HIGH-LEVEL PROGRAMMING I

Conditional Operator

by Prasanna Ghali

Reference

- Conditional operator is explained in Chapter 5 of the text

Purpose

- Write certain `if` statements with `else` clauses more concisely

```
int max(int x, int y) {
  int m;
  if (x > y) {
    m = x;
  } else {
    m = y;
  }
  return m;
}
```

```
char grade(double pts) {
  if (pts >= 70.0) {
    return 'P';
  } else {
    return 'F';
  }
}
```

Conditional Operator: Syntax (1/2)

- ***expression1 ? expression2 : expression3***

```
int max(int x, int y) {
  int m;
  if (x > y) {
    m = x;
  } else {
    m = y;
  }
  return m;
}
```

```
int max(int x, int y) {
  int m;
  m = x > y ? x : y;
  return m;
}
```

Terser but better

```
int max(int x, int y) {
  return x > y ? x : y;
}
```

Conditional Operator: Syntax (2/2)

- ***expression1 ? expression2 : expression3***

```
char grade(double pts) {
  if (pts >= 70.0) {
    return 'P';
  } else {
    return 'F';
  }
}
```

```
char grade(double pts) {
  return (pts >= 70.0) ? 'P' : 'F';
}
```

Conditional Operator and Expression

- Syntax for using conditional operator is

  expression1 **?** ***expression2*** **:** ***expression3***

- Evaluation proceeds as follows:
  - ***expression1*** is fully evaluated and tested against zero
  - If evaluation of ***expression1*** is not equal to zero, ***expression2*** is evaluated and its value is result of entire expression; ***expression3*** is not evaluated
  - If ***expression1*** is zero, ***expression3*** is evaluated and its value is result of entire expression; ***expression2*** is not evaluated

Precedence and Associativity

high to low precedence order

Operator	Meaning	Associativity
! + -	logical negation unary plus, unary minus	R-L
* / %	multiplication, division, remainder	L-R
+ -	addition, subtraction	L-R
< <= > >=	relational	L-R
== !=	equivalence	L-R
&&	logical AND	L-R
||	logical OR	L-R
?:	conditional (ternary)	R-L
=	assignment	R-L

Conditional Operator and Expression: Example


```
if (x > y) {
  printf("x is larger\n");
} else {
  printf("y is larger\n");
}
```

```
x > y ? printf("x is larger\n")
      : printf("y is larger\n");
```

Conditional Operator and Expression: Example


```
if (x > y) {
  z = x + 2;
} else {
  z = y + 5;
}
```

```
z = x > y ? x + 2 : y + 5;
```

Conditional Operator and Expression

- Write function **`signum`** that returns **`1`**, **`-1`**, or **`0`** depending on whether its argument is positive, negative, or zero

```
int signum(int x) {
 if (x > 0) return 1;
 else if (x < 0) return -1;
 else return 0;
}
```

```
int signum(int x) {
 if (x > 0) return 1;
 if (x < 0) return -1;
 return 0;
}
```

```
int signum(int x) {
  return (x > 0) ? 1 : (x < 0) ? -1 : 0;
}
```

Computing Maximum of 3 Integral Values (1/3)


```
if (x > y) {
  if (x > z) {
    max = x;
  } else {
    max = z;
  }
} else {
  if (y > z) {
    max = y;
  } else {
    max = z;
  }
}
```

```
max = x > y ? x>z ? x : z : y>z ? y : z;
```

Computing Maximum of 3 Integral Values (2/3)


```
max = x > y ? x>z ? x : z : y>z ? y : z;
```

```
max = (x > y) ? (x>z) ? x : z : (y>z) ? y : z;
```

```
max = ((x > y) ? ((x>z) ? x : z) : ((y>z) ? y : z));
```

Computing Maximum of 3 Integral Values (3/3)


```
max = ((x > y) ? ((x>z) ? x : z) : ((y>z) ? y : z));
```

- Rather than using conditional operator, this is much better:

```
int max(int x, int y) {
 return x > y ? x : y;
}

int max3(int x, int y, int z) {
 return max(max(x, y), z);
}
```

Conditional Operator and Expression: Exercises

- If `a = 1` and `b = 0`, what are values of `a`, `b` and `c` after following statements?

Statement	a	b	c
`c = a == b ? a + 2 : b + 5;`			
`c = a = b ? a + 2 : b + 5;`			
`c = a = b ? a + 2 : b += 5;`			
`c = (a = b) ? (a + 2) : (b += 5);`			