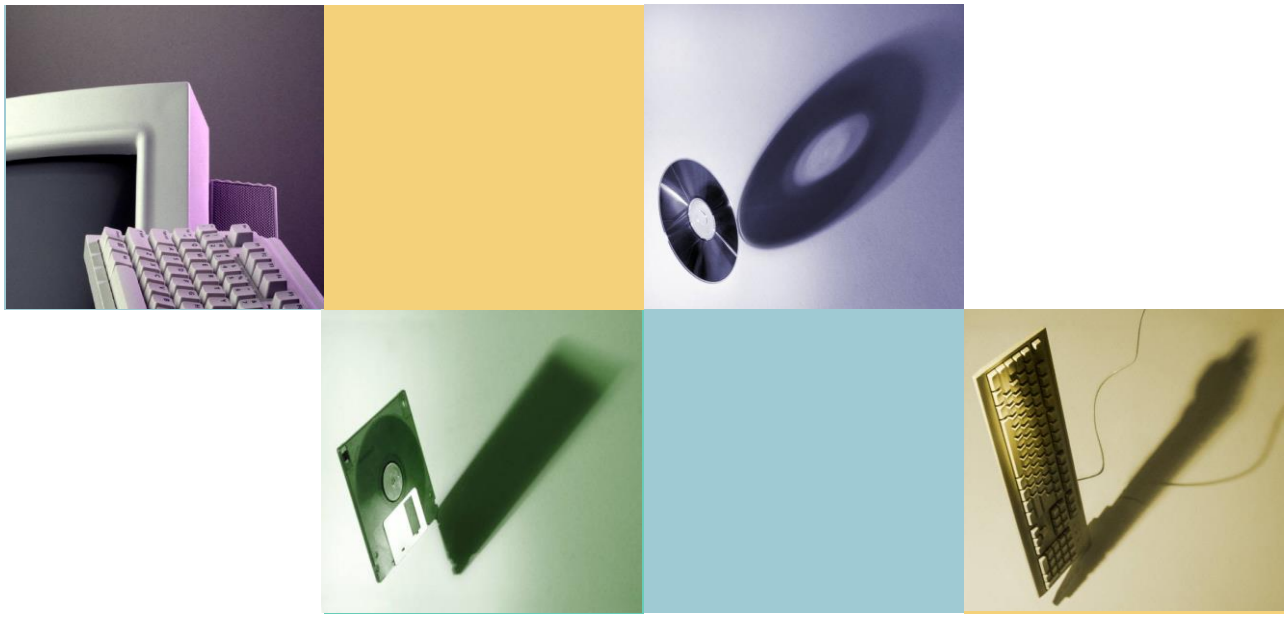
Object-Oriented Programming

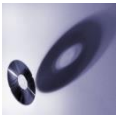


Chuan-Kang Ting

Dept of Computer Science and Information Engineering
National Chung Cheng University

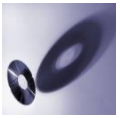
Chapter 2

Flow of Control


Outlines

- **Boolean Expressions**
- **Branching Mechanisms**
- **Loops**


Boolean Expressions (1)

- **Logical operators**
  - AND: &&
  - OR: ||
  - Comparison: (notice ==)

Display 2.1 Comparison Operators

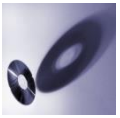
MATH SYMBOL	ENGLISH	C++ NOTATION	C++ SAMPLE	MATH EQUIVALENT
=	Equal to	==	`x + 7 == 2*y`	$x + 7 = 2y$
≠	Not equal to	!=	`ans != 'n'`	*ans* ≠ *'n'*
<	Less than	<	`count < m + 3`	*count* < m + 3
≤	Less than or equal to	<=	`time <= limit`	*time* ≤ *limit*
>	Greater than	>	`time > limit`	*time* > *limit*
≥	Greater than or equal to	>=	`age >= 21`	*age* ≥ 21

Boolean Expressions (2)

- **Short-circuit evaluation**
  - Idea:
    - `if (A && B)`: Either A or B being false makes the formula FALSE
    - `if (A || B)`: Either A or B being true makes the formula TRUE
  - Therefore, if the left most gives it enough information to determine the truth value of the expression, then C++ does NOT bother to evaluate the 2nd sub-expression
  - e.g. avoid the error "divided by zero"

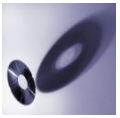
```
if ( (x!=0) && ((y/x)<1)) )
  cout << "ratio < 1" << endl;
```

- Evaluate in order
- NOT apply to overloaded || &&

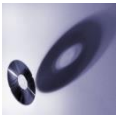

Boolean Expressions (3)

- **Integer ↔ Boolean**
  - Zero for FALSE
  - Nonzero for TRUE


Branching Mechanism

- **if-else**
- **switch**
- **enumeration**
  - A type whose values are defined by a list of constants of type int
    - Ex1. `enum MouseClick {LEFT=0, MIDDLE=1, RIGHT=2}`
    - Ex2. `enum MouseClick {LEFT, MIDDLE, RIGHT}`
  - Note that an enumeration type is different from the type int
- **conditional operator**
  - (Logical Expr.) **?** (ret. if true) **:** (ret. if false)
  - max = (n1 > n2) ? n1 : n2;

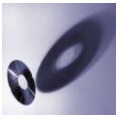

Common Pitfall

- **if-else**
  - Misusing "**=**" for "**==**"
    - e.g.

```
int x(0);
if (x = 12)
  Do_Something
else
  Do_Something_Else
```

`(x=12)` returns value **12** → **true**

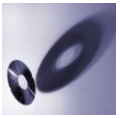
- **switch**
  - Forgetting the **`break;`**
    - No compiler error
    - Execution simply "falls thru" other cases until break;


Loops (1)

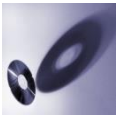
- **while**
- **do-while**
  - always executed at least once
- **for**
  - **`for`** (*Initialization_Action*; *Boolean_Expression*; *Update_Action*)
  - comma
    - a way of evaluating a list of expressions and returning the value of the last expression
      - Pitfall: No guarantee that the evaluations are in left-to-right order
      - only use it in a `for` loop, although it is legal in any expression
    - e.g.

```
for(sum=0, n=1; n<=10; sum+=n, n++);
```


Loops (2)

- **break**
  - When executed, the `break` statement ends the **nearest** enclosing `switch` or *loop* statement
- **continue**
  - When executed, the `continue` statement ends the current loop body iteration of the **nearest** enclosing *loop* statement
  - Restated, it jumps to the “}” of the nearest enclosing loop

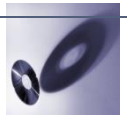
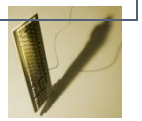




Loops (3)

- **break vs. continue**

```
#include <iostream>
using namespace std;

int main()
{
  int number, sum = 0, count = 0;
  cout << "Enter 4 negative numbers: \n";

  while (++count <= 4)
  {
    cin >> number;

    if (number >= 0)
    {
       cout << "Non-negative numbers! \n";
       break;
    }

    sum += number;
  }

  cout << sum << "is the sum of the first "
<< (count - 1) << " numbers. \n";

  return 0;
}
```

```
#include <iostream>
using namespace std;

int main()
{
  int number, sum = 0, count = 0;
  cout << "Enter 4 negative numbers: \n";

  while (count < 4)
  {
    cin >> number;

    if (number >= 0)
    {
       cout << "Non-negative numbers! \n";
       continue;
    }

    sum += number;
    count++;
  }

  cout << sum << "is the sum of the "
       << count << " numbers. \n";

  return 0;
}
```

Summary

- **Boolean Expressions**
  - Logic operators (&&, ||, comparison)
  - Short-circuit evaluation
- **Branching Mechanisms**
  - if-else
  - switch
  - conditional operator
- **Loops**
  - while and do-while
  - for
  - break and continue