Outline

- Flow control and branch
- Boolean expression
  - Comparison operators
  - Boolean operations: &&, ||, !
- Loop: to repeat statements
  - While loop
  - Do/while loop
  - Infinite loop
- Programming Style
  - Comments, indentations, named constant

Flow of Control

- **Flow of control:** order in which statements are executed
- So far, our program exits from the beginning of main() function to the end or until it reaches a return statement

```
int main()
{
      int n;
      cout <<
       cin >>
      //some calculation…
      cout << …
      return 0;

}
```

Calculate wages for hourly workers

- Imagine a program that
  - Reads number of hours someone works that week, hourly rate
  - Calculates pay as follows:
    - If works for up to 40 hours: hourly rate * number of hours
    - If works overtime (over 40 hours): hourly rate * 40 hours, and those over 40 hours are paid with .5 higher rate (i.e., 1.5* hourly rate * overtime hours)
  - And outputs the pay

- Can you write expression for second case? Assuming:
  - **int hours;**
  - **double rate;**

Calculate wages for hourly workers

- Calculates pay based upon overtime or not
- Requires the program to choose between two alternative statements:
  - Use first statement if hours is no more than 40
  - Use second statement otherwise
- **Branch**: allow program choose between two alternatives

Implement Branch

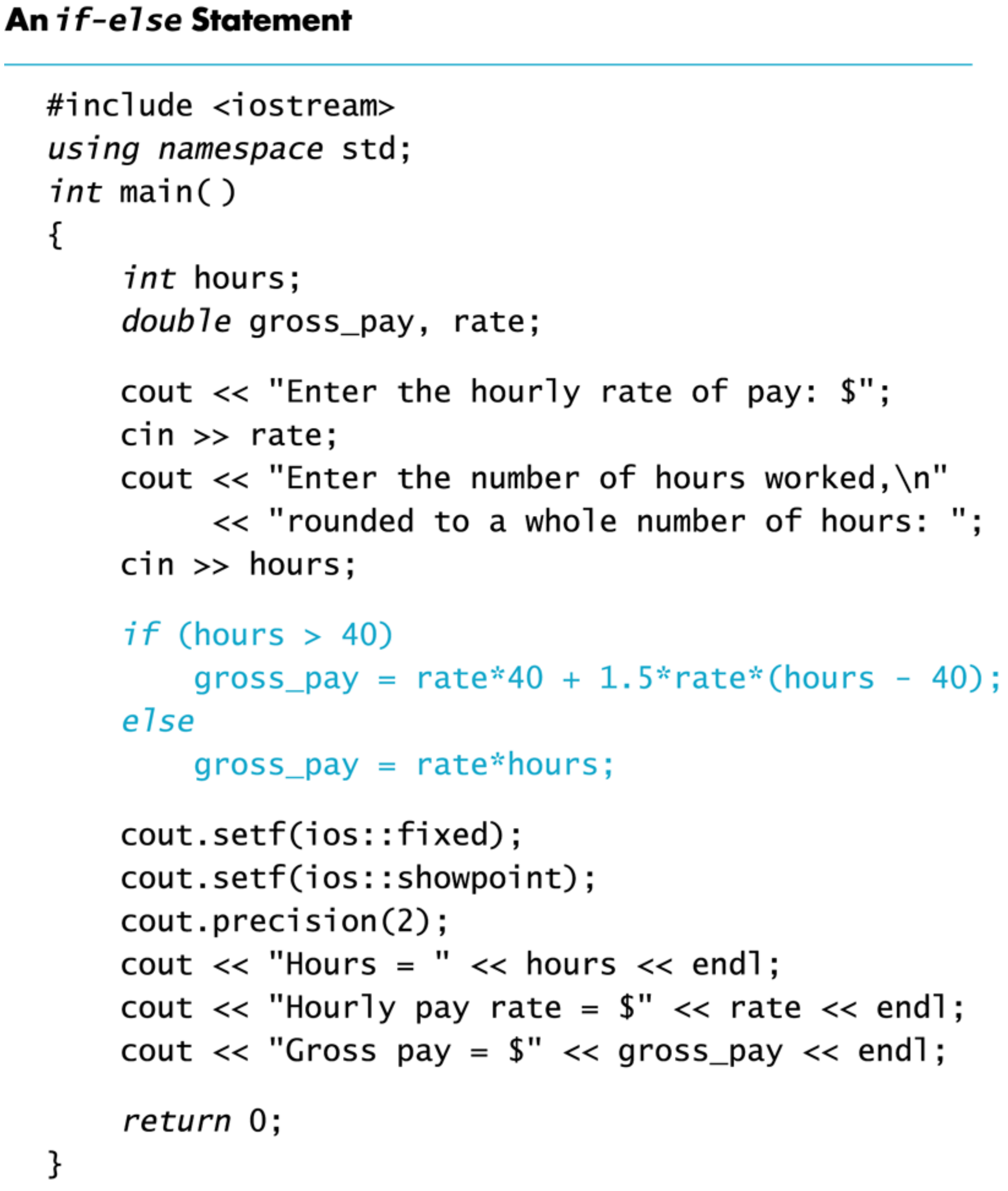
- if-else statement is used in C++ to perform a branch

```
if (hours > 40)
  wage =  rate * 40 + 1.5 * rate * (hours - 40);
else
   wage = rate * hours;
```

What happens here?

if-else statement

- if (boolean expression)
  ```
      statement1
  else
      statement2
  ```
- When boolean expression is true
  - Only statement1 is executed
- When boolean expression is false
  - Only statement2 is executed



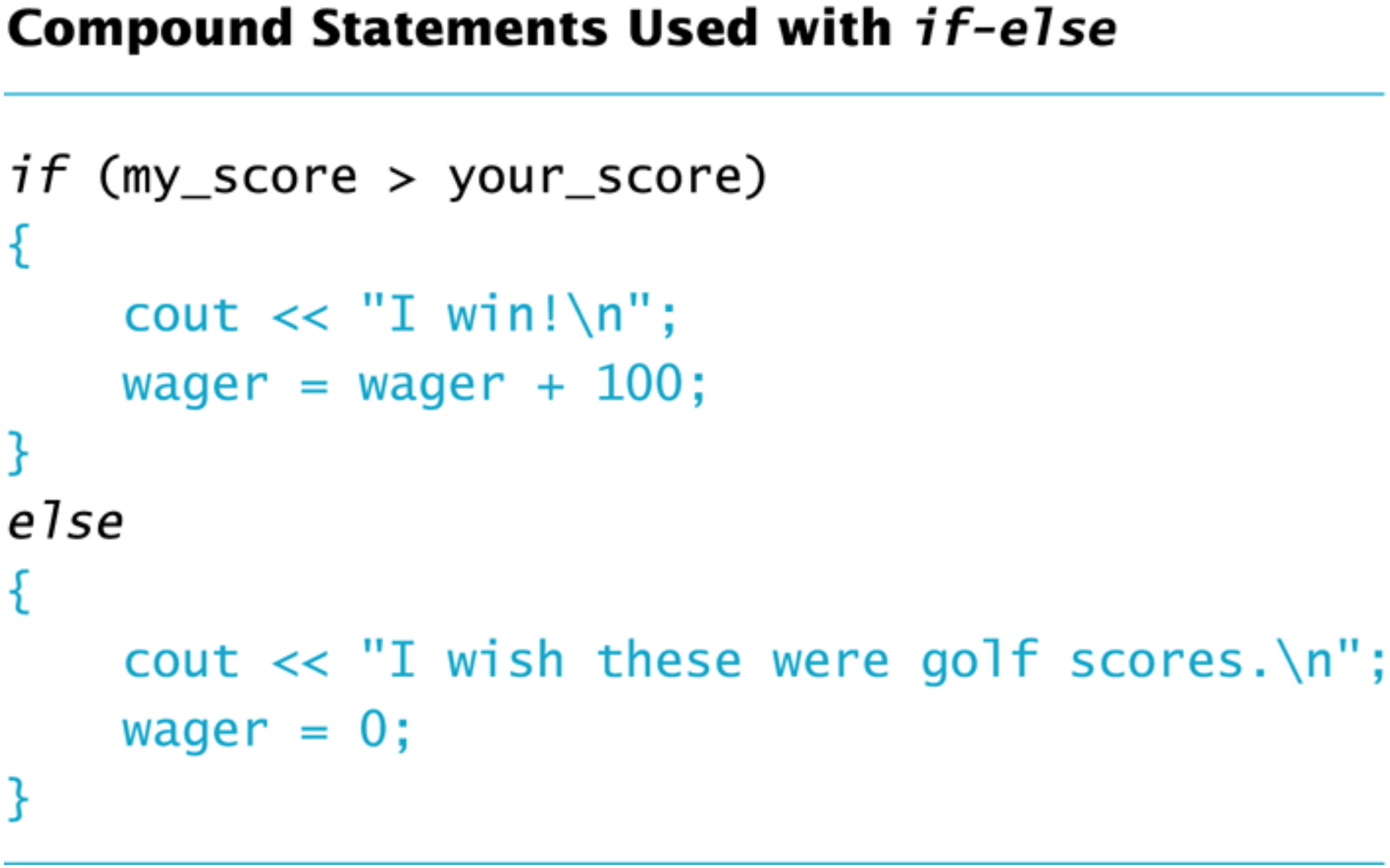
An *if-else* Statement

```
#include <iostream>
using namespace std;
int main( )
{
    int hours;
    double gross_pay, rate;

    cout << "Enter the hourly rate of pay: $";
    cin >> rate;
    cout << "Enter the number of hours worked,\n"
         << "rounded to a whole number of hours: ";
    cin >> hours;

    if (hours > 40)
        gross_pay = rate*40 + 1.5*rate*(hours - 40);
    else
        gross_pay = rate*hours;

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "Hours = " << hours << endl;
    cout << "Hourly pay rate = $" << rate << endl;
    cout << "Gross pay = $" << gross_pay << endl;

    return 0;
}
```

Compound Statements

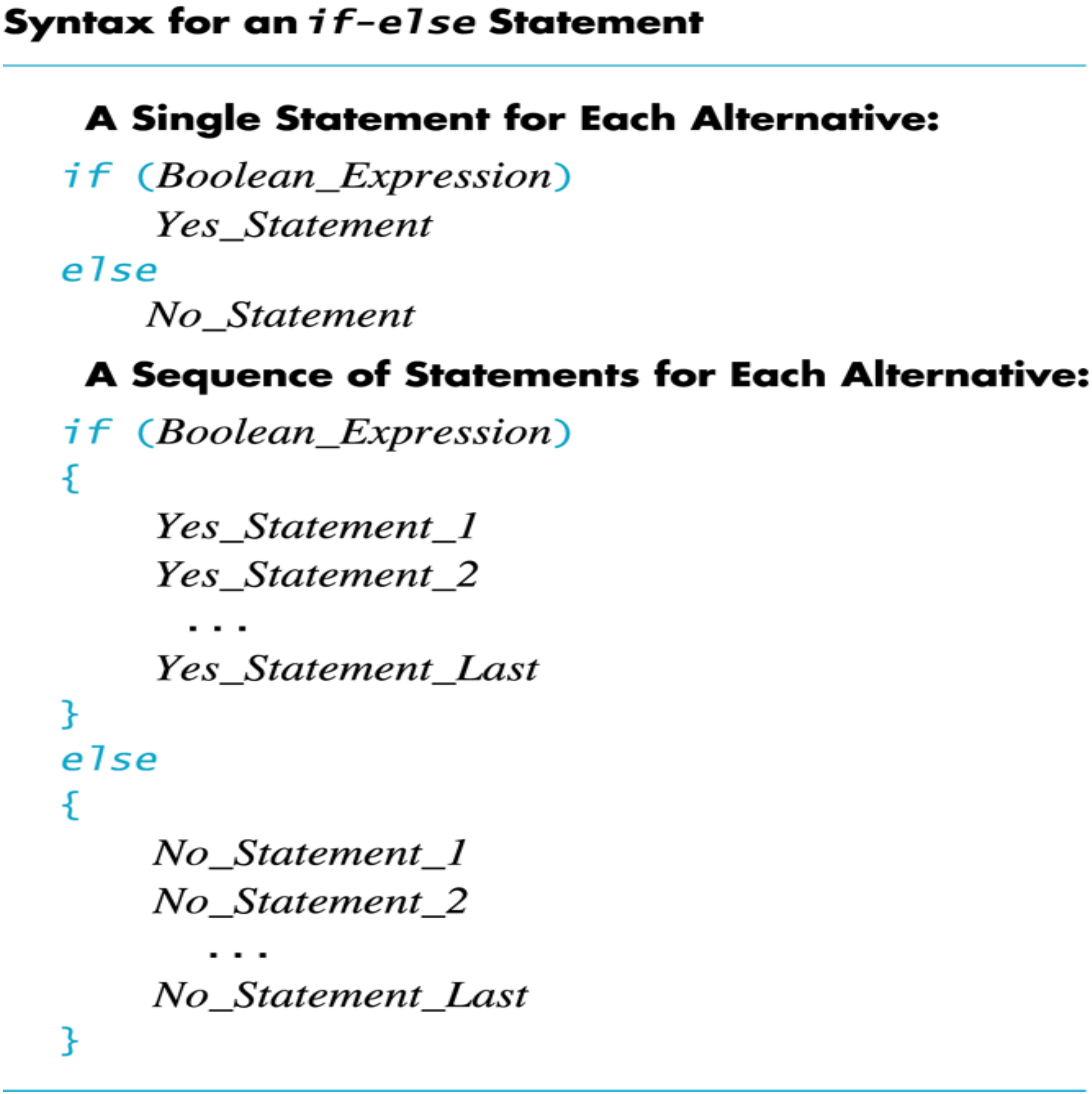
- **Compound Statement:** one or more statements enclosed in { }
  - Branches of if-else statements often need to execute more that one statement
- if (boolean expression)

```
{
       statement 1;
       ….
      statement n;
}
else
 {
     statement1;
      …
 }
```



Compound Statements Used with *if-else*

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



Syntax for an if-else Statement

A Single Statement for Each Alternative:

```
if (Boolean_Expression)
     Yes_Statement
else
     No_Statement
```

A Sequence of Statements for Each Alternative:

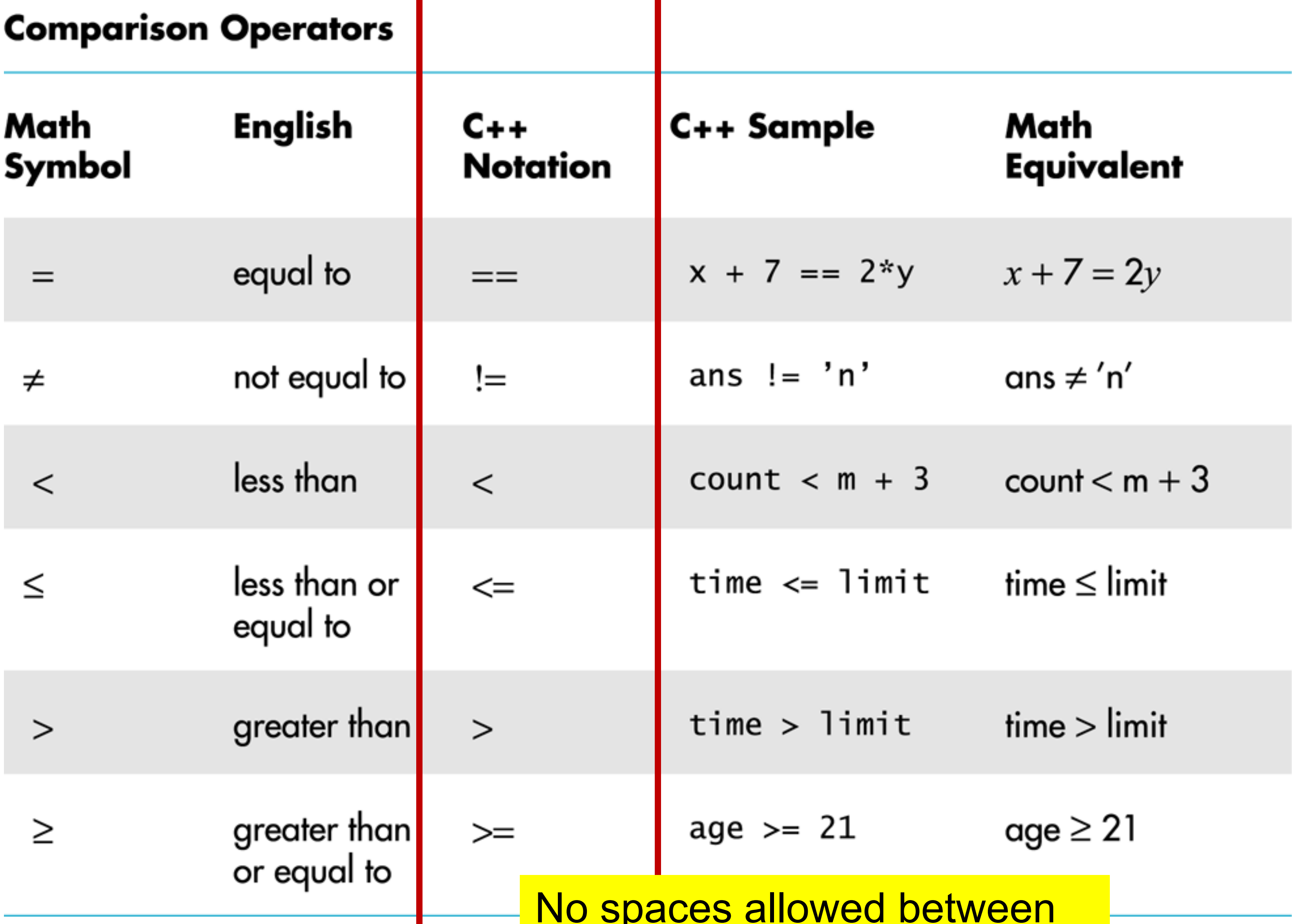
```
if (Boolean_Expression)
{
     Yes_Statement_1
     Yes_Statement_2
       . . .
     Yes_Statement_Last
}
else
{
     No_Statement_1
     No_Statement_2
       . . .
     No_Statement_Last
}
```

Outline

- Flow control and branch
- Boolean expression
  - Comparison operators
  - Boolean operations: &&, | |, !
- Loop: to repeat statements
  - While loop
  - Do/while loop
  - Infinite loop
- Programming Style
  - Comments, indentations, named constant

Boolean Expressions

- **Boolean expressions:** expressions that are either true or false, i.e., have a bool value
- Comparison operators: used to compare variables and/or numbers, and generate a bool value
  - e.g., (hours > 40) evaluates to true if value of hours is greater than 40; otherwise evaluates to false
  - e.g., (choice==‘q’) evaluates to true if choice is equal to ‘q’; otherwise, evaluates to false



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

No spaces allowed between the two symbols

Branches Exercise

- Can you
  - Write an if-else statement that outputs the word High if the value of the variable score is greater than 100 and Low if the value of score is at most 100? The variables are of type int.
  - Write an if-else statement that outputs the word Warning provided that either the value of the variable temperature is greater than or equal to 100, or the of the variable pressure is greater than or equal to 200, or both. Otherwise, the if_else sttement outputs the word OK. The variables are of type int.

Boolean operations

- **Arithmetic operations** are applied to numerical variables or constants:
  - Five operations: +, -, *, /, %
  - E.g., (f – 32) * 5.0 / 9.0   // to convert Farenhite degree to celsius degree
- Boolean Operations: applied to boolean expressions or variables
  - Three operations: &&, | |, !
  - E.g., (x > y && x > z)
  - E.g., !(x==y)

AND operation

- &&   -- AND operator
    - Syntax: (boolean_exp1) && (boolean_exp2)
    - True if both boolean expressions are true
- e.g: ((2 < x) && (x < 7))
    - True only if x is between 2 and 7
    - Inside parentheses are optional but enhance meaning

Operand_1	Operand_2	Operand1 && Operand2
True	True	true
True	False	True
False	True	False
False	False	false

OR

- | | -- The OR operator
  - Syntax: (bool_exp_1) | | (bool_exp_2)
  - True if either or both expressions are true
- e.g: if ((x = = 1) | | (x = = y))
  - True if x contains 1
  - True if x contains same value as y
  - True if both comparisons are true

Operand1	Operand2	Operand1 | | Operand2
True	True	True
True	False	True
False	True	True
False	False	False

NOT: negation

- ! -- negates any boolean expression
  - !(x < y)
    - True if x is NOT less than y
  - !(x = = y)
    - True if x is NOT equal to y
- ! Operator can make expressions difficult to understand …use only when appropriate

Operand	!operand
True	False
False	true

Boolean Expression Practices

- How to check for errors in user input?
  - e.g., In hourly worker program, hours should not be negative, should not be over 84
  - How to report such errors?

- Be careful translating inequalities to C++
- if x < y < z translates as

```
if ( ( x < y ) && ( y < z ) )
```

NOT

```
if ( x < y < z )
```

Pitfall: Using = or ==

- ' = ' is the **assignment operator**
  - Used to assign values to variables
  - x = 3;
- '= = ' is **equality operator**
  - Used to compare values
  - if (x == 3)
- The compiler will accept this :

  if (x = 3)

  but stores 3 in x instead of comparing x and 3
  - Since the result is 3 (non-zero), the expression is true

Outline

- Flow control and branch
- Boolean expression
  - Comparison operators
  - Boolean operations: &&, | |, !
- Loop: to repeat statements
  - While loop
  - Do/while loop
  - Infinite loop
- Programming Style
  - Comments, indentations, named constant

Simple Loops

- When an action must be repeated, a **loop** is used
  - C++ includes several ways to create loops
- **while-loop**

e.g.,

```
while (count_down > 0)
{
        cout << "Hello ";
        count_down  -= 1;
 }
```

Output: Hello Hello Hello

when count_down starts at 3

A *while* Loop

```
#include <iostream>
using namespace std;
int main( )
{
    int count_down;

    cout << "How many greetings do you want? ";
    cin >> count_down;

    while (count_down > 0)
    {
        cout << "Hello ";
        count_down = count_down - 1;
    }

    cout << endl;
    cout << "That's all!\n";

    return 0;
}
```

Sample Dialogue 1

```
How many greetings do you want? 3
Hello Hello Hello
That's all!
```

Sample Dialogue 2

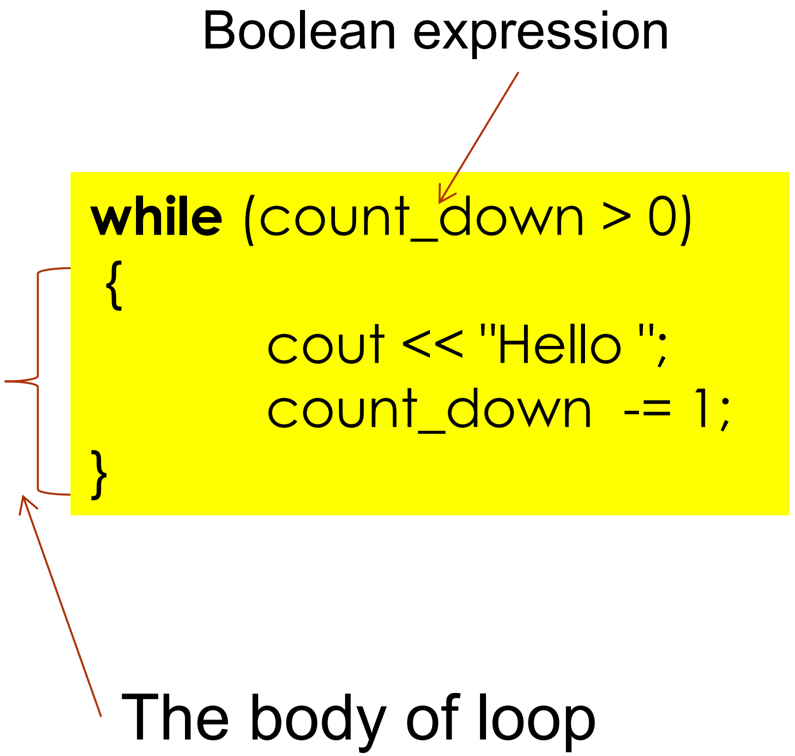
```
How many greetings do you want? 1
Hello
That's all!
```

Sample Dialogue 3

```
How many greetings do you want? 0

That's all!
```

The loop body is executed zero times.

While Loop Operation



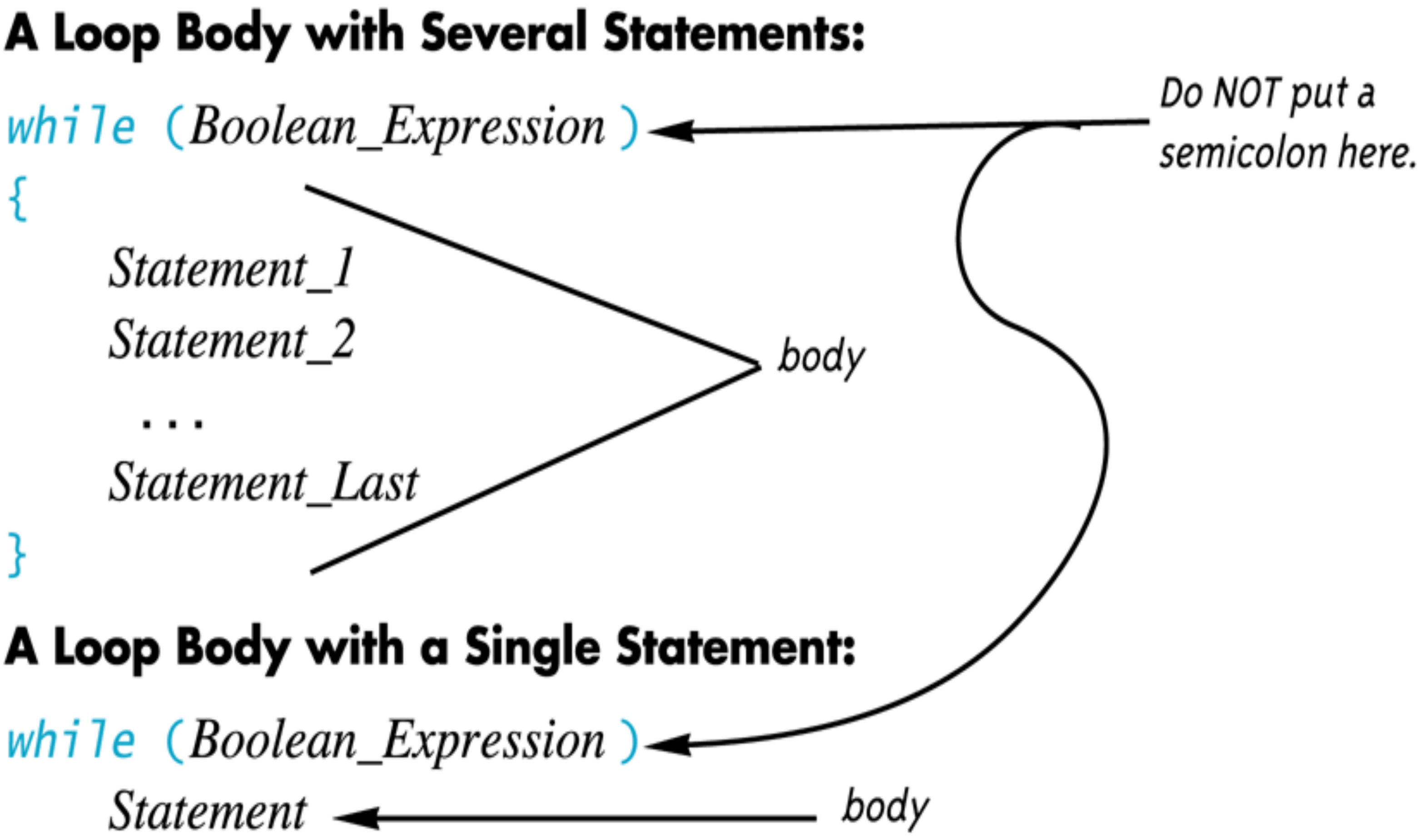
Boolean expression

```
while (count_down > 0)
 {
        cout << "Hello ";
        count_down  -= 1;
}
```

The body of loop

- First, boolean expression is evaluated
  - If false, program skips to line following while loop
  - If true, body of loop is executed
    - During execution, some item from boolean expression
      is changed
- After executing loop body, boolean expression is checked again repeating process until expression becomes false
- A while loop might not execute at all if boolean expression is false on the first check

Syntax of the *while* Statement



A Loop Body with Several Statements:

```
while (Boolean_Expression )
{
      Statement_1
      Statement_2
        . . .
      Statement_Last
}
```

body

Do NOT put a semicolon here.

A Loop Body with a Single Statement:

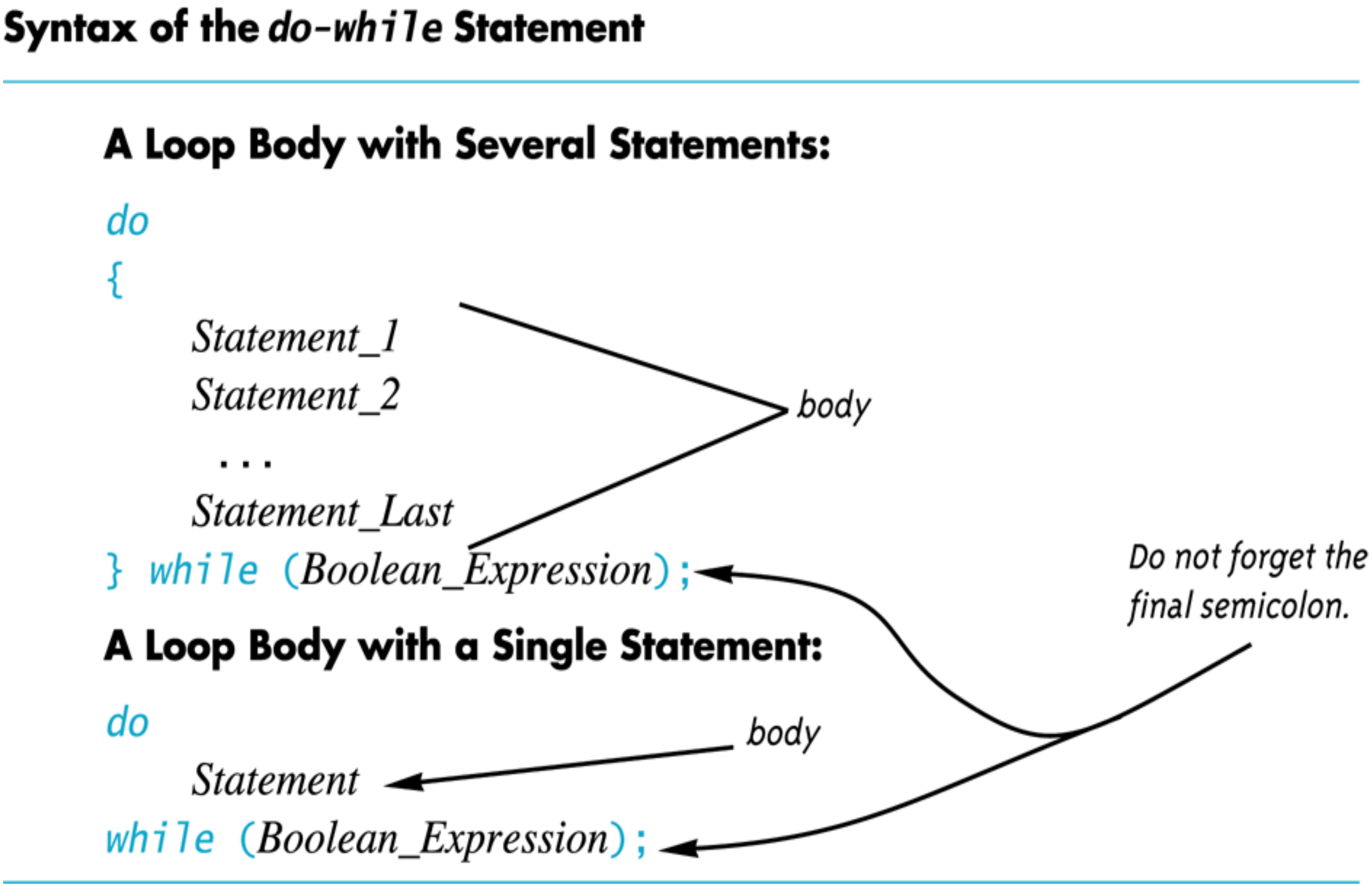
```
while (Boolean_Expression )
      Statement
```

body

do-while loop

- A do-while loop is always executed at least once
  - body of the loop is first executed
  - boolean expression is checked after the body has been executed
- Syntax:

```
do
 {
              statements to repeat
  } while (boolean_expression);
```



Syntax of the *do-while* Statement

A Loop Body with Several Statements:

```
do
{
      Statement_1
      Statement_2
        . . .
      Statement_Last
} while (Boolean_Expression);
```

body

Do not forget the final semicolon.

A Loop Body with a Single Statement:

```
do
      Statement
while (Boolean_Expression);
```

body

A *do-while* Loop

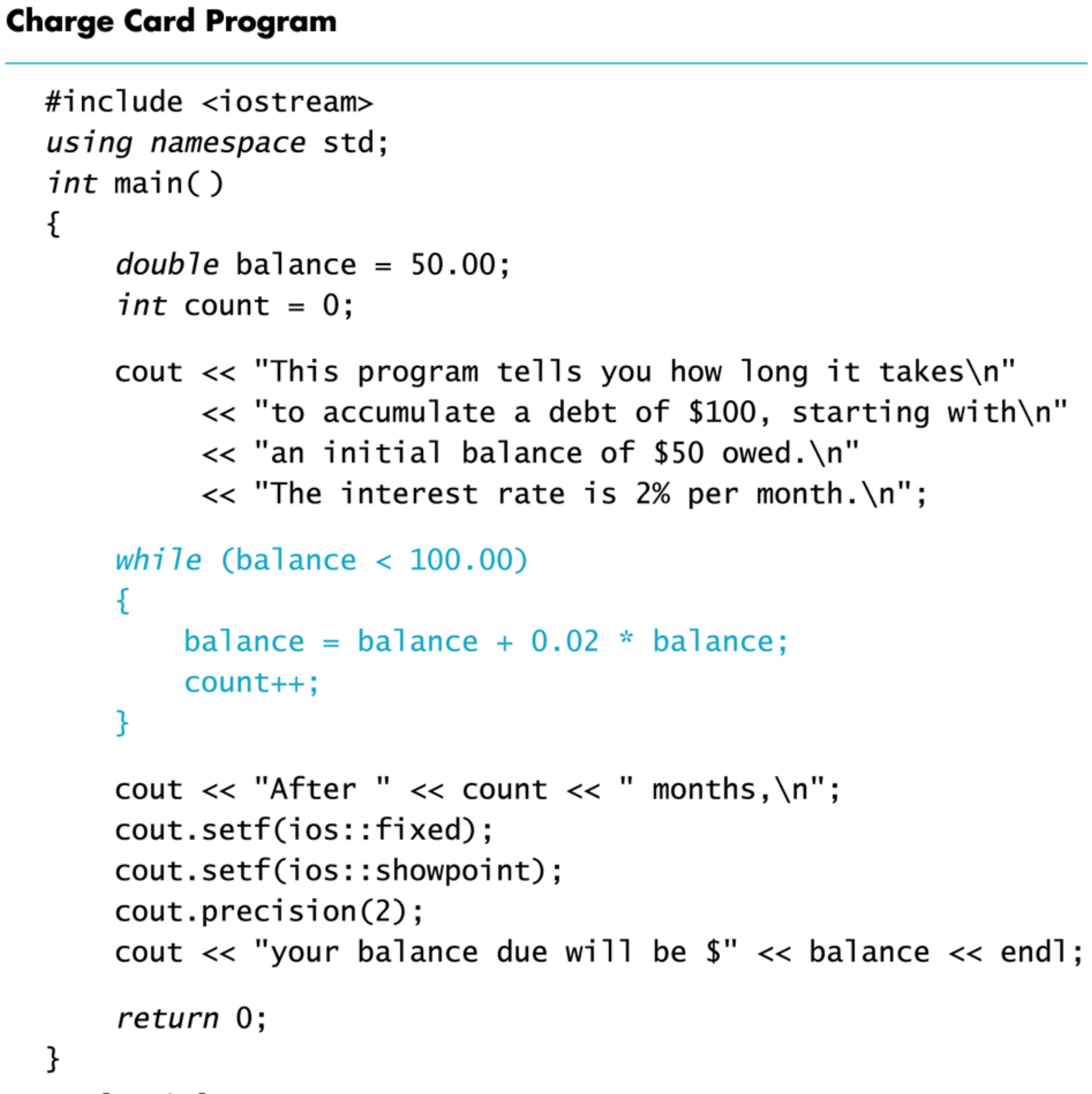
```
#include <iostream>
using namespace std;
int main( )
{
    char ans;

    do
    {
        cout << "Hello\n";
        cout << "Do you want another greeting?\n"
             << "Press y for yes, n for no,\n"
             << "and then press return: ";
        cin >> ans;
    } while (ans == 'y' || ans == 'Y');

    cout << "Good-Bye\n";

    return 0;
}
```

Sample Dialogue

```
Hello
Do you want another greeting?
Press y for yes, n for no,
and then press return: y
Hello
Do you want another greeting?
Press y for yes, n for no,
and then press return: Y
Hello
Do you want another greeting?
Press y for yes, n for no,
```

Sample Program

- Bank charge card balance of $50
- 2% per month interest
- How many months without payments before your balance exceeds $100
- After 1 month: $50 + 2% of $50 = $51
- After 2 months: $51 + 2% of $51 = $52.02
- After 3 months: $52.02 + 2% of $52.02 …



Charge Card Program

```
#include <iostream>
using namespace std;
int main( )
{
    double balance = 50.00;
    int count = 0;

    cout << "This program tells you how long it takes\n"
         << "to accumulate a debt of $100, starting with\n"
         << "an initial balance of $50 owed.\n"
         << "The interest rate is 2% per month.\n";

    while (balance < 100.00)
    {
        balance = balance + 0.02 * balance;
        count++;
    }

    cout << "After " << count << " months,\n";
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "your balance due will be $" << balance << endl;

    return 0;
}
```

Infinite Loops

- **infinite loops**: loops that never stop are
  - loop body should contain a line that will eventually cause boolean expression to become false
- e.g.,: print odd numbers less than 12

```
x = 1;
while (x != 12)
 {
          cout << x << endl;
          x = x + 2;
  }
```

- Better to use this comparison: while (x < 12)

Exercises

- Can you show the output of this code if x is of type int?

```
 x = 10;
while ( x > 0)
   {
         cout << x << endl;
         x = x – 3;
    }
```

  - Show the output of the previous code using the comparison x < 0 instead of x > 0?

Outline

- Flow control and branch
- Boolean expression
  - Comparison operators
  - Boolean operations: &&, | |, !
- Loop: to repeat statements
  - While loop
  - Do/while loop
  - Infinite loop
- Programming Style
  - Comments, indentations, named constant

Program Style

- A program written with attention to style
  - is easier to read
  - easier to correct
  - easier to change
- Indentations, Comments, Named Constants

Program Style - Indenting

- Items considered a group should look like a group
  - Skip lines 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

Program Style - Comments

- // is the symbol for a single line comment
  - Comments are explanatory notes for the programmer
  - All text on the line following // is ignored by the compiler
  - Example:

    ```
    //calculate regular wages
    gross_pay = rate * hours;
    ```
- /* **and** */ enclose multiple line comments
  - Example:

    ```
    /* This is a comment that spans
       multiple lines without a
       comment symbol on the middle line
    */
    ```

Program Style - Constants

- Number constants have no mnemonic value
- Number constants used throughout a program are difficult to find and change when needed
- Constants
  - Allow us to name number constants so they have meaning
  - Allow us to change all occurrences simply by changing the value of the constant

Constants

- const is the keyword to declare a constant
- Example:

  ```
  const int WINDOW_COUNT = 10;
  ```

  declares a constant named WINDOW_COUNT

  - Its value cannot be changed by program like a variable
  - It is common to name constants with all capitals

Exercises

- Can you
  - Create a named constant of type double?
  - Determine if a program can modify the value of a constant?
  - Describe the benefits of comments?
  - Explain why indenting is important in a program?
  - Explain why blank lines are important in a program?

Summary

- Flow control and branch
- Boolean expression
  - Comparison operators
  - Boolean operations: &&, | |, !
- Loop: to repeat statements
  - While loop
  - Do/while loop
  - Infinite loop
- Programming Style
  - Comments, indentations, named constant