CSE 5910 Control Flow

Instructor: Aijun An
Department of Computer Science and Engineering
York University
aan@cse.yorku.ca

http://www.cse.yorku.ca/course/5910

Outline

Control Flow

Two-way Branch

• if-else statement

Boolean Expressions

Multi-way Branches

- Nested if-else statement
- switch statement

Programming Style

Lab exercises (mixed with the above contents)

•

Flow of Control

A computer program is a sequence of statements.

Flow of control

• The order in which statements are executed

In a *simple* program, the statements are executed one after the other in the order that they are typed.

But in many situations we need programs in which

- statements are not necessarily executed in the order that they are typed.
- different commands are executed when the program runs with different input variables.

Types of Control Flow

Three types of control flow structures:

- Sequence
 - A group of statements are executed in the order they are typed, which specifies sequential flow.
- Branch
 - The program chooses between alternatives.
 - Depending on a given condition, the program decides
 - whether a certain statement or group of statements should be executed.
 - which statement or group of statements should be executed.
- Loop
 - Repeatedly execute one statement or a group of statements

Branch Example

To calculate the weekly wage of an employee there are two choices:

- Regular time (up to 40 hours)
 - gross_pay = hourly_rate * hours;
- Overtime (over 40 hours)
 - gross_pay = hourly_rate $*40 + 1.5 * hourly_rate * (hours 40);$
- $\bullet\,$ The program must choose which of these expressions to use

Designing the Branch

Let the user input the number of hours

Decide whether (hours >40) is true

- If it is true, then use gross_pay = hourly_rate * 40 + 1.5 * hourly_rate * (hours 40);
- If it is not true, then use gross_pay = hourly_rate * hours;

6

Implementing the Branch The if-else statement is used in C++ to perform a branch if (hours > 40) gross_pay = rate * 40 + 1.5 * rate * (hours - 40); else gross_pay = rate * hours;

```
Whole
                                                             #include <iostream
                                                             using namespace std;
int main()
 Program
                                                                    int hours;
double gross_pay, rate;
                                                                    cout << "Enter the hourly rate of pay: $";
                                                                   cont << Enter the number of hours worked,\n"
<= "rounded to a whole number of hours: ";
 Enter the hourly rate of pay: $20.00
Enter the number of hours worked,
rounded to a whole number of hours: 30
Hours = 30
                                                                    cin >> hours;
                                                                    gross_pay = rate*40 + 1.5*rate*(hours - 40);
else
                                                                   if (hours > 40)
 Hourly pay rate = $20.00
Gross pay = $600.00
                                                                          gross pay = rate*hours:
Enter the hourly rate of pay: $10.00
Enter the number of hours worked,
rounded to a whole number of hours: 41
Hours = 41
Hourly na
                                                                     cout.setf(ios::fixed);
                                                                    cout.setf(ios::showpoint);
                                                                   cout.recision(2);
cout << "Hours = " << hours << end];
cout << "Hours pay rate = $" << rate << end];
cout << "Gross pay = $" << gross_pay << end];</pre>
Hourly pay rate = $10.00
Gross pay = $415.00
```

```
Syntax for the if-else Statement
                             A sequence of statements for
A single statement for each
  alternative: () is a must
                               each alternative:
                                if (Boolean_Expression)
   if (Boolean_Expression)
      Yes statement
                                    Yes statement 1
  else
                                    Yes_statement_2
       No_statement
                                    Yes statement last
                               7}
                                else
 Braces are necessary
 when there is more than
                                    No_statement_1
 one statement in a block!
                                    No_statement_2
 They make a compound
 statement.
                                    No_statement_last
```

```
Compound Statements

A compound statement consists of one or more statements enclosed in {}. It is also called a block.

Branches of if-else statements often need to execute more that one statement. A block should be used.

Example: if (my_score > your_score)
{
    cout << "I win!\n";
    wager = wager + 100;
}
else
{
    cout << "I wish these were golf scores.\n";
    wager = 0;
}
```

```
Example of the if Statement

Display the ticket price
• senior has 10% discount

int price =50;

cout << "Enter your age: ";
cin >> age;
if (age > 64)
    price = price*0.9;
cout << "Your ticket price is: $" << price << endl;
```

Example of the if Statement

What is the output of the following piece of program?

```
cout << "Enter a value: ";</pre>
cin >> x:
y=x;
if (x<0)
  y=-x;
cout << y << endl;</pre>
```

Outline

Control Flow

Two-way Branch

• if-else statement

Boolean Expressions

Multi-way Branches

- Nested if-else statement
- switch statement

Programming Style

Lab exercises (mixed with the above contents)

Boolean Expressions

Boolean expressions are expressions whose value is either true or false

Boolean expressions are formed by combining variables, constants and expressions using relational and/or logical

- A relational operator (<, >, ==, <=, >=, !=)
 - · compares two values
 - forms a relational expression <--
- A logical operator (&&, | |, !) expressions
 - · exams true/false statements
 - forms a logical expression

Examples:

hours>40, age>=20, (x==y)&&(x!=z)

Relational operators

Relational operators:

- <, >, ==, <=, >=, !=
- less than
- greater than
- equal to or equivalent to
- <= less than or equal to
- >= greater than or equal to
- != not equal or inequality

A relational operator compares two operands, which can be numbers, variables, math expressions.

Examples:

These expressions are called relational expressions. hours>40, age>=20, x==y, answer=='y', (x+y)!=2*z

Exercise 1

Assume \mathbf{x} is 15 and \mathbf{y} is 25, what are the values of the following expressions?

- x!=y
- Value:
- true
- x<x
- Value:
- false • x>=y-x
- Value:
- (x+5)==(y-x)
- Value:
- false

Logical Operators

Logical operators:

- &&, ||, !
- && logic AND operator • A && B is true only if both A and B are true; otherwise it is false
- | logic OR operator
- logic NOT operator
 - !A is true if A is false; otherwise it is false
- Examples:
 - (score>70) && (score<=80)
 - (answer=='y')||(answer=='Y')
 - !(x==1)

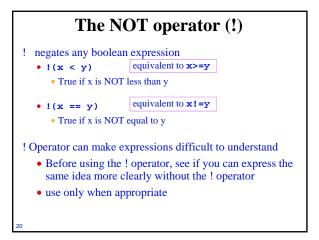
- No spaces are allowed between the symbols
 - in && and | !

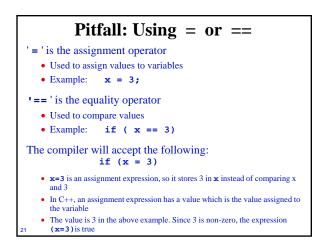
No spaces are allowed

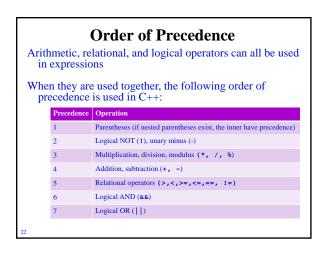
between the symbols!

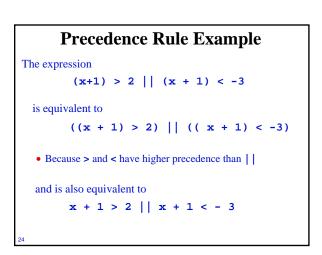
- A | B is true if either A or B or both are true; otherwise it is false
- - - These expressions are called *logical* expressions.

Logical Operators (Cont'd) Operands for a logical operator are usually relational expressions: • x>2 && y<=2 x>2 || y<=2 !(x>2) Operands can be other logical expressions: • (x>2 && y==2) | | (y<0) • x>2 && (y<=2 || y>10) Operands can be numbers as well 3 && 0 · A nonzero number is true Value: • A zero number is false false • 10 Value:









More Examples

```
Evaluating \mathbf{x} + \mathbf{1} < -(3+2) \mid \mid \mathbf{x} + \mathbf{1} > 2
• Using the precedence rules
```

- osing the precedence ru
- First apply the + in ()
- Next apply the unary -
- Next apply the other +'s
- Now apply the < and >
- Finally do the | |

```
Evaluating x+2>1 || y<2 && z >= 3
```

Short-Circuit Evaluation

Some boolean expressions do not need to be completely evaluated

• if x is negative, the value of the expression

(x >= 0) && (y > 1)

can be determined by evaluating only (x >= 0)

C++ uses short-circuit evaluation

If the value of the leftmost sub-expression determines the final value of the expression, the rest of the expression is not evaluated

6

Evaluating Boolean Expressions

```
Assume that y is 8, the expression
```

is evaluated in the following sequence

!(true)
false

If not sure, can use parenthesis

- The above expression can be equivalently written as:
- !((y < 3) | | (y > 7))

Evaluating Boolean Expressions

Assume that y is 8, the expression

is evaluated in the following sequence

true

28

Evaluating Boolean Expressions

Assume that y is 8, the expression

is evaluated in the following sequence

true || y>7

true

Using Short-Circuit Evaluation

Short-circuit evaluation can be used to prevent run time errors

• Consider this if-statement

```
if ((kids != 0) && (pieces / kids >= 2) )
    cout << "Each child may have two pieces!";</pre>
```

- If the value of kids is zero, short-circuit evaluation prevents evaluation of (pieces/0 >= 2)
 - Division by zero causes a run-time error

Exercise 2

Write an expression to test each of the following:

- age is from 18 to 21 inclusive 18<=age && age<=21
- water is less than 1.5 and also greater than 0.1
 water<1.5 && water>0.1
- speed is not greater than 55 speed<=55
- w is either equal to 6 or not greater than 3 w==6 | | w<=3

Exercise 3

Write a C++ program that

• prompts the user to input two numbers. The prompts are:

Input the first number: Input the second number:

• output the bigger number on the screen by displaying:

The bigger number is n.

where **n** is the inputted bigger number.

3

Solution to Exercise 3

```
#include<iostream>
using namespace std;
int main()
{
  int x, y, bigger;
  cout << "Please enter the first number: ";
  cin >> x;
  cout << "Please enter the second number: ";
  cin >>y;
  if (x>y)
    bigger = x;
  else
    bigger = y;
  cout << "The bigger number is " << bigger << endl;
  return 0;
}</pre>
```

Exercise 4

Write a C++ program that tells whether an input number is divisible by 4. The program should do the following:

- Prompt the user to enter a number.
- If the number is divisible by 4, output the following message on the screen:

It is divisible by 4

• If the number is not divisible by 4, output the following message on the screen:

It is not divisible by 4

Solution to Exercise 4

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

int main()
{
   int x;

   cout << "Please enter an integer: ";
   cin >> x;

   if (x*4 == 0)
      cout << "It is divisible by 4\n";
   else
      cout << "It is not divisible by 4\n";
   return 0;

36 }</pre>
```

Outline

Control Flow

Two-way Branch

• if-else statement

Boolean Expressions

Multi-way Branches

- Nested if-else statement
- switch statement

Programming Style

Lab exercises (mixed with the above contents)

7

Multiway Branches

The if-else statement allows us to create two-way branches in a program

Sometimes, it is necessary to select one out of a number of alternative actions

The if-else statement can be used to form multiway branches

An if-else statement can be a subpart of another ifelse statement.

• This forms a nested **if-else** statements

38

Nested Statements

A statement that is a subpart of another statement is a nested statement

• When writing nested statements it is normal to indent each level of nesting

```
Example:
```

```
if (gender == 'M')
  if ( hight > man_average_hight)
    cout << "You are taller than average" << endl;
  else
    cout << "You are shorter than average" << endl;
else
  if ( hight > wonman_average_hight)
    cout << "You are taller than average" << endl;
else
  cout << "You are shorter than average" << endl;</pre>
```

Nested statements are indented to show the logical structure.

Nested if-else Statements

Nested **if-else** statements can implement multi-way

But use care in nesting if-else statements

Example:

• For the example in the last slide, if we only output the message when the person's height is above the average, then you may want to write:

```
if (gender == 'M')
   if ( hight > man_average_hight)
      cout << "You are taller than average" << endl;
else
   if ( hight > wonman_average_hight)
      cout << "You are taller than average" << endl;</pre>
```

- This would compile and run, but does not produce the desired results
- The compiler pairs the "else" with the nearest previous "if"

Braces and Nested Statements

To solve the problem in the last slide, use braces to enclose the nested if statement especially when it does not have an else part:

```
if (gender == 'M')
{
    if (hight > man_average_hight)
        cout << "You are taller than average" << endl;
}
else
{
    if (hight > wonman_average_hight)
        cout << "You are taller than average" << endl;
}</pre>
```

Braces tell the compiler how to group things.

• They make a compound statement.

Multi-way if-else-statements

An **if-else** statement is a two-way branch

Three or four (or more) way branches can be designed using nested if-else statements

Example:

 The following nested statements implement the hints for a number guessing game:

Indenting Nested if-else

Notice how the code on the previous slide crept across the page leaving less and less space

Use the following for indenting such nested if-elsestatements:

```
if (guess> number)
    cout << "Too high.";
else if (guess < number)
    cout << "Too low.");
else if (guess == number)
    cout << "Correct!";</pre>
```

The Final if-else-statement

When the conditions tested in an if-else-statement include all the situations, the final "if" can be omitted.

The previous example can be written as

```
if (guess> number)
    cout << "Too high.";
else if (guess < number)
    cout << "Too low.");
else
    cout << "Correct!";</pre>
```

This is a commonly-used form of multi-way if-else statement.

44

Syntax for Multiway if-else Statement

A multiway if-else statement is written as

```
if(Boolean_Expression_1)
    Statement_1
else if ( Boolean_Expression_2)
    Statement_2
    ...
else if (Boolean_Expression_n)
    Statement _n
else
    Statement For All_Other_Possibilities
```

 ${\tt Statement_i} \ \ can \ be \ a \ single \ statement \ or \ a \ compound \ statement \ (that \ is \ a \ sequence \ of \ statements \ enclosed \ by \ \{\})$

15

Exercise 5

Write a C++ program that

- prompts the user to input a value for variable x, and
- computes the value for y given the inputted value for x, where

$$y = \begin{cases} 0 & \text{if } x \le 0\\ \frac{x^2}{2} & \text{if } 0 < x \le 1\\ 2x - \frac{x^2}{2} - 1 & \text{if } 1 < x \le 2\\ 1 & \text{if } x > 2 \end{cases}$$

46

Solution to Exercise 5

```
#include<iostream>
using namespace std;
int main()
{
    double x, y;
    cout << "Enter a value for x: ";
    cin >> x;
    if (x<=0)
        y=0;
    else if (x<=1)
        y=x*x/2;
    else if (x<=2)
        y=2*x-x*x/2-1;
    else
        y=1;
    cout << "y=" << y << endl;
    return 0;
47</pre>
```

Exercise 6

Write a C++ program that

- Prompt the user to input his/her income
- computes tax according to the following rate schedule:

No tax on first \$15,000 of income

5% tax on each dollar from \$15,001 to \$25,000

10% tax on each dollar over \$25,000

• Output to the screen:

Your income tax is \$x.

where \mathbf{x} is calculated tax and should be shown with two decimal places.

Outline

Control Flow

Two-way Branch

• if-else statement

Boolean Expressions

Multi-way Branches

- Nested if-else statement
- switch statement

Programming Style

Lab exercises (mixed with the above contents)

49

```
The switch statement

The switch-statement is an alternative for constructing multiway branches

Syntax:

switch (controlling expression)
{
    case Constant_1:
        Statement_Sequence_1
        break;
    case Constant_2:
        Statement_Sequence_2
        break;
    case Constant_n:
        Statement_Sequence_n
        break;
    default:
        Default_Statement_Sequence
}
```

Example of switch Statement

```
char grade;
cout << "Enter you grade: ";
cin >> grade;
switch (grade)
{
    case 'A':
        cout << "Excellent.\n";
        cout << "Keep up the good work!\n";
        break;
    case 'B':
        cout << "Very good.\n";
        break;
    case 'C':
        cout << "OK.\n";
        break;
    case 'P':
    case 'F':
    cout << "Not good.\n";
    cout << "Go study!\n";
    break;
    default:
    cout << "That is not a possible grade.\n";</pre>
```

The Controlling Expression

A switch statement's controlling expression must return a value of one of these types:

- A character
- An integer type
- A bool value
- An enum constant (to be described later if we have time)

The value returned is compared to the *constant* values after each "case"

• When a match is found, the code for that case is used

52

The break Statement

The **break** statement

- terminates the execution of the **switch** statement (or a loop statement to be studied later)
- continues with the statements after the switch statement (or a loop statement)

Omitting the **break** statement in a branch of the **switch** statement will cause the code for the next case to be executed!

```
x=0;
switch (x)
{
    case 0: cout << "Hello\n";
    case 1: cout << "Goodbye\n";
}</pre>
```

What is printed?

The break Statement (Cont'd)

The benefit of this "fall-through" by omitting break:

```
    Allow the use of the same code for multiple cases:
    case 'F':
```

case 'F':
 cout << "Not good.\n";
 cout << "Go study!\n";
 break;</pre>

- Another example :
 - case 'A':
 case 'a':
 cout << "Excellent.";
 break;</pre>
 - Runs the same code for either 'A' or 'a'

.

The default Branch

If no case label has a constant that matches the value of the controlling expression, the statements following the default label are executed

The default branch is optional

- If there is no default branch, nothing happens when no case label matches the value of the controlling expression
- It is a good idea to include a default section

55

Case Labels in Switch

All cases must be:

- unique (cannot duplicate cases)
- a constant expression
 - case 2 is ok
 - case 2*3: is ok.
 - case C: is ok if C is a named constant (e.g., defined by const int C=8)
 - case C+1: is ok if C is a named constant
 - case 2*x: is invalid if x is a variable

56

Exercise 7

Write a C++ program that

- prompts the use to enter the wattage of a bulb
- outputs the expected brightness of a standard light bulb with the inputted wattage, according to the following table.

Watts	Brightness (in Lumens)
15	125
25	215
40	500
60	880
75	1000
100	1675

7

Solution to Exercise 7

Outline

Control Flow

Two-way Branch

• if-else statement

Boolean Expressions

Multi-way Branches

- Nested if-else statement
- switch statement

Programming Style

Lab exercises (mixed with the above contents)

59

Program Style

A program written with attention to style

- is easier to read
- easier to correct
- easier to change

Program Style - Indenting

Items considered a group should look like a group

- Use an empty line between logical groups of statements
- Indent statements within statements

```
if (x == 0)
    statement;
```

Braces {} create groups

- Indent within braces to make the group clear
- Braces placed on separate lines are easier to locate

61

Program Style - Comments

Comments are explanatory notes in the program for the programmer to read.

Two ways to add comments in C++ programs:

- Use // for a single line comment
- Use /* and */ for multiple line comments

62

Program Style - Comments

// is the symbol for a single line comment

- All text on the line following // is ignored by the compiler
- Example:

```
//calculate regular wages
gross_pay = rate * hours;
or put the comments at the end of a line:
gross_pay = rate * hours; //calculate regular wages
```

/* and */ enclose multiple line comments

• Example:

/* This is a program that displays
the expected brightness of a
standard light bulb with an
inputted wattage.
*/

Exercise 8

Write a C++ program that

- asks the users to enter 5 numeric marks in the range of [0, 100].
- counts and outputs the number of input marks that are at least 60.

```
More on Comments

/* This is a valid comment */

/* So
    is this.

*/

/* This works
    * and looks nice
    */

/* This doesn't do
    /* what you think it would do */

*/
```