UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

# ECE160: Foundations of Computer Engineering I

## Lecture #9 – C Expressions (2)

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

# Administrative Issues

- Lab#3 due **5pm, Wednesday, Feb. 8**

- Homework #2 due **9am, Friday, Feb. 10**
  - Please follow the “submission guidelines” available in the course website to submit your answers to your name folder at the class M: drive if you haven’t
  - Late submission is subject to penalty.

## Review of Lecture #8

- Expressions are combinations of operands (data that take part into operation: variables, constants) and operators (+, -, *, etc)
- Five types of expressions in C
  - Primary expressions
  - Binary expressions: multiplicative and additive
  - Assignment expressions
  - Postfix expressions
  - Unary expressions

## Topics

- **Precedence and associativity**
- Evaluating complex expressions
- Mixed type expressions

## Precedence and Associativity

- Precedence determines the order in which different operations are evaluated.
- Associativity determines how operators with the same precedence are grouped together in complex expressions.
- Note: precedence is applied before associativity.

## Operator Precedence (in descending order)

Postfix operators: ++, --, ..
Prefix operators: ++, --, ..
sizeof
Plus/minus signs: +,-
Logical NOT: !
Type cast: ()
Multiplicative operators:  *, /, %
Addition: +, -
Shift: << , >>
Relation: < , <=, >= ..
Equality operations: ==, !=
Bitwise/Boolean AND: &
Bitwise/Boolean XOR: ^
Bitwise/Boolean OR: |
Conditional AND: &&
Conditional OR: ||
Ternary conditional operator: ?:
Assignment: = , +=, -=, etc..

## Examples of Precedence

- 10 + 3*4; $\rightarrow$ 10 + (3*4);
- 20 – 4/2; $\rightarrow$ 20 - (4/2);
- -b++; $\rightarrow$ -(b++);

## Exercise (1)

What is the value of c?

```
int a = 2;
int b = 7;
int c = 0;
c = b/a;
```

## Exercise (2)

What is the output of printf()?

```
int a = 2;
int b = 3;
int c = 7;
printf(“%d\n”, a * b + c);
printf(“%d\n”, a * (b + c));
```

## Associativity

- Associativity is used only when the operators all have the same precedence!
- Associativity can be either from the left or the right.
  - Left associativity evaluates an expression from the left.
  - Right associativity evaluates an expression from the right.
- The left type is the most common.
  - For example, addition, subtraction, multiplication, division have left associativity

## Example (Left Associativity)

6 * 3/7 *2 %3

*  / * %  have the same precedence, their associativity is from left to right:

6 * 3/7 *2 %3 ←→ (((( 6*3)/7) * 2) %3)

*What is the value of this expression?*

## Example (Right Associativity)

- Assignment has right associativity
- When more than one assignment operators occur in an expression, they must be evaluated from right to left!
- Example:

```
int a, b, c;
a = 10;
b = 20;
c = 30;
b += a *=  c -= 2;
```

*What is the value of a,b,c?*

# Review Questions (1)

- What is the output of each printf() statement in the program?

```
#include <stdio.h>
void main(void)
{
  int a=3;
  int b=7;
  float c=6.0;
  a++;
  printf("%d\n", a/b);
  printf("%f\n", a/c);
  printf("%d\n", b%a+a);
  printf("%f\n", c%a);
  b=++a;
  printf("%d\n", b);
  printf("%d\n", a);
  printf("%d\n", a--);
  printf("%d\n", a);
  printf("%d\n", --a);
  printf("%d\n", a);
}
```

## Review Questions (2)

- What is the output of each printf() statement in the program?

```
#include <stdio.h>
void main(void)
{
  int a=3;
  printf(“%d\n”, a++ + a);
  printf(“%d\n”, ++a +a);
}
```

## Side Effects

- A side effect is an action that results from the evaluation of an expression
- Example: changing the value of a variable is a side effect

x=3;

  - On the right of = is a primary expression that has value 3
  - The whole expression (x=3) also has a value of 3 (note: the value of the total assignment expression is the value of the expression on the right of =)
  - x receives the value 3 (side effect)

## Side Effects

- Other operators that have side effects:
  - side effects take place before the expression is evaluated: ++a   --a
  - side effects take place after the expression is evaluated: a++    a--

## Topics

- Precedence and associativity
- **Evaluating complex expressions**
  - Expressions without side effects
  - Expressions with side effects
- Mixed type expressions

## Evaluating Complex Expressions without Side Effects

1. Replace the variables by their values
2. Evaluate the highest precedence operators and replace them with resulting value
3. Repeat step 2 until result is a single value.

## Example

Example:

```
int a, b, c;
a = 2;
b = 30;
c = 4;
/*What is the value of*/
c *2 + b/2 –a*c ;
```

1. Replace the variables by their values

   4*2+30/2-2*4

2. Evaluate the highest precedence operators and replace them with resulting value

   (4*2) + (30/2) – (2*4)

   $\rightarrow$ 8 + 15 – 8

3. Repeat step 2 until result is a single value.

   15

# Expressions with Side Effects

```
int a=3, b=4, c=5;
--a*(3+b)/2-c++*b;
```

Rewrite the expression as follows:

- Place all the prefix expressions before the expression being evaluated. Replace each prefix expression with its value and put the new value in the original complex expression.
- Place the postfix expressions after the expression being evaluated. AFTER the original complex expression has been evaluated, compute the value of the postfix expression.

```
--a
2*(3+4)/2-5*4;
c++
```

What is the value of the expression? -13

What is the value of a, b, c?

a=2 b=4 c=6

## Exercises

```
int a =2 , b = 4, c = 5;
++a *(4+c)/3 –b++ *c;
b-1;
```

What is the value of the above expressions?

Note: In ANSI C, the result is undefined, if a single variable is modified more than once in an expression.

So,

b-- + b -- is undefined!

b++ - b++ is undefined!

*ANSI: American National Standards Institute*

## Agenda

- Precedence and associativity
- Evaluating complex expressions
  - Expressions without side effects
  - Expressions with side effects
- **Mixed type expressions**

## Mixed Type Expressions

- An expression involves different types of data
  - Multiply an integer and a float number

- In an assignment expression, the final expression value must have the same type as the left operand, the operand that receives the value!

# Mixed Type Expressions

- What happens if we have to add a float with an integer?

- Implicit type conversion takes place!

  This means that variables with low precedence are promoted to match the highest precedence hierarchy in the expression.

  The integer would be converted to a floating point value first and then addition!

## Promotion Hierarchy

|  |  |
|---|---|
| Highest → | long double |
|  | double |
|  | float |
|  | unsigned long int |
|  | long int |
|  | unsigned int |
|  | int |
|  | short |
| Lowest → | char |

## Examples

char + float → float

int + float → float

int * double → double

- Note: Implicit type conversion is done by the compiler.

# Explicit Type Conversion (cast)

- Explicit type conversion uses cast operator: (new type)

- Example:

```
int b;
(float) b; /* this casts b to a floating point value*/
```

- Explicit type conversion is done by the Programmer.

## Exercises

```
int a=2;
int b=3;
int c=0;
float d=0;
int e=0;
float f=0;

c= a/b;
e = (float) a/b;
d = (float) a/b;
f = (float) (a/b);
```

The final expression value must have the same type as the left operand, the operand that receives the value!

*What is the value of c, e, d, f?*

## Exercises

- Assume int b = 2; and the result is stored in a float variable.
  - What is the result of (float) (b/20);
  - What is the result of (float) b/20;

## Downward Cast

- Do a downward cast and see what happens.
- For example, take a float and cast it to an int. Then print it.

```
float a =2.3;
int b = (int) a;
printf("%d\n",b);
```

- The result is 2.
- So the compiler allows you to do downward casting. But remember! It is usually a dangerous thing because you lose precision.

## Exercise

- What is the value of each of these expressions?

```
float x =10 – 2*3;
int a= 15%2.0;
float y =3- 15/3.0;
int b = 30 % 14;
float z = -30 + 2*3*5.0 ;
float d = 10 + 9 –3/4 + 3.0;
```

## Exercises

- Given

```
int a = 3;
int b = 4;
int c = 5;
float x,y,z;
```

- What is the value of x,y,z? Assume that the statements are consecutive lines in the same program:

```
x = a++ + ++b +(float)b/a;
y= c-- /a + b;
z = b – c + ++a/b-- -b/a;
```

## Summary of Lectures #9

- Precedence and associativity
- Evaluating complex expressions
  - Expressions without side effects
  - Expressions with side effects
- Mixed type expressions
  - Implicit type conversion
  - Explicit type conversion

## Things To Do

- Review Lectures
- Homework #2  Due by **Friday, Feb. 10**

## Next Topic

- Decision making