SCHOOL OF ECONOMICS AND MANAGEMENT

Computer Programming Language

Session 2

Data types; Operators; Variables; Input and output

Agenda

- Variables and constants
- Data types
- Operators
- Expressions
- Output functions
- Input functions
- Symbolic constants

Variable

- **Variables** are names given by programmers to computer storage

- Define a variable

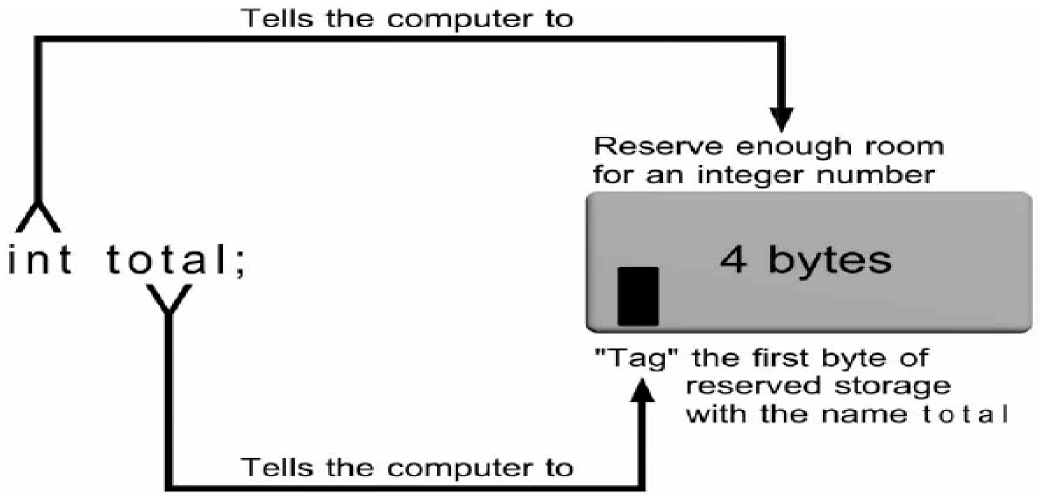

Tells the computer to
Reserve enough room for an integer number
int total;
4 bytes
"Tag" the first byte of reserved storage with the name total
Tells the computer to

Figure 2.11a Defining the integer variable named `total`

Variables are programmer-created identifiers

- **Programmer-created identifiers (标识符):** selected by the programmer
  - Also called **programmer-created names**
  - Used for naming variables and functions
  - Must conform to C's identifier rules
  - Can be any combination of letters, digits, or underscores (_) subject to the following rules:
    - First character must be a letter or underscore (_)
    - Only letters, digits, or underscores may follow the initial character
    - Blank spaces are not allowed
    - *Cannot* be a reserved word

Naming a variable

- Variable (变量) names cannot be **keywords (关键字)**

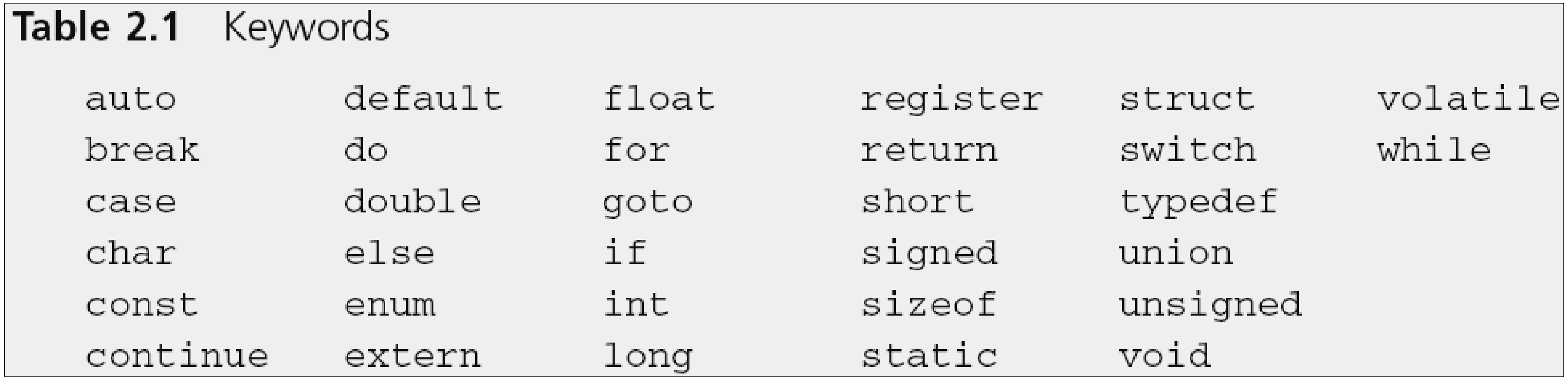


Table 2.1 Keywords

`auto`	`default`	`float`	`register`	`struct`	`volatile`
`break`	`do`	`for`	`return`	`switch`	`while`
`case`	`double`	`goto`	`short`	`typedef`	
`char`	`else`	`if`	`signed`	`union`	
`const`	`enum`	`int`	`sizeof`	`unsigned`	
`continue`	`extern`	`long`	`static`	`void`	

- **Keyword or Reserved word (保留字) :** word that is predefined by the programming language for a special purpose and can only be used in a specified manner for its intended purpose

- Variable names cannot be predefined words – function names

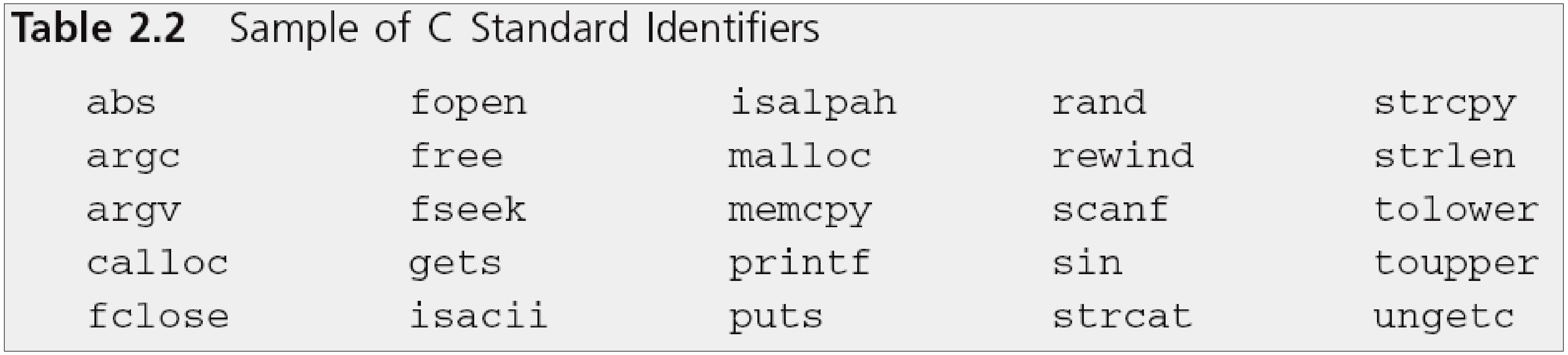


Table 2.2 Sample of C Standard Identifiers

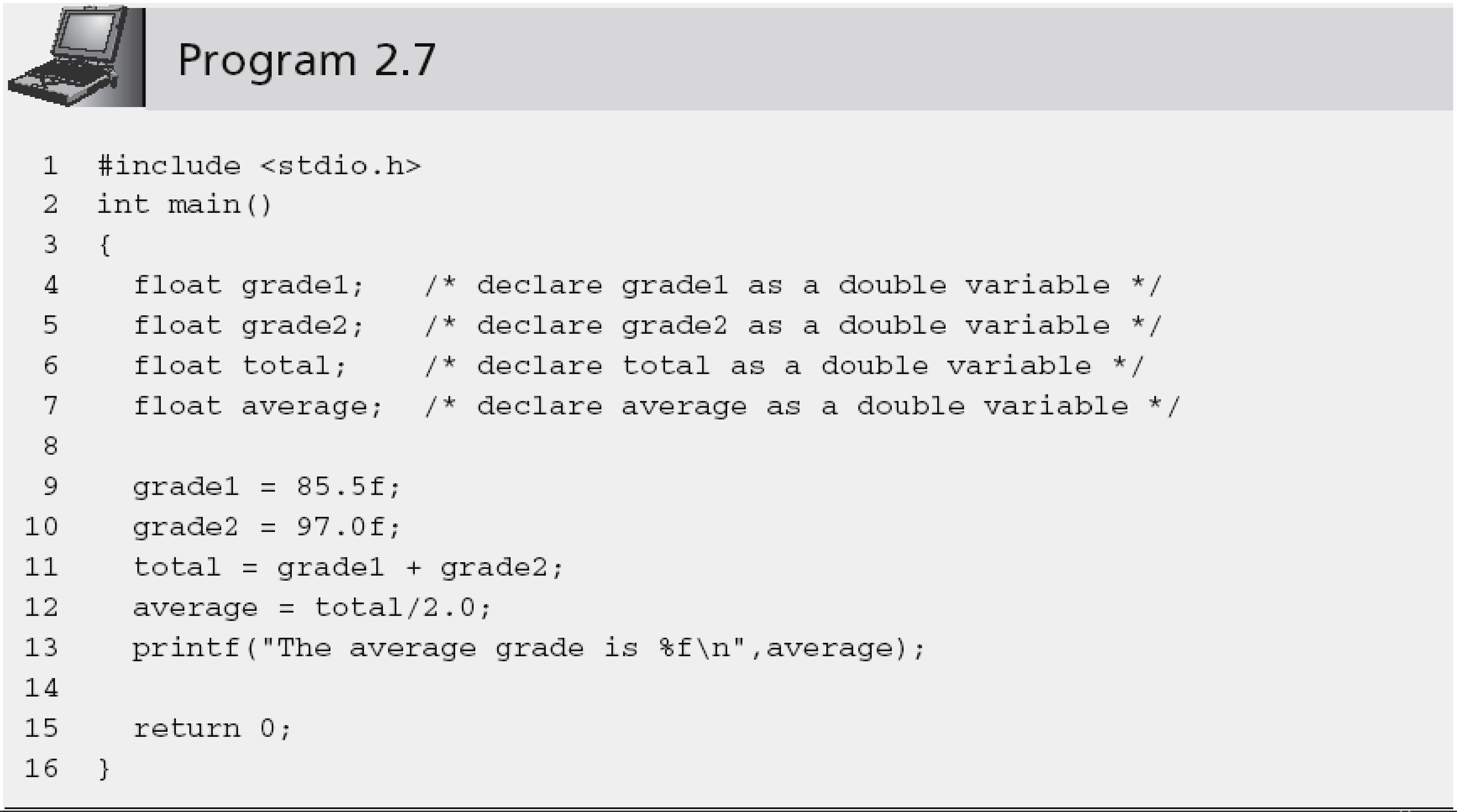
abs	fopen	isalpah	rand	strcpy
argc	free	malloc	rewind	strlen
argv	fseek	memcpy	scanf	tolower
calloc	gets	printf	sin	toupper
fclose	isacii	puts	strcat	ungetc

Examples

- Examples of ***invalid*** C programmer-created names:
  - `4ab7`
  - `calculate total`
  - `while`
- Style of naming identifiers
  - All uppercase letters used to indicate a constant
  - An identifier should be descriptive: `degToRadians()`
    - Bad identifier choices: `easy`, `duh`, `justDoIt`
- C is a case-sensitive language
  - `TOTAL`, and `total` represent different identifiers

Initialization (初始化) with a constant

- Declare and initialize a variable.
  - `int numOne = 15;`


Program 2.7

```
#include <stdio.h>
int main()
{
  float grade1;    /* declare grade1 as a double variable */
  float grade2;    /* declare grade2 as a double variable */
  float total;     /* declare total as a double variable */
  float average;   /* declare average as a double variable */

  grade1 = 85.5f;
  grade2 = 97.0f;
  total = grade1 + grade2;
  average = total/2.0;
  printf("The average grade is %f\n",average);

  return 0;
}
```

Integer

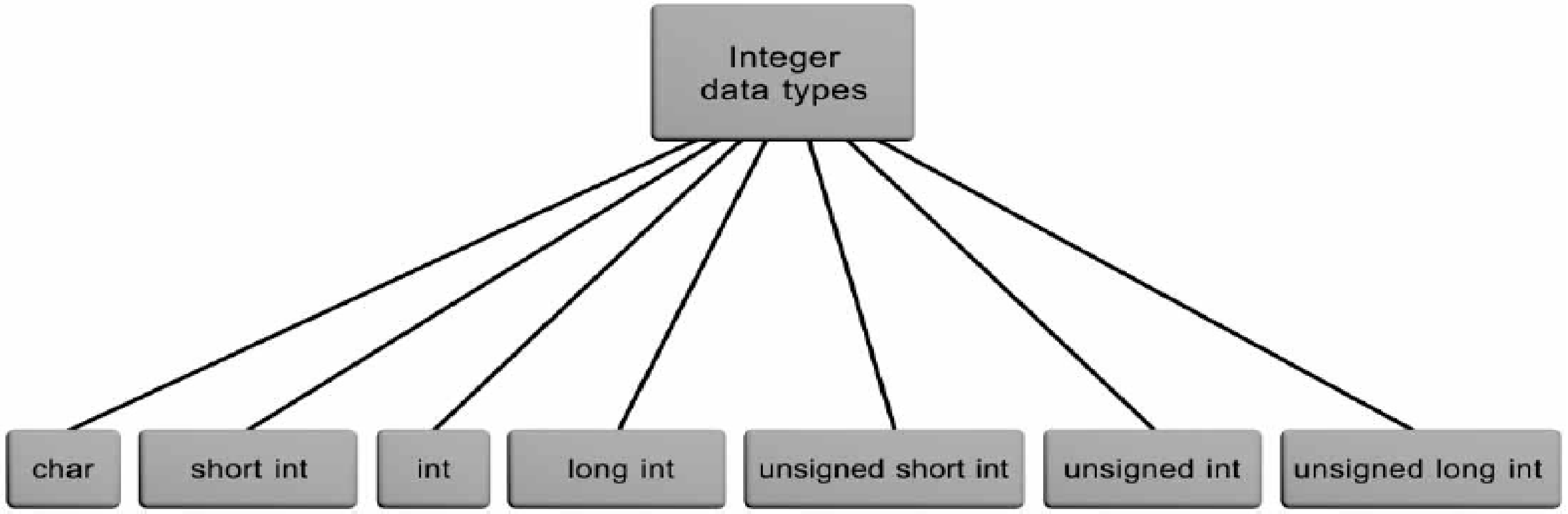


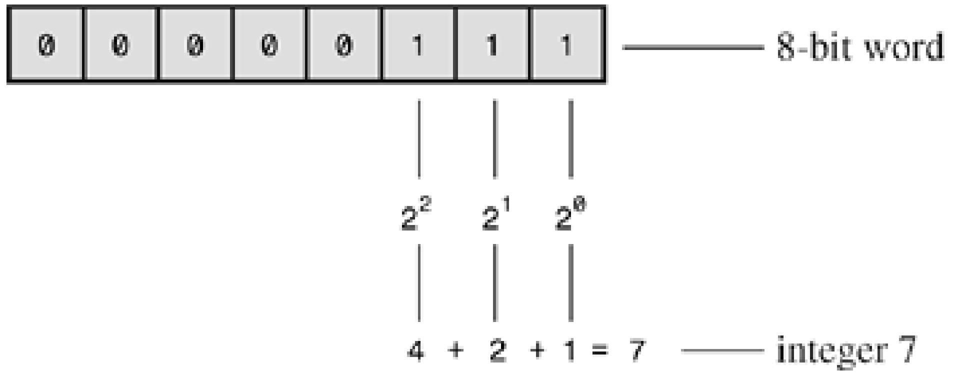
Figure 2.7 C's integer data types

Table 3.3. Integer Type Sizes (Bits) for Representative Systems

Type	Macintosh Metrowerks CW (Default)	Linux on a PC	IBM PC Windows XP and Windows NT	ANSI C Minimum
`char`	8	8	8	8
`int`	32	32	32	16
`short`	16	16	16	16
`long`	32	32	32	32
`long long`	64	64	64	64

Storing an integer

Figure 3.2. Storing the integer 7 using a binary code.



The MAX of a signed integer type: $2^{bits-1}-1$

The MIN of a signed integer type: -2^{bits-1}

The MAX of an unsigned integer type: $2^{bits}-1$

The MIN of an unsigned integer type: 0

printf

Table 2.8 Conversion Control Sequences

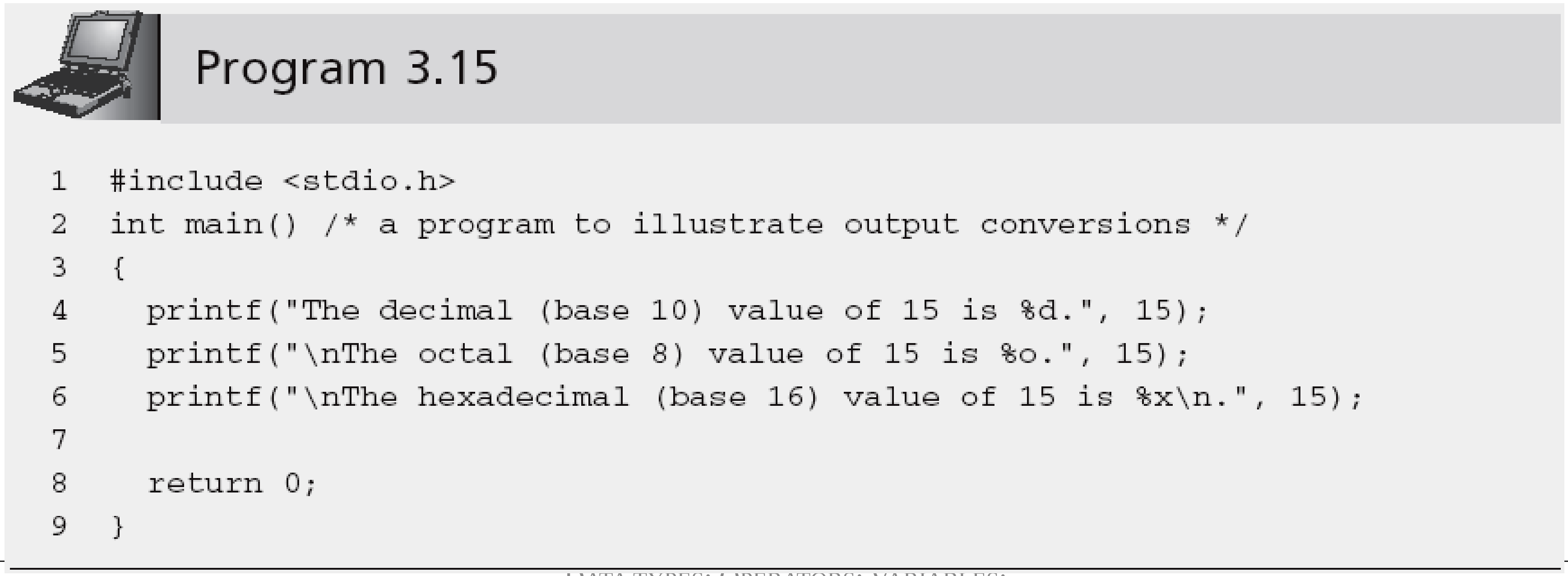
Sequence	Meaning
%d	Display an integer as a decimal (base 10) number
%c	Display a character
%f	Display the floating-point number as a decimal number with six digits after the decimal point (pad with zeros, if necessary)

```
/* print_lld.c
%lld  -- display a long long int */

#include <stdio.h>
int main()
{
        long long int x=1234567890123456789;
        printf("The value of X is %lld", x);
}
```

Other number bases

Octal number: 0-7
Hexadecimal number: 0-f

```
// base.c
#include <stdio.h>
int main()
{               printf("%d  %d  %d", 13, 013, 0x13);
}
```


Program 3.15

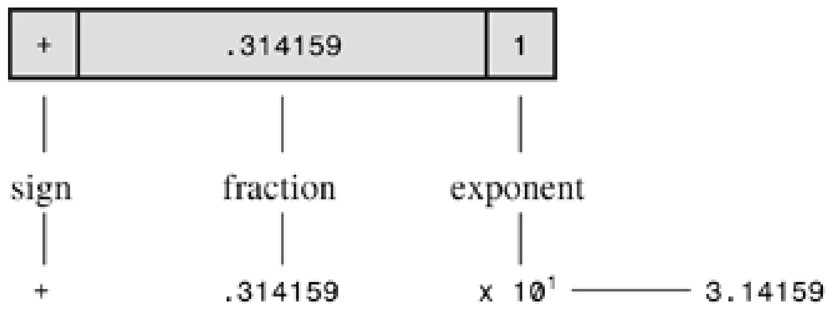
```
#include <stdio.h>
int main() /* a program to illustrate output conversions */
{
  printf("The decimal (base 10) value of 15 is %d.", 15);
  printf("\nThe octal (base 8) value of 15 is %o.", 15);
  printf("\nThe hexadecimal (base 16) value of 15 is %x\n.", 15);

  return 0;
}
```

Floating-point (浮点型)

- A **floating-point value (real number)** contains a decimal point
  - For example: `+10.625`, `5.`, `-6.2`, `3251.92`, `20.5e3`
- `float`: **single-precision** number
- `double`: **double-precision** number
- `float` literal is indicated by appending an `f` or `F`
- `long double` is created by appending an `l` or `L`
  - `9.234` indicates a `double` literal
  - `9.234f` indicates a `float` literal
  - `9.234L` indicates a `long double` literal
- Storage allocation for each data type depends on the compiler

Storing a floating-point number

Figure 3.3. Storing the number pi in floating-point format (decimal version).



	Number	
Keyword	**of Bytes**	**Range of Values**
float	4	±3.4E38 (7 decimal digits precision)
double	8	±1.7E308 (15 decimal digits precision)
long double	10	±1.2E4932 (19 decimal digits precision)

Formatted output

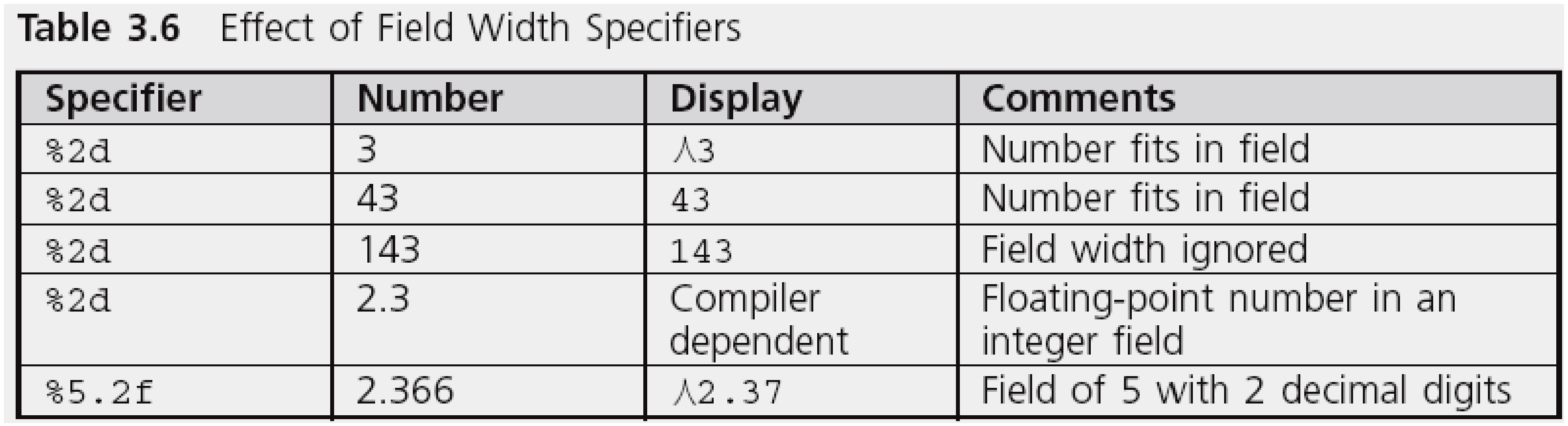


Table 3.6 Effect of Field Width Specifiers

Specifier	Number	Display	Comments
`%2d`	3	`⋏3`	Number fits in field
`%2d`	43	`43`	Number fits in field
`%2d`	143	`143`	Field width ignored
`%2d`	2.3	Compiler dependent	Floating-point number in an integer field
`%5.2f`	2.366	`⋏2.37`	Field of 5 with 2 decimal digits

- For detailed usage of printf(), check http://www.cplusplus.com/reference/clibrary/cstdio/printf.html

Print a floating-point number

```
/* print_flt.c
%f  -- display floating-point number */

#include <stdio.h>
int main()
{
       double x=1.2534;
       printf("The value of X is %f\n", x);
       //It is OK to use %f to output double
       printf("The value of X is %.3f\n", x);
       printf("The value of X is %.1f\n", x);
       //The output is automatically rounded.

       float f=1234567.890;
       printf("The value of f is %f\n", f);
       //note the significant number
}
```

Char

- **`char`:** stores individual characters (ASCII)
  - For example: `'A'`, `'$'`, `'b'`, `'!'`

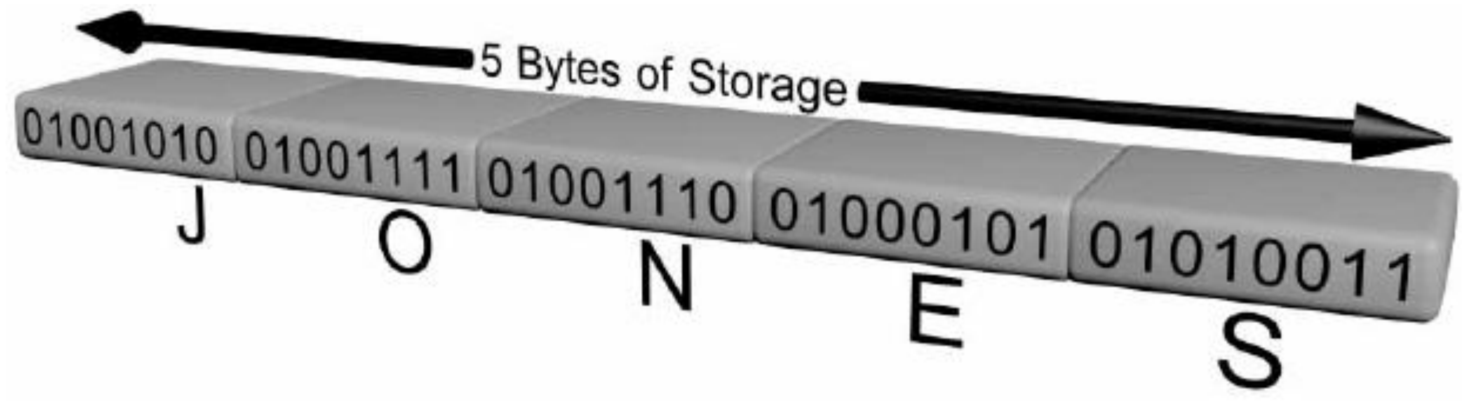
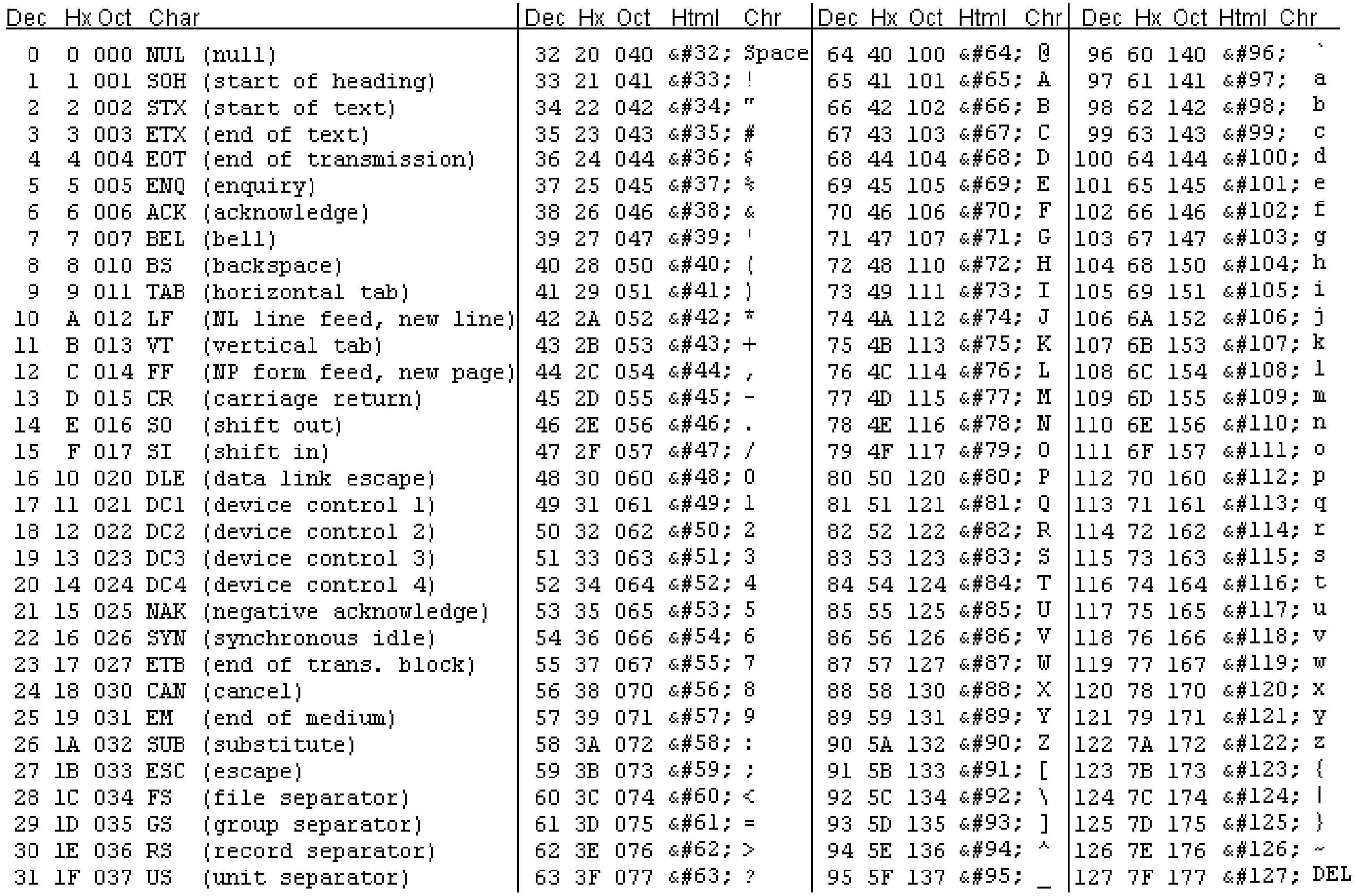


Figure 2.8 The letters JONES stored inside a computer

ASCII Table



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

printf

Table 2.8 Conversion Control Sequences

Sequence	Meaning
`%d`	Display an integer as a decimal (base 10) number
`%c`	Display a character
`%f`	Display the floating-point number as a decimal number with six digits after the decimal point (pad with zeros, if necessary)

Program 2.6

```c
#include <stdio.h>
int main()
{
  printf("\nThe first letter of the alphabet is %c", 'a');
  printf("\nThe decimal code for this letter is %d", 'a');
  printf("\nThe code for an uppercase %c is %d\n", 'A', 'A');

   return 0;
}
```

Program 3.17

```
#include <stdio.h>
int main()
{
  printf("The decimal value of the letter %c is %d.", 'a', 'a');
  printf("\nThe octal value of the letter %c is %o.", 'a', 'a');
  printf("\nThe hex value of the letter %c is %x.\n", 'a', 'a');

  return 0;
}
```

```
The decimal value of the letter a is 97.
The octal value of the letter a is 141.
The hex value of the letter a is 61.
```

Escape Sequence (转义字符）

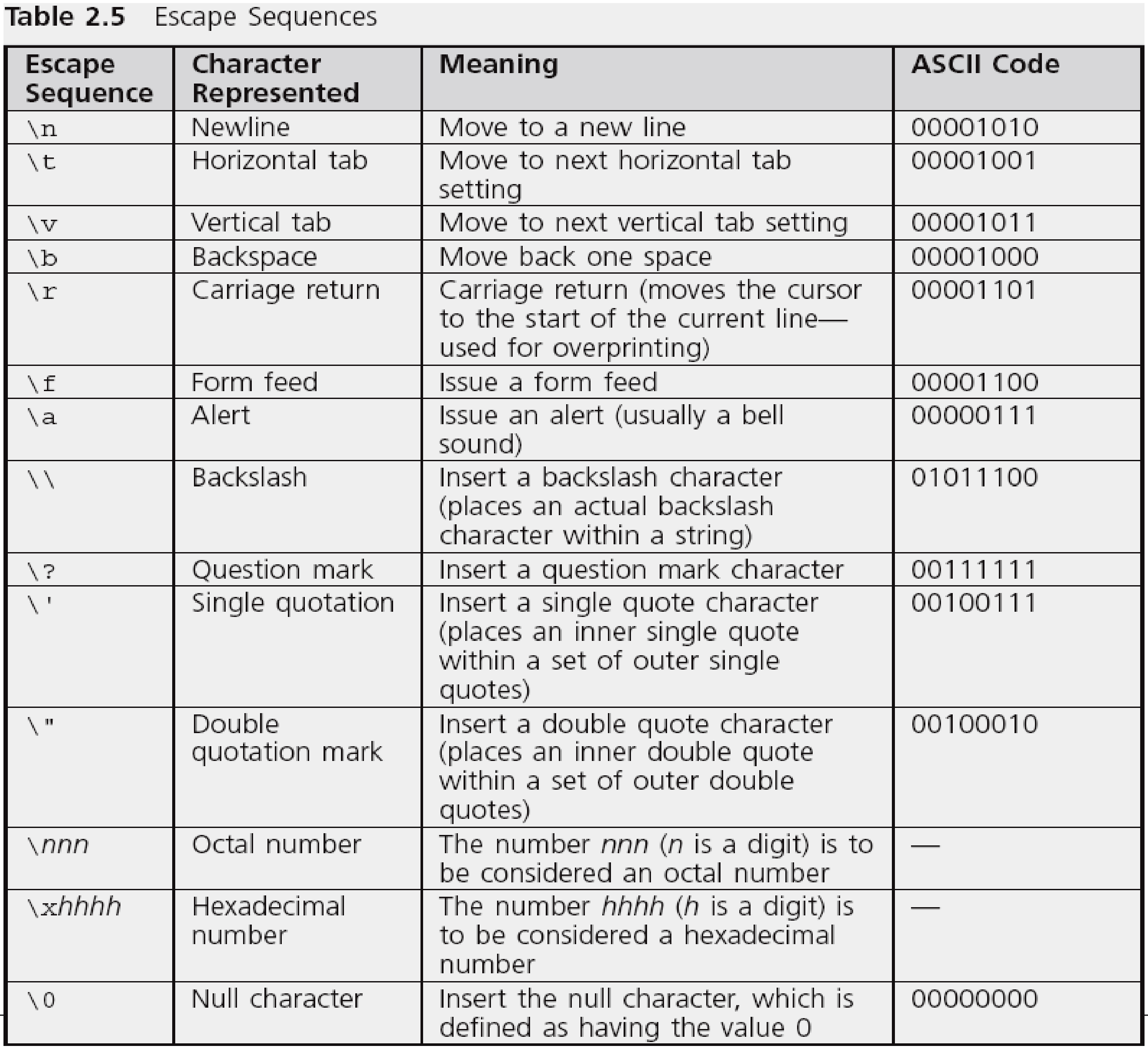


Table 2.5 Escape Sequences

Escape Sequence	Character Represented	Meaning	ASCII Code
`\n`	Newline	Move to a new line	00001010
`\t`	Horizontal tab	Move to next horizontal tab setting	00001001
`\v`	Vertical tab	Move to next vertical tab setting	00001011
`\b`	Backspace	Move back one space	00001000
`\r`	Carriage return	Carriage return (moves the cursor to the start of the current line—used for overprinting)	00001101
`\f`	Form feed	Issue a form feed	00001100
`\a`	Alert	Issue an alert (usually a bell sound)	00000111
`\\`	Backslash	Insert a backslash character (places an actual backslash character within a string)	01011100
`\?`	Question mark	Insert a question mark character	00111111
`\'`	Single quotation	Insert a single quote character (places an inner single quote within a set of outer single quotes)	00100111
`\"`	Double quotation mark	Insert a double quote character (places an inner double quote within a set of outer double quotes)	00100010
`\`*nnn*	Octal number	The number *nnn* (*n* is a digit) is to be considered an octal number	—
`\x`*hhhh*	Hexadecimal number	The number *hhhh* (*h* is a digit) is to be considered a hexadecimal number	—
`\0`	Null character	Insert the null character, which is defined as having the value 0	00000000

```
/* beep.c */
#include <stdio.h>

int main()
{
   printf("I'm here!\n");
   printf("\a");
   printf("So am I!\n");

  char c='\x61';
  printf ("c is %c", c);
}
```

sizeof()

```
/* typesize.c -- prints out type sizes */

#include <stdio.h>
int main()
{
    printf("Type char has a size of %u bytes.\n", sizeof(char));
    printf("Type int has a size of %u bytes.\n", sizeof(int));
    printf("Type short has a size of %u bytes.\n", sizeof(short));
    printf("Type long has a size of %u bytes.\n", sizeof(long));
    printf("Type long long has a size of %u bytes.\n", sizeof(long
long));
return 0;
}
```

Operator (运算符) and Expression (表达式）

+, - , *, /, %

Precedence

    How to calculate

    78-7*(8 % 98)-4*5

Precision:

    If both operands are integers, result is an integer

    If one operand is floating-point, result is double-precision

Integer Division

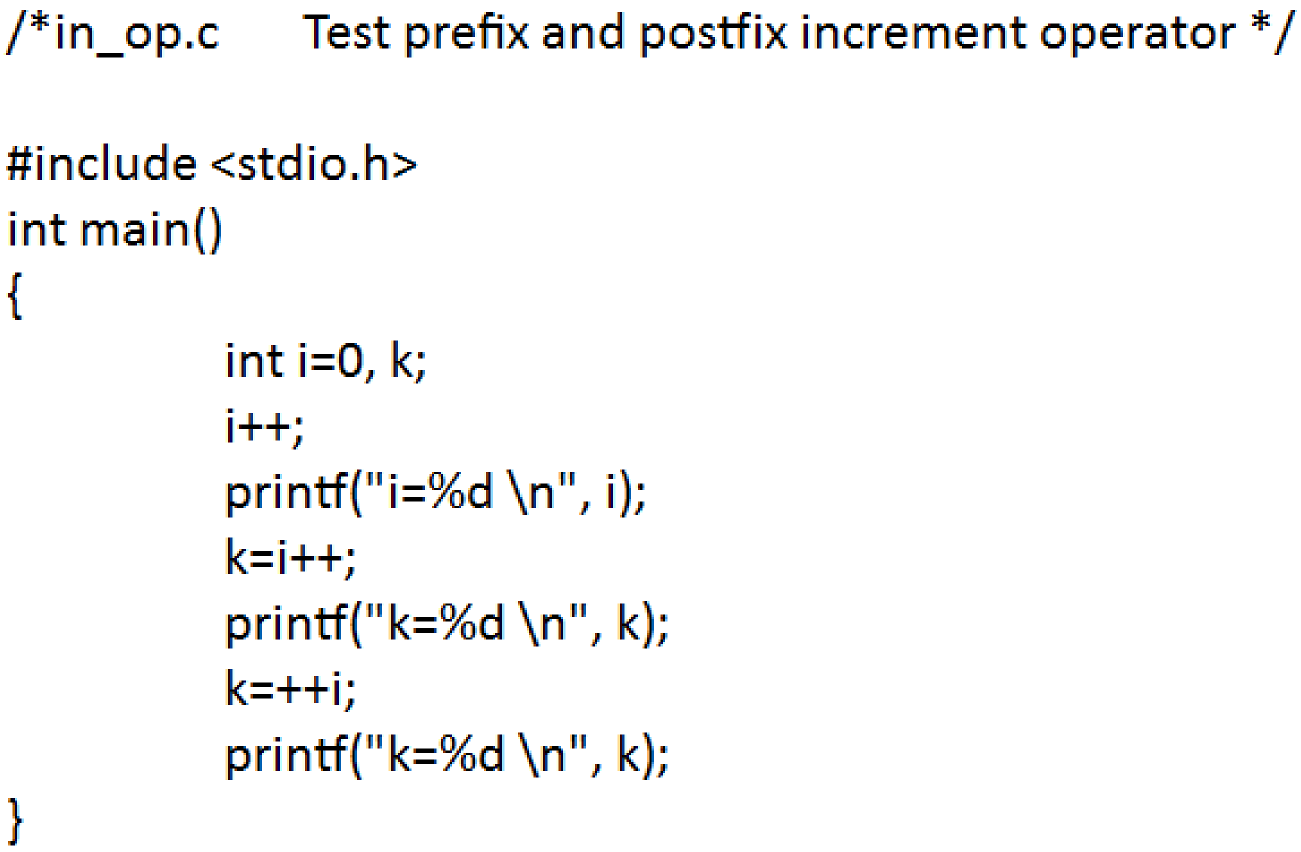
- `15/2 = 7`
  - Integers cannot contain a fractional part
  - Remainder is truncated

- % is the **modulus** or **remainder operator**
  - `9 % 4 is 1`
  - `17 % 3 is 2`
  - `14 % 2 is 0`

Variations of assignments

- Multiple assignments
  - `a = b = c = 25;`
- Various assignment operators: `+=  -=  *=  /=  %=`
  - `sum = sum + 10` can be written as `sum += 10`
  - `price *= rate` is equivalent to `price = price * rate`
  - `price *= rate + 1` is equivalent to `price = price * (rate + 1)`
- Increment operator ++
  - Prefix: `k = ++n;`
  - Postfix: `k = n++;`
- Decrement operator --

Output?



```
/*in_op.c      Test prefix and postfix increment operator */

#include <stdio.h>
int main()
{
                int i=0, k;
                i++;
                printf("i=%d \n", i);
                k=i++;
                printf("k=%d \n", k);
                k=++i;
                printf("k=%d \n", k);
}
```

Type conversion

- Implicit type conversion

  -
    ```
    double result;
    result = 4; //integer 4 is converted to 4.0
    ```

  -
    ```
    int answer;
    answer = 2.764; //2.764 is converted to 2
    ```
  - Be careful of what you will get
    - `float f;  f=999999999;`

- Explicit type conversion

  -
    ```
    double result=9.1;
    int k= (int) result;
    k = (int) 8.8;
    result = 7+(double) (7)/9 ;
    ```

Example: Temperature Conversion

The algorithm: $\text{Celsius degree} = \frac{5}{9}(\text{Fahrenheit degree} - 32)$

Program 2.9

```
/* convert a Fahrenheit temperature to Celsius */

#include <stdio.h>
int main()
{
  float celsius;
  float fahrenheit = 75;   /* declaration and initialization */

  celsius = 5.0/9.0 * (fahrenheit - 32.0);
  printf("The Celsius equivalent of %5.2f degrees Fahrenheit\n",
                                                      fahrenheit);
  printf("   is %5.2f degrees\n", celsius);

  return 0;
}
```

Input

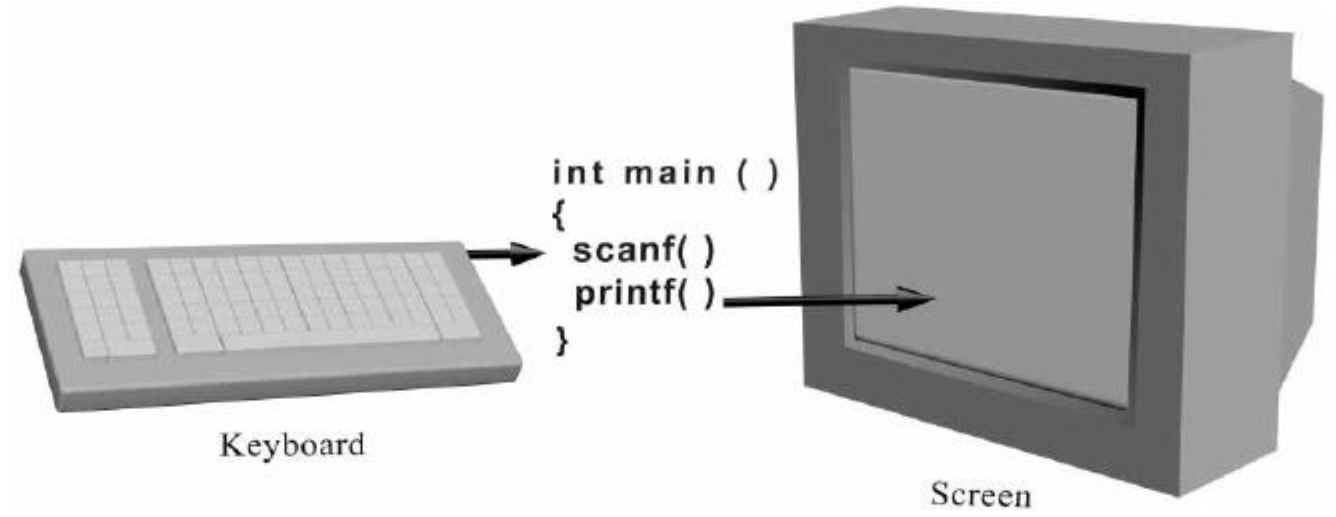


Figure 3.5 `scanf()` used to enter data; `printf()` used to display data

Scanf()

Program 3.9

```
#include <stdio.h>
int main()
{
  float num1, num2, product;

  printf("Please type in a number: ");
  scanf("%f",&num1);
  printf("Please type in another number: ");
  scanf("%f",&num2);
  product = num1 * num2;
  printf("%f times %f is %f\n", num1, num2, product);

  return 0;
}
```

This statement produces a **prompt**

Address operator (&)

Input format

- `scanf()` can be used to enter many values

  `scanf("%f %f",&num1,&num2); //"%f%f"` is the same
- A space can affect what the value being entered when `scanf()` is expecting a character data type
  - `scanf("%c%c%c",&ch1,&ch2,&ch3);` stores the next three characters typed in the variables `ch1`, `ch2`, and `ch3`; if you type `x y z`, then `x` is stored in `ch1`, a blank is stored in `ch2`, and `y` is stored in `ch3`
  - `scanf("%c %c %c",&ch1,&ch2,&ch3);` causes `scanf`() to look for three characters, each character separated by exactly one space

Note

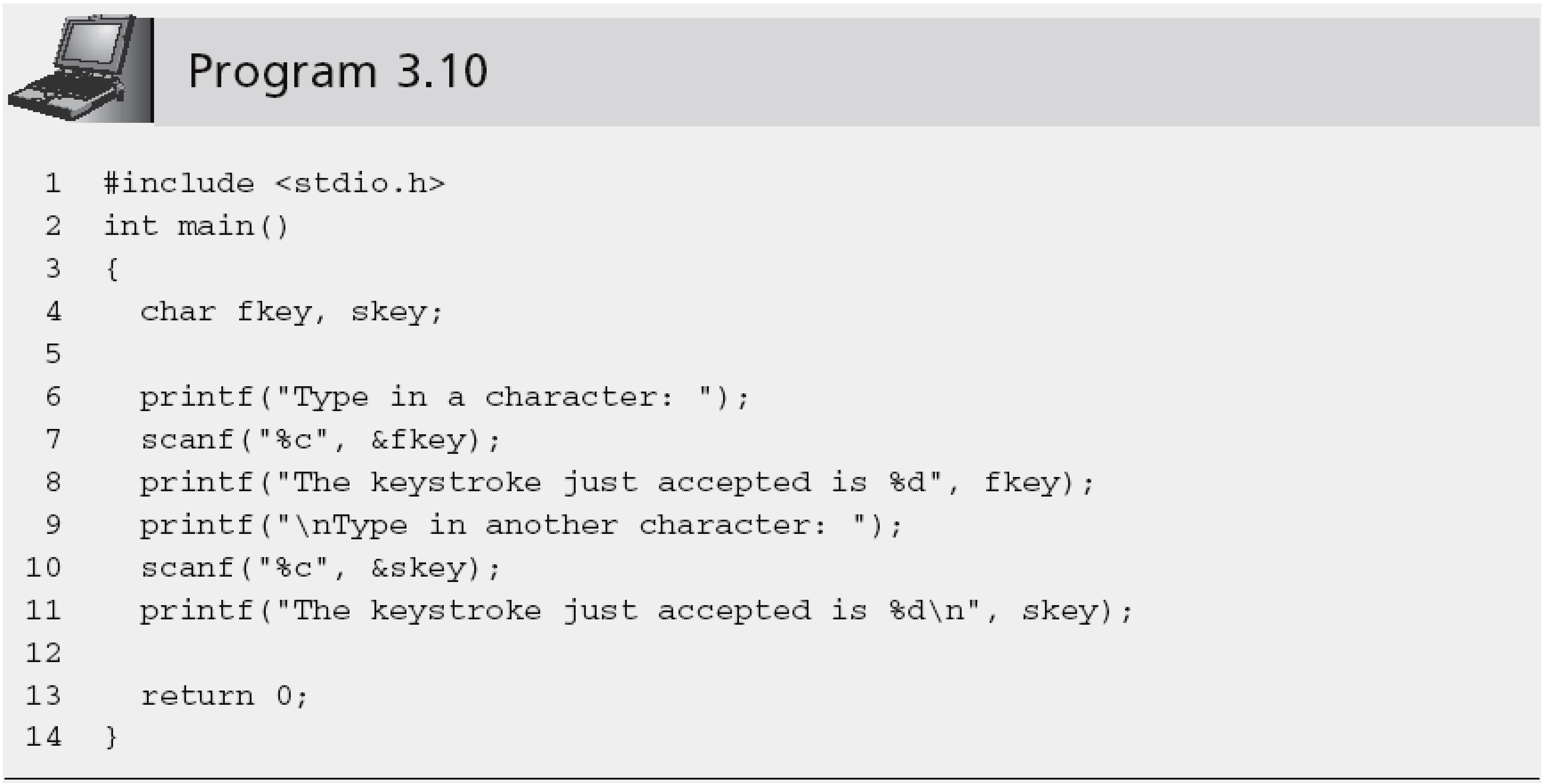
- In printing a double-precision number using `printf()`, the conversion control sequence for a single-precision variable, `%f`, can be used.

- When using `scanf()`, if a double-precision number is to be entered, you must use the `%lf` conversion control sequence.

- When input a long long int, use `%lld`

- `scanf()` does not test the data type of the values being entered
  - In `scanf("%d %f", &num1, &num2)`, if user enters `22.87`, `22` is stored in `num1` and `.87` in `num2`

Caution: The Phantom Newline Character



Program 3.10

```
#include <stdio.h>
int main()
{
  char fkey, skey;

  printf("Type in a character: ");
  scanf("%c", &fkey);
  printf("The keystroke just accepted is %d", fkey);
  printf("\nType in another character: ");
  scanf("%c", &skey);
  printf("The keystroke just accepted is %d\n", skey);

  return 0;
}
```

Input buffer

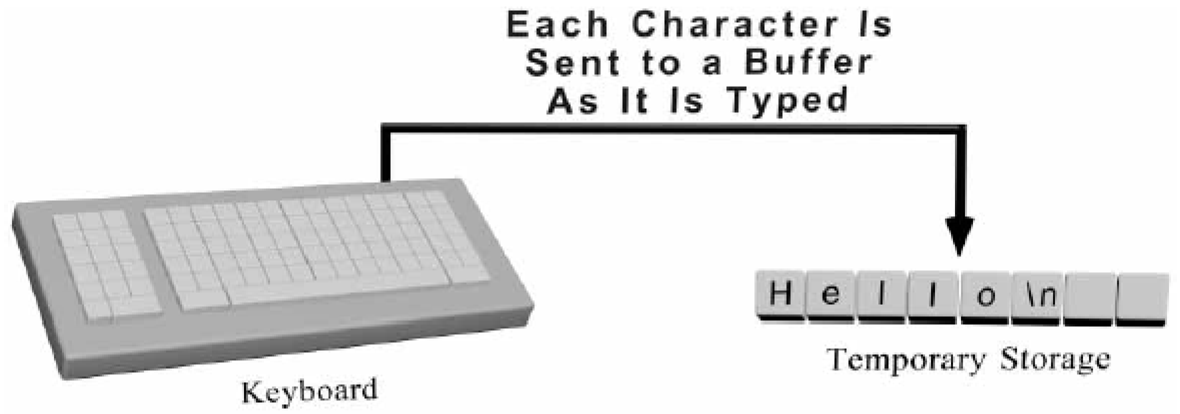


Figure 3.6 Typed keyboard characters are first stored in a buffer

Catch the newline character

Program 3.11

```c
#include <stdio.h>
int main()
{
  char fkey, skey;

  printf("Type in a character: ");
  scanf("%c%c", &fkey, &skey); /* the enter code goes to skey */
  printf("The keystroke just accepted is %d", fkey);
  printf("\nType in another character: ");
  scanf("%c", &skey); /* accept another code */
  printf("The keystroke just accepted is %d\n", skey);

  return 0;
}
```

Flush the input buffer – fflush(stdin)

```
/*Program 3.11b */

#include <stdio.h>
int main()
{
		char fkey, skey;

		printf("Type in a character: ");
		scanf("%c", &fkey);
		printf("The keystroke just accepted is %d", fkey);
		fflush(stdin);
		printf("\nType in another character: ");
		scanf("%c", &skey);
		printf("The keystroke just accepted is %d", skey);

		return 0;
}
```

Other input functions - getchar

- Read a single character from the keyboard input
- Wait for enter key to proceed
- int getchar(): returning type is int

```
/*test_getchar.c */
#include <stdio.h>

int main()
{    char c,d ;
     c = getchar();
     printf("c = %c\n", c);

     getchar();
     d = getchar();
     printf("ASCII of %c = %d", d, d);
}
```

getch()

- Read a character and do **NOT** wait for Enter
- Never echo the character on screen
- Non-standard, but supported and used widely
- int getch(): returning type is int
- #include <conio.h>

```
/*test_getch.c*/
#include <stdio.h>
#include <conio.h>

int main()
{    char c,d ;
     c = getch();
     printf("c = %c\n", c);

     d = getch();
     printf("ASCII of %c = %d", d, d);
}
```

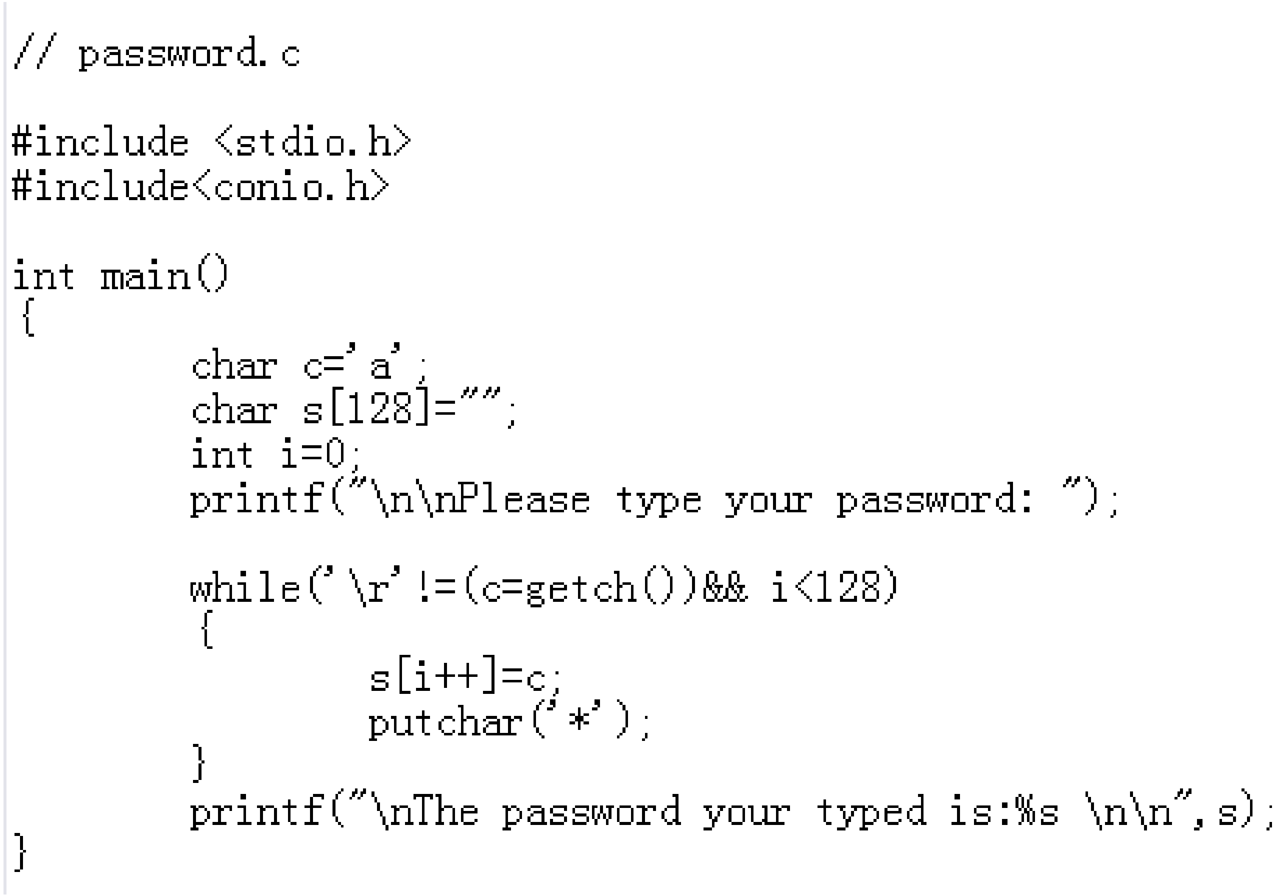
Press Enter key – what character received?

```
/*getch_getchar.c*/
#include <stdio.h>
#include <conio.h>
int main()
{
        int d;
        printf("Please press Enter key:");
        d=getchar();
        printf("getchar catches %d\n", d);
        printf("Please press Enter key:");
        d=getch();
        printf("getch catches %d", d);
}
```

'\n'	10	new line
'\r'	13	carriage return

Example: mask the password you input

- putchar(int c) – print one character



```
// password.c

#include <stdio.h>
#include<conio.h>

int main()
{
        char c=’a’;
        char s[128]=“”;
        int i=0;
        printf(“\n\nPlease type your password: ”);

        while(’\r’!=(c=getch())&& i<128)
        {
                s[i++]=c;
                putchar(’*’);
        }
        printf(“\nThe password your typed is:%s \n\n”,s);
}
```

Defensive programming

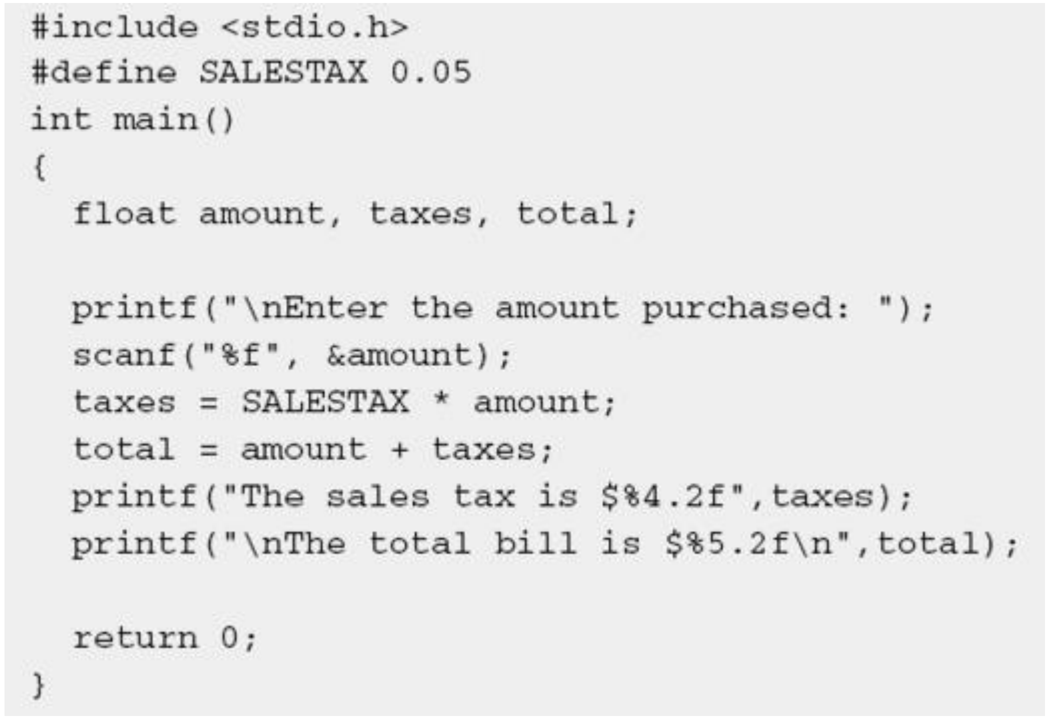
- Users may behave unexpectedly

Program 3.12

```c
#include <stdio.h>
int main()
{
  int num1, num2, num3;
  double average;

  /* get the input data */
  printf("Enter three integer numbers: ");
  scanf("%d %d %d", &num1, &num2, &num3);

  /* calculate the average*/
  average = (num1 + num2 + num3) / 3.0;

  /* display the result */
  printf("\nThe avearge of %d, %d, and %d is %f\n",
                        num1, num2, num3, average);


  return 0;
}
```

Symbolic Constants (符号常量)

- Give a constant a **symbolic name**
  - `#define SALESTAX 0.05`
  - `#define PI 3.1416`
  - Also called **symbolic constants** and **named constants**



```
#include <stdio.h>
#define SALESTAX 0.05
int main()
{
  float amount, taxes, total;

  printf("\nEnter the amount purchased: ");
  scanf("%f", &amount);
  taxes = SALESTAX * amount;
  total = amount + taxes;
  printf("The sales tax is $%4.2f",taxes);
  printf("\nThe total bill is $%5.2f\n",total);

  return 0;
}
```