

C-Learning Note

This is a note of learning C by watching Li Ge's course

JINNA

February 2017

Contents

I love you I love you I love you I love you I love you I love you

Chapter 1

c 语言中的数据成分

1.1 变量定义的含义

1. 将内存想象成火车道。每个存储单元有 8 位 (b/bit), 1B(Byte) = 8b(bit), 每 1b 存放 1 个二进制数字, 每个 Byte 都对应一个地址。2. 变量: 值可以变化的量。变量的定义包含两个部分: 变量类型变量标识符; (最好定义时就赋初始值; C 语言中, 所有的变量必须先定义再使用) 定义

```
( 变量类型 ) ( 变量标识符 );  
int      Max;  
int      Max = 0;  
char     character;  
char     character = 'A';  
double   Result = 12.345;
```

图 1: 变量定义格式

变量起到的作用: 在内存中找到一片存储空间, 将其命名为变量名, 将变量数字存储到存储空间中, 并记录存储空间的初始地址。

3. C 中的数据类型:

1.2 整数型

1.2.1 类别

按内存空间的大小区分。(C 语言要求 long 型不短于 int 型, short 型不长于 int 型, 此图表示 visual c++ 中数据的定义, 其他编译器 int 型的字节长度会改变。)

Signed 可以不写。

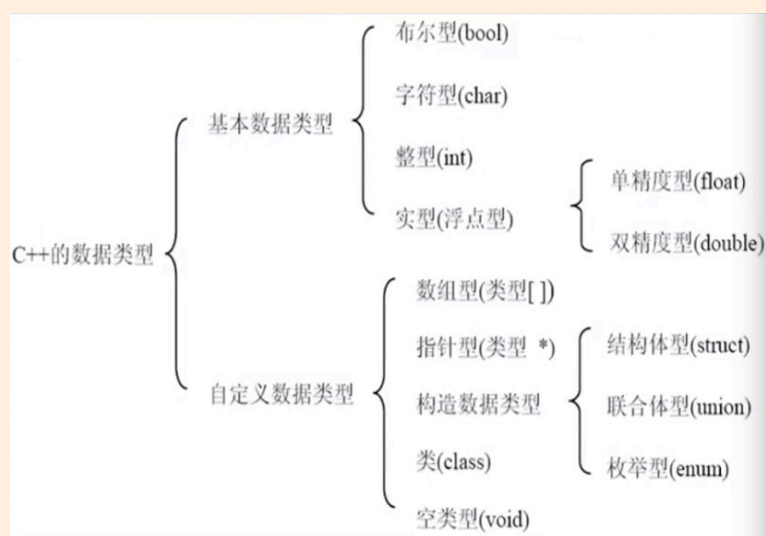


图 2: C 语言中的数据类型

| | 整型 | 内存空间 |
|-----|--------------------|-------|
| 基本型 | <u>int</u> | 32bit |
| 短整型 | short short int | 16bit |
| 长整型 | long long int | 32bit |

图 3: 整型数据的分类

1.2.2 存储

第一位不表示具体数据，只作为符号位，表示数据的正负：1 表示负，0 表示正。正整数和负整数二进制表示方法不同。计算机中存储的是补码，正整数的补码等于其原码。

1.2.3 输入输出

想知道 2 进制表示，直接用 cout 的功能打印出 16 进制表示就可以。

注意：如果不修改，则会一直按该进制输出。0x 开头，默认为 16 进制，0 开头的数，默认为 8 进制，输入二进制数字时，可通过先计算 16，8 进制再输入。

如何知道某种类型的数占多少字节？

sizeof 运算符

◆ 用于计算某种类型的对象在内存中所占的字节数。

```
#include <iostream>
using namespace std;
int main()
{
    cout << "sizeof(short int)=" << sizeof(short int) << endl;
    cout << "sizeof(int)=" << sizeof(int) << endl;
    cout << "sizeof(long int)=" << sizeof(long int) << endl;
    return 0;
}
```

图 4: 如何知道某种类型数据所占字节数

整型数据的分类

| | 有符号 | 无符号 |
|-----|--|--------------------------------------|
| 基本型 | int signed int | unsigned int |
| 短整型 | short short int signed short signed short int | unsigned short unsigned short int |
| 长整型 | long long int signed long signed long int | unsigned long unsigned long int |

图 5: 整型数据的分类

1.2.4 最大与最小整数

有符号数的最大值为：2147483647，最小值为：-2147483648。

1.3 浮点型

浮点型又称为实型 如果不设置精度，cout 默认精度为 6 位。超出字符类型精度外的位数会输出错误的数据。float 数据在内存中的存储方式如下，其他的类型不再细讲。

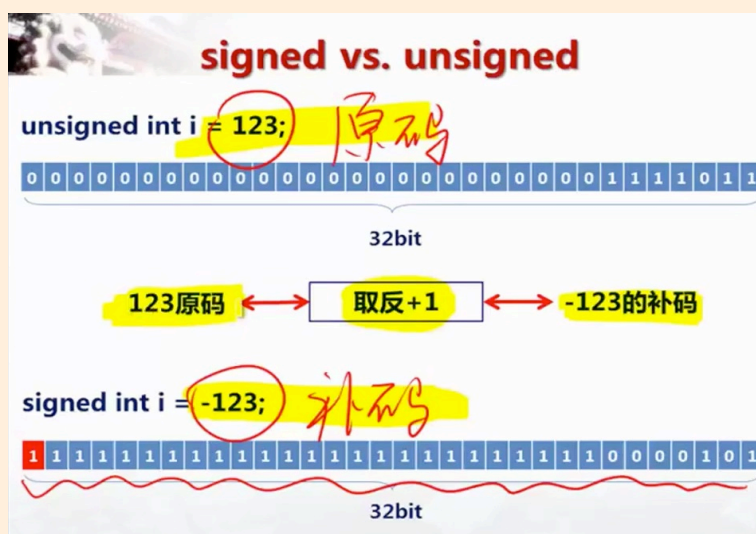


图 6: 如何求补码



图 7: 打印一个数的十六, 八, 十进制表示

1.4 字符型

一个字符型占一个字节, 一个字符在内存中转换成相应的数字来保存。字符型数据可以和整型数据相互赋值, 也可以和整数一样进行运算。

特殊字符: 转义字符, 转义字符的功能见下图:

1.5 布尔型

用于存“真”和“假”, 只占一个字节, 其值只能为 1 或 0, 1 代表 true, 0 代表 false。

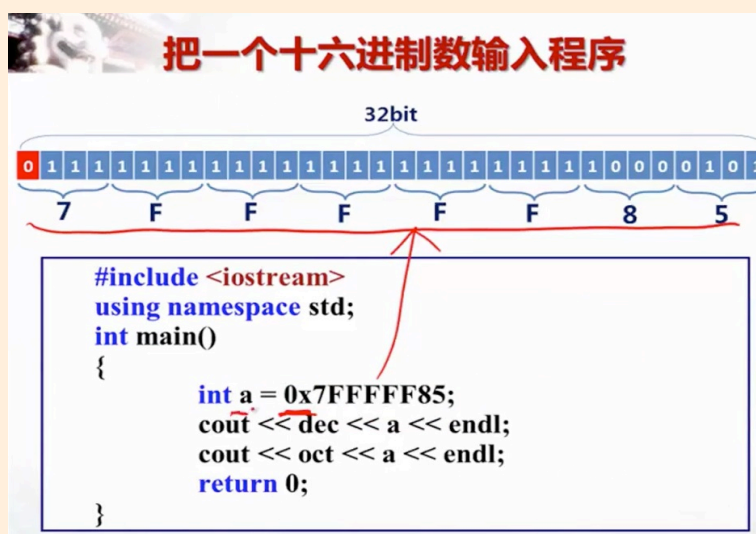


图 8: 输入十六进制数

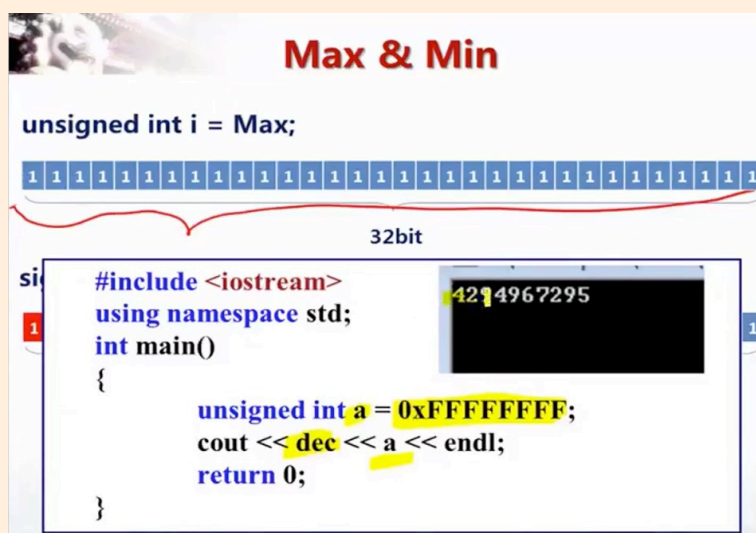


图 9: unsigned int 型数据定义的最大值

1.6 常数

常量：程序运行中，其值不变的量，有字面常量和符号常量两种（定义方法如下图）。也是有数据类型的。

常量的类型靠后缀来区分：

1.7 变量命名

现在常用的两种命名法：匈牙利命名法和驼峰命名法：

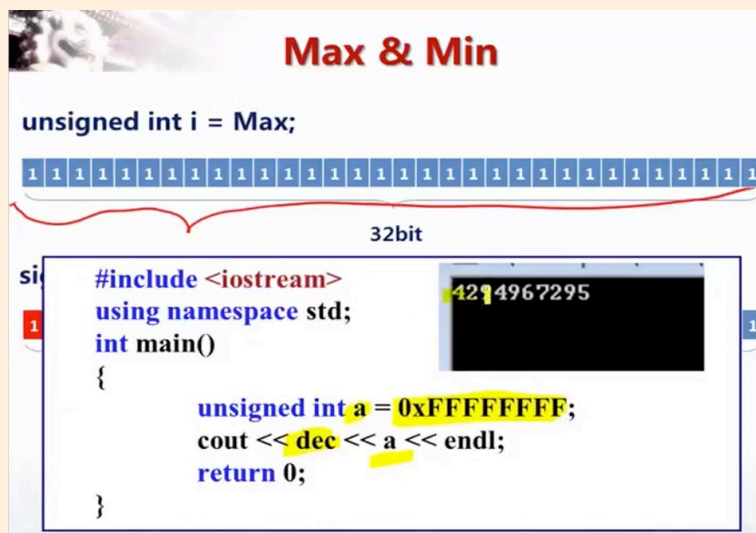


图 10: int 型数据定义的最小值

整型数据的范围

■ VC中每种类型所占内存空间和表示的范围：

| | | |
|----------------------|---|--------------------------------|
| short [int] | 2 | -32 768 ~ 32 767 |
| signed short [int] | 2 | -32 768 ~ 32 767 |
| unsigned short [int] | 2 | 0 ~ 65 535 |
| int | 4 | -2 147 483 648 ~ 2 147 483 647 |
| signed int | 4 | -2 147 483 648 ~ 2 147 483 647 |
| unsigned int | 4 | 0 ~ 4 294 967 295 |
| long [int] | 4 | -2 147 483 648 ~ 2 147 483 647 |
| signed long [int] | 4 | -2 147 483 648 ~ 2 147 483 647 |
| unsigned long [int] | 4 | 0 ~ 4 294 967 295 |

图 11: 各种数据类型的最大值和最小值

Chapter 2

c 语言中的运算成分



浮点型

浮点型 = 实型

| 浮点型 | 长度 | 有效位 | 范围 |
|---------------|-------|-----|---|
| float ✓ | 32bit | 7位 | $-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$ |
| double ✓ | 64bit | 15位 | $-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$ |
| long double ✓ | 64bit | 15位 | $-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$ |

图 12: 不同类型的浮点型数据

感受浮点型

```

#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float a = 3.141592653589793238462643383279502884197169399375
1058209749445923078164062862089986280348253421170679;
    double b = 3.141592653589793238462643383279502884197169399375
1058209749445923078164062862089986280348253421170679;
    long double c = 3.14159265358979323846264338327950288419716939
93751058209749445923078164062862089986280348253421170679;
    cout << setprecision(100) << a << endl;
    cout << b << endl;
    cout << c << endl;
    return 0;
}

```

图 13: cout 设置输出精度

2.1 赋值运算

2.2 算术运算

2.3 自增自减运算

2.4 关系运算

2.5 逻辑运算和混合运算

2.6 逗号, 条件, 强转

2.7 位运算

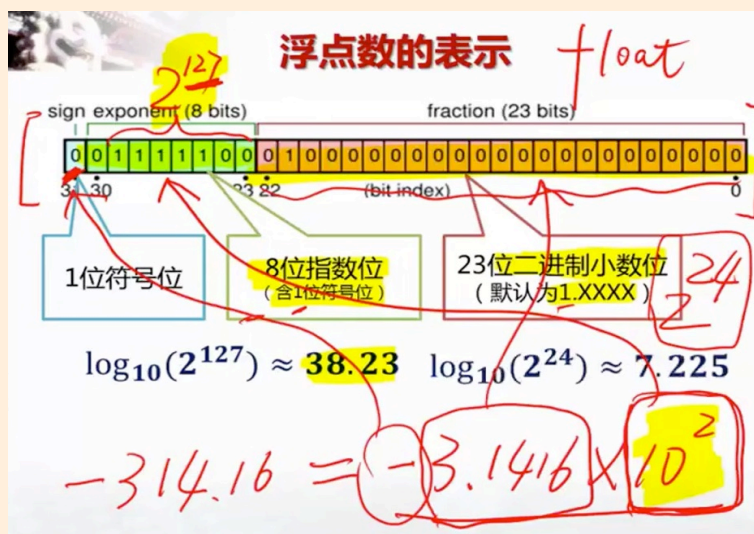


图 14: float 型数据的存储方式

| | | | | | | | | | | | | | | | |
|----|-------|----|----|----|---|-----|---|-----|---|-----|---|-----|-----|-----|---|
| 2 | <STX> | 34 | " | 66 | B | 98 | b | 130 | Ç | 162 | ç | 194 | ¬ | 226 | , |
| 3 | <ETX> | 35 | # | 67 | C | 99 | c | 131 | È | 163 | è | 195 | √ | 227 | " |
| 4 | <EOT> | 36 | \$ | 68 | D | 100 | d | 132 | Ë | 164 | ë | 196 | f | 228 | % |
| 5 | <ENQ> | 37 | % | 69 | E | 101 | e | 133 | Ö | 165 | • | 197 | ≈ | 229 | À |
| 6 | <ACK> | 38 | & | 70 | F | 102 | f | 134 | Ù | 166 | ¶ | 198 | Δ | 230 | Ê |
| 7 | <BEL> | 39 | ' | 71 | G | 103 | g | 135 | á | 167 | ß | 199 | « | 231 | Á |
| 8 | <BS> | 40 | (| 72 | H | 104 | h | 136 | à | 168 | ® | 200 | » | 232 | Ë |
| 9 | <TAB> | 41 |) | 73 | I | 105 | i | 137 | â | 169 | © | 201 | ... | 233 | È |
| 10 | <LF> | 42 | * | 74 | J | 106 | j | 138 | ä | 170 | ™ | 202 | | 234 | Í |
| 11 | <VT> | 43 | + | 75 | K | 107 | k | 139 | å | 171 | ' | 203 | Ä | 235 | Î |
| 12 | <FF> | 44 | , | 76 | L | 108 | l | 140 | ä | 172 | " | 204 | Å | 236 | Ï |
| 13 | <CR> | 45 | - | 77 | M | 109 | m | 141 | ç | 173 | # | 205 | Ö | 237 | ì |
| 14 | <SO> | 46 | . | 78 | N | 110 | n | 142 | é | 174 | Æ | 206 | Œ | 238 | Ó |
| 15 | <SI> | 47 | / | 79 | O | 111 | o | 143 | è | 175 | Ø | 207 | œ | 239 | Ô |
| 16 | <DLE> | 48 | 0 | 80 | P | 112 | p | 144 | ê | 176 | ∞ | 208 | - | 240 | • |
| 17 | <DC1> | 49 | 1 | 81 | Q | 113 | q | 145 | ë | 177 | ± | 209 | | 241 | ◊ |
| 18 | <DC2> | 50 | 2 | 82 | R | 114 | r | 146 | ë | 178 | ≤ | 210 | ˆ | 242 | Ù |
| 19 | <DC3> | 51 | 3 | 83 | S | 115 | s | 147 | ë | 179 | ≥ | 211 | ˜ | 243 | Ú |
| 20 | <DC4> | 52 | 4 | 84 | T | 116 | t | 148 | ë | 180 | ≤ | 212 | ˘ | 244 | Û |
| 21 | <NAK> | 53 | 5 | 85 | U | 117 | u | 149 | ë | 181 | ≤ | 213 | ˙ | 245 | Ü |
| 22 | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | |
| 28 | | | | | | | | | | | | | | | |
| 29 | | | | | | | | | | | | | | | |
| 30 | | | | | | | | | | | | | | | |

ASCII Character Codes

■ American Standard Code for Information Interchange
美国标准信息交换码

被ISO接纳为国际标准。

◆ ISO/IEC 646:1991
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=4777

图 15: ASCII 码

使用须知 0-255

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    char a = 64;
    int b = 'Z';
    int c = b - a;
    char d = 6 + 256;
    cout << a << " " << b << " " << c << " " << d << endl;
    return 0;
}
```

■ 由于存储类型和整型相同

- ◆ 可以与整型数据相互赋值
- ◆ 可以和整数一样进行运算

90 26

图 16: 整型和字符型的运算

| 转义字符 | 描 述 |
|------|------------------------------------|
| \a | 响铃(audible bell) |
| \b | 退格(backspace) |
| \f | 换页(formfeed) |
| \n | 换行(newline) |
| \r | 回车(carriage return) |
| \t | 水平制表(horizontal tab) |
| \v | 垂直制表(vertical tab) |
| \\ | 反斜线(backslash) |
| \' | 单引号(single quote) |
| \" | 双引号(double quote) |
| \DDD | 八进制数 DDD 对应的字符(octal number) |
| \xHH | 十六进制数 HH 对应的字符(hexadecimal number) |

图 17: 整型和字符型的运算

布尔型

```
#include <iostream>
using namespace std;
int main()
{
    bool b1 = true, b2 = false;
    cout << "b1 = true 时, b1 = " << b1 << endl;
    cout << "b2 = false 时, b2 = " << b2 << endl;
    b1 = 7 > 3;
    b2 = -100;
    cout << "b1 = 7 > 3 时, b1 = " << b1 << endl;
    cout << "b2 = -100 时, b2 = " << b2 << endl;
    return 0;
}
```

b1 = true 时, b1 = 1
 b2 = false 时, b2 = 0
 b1 = 7 > 3 时, b1 = 1
 b2 = -100 时, b2 = 1

图 18: 布尔型“非 0 即 1”

C/C++ 中的常量

■ 常量：在程序运行过程中，其值保持不变的量

- ◆ 字面常量
 - - 1, 0, 123, 4.6, - 1.23 ;
- ◆ 符号常量
 - 用一个标识符代表一个常量的，称为符号常量

```
#include <iostream>
using namespace std;
int main()
{
    const double PI = 3.14159;
    float r, area;
    cin >> r;
    area = r * r * PI;
    cout << "Area = " << area;
    return 0;
}
```

图 19: 符号常量 PI 的定义方法

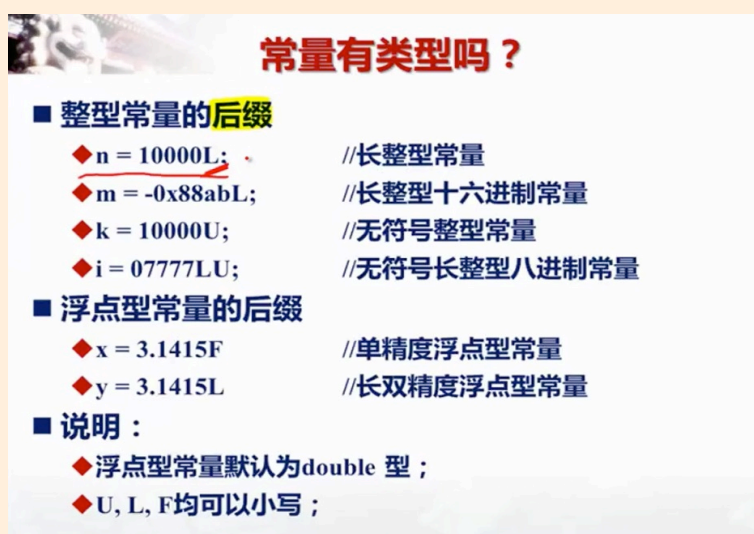


图 20: 符号常量 PI 的定义方法

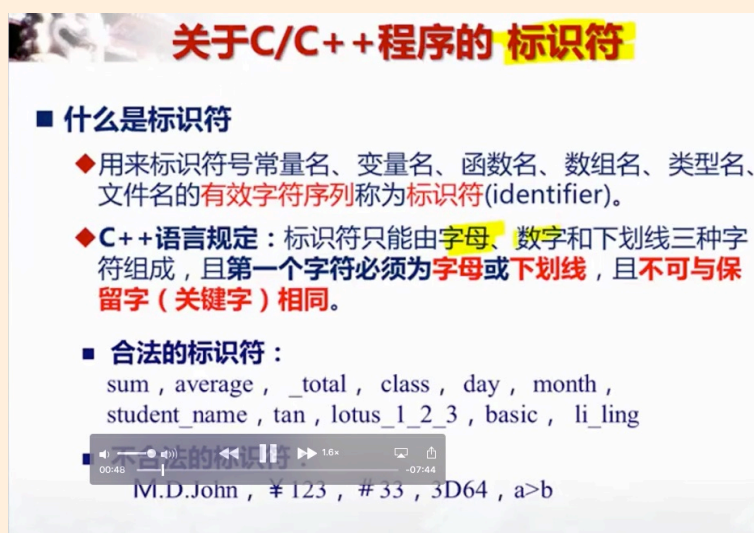
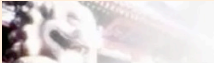


图 21: 变量标识符的定义



关于变量的命名

■ 匈牙利命名法


由Microsoft著名开发人员Excel主要设计者Charles Simonyi在其博士论文中提出。



1. 以一个或者多个小写字母开头，来指定数据类型。
2. 其后是一个或者多个第一个字母大写的单词，指出变量的用途。

如：`chGrade`; `nLength`; `bOnOff`; `strStudentName`;

图 22: 匈牙利命名法



关于变量的命名

■ 驼峰命名法 (骆驼式命名法)

- ◆ 由一个或多个单词连结在一起；
- ◆ 第一个单词以小写字母开始；
- ◆ 第二个单词的首字母大写或每一个单词的首字母都采用大写字母；

`myFirstName`, `myLastName`, `nextStudentName`,
`intCount`, `printEmployeePaychecks()`

图 23: 驼峰命名法