UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #10 – Decision Making (I)

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

Administrative Issues

- Lab#4 starts on Monday, Feb. 13
  - Due **5pm, Wednesday, Feb. 15**
- Exam#1 on **Friday, Feb. 17**
  - Review session on Wednesday, Feb. 15

Review 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 using cast operator (new type)

Outline

- Logical data and operations
- Relational operators
- Two-way selection (*if …else* statement)

Logical Data in C

- Logical data: true (1) or false (0).
- C does not have a logical data type.

- We can use other data types (usually int) to represent logical data.
  - 0 is considered false
  - any nonzero value is considered true.

Logical Operators

- ! →logical NOT
  - It is a unary operator and it changes a true (nonzero) value to false (zero) and vice versa.
- && → logical AND
  - It is a binary operator and the result is true only when both operands are true
- || → logical OR
  - It is a binary operator and the result is true if any of the operands is true. It is false when both operands are false.

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: |
Logical AND: &&
Logical OR: ||
Ternary conditional operator: ?:
Assignment: = , +=, -=, etc..

Exercise (1)

What is the value of
each logical expression?

```
! 7
! 0
3 && 0
1 && 0
1 && 1
7 && 1
1 || 0
1 || 3
0 || 0
3 || 0
0 || 7
!0 && 7
```

Exercise (2)

- If x = 2, y = 5, z = 9, what is the value of the following expressions?

```
(x && y) || z

! x || (z && y)

!y && (!x && z)
```

Exercise (3)

If x = 1, y = 5, z = 3 what is the result of the following expressions?

(3*y + 5 –(x%5))&& z

x && y %z

Exercise (4)

- Write a program that reads two integers from the keyboard and computes their logical AND, OR and NOT operations.

Outline

- ✓ Logical data and operations
  - 0: false (0)
  - any nonzero value: true (1)
  - Logical NOT (!), Logical AND (&&), Logical OR (||)
- Relational operators
- Two-way selection (*if …else* statement)

Relational Operators

- They are all binary operators for comparing two operands

<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
==	equal
!=	not equal

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: |
Logical AND: &&
Logical OR: ||
Ternary conditional operator: ?:
Assignment: = , +=, -=, etc.

Exercise (5)

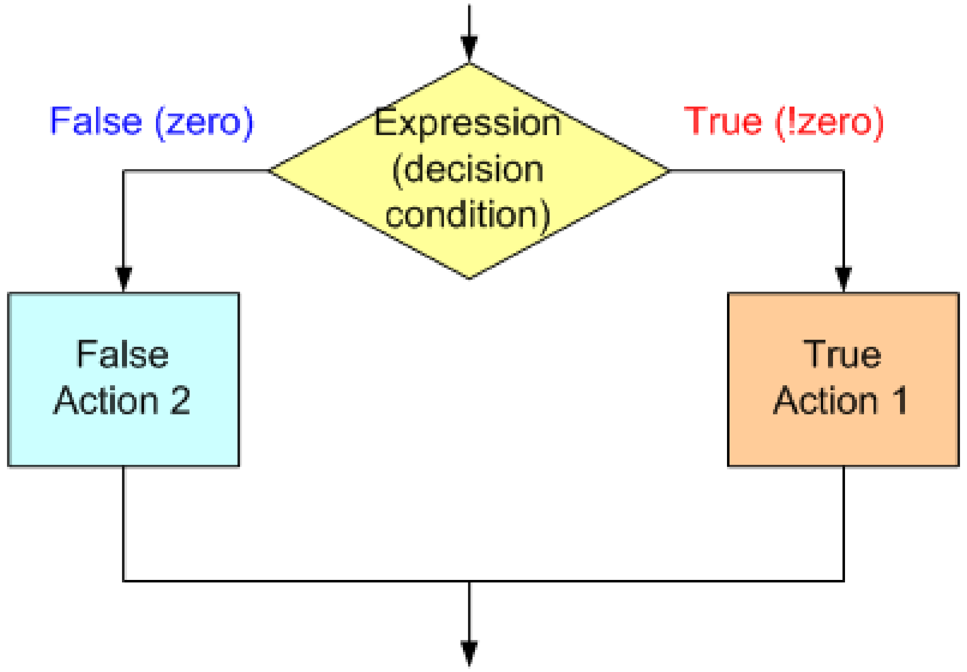
- Write a program that reads in two integers and prints the result of:

```
a > b
a >= b
a == b
a != b
a < b
a <= b
```

Agenda

- ✓ Logical data and operations
- ✓ Relational Operators
- **Two-way selection (*if …else* statement)**

Two-Way Selection



if-else statements

Syntax:

```
if (expression)
{
   Action 1
 }
 else
{
 Action 2
 }
```

Simple form:

```
if (expression)
   statement 1;
    else
   statement 2;
```

If expression is true (evaluates to 1), perform Action 1, else perform Action 2.

No semicolon (;) is needed for an if…else statement
Statement 1 and 2 may have a ; as required by their types

Exercise (6)

- Write a program that reads a number from the keyboard. If the number you entered is an even number, it outputs: “You entered an even number”. If the number you entered is an odd number, it outputs: “You entered an odd number”.

```
#include <stdio.h>
void main(void)
{
        int a=0;
        printf("Enter an integer\n");
        scanf("%d", &a);
        if ((a%2)==0)
                printf("You entered an Even number");
        else
                printf("You entered an Odd number");
}
```

Exercise (7)

- If the expression is changed to (a%2)==1, what changes should be made to the program?

```
#include <stdio.h>
void main(void)
{
        int a=0;
        printf("Enter an integer\n");
        scanf("%d", &a);
        if ((a%2)==1)
                printf("You entered an Odd number");
        else
                printf("You entered an Even number");
}
```

Note (1)

- We don't have to have an else statement. If we need to take action, only when a certain condition is met, then we only need an *if*.

```
if (expression)
  {
   ......;
   ......;
   }
```

- If we need to take an action when a condition is met and a different action when the condition is not met, then we need an *else* too.

Note (2)

- You can have multiple *if* like below. And there may or may not be an *else* statement.

```
if (expression 1) {
 ……..
}
if(expression 2) {
…..
}
if(expression 3) {
…
}
```

Exercise (8)

- Write a program that reads 3 numbers from the keyboard and adds the first two.
  – If their sum is greater than the third number, it prints “Sum is greater than the third number”.
  – If their sum is equal to the third number it prints “Sum is equal to the third number”.
  – If their sum is less than the third number, it prints “Sum is less than the third number”.

Solution

```
#include <stdio.h>
void main(void)
{
    int a=0,b=0,c=0;
    printf("Enter three numbers\n");
    scanf("%d%d%d",&a,&b,&c);
     if ((a+b) > c) {
            printf("Sum is greater than the third number");
     }
      if((a+b)==c) {
            printf("Sum is equal to the third number");
     }
      if((a+b) < c) {
            printf("Sum is less than the third number");
     }
}
```

Nested *if* Statements

An *if…else* is included within another *if…else*

```
if (expression)
{
   if … else statement
  }
 else
{
 Action 2
 }
```

Example

```
#include "stdio.h"
void main(void)
{
    int a,b;
    printf("Enter two integers:\n");
    scanf("%d%d",&a, &b);
    if(a >= b)
        {
            if(a > b)
                printf("%d > %d",a,b);
            else
                printf("%d == %d",a,b);

        }
    else
        {
            printf("%d < %d", a, b);
        }
}
```

Good programming style:
Using indention
Line up opening and closing braces

If you enter 3 and 7 from the keyboard, what is the output of the program?

Dangling *else* Problem

- The problem is created when there is no matching *else* for every *if*
- Solution:
  - *Always pair an “else” to the most recent unpaired “if” in the current block!*

Example

```
if(a >= b)
        if(a > b)
          printf("%d > %d",a,b);
else
   printf("%d == %d", a, b);
```

Which *if* does this *else* belong to?

The second one

Example

```
if(a >= b) {
        if(a > b)
          printf("%d > %d",a,b);
}
    else
          printf("%d == %d", a, b);
```

Which *if* does this *else* belong to?

The first one

Conditional Operator

- C provides a convenient alternative to *if…else*: the ternary conditional operator

  expression1 ? expression2 : expression3

- This means that if expression1 is true, then the overall expression evaluates to expression 2, else it evaluates to expression3.

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: |
Logical AND: &&
Logical OR: ||
Ternary conditional operator: ?:
Assignment: = , +=, -=, etc..

Exercise (9)

`x = (a == b) ? c--: c++`

- If a is equal to b, c-- will be evaluated, its value is assigned to x, and 1 will be subtracted from c (side effect)
- Else (if a is not equal to b), c++ will be evaluated and assigned to x, and 1 will be added to c (side effect)

For a=3, b=7, c=0, what is the value of x and c after the expression is evaluated?

How about for a=3, b=3, c=0?

Review Questions

Exercise (10)

If x = 3, y = 2, z = 9, what is the value of x,y,z after executing the following code:

```
If( x && y)
      x = 10;
 else
    y = 5;
```

Exercise (11)

If originally x=0, y =1 and z = 2, what is the value of x,y,z after the execution of the code?

```
if (y)
   if(x || y)
        z = 10;
   else
     z = 5;
```

Exercise (12)

If originally x = 0, y = 0 and z = 20, what is the value of x,y,z after executing the following piece of code?

```
if( z == y) {
        x++;
        y++;
       }
 else
     y--;
```

Common Errors (1)

- Be aware of dangling else.
  - Always pair an “else” to the most recent unpaired “if” in the current block!
  - Use braces to avoid them.
- Be aware of side effects inside *if else* statements, e.g.: if (a--)
- Do not use the equal (==) operator with a floating point number. It almost never works.

Common Errors (2)

- DO NOT CONFUSE == (equal) with = (assignment).
- It is a compile error to have an *else* without a matching *if*.
- It is a compile error to forget the parentheses in the *if* expression.
- It is a compile error to put space between ==
  != >= <=

Summary of Lectures #10

- Logical data
- 3 logical operators
- 6 relational operators
- *if…else* statement
- Nested *if…else* statement
- Dangling *else* problem
- Ternary conditional operator ?:

Things To Do

- Review Lecture Note
- Run the programs in the exercises

Next Topic

- Decision Making II (switch)