



实验UNIT 02 简单程序设计

《程序设计》课程组



第2讲上机实验

◆ 实验目的：

1. 学会编写简单的C++程序
2. 基本数据类型变量和常量的应用
3. 运算符和表达式的应用
4. 结构化程序设计基本控制结构的运用
5. 简单的输入输出语句运用
6. 头文件的运用
7. Debug特殊功能：单步执行、设置断点、观察变量值



第2讲上机实验

◆ 实验任务：

1. 实验课堂练习1：基本数据类型
2. 实验课堂练习2：算术运算符和赋值运算符
3. 实验课堂练习3：关系和逻辑运算符、sizeof()、位运算符
4. 上机练习：结构化程序设计控制结构编程练习



第2讲上机实验

◆ 实验步骤提示：

1. 为每个题目建立一个新的控制台项目文件；
2. 向其中提交一个C++源文件；
3. 录入代码，检查是否有错误？有则改之；
4. 选择菜单“生成解决方案”编译源程序；
5. 执行程序，观察输出结果是否正确观察输出结果是否正确，如果有错误，可以执行第6步；
6. 使用debug功能：单步执行（step over）、断点执行、观察变量值和输出结果是否正确？



实验课堂练习1!

练习内容1：基本数据类型



补充2-1：读入并显示整数

主要知识点：

- ✓ 常量：源程序中直接写明的数据，其值在整个程序运行期间不可改变，这样的数据称为常量。
- ✓ 变量：在运行过程中从计算机的外部设备（例如键盘、硬盘）读取的，这些数据的值在程序运行过程中允许改变，这样的数据称为变量
- ✓ 从键盘输入数据：iostream类的对象cin的>>操作，可以从标准输入设备（通常是键盘）读入数据
- ✓ 数据的存储：
为了存储数据，需要预先为这些数据分配内存空间。
变量的定义就是在给变量命名的时候分配内存空间。



补充2-1：读入并显示整数

```
#include <iostream>
using namespace std;
int main()
{
    int radius;
    cout<<"Please enter the radius!\n";
    cin>>radius;
    cout<<"The radius is:"<<radius<<"\n";
    cout<<"PI is:"<<3.14<<"\n";
    cout<<"Please enter a different radius!\n";
    cin>>radius;
    cout<<"Now the radius is changed to:"<<radius<<"\n";
    return 0;
}
//观察：通过调试功能跟踪观察变量的值
```

补充2-2：为常量命名

主要知识点：符号常量

```
#include <iostream>
using namespace std;
int main()
{   const double pi(3.14159);
    int radius;
    cout<<"Please enter the radius!\n";
    cin>>radius;
    cout<<"The radius is:"<<radius<<"\n";
    cout<<"PI is:"<<pi<<"\n";
    cout<<"Please enter a different radius!\n";
    cin>>radius;
    cout<<"Now the radius is changed to:"<<radius<<"\n";
    cout<<"PI is still:"<<pi<<"\n";
    //cin>>pi;
    return 0;
}
```

//观察：通过调试工具跟踪观察符号常量。

//思考：能给常量输入新值吗？如定义pi时不初始化会怎样？



补充2-2 (续)

运行结果:

Please enter the radius!

2

The radius is:2

PI is:3.14159

Please enter a different radius!

3

Now the radius is changed to:3

PI is still:3.14159



补充2-3：变量的初始化

主要知识点：变量的初始化

- ✓ 虽然变量的值是可以是在运行时获得的，但是在定义变量时也可以进行初始化，而且应该提倡进行初始化；
- ✓ 未经初始化的变量，其值可能是随机的。如果误用了未经初始化也没有给予确定值的变量，就会引起错误。



补充2-3：变量的初始化（续）

```
#include <iostream>
using namespace std;
int main()
{
    const double pi(3.14159);
    int radius(0);
    cout<<"The initial radius is:"<<radius<<"\n";
    cout<<"PI is:"<<pi<<"\n";
    cout<<"Please enter a different radius!\n";
    cin>>radius;
    cout<<"Now the radius is changed to:"<<radius<<"\n";
    cout<<"PI is still:"<<pi<<"\n";
    return 0;
}
```

//观察：通过调试工具跟踪观察变量。

补充2-4：整数变量的定义与输出

主要知识点：有符号整数与无符号整数的差别

- ✓ 无符号整数unsigned short取值范围为0~65 535;
- ✓ 有符号整数short的取值范围为-32768~32767;
- ✓ 如果超出数值类型取值范围，则会出现数值溢出。



补充2-4：整数变量的定义与输出（续）

```
#include <iostream>
using namespace std;
int main()
{
    short int i;
    unsigned short int j;
    j = 50000;
    i = j;
    cout << "Short int is:"<<i<<endl;
    cout << "Short unsigned int is: "<<j<<endl;
    return 0;
}
```

//观察思考：将正数50000赋值给变量i以后，输出i的结果是什么？

//为什么？输出j的结果是什么？为什么？



补充2-5：不同类型整数的最值

主要知识点：整数类型变量

标准C++中有6种整数类型，它们是short、int、long、unsigned short、unsigned int 和 unsigned long。

C++头文件limits中定义了一系列符号常量来表示这些最值。



补充2-5：不同类型整数的最值（续）

```
#include <iostream>
#include <climits>
using namespace std;
int main()
{ cout <<"Min of short is: "<<SHRT_MIN<<endl;
  cout <<"Max of short is: "<<SHRT_MAX<<endl;
  cout <<"Min of int is: "<<INT_MIN<<endl;
  cout <<"Max of int is: "<<INT_MAX<<endl;
  cout <<"Min of long is: "<<LONG_MIN<<endl;
  cout <<"Max of long is: "<<LONG_MAX<<endl;
  cout <<"Max of unsigned short is: "<<USHRT_MAX<<endl;
  cout <<"Max of unsigned int is: "<<UINT_MAX<<endl;
  cout <<"Max of unsigned long is: "<<ULONG_MAX<<endl;
  return 0;
}
//观察：输出的值
```

补充2-6：整型文字常量的应用

标准C++中整数文字常量有下列形式。

- ✓ 3种进位计数制——八进制、十进制、十六进制
- ✓ 无符号整数文字常量（后缀u或U）
- ✓ 长整数文字常量（后缀l或L）

各种进制整数的输入与输出

- ✓ 程序中在输入与输出数据时可以使用操纵符来指出进制。dec是十进制操作符，oct是八进制操作符，hex是十六进制操作符。



补充2-6：整型文字常量的应用（续）

```
#include <iostream>
#include <limits>
using namespace std;
int main()
{
    int isample, osample, hsample;
    unsigned long ulsample;
    cin>>isample>>oct>>osample>>hex>>hsample;
    cout<<isample<<'<<oct<<osample<<'<<
        <<hex<<hsample<<endl;
    isample=123;
    osample=0173;
    hsample=0x7B;
    ulsample=4294967295UL;
```



补充2-6 (续)

```
cout<<dec<<isample<<';'<<oct<< isample  
    <<';'<<hex<< isample <<endl;  
cout<<dec<< osample<<';' <<oct<<osample  
    <<';'<<hex<< osample <<endl;  
cout<< dec<<hsample <<';'<<oct<< hsample<<';'  
    <<hex<<hsample<<endl;  
cout<<dec<< ulsample <<';'<<oct<<ulsample<<';'  
    <<hex<<ulsample<<endl;  
return 0;  
}
```

//观察：123、0173、0x7B是同一个数的不同进制表示形式，
//输入和输出时都可以采用不同的进制。



补充2-7：不同类型浮点数的应用

主要知识点：浮点类型文字常量

- ✓ 默认情况下浮点文字常量的类型是double,
- ✓ float的浮点文字常量，需要后缀f或者F
- ✓ long double的浮点文字常量，后缀l或者L
- ✓ C++标准没有规定每一种浮点类型的字节数，如果需要知道字节数，可以用sizeof运算得到。



补充2-7：不同类型浮点数的应用（续）

```
#include <iostream>
#include <limits>
using namespace std;
const float PI_FLOAT = 3.1415926f;
const double PI_DOUBLE = 3.1415926;
const long double PI_LDOUBLE = 3.1415926;
int main()
{
    float nRadiusFloat = 5.5f, nAreaFloat;
    double nRadiusDouble = 5.5, nAreaDouble;
    long double nRadiusLDouble = 5.5, nAreaLDouble;
    nAreaFloat = PI_FLOAT* nRadiusFloat* nRadiusFloat;
    nAreaDouble = PI_DOUBLE* nRadiusDouble *
        nRadiusDouble;
```



补充2-7 (续)

```
nAreaLDouble = PI_DOUBLE* nRadiusDouble*
                nRadiusDouble;
cout << "nAreaFloat = " << nAreaFloat << " ,
        sizeof(nAreaFloat) = " << sizeof(nAreaFloat) << endl;
cout << "nAreaDouble = " << nAreaDouble
        << " , sizeof(nAreaDouble) = "
        << sizeof(nAreaDouble) << endl;
cout << "nAreaLDouble = " << nAreaLDouble
        << " , sizeof(nAreaLDouble) = "
        << sizeof(nAreaLDouble) << endl;
return 0;
}
//观察运行结果
```



补充2-8：字符数据的应用

主要知识点：字符常量、字符串常量、转义序列

- ✓ 字符常量的一般形式是单引号括起来的一个字符；
- ✓ 一些不可显示的字符，无法通过键盘输入，可以用转义序列来表示。



补充2-8：字符数据的应用（续）

```
#include <iostream>
using namespace std;
int main()
{
    cout << 'A'<< ' '<<'a'<<endl;    //输出普通字符
    cout << "one\ttwo\tthree\n";      //使用水平制表符
    cout << "123\b\b45\n";             //使用退格符
    cout << "Alert\a\n";               //使用响铃键
    return 0;
}
//观察运行结果
```



补充2-9：定义bool类型的变量并进行算术运算

主要知识点：bool类型的应用

- ✓ 在算术运算表达式里，当表达式需要整数时，bool值将被转化为int，true为1，false为0。
- ✓ 如果需要将整数转换为bool值，那么0转换为false，所有非0值都为true。



补充2-9：定义bool类型的变量并进行算术运算（续）

```
#include <iostream>
using namespace std;
int main()
{
    bool bV1= true, bV2= false;
    cout <<"bool value bV1 = "<<bV1<<endl;
    cout <<"bool value bV2 = "<<bV2<<endl;
    int nV1=bV1, nV2 = 0;
    bV1 = nV2;
    cout <<"int value nV1 = "<< nV1<<endl;
    cout <<"bool value bV1 = "<<bV1<<endl;
    return 0;
}
```

//观察程序运行过程中内存中bool变量的值



实验课堂练习1结束!



实验课堂练习2!

练习内容2：算术运算符和赋值运算符



补充2-10：算术运算

主要知识点：算术运算符的使用和相关注意事项

- ✓ 算术运算符+、-、*、/的含义及运算次序与数学中是一样的，但是要注意两个整数相除时，结果是取整，小数部分会被截掉。
- ✓ %运算符的作用是两数相除，取余数作为结果。
- ✓ ++、--运算符实现变量值增加1和减小1的功能。后置的情况是先使用变量的值然后增1或减1；前置的情况是变量先增加1或减小1之后再参与其他运算。



补充2-10：算术运算（续）

```
#include <iostream>
using namespace std;
int main()
{
    int val1=24;
    int val2=5;
    double val3=24;
    double val4=5;
    cout << "int/int, 24/5= " << val1/val2 << endl;
    cout << "int/int, 24%5= " << val1%val2 << endl;
    val2=-5;
    cout << "int/int, 24%(-5)= " << val1%val2 << endl;
    cout << "double/double, 24/5= " << val3/val4 << endl;
```



补充2-10 (续)

```
// cout << "double/double, 24%5= "<<val3%val4<<endl;
cout << "double/int, 24/5= "<<val3/val2<<endl;
cout << "int/double, 24/5= "<<val1/val4<<endl;
val1=5;
cout << "val1 = "<<val1<<endl;
cout <<val1++<< " , ";
cout << ++val1<< " , ";
cout <<val1--<< " , ";
cout << --val1<<endl;
val1=5;
cout << "val1 = "<<val1<<endl;
cout << val1++<< " , "<< ++val1<< " , "<< val1--<< " , "<< --val1<<endl;
// 不同的编译器求值顺序可能不同
return 0;
}
```

补充2-10 (续)

提示与思考：

- ✓ 求余运算 (%) 只能用于整数。
- ✓ 两个整数相除的结果只取整数。
- ✓ 请在不同编译系统中运行本程序，看看最后一句的输出结果是不是我们期望的输出结果：5, 7, 7, 5。

如果不是的话，这是运行顺序问题，按照从右向左的顺序计算各输出项的值，然后按照从左向右的顺序进行输出的话，就可能不是上面的结果。



补充2-11：赋值运算的应用

主要知识点：赋值运算符“=”

- ✓ C++没有赋值语句，将赋值作为一个运算；
- ✓ 赋值运算符将右边操作数的值赋给左边的操作数；
- ✓ 整个赋值表达式的值就是被赋给左边变量的值。



补充2-11：赋值运算的应用（续）

```
#include <iostream>
using namespace std;
int main()
{
    int ival1, ival2;
    double fval;
    char cval;
    ival1=1;
    ival2=2;
    cout<<"ival1="<<ival1<<endl;
    cout<<"ival2="<<ival2<<endl;
    ival1=ival2=0;
    ival1=fval=0;
    ival1=cval='a';
```



补充2-11 (续)

```
cout<<"ival1= "<<ival1<<endl;
cout<<"ival2= "<<ival2<<endl;
cout<<"fval= "<<fval<<endl;
cout<<"cval= "<<cval<<endl;
int ival3=fval=8;
cout <<"ival3= "<<ival3<<endl;
cout <<"fval= "<<fval<<endl;
cout << "ival1 = "<<ival1<< ", ival2 = "<<ival2<<endl;
ival2 = -ival1++ ;
cout <<"ival2 = -ival1++, ival1= "<<ival1<< ", ival2
    = "<<ival2<<endl;
ival1 = ++ival2+ival1;
cout <<"ival1 = ++ival2+ival1, ival1= "<<ival1<<endl;
return 0;
}
```

补充2-11 (续)

运行结果：

ival1= 1

ival2= 2

ival1= 97

ival2= 0

fval= 0

cval= a

ival3= 8

fval= 8

ival1 = 97, ival2 = 0

ival2 = -ival1++, ival1= 98, ival2 = -97

ival1 = ++ival2+ival1, ival1= 2

补充2-11 (续)

提示与思考：

- ✓ 由于“-”，即负运算符的运算优先级高于后置“++”运算符，而且赋值运算符后置运算符“++”的含义是用加1之前的值完成余下的运算，因此 $val2 = -val1++$ ；的运算顺序是：首先将 $val1$ 变量的值的相反数赋给 $val2$ ，然后 $val1$ 实现自增运算。
- ✓ 前置运算符“++”的优先级比“+”和“=”都高，因此，语句 $val1 = ++val2 + val1$ ；首先实现变量 $val2$ 的自增运算，然后实现加法运算，最后才进行赋值。



实验课堂练习2结束



实验课堂练习3!

练习内容3：关系和逻辑运算符、
sizeof()、位运算符



补充2-12：比较数据并输出结果

主要知识点：关系运算、相等运算、逻辑运算

- ✓ 关系运算、相等比较运算、逻辑运算的结果都是bool类型；
- ✓ 关系运算用于比较数据之间的大小关系；
- ✓ 相等比较运算比较两个数据相等与否；
- ✓ 逻辑运算可以将多个关系表达式和相等表达式组合起来，构成复杂的逻辑判断。



补充2-12：比较数据并输出结果（续）

```
#include <iostream>
using namespace std;
int main()
{
    int ival1=1, ival2=2, ival3=3, ival4=4;
    bool nFlag;
    nFlag = ival1==ival2?true:false;
    cout <<"Is 1 equals 2?: "<<nFlag<<endl;
    nFlag = ival1<ival2?true:false;
    cout <<"Is 1less than 2?: "<<nFlag<<endl;
    nFlag = (ival1<ival2 && ival3<ival4)?true:false;
    cout <<"Is 1less than 2 and 3 less than 4: "<<nFlag<<endl;
    return 0;
}
```

//跟踪观察运行情况

补充2-13: *sizeof*运算符的应用

主要知识点: *sizeof*运算

- ✓ 对于C++标准中没有具体规定字节数的数据类型, 以及自定义的复杂数据类型, 如果需要在程序中知道其字节数, 最简单、准确的办法就是使用*sizeof*运算。



补充2-13: *sizeof*运算符的应用 (续)

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    cout << "sizeof(short)= " << sizeof(short) << endl;
    cout << "sizeof(unsigned short)= " << sizeof(unsigned short) << endl;
    cout << "sizeof(int)= " << sizeof(int) << endl;
    cout << "sizeof(unsigned int)= " << sizeof(unsigned int) << endl;
    cout << "sizeof(long)= " << sizeof(long) << endl;
}
```



补充2-13 (续)

```
cout <<"sizeof(unsigned long)= "<<sizeof(unsigned long)<<endl;  
cout <<"sizeof(float)= "<<sizeof(float)<<endl;  
cout <<"sizeof(double)= "<<sizeof(double)<<endl;  
cout <<"sizeof(long double)= "<<sizeof(long double) <<endl;  
cout <<"sizeof(char)= "<<sizeof(char)<<endl;  
return 0;  
}
```



补充2-13 (续)

运行结果：

```
sizeof(short)= 2
sizeof(unsigned short)= 2
sizeof(int)= 4
sizeof(unsigned int)= 4
sizeof(long)= 4
sizeof(unsigned long)= 4
sizeof(float)= 4
sizeof(double)= 8
sizeof(long double)= 8
sizeof(char)= 1
```



补充2-14：位运算符的应用

```
#include <iostream>
#include <bitset>
using namespace std;
int main()
{
    cout << "~15 = " << (~15) << endl;
    cout << "15 & 21 = " << (15 & 21) << endl;
    cout << "15 ^ 21 = " << (15 ^ 21) << endl;
    cout << "15 | 21 = " << (15 | 21) << endl;
    unsigned int nTest = 9;
    cout << "nTest = " << nTest << endl;
    nTest |= 1 << 4;    //将第4位置为1
    cout << "After set the position 4 to 1, nTest = " << nTest << endl;
```



补充2-14 (续)

```
nTest &= ~(1<<4);           //将第4位置0
cout <<"After set the position 4 to 0, nTest = "<<nTest<<endl;
bool nFlag;
for (int i=0; i<16; i++){    //实现翻转
    nFlag = nTest & (1<<i);
    if (nFlag) {
        nTest &= ~(1<<i);
    }
    else{
        nTest |= 1<<i;
    }
}
cout <<"After flip, nTest = "<<nTest<<endl;
}
```



补充2-14 (续)

运行结果：

$\sim 15 = -16$

$15 \& 21 = 5$

$15 \wedge 21 = 26$

$15 | 21 = 31$

nTest = 9

After set the position 4 to 1, nTest = 25

After set the position 4 to 0, nTest = 9

After flip, nTest = 65526



补充2-14 (续)

提示语思考：

- ✓ 将本例中参与运算的数据，及运算结果数据都转换为二进制形式，观察、理解位运算的运算规则和效果。
- ✓ 使用按位与操作可以将操作数中的若干位置0（其他位不变），或者取操作数中的若干指定位。
- ✓ 使用按位或操作可以将操作数中的若干位置1，（其他位不变）。
- ✓ 使用按位异或操作可以将操作数中的若干指定位翻转。



实验课堂练习3结束!



上机练习-任务！

练习内容4：结构化程序设计的控制结构编程



编程上机验证教材以下题目你的结果：

2-22、写出下列表达式的值：

- (1) $2 < 3 \&\& 6 < 9$ (2) $!(4 < 7)$
(3) $!(2 < 3) \|\ (6 < 2)$

2-23、若 $a=1, b=2, c=3$ ，下列各式的结果是什么？

- (1) $a|b-c$ (2) $a^b \& -c$
(3) $a \& b|c$ (4) $a|b \& c$

2-24、若 $a=1$ ，下列各式的结果是什么？

- (1) $!a|a$ (2) $\sim a|a$ (3) a^a (4) $a \gg 2$

对每一题使用debug功能分别单步执行、断点执行步骤2的程序，仔细观察变量值和输出结果的变化。



完成教材以下题目的编程练习：

2-26、编写一个完整的程序，运行时向用户提问“你考试考了多少分？（0~100）”，接收输入后判断其等级并显示出来。规则如下：90≤分数≤100为优；80≤分数<90为良；60≤分数<80为中；0≤分数<60为差。

2-28、用穷举法找出1~100的质数并显示出来。分别用while、do...while、for循环实现。

2-30、声明一个表示时间的结构体，可以精确表示年、月、日、小时、分、秒；提示用户输入年、月、日、小时、分、秒的值，然后完整地显示出来。

2-31、在程序中定义一个整型变量，赋以1~100的值，要求用户猜这个数，比较两个数的大小，把结果提示给用户，直到猜对为止。分别用while、do...while、for循环实现。

对每一题使用debug功能分别单步执行、断点执行步骤2的程序，仔细观察变量值和输出结果的变化。



本讲结束

