中山大學
SUN YAT-SEN UNIVERSITY

变量、常量和存储类I

中山大学计算机学院

主讲人：黎卫兵

中山大学MOOC工作组

目录

CONTENTS

01 变量

02 输入输出

03 常量

04 存储类

05 强制类型转换

A Simple C Program

```
/*compute area*/
#include<stdio.h>

int main() {
        /*define variables and constants*/
        double radius, area;
        const double PI = 3.14;

        /*input radius*/
        printf("input radius:");
        scanf("%lf",&radius);

        /*compute area*/
        area = radius * radius * PI;

        /*output results*/
        printf("the area for the circle of radius %lf is %lf",
                radius, area);
}
```

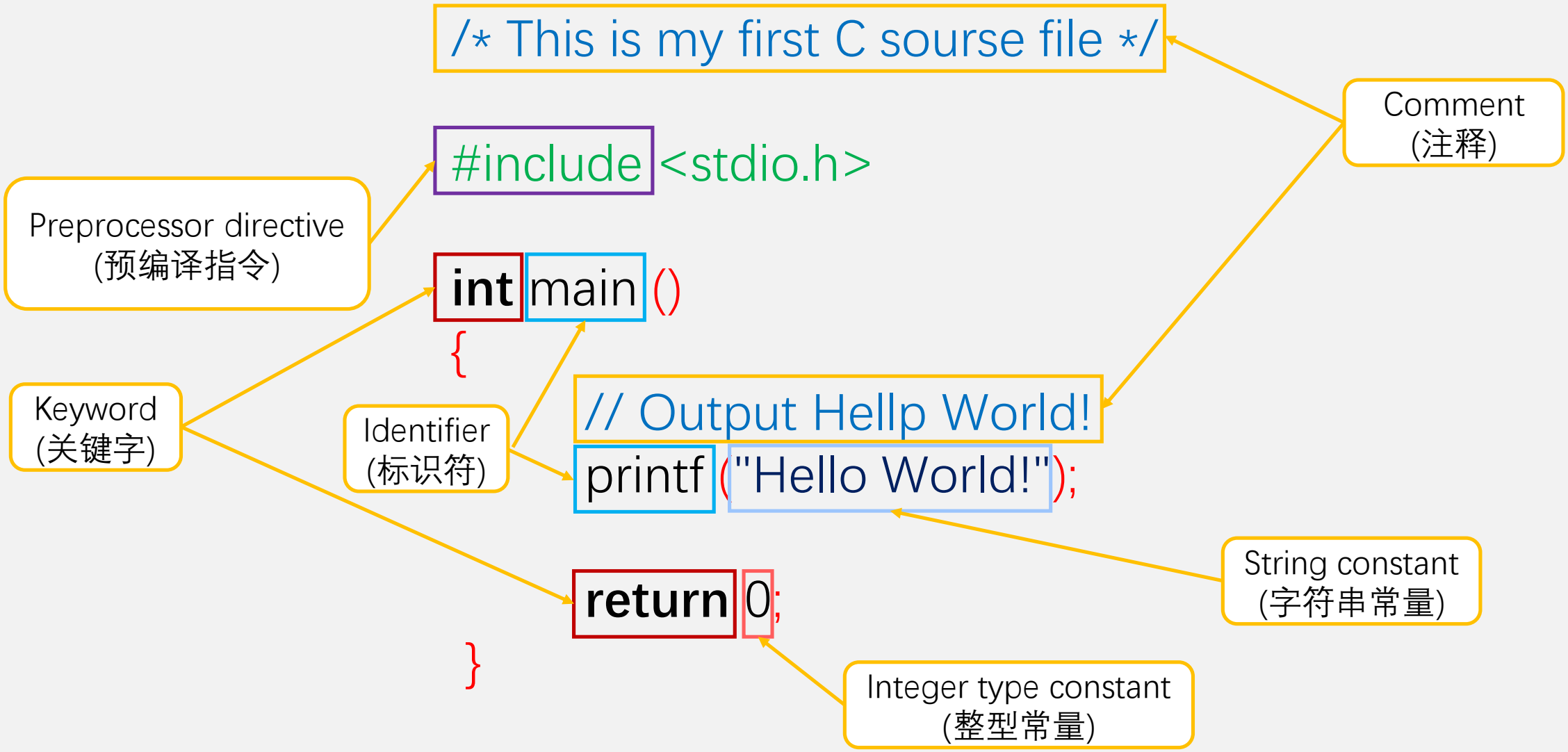
这是一个简单的程序:
①声明变量和常量;
②从键盘接收一个实数作为半径;
③计算面积;
④打印圆的面积。

求解过程非常接近自然语言，它包含哪些语言元素?

运行结果:

```
input radius:5
the area for the circle of radius 5.000000 is 78.500000
--------------------------------
Process exited after 3.542 seconds with return value 0
请按任意键继续. . .
```

Hello World


/* This is my first C sourse file */
#include <stdio.h>
int main ()
{
// Output Hellp World!
printf ("Hello World!");
return 0;
}
Comment
(注释)
Preprocessor directive
(预编译指令)
Keyword
(关键字)
Identifier
(标识符)
String constant
(字符串常量)
Integer type constant
(整型常量)

Another Example

```
#include <stdio.h>
int main ()
{
        int a = 1;
        double b = 2.0;

        // Output the values of a and b!
        printf("a = %d\n", a);
        printf("b = %lf", b);

        return 0;
}
```

Integer type constant
(整型常量)

Floating type constant
(实型/浮点型常量)

Variable
(变量)

Format specifier
(格式化控制符)

Escape character
(转义字符)

One More Example (C语言常见符号分类)

- ❖关键字（Keyword）
  - ➢又称保留字（C Reserved Word）
  - ➢C语言预先规定的，具有特殊意义的单词，不能用作普通标识符使用
- ❖标识符（Identifier）
  - ➢系统预定义标识符（C Standard Identifier）
    - ✓main, printf, scanf, etc.
  - ➢用户自定义标识符
    - ✓variable (变量)
    - ✓function name (函数名), etc.

```c
#include <stdio.h>

int Add(int a, int b)
{
    return (a + b);
}

main()
{
    int x, y, sum = 0;

    printf("Input two integers:");
    scanf("%d%d", &x, &y);
    sum = Add(x, y);
    printf("sum = %d\n", sum);
}
```

One More Example (C语言

- ❖关键字（Keyword）
  - ➢又称保留字（C Reserved Word）
  - ➢C语言预先规定的，具有特殊意义的单词，不能用作普通标识符使用
- ❖标识符（Identifier）
  - ➢系统预定义标识符（C Standard Identifier）
    - ✓main, printf, scanf, etc.
  - ➢用户自定义标识符
    - ✓variable (变量)
    - ✓function name (函数名), etc.

32 Keywords in C Programming Language with their Meaning

S.No	Keyword	Meaning
1	auto	Used to represent automatic storage class
2	break	Unconditional control statement used to terminate swicth & looping statements
3	case	Used to represent a case (option) in switch statement
4	char	Used to represent character data type
5	const	Used to define a constant
6	continue	Unconditional control statement used to pass the control to the begining of looping statements
7	default	Used to represent a default case (option) in switch statement
8	do	Used to define do block in do-while statement
9	double	Used to present double datatype
10	else	Used to define FALSE block of if statement
11	enum	Used to define enumarated datatypes
12	extern	Used to represent external storage class
13	float	Used to represent floating point datatype
14	for	Used to define a looping statement
15	goto	Used to represent unconditional control statement
16	if	Used to define a conditional control statement
17	int	Used to represent integer datatype
18	long	It is a type modifier that alters the basic datatype
19	register	Used to represent register storage class
20	return	Used to terminate a function execution
21	short	It is a type modifier that alters the basic datatype
22	signed	It is a type modifier that alters the basic datatype
23	sizeof	It is an operator that gives size of the memory of a variable
24	static	Used to create static variables - constants
25	struct	Used to create structures - Userdefined datatypes
26	switch	Used to define switch - case statement
27	typedef	Used to specify temporary name for the datatypes
28	union	Used to create union for grouping different types under a name
29	unsigned	It is a type modifier that alters the basic datatype
30	void	Used to indicate nothing - return value, parameter of a function
31	volatile	Used to creating volatile objects
32	while	Used to define a looping statement

- All the keywords are in lowercase letters
- Keywords can't be used as userdefined name like variable name, function name, lable, etc...
- Keywords are also called as Reserved Words

One More Example (C语言常见符号分类)

- ❖运算符（Operator）
  - ➢对操作数进行运算的符号
  - ➢+, -, *, /, %, etc.
- ❖分隔符（Separator）
  - ➢空格、回车/换行、逗号等
- ❖其他符号
  - ➢{和}标识函数体或语句块
  - ➢/*和*/是程序注释的定界符

```c
#include <stdio.h>

int Add(int a, int b)
{
    return (a + b);
}

main()
{
    int  x, y, sum = 0;

    printf("Input two integers:");
    scanf("%d%d", &x, &y);
    sum = Add(x, y);
    printf("sum = %d\n", sum);
}
```

Variable (变量)

【数学】**变量**是用于指代特定的**值（Value）**的**符号（Symbolic）**。例如

$$y = f(x) = 2x + 3$$

这里，y, x 都是变量，表示可变化的值

【计算机】**变量**是**储存**某个值或**对象（Object）**的**单元**名称。例如

$$x = 3;$$

$$y = 2 * x + 3;$$

这里：x, y 是两个**存储单元**。两个命令（指令/语句）含义是：第一步，数值 3存入x；第二步，x乘2+3后，结果9存入y。

Variable (变量)

```
int a = 1;
```

Variable (**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。变量的值在程序执行过程中是可以改变的。

- 变量的属性
  - Type (变量类型)
  - Address (变量的地址)
  - Name (变量名称)
  - Value (变量的值)

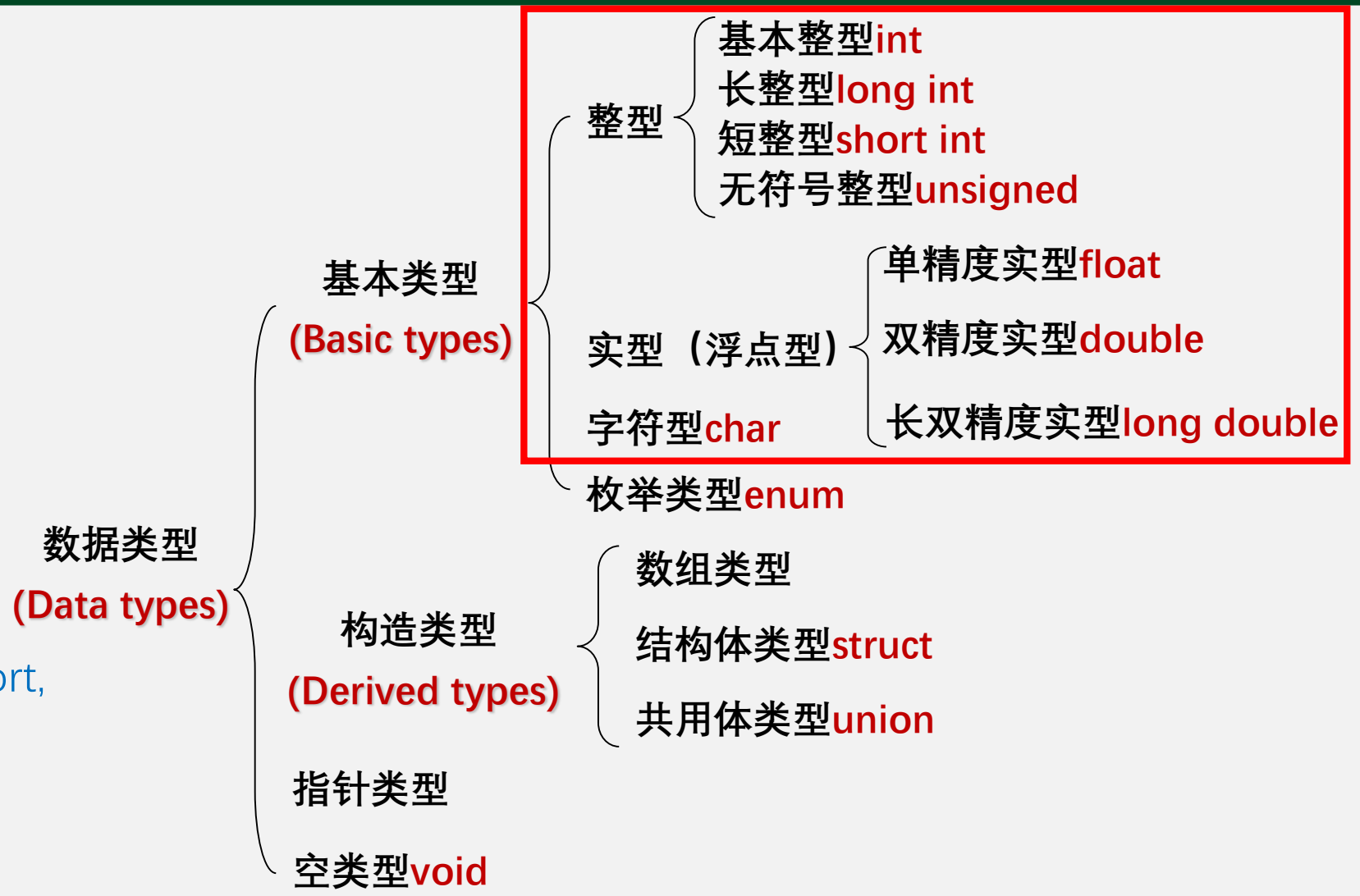
Data Types (数据类型)

❖基本数据类型

- int (整型)
- float (单精度浮点数)
- double (双精度浮点数)
- char (字符)

❖数据类型修饰符

- short: int
- long: int, double
- unsigned: char, int, short, long
- signed: char, int, short, long (default)



Basic Data Types (基本数据类型)

C语言中变量只能储存数值。数值分为

- 整数（int）
  - 自然数（unsigned）
  - 有符号数（signed）
- 实数（real）
  - 浮点数（float）
  - 双精度数（*double*）

用 char，short，int，long 存储不同范围的整数
用 float, double 存储不同精度的实数

Basic Data Types (基本数据类型)

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

变量名-User defined identifier (用户自定义标识符)

```
int a = 1;
```

- **Naming rules (命名规则)**
  - 只能由英文字母、数字、下划线组成
  - 必须以字母或下划线开头
  - 禁止使用C语言关键字（保留字）
  - 大小写敏感（sum, Sum, SUM）
  - 最大长度限制，与编译器相关
  - A variable name in C can be any valid identifier. An identifier is a series of characters consisting of letters (a-z, A-Z), digits (0-9) and underscores (_) that does not begin with a digit. C is case sensitive.

变量的定义-Variable declaration (变量声明)

变量定义就是告诉编译器在内存中创建变量的存储，并制定用这个名称（标识符）访问、或修改存储的内容。具体的语法:

$$\boldsymbol{type}\ variable\_list;$$

这里，$\boldsymbol{type}$ 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:

```
/*define vars and constants*/
double radius, area;

int i,j;
char c;
```

通常，我们使用熟悉的单词作为变量名称，例如 radius, height, width 等

变量的定义-Variable declaration (变量声明)

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。**变量必须先声明，赋初值，后使用！**

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。

变量的定义-Variable initialization (变量初始化)

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：

```
/*define vars and constants*/
double radius = 5;
double area;

int i = 0, j = 0;
```

注意，这样对于a、b的定义及初始化是**不允许的！！！**

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

更多关于定义与申明的知识：声明 - cppreference.com

Variable assignment (变量的赋值)：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;`

Variable assignment (变量的赋值)

定义或声明后的变量，可用 “=” 赋予值，称为**赋值**（assignment）；

有定义的变量可以在表达式中使用。例如:

```
/*define vars and constants*/
double radius = 5;
double area;

/*compute area*/
area = radius * radius * 3.14;
```

初始化!

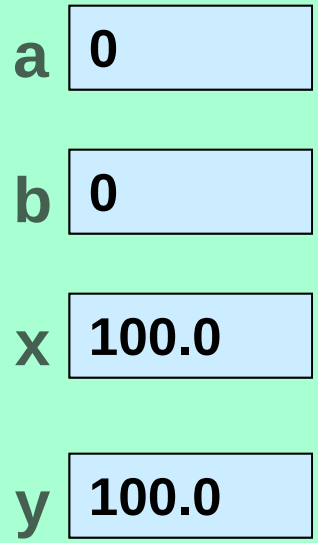
赋值!

最后一个语句是赋值语句。等号右边是一个表达式，它使用radius变量的值计算出圆面积，通过赋值把结果值写入左边 area 这个变量之中。

Variable assignment (变量的赋值)

Example:

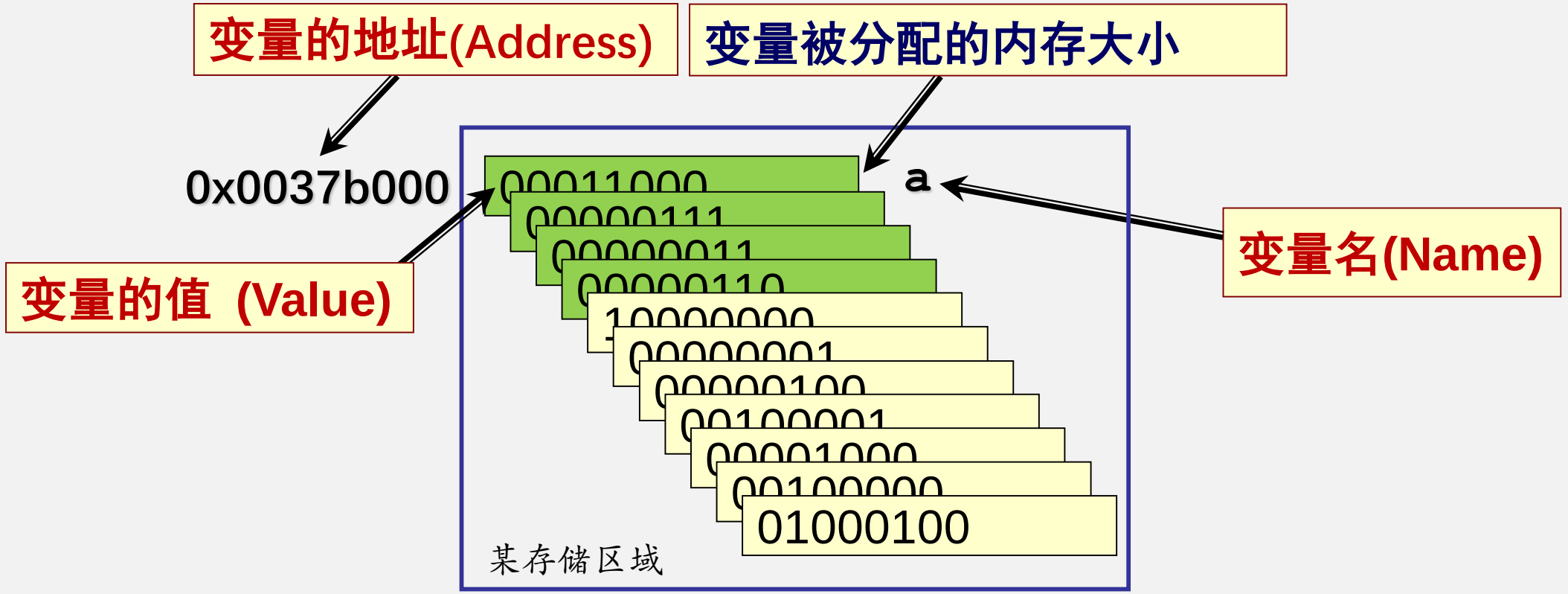
```
int a, b;
float x, y;
. . .
a = b = 0;
x = y = 100.0;
```



Variable assignment (变量的赋值)

- 变量类型-决定变量被分配内存空间的大小、数据存储方式、合法取值范围、可参与的运算

```
int a = 100861720;   ⟺   int a;
                         a = 100861720;
```



Input & Output (输入与输出)

如何输出变量的内容让我们看到?

把内存中变量或表达式计算结果在显示器、打印机或磁盘文件等外部设备，称为输出。默认的输出设备称为**标准输出**（standard output），一般是**控制台**（console）。
从外部设备，如键盘，读入数据到内存变量，称为输入。默认的输入设备称为**标准输入**（standard input），一般是**键盘**（keyboard）。

操作方法主要是是 **printf、scanf**、puts、gets，getchar、putchar 等。这些函数在 stdio.h 中定义。

变量输出的案例

```
/*compute area primary*/
#include<stdio.h>

int main() {
        /*define vars and constants*/
        double radius = 5;
        double area;

        /*compute area*/
        area = radius * radius * 3.14;

        /*output results*/
        printf("the area for the circle of redius %lf is
%lf\n",
                radius, area);
}
```

注意：
①格式化字符中转换数据的个数与参数一致；
②转换数据类型与参数类型一致。

使用 printf 输出到标准输出设备；
printf 语法格式：

- printf(格式化字符串， 变量或表达式列表）；
- 格式化字符串中指示符：
  - % 是输出引导符
  - **hhd,hd, d, ld** 对应 char, short, int, long int
  - **f,lf** 对应 float, double

运行结果：

```
input radius:5
the area for the circle of radius
5.000000 is 78.500000
--------------------------------
Process exited after 3.542 seconds
with return value 0
请按任意键继续. . .
```

中山大學
SUN YAT-SEN UNIVERSITY

格式化（format）与指示符

格式字符串由**普通字符**（除了 % ）和**转换指示**构成，前者被复制到输出而无更改。
每个转换指示拥有下列格式：

- 引入的 % 字符
- (可选)一或多个修改转换行为的标签：
  - -：转换结果在域内左校正（默认为右校正）
  - +：有符号转换的符号始终前置于转换结果（默认结果前置负号仅当它为负）
  - 空格：若有符号转换的结果不以符号开始，或为空，则前置空格于结果。若存在 + 标签则忽略空格。
  - #：进行替用形式的转换。准确的效果见下表，其他情况下行为未定义。
  - 0：对于整数和浮点数转换，使用前导零代替空格字符填充域。对于整数，若显式指定精度，则忽略此标签。对于其他转换，使用此标签导致未定义行为。若存在 - 标签则忽略 0 。
- (可选)指定最小**域宽**的整数值或 * 。若有要求，则结果会以空格字符（默认情况）填充，在右校正时于左，左校正时于右。使用 * 的情况下，以一个额外的 int 类型参数指定宽度。若参数值为负数，则它导致指定 - 标签和正域宽。（注意：这是最小宽度：决不被截断值）。
- (可选)后随整数或 * 或两者皆无的 . 指示转换的**精度**。在使用 * 的情况下，精度由额外的 int 类型参数指定。若此参数的值为负数，则它被忽略。若既不使用数字亦不使用 * ，则精度采用零。精度的准确效果见下表。
- (可选)指定参数大小的长度修饰符

```
/*format testing*/
#include<stdio.h>

int main() {
    double radius = 5;

    printf("%lf\n",radius);
/*default 8.6lf*/
    printf("%10lf\n",radius);
    printf("%10.2lf\n",radius);
    printf("%-10.2lf\n",radius);
    printf("%+10.2lf\n",radius);
    printf("% 10.2lf\n",radius);
    printf("%010.2lf\n",radius);
    return 0;
}
```

详细格式指示符：printf, fprintf, sprintf, snprintf, printf_s, fprintf_s, sprintf_s, snprintf_s - cppreference.com

课堂练习

将下面自然语言翻译成 C 代码

- 第一步：使用一个名为 miles 初始值为 100 的实数变量。
- 第二步：将 miles 乘以 1.609 并将它赋值给一个名为 kilometers 的变量。
- 第三步：显示 kilometers 的值，小数部分保留两位。

输入变量的案例

```
/*io simple*/
#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;
}
```

注意：
①格式化字符中普通字符必须与输入的一样；
②变量数量和类型必须与格式化字符串需要的类型一致；
③必须是变量的地址。

使用 scanf 从标准输入读入；
scanf 语法格式：

- scanf(格式化字符串， 变量**地址**列表）；
- 格式化字符串：
  - % 是输出引导符
  - **hhd,hd, d, ld** 对应 char, short, int, long int
  - **f,lf** 对应 float, double

运行结果：

```
请输入： 1 2,3
1,2,3

--------------------------------
Process exited after 6.917 seconds with return value 0
请按任意键继续. . .
```

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
%d	读十进制整数
%l	读十进制Long型整数
%ll	读十进制Long Long型整数
%i	读十进制、八进制、十六进制整数
%e	读浮点数
%E	读浮点数
%f	读浮点数

%g	读浮点数
%G	读浮点数
%o	读八进制数
%s	读字符串
%x	读十六进制数（其中abcdef小写，大写时会被忽略）
%X	读十六进制数（其中ABCDEF大写，小写时会被忽略）
%p	读指针值

一个简单、完整的 C 程序

```
/*compute area*/
#include<stdio.h>

int main() {
	/*define vars and constants*/
	double radius, area;
	const double PI = 3.14;

	/*input radius*/
	printf("input radius:");
	scanf("%lf",&radius);

	/*compute area*/
	area = radius * radius * PI;

	/*output results*/
	printf("the area for the circle of radius %lf is %lf",
		radius, area);
}
```

对于单一main函数的程序，多数会呈现四个步骤：

- 定义变量
- 输入数据
- 处理或计算
- 输出数据

运行结果：

```
input radius:5
the area for the circle of radius 5.000000 is 78.500000
--------------------------------
Process exited after 3.542 seconds with return value 0
请按任意键继续. . .
```

Constant (常量)

Constant (**常量**) 是固定值，在程序执行期间不会改变。

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；也可以是复合数据类型（称为literal**字面量**），如字符串字面量。

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，字符常量，字符串字面量等。

Integer constant (整数常量)

整数常量可以是Binary(二进制)、Decimal(十进制)、Octal(八进制)或Hexadecimal(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：C23开始，才有二进制常量表示方法。

本课程标准为C99，二进制常量可能编译无法通过！！！

Floating-point constant (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 */
```

Character constant (字符常量)

英文字母用0-127的整数表示，类型为 char

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

String constant (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"
```

Constant declaration (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： **无等号=和分号;**

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

- 使用 const 关键字。

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

字符串（string）的定义、输入与输出

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。

Storage class (存储类)

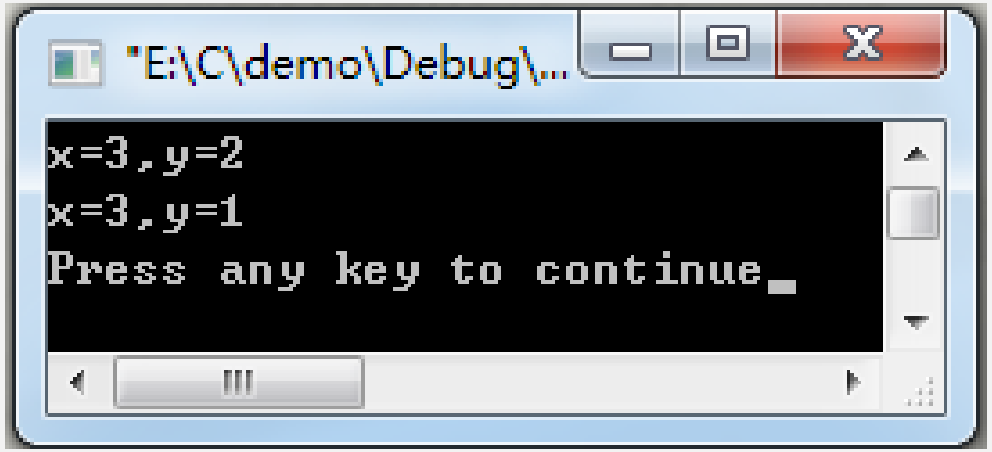
存储类定义了 C 程序中变量/函数的范围（可见性,作用域）和生命周期。这些说明符放置在它们所修饰的类型之前。下面是 C 中可用的存储类:

- auto (Automatic variable)
- register (Register variable)
- static (Static variable)
- extern (Extern variable)

Local variable (局部变量)

- ❖有效范围仅为该语句块（函数，复合语句）
- ❖仅能由语句块内的语句访问，退出语句块时释放内存，不再有效

```
int main(void)
{
    int  x = 1, y = 1;
    {
        int  y = 2;
        x = 3;
        printf("x=%d,y=%d\n", x, y);
    }
    printf("x=%d,y=%d\n", x,  y);
    return 0;
}
```

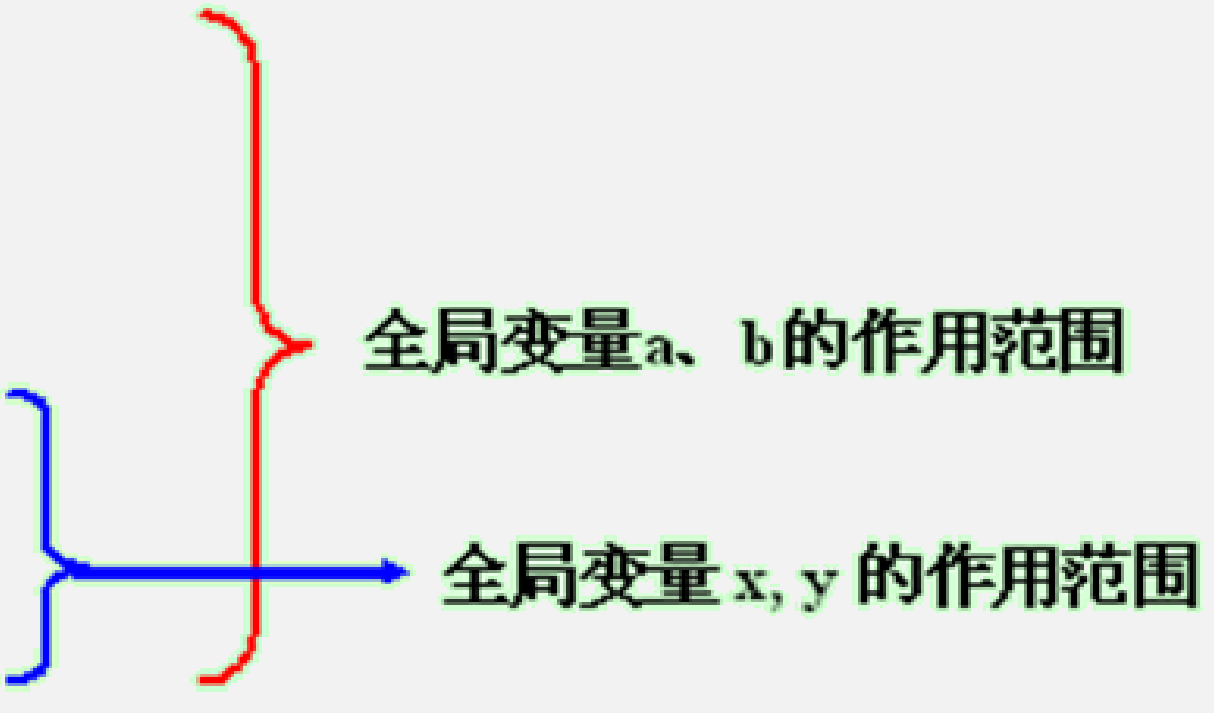


Global variable (全局变量)

❖有效范围是从定义变量的位置开始，到本程序结束

例如：

```
int a,b;
main()
{
……
}
int x,y;
f()
{
……
}
```



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**局部变量**。

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

这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取址 & 运算符（因为它没有内存位置）。所以下面这样写是错的。

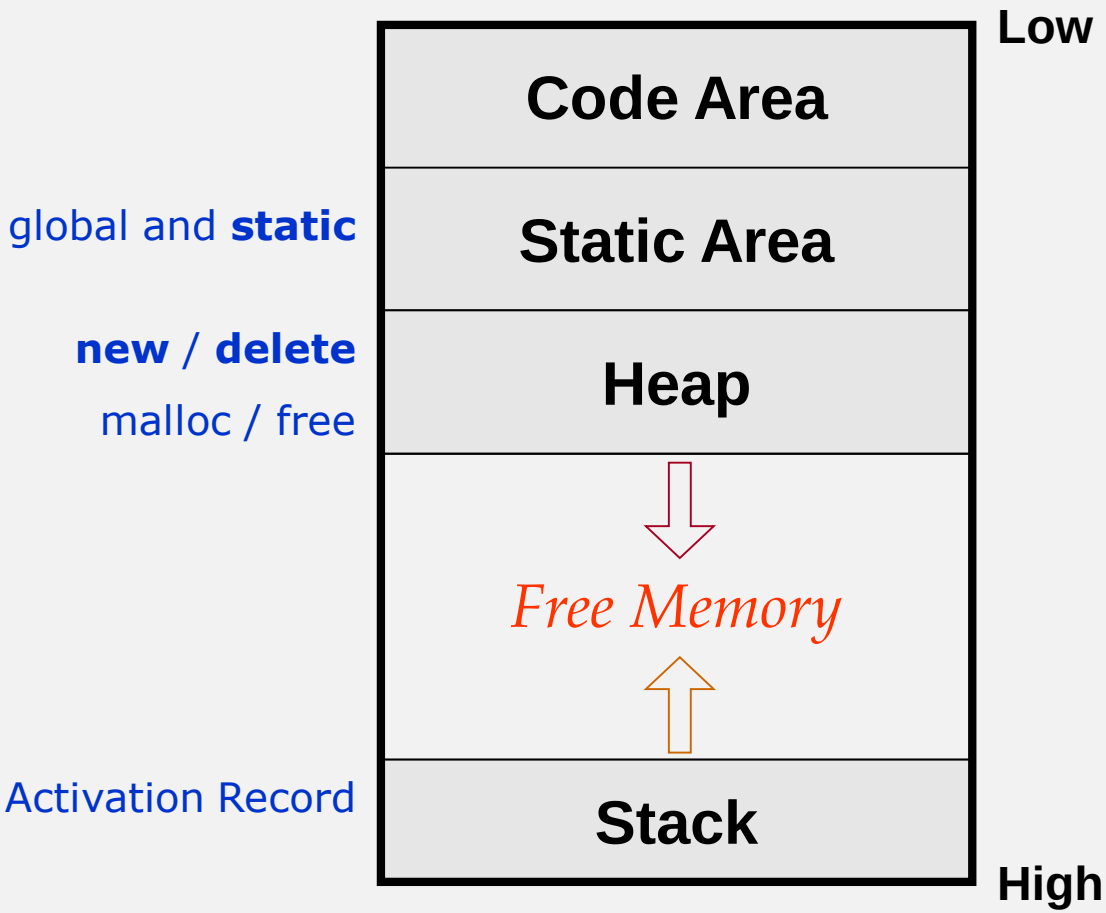
```
{
    register int x;
    int *a=&x;
}
```

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（动态区的一种）**。

static存储类


Low
Code Area
global and static
Static Area
new / delete
malloc / free
Heap
Free Memory
Activation Record
Stack
High

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
int a=1;
```

第二个文件

```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
```

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。

课堂练习

编写 C 程序

（计算圆柱体的体积）编写一个读取圆柱的半径和高并，利用下面的公式计算圆柱体底面积和体积的程序：

$$area = radius * radius * \pi$$
$$volume = area * length$$

这里是运行示例：
Enter the radius and length of a cylinder： 5.512
The area is 95.03
The volume is 1140.40

中山大學
SUN YAT-SEN UNIVERSITY

谢谢

中山大学计算机学院

主讲人：黎卫兵　　中山大学MOOC课程组