

此题未设置答案，请点击右侧设置按钮

对于C++的支持程序方式，下列说法正确的是（）

- ☐ A C++只支持结构化程序设计
- ☐ B C++只支持面向对象程序设计
- ☐ C C++既支持结构化程序设计又支持面向对象程序设计
- ☐ D 上述说法全不正确

提交

此题未设置答案，请点击右侧设置按钮

在C++中，源文件变为可执行程序的正确顺序应该是（）

- ☐ A 编辑、连接、编译、执行
- ☐ B 编辑、编译、连接、执行
- ☐ C 编译、编辑、连接、执行
- ☐ D 编译、连接、编辑、执行

提交

此题未设置答案，请点击右侧设置按钮

下列程序中，注释有误的是（）

- ☐ A `cout<<" 请输入两个数值：\n" ; // 提示输入数值`
- ☐ B `cout<<" 请输入两个数值：\n" ; /* 提示输入数值`
- ☐ C `cout<<" 请输入两个数值：\n" ; /* 提示输入数值 */`
- ☐ D `cout<<" 请输入两个数值：\n" ; /* // 提示输入数值 */`

提交

此题未设置答案，请点击右侧设置按钮

C++语言对C语言做了很多改进，下列描述中，使得C语言发生了质变，从面向过程变成了面向对象（）

- ☐ A 增加了一些新的运算符
- ☐ B 允许函数重载，并允许设置缺省参数
- ☐ C 规定函数说明必须用原型
- ☐ D 引进了类和对象的概念

提交

此题未设置答案，请点击右侧设置按钮

下列语言属于计算机低级语言的是（）

- ☐ A C语言
- ☐ B Basic语言
- ☐ C 汇编语言
- ☐ D C++语言

提交



桂林电子科技大学
GUILIN UNIVERSITY OF ELECTRONIC TECHNOLOGY



人工智能学院
SCHOOL OF ARTIFICIAL INTELLIGENCE

第二讲 C++程序设计基础

赵彬

zhaobin@guet.edu.cn

2.1 C++的数据类型

数据类型指明变量或表达式的状态和行为，数据类型决定了数的取值范围和允许执行的运算符集。C++语言数据类型可以分为两大类：**基本类型**和**非基本数据类型**。

基本类型是指**不能再分解**的数据类型；

非基本数据类型是用基本数据类型构造的**用户自定义**数据类型，用于对复杂的数据对象进行描述和处理。

数据类型

基本数据类型

整数型(int)
实数型(float)
字符型(char)
逻辑型(bool)
空值型(void)

非基本数据类型

数组型([])
指针型(*)
枚举型(enum)
结构型(strut)
联合型(union)
引用型(&)
类类型(class)

2.1 C++的数据类型

表1 基本数据类型及其取值范围

类别	数据类型	字节数	数据表达范围
bool	bool	1	false, true(0,1)
int	基本型 (int)	4	-2 147 483 648 ~ 2 147 483 647
	短整型 (short)	2	-32 768 ~ 32 767
	长整型 (long)	4	-2 147 483 648 ~ 2 147 483 647
	无符号整型 (unsigned [int])	4	0 ~ 4 294 967 295
	无符号短整型 (unsigned short [int])	2	0 ~ 65 535
	无符号长整型 (unsigned long [int])	4	0 ~ 4 294 967 295
	有符号整型 (signed [int])	4	-2 147 483 648 ~ 2 147 483 647
	有符号短整型 ([signed] short [int])	2	-32 768 ~ 32 767
	有符号长整型 ([signed] long [int])	4	-2 147 483 648 ~ 2 147 483 647
char	无符号unsigned char	1	0 ~ 255
	有符号[signed] char	1	-128 ~ 127
	wchar_t宽字符	2	0 ~ 65 535
float	单精度浮点型 (float)	4	$-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$
	双精度浮点型 (double)	8	$-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$
	长双精度浮点型 (long double)	10或8	$-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$

2.1 C++的数据类型

关于表1的说明

- 表中的[]表示默认，代表括号中内容可选。signed [int], 表示只有signed时，默认是有符号整型数据
- 基本数据类型修饰符有四种，即long（长型符）、short（短型符）、signed（有符号型）、unsigned（无符号型），可以使用一个或多个类型修饰
- 整型数据的长度随着系统的不同而不同，在16位系统下，长度与短整型相同，在32位系统下，其长度为32位。默认int为有符号整型signed int
- wchar_t是C/C++的字符类型，是一种扩展的存储方式，wchar_t数据类型一般为16位或32位，超过char类型

2.1 C++的数据类型

C++字符集和标识符

(1) 字符集

字符集是构成C++程序语句的最小元素，C++程序语句（字符串除外），只能由字符集中的字符构成。

C++的字符集组成如下：

- 26个小写字母：a ~ z。
- 26个大写字母：A ~ Z。
- 10个数字：0 ~ 9。
- 标点和特殊字符：+ - * / , ; : ? \ " ' ~ | ! # % & () [] { } ^ < > 空格。
- 空字符：ASCII码为0的字符，用作字符串的结束符。

2.1 C++的数据类型

C++字符集和标识符

(2) 标识符

由字母、下划线和数字组成的字符序列，用来命名C++程序中的常量、变量、函数、语句标号及类型定义符等。

- 可以大小写字母或下划线开头，第1个不能是数字。
- 大写字母和小写字母分别代表不同的标识符。
- 不能是C++的关键字。

如: Aa、ABC、A_Y、ycx11、_name是合法标识符;
而5xyz、m.x、!abc、x-y是非法标识符。

2.1 C++的数据类型

C++字符集和标识符

(3) 关键字

关键字(keyword)又称**保留字**，是整个语言范围内预先保留的**标识符**，有特殊的含义。经过预处理后，关键字从预处理记号中区别出来，剩下的标识符，用于声明对象、函数、类型、命名空间等。关键字都是保留字，用户不可再重新定义作为标识符。

注意啦！



- 一般标识符的命名除了要符合定义规则外，最好还要有意义、简洁、易区分，这样可以增加程序的可读性；
- 关键字不用死记硬背，边学习边积累；
- 当使用关键字作为标识符时，系统会给出警告。

2.1 C++的数据类型

C++字符集和标识符

表2 关键字表

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	

2.1 C++的数据类型

关键字分类

- 数据类型相关关键字
- 语句相关关键字
- 定义、初始化相关关键字
- 系统操作相关关键字
- 命名相关关键字
- 函数和返回值相关关键字
- 其他关键字

2.1 C++的数据类型

关键字分类

数据类型相关关键字

- bool、true、false 逻辑布尔类型、真、假
- char、wchar_t 字符型和宽字符型
- int、double、float、short、long、signed、unsigned 数值修饰型
- signed、unsigned 修饰整数类型，有符号和无符号
- double和float用于浮点数，double双精度，long double 长双精度
- explicit 避免自定义类型隐式转换为类类型
- auto 根据初试值自动推断变量的数据类型（不是每个编译器都支持auto）

2.1 C++的数据类型

关键字分类

语句相关关键字

- switch、case、default 分支语句
- do、for、while 循环语句
- if和else 条件语句
- break、continue、goto 跳转语句

2.1 C++的数据类型

关键字分类

定义、初始化相关关键字

- public、protected、private 权限修饰符，公有的、保护的、私有的
- template 模板
- static 静态全局变量，静态局部变量，也可以修饰函数和类中的成员函数
- struct、class、union 结构体、类、联合体
- mutable 被mutable修饰的变量，将永远处于可变的动态
- virtual 声明虚基类，虚函数，实现多态
- enum 构成枚举类型名的关键字
- export 实现模板函数的外部调用
- extern 声明变量
- const、volatile 类型修饰符

2.1 C++的数据类型

关键字分类

系统操作相关关键字

- catch、throw、try 用于异常处理
- new、delete 内存申请、释放
- friend、inline 友元函数或友元类、内联函数
- operator和操作符连用，指定一个重载了的操作符函数，比如operator+
- register 提示编译器尽可能把变量存入到CPU内部存储器中
- typename 告诉编译器把一个特殊的名字解释为一个类型

2.1 C++的数据类型

关键字分类

命名相关关键字

- using、name、std 使用命名空间
- typedef 声明，为现有的数据类型创建一个新的名字，便于程序的阅读和编写

函数和返回值相关关键字

- void 特殊的“空”类型，指定函数无返回值或无参数
- return 返回，可附带一个返回值
- sizeof 返回类型名或表达式具有的类型对应的大小
- typeid 返回结果类型信息

其他关键字

- this 每个类成员函数都隐含了一个this指针，用来指向类本身
- asm 嵌入汇编指令关键字

2.2 常量与变量

常量

常量通常指在程序运行过程中不能被改变的量。常量可以是任何的基本数据类型，可以为整型数字、浮点数字、字符、字符串和布尔值。常量包括字面常量，字符常量，符号常量等多种类型。

字面常量

程序中表示的直接参加运算的数，被称为字面常量或字面值、常量。字面是指程序中直接用符号表示的数值，而不是机器码。

字面常量的类型可以根据书写形式来区分的，例如15，-2.2，'a',"hello"等都是字面常量，它们的类型分别为：整型、浮点型、字符型、字符串型，每个字面常量的字面本身就是它的值。

2.2 常量与变量

常量

字符常量

字符常量所表示的值是字符型变量所包含的值。

注意啦！



- 字符常量可以赋值给字符变量，如"char b='a';"，但不能把一个字符串常量赋给一个字符变量，也不能对字符串常量赋值
- 字符串中的字符依次存储在内存中一块连续的区域，并且把空字符' \0'自动附加到字符串的尾部作为字符串的结束标志。故字符个数为n的字符串在内存中应占(n+1)个字节

2.2 常量与变量

常量

字符常量

字符常量所表示的值是字符型变量所包含的值。

(1) 普通的字符常量

用一对单引号括起来的一个字符就是字符型常量，如 ‘a’， ‘T’ 都是合法的字符常量。

(2) 转义字符常量

在C++语言中，有一些字符用于控制输出或编译系统本身保留，无法作为字符常量来表示。为了表示这些特殊字符，C++中引入了转义字符的概念。C++语言规定，采用反斜杠后跟一个字母来代表一个控制字符，反斜杠后的字符不再作原有的字符使用，而转义为具有某种控制功能的字符，称为转义字符。

2.2 常量与变量

常量

字符常量

表3 转义序列表

转义序列	含义	ASCII码值（16/10进制）
\\	反斜杠字符	5CH/092
\'	'字符	27H/039
\"	"字符	22H/034
\?	? 字符	3FH/063
\a	警报铃声	07H/007
\b	退格键	08H/008
\f	换页符	0CH/012
\n	换行符	0AH/010
\r	回车	0DH/013
\t	水平制表符	09H/009
\v	垂直制表符	0BH/011
\ddd	任意字符	三位八进制数
\xhh	任意字符	二位十六进制数

2.2 常量与变量

常量

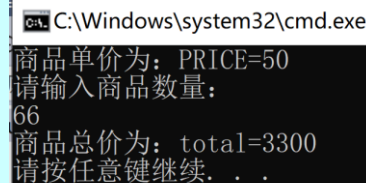
符号常量

用一个标识符来表示一个常量，称之为符号常量。符号常量在使用之前必须先进行定义。其一般形式为：**#define 标识符 常量**

【例】已知商品的单价是50元，并将其定义为常量，商品数量为10，求商品的总价并进行输出。

```
#include <iostream>
using namespace std;
#define PRICE 50
int main ( )
{ int num,total;//num代表购货数量,total代表总货款
  cout<<"商品单价为: PRICE="<<PRICE<<endl;
  cout<<"请输入商品数量: "<<endl;
  cin>>num;
  total=num * PRICE;//PRICE是符号常量,代表50(单价)
  cout<<"商品总价为: total="<<total<<endl;
  return 0;
}
```

运行结果



```
C:\Windows\system32\cmd.exe
商品单价为: PRICE=50
请输入商品数量:
66
商品总价为: total=3300
请按任意键继续. . .
```


2.2 常量与变量

常量

符号常量(续)

注意啦！



- #define定义的符号常量，末尾不要有分号，是预处理语句
- 习惯上符号常量的标识符用大写字母表示，变量标识符用小写字母表示
- 使用符号常量的好处是，含义清楚，能做到一改全改

2.2 常量与变量

变量

变量用于保存程序运算过程中所需要的原始数据、中间运算结果和最终结果，其值是可以改变的量。

变量有类型、名字和值三个要素。通过定义变量，程序给该变量一个标识符作为变量名，指定该变量的数据类型，并根据数据类型分配存储空间的大小。某个变量的值被改变后，将一直保持到下一次被改变。

变量的定义形式：

数据类型 变量名列表；

例如：

```
short len;
```

2.2 常量与变量

变量

变量的作用域

(1) 在函数体内部

局部变量，不同函数体内部可以定义相同名称的变量，而互不干扰

(2) 形式参数

当定义一个有参函数时，函数名后面括号内的变量统称为形式参数

```
int func(int x, int y)
```

```
{  if (x>y)
    return x;
    else
    return y;
}
```

2.2 常量与变量

变量

变量的作用域（续）

（3）全局变量

在所有函数体的外部定义的变量，其作用范围是整个程序，并在整个程序运行期间有效。

：： 同名变量：对被隐藏的同名全局变量进行访问

```
int x;  
void f()  
{  
    {  
        int x;  
        x=1; //引用局部变量x  
        ::x=2; //引用全局变量x  
    }  
    x=3; //引用全局变量x  
}
```

2.2 常量与变量

变量

变量的赋值

定义变量的同时，也可以给它赋予一个初值，说明它代表的的数据是什么，称为变量的初始化

```
int a=3,b=6*(3+5);
```

```
double sum=0.618;
```

```
char c1='a',c2='b'
```

同一个初值分别同时赋给变量a,b,c，可以有如下两种写法：

写法一：

```
int a=8,b=8,c=8;
```

写法二：

```
int a,b,c=8;
```

```
a=b=c;
```

🔔注意：

✘ int a=b=c=8;是错误的。

✘ 如果定义变量时没有赋初值，则该变量的值是不可预测的，即对应的存储单元内容是不确定的，若该变量参与运算则会导致程序的逻辑错误。

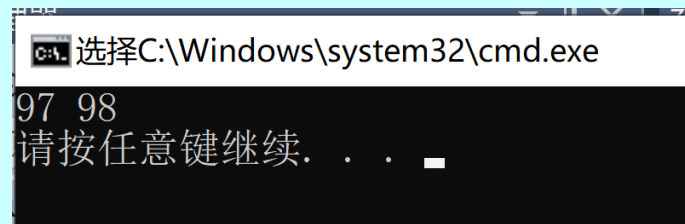
2.2 常量与变量

变量

【例】将字符常量赋给整型变量，并查看输出结果

```
/******  
功 能：整型变量实例  
*****/  
  
#include <iostream>  
using namespace std;  
int main()  
{  
    int i,j;    //i,j是整型变量  
    i='a';    //将字符常量赋给整型变量  
    j='b';  
    cout <<i<<' '<<j<<'\n'; //输出整型变量i, j的值,  
    return 0;  
}
```

运行结果



2.2 常量与变量

变量

常量

在定义变量时，如果加上关键字`const`，则变量的值在程序运行期间不能改变，这种变量称为常量(`constant variable`)。

常量定义语句同变量定义语句类似，其语法格式为：

`const 类型名 常量名=<初值表达式>`

例如：`const int A=3;`

//用`const`来声明这种变量的值不能改变，指定其值始终为3

关键字`const`指明这是一条常量定义语句，后面跟着常量的类型名，接着是常量名，它是一个用户定义的标识符，常量名之后为一个赋值号和一个初值表达式。该语句也可以定义多个常量。

常量在定义时，必须同时对它初始化。



2.2 常量与变量

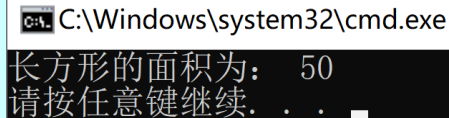
变量

常变量

【例】将长方形的长和宽定义为常变量，并算得面积进行输出。

```
#include <iostream>
using namespace std;
int main()
{
    const int LENGTH = 10;
    const int WIDTH = 5;
    const char NEWLINE = '\n';
    int area;
    area = LENGTH * WIDTH;
    cout << "长方形的面积为: " << area;
    cout << NEWLINE;
    return 0;
}
```

运行结果



```
C:\Windows\system32\cmd.exe
长方形的面积为: 50
请按任意键继续. . .
```


2.2 常量与变量

变量

常变量

- 常变量通常定义为大写字母形式，具有较好的可读性。常变量定义的最后一定要有分号
- 定义一个常量的时候，`const`和`#define` 都可以达到效果，但是一般采用`const`， 因为`#define`只是简单的符号替代，而`const`可以进行类型检查
- 定义的位置可以在函数体外或函数体内，只是作用域不同
- `#define`不是常变量，因此也没有作用域，也就是说如果你没有`#undef`的话，是一直有效的。而`const`的话是具有作用域的

注意啦！



2.3 运算符与表达式

对各种类型的数据进行加工的过程称为**运算**，表示各种不同运算的符号称为**运算符**，参与运算的数据称为**操作数**。

表达式是运算符与数据连接起来的表达运算的式子，表达式也称运算式。

根据参加运算对象的个数，C++语言的运算符可以分为：

- 一元运算符，或称“单目算符”，即参加运算对象的数目为一个
- 二元运算符，或称“双目算符”，即参加运算对象的数目为两个
- 三元运算符，或称“三目算符”，即参加运算对象的数目为三个

2.3 运算符与表达式

根据运算符功能划分，C++语言的运算符可以分为：

- 算术运算符
- 赋值运算符
- 关系运算符
- 逻辑运算符
- 条件运算符

2.3 运算符与表达式

➤ 算术运算符

表4 算术运算符

优先级	运算符	含义	结合性
2	+	正号	从右向左
	-	负号	
4	*	乘	从左向右
	/	除	
	%	取余	
5	+	加	
	-	减	

说明

(1) 算术运算符的意义、优先级与数学中一致

- +(正号), -(负号)是一元运算, 优先级高于二元运算
- *、/、%优先级高于+(加)、-(减)运算

2.3 运算符与表达式

➤ 算术运算符（续）

说明：（2）求余运算符%

- 要求两个操作数的值必须为整数或字符型数
- 当两个操作数都是整数时，结果为正
- 如果一个（或两个）操作数为负，余数的符号取决于机器

例： 21%6 //结果是3 4%2 //结果是0

 21%-5 //机器相关：结果为-1或1

（3）除法运算符/

- 当用于两个整数相除时，如果商含有小数部分，将被截掉
- 若要进行通常意义的除法运算，则至少保证除数或被除数中有一个是浮点数或双精度数

例：

5/4 //结果是1

4/5 //结果是0

5/4.0 //结果是1.25

4.0/5 //结果是0.8

2.3 运算符与表达式

➤ 赋值运算符

C++语言提供了两类赋值运算符：**基本赋值运算符**和**复合赋值运算符**。

赋值运算符是双目运算符，从右向左进行。例如，`sum1=sum2=0`相当于`sum1=(sum2=0)`，先执行`sum2=0`，后执行`sum1=sum2`。

基本赋值运算符“=”是将一个数据赋给一个变量。**复合赋值运算符**是在赋值运算符的前面加上其他的运算符组合而成的新运算符。

2.3 运算符与表达式

➤ 赋值运算符（续）

表5 赋值运算符种类

优先级	运算符	含义	格式	举例
15	=	赋值运算符	变量=表达式	a=3;a的值为3
	/=	除后赋值	变量/=表达式	a/=3;即a=a/3
	=	乘后赋值	变量=表达式	a*=3;即a=a*3
	%=	取模后赋值	变量%=表达式	a%=3;即a=a%3
	+=	加后赋值	变量+=表达式	a+=3;即a=a+3
	-=	减后赋值	变量-=表达式	a-=3;即a=a-3
	&=	位与赋值	变量&=表达式	a&=3;即a=a&3
	^=	位异或赋值	变量^=表达式	a^=3;即a=a^3
	<<=	左移赋值	变量<<=表达式	a<<=3;即a=a<<3
	>>=	右移赋值	变量>>=表达式	a>>=3;即a=a>>3

2.3 运算符与表达式

➤ 关系运算符

在 C++ 中比较数值数据时可以使用**关系运算符**。**字符也可以使用这些操作符进行比较**，因为在 C++ 中字符被认为是数值。每个关系运算符确定两个值之间是否存在特定的关系。

表6 关系运算符

优先级	关系运算符	含义
7	>	大于
	<	小于
	>=	大于或等于
	<=	小于或等于
8	==	等于
	!=	不等于

2.3 运算符与表达式

➤ 关系运算符（续）

注意啦！



关系运算符中的等号运算符是两个=符号在一起。不要将此运算符与赋值运算符混淆，后者只有一个=符号。==运算符确定变量是否等于另一个值，而=运算符则是将运算符右侧的值赋给左侧的变量。

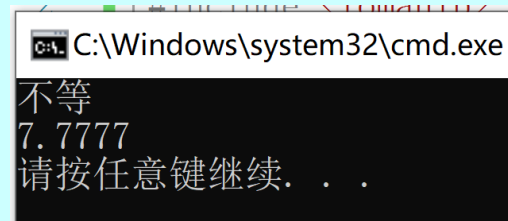
2.3 运算符与表达式

➤ 关系运算符（续）

由于浮点数在计算机内进行运算和存储时会产生误差，因此在比较两个浮点数时，建议不要直接比较两数是否相等。

```
double d1 = 3.3333, d2 = 4.4444;  
if(d1 + d2 == 7.7777)  
    cout<<"相等"<<endl;  
else{  
    cout<<"不等"<<endl;  
    cout<<d1 + d2<<endl;  
}
```

运行结果



```
C:\Windows\system32\cmd.exe  
不等  
7.7777  
请按任意键继续. . .
```

在if条件语句中用“==”来判断浮点数是否相等，结果是不等，但d1+d2输出结果却是7.7777。
判断两个实数是否相等的正确方法是：判断两个实数之差的绝对值是否小于一个给定的允许误差数，如：

```
fabs(d1 - d2) <= 1e-6           //判断d1是否等于d2时;  
fabs(d1 + d2 - 7.7777) <= 1e-6 //判断d1与 d2的和是否等于7.7777时;  
其中，fabs()是计算绝对值的一个库函数，使用时要包含头文件math.h。
```

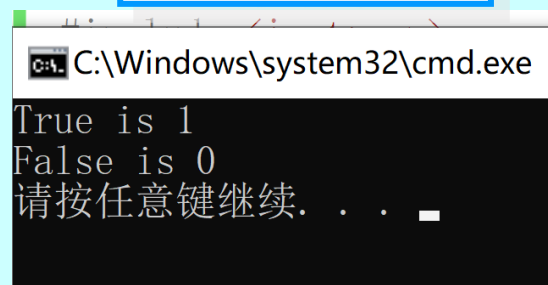
2.3 运算符与表达式

➤ 关系运算符（续）

【例】定义两个布尔型逻辑变量，执行相关的关系运算，并进行输出。

```
#include <iostream>
using namespace std;
int main()
{
    bool trueValue, falseValue;
    int x=5, y = 10;
    trueValue = (x < y);
    falseValue = (y == x);
    cout << "True is " << trueValue << endl;
    cout << "False is " << falseValue << endl;
    return 0;
}
```

运行结果



```
C:\Windows\system32\cmd.exe
True is 1
False is 0
请按任意键继续. . .
```

2.3 运算符与表达式

➤ 逻辑运算符

逻辑运算符实现逻辑运算，用于复杂的逻辑判断，一般以关系运算的结果作为操作数。可以将两个或多个关系表达式连接成一个，使表达式的逻辑反转。

表7 逻辑运算符

优先级	运算符	含义	效果
12	&&	与	两个表达式必须都为 true，整个表达式才为 true
13		或	必须有一个或两个表达式为 true，才能使整个表达式为 true。
2	!	非	反转一个表达式的逻辑值。 使表达式从 true 变成 false，或者从 false 变成了 true。

2.3 运算符与表达式

➤ 逻辑运算符

&& 运算符

&& 运算符被称为逻辑与运算符。它需要两个表达式作为操作数，并创建一个表达式，只有当两个子表达式都为 true 时，该表达式才为 true。

```
if ((temperature < 20) && (minutes > 12))  
    cout << "The temperature is in the danger zone.";
```

语句说明：

这两个被逻辑与运算符连接在一起的表达式都是完整的表达式，它们的值可以被评估为 true 或 false。首先评估 temperature < 20 以产生 true 或 false 的结果，然后评估 minutes > 12 以产生 true 或 false 的结果，最后，这两个结果被 AND 并列在一起，以得出整个表达式的最终结果。

仅当 temperature 小于 20 且 minutes 大于 12 时，cout 语句才会执行，其中只要有一个表达式的结果被评估为 false，则整个表达式为 false，不执行 cout 语句。

2.3 运算符与表达式

➤ 逻辑运算符

|| 运算符

```
if ((temperature < 20) || (temperature > 100))  
    cout << "The temperature is in the danger zone.";
```

语句说明:

- 如果 temperature 小于 20 或者大于 100, 那么 cout 语句将被执行。这两个子表达式只要其中一个为 true, 则整个表达式为 true, 执行 cout 语句。
- 如果 || 运算符左侧的子表达式为 true, 则右侧的子表达式将不被检查, 因为只要有一个子表达式为 true, 那么整体表达式就可以被评估为 true。
- 被逻辑或连接在一起的应该都是逻辑表达式, 它们的值为 true 或 false, 按以下形式书写的if条件是不正确的: if (temperature <20 || >100)

2.3 运算符与表达式

➤ 逻辑运算符

! 运算符

```
if (!(temperature > 100))  
    cout << "You are below the maximum temperature. \n";
```

语句说明:

- 首先，表达式 `(temperature > 100)` 相当于问“温度是不是不大于 100?”或“温度大于 100 是不是假的?” 将被测试为 `true` 或 `false`，然后 `!` 运算符被应用于该值。
- 如果表达式 `(temperature > 100)` 为 `true`，则 `!` 运算符返回 `false`。如果为 `false`，则 `!` 运算符返回 `true`。

2.3 运算符与表达式

➤ 条件运算符

条件运算符“?:”是C++中位移的一个三目运算符，其使用的一般形式为：
表达式1? 表达式2: 表达式3

思考：以下语句的功能是什么？

```
x == y ? 'Y' : 'N'
```

```
(d = b*b-4*a*c) >= 0 ? sqrt(d) : sqrt(-d)
```

//sqrt是开平方函数

```
ch = ch >= 'A' && ch <= 'Z' ? ch+'a'-'A': ch;
```

//大写字母转小写

```
max=((a>b)?a:b)
```

//将a和b二者中较大的一个赋给max。

```
min=(a<b)?a:b
```

//将a和b二者中较小的一个赋给min。

2.3 运算符与表达式

➤ 其他运算符

++、--运算符

++、--运算符可以放置在变量前面，或者变量后面，都可以使操作数的值增1或减1，但对表达式的值的影响却完全不同。前置是自身先变化，后参与运算，后置是先参与运算，后变化。

表8 ++、--运算符的优先级和结合顺序

优先级	运算符	含义	结合性
1	++	后置自增	从左向右
	--	后置自减	
2	++	前置自增	从右向左
	--	前置自减	

Int i=5; x=i++; y=i; i=?,x=?,y=?

Int i=5; x=++i; y=i; i=?,x=?,y=?

调试以下程序，并改正错误。

```
Using namespace std
#include<iostream>;
Using std::endl;
int main()
float num1,num2,num3;
cin<<num1<<num2<<num3;
cout>> “The average is” >>setw(30)>> num1*num2*num3>>endl;
}
```

正常使用主观题需2.0以上版本雨课堂

作答

2.3 运算符与表达式

➤ 其他运算符

++、--运算符

【例】定义a, b, m, n变量, 赋值, 进行不同的运算, 并进行输出。

```
#include<iostream>
using namespace std;
int main(){
    int a;
    a=7*2+-3%5-4/3;
    float b;
    b=510+3.2e3-5.6/0.03;
    cout<<a<<"\t"<<b<<endl;
    int m(3),n(4);
    a=m++- --n;
    cout<<a<<"\t"<<m<<"\t"<<n<<endl;
    return 0;
}
```

运行结果

```
10      3523.33
0        4        3
请按任意键继续. . .
```

2.3 运算符与表达式

➤ 其他运算符

位运算符

位运算符是对其操作数按二进制形式逐位进行运算，参与运算的操作数都应为整型数，不能是实型数。c++提供了6个位运算符： \sim (按位求反)， $\&$ (按位与)， $|$ (按位或)， \wedge (按位异或)， \ll (左移位)， \gg (右移位)。

例如 ~ 9 的运算为：

$\sim(0000000000001001)$ 结果为：111111111110110

例如：9 $\&$ 5可写算式如下：

00001001 (9的二进制补码)

$\&00000101$ (5的二进制补码)

00000001 (1的二进制补码)

例如：9 $|$ 5可写算式如下：

00001001 (9的二进制补码)

$|00000101$ (5的二进制补码)

00001101 (十进制为13)

例如9 \wedge 5可写成算式如下：

00001001

$\wedge 00000101$

00001100 (十进制为12)

2.3 运算符与表达式

➤ 其他运算符

逗号运算符

由逗号运算符构成的表达式称为逗号表达式，其一般形式为：
表达式1，表达式2.....，表达式n

【例】假设b=2，c=7，d=5，则下面两个变量a1，a2各是多少？

a1=(++b,c--,d+3);

a2=++b,c--,d+3;

- 对于第一行代码，括号中有一个逗号表达式，包含三个部分，用逗号分开，所以最终的值应该是最后一个表达式的值，也就是d+3，为8，所以a1=8。
- 对于第二行代码，由于赋值运算符比逗号运算符优先级高，所以整个看成一个逗号表达式，这时逗号表达式中的三个表达式为a2=++b、c--、d+3，所以虽然最终整个表达式的值也为8，但a2=3。

2.3 运算符与表达式

➤ 其他运算符

sizeof运算符

用于测试某种数据类型或表达式的类型在内存中所占的字节数，它是一个一元运算符。其语法格式为：

sizeof(<类型名>)

或sizeof(<表达式>)

例如：

sizeof (int) //整数类型占4个字节，结果为4

sizeof (3+3.6) //3+3.6的结果为double实数，结果为8

2.3 运算符与表达式

表达式是由运算符、括号和操作数构成的序列，在运算时能计算出一个值的结果。运算对象包括常量、变量和函数等。

按运算符的不同，可以分为**算术表达式**、**赋值表达式**、**关系表达式**、**逻辑表达式**等。

2.4 类型转换

● 隐式类型转换

对数据类型不一致的两个运算量，系统会进行数据类型转换。即将其中的一个低级别类型的数据向另一个高级别类型的数据转换（按空间大小和数值范围），然后才进行相应的算术运算，运算的结果为高级别类型。这种按照固有的规则自动进行的内部转换称为隐式类型转换。

```
double f=7/3;  
int a='a'+3;
```



隐式类型转换时类型之间必须相容；
各整型之间，浮点与整型之间是相容的；
整型与指针之间，浮点与指针之间是不相容；（The first class）

● 强制类型转换

具体有两种形式：

（目标类型名） 待转换源数据
目标类型名 （待转换源数据）

例如：

```
float f=100.23;  
int x=(int)f; //或int x=int(f)
```

注意啦！

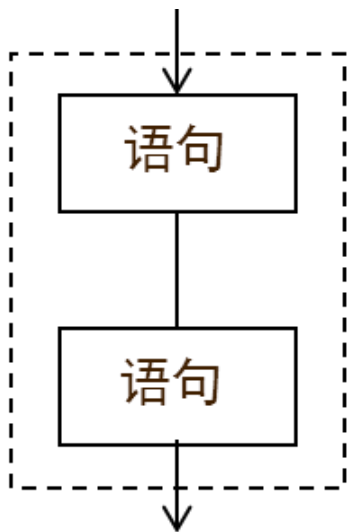


采用强制类型转换将高级别类型数据（按空间大小和数值范围）转换为低级别类型数据时，可能数据精度会受到损失。

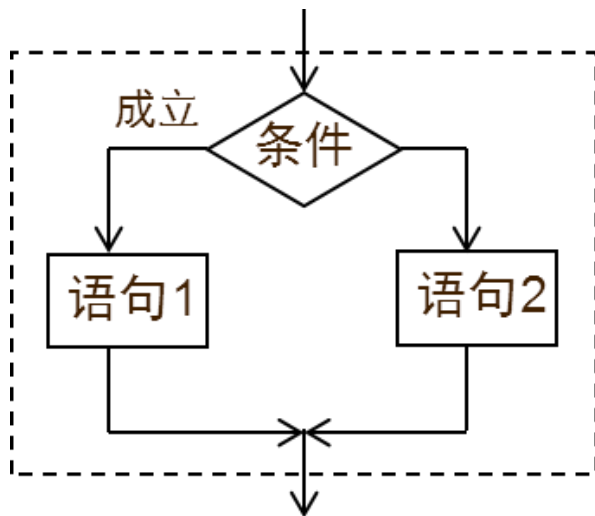
2.5 程序流程控制结构

顺序结构按照语句出现的先后顺序，自上而下的执行，是C++程序最基本的结构。默认的情况下采取顺序结构，除非特别指明，计算机总是按语句顺序一条一条地执行。

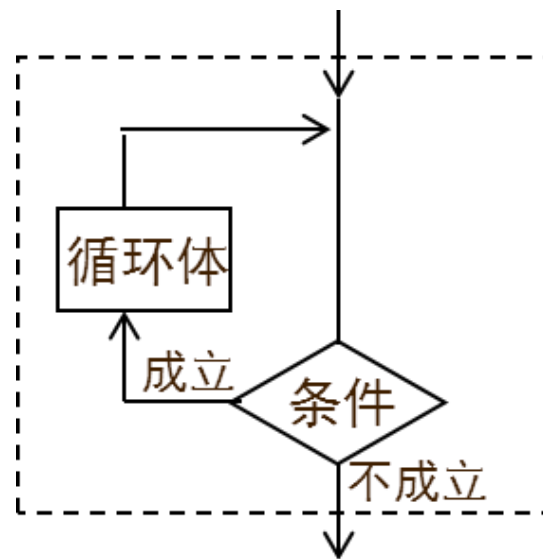
当遇到选择或需要循环工作时，会使用**选择结构**和**循环结构**。



顺序结构



选择结构



循环结构

图1 三种程序流程控制基本结构图

2.5 程序流程控制结构

➤ 顺序结构

顺序结构的程序是指程序中的所有语句都是按照书写顺序逐一执行的，只包含顺序结构的程序功能有限。

【例】 已知球的体积公式为如下，计算球的体积。 $V = 4\pi R^3/3$

```
#include<iostream>
using namespace std;
int main( )
{
    float radius , v ;
    cout << "球的radius : " ;
    cin >> radius ;           // 输入半径
    v = (4*3.1416* radius* radius* radius)/3 ;
    cout << "球的体积= " << v << endl ;       // 输出体积
    return 0;
}
```

2.5 程序流程控制结构

➤ 选择结构

选择类语句包括if语句和switch语句，用它们来解决实际应用中按不同的情况进行不同处理的问题。C++提供三种选择结构，即if选择结构、if-else选择结构和switch选择结构。

- if 选择结构称为单分支选择结构，选择或忽略一个分支的操作。
- if/else 选择结构称为双分支选择结构，在两个不同分支中选择。
- switch 选择结构称为多分支（或多项）选择结构，以多种不同的情况选择多个不同的操作。

2.5 程序流程控制结构

➤ 选择结构-if

格式 1:

if (条件表达式)

语句1;

格式 2:

if (条件表达式)

{

语句1;

}



- “条件表达式”要加括号，语句相对于if缩进两格
- 若执行的操作包含多个句子，必须将它们写在一个花括号内
- 书写复合语句时，左右花括号对齐，组成语句块的各语句相对花括号缩进一层并对齐

2.5 程序流程控制结构

➤ 选择结构-if

【例】读入a, b, 若 $a > b$, 则交换a, b的值并输出a, b

```
#include<iostream>
using namespace std;
int main ( )
{
    cout<<"请输入两个数: ";
    float a,b,c;
    cin>>a>>b;
    if (a>b)
    {
        c=a; a=b; b=c;
    }
    cout<<"a="<<a<<" b="<<b;
    return 0;
}
```

2.5 程序流程控制结构

➤ 选择结构-if-else

格式 1:

if (条件表达式)

 语句1;

else

 语句2;



➤ 如果“条件表达式”为真，则执行语句1，然后执行整个if...else语句的后续语句

➤ 如果“条件表达式”为假，则执行语句2，然后执行整个if...else语句的后续语句

格式 2:

if (条件表达式)

{ 语句1; ... }

else

{ 语句2; ... }

2.5 程序流程控制结构

➤ 选择结构-if-else

【例】输入温度 t 的值，判断是否适合晨练。 $(25 \leq t \leq 30)$ ，则适合晨练ok，否则不适合no)

```
#include<iostream>
using namespace std;
int main()
{
    int t;
    cout<<"请输入温度: ";
    cin >> t;
    if ((t>=25) &&(t<=30))
        cout<<"ok! 适合晨练! \n";
    else
        cout<<"no! 不适合晨练\n";
    return 0;
}
```

2.5 程序流程控制结构

➤ 选择结构-if语句的嵌套

格式 1:

```
if (条件表达式1)
    if (条件表达式)
        {...}
else
    {...}
else
    if (条件表达式3)
        {...}
else
    {...}
语句2;
{ 语句2; ...}
```

格式 2:

```
if (条件表达式1) 语句1;
else if (条件表达式2) 语句2;
else if (条件表达式3) 语句3;
...
else 语句n;
```


2.5 程序流程控制结构

➤ 选择结构-if语句的嵌套

【例】求三个数中的最大值并进行输出

```
#include<iostream>
using namespace std;
int main()
{
    int a1, a2, a3, max;
    cin>>a1>>a2>>a3;
    if (a1>=a2&&a1>=a3)    max=a1;
    else if(a2>=a1&&a2>=a3)    max=a2;
    else max=a3;
    cout<<"The largest number is "<<max<<endl;
    return 0;
}
```

2.5 程序流程控制结构

➤ 选择结构-if语句的嵌套

【例】根据用户输入的成绩，将成绩分成4个等级，90-100对应等级“优秀”，80-90对应等级“良好”，60-79对应等级“中等”，0-59分对应等级“不及格”

```
#include<iostream >
using namespace std;
int main()
{
    int c;
    cout<<"请输入一个成绩: ";
    cin >> c;
    if (c>=90)
        cout<<"成绩优秀! \n";
    else if (c>=80)
        cout<<"成绩良好! \n";
    else if (c>=70)
        cout<<"成绩中等! \n";
    else
        cout<<"成绩不及格! \n";
    return 0;
}
```

2.5 程序流程控制结构

➤ 选择结构-switch

switch语句又称为开关语句，其格式如下：

switch (表达式)

{

case 常量表达式1:

语句序列1

break;

case 常量表达式2:

语句序列2

break;

.....

case 常量表达式n:

语句序列n

break;

default:

语句序列n+1

break;

}

注意啦！



- switch 后的表达式类型必须与case后的常量表达式类型一致，而且只能是字符型、整型或枚举型（**不能是浮点型**）
- case后的常量表达式不能含有变量，同一个switch 中不能有重复的常量表达式
- 当表达式的值与某一个case后的常量表达式相等时，执行case后的语句
- break用于跳出整个循环
- 若都不满足条件，则执行default

2.5 程序流程控制结构

➤ 选择结构-switch

【例】 根据一个代表星期的0到6之间的整数，在屏幕上打印出它代表星期几

```
{
    int w ; //代表星期的整数
    cout << "Please enter the number of week : " ;
    cin >> w ;
    switch ( w ) {
        case 0 :
            cout << " It's Sunday ." << endl ;
            break ;
        case 1 :
            cout << " It's Monday ." << endl ;
            break ;
        case 2 :
            cout << " It's Tuesday ." << endl ;
            break ;
        .....
        case 5 : cout << " It's Friday ." << endl ;
            break ;
        case 6 :
            cout << " It's Saturday ." << endl ;
            break ;
        default : cout << "Invalid data !" << endl ;
    }
}
```

2.5 程序流程控制结构

➤ 循环结构

C++中有三种循环语句可用来实现循环结构：**while**语句、**do while**语句和**for**语句。这些语句各有各的特点，而且常常可以互相替代。在编程时应根据题意选择最适合的循环语句。

while(条件表达式)

{

语句; //循环体

}

do

{

语句; //循环体

} **while**(条件表达式);

for (表达式1;表达式2;表达式3)

{

语句; //循环体

}

- **do while**语句是先执行一次循环体之后，再根据条件表达式的值确定是否还要继续执行循环体，也即“先执行，后判断”，循环体至少会被执行一次，**条件表达式后面的分号不能少**
- 表达式1对循环控制变量初始化
- 表达式2表示循环是否结束的条件
- 表达式3对循环控制变量做修改

2.5 程序流程控制结构

➤ 循环结构-while

【例】计算100之内的奇数之和

```
#include<iostream>
using namespace std;
int main( )
{
    int n = 1 ;           // 奇数
    int sum = 0 ;         // 奇数的累加和
    while ( n < 100 ) {   // n不能超过100
        sum += n ;       // 累加
        n += 2 ;         // 修改为下一个奇数
    }
    cout << "The sum is : " << sum << endl ;
    return 0;
}
```

2.5 程序流程控制结构

➤ 循环结构-do-while

【例】根据下面的公式求 π 的值。计算到最后一项的绝对值小于 $1e-6$ 时停止

```

$$\pi/4 = 1 - 1/3 + 1/5 - 1/7$$
  
#include<iostream>  
using namespace std;  
int main( )  
{  
    double pi = 0 , x = 1 ;  
    int s = 1 ;  
    do {  
        pi = pi + s / x ;  
        x += 2 ;  
        s = - s ;  
    } while ( 1 / x >= 1e-6 ) ;  
    pi = pi * 4 ;  
    cout << "pi=" << pi << endl ;  
    return 0;  
}
```

2.5 程序流程控制结构

➤ 循环结构-for

【例】用for语句求1-100以内奇数的和

```
#include<iostream>
using namespace std;
int main( )
{
    int sum = 0 ;           // 奇数的累加和
    for ( int n = 1 ; n < 100 ; n = n + 2 )
        sum += n ;         // 累加
    cout << "The sum is : " << sum << endl ;
    return 0;
}
```


2.5 程序流程控制结构

➤ 循环结构-多重循环

【例】 观察二重循环的执行流程，分析下面程序的运行结果

```
#include<iostream>
using namespace std;
int main( )
{ cout<< "行列坐标相加的和为: " << endl;
  for ( int i = 1 ; i <= 4 ; i++ )
  {
    for ( int j = 1 ; j <= 5 ; j++ )
      cout << i + j << " ";
    cout << endl ;
  }
  return 0;
}
```

2.5 程序流程控制结构

➤ break和continue

- break语句

格式: *break*;

作用: 在switch语句中用来立即终止执行switch语句。在for、while、do-while语句中, break可用来跳出循环体, 提前结束循环, 不再执行剩余的若干次循环, 而转去执行该循环后面的语句。但是若是多重循环, break语句的作用只能退出离break最近的一层循环。

- continue语句 (结束本次循环)

格式: *continue*;

作用: 能使当前正在进行的循环(for, while, do-while)跳过continue; 后的循环体语句, 立即开始下一次循环的判断。

- 二者区别

continue语句通常用在循环体中, 只结束本次循环, 忽略循环体中位于它之后的语句, 重新回到条件表达式的判断, 而不是终止整个循环的执行。

break语句不仅可用于跳出switch语句, 还可用于跳出循环, 不再判断执行循环的条件是否成立。但它只能跳出它所在的循环语句或switch语句, 不能跳出外层的循环语句或switch语句。若想跳出外层语句, 还要在外层中使用break。

2.5 程序流程控制结构

➤ break和continue

【例】结合while和continue语句，从键盘接收10个整数，求它们的平方根，遇到负数则忽略并重新输入下一个数据

```
#include<iostream>
using namespace std;
#include<math.h>
int main( )
{
    int i = 1 , num ;
    double root ;
    while ( i <= 10 ) {
        cout << "Please enter a number : " ;
        cin >> num ;
        if ( num < 0 ) {      //若num是负数则回到循环开始处
            cout << "invalid number!" ;
            continue ;
        }
        root = sqrt(num) ;
        cout << root << endl ;
        i++ ;
    }
    return 0;
}
```

2.5 程序流程控制结构

➤ goto

`goto`语句也称为无条件转移语句，当程序执行到`goto`这个语句时，程序就转跳到标号后面的语句。

其一般格式如下：

语句标号： 语句；

.....

goto 语句标号；

例：分析以下语句流程，并思考其输出结果

```
int x=1;  
biaohao:   x=x+1;  
if(x<100)  goto biaohao;  
cout<<"x=100"<<endl;
```

2.6 本讲小结

1. 标识符：由字母、下划线和数字组成的字符序列，用来命名C++程序中的常量、变量、函数、语句标号及类型定义符等。**关键字(keyword)**又称保留字，是整个语言范围内预先保留的标识符。

2. 常量常指在程序运行过程中不能被改变的量。可以是任何的基本数据类型。

3. 变量用于保存程序运算过程中所需要的原始数据、中间运算结果和最终结果，其值是可以改变的量。

4. 基本数据类型包括布尔型、字符型、整型、浮点型、双浮点型、无类型、宽字符型等。

5. C++中的运算符又分为：算术运算符、关系运算符、逻辑运算符、条件运算符、赋值运算符等

6. 程序中除了顺序结构以外，通常还有选择结构、循环结构以及转移机制等程序流程控制方式。。

本次作业

问题1（使用选择结构实现） 某单位向职工按月发放医疗补贴的具体方案如下：

- 职工工龄在10年以下的，医疗补贴为其基本工资的10%；
- 工龄在10年以上20年以下的，医疗补贴为其基本工资的15% ；
- 工龄在20年以上30年以下的，医疗补贴为其基本工资的20% ；
- 工龄在30年以上的，医疗补贴为其基本工资的30%。
- 输入某职工的工龄及基本工资，能计算出他每月应得的医疗补贴。

问题2（使用循环结构实现）求三位数中的水仙花数。若abc是水仙花数，则有 $abc = a^3 + b^3 + c^3$ ，例如： $153 = 1^3 + 5^3 + 3^3$