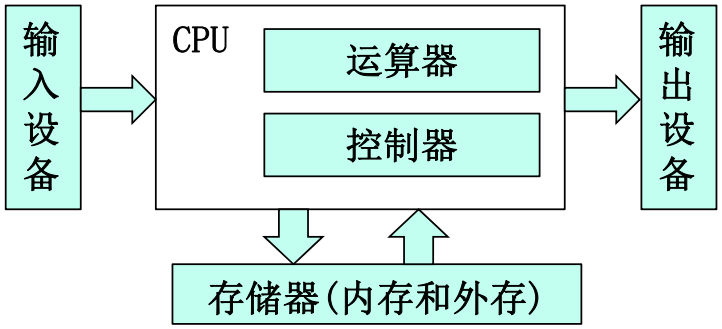
§2. 基础知识

2.1 计算机的组成



§2. 基础知识

2.2. 二进制

2.2.1. 进制的基本概念

★ 进制：按进位原则进行记数的方法叫做进位记数制，简称为进制

十进制　　：0-9，逢十进一

二十四进制：0-23，逢二十四进一

六十进制　：0-59，逢六十进一

=> N进制，0 ~ N-1，逢N进一

★ 基数：指各种位制中允许选用基本数码的个数

十进制　　：0-9　=> 基数为10

二十四进制：0-23 => 基数为24

六十进制　：0-59 => 基数为60

★ 位权：在N进制数中，每个数码所表示的数值等于该数码乘以一个与数码所在位置相关的常数，这个常数叫做位权。
其大小是以基数为底、数码所在位置的序号为指数的整数次幂

$215 = 2*10^2 + 1*10^1 *5*10^0$

2的位权是10^2（百位，基数10为底，序号为2）

1的位权是10^1（十位，基数10为底，序号为1）

5的位权是10^0（个位，基数10为底，序号为0）

§2. 基础知识

2.2. 二进制

2.2.2. 二进制的基本概念

★ 二进制：基数为2，只有0、1两个数码，逢二进一

★ 二进制是计算机内部表示数据的方法，因为计算机就其本身来说是一个电器设备，为了能够快速存储/处理/传递信息，其内部采用了大量电子元件；在这些电子元件中，电路的通和断、电压的高与低，这两种状态最容易实现，也最稳定、也最容易实现对电路本身的控制。我们将计算机所能表示这样的状态，用0、1来表示、即用二进制数表示计算机内部的所有运算和操作

★ 位与字节：每个二进制位只能表示0/1两种状态，当表示较大数据时，所用位数就比较长，为便于管理，每8位称为一个字节

(位：bit 字节：byte)

8 bit = 1 byte

bit ：计算机内表示数据的最小单位

byte ：计算机表示数据的基本单位（数据表示为1-n个byte）

§2. 基础知识

2.2.2. 二进制的基本概念

★ 位与字节：每个二进制位只能表示0/1两种状态，当表示较大数据时，所用位数就比较长，为便于管理，每8位称为一个字节

(位：bit 字节：byte)

8 bit = 1 byte

bit ：计算机内表示数据的最小单位

byte ：计算机表示数据的基本单位（数据表示为1-n个byte）

二进制大数的表示单位：

2^{10} = 1024 =1 KB (KiloByte)
2^{20} = 1024 KB =1 MB (MegaByte)
2^{30} = 1024 MB =1 GB (GigaByte)
2^{40} = 1024 GB =1 TB (TeraByte)
2^{50} = 1024 TB =1 PB (PeraByte)
2^{60} = 1024 PB =1 EB (ExaByte)
2^{70} = 1024 EB =1 ZB (ZetaByte)
2^{80} = 1024 ZB =1 YB (YottaByte)
2^{90} = 1024 YB =1 NB (NonaByte)
2^{100} = 1024 NB =1 DB (DoggaByte)

- 二进制的大数表示及不同单位的换算如表所示

- 简写的时候，b=bit/B=byte

例：某智能手机，存储空间 128GB
某智能电视，存储空间 128Gb

- 实际表述中，K/M/G既可以表示$2^{10}/2^{20}/2^{30}$，也可以是$10^3/10^6/10^9$，因此部分表述有二义性，折算时有误差，要根据语境理解

(计算机内一般按二进制理解，其余十进制)

例：某程序猿工资：18K = 18000
宽带速率：20Mbps = 20 x 10^6 (bit per second)
笔记本内存：8GB = 8 x 2^{30}
硬盘：1TB = 1 x 10^{12} (厂商标注)
= 1 x 2^{40} (计算机理解)
⇨ 约9%误差

§2. 基础知识

2.2. 二进制

2.2.2. 二进制的基本概念

★ 计算机内数据的表示：因为采用二进制，只有两个数码（0/1），任何复杂的数据都是由0/1的基本表示组成的

- 数字(有数值含义的数字序列)
- 文本(包括数码，即无数值含义的数字序列)
- 静态图像
- 动态视频
- 声音

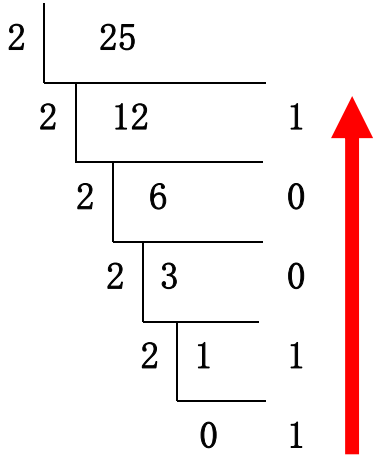
§2. 基础知识

2.2.3. 二进制与十进制的互相转换

★ 数制转换：计算机内部采用二进制，但用计算机解决实际问题时，数据的输入输出通常使用十进制(人易于理解)。
因此，使用计算机进行数据处理时必须先将十进制转换为二进制，处理完成后再将二进制转换为十进制，
这种将数字由一种数制转换成另一种数制的方法称为数制转换

★ 十进制转二进制(整数)：除2取余法

基本方法：
(1) 要转换的整数除以2，得商和余数(整除)
(2) 继续用商除以2，得新的商和余数(整除)
(3) 重复(2)直到商为零时为止
(4) 把所有余数逆序排列，即为转换的二进制数



$(25)_{10} = (11001)_2$

★ 二进制转十进制(整数)：按权相加法

基本方法：
(1) 把二进制数写成加权系数展开式
(2) 按十进制加法规则求和

$$11001 = 1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$$
$$= 16 + 8 + 0 + 0 + 1$$
$$= 25$$

★ 暂不考虑负数(后续讲)

★ 二进制/十进制小数的相互转换(需自学，作业要求)

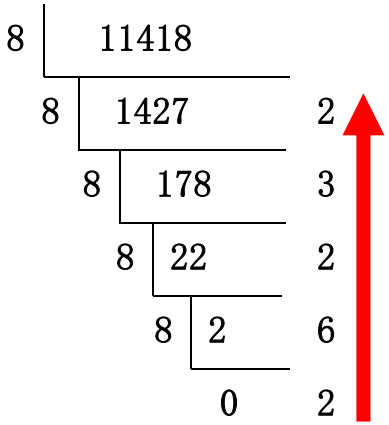
§2. 基础知识

2.2.4. 八进制、十六进制

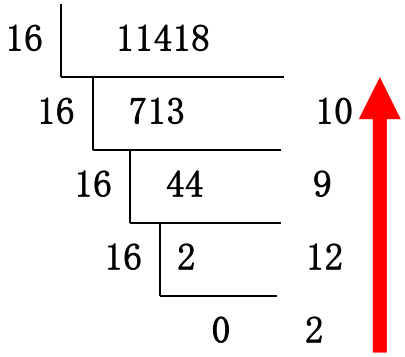
★ 八进制：基数为8，数码为0-7，逢八进一

★ 十六进制：基数为16，数码为0-15，逢十六进一，为避免歧义，用A-F(a-f)替代10-15表示

★ 十进制转八、十六进制：除8/16取余法



$(11418)_{10} = (26232)_8$



$(11418)_{10} = (2C9A)_{16}$

★ 八、十六进制转十进制：按权相加法

$(26232)_8 = 2x8^4 + 6x8^3 + 2x8^2 + 3x8^1 + 2x8^0$

$= 2x4096 + 6x512 + 2x64 + 3x8 + 2x1$

$= 11418$

$(2C9A)_{16} = 2x16^3 + 12x16^2 + 9x16^1 + 10x16^0$

$= 2x4096 + 12x256 + 9x16 + 10x1$

$= 11418$

§2. 基础知识

2.2.4. 八进制、十六进制

★ 二进制转八、十六进制：低位开始，每3/4位转1位

$(1011100101011)_2 = 1\ 011\ 100\ 101\ 011 = (13453)_8$

$(1011100101011)_2 = 1\ 0111\ 0010\ 1011 = (172B)_{16}$

★ 八、十六进制转二进制：每1位转3/4位，不足补0

$(13453)_8$ =001 011 100 101 011=$(1011100101011)_2$

$(172B)_{16}$ =0001 0111 0010 1011=$(1011100101011)_2$

★ 暂不考虑负数(后续讲)

★ 八进制和十六进制的相互转换（需自学，作业要求）

阅读：教材 P.839 附录A 进制系统

§2. 基础知识

2.3. C++程序简介

2.3.1. 最简单的C++程序

例 1

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, World." << endl;
    return 0;
}
```

功能：在屏幕上输出"Hello, World."

例 2

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,sum;
    cin >> a >> b;
    sum=a+b;
    cout << "a+b=" << sum << endl;
    return 0;
}
```

功能：从键盘上输入两个整数(第1行，空格分开)
在屏幕上输出和(第2行)

例 3

```
#include <iostream>
using namespace std;
int max(int x, int y)
{
    int z;
    if (x>y)
        z=x;
    else
        z=y;
    return z;
}
```

功能：从键盘上输入两个整数(第1行)
在屏幕上输出大的那个(第2行)

```
int main()
{
    int a,b,m;
    cin >> a >> b;
    m=max(a,b);
    cout << "max=" << m << '\n';
    return 0;
}
```

§2. 基础知识

2.3. C++程序简介

2.3.2. 程序结构的基本形式

包含的头文件 →

命名空间 →

暂 常量定义

函数的定义

无 全局变量的定义

函数1 →

...

...

函数n

```
#include <iostream>
using namespace std;

int main()
{
    int a,b,sum;
    cin >> a >> b;
    sum=a+b;
    cout<<"a+b="<<sum<< endl;
    return 0;
}
```

例 3

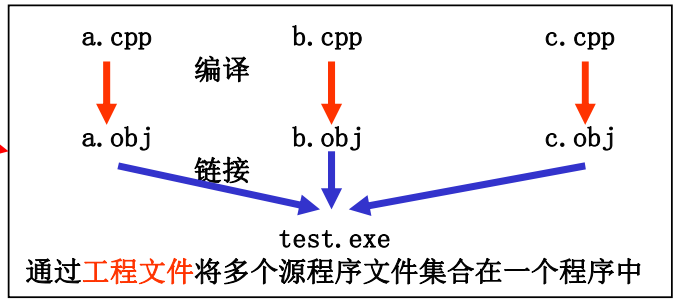
```
#include <iostream>
using namespace std;

int max(int x, int y)
{
    int z;
    if (x>y)
       z=x;
    else
        z=y;
    return (z);
}

int main()
{
    int a,b,m;
    cin >> a >> b;
    m=max(a,b);
    cout<<"max="<<m<<'\n';
    return 0;
}
```

★ C++程序由函数组成，函数是C++程序的基本单位

★ 一个C++程序可由若干源程序文件（*.cpp）组成，每个源程序文件可以包含若干函数

★ 有且仅有一个名为main()的函数，称为主函数，程序的执行从它开始

★ C++提供许多库函数（已做好的）



通过工程文件将多个源程序文件集合在一个程序中

§2. 基础知识

2.3. C++程序简介

2.3.3. 函数的组成

函数返回类型　函数名（形式参数表）

{

　　函数体（局部变量的定义、函数体的可执行部分）

}

```
int max(int x, int y)
{
    int z;
    if (x>y) z=x;
      else z=y;
    return (z);
}
```

★ 从main()开始执行，函数相互间的位置不影响程序的正确性

```
#include <iostream>
using namespace std;
int max(int x, int y)
{ ...
}
int main()
{ ...
}
```

```
#include <iostream>
using namespace std;

//此处需补函数max的声明(暂略)
int main()
{ ...
}
int max(int x, int y)
{ ...
}
```

§2. 基础知识

2.3. C++程序简介

2.3.3. 函数的组成

★ 从main()开始执行，函数相互间的位置不影响程序的正确性

★ 函数平行定义，嵌套调用

★ 函数由语句组成，一个语句以;结尾（必须有），语句分为定义语句和执行语句，定义语句用于声明某些信息，执行语句用于完成特定的操作

★ 书写格式自由，可以一行多个语句，也可以一个语句多行（以\表示分行）

例3

```
#include <iostream>
using namespace std;
int max(int x, int y)
{
    int z;
    if (x>y) z=x;
    else z=y;
    return (z);
}

int main()
{
    int a,b,m;
    cin >> a >> b;
    m=max(a,b);
    cout<<"max="<<m<<'\n';
    return 0;
}
```

定义

调用

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, World." << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{ cout << "Hello, World." << endl; return 0;}
```

```
#include <iostream>
using namespace std;
int main()
{   cout    \
    <<      \
    "Hello, World." \
    << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{   cout
    <<
    "Hello, World."
    << endl;
    return 0;
}
```

同一个程序，这些格式编译器都认为是正确的

§2. 基础知识

2.3. C++程序简介

2.3.3. 函数的组成

★ 书写格式自由，可以一行多个语句，也可以一个语句多行（以\表示分行）

- 不同书写格式，编译器都正确，但对人的阅读而言，友好程度不同
- 建议：一句一行（本课程强制要求：一句一行）

  => 违反格式规定的，本题分数扣20%

✓
`#include <iostream>`
`using namespace std;`
`int main()`
`{`
`  cout << "Hello, World." << endl;`
`  return 0;`
`}`
	✗
	`#include <iostream>`
	`using namespace std;`
	`int main()`
	`{ cout << "Hello, World." << endl;return 0;}`
		`int max(int x, int y)`
		`{`
		`  int z;`
		`  if (x>y)`
		`  z=x;`
		`  else`
		`  z=y;`
		`  return z;`
		`}`
		✓
			`int max(int x, int y)`
			`{`
			`  int z;`
			`  if (x>y) z=x;`
			`  else z=y;`
			`  return z;`
			`}`
			✗

`#include <iostream>`
`using namespace std;`
`int main()`
`{`
`  cout  \`
`  <<  \`
`  "Hello, World." \`
`  << endl;`
`  return 0;`
`}`
✗
	`#include <iostream>`
	`using namespace std;`
	`int main()`
	`{`
	`  cout`
	`  <<`
	`  "Hello, World."`
	`  << endl;`
	`  return 0;`
	`}`
	✗
		`int max(int x, int y)`
		`{`
		`  return 0;`
		`}`
		✓
			`int max(int x, int y) {`
			`  return 0;`
			`}`
			✗

§2. 基础知识

2.3. C++程序简介

2.3.3. 函数的组成

★ 可以用/* … */(多行)或 //…(单行)两种方式加入注释，注释中的内容是为了增加程序的可读性，因此不需要符合C++的语法及规定

```
#include <iostream>
using namespace std;     //命名空间

int main()               /* function main */
{
    cout << "Hello, World." << endl; //Ausgabe(德文-输出)
    return 0;            /* fanhui :-) ^-^ */
}
```

```
#include <iostream>
using namespace std;     //命名空间
/* 下面是主函数，仅用于输出一个字符串，
   输出后程序即运行结束    */

int main()               /* function main */
{
    cout << "Hello, World." << endl;
    return 0;            /* fanhui :-) ^-^ */
}
```

★ 系统提供的库函数和自己编写的函数调用方法相同

§2. 基础知识

2.4. 编译器的安装与使用、编程的基本步骤

2.4.1. VS2019、Dev C++双编译器的安装及配置

另行下发文档

2.4.2. VS2019、Dev C++双编译器的使用(编译C++程序)

另行下发文档

2.4.3. 编程的基本过程

★ 建立新的源程序文件(*.cpp)

★ 对源程序文件(*.cpp)进行编译，检查其中的语法错误，错误分致命错误(error)及警告错误(warning)，生成编译结果文件(*.o/*.obj)

★ 对编译结果文件(*.o/*.obj)进行链接，检查链接错误，形成可执行程序文件(*.exe)

★ 运行可执行程序文件(*.exe)，检查其中的逻辑错误，验证程序的正确性

★ 编译执行过程中会产生很多临时文件（交作业时只有*.cpp是需要的）

2.4.4. 作业命名及格式要求

文档另行下发

★ 仅是初步要求，随着内容深入会逐步提出新要求

★ 认真阅读

§2. 基础知识

2.5. C++的数据类型

2.5.1. 数据类型分类

§2. 基础知识

2.5. C++的数据类型

2.5.2. 各种数据类型所占字节及表示范围(以VS2019 x86/32bit为基准)

[]表示可省略

类型	类型标识符	字节	数值范围	数值范围
整型	[signed] int	4	-2147483648 ～ +2147483647	-2^{31} ～ $+2^{31}-1$
无符号整型	unsigned [int]	4	0 ～ 4294967295	0 ～ $+2^{32}-1$
短整型	short [int]	2	-32768 ～ +32767	-2^{15} ～ $+2^{15}-1$
无符号短整型	unsigned short [int]	2	0 ～ 65535	0 ～ $+2^{16}-1$
长整型	long [int]	4	-2147483648 ～ +2147483647	-2^{31} ～ $+2^{31}-1$
无符号长整型	unsigned long [int]	4	0 ～ 4294967295	0 ～ $+2^{32}-1$
长长整型	long long [int]	8	-9223372036854775808 ～ 9223372036854775807	-2^{63} ～ $+2^{63}-1$
无符号长长整型	unsigned long long [int]	8	0 ～ 18446744073709551616	0 ～ $+2^{64}-1$
字符型	[signed] char	1	-128 ～ +127	-2^{7} ～ $+2^{7}-1$
无符号字符型	unsigned char	1	0 ～ +255	0 ～ $+2^{8}-1$
单精度型	float	4	-3.4×10^{38} ～ 3.4×10^{38}	-3.4×10^{38} ～ 3.4×10^{38}
双精度型	double	8	-1.7×10^{308} ～ 1.7×10^{308}	-1.7×10^{308} ～ 1.7×10^{308}
长双精度型	long double	8	-1.7×10^{308} ～ 1.7×10^{308}	-1.7×10^{308} ～ 1.7×10^{308}

§2. 基础知识

2.5. C++的数据类型

2.5.2. 各种数据类型所占字节及表示范围(以VS2019 x86/32bit为基准)

★ 如何确定数据类型所占的字节：sizeof(类型)

```
#include <iostream>
using namespace std;

int main()
{
    cout << sizeof(int)                 << endl;
    cout << sizeof(unsigned int)        << endl;
    cout << sizeof(long)                << endl;
    cout << sizeof(unsigned short)      << endl;
    cout << sizeof(unsigned long long)  << endl;
    cout << sizeof(float)               << endl;
    cout << sizeof(long double)         << endl;

    return 0;
}
```

用VS2019的32位编译器
64位编译器
DevC++的32位编译器
64位编译器
分别编译运行并观察结果

注意：VS2019和DevC++如何切换32位和64位编译器？

§2. 基础知识

2.5. C++的数据类型

2.5.2. 各种数据类型所占字节及表示范围(以VS2019 x86/32bit为基准)

★ 如何确定数据类型的上下限：

```
#include <iostream>
#include <climits>
using namespace std;

int main()
{
    cout << INT_MIN << endl;
    cout << UINT_MAX << endl;
    cout << LLONG_MAX << endl;
    cout << SHRT_MAX << endl;

    return 0;
}
```

方法1：根据sizeof的结果自行推算
方法2：打印系统预定义的值
(1) 需要包含climits头文件
(2) 预置定义见P.41~42 表3.1

趣味思考题(看完P.24后再想)：
前提条件：
1、不准用climits（或者不知道）
2、知道sizeof()
3、不准用其它工具(书/网络/计算器)
问题：想知道long long型(或其它)数据的最大值，怎么办？

§2. 基础知识

2.5. C++的数据类型

2.5.2. 各种数据类型所占字节及表示范围(以VS2019 x86/32bit为基准)

★ 对于整型数(含char)，均有signed及unsigned的区别，缺省为signed

★ 不同的编译系统中，不同数据类型的所占字节/表示范围可能不同

★ 本课程中若不加以特别说明，均认为：

short ：2字节
int/long ：4字节
long long ：8字节
long double ：8字节

★ 因为数据必须占用一定字节，因此计算机不可能表示数学中的无穷概念

★ 对于整型数，存储为二进制数形式，占满对应字节长度

例：85(十进制) = 1010101(二进制)
则：int型 ：00000000 00000000 00000000 01010101

long型 ：00000000 00000000 00000000 01010101

short型： 00000000 01010101

char型 ： 01010101

★ 浮点型数有有效位数的限定，可能存在一定的误差

§2. 基础知识

2.5. C++的数据类型

2.5.3. 整型数的符号位

引入：以short型数据(2字节)为例

如果是unsigned short，则 00000000 00000000 = 0

11111111 11111111 = $2^{16} - 1 = 65535$

如果是signed short，如何表示正负？

=> 将某类型整型数的若干字节的最高bit位(最左)的0/1理解为正负号，

将该bit位称为符号位

00000000 00000001 = +1

01111111 11111111 = +32767

10000000 00000001 = -1

11111111 11111111 = -32767

=> 将这种表示方式称为二进制的原码表示方式

2.5.4. 补码的基本概念

★ 原码的缺陷：+0与-0的二义性问题

1：1000000000000000

0：0000000000000000

补码：计算机内整型数值的表示方法
- 正数与原码相同
- 负数：绝对值的原码取反+1

例：short型整数

数值	二进制表示	原码	补码
100	1100100	0000000001100100	0000000001100100
-10	1010（绝对值）	0000000000001010	1111111111110101 +) 1 ---------- 1111111111110110
0	0(正) 0(负)	0000000000000000 0000000000000000	0000000000000000 1111111111111111 +) 1 ---------- 1 0000000000000000 （高位溢出，舍去）
-1	1（绝对值）	0000000000000001	1111111111111110 +) 1 ---------- 1111111111111111
-32768	1000000000000000	1000000000000000	0111111111111111 +) 1 ---------- 1000000000000000
-32767	0111111111111111	0111111111111111	1000000000000000 +) 1 ---------- 1000000000000001

思考1：
-32768 + 1 = -32767
-32767 + 1 = -32766
…
-1 +1 = 0
0 + 1 = 1
以上运算符合十进制规则，是否符合二进制规则？

思考2：
现在看到一个以1开始的二进制整数(1***)，到底是unsigned正数还是signed的负数？

§2. 基础知识

2.6. 常量

2.6.1. 基本概念

常量：在程序运行过程中值不能改变的量称为常量

- 字面常量(直接常量)：直接字面形式表示
- 符号常量：通过标识符表示

2.6.2. 数值常量

2.6.2.1. 整型常量

整型常量在C/C++源程序中的四种表示方法：

- 十进制：　正常方式
- 二进制：　0b+0~1
- 八进制：　0+0~7
- 十六进制：　0x/0X+0~9,A-F,a-f

123　　10进制表示 0b1111011　　2进制表示 0173　　8进制表示 0x7B　　16进制表示 四个值相等，都是10进制的123	请把下列三个数从大到小排列： (1) 123 (2) 0123 (3) 0x123　　？

§2. 基础知识

2.6. 常量

2.6.2. 数值常量

2.6.2.1. 整型常量

整型常量在C/C++源程序中的四种表示方法：

- 十进制： 正常方式
- 二进制： 0b+0~1
- 八进制： 0+0~7
- 十六进制： 0x/0X+0~9,A-F,a-f

```
#include <iostream>
using namespace std;
int main()
{
    cout << 123 << endl;
    cout << 0b1111011 << endl;
    cout << 0173 << endl;
    cout << 0x7B << endl;

    return 0;
}
```

输出？

```
#include <iostream>
using namespace std;
int main()
{
    cout << hex << 123 << endl;
    cout << hex << 0b1111011 << endl;
    cout << hex << 0173 << endl;
    cout << hex << 0x7B << endl;

    return 0;
}
```

输出？

```
#include <iostream>
using namespace std;
int main()
{
    cout << dec << 123 << endl;
    cout << "0x" << hex << 0b1111011 << endl;
    cout << "(" << oct << 0173 << ")8" << endl;

    return 0;
}
```

输出？

三个例题的结论：

★ 无论在源程序表示为何种进制，在内存中只有二进制补码一种

★ 无论在源程序表示为何种进制，和输出无关，输出缺省为十进制，可加前导进制转换改为其它进制

★ 输出只负责最基本的内容，其余需要自行按需添加

§2. 基础知识

2.6. 常量

2.6.2. 数值常量

2.6.2.1. 整型常量

整型常量在C/C++源程序中的四种表示方法：

- 十进制： 正常方式
- 二进制： 0b+0~1
- 八进制： 0+0~7
- 十六进制： 0x/0X+0~9,A-F,a-f

★ 无直接的二进制输出方法

★ 老版本编译器无源程序中的二进制表示方法

★ 二进制方式表示不方便，今后不再讨论(以后也可能会说三种表示方法)

★ 整型常量缺省是int型，long型通常加l(L)表示，unsigned通常加u(U)，short型无特殊后缀

下列整型常量分别是什么类型？

123 ：

123L ：

123U ：

123UL：

123LU：

?

另：L不建议小写，为什么？

§2. 基础知识

2.6. 常量

2.6.2. 数值常量

2.6.2.2. 浮点型常量

两种表示方式：

- 十进制数（带小数点的数字）
  - 0.123 123.456 123.0
- 指数形式（科学记数法）
  - e前面为尾数部分，必须有数，e后为指数部分,必须为整数

1.23e4	✓
1.23e-4	✓
-1.23e-4	✓
e4	×
1.23e4.5	×

- 尾数的整数部分为0，第1位小数非零的表示形式称为规范化的指数形式，无论源程序中如何表示，机内存储都是规范化指数形式(思考：好处是什么？)

123e4	(123×10^4)	
12.3e5	(12.3×10^5)	
1.23e6	(1.23×10^6)	
0.123e7	(0.123×10^7)	机内存储形式
0.0123e8	(0.0123×10^8)	

§2. 基础知识

2.6. 常量

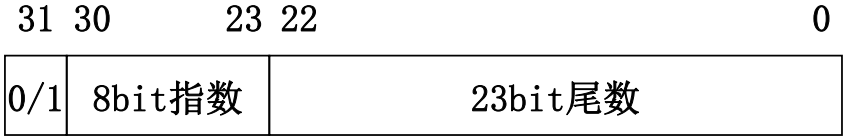
2.6.2. 数值常量

2.6.2.2. 浮点型常量

两种表示方式：

- 十进制数（带小数点的数字）
- 指数形式（科学记数法）

★ 浮点数在内存中的存储分为三部分，分别是符号位、指数部分和尾数部分



浮点数的存储遵从 IEEE 754 规范
具体暂时不做要求，等学会读内存数据后再说[后续课程]

★ 浮点数有指定有效位数(float:6位/double:15位)，超出有效位数则舍去(四舍五入)，因此会产生误差

例1：常量1：123456.7890123456**7**e5　　常量2：123456.7890123456**8**e5

内部存储都是 0.1234567890123457e11

例2：1.0/3*3　　不同编译系统，可能是0.999999或1

★ 浮点常量缺省为double型，如需表示为float型，可以在后面加f(F)

1.23　：double型，占8个字节

1.23F ：float型，占4个字节

可自行写测试程序来证明

```
cout << sizeof(1.23)  << endl;
cout << sizeof(1.23F) << endl;
```

§2. 基础知识

2.6. 常量

2.6.3. 字符常量与字符串常量

2.6.3.1. ASCII码

★ P.845 附录C ASCII码表

★ ASCII码占用一个字节，共可表示256个字符

0XXXXXXX ： 基本ASCII码 128个(0-127)

1xxxxxxx ： 扩展ASCII码 128个(128-255)

★ 基本ASCII码分图形字符和控制字符

0-32，127：控制字符，34个

33-126 ：图形字符，94个(键盘上都能找到)

★ 几个基本的ASCII码值

0 - 48/0x30 空格 - 32

A - 65/0x41 a - 97/0x61

★ 汉字的表示：

GB2312-80 ：用两个字节表示一个汉字，共6733个，两字节的高位为1(与扩展ASCII码冲突)

GBK ：1995年公布，2字节表示，收录汉字20000+

GB18030-2005：2-4个字节表示，收录汉字70000+

§2. 基础知识

2.6. 常量

2.6.3. 字符常量与字符串常量

2.6.3.2. 普通字符常量与转义字符常量

★ 直接表示　　'空格或大部分可见的图形字符'

★ 转义符表示　'\字符、八、十六进制数'

● '\字符'：P.50 表3.2

在表格后加：'\ddd'：000-377 （0-255）：8进制值对应的ASCII码

'\xhh'：00-ff/FF（0-255）：16进制值对应的ASCII码

注意：\x的x必须小写，否则warning('\X41'错)，后面字符大小写不限('\x1A'/'x1a'均可)

相似概念：整型常量的16进制，大小写均可0x41 / 0X41

```
#include <iostream>
using namespace std;
int main()
{
    cout << '\377' << endl;
    cout << '\477' << endl;
    cout << '\x41' << endl;
    cout << '\X41' << endl;
    return 0;
}
```

VS2019/DevC++：分别提示什么信息？

体会编译器的差异

§2. 基础知识

2.6. 常量

2.6.3. 字符常量与字符串常量

2.6.3.2. 普通字符常量与转义字符常量

★ 一个字符常量可有几种表示形式

A（ASCII=65）	'A' '\101' '\x41'（'\X41'错!!!）	ESC(ASCII=27)	'\33' '\033' '\x1b' '\x1B'
换行(ASCII=10)	'\n' '\12' '\012' '\xA' '\x0A' '\xa' '\x0a'	双引号(ASCII=34)	'\"' '\42' '\042' '\x22'

★ '0'与'\0'的区别（非常重要!!!）

'0' - ASCII 48 '\60' '\060' '\x30'

'\0' - ASCII 0 '\00' '\000' '\x0' '\x00'

★ 控制字符中，除空格外，都不能直接表示，\ ' " 等特殊图形字符也不能直接表示

2.6.3.3. 字符在内存中的存储

★ 一个字符常量只能表示一个字符

★ 一个字符在内存中占用一个字节

★ 字节的值为该字符的ASCII码

§2. 基础知识

2.6. 常量

2.6.3. 字符常量与字符串常量

2.6.3.4. 字符串常量

含义：连续多个字符组成的字符序列

表示："字符串"

★ 可以是图形字符，也可以转义符

字符串的长度：字符序列中字符的个数

"abc123*#" = ?

"\x61\x62\x63\061\62\063\x2a\043" = ?

"\r\n\t\\A\\t\x1b\"\1234\xft\x2f\33" = ?

strlen是系统函数，打印字符串的长度

```
#include <iostream>
using namespace std;

int main()
{
    cout << strlen("abc123*#") << endl;
    cout << strlen("\x61\x62\x63\061\62\063\x2a\043") << endl;
    cout << strlen("\r\n\t\\A\\t\x1b\"\1234\xft\x2f\33") << endl;

    return 0;
}
```

§2. 基础知识

2.6. 常量

2.6.3. 字符常量与字符串常量

2.6.3.4. 字符串常量

含义：连续多个字符组成的字符序列

表示：″字符串″

★ 可以是图形字符，也可以转义符

字符串的长度：字符序列中字符的个数

″abc123*#″ = ?

″\x61\x62\x63\061\62\063\x2a\043″ = ?

″\r\n\t\\A\\t\x1b\″\1234\xft\x2f\33″ = ?

″\r\n\t\\A\\t\x1b\″\9234\xft\x2f\33″ = ?

思考：（课上未讲，也不再深入讨论，可自行查找资料）

C/C++在编译8/16进制转义符时有区别：

1、转义符后的合法8/16进制数若多于3/2个

″\1234″ ：8进制，编译不报错，长度为2

″\x2fa″ ：16进制，编译报error错

2、转义符后跟非法字符

″\9234″ ：8进制，编译报warning错，长度为4

″*123″ ：同上

″\xg123″ ：16进制，编译报error错

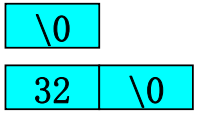
″\x*123″ ：同上

在内存中的存放：每个字符的ASCII码+字符串结束标志 ’\0’ （ASCII 0、尾0）

★ ″″与″ ″的区别

″″ - 空字符串，长度为0

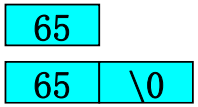
″ ″ - 含一个空格的字符串，长度为1



★ ’A’与″A″的区别

’A’ - 字符常量，内存中占一个字节

″A″ - 字符串常量，内存中占两个字节



★ 暂不讨论字符串中含尾0的情况（后续模块再讨论）

″Hello\0ABC″

可写测试程序，观察下面的值

```
sizeof("");  strlen("");
sizeof(" "); strlen(" ");
sizeof('A'); strlen('A');
sizeof("A"); strlen("A");
```

问题1：上述8个中哪个错？

2：sizeof和strlen差异？

§2. 基础知识

2.6. 常量

2.6.4. 符号常量

用一个标识符代表的常量称为符号常量

```
#define   pi   3.14159
```

★ 优点：含义清晰，修改方便

```
#include <iostream>
using namespace std;

int main()
{   ...;
    3.14159 * ...;
    ...;
    3.14159 * ...;
    ...;
    3.14159 * ...;
    ...;
    return 0;
}
```

```
#include <iostream>
using namespace std;
#define pi 3.14159
int main()
{   ...;
    pi * ...;
    ...;
    pi * ...;
    ...;
    pi * ...;
    ...;
    return 0;
}
```

假设共10000处使用π值

1、要求降低π的精度为3.14

2、要求提高π的精度为3.1415926

那种方法方便？易于修改？

在VS2019下编译运行下面的代码，观察结果

```
#define 喵喵喵 main
#define 喵喵 int
#define 喵 (
#define 喵喵呜 )
#define 喵喵喵喵 {
#define 喵呜喵 }
#define 呜喵喵 <<
#define 呜 cout
#define 呜呜 endl
#define 呜呜呜 "喵喵喵!"
#define 呜呜呜呜 return
#define 呜喵 0
#define 喵呜 ;
#include <iostream>
using namespace std;
喵喵 喵喵喵 喵 喵喵呜
喵喵喵喵
呜 呜喵喵 呜呜呜 呜喵喵 呜呜
喵呜
呜呜呜呜 呜喵 喵呜
喵呜喵
```