C Programming #2

Processing and Interactive Input

International College, KMITL

Assignment

- The general syntax for an assignment statement is

  variable = operand;

  - The operand to the right of the assignment operator (=) can be a constant, a variable, or an expression
- Multiple assignments are possible in the same statement

  a = b = c = 25;

  - All = operators have the same precedence
  - Operator has right-to-left associativity

    c = 25;

    b = c;

    a = b;

Implicit Type Conversions

- Data type conversions take place across assignment operators

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

- The automatic conversion across an operator is called an implicit type conversion

```
int answer;
answer = 2.764; //2.764 is converted to 2
```

  - Here the implicit conversion is from a higher precision to a lower precision data type; the compiler will issue a warning

Explicit Type Conversions (Casts)

- The operator used to force the conversion of a value to another type is the cast operator

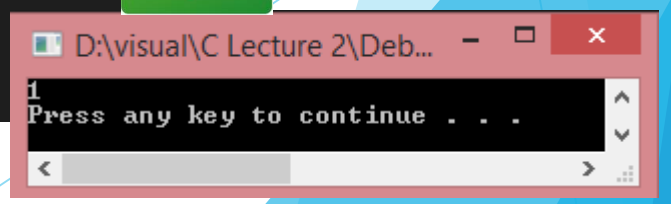
  (dataType) expression

  - where dataType is the desired data type of the expression following the cast
- Example:
  - If sum is declared as double sum;, (int) sum is the integer value determined by truncating sum's fractional part

```
#include <stdio.h>

void main()
{
    double n = 1.354234;
    printf("%d\n", n);

    system("pause");
}
```

garbage value

```
#include <stdio.h>

void main()
{
    double n = 1.354234;
    printf("%d\n", (int)n);

    system("pause");
}
```



Assignment Variations

- Assignment expressions like sum = sum + 25 can be written using the following 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)

Counting

- A counting statement is very similar to the accumulating statement

  variable = variable + fixedNumber;

- Examples: i = i + 1; and m = m + 2
- Increment operator (++): variable = variable + 1 can be replaced by variable++ or ++variable

```c
#include < stdio.h >

void main()
{
    int sum = 0;

    printf("sum = %d\n", sum);
    sum += 1;
    printf("sum = %d\n", sum);

    system("pause");
}
```

D:\visual\C Lecture 2\Debug\C ...

```
sum = 0
sum = 1
Press any key to continue . . .
```

```c
#include < stdio.h >

void main()
{
    int sum = 0;

    printf("sum = %d\n", sum);
    sum++;
    printf("sum = %d\n", sum);

    system("pause");
}
```

D:\visual\C Lecture 2\Debug\C ...

```
sum = 0
sum = 1
Press any key to continue . . .
```

Counting (continued)

- When the ++ operator appears before a variable, it is called a prefix increment operator; when it appears after a variable, it is called postfix increment operator
  - k = ++n; is equivalent to
    - n = n + 1; // increment n first
    - k = n; // assign n's value to k
  - k = n++; is equivalent to
    - k = n; // assign n's value to k
    - n = n + 1; // and then increment n

Counting (continued)

Expression	Alternative
`i = i + 1`	`i++ and ++i`
`n = n + 1`	`n++ and ++n`
`count = count + 1`	`count++ and ++count`

Example of the Increment Operator

Expression	Alternative
`i = i - 1`	`i-- and --i`
`n = n - 1`	`n-- and --n`
`count = count - 1`	`count-- and --count`

Example of the Decrement Operator

Mathematical Library Functions

Function	Description	Example	Returned Value	Comments
sqrt(x)	Square root of x	sqrt(16.00)	4.000000	an integer value of x results in a compiler error
pow(x,y)	x raised to the y power (x^y)	pow(2, 3) pow(81, .5)	8.000000 9.000000	integer values of x and y are permitted
exp(x)	e raised to the x power (e^x)	exp(-3.2)	0.040762	an integer value of x results in a compiler error
log(x)	Natural log of x (base e)	log(18.697)	2.928363	an integer value of x results in a compiler error
log10(x)	Common log of x (base 10)	log10(18.697)	1.271772	an integer value of x results in a compiler error
fabs(x)	Absolute value of x	fabs(-3.5)	3.5000000	an integer value of x results in a compiler error
abs(x)	Absolute value of x	abs(-2)	2	a floating-point value of x returns a Value of 0

#include <math.h>

Check Point #1

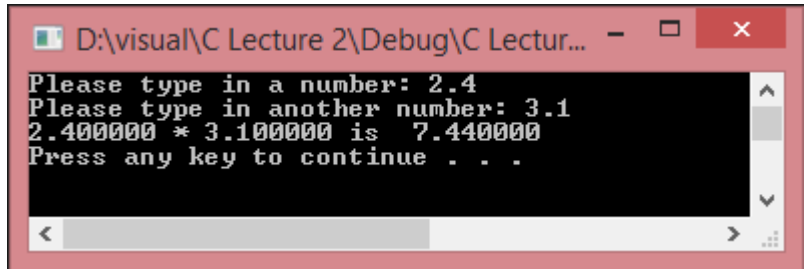
Interactive Input

- scanf() is used to enter data into a program while it is executing; the value is stored in a variable
  - It requires a control string as the first argument inside the function name parentheses
- The control string passed to scanf() typically consists of conversion control sequences only
- scanf() requires that a list of variable addresses follow the control string
  - scanf("%d", &num1);
- #define _CRT_SECURE_NO_WARNINGS is required for using scanf()
- scanf() can be used to enter many values

```
scanf("%f %f",&num1,&num2); //"%f%f" is the same
```

Interactive Input (continued)

```c
#define _CRT_SECURE_NO_WARNINGS
#include < stdio.h >

void 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 * %f is  %f\n", num1, num2, product);

    system("pause");
}
```



Check Point #2

Formatted Output

```c
#include < stdio.h >

void main()
{
    printf("%d\n", 6);
    printf("%d\n", 18);
    printf("%d\n", 224);
    printf("---\n");
    printf("%d\n", 6 + 18 + 124);

    system("pause");
}
```

```c
#include < stdio.h >

void main()
{
    printf("%3d\n", 6);
    printf("%3d\n", 18);
    printf("%3d\n", 224);
    printf("---\n");
    printf("%3d\n", 6 + 18 + 124);

    system("pause");
}
```

D:\visual\C Lecture 2\Debug\C Lec...

```
6
18
224
---
148
Press any key to continue . . .
```

D:\visual\C Lecture 2\Deb...

```
  6
 18
224
---
148
Press any key to continue . . .
```

Formatted Output (continued)

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
%5.2f	42.3	42.30	Number fits in field
%5.2f	142.364	142.36	Field width ignored but fractional specifier is used
%5.2f	142	Compiler dependent	Integer in a floating-point field

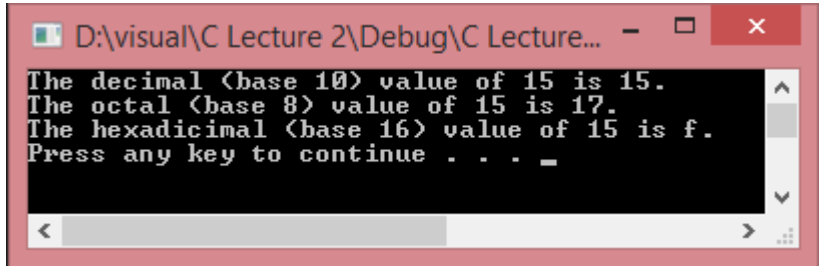
Effect of Field Width Specifiers

Format Modifiers

- Left justification: printf("%-10d",59); produces the display 59^^^^^^^^
- Explicit sign display: printf("%+10d",59); produces the display ^^^^^^^+59
- Format modifiers may be combined
  - %-+10d would cause an integer number to both display its sign and be left-justified in a field width of 10 spaces
    - The order of the format modifiers is not critical %+-10d is the same

Other Number Bases

```c
#include < stdio.h >

void main()
{
    printf("The decimal (base 10) value of 15 is %d.\n", 15);
    printf("The octal (base 8) value of 15 is %o.\n", 15);
    printf("The hexadicimal (base 16) value of 15 is %x.\n", 15);

    system("pause");
}
```

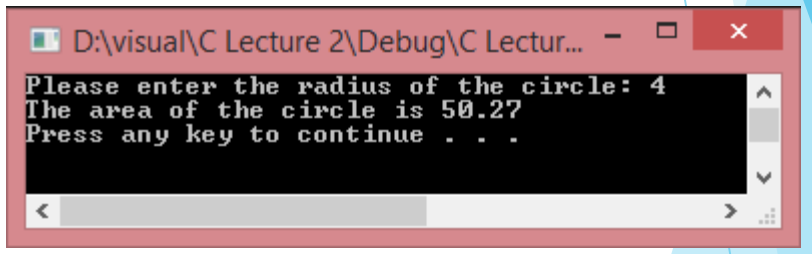


Symbolic Constants

- Literal data refers to any data within a program that explicitly identifies itself
- Literal values that appear many times in the same program are called magic numbers
- C allows you to define the value once by the number to a symbolic name
  - #define SALESTAX 0.05
  - #define PI 3.1416
  - Also called symbolic constants and named constants

Symbolic Constants (continued)

```c
#define _CRT_SECURE_NO_WARNINGS
#include <  stdio.h  >
#include <math.h>
#define PI 3.1416

void main()
{
    float radius;
    printf("Please enter the radius of the circle: ");
    scanf("%f", &radius);
    printf("The area of the circle is %.2f\n", PI * pow(radius, 2));

    system("pause");
}
```



Check Point #3

References

- A First Book of ANSI C, Fourth Edition