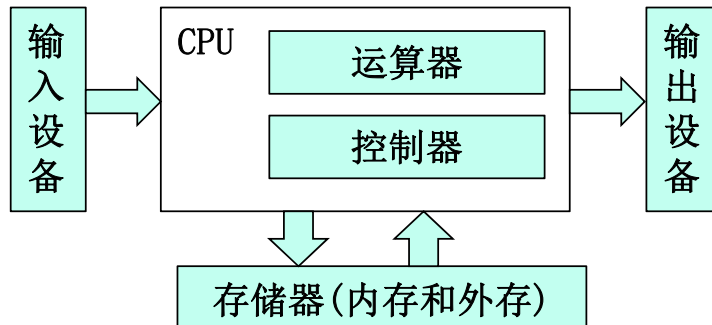




§ 2. 基础知识

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, $0 \sim N-1$, 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

$$215 = 2 \times 10^2 + 1 \times 10^1 + 5 \times 10^0$$

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

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	(ZettaByte)
2^{80}	= 1024 ZB	=1 YB	(YottaByte)
2^{90}	= 1024 YB	=1 BB	(BrontoByte)
2^{100}	= 1024 BB	=1 NB	(NonaByte)
2^{110}	= 1024 NB	=1 DB	(DoggaByte)
2^{120}	= 1024 DB	=1 CB	(CorydonByte)

● 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 = 20×10^6 (bit per second)

笔记本内存: 8GB = 8×2^{30}

硬盘: 1TB = 1×10^{12} (厂商标注)

= 1×2^{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

课上会有简单的介绍，了解即可；
听不懂也不影响后续学习



§ 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

★ 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

$$\begin{aligned} 11001 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 16 + 8 + 0 + 0 + 1 \\ &= 25 \end{aligned}$$

★ 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		25	
2		12	1
2		6	0
2		3	0
2		1	1
		0	1

(25)₁₀ = (11001)₂



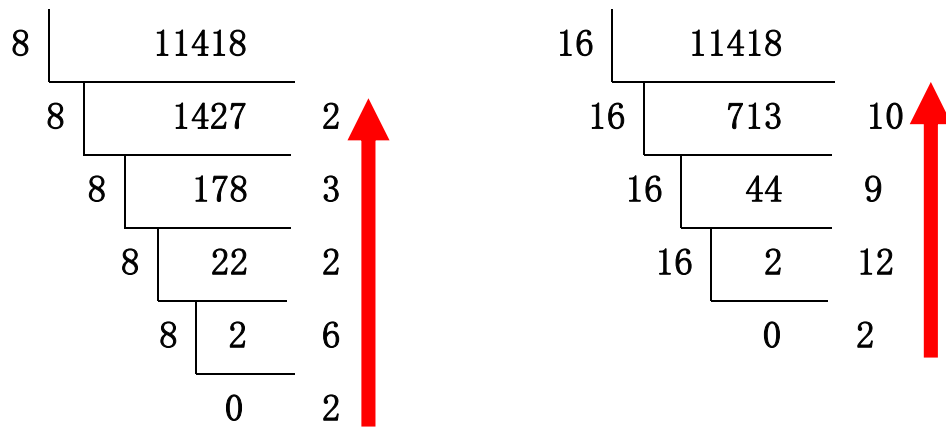
§ 2. 基础知识

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

$$\begin{aligned}(26232)_8 &= 2 \times 8^4 + 6 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 \\ &= 2 \times 4096 + 6 \times 512 + 2 \times 64 + 3 \times 8 + 2 \times 1 \\ &= 11418\end{aligned}$$

$$\begin{aligned}(2C9A)_{16} &= 2 \times 16^3 + 12 \times 16^2 + 9 \times 16^1 + 10 \times 16^0 \\ &= 2 \times 4096 + 12 \times 256 + 9 \times 16 + 10 \times 1 \\ &= 11418\end{aligned}$$



§ 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：下发的附录（A-计数系统）



§ 2. 基础知识

2.3. C++程序简介

2.3.1. 最简单的C++程序

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

例1

功能：在屏幕上输出“Hello, World.”

```
#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2

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)

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

例3

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 = my_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;
}
```

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

例 3

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

函数my_max

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

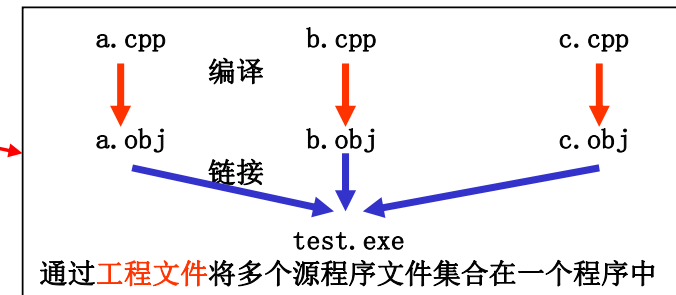
函数main

★ 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)





§ 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 my_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 my_max(int x, int y)  
{ ...  
}  
int main()  
{ ...  
}
```

```
#include <iostream>  
using namespace std;  
  
//此处需补函数my_max的声明(暂略)  
int main()  
{ ...  
}  
int my_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 my_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 = my_max(a,b);
    cout<<"max="<<m<<' \n' ;
    return 0;
}
```

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

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%

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



§ 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

<pre>#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 :-) ^-^ */ }</pre>	<pre>#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 :-) ^-^ */ }</pre>
--	---

- ★ 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. VS2022、Dev C++双编译器的安装及配置

另行下发文档

2. 4. 2. VS2022、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VS2022 x86/32bit为基准)

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

[]表示可省略
例: signed int } 等价
int

long } 等价
long int }



§ 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VS2022 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VS2022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: VS2022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VS2022 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

- 可能需要包含climits头文件
(目前所用双编译器均不需要)
- 预置定义见P. 36 表3.1

趣味思考题(学到P. 24后再想):

前提条件:

- 1、不准用INT_MIN等预置值(或不知道)
- 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VS2022 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	绝对值的二进制表示	原码	补码
100	1100100	0000000001100100	0000000001100100
-10	1010 (绝对值)	0000000000001010	111111111110101 +) 1 ----- 111111111110110
0	0(正) 0(负)	0000000000000000 0000000000000000	0000000000000000 111111111111111 +) 1 ----- 1 0000000000000000 (高位溢出, 舍去)
-1	1 (绝对值)	0000000000000001	1111111111111110 +) 1 ----- 111111111111111
-32768	1000000000000000 (1表示2 ¹⁵)	1000000000000000	011111111111111 +) 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	请把下列三个数从大到小排列： (1) 123 (2) 0123 (3) 0x123 ?
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		



§ 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:

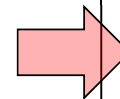
123LU:

?

另：L不建议小写，为什么？

问题：这几种在vs下都是4字节，
如何证明？

答：C++提供typeid运算符，
基本用法为typeid(数据).name



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

int main()
{
    cout << typeid(123).name() << endl;
    cout << typeid(123L).name() << endl;
    cout << typeid(123U).name() << endl;
    cout << typeid(123UL).name() << endl;
    cout << typeid(123LU).name() << endl;

    return 0;
}
```



§ 2. 基础知识

2.6. 常量

2.6.2. 数值常量

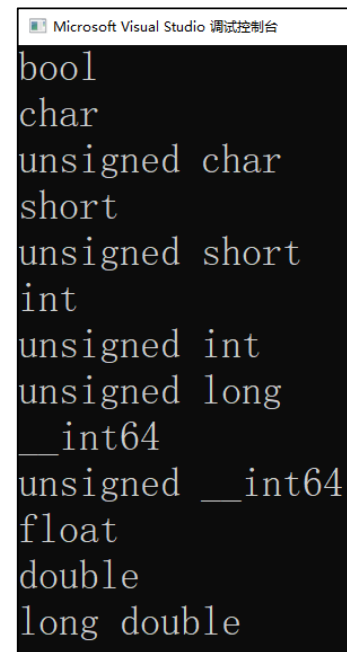
2.6.2.1. 整型常量

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

int main()
{
    cout << typeid(bool).name() << endl;
    cout << typeid(char).name() << endl;
    cout << typeid(unsigned char).name() << endl;
    cout << typeid(signed short).name() << endl;
    cout << typeid(unsigned short).name() << endl;
    cout << typeid(int).name() << endl;
    cout << typeid(unsigned).name() << endl;
    cout << typeid(unsigned long).name() << endl;
    cout << typeid(long long).name() << endl;
    cout << typeid(unsigned long long).name() << endl;
    cout << typeid(float).name() << endl;
    cout << typeid(double).name() << endl;
    cout << typeid(long double).name() << endl;
    return 0;
}
```

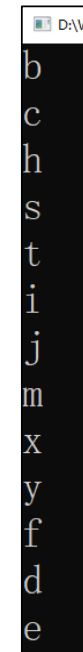
VS打印完整信息，Dev打印
单字母，后续不再解释Dev，
有兴趣自行研究即可

VS



```
Microsoft Visual Studio 调试控制台
bool
char
unsigned char
short
unsigned short
int
unsigned int
unsigned long
__int64
unsigned __int64
float
double
long double
```

Dev



```
D:\W4
b
c
h
s
t
i
j
m
x
y
f
d
e
```