Formatting Output of a Program (int)

- When displaying an int value, place a number between the % and d which will specify the number of columns to use for displaying the int value (such as %5d).

```
int x = 2345, y=123;
printf("%d\n",x); //Usual

printf("%6d\n",x); //Display using 6 columns

printf("%6d\n",y); //Note: Right aligned

printf("%2d\n",x); //Less columns, same as %d
```

Output

```
2345
  2345
   123
2345
```

Formatting Output of a Program (float)

- Format placeholder id is %n.mf where
  - n is the total field width (both before and after the decimal point), and
  - m is the number of digits to be displayed after the decimal

```
float pi = 3.141592;
printf("%f\n",pi); //Usual

printf("%6.2f\n", pi); //2 decimal

printf("%.4f\n",pi); //4 decimal
                     // Note rounding off!
```

Output

```
3.141592
  3.14
3.1416
```

Good and Not so good printf’s

```
# include <stdio.h>
int main() {
    float x;
    x=5.67123;
    printf(“%f”, x);
    return 0;
}
```

Compiles ok

Output

```
5.671230
```

```
# include <stdio.h>
int main() {
    float x;
    x=5.67123;
    printf(“%d”, x);
    return 0;
}
```

Compiles ok

```
-14227741
```

Printing a float using %d option is undefined. Result is machine dependent and can be unexpected. AVOID!

C often does not give compilation errors even when operations are undefined. But output may be unexpected!

Comments

- Supplementary information in programs to make understanding easier
  - Only for Humans!
  - Ignored by compilers

Comments in C

- Anything written between /* and */ is considered a comment.

```
diameter = 2*radius; /* diameter of a circle */
```

- Comments can NOT be nested.

```
/* I am /* a comment */ but I am not */
```

First */ ends the effect of all unmatched start-of-comments (/*).

Comments in C

- Anything written after // up to the end of that line

```
diameter = 2*radius; // diameter of a circle
area = pi*radius*radius; // and its area
```

- Not all C compilers support this style of comments.
  - Our lab compiler **does** support it.

Summary: An Example Program

```
#include <stdio.h>
int main()
{
   float mi, km; // decl without initialization

   scanf("%f",&mi); // get miles from user
   km = mi * 1.609; // compute and store km

   printf(“%.3f miles = %.3f kms.\n”,
           mi, km); // show the answer.
   return 0;
}
```

ESC101: Introduction to Computing

Operators and Expressions

Binary Operations

- Operate on int, float, double (and char)

Op	Meaning	Example	Remarks
+	Addition	9+2 is 11	
		9.1+2.0 is 11.1	
-	Subtraction	9-2 is 7	
		9.1-2.0 is 7.1	
*	Multiplication	9*2 is 18	
		9.1*2.0 is 18.2	
/	Division	9/2 is 4	Integer div.
		9.1/2.0 is 4.55	Real div.
%	Remainder	9%2 is 1	Only for int

Unary Operators

- Operators that take only one argument (or operand)
  - -5
  - +3.0123
  - -b
- Observe that + and – have two purposes
  - Meaning depends on context
  - This is called overloading

The / operator

- When both (left and right) operand of / are of type int
  - The result is the integral part of the real division
  - The result is of type int
- Examples

    9/4 is 2

    1/2 is 0

The / operator

- When at least one (left or right or both) operands of / are of type float (double)
  - The result is the real division
  - The result is of type float (double)
- Examples

  9/4.0 is 2.250000

  1.0/2 is 0.500000,

  so is 1/2.0

  and 1.0/2.0

The % operator

- The remainder operator % returns the integer remainder of the result of dividing its first operand by its second.
- Both operands must be integers.
- Defined only for integers (int and long)

  4%2 is 0

  31%4 is 3

Divison(/) and Remainder(%)

- Second argument can not be 0
  - Run time error
- For integers a and b (b≠0), / and % have the following relation

  a = (a/b)*b + (a%b)
- If a or b or both are negative, the result of / and % is system dependent.
  - But satisfies the above relation

Program Example

Volume of a cone = $\frac{1}{3} \times \pi \times radius^2 \times height$

```
float r,h;
scanf("%f", &r);
scanf("%f", &h);
printf("Volume is %.1f.",
     1/3*3.14*r*r*h);
```

Where did my ice cream go?

Input:
10.0
3.0

Output?

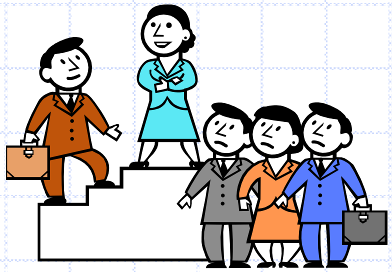
0.0

1/3 evaluates to 0
1.0/3.0 evaluates to 0.3333...
Remember: use floats for real division

Type of Arithmetic Expr

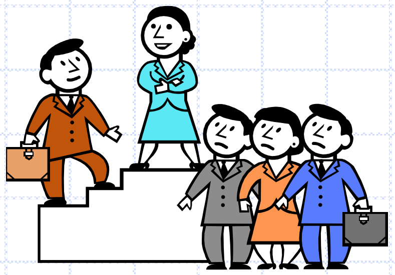
- Type of (result of) arithmetic expr depends on its arguments
- Rule of thumb:
- For binary operator
  - If both operands are int, the result is int
  - If one or both operands are float, the result is float
- For unary operator
  - Type of result is same as operand type

Operator Precedence



- More than one operator in an expression
  - Evaluation is based on precedence
- Parenthesis (...) have the highest precedence
- Precedence order for some common operators coming next

Operator Precedence



HIGH

INCREASING

LOW

Operators	Description	Associativity
(unary) + -	Unary plus/minus	Right to left
* / %	Multiply, divide, remainder	Left to right
+ -	Add, subtract	Left to right
< > >= <=	less, greater comparison	Left to right
== !=	Equal, not equal	Left to right
=	Assignment	Right to left