中山大學
SUN YAT-SEN UNIVERSITY

变量、常量和存储类

中山大学计算机学院

讲课人：万海

目录

CONTENTS

变量是啥？

变量是计算机语言中能储存计算结果或能表示值的抽象概念。简单来说，就是程序可操作的存储区的名称。C 中每个变量都有特定的类型，类型决定了变量存储的大小和布局，该范围内的值都可以存储在内存中，各种运算符可应用于变量上。

常见基本变量类型

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}

[]表示可省略

简单的二进制

128瓦	64瓦	32瓦	16瓦	8瓦	4瓦	2瓦	1瓦
1	1	1	0	1	0	1	0

- 信息复制的精确性
- 运算规则简单（R(R+1)/2）
- 电子线路制造计算机成为可能

我认识她已有1111年了。

简单的二进制

一般形式：

$$B = k_n*2^n + \ldots + k_0*2^0 + k_{-1}*2^{-1} + \ldots + k_{-m}*2^{-m}$$

基数为2，系数（数字）属于 {0, 1}

$$110110_{(2)} = 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$$

$$= 32 + 16 + 0 + 4 + 2 + 0 = 54_{(10)}$$

运算规则：

- 0+0=0　0+1=1　1+0=1　1+1=10
- 0*0=0　0*1=0　1*0=0　1*1=1

简单的二进制

二进制数位数与整数的表示范围的关系：

位数	1	3	8	16	n
范围	0~1	0~7	0~255	0~65535	$0 \sim 2^n-1$

计算机里常用固定位数方式表示整数，常见：

16位：$0 \sim 65535$　　32位：$0 \sim 2^{32} - 1$ (约40亿)

64位：$0 \sim 2^{64} - 1$ (约1600亿亿)

近似公式：$2^{10} = 1024 \approx 10^3$

简单的二进制

位（Bit）：度量数据的最小单位

字节（Byte）：最常用的基本单位

1 0 0 1 0 1 0 1 $=2^7+2^4+2^2+2^0=149$

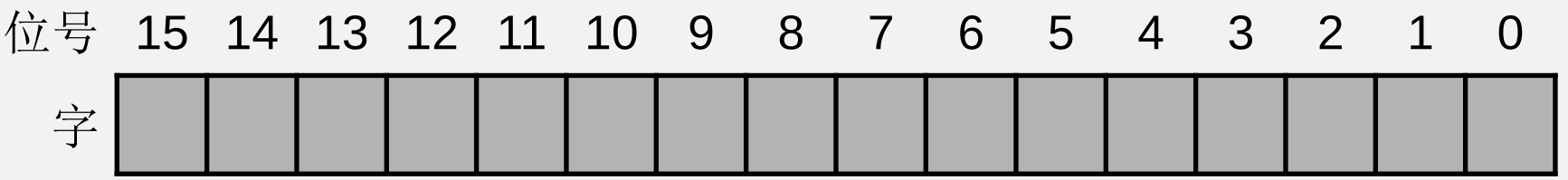
K 字节　　1K = 1024 byte

M（兆）字节　　1M = 1024 K

G（吉） 字节　　1G = 1024 M

T（太）字节　　1T = 1024 G

简单的二进制



双字节表示范围：0~65535

十六进制是人们在计算机指令代码和数据的书写中经常使用的数制。

用0，1，……，9和A，B，……，F（或a，b，……，f）这16个符号来描述 。

变量的定义

变量定义就是告诉编译器在何处创建变量的存储，以及如何创建变量的存储。变量定义指定一个数据类型，并包含了该类型的一个或多个变量的列表，如右所示 `type variable_list;`

在这里，type 是一个有效的 C 数据类型，variable_list 可以由一个或多个标识符名称组成，多个标识符之间用逗号分隔。下面列出几个有效的声明：

```
int     I, j, k;
char    _c, _ch;
float   f, salary;
double d;
```

变量的定义

变量可以在声明的时候被初始化（指定一个初始值）。初始化器由一个等号，后跟一个常量表达式组成，例如：

```
int d = 3, f = 5;                    // 定义并初始化 d 和 f
double y = 1.0;                      // 定义并初始化 y
char x = 'x';                        // 定义并初始化 x
```

注意，这样对于a、b的定义及初始化是不允许的！！！

```
int a = b = 3;
```

变量的声明

变量声明向编译器保证变量以指定的类型和名称存在，这样编译器在不需要知道变量完整细节的情况下也能继续进一步的编译。变量声明只在编译时有它的意义，在程序连接时编译器需要实际的变量声明。

变量的声明有两种情况：

- 定义：给变量分配存储空间；也叫定义声明。
- 声明：不给变量分配存储空间，因为它引用已有的变量，也叫引用声明；使用关键词**extern**，且**不进行初始化**

变量只能有**一次**定义,但是可以有**多次**声明。

左值和右值

C 中有两种类型的表达式：

1.左值（lvalue）：指向内存位置的表达式被称为左值（lvalue）表达式。左值可以出现在赋值号的左边或右边。

2.右值（rvalue）：术语右值（rvalue）指的是存储在内存中某些地址的数值。右值是不能对其进行赋值的表达式，也就是说，右值可以出现在赋值号的右边，但不能出现在赋值号的左边。

例如这是一个有效的赋值语句：`int g = 20;`

但是这个就不是一个有效的语句，会产生编译错误：`10 = 20;`

常量

常量是固定值，在程序执行期间不会改变。这些固定的值，又叫做**字面量**。常量可以是任何的基本数据类型，比如整数常量、浮点常量、字符常量，或字符串字面值，也有枚举常量。

常量就像是常规的变量，只不过常量的值在定义后不能进行修改。

常见的常量类型包括整数常量，浮点常量，字符(串)常量等。

整数常量

整数常量可以是二进制、十进制、八进制或十六进制的常量。
十进制：无特殊前缀　　二进制：前缀0b ；
八进制：前缀0　　十六进制：前缀0x
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
下面是几个例子：

```
212         /* 合法的 */
215u        /* 合法的 */
0xFeeL      /* 合法的 */
078         /* 非法的：8 不是八进制的数字 */
032UU       /* 非法的：不能重复后缀 */
```

注：老版本编译器无二进制表示方法

浮点常量

•浮点常量由整数部分、小数点、小数和指数部分组成，可以使用小数形式或者指数形式来表示浮点常量。
•小数形式表示时，必须包含整数部分、小数部分，或同时包含两者。
•指数形式表示时，e前面为尾数部分，必须有数，e后为指数部分，必须为整数
下面是几个例子：

```
3.14159          /* 合法的 */
314159E-5L       /* 合法的 */
510E             /* 非法的：不完整的指数 */
210f             /* 非法的：没有小数或指数 */
.e55             /* 非法的：缺少整数或分数 */
```

字符常量

常见的主要有两种表示方法：

- 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'

右图是一些常见的转义符。

转义符	含义
\\	\ 字符
\'	' 字符
\"	" 字符
\?	? 字符
\a	警报铃声
\b	退格键
\f	换页符
\n	换行符
\r	回车
\t	水平制表符
\v	垂直制表符
\ooo	一到三位的八进制数
\xhh . . .	一个或多个数字的十六进制数

字符串常量

- 含义:连续多个字符组成的字符序列
- 表示：括在双引号中，可以是图形字符，也可以是转义符
- 例如

```
"abc 123*#"
"\x61 \x62\x63\061 \62\063\x2a\043"
"\r\n\t\\A\\t\x1b\"\1234\xft \x2f\33"
```

常量定义声明

- 在 C 中，有两种简单的定义常量的方式：
- 使用 #define 预处理器。

```
#define N 10
#define M 100
#define CH '\t'
```

- 使用 const 关键字。

```
const int N =10000;
```

存储类

存储类定义了 C 程序中变量/函数的范围（可见性）和生命周期。这些说明符放置在它们所修饰的类型之前。
下面是 C 中可用的存储类：

- auto
- register
- static
- extern

auto存储类

auto 存储类是所有局部变量默认的存储类。

下面定义的两个变量类型是一样的，auto相当于限定变量只能在函数内部使用。

```
{
    int x;
    auto int y;
}
```

register存储类

register 存储类用于定义存储在寄存器中而不是 RAM 中的局部变

```
{
    register int  x;
}
```

意味着变量的最大尺寸等于寄存器的大小，且不能对它应用一元的‘&’ 运算符（因为它没有内存位置）。所以下面这样写是错的。

```
{
    register int x;
    int *a=&x;
}
```

static存储类

- 用static修饰变量后，变量的生命周期变成程序结束为止。
- 用static修饰全局变量，则全局变量的作用域从原来的整个程序，变为静态全局变量所在的文件内部；修饰局部变量，作用域不变。
- 用static修饰的变量，存储位置是全局数据区（即静态区）。

PS：修饰之前，全局变量存储在全局数据区，局部变量存储在栈区。

extern存储类

extern 存储类用于提供一个全局变量的引用，全局变量对所有的程序文件都是可见的。当有多个文件且定义了一个可以在其他文件中使用的全局变量或函数时，可以在其他文件中使用 **extern** 来得到已定义的变量或函数的引用。
下面是一个例子：

第一个文件：

```c
// a.c
int a=1;
```

第二个文件

```c
// b.c
#include <stdio.h>

extern int a;
int main()
{
    printf("a=%d",a);
}
```

编译运行结果：

```
PS C:\Users\86147\Desktop> gcc b.c a.c -o Moon.exe
PS C:\Users\86147\Desktop> ./Moon.exe
a=1
```

extern存储类

如果把a.c中a的定义加上static就会编译错误，因为用static修饰后对于其他文件就是不可见的了。

第一个文件：

```c
// a.c > ...
1   static int a=1;
2
3
```

编译结果：

```
PS C:\Users\86147\Desktop> gcc b.c a.c -o Moon.exe
C:\Users\86147\AppData\Local\Temp\ccAHgCvX.o:b.c:(.rdata$.refptr.a[.refptr.a]+0x0): undefined reference to `a'
collect2.exe: error: ld returned 1 exit status
```

输入输出

标准输入、输出主要包括缓冲区和操作方法两部分。缓冲区实际上可以看做内存中的字符串数组，而操作方法主要是指**printf、scanf、puts、gets，getcha、putcahr**等操作缓冲区的方法。在C++以及Java等面向对象的编程语言中，将缓冲区以及操作缓冲区的方法封装成一类对象，这类对象就称为流。

下面主要介绍一下**printf**和**scanf**函数。

printf和scanf函数

printf调用格式：printf("<格式化字符串>", <参量表>);
scanf调用格式：scanf("<格式化字符串>", <地址表>);
常见格式化控制符：

%c	读单字符	%g	读浮点数
%d	读十进制整数	%G	读浮点数
%l	读十进制Long型整数	%o	读八进制数
%ll	读十进制Long Long型整数	%s	读字符串
%i	读十进制、八进制、十六进制整数	%x	读十六进制数（其中abcdef小写，大写时会被忽略）
%e	读浮点数	%X	读十六进制数（其中ABCDEF大写，小写时会被忽略）
%E	读浮点数	%p	读指针值
%f	读浮点数		

printf和scanf函数

示例：

```c
#include<stdio.h>
int main()
{
    int a,b,c;

    printf("请输入: ");
    scanf("%d%d%d",&a,&b,&c);
    printf("%d %d %d\n",a,b,c);
    return 0;
}
```

运行结果：

```
PS C:\Users\86147\Desktop> gcc a.c
PS C:\Users\86147\Desktop> ./a.exe
请输入: 1 2 3
1 2 3
```

PS: printf和scanf在标准库stdio.h中。
&a、&b、&c 中的 & 是地址运算符，
分别获得这三个变量的内存地址。

printf和scanf函数

示例：

```c
#include <stdio.h>

int main()
{
    char str[233];

    printf("请输入: ");
    scanf("%s", str);
    printf("%s\n", str);

    return 0;
}
```

运行结果：

```
PS C:\Users\86147\Desktop> gcc a.c
PS C:\Users\86147\Desktop> ./a.exe
请输入: abc
abc
```

PS: 读取字符串的时候不需要加&，因为这里str本身就是指针，已经可以认为是地址了。

强制类型转换

强制类型转换是把变量从一种类型转换为另一种数据类型。例如，如果要把一个 long 类型的值转到一个int中，使用**强制类型转换运算符**来把值显式地从一种类型转换为另一种类型，如下所示：

```c
#include <stdio.h>

int main()
{
    int a = 21, b = 4;
    double c,d;
    c = (double) a / b;
    d = (double)(a / b);
    printf("c : %f\nd : %f\n", c ,d);

}
```

运行结果：

```
PS C:\Users\861
c : 5.250000
d : 5.000000
```

PS：这里输出结果会有差异，是因为对于整数a、b默认是整数的除法，需要强制转换后才能变成实数的除法。

中山大學
SUN YAT-SEN UNIVERSITY

谢谢

中山大学计算机学院

中山大学MOOC课程组