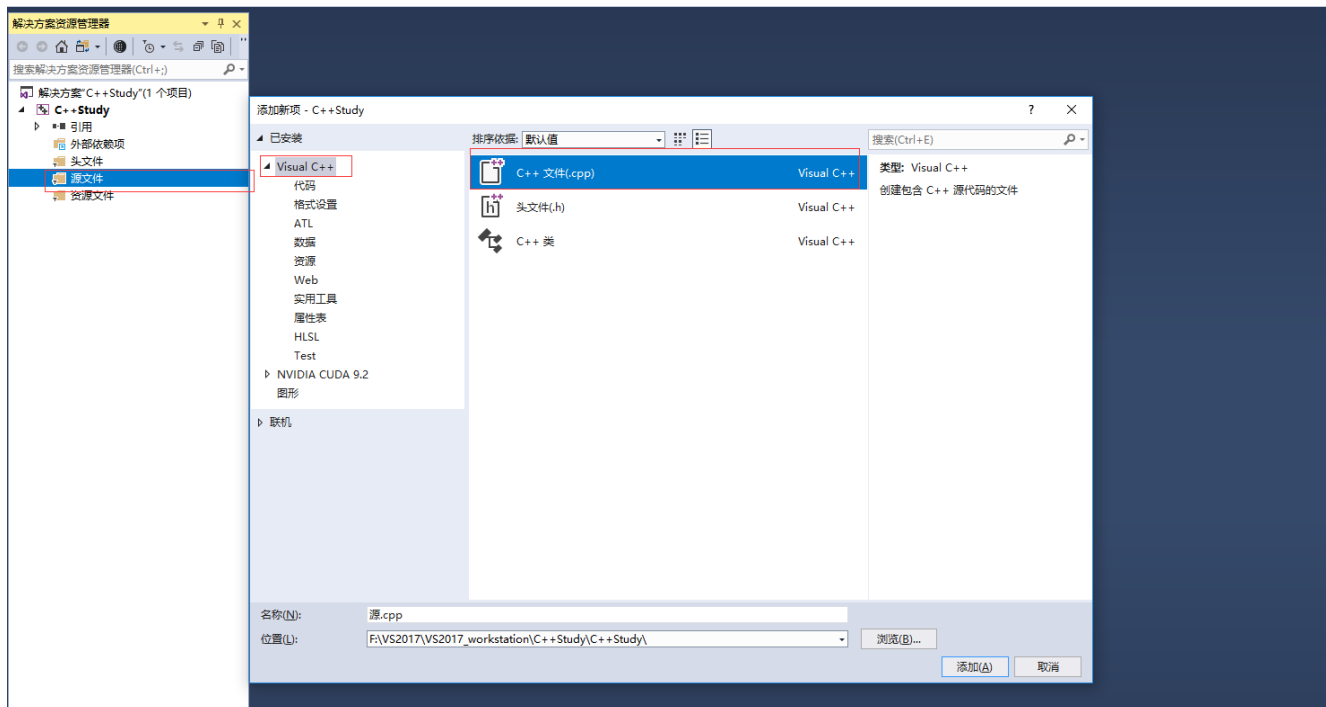
1.1.1 创建项目





1.1.2 创建文件

右键源文件, 选择添加->新建项



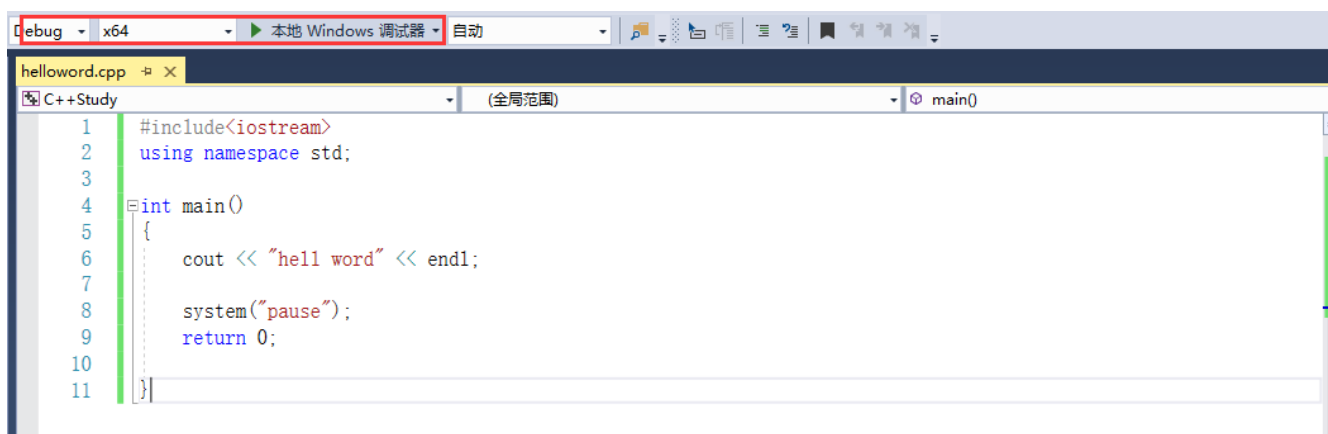
1.1.3 编写代码

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "hell word" << endl;
7
8      system("pause");
9      return 0;
10
11 }

```

1.1.4 运行程序



1.2 注释

作用：在代码中加一些说明和解释，方便自己或其他程序员程序员阅读代码

两种格式

1. **单行注释：** `// 描述信息`

- 通常放在一行代码的上方，或者一条语句的末尾，**对该行代码说明**

2. **多行注释：** `/* 描述信息 */`

- 通常放在一段代码的上方，**对该段代码做整体说明**

提示：编译器在编译代码时，会忽略注释的内容

1.3 变量

变量存在的意义：方便我们管理内存空间

作用：给一段指定的内存空间起名，方便操作这段内存

语法： `数据类型 变量名 = 初始值;`

示例：

```

1  #include<iostream>
2  using namespace std;

```

```

3
4 //1、单行注释
5 //2、多行注释
6 /**
7     main是一个程序的入口
8     每个程序都必须有这么一个函数
9     有且仅有一个
10 */
11
12 int main()
13 {
14
15     // 变量创建的语法: 数据类型 变量名 = 变量的初始值
16     int a = 10;
17     cout << "a=" << a << endl;
18     cout << "hell word" << endl;
19     system("pause");
20     return 0;
21
22 }

```

注意: C++在创建变量时, 必须给变量一个初始值, 否则会报错

1.4 常量

作用: 用于记录程序中不可更改的数据

C++定义常量两种方式

1. **#define** 宏常量: `#define 常量名 常量值`
 - 通常在文件上方定义, 表示一个常量
2. **const**修饰的变量 `const 数据类型 常量名 = 常量值`
 - 通常在变量定义前加关键字const, 修饰该变量为常量, 不可修改

示例:

```

1 #include<iostream>
2 using namespace std;
3
4 // 常量的定义方式
5 // 1、#define 宏常量
6 // 2、const修饰的变量
7
8 // 1、#define 宏常量
9 #define Day 7 // 定义一个宏常量day
10
11 int main()
12 {
13     // Day = 14; // 错误, Day是常量, 一旦修改就会报错
14     cout << "一周总共有" << Day << "天" << endl;

```

```

15
16 // 2、const修饰的变量
17 const int month = 12; // const修饰的变量也称为常量，一旦定义，无法修改
18 cout << "一年总有：" << month << "个月份" << endl;
19
20 system("pause");
21 return 0;
22 }

```

1.5 关键字

作用：关键字是C++中预先保留的单词（标识符）

- 在定义变量或者常量时候，不要用关键字

C++关键字如下：

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	

提示：在给变量或者常量起名称时候，不要用C++得关键字，否则会产生歧义。

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      // 创建变量： 数据类型， 变量名称 = 变量初始值
7      // 不要用关键字给变量或者常量起名称
8      // int int = 10; 错误，第二个int是关键字，不可以作为变量的名称
9      system("pause");
10     return 0;
11 }
12

```

1.6 标识符命名规则

作用：C++规定给标识符（变量、常量）命名时，有一套自己的规则

- 标识符不能是关键字
- 标识符只能由字母、数字、下划线组成
- 第一个字符必须为字母或下划线
- 标识符中字母区分大小写

建议：给标识符命名时，争取做到见名知意的效果，方便自己和他人的阅读

```

1  #include<iostream>
2  using namespace std;
3
4  // 标识符命名规则
5  // 1、标识符不可以是关键字
6  // 2、标识符是由字母、数字、下划线构成
7  // 3、第一个字符只能是字母或者下划线
8  // 4、标识符是区分大小写的
9
10 int main()
11 {
12     // 1、标识符不可以是关键字
13     // int int = 10;
14
15     // 2、标识符是由字母、数字、下划线构成
16     int abc = 10;
17     int _abc = 20;
18
19     // 3、第一个字符只能是字母或者下划线
20     // int 45abc = 50; //这是一个错误的
21
22     // 4、标识符是区分大小写的
23     int aaa = 10;
24     int AAA = 100;
25     cout << "aaa" << aaa << endl;
26
27     // 建议：给变量起名的时候，最后能够做到见名知意
28     int num1 = 10;

```

```

29     int num2 = 20;
30     int sum = num1 + num2;
31
32     system("pause");
33     return 0;
34 }

```

2、数据类型

C++规定在创建一个变量或者常量时，必须要指定出相应的数据类型，否则无法给变量分配内存

2.1 整型

语法：数据类型 变量名 = 变量初始值

```
int a= 10;
```

数据类型存在的意义：给变量分配合适的内存空间

作用：整型变量表示的是整数类型的数据

C++中能够表示整型的类型有以下几种方式，区别在于所占内存空间不同：

数据类型	占用空间	取值范围
short(短整型)	2字节	$(-2^{15} \sim 2^{15}-1)$
int(整型)	4字节	$(-2^{31} \sim 2^{31}-1)$
long(长整形)	Windows为4字节，Linux为4字节(32位)，8字节(64位)	$(-2^{31} \sim 2^{31}-1)$
long long(长长整形)	8字节	$(-2^{63} \sim 2^{63}-1)$

```

1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      // 整型
6      // 1、短整型(-32768~32767),如果超出最大值，则返回到最小值-32768
7      short num1 = 10;
8
9      // 2、整型
10     int num2 = 10;
11
12     // 3、长整型
13     long num3 = 10;
14
15     // 4、长长整型
16     long long num4 = 10;
17
18     cout << "num1 = " << num1 << endl;

```

```

19     cout << "num2 = " << num2 << endl;
20     cout << "num3 = " << num3 << endl;
21     cout << "num4 = " << num4 << endl;
22
23     system("pause");
24     return 0;
25 }

```

如果没有特殊要求，int型通常够用了。

2.2 sizeof关键字

作用：利用sizeof关键字可以统计数据类型所占内存大小

语法： sizeof(数据类型 / 变量)

示例：

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      // 整型: short(2)   int(4)       long(4)       long long(8)
7      // 可以利用sizeof求出数据类型占用内存大小
8      // 语法:  sizeof (数据类型/变量)
9      short num1 = 10;
10     cout << "short 类型所占内存空间: " << sizeof(short) << endl;
11
12     int num2 = 10;
13     cout << "int 类型所占内存空间: " << sizeof(int) << endl;
14
15     long num3 = 10;
16     cout << "long 类型所占内存空间: " << sizeof(long) << endl;
17
18     long long num4 = 10;
19     cout << "long long 类型所占内存空间: " << sizeof(long long) << endl;
20
21     system("pause");
22     return 0;
23 }

```

整型结论： short < int <= long <= long long

2.3 实型（浮点型）

作用：用于表示小数

浮点型变量分为两种：

1. 单精度float

2. 双精度double

两者的区别在于表示的有效数字范围不同。

数据类型	占用空间	有效数字范围
float	4字节	7位有效数字
double	8字节	15 ~ 16位有效数字

示例：

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      // 1、单精度 float
7      // 2、双精度 double
8      // 默认情况下，输出一个小数，会显示出6位有效数字
9
10     float f1 = 3.1415926f; // 多写一个f，代表float，如果不写，编辑器会认为是double
11     cout << "f1=" << f1 << endl;
12
13     double d1 = 3.1415926;
14     cout << "d1=" << d1 << endl;
15
16     // 统计float和double占用内存空间
17
18     cout << "float占用的内存空间为：" << sizeof(float) << endl; // 4字节
19     cout << "double占用的内存空间为：" << sizeof(double) << endl; // 8字节
20
21     // 科学计数法
22     float f2 = 3e2; // 3*10^2
23     cout << "f2=" << f2 << endl;
24
25     float f3 = 3e-2; // 3*0.1^2
26     cout << "f3=" << f3 << endl;
27
28     system("pause");
29     return 0;
30 }
```

2.4 字符型

作用：字符型变量用于显示单个字符

语法：char ch = 'a';

注意1：在显示字符型变量时，用单引号将字符括起来，不要用双引号

注意2：单引号内只能有一个字符，不可以是字符串

- C和C++中字符型变量只占用1个字节。
- 字符型变量并不是把字符本身放到内存中存储，而是将对应的ASCII编码放入到存储单元

示例：

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      // 1、字符型变量创建方式
7      char ch = 'a';
8      cout << ch << endl;
9      // 2、字符型变量所占内存大小
10     cout << "char字符型变量所占内存大小: " << sizeof(char) << endl; //字符所占内存大小
11     // 3、字符型变量常见错误
12     char ch2 = 'b';
13     //char ch3 = "b"; 创建字符型变量时候，要用单引号
14     //char ch4 = 'abc'; //创建字符型变量时候，单引号只能有一个字符
15
16     // 4、字符型变量对应ASCII编码
17     // a --97
18     // A --65
19     cout << "(int)ch对应的ASCII编码: " << (int)ch << endl;
20
21     system("pause");
22     return 0;
23 }
```

ASCII码表格：

ASCII值	控制字符	ASCII值	字符	ASCII值	字符	ASCII值	字符
0	NUT	32	(space)	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(72	H	104	h
9	HT	41)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[123	{

ASCII值	控制字符	ASCII值	字符	ASCII值	字符	ASCII值	字符
28	FS	60	<	92	/	124	
29	GS	61	=	93]	125	}
30	RS	62	>	94	^	126	`
31	US	63	?	95	_	127	DEL

ASCII 码大致由以下**两部分**组成：

- ASCII 非打印控制字符：ASCII 表上的数字 **0-31** 分配给了控制字符，用于控制像打印机等一些外围设备。
- ASCII 打印字符：数字 **32-126** 分配给了能在键盘上找到的字符，当查看或打印文档时就会出现。

2.5 转义字符

作用：用于表示一些不能显示出来的ASCII字符

现阶段我们常用的转义字符有：`\n \\ \t`

转义字符	含义	ASCII码值（十进制）
<code>\a</code>	警报	007
<code>\b</code>	退格(BS)，将当前位置移到前一位	008
<code>\f</code>	换页(FF)，将当前位置移到下页开头	012
<code>\n</code>	换行(LF)，将当前位置移到下一行开头	010
<code>\r</code>	回车(CR)，将当前位置移到本行开头	013
<code>\t</code>	水平制表(HT)（跳到下一个TAB位置）	009
<code>\v</code>	垂直制表(VT)	011
<code>\\</code>	代表一个反斜线字符"\ "	092
<code>\'</code>	代表一个单引号（撇号）字符	039
<code>\"</code>	代表一个双引号字符	034
<code>\?</code>	代表一个问号	063
<code>\0</code>	数字0	000
<code>\ddd</code>	8进制转义字符，d范围0~7	3位8进制
<code>\xhh</code>	16进制转义字符，h范围0~9, a~f, A~F	3位16进制

示例：

1	<code>#include<iostream></code>
---	---------------------------------------

```

2  using namespace std;
3
4  int main()
5  {
6
7      // 转义字符
8      // 换行符 \n
9      cout << "hello world" << endl;
10     cout << "hello world\n";
11
12     // 反斜杠 \\
13
14     cout << "\\ "<<endl;
15
16     // 水平制表符 \t 作用可以整齐的输出数据
17     cout << "aaa\ttheword" << endl;
18
19     system("pause");
20     return 0;
21 }

```

2.6 字符串型

作用：用于表示一串字符

两种风格

1. **C风格字符串：** `char 变量名[] = "字符串值"`

示例：

```

1  #include<iostream>
2  #include<string> // 用C++风格字符串时候，要包含这个头文件
3  using namespace std;
4
5  int main()
6  {
7      //1、C风格字符串
8      // 注意事项1: char 字符串名 []
9      // 注意实现2: 等号后面 要用双引号，包含起来字符串
10     char str1[] = "hello world";
11     cout << str1 << endl;
12     system("pause");
13     return 0;
14 }

```

注意：C风格的字符串要用双引号括起来

1. **C++风格字符串：** `string 变量名 = "字符串值"`

示例:

```
1 #include<iostream>
2 #include<string> // 用C++风格字符串时候, 要包含这个头文件
3 using namespace std;
4
5 int main()
6 {
7
8     //2、C++风格字符串
9     //注意事项: 包含一个头文件: #include<string>
10    string str2 = "hello world";
11    cout << "str2=" <<str2 << endl;
12
13    system("pause");
14    return 0;
15 }
```

注意: C++风格字符串, 需要加入头文件 `#include<string>`

2.7 布尔类型 bool

作用: 布尔数据类型代表真或假的值

bool类型只有两个值:

- true --- 真 (本质是1)
- false --- 假 (本质是0)

bool类型占1个字节大小

示例:

```
1 #include<iostream>
2 using namespace std;
3
4 int main()
5 {
6
7     // 1、创建bool数据类型
8     bool flag = true; // true代表真
9     cout << flag << endl; //1
10
11    flag = false; // false代表真
12    cout << flag << endl; //0
13
14    // 本质上: 1代表真, 0代表假
15    // 2、查看bool类型所占内存空间
16    cout << "bool类型所占内存空间: " << sizeof(bool) << endl;
17    system("pause");
18    return 0;
```

2.8 数据的输入

作用：用于从键盘获取数据

关键字：cin

语法：cin >> 变量

示例：

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7
8      // 1、整型输入
9      int a = 0;
10     cout << "请给整型变量a赋值：" << endl;
11     cin >> a; //等待数据输入
12     cout << "整型变量a=" << a << endl;
13
14     // 2、浮点型输入
15     float f = 3.14f;
16     cout << "请给浮点型变量f赋值：" << endl;
17     cin >> f;
18     cout << "浮点型变量f=" << f << endl;
19
20     // 3、字符型输入
21     char ch = 'a';
22     cout << "请给字符型变量ch赋值：" << endl;
23     cin >> ch;
24     cout << "字符型变量ch=" << ch << endl;
25
26     // 4、字符串型输入
27
28     string str = "jiajikang";
29     cout << "请给字符串 str赋值" << endl;
30     cin >> str;
31     cout << "字符串变量str=" << str << endl;
32
33     // 5、布尔型输入
34
35     bool flag = false;
36     cout << "请给布尔类型flag赋值" << endl;
37     cin >> flag; //bool类型，只要是非0的值都代表真
38     cout << "布尔类型flag=" << flag << endl;
39
```

```
40 |     system("pause");
41 |     return 0;
42 | }
```

3、运算符

作用：用于执行代码的运算

本章我们主要讲解以下几类运算符：

运算符类型	作用
算术运算符	用于处理四则运算
赋值运算符	用于将表达式的值赋给变量
比较运算符	用于表达式的比较，并返回一个真值或假值
逻辑运算符	用于根据表达式的值返回真值或假值

3.1 算术运算符

作用：用于处理四则运算

算术运算符包括以下符号：

运算符	术语	示例	结果
+	正号	+3	3
-	负号	-3	-3
+	加	10 + 5	15
-	减	10 - 5	5
*	乘	10 * 5	50
/	除	10 / 5	2
%	取模(取余)	10 % 3	1
++	前置递增	a=2; b=++a;	a=3; b=3;
++	后置递增	a=2; b=a++;	a=3; b=2;
--	前置递减	a=2; b=--a;	a=1; b=1;
--	后置递减	a=2; b=a--;	a=1; b=2;

示例1：

```
1 | #include<iostream>
```



```

2  #include<string>
3  using namespace std;
4
5  // 加减乘除
6  int main()
7  {
8      int a1 = 10;
9      int b1 = 3;
10     cout << a1 + b1 << endl;
11     cout << a1 - b1 << endl;
12     cout << a1 * b1 << endl;
13     cout << a1 / b1 << endl; //两个整数相除结果依然是整数
14
15     int a2 = 10;
16     int b2 = 20;
17     cout << a2 / b2 << endl;
18
19     int a3 = 10;
20     int b3 = 0;
21     //cout <<a3/b3 <<endl; //报错，除数不可以是0
22
23     //两个小数可以相除
24     double d1 = 0.5;
25     double d2 = 0.25;
26     cout << d1 / d2 << endl;
27
28     system("pause");
29     return 0;
30 }

```

总结：在除法运算中，除数不能为0

示例2:

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  // 取模
6  int main()
7  {
8      int a1 = 10;
9      int b1 = 3;
10     cout << 10 % 3 << endl;
11
12
13     int a2 = 10;
14     int b2 = 20;
15     cout << a2 % b2 << endl;
16
17     int a3 = 10;
18     int b3 = 0;
19     //cout <<a3%b3 <<endl; //报错，除数不可以是0

```

```

20
21 //两个小数可以相除
22 double d1 = 0.5;
23 double d2 = 0.25;
24 //cout << d1 % d2 << endl;
25
26 system("pause");
27 return 0;
28 }

```

总结：只有整型变量可以进行取模运算

示例3:

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  // 递增
6  int main()
7  {
8      //后置递增
9      int a = 10;
10     a++; // 等价于a = a+1
11     cout << a << endl; //11
12
13     //前置递增
14     int b = 10;
15     ++b;
16     cout << b << endl; //11
17
18     //区别
19     //前置递增先对变量进行++, 再计算表达式
20     int a2 = 10;
21     int b2 = ++a2 * 10;
22     cout << b2 << endl;
23
24     // 后置递增先计算表达式, 后对变量进行++
25     int a3 = 10;
26     int b3 = a3++ * 10;
27     cout << b3 << endl;
28     system("pause");
29     return 0;
30 }

```

总结：前置递增先对变量进行++, 再计算表达式, 后置递增相反

3.2 赋值运算符

作用：用于将表达式的值赋给变量

赋值运算符包括以下几个符号：

运算符	术语	示例	结果
=	赋值	a=2; b=3;	a=2; b=3;
+=	加等于	a=0; a+=2;	a=2;
-=	减等于	a=5; a-=3;	a=2;
=	乘等于	a=2; a=2;	a=4;
/=	除等于	a=4; a/=2;	a=2;
%=	模等于	a=3; a%=2;	a=1;

示例：

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7      // 赋值运算符
8      // 1、 =
9      int a = 10;
10     a = 100;
11     cout << "a=" << a << endl;
12
13     // 2、 +=
14     a = 10;
15     a += 2;
16     cout << "a=" << a << endl;
17
18     // 3、 -=
19     a = 10;
20     a -= 2; // a = a-2
21     cout << "a=" << a << endl;
22
23     // 4、 *=
24     a = 10;
25     a *= 2; // a=a*2
26     cout << "a=" << a << endl;
27
28     // /=
29     a = 10;
30     a /= 2; // a = a/2
31     cout << "a=" << a << endl;
32
33     // %=
34     a = 10;
35     a %= 2; //a = a%2
36     cout << "a=" << a << endl;
37

```

```
38     system("pause");
39     return 0;
40 }
```

3.3 比较运算符

作用：用于表达式的比较，并返回一个真值或假值

比较运算符有以下符号：

运算符	术语	示例	结果
==	相等	4 == 3	0
!=	不等	4 != 3	1
<	小于	4 < 3	0
>	大于	4 > 3	1
<=	小于等于	4 <= 3	0
>=	大于等于	4 >= 1	1

示例：

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7
8      // 比较运算符
9      // ==
10     int a = 10;
11     int b = 20;
12     cout << (a == b) << endl; // 0
13
14     //!=
15     cout << (a != b) << endl; // 1
16
17     //>
18     cout << (a > b) << endl; // 0
19
20     // <
21     cout << (a < b) << endl; // 1
22
23     //>=
24     cout << (a >= b) << endl; // 0
25
26     //<=
27     cout << (a <= b) << endl; // 1
```

```

28
29     system("pause");
30     return 0;
31 }

```

注意：C和C++ 语言的比较运算中，“真”用数字“1”来表示，“假”用数字“0”来表示。

3.4 逻辑运算符

作用：用于根据表达式的值返回真值或假值

逻辑运算符有以下符号：

运算符	术语	示例	结果
!	非	!a	如果a为假，则!a为真；如果a为真，则!a为假。
&&	与	a && b	如果a和b都为真，则结果为真，否则为假。
	或	a b	如果a和b有一个为真，则结果为真，二者都为假时，结果为假。

示例1：逻辑非

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4  //逻辑运算符---非
5  int main()
6  {
7      int a = 10;
8      cout << !a << endl; // 0
9      cout << !!a << endl; // 1
10
11
12     system("pause");
13     return 0;
14 }

```

总结：真变假，假变真

示例2：逻辑与

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4  //逻辑运算符---与
5  int main()
6  {
7      int a = 10;
8      int b = 10;
9      cout << (a&&b) << endl; // 1
10

```

```

11     a = 0;
12     b = 10;
13     cout << (a&&b) << endl; // 0
14
15     a = 0;
16     b = 0;
17     cout << (a&&b) << endl; // 0
18     //同真为真，其余为假
19     system("pause");
20     return 0;
21 }

```

总结：逻辑与运算符总结：同真为真，其余为假

示例3：逻辑或

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4  //逻辑运算符---或
5  int main()
6  {
7      int a = 10;
8      int b = 10;
9      cout << (a || b) << endl; // 1
10
11     a = 0;
12     b = 10;
13     cout << (a || b) << endl; // 1
14
15     a = 0;
16     b = 0;
17     cout << (a || b) << endl; // 0
18
19     // 逻辑或：同假为假，其余为真
20     system("pause");
21     return 0;
22 }

```

逻辑或运算符总结：同假为假，其余为真

4、程序流程结构

C/C++支持最基本的三种程序运行结构：顺序结构、选择结构、循环结构

- 顺序结构：程序按顺序执行，不发生跳转
- 选择结构：依据条件是否满足，有选择的执行相应功能
- 循环结构：依据条件是否满足，循环多次执行某段代码

4.1 选择结构

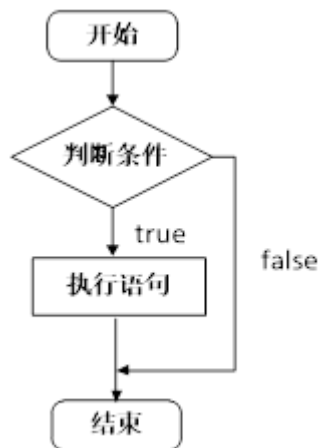
4.1.1 if语句

作用：执行满足条件的语句

if语句的三种形式

- 单行格式if语句
- 多行格式if语句
- 多条件的if语句

1. 单行格式if语句： `if(条件){ 条件满足执行的语句 }`

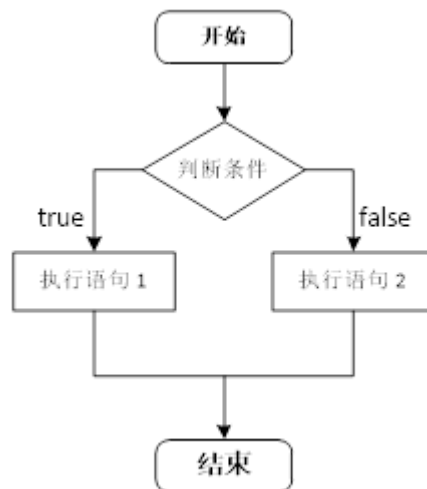


示例：

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7      // 选择结构-单行if语句
8      // 输入一个分数，如果分数大于600分，视为考上一本大学，否则你懂滴，哈哈
9      // 1、用户输入分数
10     int score = 0;
11     cout << "请输入一个分数：" << endl;
12     cin >> score;
13
14     // 2、打印用户输入的分数
15     cout << "您输入的分数是：" << score << endl;
16
17     // 3、判断分数是否大于600，如果大于，那么输出
18     if (score>600)
19     {
20         cout << "Jjk恭喜你，考上了一本大学哇"<<endl;
21     }
22
23     system("pause");
24     return 0;
25 }
26 }
```

注意：if条件表达式后不要加分号

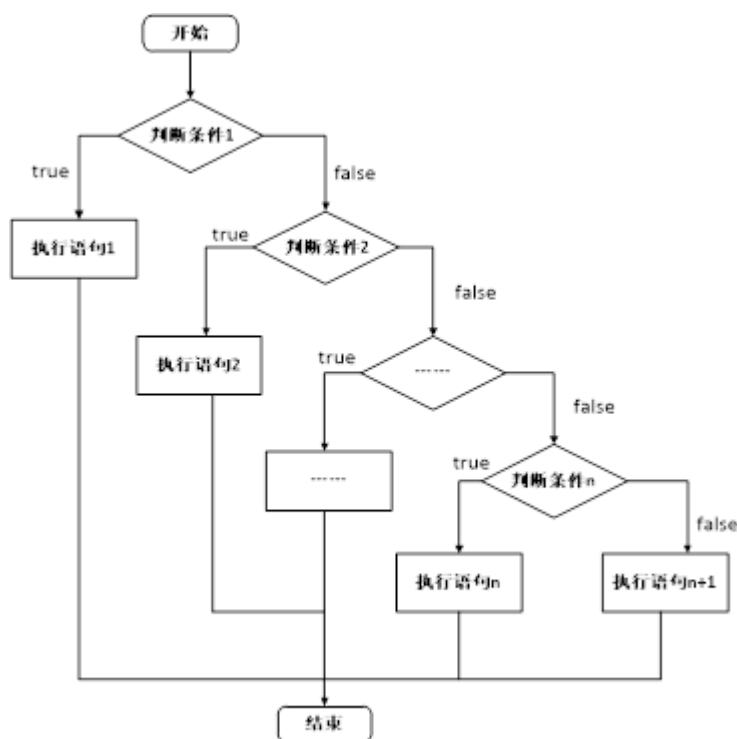
2. 多行格式if语句：if(条件){ 条件满足执行的语句 }else{ 条件不满足执行的语句 };



示例：

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7      // 选择结构-多行if语句
8      // 输入一个分数，如果分数大于600分，视为考上一本大学，否则，打印没考上。
9
10     // 1、用户输入分数
11     int score = 0;
12     cout << "请输入一个分数：" << endl;
13     cin >> score;
14
15     // 2、打印用户输入的分数
16     cout << "您输入的分数是：" << score << endl;
17
18     // 3、判断分数是否大于600，如果大于，那么输出
19     if (score>600)
20     {
21         cout << "jkk恭喜你，考上了一本大学哇"<<endl;
22     }
23     else
24     {
25         cout << "很遗憾，您和作者一样的菜，哈哈哈" << endl;
26     }
27
28 }
29
30 system("pause");
31 return 0;
32 }
```


3. 多条件的if语句: if(条件1){ 条件1满足执行的语句 }else if(条件2){条件2满足执行的语句}... else{ 都不满足执行的语句}



示例:

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7      // 选择结构-多条件if语句
8      // 输入一个分数, 如果分数大于600分, 视为考上一本大学, 在屏幕输出
9      // 大于500分, 视为考上二本大学, 屏幕输出
10     // 大于400分, 视为考上三本大学, 屏幕输出
11     // 小于等于400分, 未考上本科, 屏幕输出
12
13     // 1、用户输入分数
14     int score = 0;
15     cout << "请输入一个分数: " << endl;
16     cin >> score;
17
18     // 2、打印用户输入的分数
19     cout << "您输入的分数是: " << score << endl;
20
21     // 3、判断
22     // 如果大于600, 考上一本
23     // 如果大于500, 考上二本
24     // 如果大于400, 考上三本
25     // 前三个都不满足, 凉凉啦!!!
26     if (score>600)
27     {
28         cout << "jkl恭喜你, 考上了一本大学哇"<<endl;
```

```

29
30     }
31     else if (score > 500)
32     {
33         cout << "jkk恭喜你，考上了二本大学哇" << endl;
34     }
35     else if (score>400)
36     {
37         cout << "jkk恭喜你，考上了三本大学哇" << endl;
38     }
39     else
40     {
41         cout << "你和作者一样low逼，哈哈哈" << endl;
42     }
43
44     system("pause");
45     return 0;
46 }

```

嵌套if语句：在if语句中，可以嵌套使用if语句，达到更精确的条件判断

案例需求：

- 提示用户输入一个高考考试分数，根据分数做如下判断
- 分数如果大于600分视为考上一本，大于500分考上二本，大于400考上三本，其余视为未考上本科；
- 在一本分数中，如果大于700分，考入北大，大于650分，考入清华，大于600考入人大。

示例：

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4  // 嵌套if语句
5  int main()
6  {
7      /*
8
9      - 提示用户输入一个高考考试分数，根据分数做如下判断
10     - 分数如果大于600分视为考上一本，大于500分考上二本，大于400考上三本，其余视为未考上本科；
11     - 在一本分数中，如果大于700分，考入北大，大于650分，考入清华，大于600考入人大。
12     */
13
14     // 1、用户输入分数
15     int score = 0;
16     cout << "请输入一个分数：" << endl;
17     cin >> score;
18
19     // 2、打印用户输入的分数
20     cout << "您输入的分数是：" << score << endl;
21
22     // 3、判断
23     // 如果大于600，考上一本
24     // 大于700 北大

```

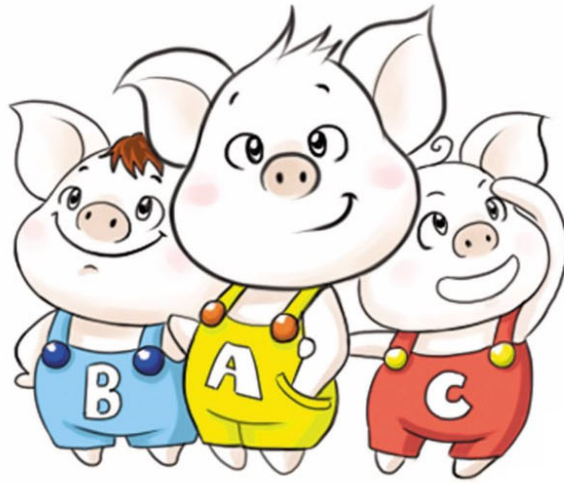
```

25     // 大于650 清华
26     // 其余    人大
27     // 如果大于500, 考上二本
28     // 如果大于400, 考上三本
29     // 前三个都不满足, 凉凉啦!!!
30     if (score>600)
31     {
32         cout << "jkk恭喜你, 考上了一本大学哇"<<endl;
33         if (score>700)
34         {
35             cout << "北大" << endl;
36         }
37         else if (score>650)
38         {
39             cout << "清华" << endl;
40         }
41         else
42         {
43             cout << "人大" << endl;
44         }
45     }
46     else if (score > 500)
47     {
48         cout << "jkk恭喜你, 考上了二本大学哇" << endl;
49     }
50     else if (score>400)
51     {
52         cout << "jkk恭喜你, 考上了三本大学哇" << endl;
53     }
54     else
55     {
56         cout << "你和作者一样low逼, 哈哈" << endl;
57     }
58 }
59
60 system("pause");
61 return 0;
62 }

```

练习案例：三只小猪称体重

有三只小猪ABC，请分别输入三只小猪的体重，并且判断哪只小猪最重？



```
1 1、先判断A和B谁重
2   A重    让A和C比较
3           A重：结果是A最重
4           C重：结果是C最重
5   C重    让B和C比较
6           B重：结果是B最重
7           C重：结果是C最重
```

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  // 嵌套if语句
5  int main()
6  {
7
8      /*
9      1、先判断A和B谁重
10         A重    让A和C比较
11                A重：结果是A最重
12                C重：结果是C最重
13         C重    让B和C比较
14                B重：结果是B最重
15                C重：结果是C最重
16
17         */
18
19         // 需求：三只小猪称体重，判断哪只最重
20         // 创建三只小猪的体重变量
21         int num1 = 0;
22         int num2 = 0;
23         int num3 = 0;
```

```

24
25 // 让用户输入三只小猪的重量
26 cout << "请输入小猪A的体重: " << endl;
27 cin >> num1;
28
29 cout << "请输入小猪B的体重: " << endl;
30 cin >> num2;
31
32 cout << "请输入小猪C的体重: " << endl;
33 cin >> num3;
34
35 cout << "小猪A的体重为: " << num1 << endl;
36 cout << "小猪B的体重为: " << num2 << endl;
37 cout << "小猪C的体重为: " << num3 << endl;
38
39 // 判断哪只最重
40 if (num1>num2) // A>B
41 {
42     if (num1>num3)
43     {
44         cout << "小猪A最重" << endl;
45     }
46     else
47     {
48         cout << "小猪C最重" << endl;
49     }
50 }
51 else //B>A
52 {
53     if (num2>num3)
54     {
55         cout << "小猪B最重" << endl;
56     }
57     else
58     {
59         cout << "小猪C最重" << endl;
60     }
61 }
62 }
63 system("pause");
64 return 0;
65 }

```

4.1.2 三目运算符

作用： 通过三目运算符实现简单的判断

语法： 表达式1 ? 表达式2 : 表达式3

解释：

如果表达式1的值为真，执行表达式2，并返回表达式2的结果；

如果表达式1的值为假，执行表达式3，并返回表达式3的结果。

示例：

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  int main()
5  {
6      // 三目运算符
7
8      // 创建三个变量a,b,c
9      // 将a和b做比较，将变量大的值赋值给变量c
10     int a = 10;
11     int b = 20;
12     int c = 0;
13
14     c = (a > b ? a : b); //返回值赋值给c
15     cout << "变量c=" << c << endl;
16
17     //在c++中三目运算符返回的是变量，可以继续赋值
18     (a > b ? a : b) = 100;
19     cout << "变量a=" << a << endl; // 10
20     cout << "变量b=" << b << endl; // 100
21
22     system("pause");
23     return 0;
24 }
```

总结：和if语句比较，三目运算符优点是短小整洁，缺点是如果用嵌套，结构不清晰

4.1.3 switch语句

作用：执行多条件分支语句

语法：

```
1  switch(表达式)
2
3  {
4
5      case 结果1: 执行语句;break;
6
7      case 结果2: 执行语句;break;
8
9      ...
10
11     default:执行语句;break;
12
13 }
```

示例：

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4  // switch语句
5  int main()
6  {
7      //给电影打分
8      // 10~9 经典
9      // 8~7 非常好
10     // 6~5 一般
11     // 5以下 烂片
12
13     // 1、提示用户给电影评分
14     cout << "请给电影进行打分: " << endl;
15
16     // 2、用户开始进行打分
17     int score = 0;
18     cin >> score;
19     cout << "您打的分数是:" << score << endl;
20     // 3、根据用户输入的分数来提示用户最后的结果
21     switch (score)
22     {
23     case 10:
24         cout << "您认为是经典电影" << endl;
25         break;
26     case 9:
27         cout << "您认为是经典电影" << endl;
28         break; // 退出当前分支
29     case 8:
30         cout << "您认为这个电影非常好" << endl;
31         break;
32     case 7:
33         cout << "您认为这个电影非常好" << endl;
34         break;
35     case 6:
36         cout << "您认为这个电影一般" << endl;
37         break;
38     case 5:
39         cout << "您认为这个电影一般" << endl;
40     default:
41         cout << "您认为这是一个烂片" << endl;
42
43     }
44
45     system("pause");
46     return 0;
47 }
48

```

注意1: switch语句中表达式类型只能是整型或者字符型

注意2: case里如果没有break, 那么程序会一直向下执行

总结：与if语句比，对于多条件判断时，switch的结构清晰，执行效率高，缺点是switch不可以判断区间

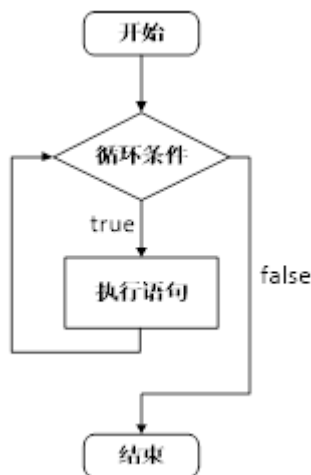
4.2 循环结构

4.2.1 while循环语句

作用：满足循环条件，执行循环语句

语法： `while(循环条件){ 循环语句 }`

解释：只要循环条件的结果为真，就执行循环语句



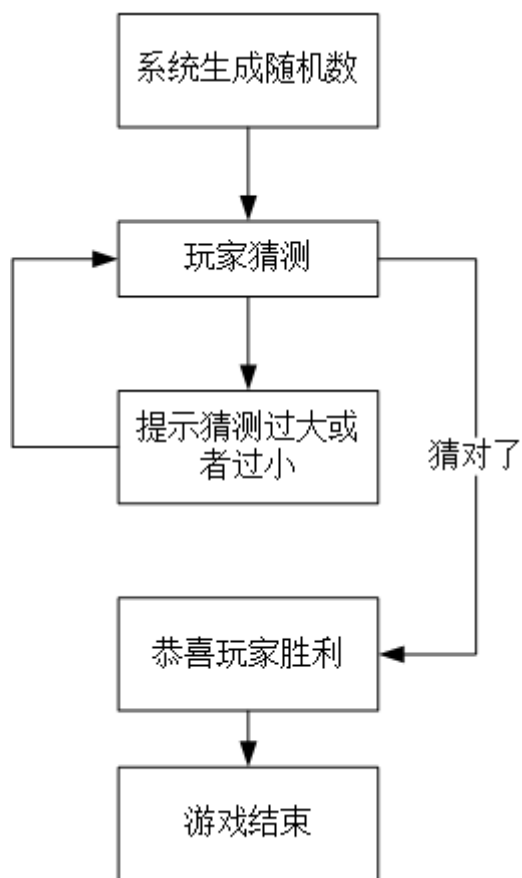
示例：

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  // switch语句
5  int main()
6  {
7      // while循环
8      // 屏幕上打印0~10这10个数字
9      int num = 0;
10
11     // while()中循环的条件
12     // 注意事项：在写循环一定要避免出现死循环现象
13     while (num<10)
14     {
15         cout << num << endl;
16         num++;
17     }
18     system("pause");
19     return 0;
20 }
```

注意：在执行循环语句时候，程序必须提供跳出循环的出口，否则出现死循环

while循环练习案例：猜数字

案例描述：系统随机生成一个1到100之间的数字，玩家进行猜测，如果猜错，提示玩家数字过大或过小，如果猜对恭喜玩家胜利，并且退出游戏。



```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7
8      // 添加随机数种子，作用利用当前系统时间随机随机数，防止每次随机数都一样
9      srand((unsigned int)time(NULL));
```

```

10 //1、系统生成随机数
11 int num = rand() % 100 + 1; //rand()%100+1生成0+1~99+1的随机数
12 //cout << num << endl;
13
14 //2、玩家进行猜测
15 int val = 0; // 玩家输入的数据
16 while (1)
17 {
18     cin >> val;
19
20     //3、判断玩家的猜测
21     // 猜错：提示猜的结果，过大或者过小，重新返回第二步
22     if (val > num)
23     {
24         cout << "猜测过大" << endl;
25     }
26     else if (val < num)
27     {
28         cout << "猜测过小" << endl;
29     }
30     else
31     {
32         cout << "恭喜您，猜对了" << endl;
33         break; // 猜对：退出游戏
34     }
35 }
36
37 system("pause");
38 return 0;
39 }

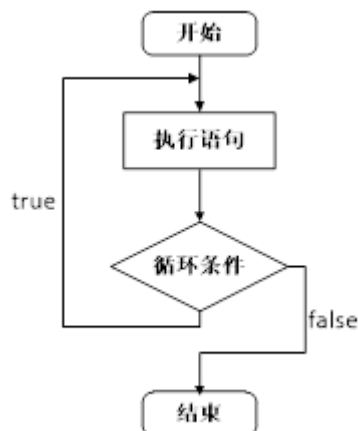
```

4.2.2 do...while循环语句

作用： 满足循环条件，执行循环语句

语法： `do{ 循环语句 } while(循环条件);`

注意： 与while的区别在于do...while会先执行一次循环语句，再判断循环条件



示例：

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7      //do...while语句
8      //在屏幕中输出0-9这10个数字
9      int num = 0;
10     do
11     {
12         cout << num << endl;
13         num++;
14     } while (num<10);
15     // do...while和while循环区别在于do...while会先执行一次循环语句
16
17     system("pause");
18     return 0;
19 }

```

总结：与while循环区别在于，do...while先执行一次循环语句，再判断循环条件

练习案例：水仙花数

案例描述：水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身

例如： $1^3 + 5^3 + 3^3 = 153$

请利用do...while语句，求出所有3位数中的水仙花数

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7
8      //1、将所有的三位数进行输出(100-999)
9      //2、在所有的三位数中找到水仙花数
10     /*
11         水仙花数
12         获取个位：对数字取模于10可以获取个位
13         获取十位：对数字先整除于10，然后再取模于10，得到十位数字
14         获取百位：对数字整除于100，获取百位
15
16         判断：个位^3+十位^3+百位^3 = 本身
17
18     */
19
20     //1、将所有的三位数进行输出(100-999)
21     int num = 100;

```

```

22     do
23     {
24         //cout << num << endl;
25         //2、在所有的三位数中找到水仙花数
26         int a = 0; // 个位
27         int b = 0; // 十位
28         int c = 0; // 百位
29         a = num % 10; // 获取数字的个位
30         b = num / 10 % 10; // 获取数字的十位
31         c = num / 100; // 获取数字的百位
32
33         if (a*a*a+b*b*b+c*c*c==num) //如果是水仙花数，才打印
34         {
35             cout << num << endl;
36         }
37         num++;
38
39     } while (num<1000);
40     system("pause");
41     return 0;
42 }

```

4.2.3 for循环语句

作用： 满足循环条件，执行循环语句

语法： for(起始表达式;条件表达式;末尾循环体) { 循环语句; }

示例：

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  //for循环
6  int main()
7  {
8      //从数字0 打印到 数字9
9      for (int i = 0; i < 10; i++)
10     {
11         cout << i << endl;
12     }
13     system("pause");
14     return 0;
15 }

```

详解：

```

int main() {
    执行一次 → 0      1      3
    for (int i = 0; i < 10; i++)
    {
        2 cout << i << endl;
    }

    执行顺序: 0123123123...

    system("pause");

    return 0;
}

```

注意: for循环中的表达式, 要用分号进行分隔

总结: while, do...while, for都是开发中常用的循环语句, for循环结构比较清晰, 比较常用

练习案例: 敲桌子

案例描述: 从1开始数到数字100, 如果数字个位含有7, 或者数字十位含有7, 或者该数字是7的倍数, 我们打印敲桌子, 其余数字直接打印输出。



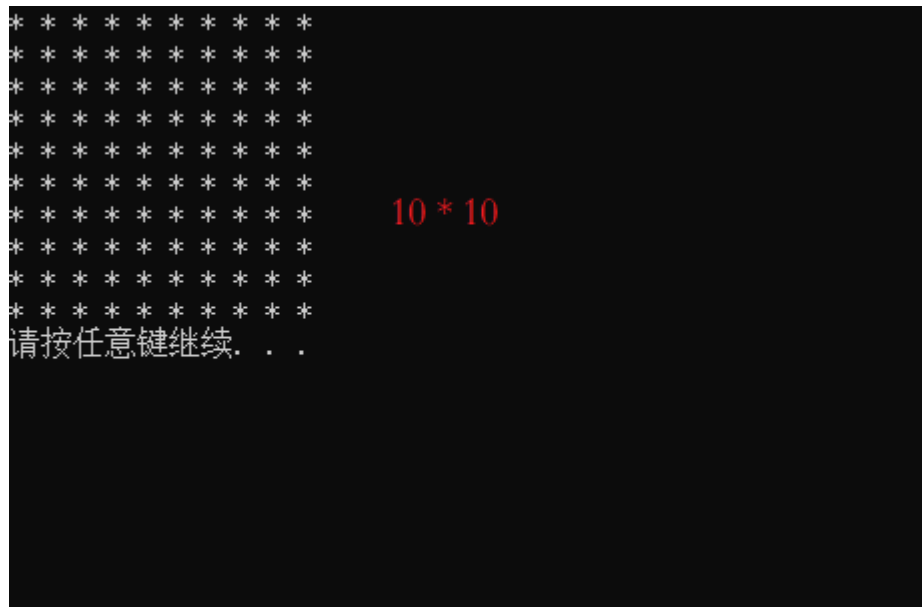
```
1 1、先来输出1-100这些数字
2 2、从这100个数字中找到特殊的数字，改为敲桌子
3 特殊数字：
4     7的倍数：取模为0
5     个位有7：取模于10=7
6     十位有7：取整数于10=7
```

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  //for循环
6  int main()
7  {
8      //从数字0 打印到 数字100
9      // 1、输出1-100数字
10     for (int i = 0; i < 100; i++)
11     {
12         //2、如果是特殊数字
13         if (i%7==0 || i%10==7 || i/10==7)
14         {
15             cout << "敲桌子" << endl;
16         }
17         else
18         {
19             cout << i << endl;
20         }
21     }
22     system("pause");
23     return 0;
24 }
25 }
```

4.2.4 嵌套循环

作用： 在循环体中再嵌套一层循环，解决一些实际问题

例如我们想在屏幕中打印如下图片，就需要利用嵌套循环



示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7      //利用嵌套循环实现星图
8      //外层执行一次，内层执行一周
9      for (int i = 0; i < 10; i++)
10     {
11         for (int j = 0; j < 10; j++)
12         {
13             cout << "* ";
14         }
15         cout << endl;
16     }
17     system("pause");
18     return 0;
19 }
```

练习案例：乘法口诀表

案例描述：利用嵌套循环，实现九九乘法表

乘法口诀表

1x1=1									
1x2=2	2x2=4								
1x3=3	2x3=6	3x3=9							
1x4=4	2x4=8	3x4=12	4x4=16						
1x5=5	2x5=10	3x5=15	4x5=20	5x5=25					
1x6=6	2x6=12	3x6=18	4x6=24	5x6=30	6x6=36				
1x7=7	2x7=14	3x7=21	4x7=28	5x7=35	6x7=42	7x7=49			
1x8=8	2x8=16	3x8=24	4x8=32	5x8=40	6x8=48	7x8=56	8x8=64		
1x9=9	2x9=18	3x9=27	4x9=36	5x9=45	6x9=54	7x9=63	8x9=72	9x9=81	

1、列数 * 行数 = 计算结果

2、列数<=当前行数

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7      //乘法口诀表
8      // 打印行数
9      for (int i = 1; i <=9; i++)
10     {
11         //cout << i << endl; //
12         for (int j = 1; j <=i; j++)
13         {
14             cout << j<<"*"<<i<<"="<<j*i<<"\t"; //列数
15         }
16         cout << endl;
17     }
18     system("pause");
19     return 0;
20 }
```


4.3 跳转语句

4.3.1 break语句

作用: 用于跳出选择结构或者循环结构

break使用的时机:

- 出现在switch条件语句中，作用是终止case并跳出switch
- 出现在循环语句中，作用是跳出当前的循环语句
- 出现在嵌套循环中，跳出最近的内层循环语句

示例1:

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7      //break使用时机
8      //1、出现在switch
9      cout << "请选择副本的难度" << endl;
10     cout << "1、普通" << endl;
11     cout << "2、中等" << endl;
12     cout << "3、困难" << endl;
13
14     int select = 0; // 创建选择结果的变量
15     cin >> select; // 等待用户输入
16     switch (select)
17     {
18     case 1:
19         cout << "你选择的是普通难度" << endl;
20         break;
21     case 2:
22         cout << "你选择的是中等难度" << endl;
23         break;
24     case 3:
25         cout << "你选择的是困难难度" << endl;
26         break;
27
28     default:
29         break;
30     }
31
32
33     //2、出现在循环语句中
34     //3、出现在嵌套循环语句中
35     system("pause");
36     return 0;
37 }
```

示例2:

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7      //break使用时机
8      //2、出现在循环语句中
9      for (int i = 0; i < 10; i++)
10     {
11         if (i==5)
12         {
13             break; // 跳出循环语句
14         }
15         cout << i << endl;
16     }
17
18     //3、出现在嵌套循环语句中
19     system("pause");
20     return 0;
21 }
```

示例3:

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //break使用时机
7      //3、出现在嵌套循环语句中
8      // 在嵌套循环语句中使用break, 退出内层循环
9      for (int i = 0; i < 10; i++)
10     {
11         for (int j = 0; j < 10; j++)
12         {
13             if (j == 5)
14             {
15                 break;
16             }
17             cout << "*" << " ";
18         }
19         cout << endl;
20     }
21
22     system("pause");
23     return 0;
24 }
```

4.3.2 continue语句

作用：在循环语句中，跳过本次循环中余下尚未执行的语句，继续执行下一次循环

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7
8      //continue语句
9
10     for (int i = 0; i <=100; i++)
11     {
12         //如果是奇数，输出
13         if (i%2==0)
14         {
15             continue; //可以筛选条件，执行到此就不在向下执行， 执行下一次循环
16             // break会退出循环，而continue不会
17         }
18         cout << i << endl;
19     }
20
21     system("pause");
22     return 0;
23 }
```

注意：continue并没有使整个循环终止，而break会跳出循环

4.3.3 goto语句

作用：可以无条件跳转语句

语法： goto 标记;

解释：如果标记的名称存在，执行到goto语句时，会跳转到标记的位置

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7      //goto语句——推荐不适用：影响代码的逻辑结构
8
9      cout << "1、xxxx" << endl;
10     cout << "2、xxxx" << endl;
11 }
```

```
12     goto FLAG;
13     cout << "3、xxxx" << endl;
14     cout << "4、xxxx" << endl;
15     FLAG:
16     cout << "5、xxxx" << endl;
17     system("pause");
18     return 0;
19 }
```

注意：在程序中不建议使用goto语句，以免造成程序流程混乱

5、数组

5.1 概述

所谓数组，就是一个集合，里面存放了相同类型的数据元素

特点1：数组中的每个数据元素都是相同的数据类型

特点2：数组是由连续的内存位置组成的



5.2 一维数组

5.2.1 一维数组定义方式

一维数组定义的三种方式：

1. 数据类型 数组名 [数组长度] ;
2. 数据类型 数组名 [数组长度] = { 值1, 值2 ... } ;
3. 数据类型 数组名 [] = { 值1, 值2 ... } ;

示例

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
7
8      //数组
9      /*
10
11      1. 数据类型  数组名[ 数组长度 ];
12      2. 数据类型  数组名[ 数组长度 ] = { 值1, 值2 ...};
13      3. 数据类型  数组名[ ] = { 值1, 值2 ...};
14      */
15      //1. 数据类型  数组名[数组长度];
16      int arr[5];
17      //给数组中的元素进行赋值
18      //数组元素的下标是从0开始索引的
19      arr[0] = 10;
20      arr[1] = 20;
21      arr[2] = 30;
22      arr[3] = 40;
23      arr[4] = 50;
24      //访问数组元素
25      cout << arr[0] << endl;
26      cout << arr[1] << endl;
27      cout << arr[2] << endl;
28      cout << arr[3] << endl;
29      cout << arr[4] << endl;
30
31      // 2. 数据类型  数组名[ 数组长度 ] = { 值1, 值2 ...};
32      //如果在初始化数据的时候，没有全部填写完，会用0进行填充剩余的数据
33      int arr2[5] = { 10,20,30,40,50 };
34      //cout << arr2[0] << endl;
35      //cout << arr2[1] << endl;
36      //cout << arr2[2] << endl;
37      //cout << arr2[3] << endl;
38      //cout << arr2[4] << endl;
39      for (int i = 0; i < 5; i++)
40      {
41          cout << arr2[i] << endl;
42      }
43      //3. 数据类型  数组名[] = { 值1, 值2 ... };
44      //定义数组的时候，必须有初始长度
45      int arr3[] = { 10,20,30,40,50 };
46      for (int j = 0; j < 5; j++)
47      {
48          cout << arr2[j] << endl;
49      }
50
51      system("pause");
```

```
52     return 0;
53 }
```

总结1：数组名的命名规范与变量名命名规范一致，不要和变量重名

总结2：数组中下标是从0开始索引

5.2.2 一维数组数组名

一维数组名称的用途：

1. 可以统计整个数组在内存中的长度
2. 可以获取数组在内存中的首地址

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //数组名用途
7      //1、可以统计整个数组在内存的长度
8      int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };
9      cout << "整个数组占用内存空间为: " << sizeof(arr) << endl;
10     cout << "每个元素占用内存空间: " << sizeof(arr[0]) << endl;
11     cout << "数组中元素的个数: " << sizeof(arr) / sizeof(arr[0]) << endl;
12
13     //2、可以获取数组在内存中的首地址
14     cout << "数组首地址: " << arr << endl;
15     cout << "数组中第一个元素地址为: " << &arr[0] << endl;
16
17     //数组名是常量，不可以进行赋值操作
18     system("pause");
19     return 0;
20 }
```

注意：数组名是常量，不可以赋值

总结1：直接打印数组名，可以查看数组所占内存的首地址

总结2：对数组名进行sizeof，可以获取整个数组占内存空间的大小

练习案例1：五只小猪称体重

案例描述：

在一个数组中记录了五只小猪的体重，如：int arr[5] = {300,350,200,400,250};

找出并打印最重的小猪体重。

思想：访问数组中每个元素，如果这个元素比我认定的最大值要打，更新最大值。

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      // 1、创建5只小猪体重的数组
7      int arr[5] = {300,350,200,400,250};
8
9      // 2、从数组中找出最大值
10     //先假定一个最大值, arr[0]
11     int max = 0;
12     for (int i = 0; i < 5; i++)
13     {
14         //cout << arr[i] << endl;
15         //如果访问的数组中元素比我认定的最大值还要大, 更新最大值
16         if (arr[i]>max)
17         {
18             max = arr[i];
19         }
20
21     }
22     // 3、打印最大值
23     cout << "最重的小猪体重为: " << max << endl;
24     system("pause");
25     return 0;
26 }

```

练习案例2：数组元素逆置

案例描述：请声明一个5个元素的数组，并且将元素逆置.

(如原数组元素为：1,3,2,5,4;逆置后输出结果为:4,5,2,3,1);

int start = 0;

int end = sizeof(arr)/sizeof(arr[0]) -1; //末尾元素下标

start和end下标元素进行互换，还需要一个零时的变量

```
int start = 0; //起始元素下标
int end = sizeof(arr)/sizeof(arr[0]) - 1; //末尾元素下标
```



```
int temp = arr[start];
arr[start] = arr[end]
arr[end] = temp;

start++; end--;
```

如果start < end
执行互换

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //实现数组元素逆置
7      // 1、创建数组
8      int arr[5] = { 1,2,3,4,5 };
9      cout << "元素数组逆置前结果: " << endl;
10     for (int i = 0; i < 5; i++)
11     {
12         cout << arr[i] << endl;
13     }
14
15     // 2、实现逆置
16     // 2.1 记录起始下标位置
17     // 2.2 记录结束下标位置
18     // 2.3 记录起始下标与结束下标的元素互换
19     // 2.4 起始位置++, 结束位置--
20     // 2.5 循环执行2.1操作, 知道起始位置>=结束位置
21     int start = 0; //起始下标
22     int end = sizeof(arr) / sizeof(arr[0]) - 1; //结束下标
23
24     while (start<end)
25     {
26         //实现元素互换
27         int temp = arr[start];
28         arr[start] = arr[end];
29         arr[end] = temp;
30
31         //下标更新
```



```

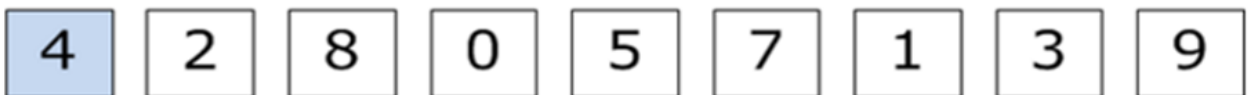
32     start++;
33     end--;
34
35 }
36
37 // 3、打印逆置后的数组
38 cout << "数组元素逆置后结果：" << endl;
39 for (int i = 0; i < 5; i++)
40 {
41     cout << arr[i] << endl;
42 }
43
44 system("pause");
45 return 0;
46 }

```

5.2.3 冒泡排序

作用：最常用的排序算法，对数组内元素进行排序

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素做同样的工作，执行完毕后，找到第一个最大值。
3. 重复以上的步骤，每次比较次数-1，直到不需要比较



示例：将数组 { 4,2,8,0,5,7,1,3,9 } 进行升序排序



```

1 #include<iostream>
2 #include<ctime> //time系统时间头文件
3 using namespace std;
4 int main()
5 {

```

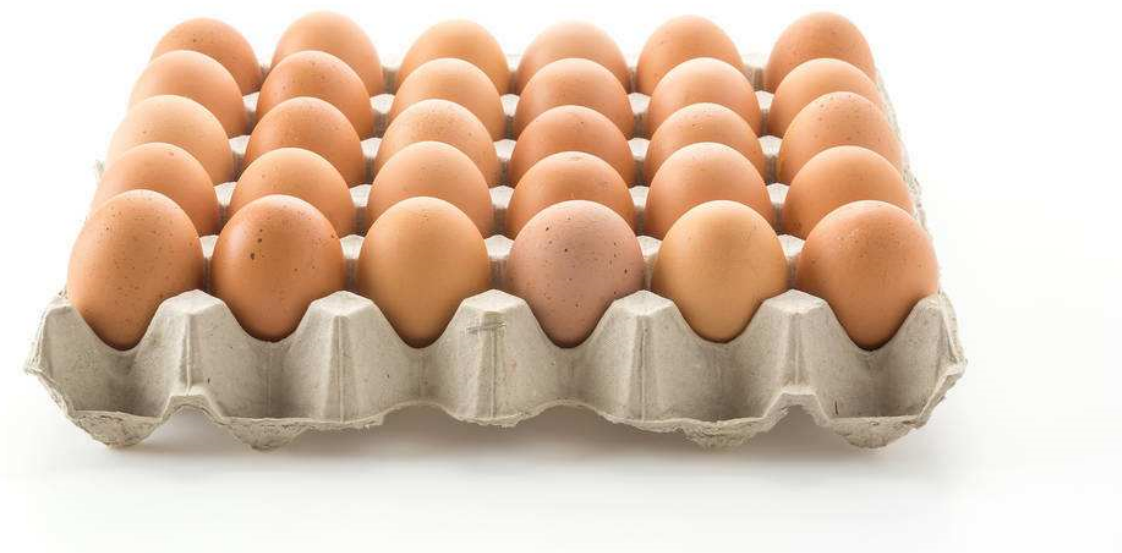
```

6 //利用冒泡排序实现升序序列
7 int arr[9] = { 4,2,8,0,5,7,1,3,9 };
8 cout << "排序前: " << endl;
9 for (int i = 0; i < 9; i++)
10 {
11     cout << arr[i] << "\t";
12 }
13 //开始排序
14 // 总共排序轮数为: 元素个数-1
15 for (int i = 0; i < 9-1; i++)
16 {
17     //内层循环对比次数 = 元素个数-当前轮数-1
18     for (int j = 0; j < 9-i-1; j++) //sizeof(arr)/size(arr[0])
19     {
20         //如果第一个数字, 比第二个数字大, 交换两个数字
21         if (arr[j] > arr[j + 1])
22         {
23             int temp = arr[j];
24             arr[j] = arr[j + 1];
25             arr[j + 1] = temp;
26         }
27     }
28 }
29 //排序后结果
30 cout << "排序后结果: " << endl;
31 for (int i = 0; i < 9; i++)
32 {
33     cout << arr[i] << "\t";
34 }
35
36 cout << endl;
37 system("pause");
38 return 0;
39 }

```

5.3 二维数组

二维数组就是在一维数组上, 多加一个维度。



5.3.1 二维数组定义方式

二维数组定义的四种方式：

1. 数据类型 数组名[行数][列数];
2. 数据类型 数组名[行数][列数] = { {数据1, 数据2 } , {数据3, 数据4 } };
3. 数据类型 数组名[行数][列数] = { 数据1, 数据2, 数据3, 数据4};
4. 数据类型 数组名[][列数] = { 数据1, 数据2, 数据3, 数据4};

建议：以上4种定义方式，利用**第二种更加直观，提高代码的可读性**

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //二维数组定义方式
7
8      /*
9      1. 数据类型 数组名[ 行数 ][ 列数 ];
10     2. 数据类型 数组名[ 行数 ][ 列数 ] = { {数据1, 数据2 } , {数据3, 数据4 } };
11     3. 数据类型 数组名[ 行数 ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
12     4. 数据类型 数组名[ ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
13     */
14     //1. 数据类型 数组名[ 行数 ][ 列数 ];
15     int arr[2][3]; // 2行3列数组
16     arr[0][0] = 1;
17     arr[0][1] = 2;
18     arr[0][2] = 3;
```

```

19 arr[1][0] = 4;
20 arr[1][1] = 5;
21 arr[1][2] = 6;
22 cout << "输出每一个元素: " << endl;
23 //外层循环打印行数, 内层循环打印列数
24 for (int i = 0; i < 2; i++)
25 {
26     for (int j = 0; j < 3; j++)
27     {
28         cout << arr[i][j]<<endl;
29     }
30 }
31
32
33 //2. 数据类型 数组名[行数][列数] = { {数据1, 数据2 } , {数据3, 数据4 } };
34 int arr2[2][3] =
35 {
36     {1,2,3},
37     {4,5,6},
38 };
39 for (int i = 0; i < 2; i++)
40 {
41     for (int j = 0; j < 3; j++)
42     {
43         cout << arr2[i][j]<<" ";
44     }
45     cout << endl;
46 }
47
48
49 //3. 数据类型 数组名[行数][列数] = { 数据1, 数据2, 数据3, 数据4 };
50 int arr3[2][3] = { 1,2,3,4,5,6 };
51 for (int i = 0; i < 2; i++)
52 {
53     for (int j = 0; j < 3; j++)
54     {
55         cout << arr3[i][j] << " ";
56     }
57     cout << endl;
58 }
59
60
61 //4. 数据类型 数组名[][列数] = { 数据1, 数据2, 数据3, 数据4 };
62 int arr4[][3] = {1,2,3,4,5,6};
63 for (int i = 0; i < 2; i++)
64 {
65     for (int j = 0; j < 3; j++)
66     {
67         cout << arr4[i][j] << " ";
68     }
69     cout << endl;
70 }
71

```

```

72     system("pause");
73     return 0;
74 }

```

总结：在定义二维数组时，如果初始化了数据，可以省略行数

5.3.2 二维数组数组名

- 查看二维数组所占内存空间
- 获取二维数组首地址

示例：

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //二维数组名称用途
7      //1、可以查看占用的内存空间大小
8      int arr[2][3] =
9      {
10         {1,2,3},
11         {4,5,6}
12     };
13     cout << "二维数组占用的内存空间大小: " << sizeof(arr) << endl;
14     cout << "二维数组第一行占用内存为: " << sizeof(arr[0]) << endl; //0表示行号
15     cout << "二维数组第一个元素占用内存为: " << sizeof(arr[0][0]) << endl;
16
17     cout << "二维数组的行数为: " << sizeof(arr) / sizeof(arr[0]) << endl; // 行数
18     cout << "二维数组的列数为: " << sizeof(arr[0]) / sizeof(arr[0][0]) << endl; //列数
19
20     //2、可以查看二维数组的首地址
21     cout << "二维数组的首地址为: " << arr << endl;
22     cout << "二维数组第一行首地址为: " << arr[0] << endl;
23     cout << "二维数组第二行首地址为: " << arr[1] << endl;
24     cout << "二维数组第一个元素首地址: " << &arr[0][0] << endl; // 具体元素的地址，需要加一个取地
符
25     system("pause");
26     return 0;
27 }

```

总结1：二维数组名就是这个数组的首地址

总结2：对二维数组名进行sizeof时，可以获取整个二维数组占用的内存空间大小

5.3.3 二维数组应用案例

考试成绩统计：

案例描述：有三名同学（张三，李四，王五），在一次考试中的成绩分别如下表，请分别输出三名同学的总成绩

	语文	数学	英语
张三	100	100	100
李四	90	50	100
王五	60	70	80

- 1、创建一个二维3行3列数组
- 2、统计考试成绩，让每行的3列数据相加，统计出来一个综合

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  #include<string>
4  using namespace std;
5  int main()
6  {
7      //二维数组的案例——考试成绩统计
8      int score[3][3] =
9      {
10         {100,100,100},
11         {90,50,100},
12         {60,70,80}
13     };
14
15     string names[3] = { "张三","李四","王五" };
16     //2、统计每个人的总和分数
17     for (int i = 0; i < 3; i++)
18     {
19         int sum = 0; // 统计分数总和的变量
20         for (int j = 0; j < 3; j++)
21         {
22             sum += score[i][j];
23             //cout << score[i][j] << "\t";
24         }
25         cout << names[i] << "个人的总分为: " << sum << endl;
26         cout << endl;
27     }
28
29     }
30     system("pause");
31     return 0;
32 }
```

6、函数

6.1 概述

作用：将一段经常使用的代码封装起来，减少重复代码

一个较大的程序，一般分为若干个程序块，每个模块实现特定的功能。

6.2 函数的定义

函数的定义一般主要有5个步骤：

- 1、返回值类型
- 2、函数名
- 3、参数表列
- 4、函数体语句
- 5、return 表达式

语法：

```
1 返回值类型 函数名 （参数列表）
2  {
3
4      函数体语句
5
6      return表达式
7
8  }
```

- 返回值类型：一个函数可以返回一个值。在函数定义中
- 函数名：给函数起个名称
- 参数列表：使用该函数时，传入的数据
- 函数体语句：花括号内的代码，函数内需要执行的语句
- return表达式：和返回值类型挂钩，函数执行完后，返回相应的数据

示例：定义一个加法函数，实现两个数相加

- 1、返回值类型 int
- 2、函数的名称 add
- 3、参数列表 int num1,intnum2
- 4、函数体语句 int sum = num1+num2
- 5、return 表达式 return sum

```
1  //函数定义
2  int add(int num1, int num2)
3  {
4      int sum = num1 + num2;
5      return sum;
6  }
```

6.3 函数的调用

功能：使用定义好的函数

语法：函数名 (参数)

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  #include<string>
4  using namespace std;
5  //函数的定义
6  // 语法
7  // 返回值类型, 函数名 参数列表 具体的函数体语句 return表达式
8  // 加法函数, 实现两个整数相加, 并且将相加的结果进行返回
9
10 // 函数定义的时候, num1和num2并没有真的数据, 他只是一个形式上的参数, 简称形参
11 int add(int num1, int num2)
12 {
13     int sum = num1 + num2;
14     return sum;
15 }
16
17 int main()
18 {
19     int num1 = 1;
20     int num2 = 2;
21     // 调用函数
22     // 函数调用语法: 函数名称(参数)
23     // num1和num2成为实际参数, 简称实参; 在函数中并称之为形参
24     // 当调用函数的时候, 实参的值会传递给形参
25     int sum = add(num1, num2);
26     cout << "sum=" << sum << endl;
27
28     system("pause");
29     return 0;
30 }
```

总结：函数定义里小括号内称为形参，函数调用时传入的参数称为实参

6.4 值传递

- 所谓值传递，就是函数调用时实参将数值传入给形参
- 值传递时，如果形参发生，并不会影响实参

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  #include<string>
4  using namespace std;
```



```

5
6 //值传递
7 // 定义函数，两个数字进行交换函数
8 void swap(int num1, int num2)
9 {
10
11     cout << "交换前: " << endl;
12     cout << "num1=" << num1 << endl;
13     cout << "num2=" << num2 << endl;
14
15     int temp = num1;
16     num1 = num2;
17     num2 = temp;
18
19     cout << "交换前: " << endl;
20     cout << "num1=" << num1 << endl;
21     cout << "num2=" << num2 << endl;
22
23     return; //或者都不需要写，或者返回值不需要的时候，可以不写return
24 }
25
26 int main()
27 {
28
29     int a = 10;
30     int b = 20;
31     cout << "a=" << a << endl;
32     cout << "b=" << b << endl;
33     // 当我们做值传递的时候，函数的形参发生改变，并不会影响实参
34     swap(a, b);
35
36     cout << "a=" << a << endl;
37     cout << "b=" << b << endl;
38     system("pause");
39     return 0;
40 }

```

总结：值传递时，形参是修饰不了实参的

6.5 函数的常见样式

常见的函数样式有4种

1. 无参无返
2. 有参无返
3. 无参有返
4. 有参有返

示例：

```

1 #include<iostream>
2 #include<ctime> //time系统时间头文件
3 #include<string>

```

```

4  using namespace std;
5
6  //函数常见样式
7
8  //1、无参无返
9  void test01()
10 {
11     cout << "this is test01" << endl;
12
13 }
14 //2、有参无返
15 void test02(int a)
16 {
17     cout << "this is test02 a=" << a << endl;
18     return;
19 }
20 //3、无参有返
21 int test03()
22 {
23     cout << "this is test03" << endl;
24     return 100;
25
26 }
27
28 //4、有参有返
29 int test04(int a)
30 {
31     cout << "this is test04 a=" << a << endl;
32     return a;
33 }
34
35 int main()
36 {
37     // 无参无返函数调用
38     test01();
39     // 有参无返函数调用
40     test02(100);
41     // 无参有返函数调用
42     int num1 = test03();
43     cout << "num1=" << num1 << endl;
44     // 有参有返函数调用
45     int num2 = test04(1000);
46     cout << "num2=" << num2 << endl;
47     system("pause");
48     return 0;
49 }

```

6.6 函数的声明

作用：告诉编译器函数名称及如何调用函数。函数的实际主体可以单独定义。

- 函数的**声明**可以多次，但是函数的**定义**只能有一次

示例：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  #include<string>
4  using namespace std;
5
6  // 提前告诉编译器函数的存在，可以利用函数的声明
7  int max(int a, int b); // 函数声明
8
9
10 // 声明可以多次，定义只能一次
11 // 函数声明
12 // 比较函数，实现两个整型数字进行比较，返回较大的值
13 int max(int a, int b)
14 {
15     return a > b ? a : b; // 三目运算符
16 }
17
18
19 int main()
20 {
21     int a = 10;
22     int b = 20;
23     cout << max(a, b) << endl;
24     system("pause");
25     return 0;
26 }
```

6.7 函数的分文件编写

作用：让代码结构更加清晰

函数分文件编写一般有4个步骤

1. 创建后缀名为.h的头文件
2. 创建后缀名为.cpp的源文件
3. 在头文件中写函数的声明
4. 在源文件中写函数的定义

示例：

```
1 //swap.h文件
2 #include<iostream>
3 using namespace std;
4
5 // 函数的声明
6 void swap(int a, int b);
7
```

```
1 //swap.cpp文件
2 #include "swap.h" //自定义文件
3
4 //函数的定义
5 void swap(int a, int b)
6 {
7     int temp = a;
8     a = b;
9     b = temp;
10    cout << "a=" << a << endl;
11    cout << "b=" << b << endl;
12
13 }
```

```
1 //main函数文件
2 #include<iostream>
3 using namespace std;
4
5 #include "swap.h"
6
7 // 1、创建.h后缀名的头文件 swap.h
8 // 2、创建.cpp后缀名的源文件 swap.cpp
9 // 3、在头文件中写函数的声明
10 // 4、在源文件中先函数的定义
11
12 int main()
13 {
14     int a = 10;
15     int b = 20;
16     swap(a, b);
17
18     system("pause");
19     return 0;
20 }
```

7、指针
