变量和基本类型

COMP250205：计算机程序设计

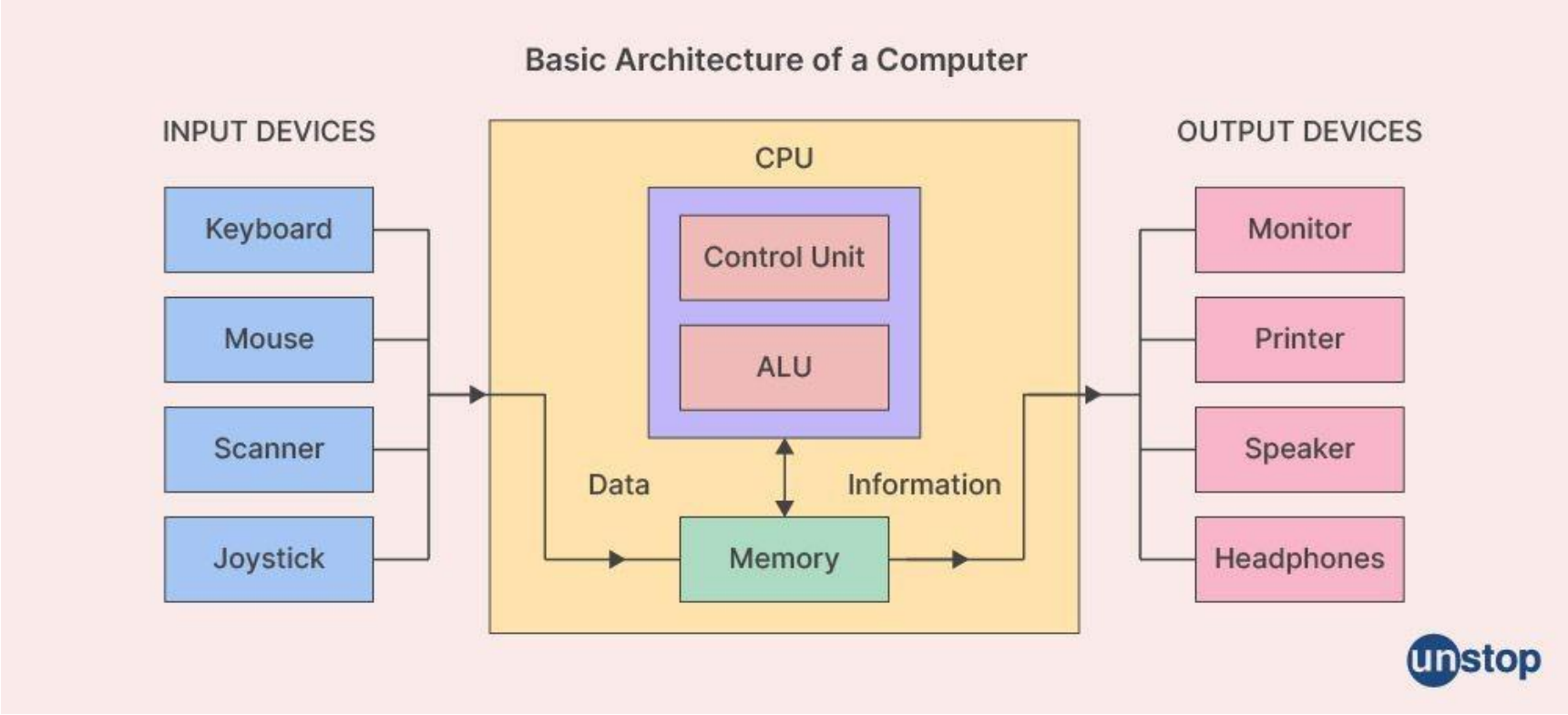
李昊

hao.li@xjtu.edu.cn

西安交通大学计算机学院

数据与数据类型

冯诺依曼体系架构


Basic Architecture of a Computer
INPUT DEVICES
Keyboard
Mouse
Scanner
Joystick
CPU
Control Unit
ALU
Data
Information
Memory
OUTPUT DEVICES
Monitor
Printer
Speaker
Headphones
unstop

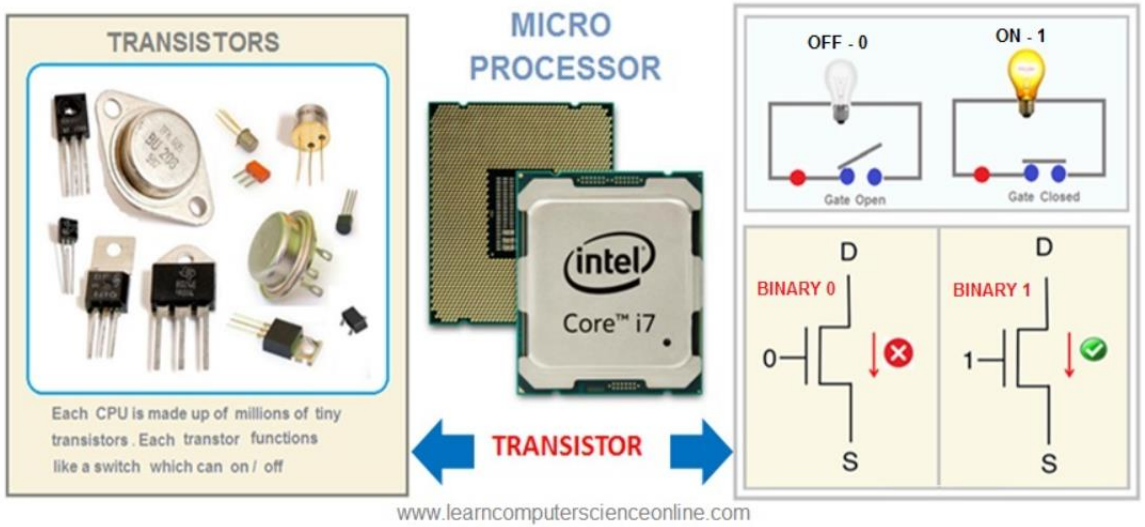
记数法：X进制

进制	元素	例子
10进制（decimal）	{0,⋯,9}	1+9=10
2进制（binary）	{0,⋯,1}	1+1=10
8进制（octal）	{0,⋯,7}	7+1=10
16进制（hexadecimal）	{0,⋯9,A,B,C,D,E}	F+1=10

二进制

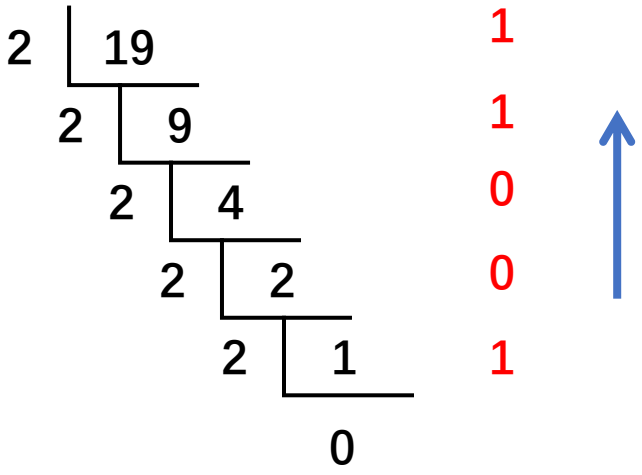
二进制：天然适配计算机结构的计数法

世界上只有10种人，1种人懂二进制，1种人不懂



二进制与十进制的转换

十进制 → 二进制：带余数除法



二进制 → 十进制：逐位相乘

$$(10011)_2 = 1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0 = (19)_{10}$$

数据的单位

单位	换算关系	读法
Bit (b)	最小单位	比特t
Byte (B)	1 B = 8 b	字节
KB	1 KB = 1024 B	千
MB	1 MB = 1024 KB	兆
GB	1 GB = 1024 MB	吉
TB	1 TB = 1024 GB	太
PB	1 PB = 1024 TB	拍
EB	1 EB = 1024 PB	艾

Bit和Byte

Bit只能表示“0”或“1”，是计算机数据的最小单位

Byte（字节）由8个Bit组成

- 数字范围 0 - 255
- 大多数高级语言中数据类型的最小单位
- 存储一个字符（英文字母、0-255的数字、简单符号）

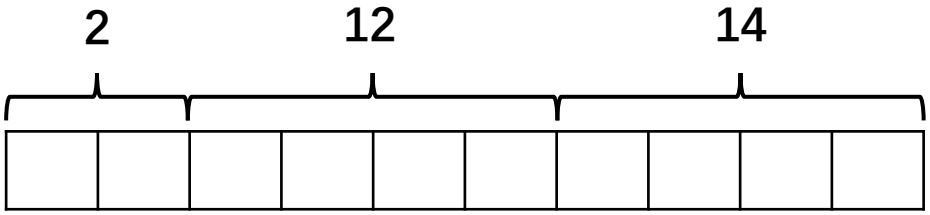
ASCII码

Control Characters				Graphic Symbols											
Name	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	'	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	−	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

数据的存储

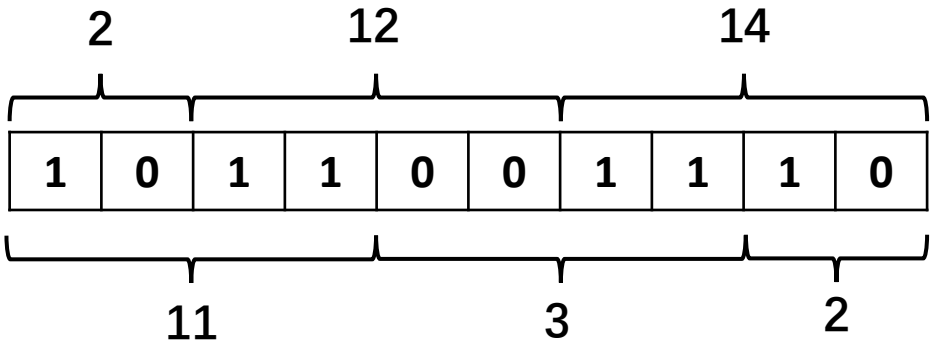
存储器：可简单抽象为一片无限大的、均匀的线性空间

在计算机中，所有数据均以二进制方式公平的存储

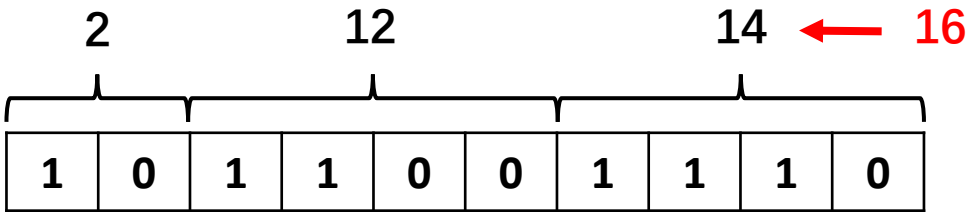


线性空间存储的问题

线性均一空间的数据难以理解

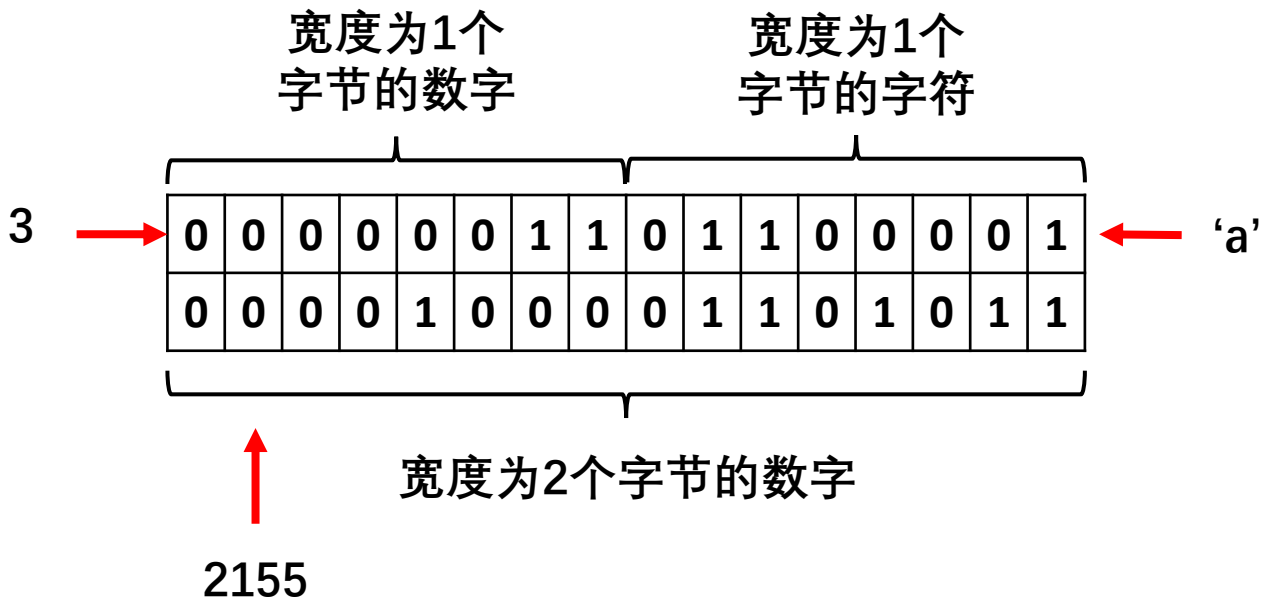


线性均一空间的数据难以调整



数据类型

数据类型：决定了数据存储的宽度，和允许的操作



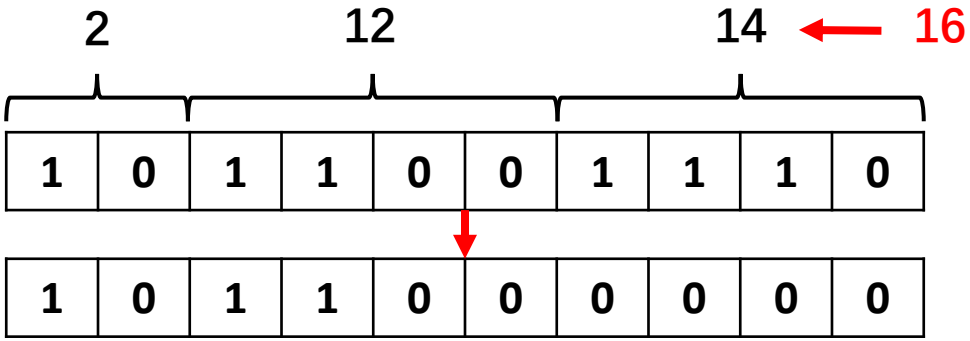
按照数据类型访问数据

访问数据：数据的地址（存储起点）+ 数据类型（操作的宽度）

按照数据类型存储

一种数据类型总占据固定的宽度

如果数据的修改超出了固定的宽度 → 溢出



按照数据类型访问数据

使用者需要对采用何种数据类型负责

C++中的数据

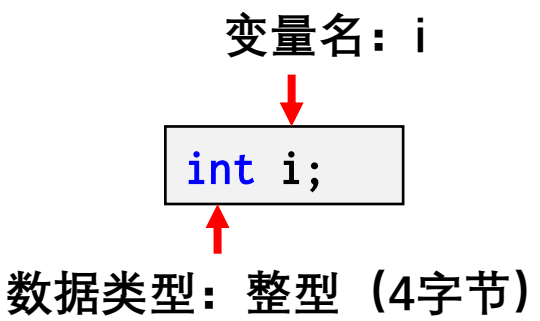
变量

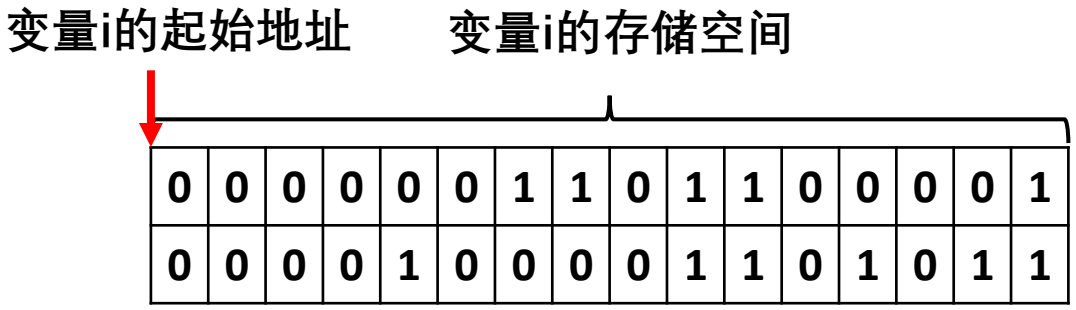
变量：值可以被改变的量

指明其数据类型（程序员负责）

对变量命名（程序员负责）

预留对应宽度的存储空间，关联到变量名、数据类型上（计算机自动处理）





变量命名

命名规则：只能由字母、数字和下划线组成

- 第一个字符必须是字母或者下划线
- 合法命名：a，a1，a_1，_a
- 非法命名：a+b，1a，(a)

命名区分大小写

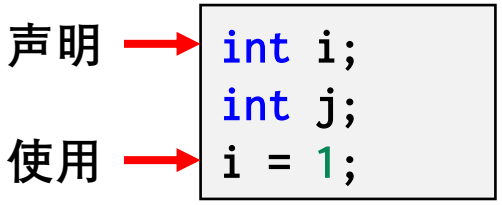
- Sum，sum，SUM代表不同变量

变量的使用 - 1

先声明，后使用

声明：确定变量名，变量类型，预留空间

使用：读取、修改变量的值



这时j的值是什么?

变量的使用 - 2

初始化：声明式同时给定初值

赋值：以新值覆盖原始值；初始化：给定初值

使用赋值操作初始化 →
```
int a = 1;
```
使用列表初始化 →
```
int a = {1};
```

拷贝初始化
在构建对象过程中将值复制进去

初窥对象：不同的初始化概念

直接初始化
实际上调用了对象的构造函数 →
```
int a(1);
int a{1};
```

显式类型转换
实际上进行了对象类型转换 →
```
int a = int(1);
int a = int{1};
```

变量的作用域（scope）

每个名字（变量、函数、类型）都指向特定实体

相同名字也可能指向不同实体：不同作用域

大部分作用域以花括号分割

在main函数中可以访问

只能在循环体中访问

```
int main()
{
    int sum = 0, val = 1;
    while (val <= 10) {
        int step = 1;
        sum = sum + val;
        val = val + step;
    }
    cout << sum << endl;
    return 0;
}
```

变量的作用域 - QUIZ

```
int i = 42;
int main()
{
    int i = 100;
    int j = i;
}
```

for循环 →

```
int i = 100, sum = 0;
for (int i = 0; i != 10; i = i+1) {
    sum += i;
}
cout << i << " " << sum << endl;
```

常量

常量：值不能被修改的固定值

立即数：1，1.5， 0x01

字符：‘a’，‘\n’

字符串：“hello”，“xjtu”

布尔值：true，false

符号常量：const int a = 1;

立即数常量

各种进制的整数

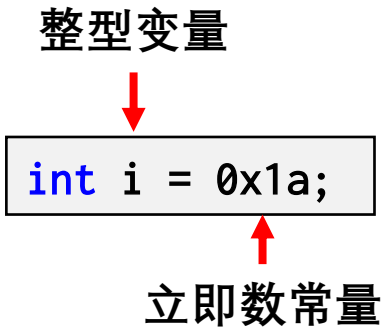
十进制整数：123

二进制整数（0b开头）：0b0101

八进制整数（0开头）：011

十六进制整数（0x开头）：0x1a

浮点数常量：1.0，0.123，123E-3

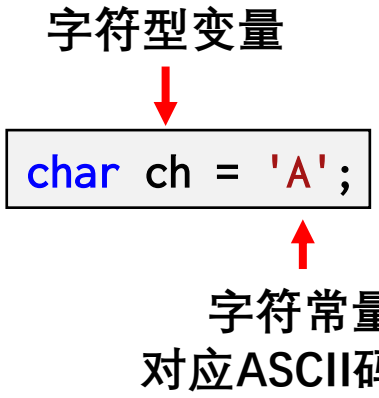


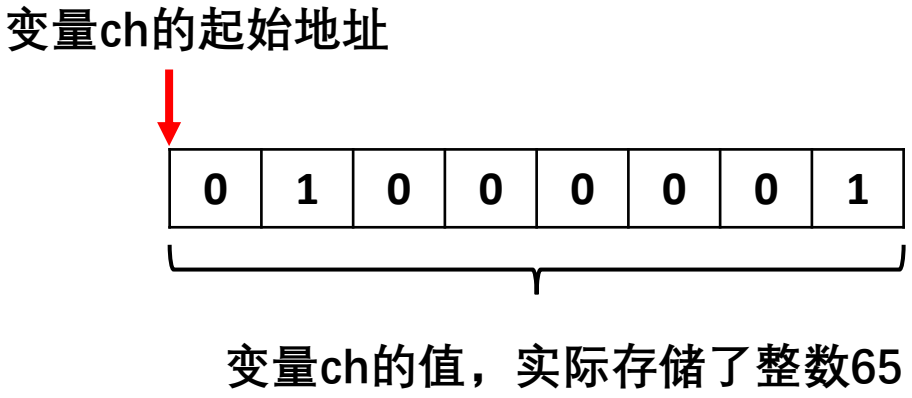
将常量赋给了变量（复制）
更改的是变量

字符常量

单引号括起来的单个字符

字符常量实际上是一个宽度1字节的整数，对应所属字符集的编码值（ASCII码）





转义字符常量

以反斜杠开头，表达一些无法直接写出的字符

转义字符	含义
\n	回车换行
\t	横向制表符，tab
\\	反斜杠'\'
\'	单引号
\"	双引号

符号常量 - 1

字面量没有名字，无法被重复引用

在大型代码管理中带来不便

需求更改
打印前200个数字

也得跟着改

```
// first 100 numbers
int i = 0;
while (i < 100) {
    cout << i << endl;
    i = i + 1;
}
// 1000 lines later
int j = 0;
while (j < 100) {
    cout << j << endl;
    j = j + 1;
}
```

符号常量 - 2

使用const修饰普通变量，使其成为符号常量

整型符号常量
名为number_limit

引用符号常量

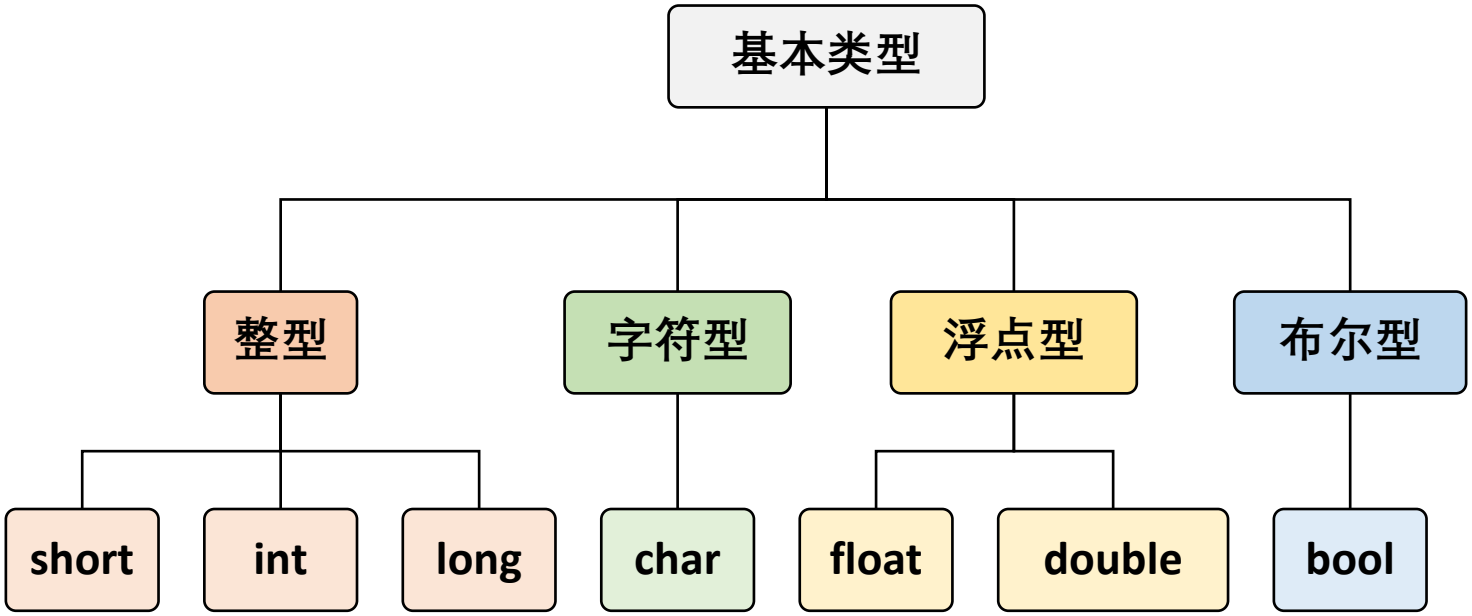
```
// first 100 numbers
const int number_limit = 100;
int i = 0;
while (i < number_limit) {
    cout << i << endl;
    i = i + 1;
}
// 1000 lines later
int j = 0;
while (j < number_limit) {
    cout << j << endl;
    j = j + 1;
}
```

number_limit = 300;

C++中的数据类型

C++内置基本数据类型

可以直接使用的数据类型



基本数据类型的宽度

数据类型	含义	宽度（字节）
short	短整型	2
int	整型	4
long	长整型	8
float	单精度浮点数	4
double	双精度浮点数	8
char	字符	1
bool	布尔值（true，false）	1

以上宽度基于x86-64体系结构

在不同体系结构的计算机中，数据类型的宽度可能会有不同

基本数据类型的宽度

看看自己机器的基本数据类型宽度

C++提供了sizeof操作符，查看数据类型的宽度

C++中提供了一些“宏”，代表数据类型值上限和下限

```cpp
#include <iostream>
#include <climits>
int main()
{
    using namespace std;
    int n_int = INT_MAX;     // max int value
    short n_short = SHRT_MAX; // max short value
    long n_long = LONG_MAX;

    cout << "short is " << sizeof(short) << " bytes." << endl;
    cout << "int is " << sizeof(int) << " bytes." << endl;
    cout << "long is " << sizeof(long) << " bytes." << endl;

    cout << "Max values:" << endl;
    cout << "short: " << n_short << endl;
    cout << "int: " << n_int << endl;
    cout << "long: " << n_long << endl;

    return 0;
}
```

包含各种数字极限的宏

选择合适的数据类型

根据需要处理的数据大小决定，以整数为例

- 根据大小选择short，int和long
- 非常小的数字，可以使用char
- 仅代表两种状态（开关量），使用bool

考虑无符号和有符号整数

- 默认为有符号整数，如short表达 -32768 到 32767
- 声明为无符号整数，如unsigned short表达 0 到 65535

```
unsigned int a;
```

浮点型变量的精度陷阱 - 1

小数也使用二进制存储

$(0.1101)_2 = 1/2 + 1/4 + 1/16 = (0.8125)_{10}$

使用二进制有可能无法准确表达十进制小数

如何使用二进制表达$(0.1)_{10}$?

```
float f = 123.456;

if (f == 123.456)  {
    cout << "Yes, f equals to 123.456" << endl;
}
else {
    cout << "Where's my math teacher?" << endl;
}
```

浮点型变量的精度陷阱 - 2

```
#include <iostream>
int main()
{
    using namespace std;
    cout.setf(ios_base::fixed, ios_base::floatfield);
    float tub = 10.0 / 3.0;
    double mint = 10.0 / 3.0;
    float million = 1.0e6;

    cout << "tub = " << tub;
    cout << ", a millon tubs = " << million * tub;
    cout << ",\nand ten million tubs = " << 10 * million * tub << endl;

    cout << "mint = " << mint << " and a million mints = ";
    cout << million * mint << endl;

    return 0;
}
```

固定输出精度（定点数）

单精度浮点变量，小数点6位左右精确

双精度浮点变量，小数点15位左右精确

类型转换

数据公平的以二进制存储

以不同的数据类型（强行）重新理解这个数据

隐式类型转换：在某些数据类型之间自动完成

显式类型转换：由程序员手动指定新的数据类型

```
int b = 9;
double a = b;
```

隐式类型转换
将9以浮点数形式存在a中，9.0

```
char ch = 'A';
cout << ch;
cout << int(ch);
```

显式类型转换
以整数方式输出ch

隐式类型转换

相同类型的数据操作，结果还是同一个类型

不同类型的数据操作，结果是范围大的那个类型

取值范围小转为取值范围大的类型是安全的

char -> short -> int -> long -> float -> double

但隐式类型转换并不保证这一点

```
double b = 9.8;
int a = b;
```

隐式类型转换
将9.8截断为整数并存储到a中

显式类型转换

强制按照某种类型进行数据解读

灵活、强大但是很危险

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

int main() {
    unsigned int largeNumber = 300;
    char smallNumber = (char)largeNumber;

    cout << "Large Number: " << largeNumber << endl;
    cout << "Small Number: " << (int)smallNumber << endl;

    return 0;
}
```

显式类型转换
300被截断存储到一个字节

显式类型转换
以int方式打印smallNumber

类型转换 - QUIZ

整型与整型计算，结果是整型

将整型结果转换为浮点型

浮点型与整型计算，结果是浮点型

```
#include <iostream>

int main()
{
    using namespace std;
    int f = 5;
    cout << f << endl;
    cout << f / 2 << endl;

    cout << float(f/2) << endl;

    cout << float(f) / 2 << endl;

    return 0;
}
```

内容总结

数据与数据类型

数据：二进制存储均一存储

数据类型：决定数据的宽度和操作方式

C++数据和数据类型

变量与常量：命名规则，常量表达形式

数据宽度问题：整型溢出，浮点类型精度问题

数据类型转换：隐式，显式