

# 2 初识C程序(下)

郭延文

2022级苏州校区技术科学试验班

# C语言的字符集

## ▶ 字符集(symbol set): 构成语言的基本符号

- ▶ 大小写英文字母
- ▶ 阿拉伯数字
- ▶ 特殊符号

~!#%^&\* \_ - + = | \ : ; " ' , . ? / ( ) { } [ ] < > 制表 空格 回车换行

## ▶ 程序中不能出现字符集之外的字符 (“引号内” 除外)

×	√	π	`	@	\$	“”	,	‘’	;
---	---	---	---	---	----	----	---	----	---



键盘上没有的符号

键盘上有的符号

汉字库里的符号!

# C程序的最基本“元素”：单词

---

- ▶ 单词 (token)：由字符集中的字符按照一定规则构成，是程序设计语言的基本单位
  - ▶ 关键词
  - ▶ 标识符
  - ▶ 字面常量

# 关键词 (keyword)

---

- ▶ 保留词汇，有固定的作用和含义，通常由小写字母组成，在程序中不能用作其他目的
  - ▶ 表示数据类型：auto、**char**、const、**double**、enum、float、**int**、long、register、short、signed、struct、union、unsigned、**void**...
  - ▶ 表示语句：break、continue、do、else、for、goto、if、**return**、switch、**while**...
  - ▶ 表示标号：case、default...
  - ▶ 其他关键词：extern、sizeof、static、typedef...

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;
```

```
double sum = 0;
```

```
char ch = 'm';
```

```
printf("Input n: ");
```

```
scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
sum = sum + PI * d;
```

```
d = d + 1;
```

```
}
```

```
printf("The sum is: %f ", sum);
```

```
putchar(ch); //显示计量单位
```

# 标识符 (identifier)

---

- ▶ 由字符集中的大小写英文字母、阿拉伯数字和下划线组成，**其首字符必须字母或下划线**

- ▶ i、price\_car、myFun、Node\_3、Loop4、PI、pi

- 可以用作变量、函数、形式参数、类型、符号常量、以及语句标号的名称。

5x、stu.score、number 1、y-average

×

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;  
  double sum = 0;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
  return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f", sum);
```

```
putchar(ch); //显示计量单位
```

# 标识符 (identifier)

- ▶ 程序中的标识符必须有定义 (definition)，即必须赋予某标识符一定的含义，未定义的 (undefined) 标识符不能使用
  - ▶ 或系统 (预) 定义；或我们自己定义
- ▶ 预定义标识符
  - ▶ 在源代码中前面加 #，编译后被相应的内容取代，如：define、include、if、else、ifdef、endif、...
- ▶ 自定义标识符
  - ▶ 不能和关键词或预定义标识符重复。int ✗
  - ▶ 在相同的有效范围内，已定义的标识符不能重复定义



# 字面常量 (literal constant)

---

- ▶ 常量用于表示在程序执行过程中不会改变或不允许被改变的数据，如：圆周率、一个星期的天数等。
- ▶ 程序中直接书写的常量
  - ▶ 整数 (如10)
  - ▶ 小数 (如3.14)
  - ▶ 字符常量 (如'm')
  - ▶ 字符串常量 (如"Hello World!")

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;  
  double sum = 0;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f", sum);  
putchar(ch); //显示计量单位
```

# C语言的符号常量 (manifest constant)

符号常量 (又叫命名常量)：标识符的一种，对于程序中多次使用的字面常量，可以定义成符号常量，即通过常量定义给常量取一个名字，在程序中通过常量名来使用这个字面常量。

## ▶ 常量定义

### ▶ #define <符号常量> <字面常量>

#### ▶ 如 #define PI 3.1415926

#### ▶ define: 系统预定义标识符, 预处理命令

- 预处理器会在编译前把源程序语句中的符号常量全部替换成相应的字面常量，参加编译的是替换后的字面常量。

#### ▶ 预处理命令行的末尾一般不加分号

- 常量定义的末尾如果有分号，分号会被当成是字面常量的一部分，从而有可能造成语法错误。

#### ▶ 这种定义方式又叫宏 (Macro) 定义，不会像变量定义那样检查类型。

# C语言的操作符 (operator) 与表达式 (expression)

- ▶ 操作符(运算符): 用于描述程序中的操作
  - ▶ 系统定义了一批操作符的功能和操作规则, 实现基本运算
    - ▶ 如=、<=、+、\*、()、-、/、%
  - ▶ 字面常量、符号常量、变量与函数可以作为操作符的基本操作对象, 又叫操作数
- ▶ 表达式: 用操作符将操作数连接起来的式子, 分号结尾
  - ▶ 如 `d = d + 1;`
  - ▶ 最简单的表达式是一个操作数, 如一个字面常量 `1`; 一个变量 `d`;
  - ▶ 表达式也可以作为操作数参加运算

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;  
  double sum = 0;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f", sum);  
putchar(ch); //显示计量单位
```

# C语言的标点符号 (punctuation)

---

- ▶ 标点符号在程序中起到某些语法、语义上的作用，特别是分隔作用。
  - ▶ 井号 (#) 表示预处理命令行     `# define PI 3.14`
  - ▶ 分号 (;) 表示一条语句的结束     `int i = 1;`
  - ▶ 逗号 (,) 分隔多个参数或变量     `int i, j;`
  - ▶ 空格 ( ) 作为词汇之间的间隔     `# define PI 3.14`
  - ▶ 冒号 (:) 分隔语句标号与语句
  - ▶ 行尾的反斜杠 (\) 是续行符，可以表示该字符串未完待续
  - ▶ ...

# C语言的注释 (comments)

---

注释不被编译和执行,单行注释: 以//开始;多行注释: 以/\*开始, 以\*/结束

- ▶ 用来提示或解释程序的含义

/\* This is a ... Program written by Yanwen Guo.

It's function is to ... ..

If any questions, please feel free to contact ... \*/

- ▶ 在调试程序时, 对暂时不执行的语句或备份的代码段落也可用注释符分离出来

...

// solution1;

solution2;

....



# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 2;  
  double sum = PI;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f ", sum);  
putchar(ch); //显示计量单位
```



# C语言的语句 (statement)

---

- ▶ 表达式末尾加一个分号可以构成语句
  - ▶ `d = d + 1;`
- ▶ 最简单的语句是一个分号，即空语句，不执行任何操作
- ▶ 可以用一对花括号将多个语句括起来，形成复合语句，一个复合语句可以看作一个语句块

```
{  
    sum = sum + PI * d;  
    d = d + 1;  
}
```
- ▶ 一些关键词，如while、return等，与表达式、分号或（子）语句配合也可以构成语句

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 2;  
  double sum = PI;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)  
{
```

```
    sum = sum + PI * d;  
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f", sum);  
putchar(ch); //显示计量单位
```

## 举 例

---

int n=1;

double sum = PI;    // 园周长之和, 为什么不是  
                      // int sum; ?

char ch = 'm';



# C语言常用的简单数据类型

---

## ▶ 数据类型

- ▶ 一个值的集合以及定义在这个值集上的一组操作
- ▶ 描述数据的取值属性

## ▶ 常用的基本类型

- ▶ 整型 int
- ▶ 实型 float, double
- ▶ 字符型 char
- ▶ 空类型 void



# C语言变量（注意以分号;结尾！）

---

## ▶ 变量的定义（definition）

▶ 通过给出变量的类型及其名字来规定变量的类型和名字属性

▶ `int i = 1;`

▶ 相同类型的多个变量可以并列定义

▶ `int i, j, k;`

▶ `float f, d;`

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;
```

```
    double sum = 0;
```

```
    char ch = 'm';
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    .....
```

```
    return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f ", sum);  
putchar(ch); //显示计量单位
```

# 变量的赋值（assignment）与初始化（initialization）

- ▶ 通过给变量赋值，可以使变量获得有意义的值；变量的值在程序运行过程可能发生改变。
- ▶ C语言中用**等号**表示赋值：

```
int i;
```

```
i = 5;
```

- ▶ 变量的值可以在定义的时候赋一个初值，即初始化（在程序编译期间完成赋值任务），如：

```
int n, d = 2; // 建议定义变量的时候同时赋初值
```

- ▶ 也可以在程序执行期间赋值，如：

```
d = d + 1;
```

# Tips

- ▶ 一个良好的习惯：尽可能在定义变量的同时初始化该变量！
  - ▶ `int i = 0;`
  - ▶ `float d = 0;`
- ▶ 如果变量的引用处和其定义处相隔较远，变量的初始化很容易被忘记
- ▶ 使用没有被初始化的变量，是非常“危险”的！



# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;  
  double sum = 0;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f ", sum);  
putchar(ch); //显示计量单位
```

## 变量值的输入 (C)

- ▶ 输入 (input) : 程序执行过程中, 将需要的数据从键盘、文件等输入内存
- ▶ 程序员可以在标准库的基础上实现输入功能, 此时需要在程序头部用 `#include <stdio.h>` 包含输入函数的说明信息
- ▶ scanf
- ▶ getchar

```
#include <stdio.h>
int main()
{
    char ch;
    ch = getchar();           // scanf("%c", &ch);
    return 0;
}
```

格式符

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;  
  double sum = 0;  
  char ch = 'm';  
  printf("Input n: ");  
  scanf("%d", &n);
```

```
.....
```

```
return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f ", sum);  
putchar(ch); //显示计量单位
```

```
#include <stdio.h>
int main()
{
    int m, n;
    scanf("%d%d", &m, &n);
    printf("%d\n", m + n);
    printf("%d\n", m * n);
    return 0;
}
```

567 85  
652  
48195

567  
85  
652  
48195

```
scanf("m=%d%d", &m, &n);
```

m=567  
85  
652  
48195

# 求两个输入数的和与乘积 (C)

```
#include <stdio.h>
int main()
{
    double m, n;
    scanf("%lf%lf", &m, &n);
    printf("%.1f\n", m + n);
    printf("%.1f\n", m * n);
    return 0;
}
```

567.1 85.1  
652.2  
48260.2

567.1  
85.1  
652.2  
48260.2

# 数据的输出 (C)

---

- ▶ 输出 (output)：将程序执行的结果输出（显示）到显示器、文件和打印机上等。
- ▶ 程序员可以在标准库的基础上实现输出功能，调用函数库中的输出函数，需要在程序头部用 `#include <stdio.h>` 包含输出函数的说明信息
  - ▶ `printf`
  - ▶ `putchar()`

# 一个完整的例子

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
```

```
#define PI 3.14
```

```
int main( )
```

```
{ int n, d = 1;
```

```
    double sum = 0;
```

```
    char ch = 'm';
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    .....
```

```
    return 0;
```

```
}
```

```
while(d <= n)
```

```
{
```

```
    sum = sum + PI * d;
```

```
    d = d + 1;
```

```
}
```

```
printf("The sum is: %f ", sum);
```

```
putchar(ch); //显示计量单位
```

```

#include <stdio.h>
int main()
{
    printf("a simple example of calculation : \n");
    printf("567 + 85 = %d \n", 567 + 85);
    printf("567 * 85 = %d \n", 567 * 85);
    printf("5.67 * 8.5 = %.1f\n", 5.67 * 8.5);
    printf("%c\n", 'A');
    return 0;
}

```

**a simple example of calculation:**

567 + 85 = 652

567 \* 85 = 48195

5.67 \* 8.5 = 48.2

A

```

#include <iomanip>
cout << setiosflags(ios::fixed) << setprecision(1) << ...

```



```
#include <stdio.h>
int main()
{
    printf("a simple example of calculation : \n");
    int m, n;
    scanf("%d%d", &m, &n);
    printf("%d + %d = %d \n", m, n, m + n);
    printf("%d * %d = %d \n", m, n, m * n);
    return 0;
}
```

**a simple calculation:**

**567**

**85**

**567 + 85 = 652**

**567 \* 85 = 48195**

```
#include <stdio.h>
int main()
{
    printf("a simple example of calculation: \n");
    int m, n;
    scanf("%d%d", &m, &n);
    printf("%d + %d = %d \n", m, n, m + n);
    printf("%d * %d = %d \n", m, n, m * n);
    return 0;
}
```

转义符 (escape sequence)

\n 回车换行

如何显示 " % 和 \?

\" 显示双引号本身

\% 显示百分号本身

\\ 显示反斜杠本身

# 良好的编程习惯

- ▶ 采用适合计算机的算法
- ▶ 合理组织数据
- ▶ 提高程序的易读性
  - ▶ 注意自定义标识符的命名风格
  - ▶ 注意程序的排版
  - ▶ 为程序书写注释，注释的位置应与被描述的代码相邻，可以放在代码的上方或右方；当代码比较长，特别是有多重嵌套时，应在一些段落的结束处加注释
  - ▶ ...

好的程序：

正确（correct）

可靠（reliable）

高效（efficient）

易读（readable）

可重用（re-usable）

可移植（portable）


.....

- 
- ▶ 良好的书写格式不仅可以使程序美观，还有利于提高程序的可读性，便于程序的调试和维护。
  - ▶ 基本的代码“**书写习惯**”：
    - ▶ 一行只写一个语句 // 不要一行写多个语句
    - ▶ 采用好的缩进模式
      - ▶ 同一块语句前插入等量的空格-用Tab键
      - ▶ 保持“同级”前后缩进一致
    - ▶ 在运算符与操作数之间恰当地添加空格
    - ▶ 在程序段落之间恰当地添加空行
    - ▶ ...
-

# 一个完整的例子 (C++)

- ▶ 例0 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <iostream>
using namespace std;
const double PI = 3.14;
int main( )
{
    int n, d = 1;
    double sum = 0; //
    char ch = 'm';
    cout << "Input n: "
    cin >> n;
    .....
    return 0;
}
```



```
while(d <= n)
{
    sum = sum + PI * d;
    d = d + 1;
}

cout << "The sum is: " << sum;
cout << ch;
```

# 关于自定义标识符命名规则

- ▶ 自定义标识符命名在项目中往往是一个比较难处理的议题，程序员倾向于使用其**个人**的命名约定，而不喜欢别人规定他们如何编写代码。
- ▶ 然而，当代码需要被**团队**内的其他成员阅读时(特别是代码检查的时候)，拥有通用的命名约定是很有价值的，也便于自己日后再阅读自己的代码。
- ▶ 一直以来,最流行的变量命名约定是**匈牙利表示法** (Hungarian Notation)，最初由Microsoft的Charles Simonyi提出，并且在Microsoft内部使用了许多年。  
(这个约定规定了以标准的3或4个字母前缀来表示变量的数据类型，比如表示学生年龄的整型变量就应该命名为intStuAge.)

# 匈牙利命名法：一般C变量

前缀	类型名	意义
i	Integer	整数
l	Long	长整数
lp	Long pointer	长指针
m_	Data member of a class	一个类的数据成员
n	Short int	短整数
p	Pointer	指针
s	String	字符串
u	Unsigned int	无符号整数
ul	Unsigned long (ULONG)	无符号长整数
w	WORD (unsigned short)	无符号短整数
x, y	x, y coordinates	(short) 坐标值/短整数
v	void	空

# 匈牙利命名法：一般C变量

前缀	类型名	意义
a	Array	数组
b	BOOL (int)	布尔(整数)
by	Unsigned Char (Byte)	无符号字符(字节)
c	Char	字符(字节)
cb	Count of bytes	字节数
cr	Color reference value	颜色(参考)值
cx	Count of x (Short)	x的集合(短整数)
dw	DWORD (unsigned long)	双字(无符号长整数)
f	Flags (usually multiple bit values)	标志(一般是有多位的数值)
fn	Function	函数
g_	global	全局的
h	Handle	句柄



# 匈牙利命名法：常用前缀

前缀	类型	描述	例子
ch	char	字符	chGrade
b	BOOL	布尔变量	bEnabled
n	int	整型（其大小由操作系统决定）	nLength
w	WORD	16位无符号整型	wPos
p	*Ambient memory model pointer	内存模块指针，指针变量	pDoc
lp	FAR*	长指针	lpDoc
lpsz	LPSTR	32位字符串指针	lpszName
h	handle	Windows对象句柄	hWnd
lpfn	(*fn)()	回调函数指针 Callback Far pointer to CALLBACK function	lpfnAbort

# 本课程自定义标识符命名具体建议☆

- ▶ **【总则】** 采用一致的、有意义的标识符名字。对不同种类的标识符采用不同风格的名字。
- ▶ **【建议1】** 自定义标识符应直观，用词尽量准确，可望文知意。切忌使用汉语拼音来命名！
- ▶ **【建议2】** 标识符的长度应当符合“**min-length && max-information**”原则。一般来说，长名字能更好地表达含义，但名字并非越长越好，单字符的名字也是有用的，常见的如*i, j, k, m, n, x, y, z*等，它们通常可用作函数内的局部变量。

- 
- ▶ **【建议3】** 程序中不要出现仅靠大小写区分的相似的标识符。例如：

```
int x, y, X; // 变量x 与 X 容易混淆
```

```
void foo(int x); // 函数foo 与FOO容易混淆
```

```
void FOO(int y);
```

- ▶ **【建议4】** 用一对反义词命名具有相反含义的变量或函数等。例如：

```
int minValue, maxValue;
```

```
int SetValue(...), GetValue(...);
```

- 
- ▶ **【建议5】** 函数名和类型名用大写字母开头的单词组合而成。如：

*void Init(void);*

*void SetValue(int value);*

系统定义的类型名、  
main函数名及库函数名  
除外

- ▶ **【建议6】** 变量名和参数名的首单词用小写字母开头，且使用前缀（参考匈牙利命名法）。如：

*int nFlag;*

*int nStuAge;*

*int nCurrent\_value*

- 
- ▶ **【建议7】** 习惯使用符号常量，符号常量名全用大写字母，用下划线分割单词。如：

*#define MAX\_LENGTH 100*

*#define PI 3.14*

# 匈牙利命名法：C++类

---

前缀	类型	例子
g_	全局变量	g_Servers
C	类或者结构体	CDocument, CPrintInfo
m_	成员变量	m_pDoc, m_nCustomers



# 小结

## ► C程序的组成

### C语言基本元素

字符集

单词

表达式

语句

语句块

函数

模块

程序

### C的单词

关键词

标识符

预定义标识符

自定义标识符

字面常量

运算符

标点符号

# 要求：

---

- ▶ 会编写简单的C程序
  - 在main函数中完成变量定义、输入、简单处理、输出
- ▶ 熟练C语言程序的上机步骤
  - ▶ 参照魏老师课程网站：[docs.cpl.icu](http://docs.cpl.icu)
- ▶ C语言的基本词法
- ▶ 按建议规则编程，养成良好的编程习惯
  - ▶ 本课程课件有时没有遵循上述规则，这是为了将相关内容放在一张幻灯片上，便于讲解。



## Q & A

---

