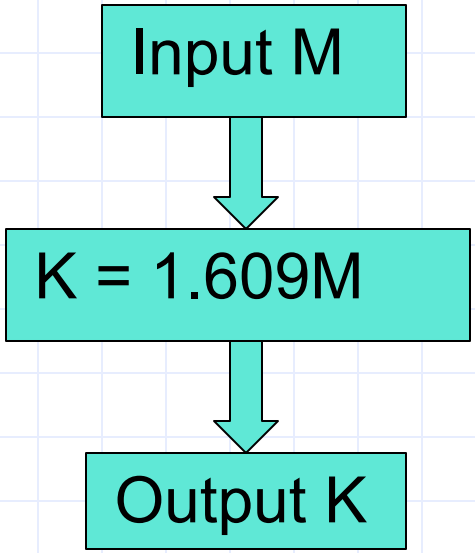
# ESC101: Introduction to Computing

## Operators and Expressions

# An example problem

**Problem: Read a distance in miles.**
**Convert it into kilometres and print it.**




Flowchart

# 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;
}
```

# A simple program

- **A program that takes in two numbers and averages them**

```
# include <stdio.h>
int main(){
   float a, b;
   float c;
   scanf("%f %f",&a, &b);
   c = (a+b)/2;
   printf("avg is %f\n",c);
   return 0;
}
```

**Input: 3, 4: Output =**

# A simple program

- **A program that takes in two numbers and averages them**

```
# include <stdio.h>
int main(){
    float a, b;
    float c;
    scanf("%f %f",&a, &b);
    c = (a+b)/2;
    printf("avg is %f\n",c);
    return 0;
}
```

**Input: 3, 4: Output = undefined, depends on system, 1.500000 on my system**

# A simple program

- **A program that takes in two numbers and averages them**

```
# include <stdio.h>
int main(){
   float a, b;
   float c;
   scanf("%f %f",&a, &b);
   c = (a+b)/2;
   printf("avg is %f\n",c);
   return 0;
}
```

**Input: 3 4: Output = 3.500000**

# 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

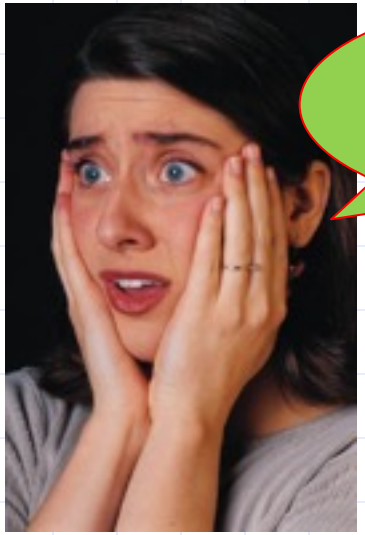
  a = (a/b)*b + (a%b)

- If a or b or both are negative, the result of / and % is system dependent
- https://groups.google.com/forum/#!msg/comp.std.c/hjGK3cx_I-o/_zB17QyViDgJ

# Program Example

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

```
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

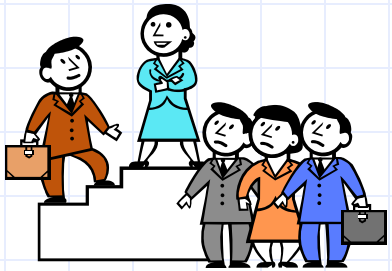
# Type of an Arithmetic Expression

- Type of (result of) an arithmetic expression depends on its arguments

Rule of thumb:

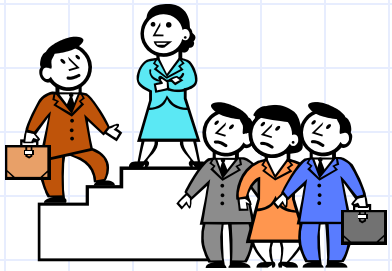
- For binary operator
  - If both operands are int, the result is int
  - If one or both operands are float, the result is float
- For unary operator
  - Type of result is same as operand type

# 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 |

# Operator Precedence

| Expression | Evaluation |
|---|---|
| 1+2*2 | 1+(2*2) |
| 1+2*2*3 | 1+((2*2)*3) |
| (1+2)*3*4 | (((1+2)*3)*4) |

# Operator Precedence

Question: What is the value assigned to the variable x by the following statement?

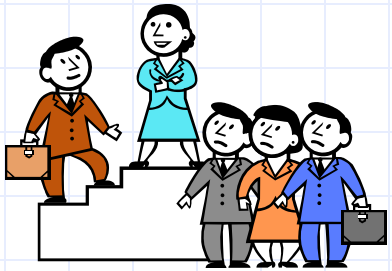
```
x = -5*4/2*3+-1*2;
```

# Operator Precedence

Question: What is the value assigned to the variable x by the following statement?

```
x = ( ( (-5)*(4/2) )*3)+( (-1)*2 );
```

# 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 |

# Fun with operators

- **Find the missing operators**

```
# include <stdio.h>
int main(){
    int x=3, y=7, z=9;
    int m=_______________
    printf("%d",m);//output m
    return 0;
}
```

If the output is: 78, what is an operator sequence that does not use any other constants?

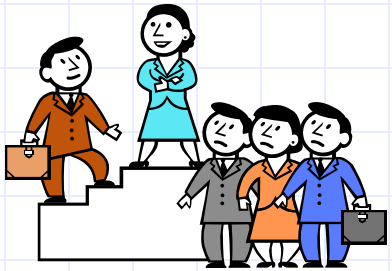
# Fun with operators

- **Find the missing operators**

```
# include <stdio.h>
int main(){
    int x=3, y=7, z=9;
    int m=z*z-x;
    printf("%d",m);//output m
    return 0;
}
```

What other sequences can you find?
Another valid option is: (z-y)*(x*y +(z-y)*z)

# Operator Precedence



- Always use parenthesis to define precedence. It is safer and easier to read.
- Avoid relying on operator precedence. Can give absurd results if not used correctly.
- Consult any textbook to know more about precedence.

# Type Conversion (Type casting)

- Converting values of one type to other.
  - Example: int to float and float to int (also applies to other types)
- Can be implicit or explicit

```
int k =5;
float x = k;            // implicit conversion, x gets 5.0
                        // value of k is not changed.
float y = k/10;         // y is assigned 0.0
                        // WHY?
float z = ((float) k)/10; // Explicit conversion
                          // z is assigned 0.5
```

# Loss of Information!

- Type conversion may result in lost information.
- Larger sized type (e.g. float ) converted to smaller sized type (e.g. int) is unpredictable.
- Smaller sized type (e.g. int) converted to larger type (e.g. float) may also result in loss. Take care!

# float to int: type conversion

```
#include<stdio.h>
int main() {
    float x;  int y;      /* define two variables */
    x = 5.67;
    y = (int) x;          /* convert float to int    */
    printf("%d", y);
    return 0;
}
```

Output : 5

```
float x;
…
(int) x;
```

converts the real value stored in x into an integer. Can be used anywhere an int can.

# float to int: type conversion

- float is a larger box, int is a smaller box. Assigning a float to an int may lead to loss of information and unexpected values.

# Operators on char

- Basic facts
  - Characters in C are encoded as numbers using the ASCII encoding
  - ASCII : American Standard Code for Information Interchange
- Encodings of some of the common characters:
  - 'A' is 65, 'B' is 66, 'C' is 67 ... 'Z' is 90
  - 'a' is 97, 'b' is 98 ... 'z' is 122
  - '0' is 48, '1' is 49 ... '9' is 57

# Operators on char

- Range: 0 to 255
- You do not have to remember ASCII values
  - Encoding/programming languages provide alternatives to use them
- C treats characters as integers corresponding to their ASCII value.
- While displaying with %c placeholder, the ASCII value is converted to its corresponding character.

# Operators on char

- Interconversion between character and integer datatypes can be exploited to write programs.

```
printf("%d\n", 'A');
printf("%d\n", '7');
printf("%c\n", 70);
printf("%c\n", 321);
```

Output:
65
55
F

321 is outside range!
What do you think will be the output of
printf("%c\n",321);
Try it out

# Operators on char

- Interconversion between character and integer datatypes can be exploited to write programs.

```
printf("%c\n",'C'+5);
printf("%c\n",'D' - 'A' + 'a' );
printf("%d\n",'3' + 2);
```

Output:
H
d
53

- Placeholder determines the output.
- Use with caution.
- Avoid arithmetic operation such as * and / on characters.
- Common Mistake: Incorrect data type of placeholder.

# Fun with characters

- **Find the missing operators**

```
# include <stdio.h>
int main(){
    int x=3, y=7, z=9;
    char a=‘A’, b=‘d’,c=‘x’;
    a=_____________;
    b=_____________;
    c=_____________;
    printf(“%c%c%c”,a,b,c);//output m
    return 0;
}
```

If the output of the program is Esc, what are the missing statements?

# Fun with characters

- **Find the missing operators**

```
# include <stdio.h>
int main(){
   int x=3, y=7, z=9;
   char a=‘A’, b=‘d’,c=‘x’;
   a=a+x+x+y-z;
   b=c-x-z+y;
   c=c-x*y;
   printf(“%c%c%c”,a,b,c);//output m
   return 0;
}
```

What are other ways of obtaining the same output?

# Next week

- Conditional expressions and statements