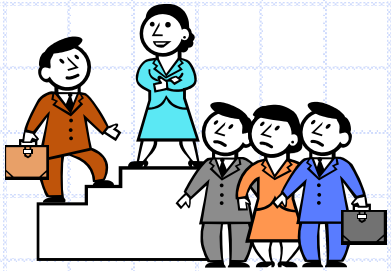
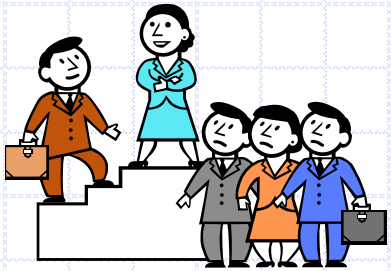
Operator Precedence



- 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



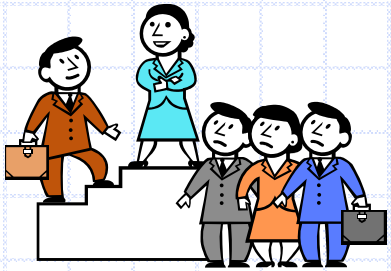
HIGH

INCREASING

LOW

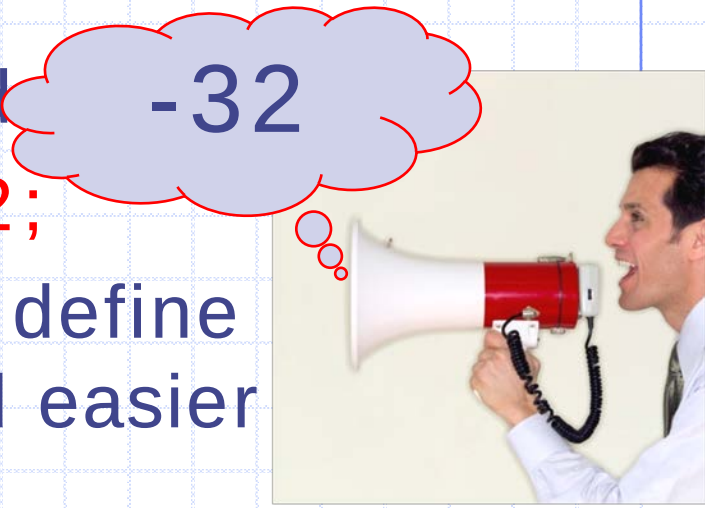
Operators	Description	Associativity
(unary) + -	Unary plus/minus	Right to left
* / %	Multiply, divide, remainder	Left to right
+ -	Add, subtract	Left to right
< > >= <=	less, greater comparison	Left to right
== !=	Equal, not equal	Left to right
=	Assignment	Right to left

Operator Precedence



- What is the value assigned

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



- 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 undefined/ 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 (result ok)

```
#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 (not ok!)

- 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.

The floating point number 1E50 is too large to fit in an integer box.

```
# include <stdio.h>
int main() {
    float x; int y;
    x  = 1.0E50; // 10^50
    y = (int) x;
    printf("%d", y);
    return 0;
}
```

Output:
2147483647

Careful when converting from a 'larger' type to 'smaller' type. Undefined.

int to float (take care!)

```
# include <stdio.h>
int main() {
    int y;
    y = 1000001;
    printf("%f", (float) y);
    return 0; }
```

Output:
1000001.000000

Result is correct

```
# include <stdio.h>
int main() {
    int y;
    y = 10000009;
    printf("%f\n", (float) y);
    printf(" %d", y);
    return 0; }
```

Output:
10000008.000000
10000009

Result is not correct.
Information is lost.

temp_conversion.c

```
# include <stdio.h>
int main()  {
      float C;
      float F;
      C=50;
      F = ((9*C)/5) + 32;
      printf("The temperature");
      printf( " %f ", C);
      printf("Celsius equals");
      printf(" %f ", F);
      printf("Fahrenheit");
      return 0;
}
```

- Microprocessors represent real numbers using ***finite precision***, i.e., using *limited number of digits after decimal point.*

- Typically uses scientific notation: 12.3456789 represented as 1.23456789E+1. Bit more later.

"%f" signifies that the corresponding variable is to be printed as a real number in decimal notation.

C 50.000000 122.000000 F

Compile and Run

```
The temperature  50.000000  Celsius equals  122.000000  Fahrenheit
```

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

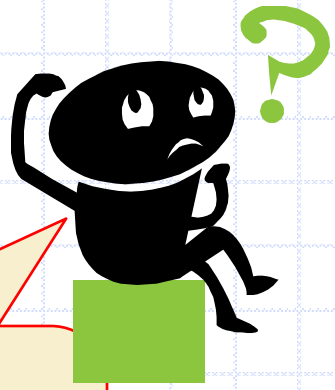
- Range: 0 to 255
- You should NOT try to remember ASCII values
  - Encoding/programming languages provide alternatives to use them
- C treats characters as integers corresponding to their ASCII value.
- While displaying with %c placeholder, the ASCII value is converted to its corresponding character.

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.

ESC101: Introduction to Computing

Conditional Expressions