

CSC 211: Computer Programming

Expressions and Selection Statements

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Spring 2024



Original design and development by Dr. Marco Alvarez

Expressions

Common arithmetic operators



- Can be used with any numeric type (integers and floating point numbers)
- Result of the **operator** depends on the type of the **operands**
- Be aware of the **integer division** (fractional part discarded)
 - ✓ $22/4$ is 5

Integer Division

$$\begin{array}{r} 4 \longleftarrow 12/3 \\ 3 \overline{) 12} \\ \underline{12} \\ 0 \longleftarrow 12\%3 \end{array} \qquad \begin{array}{r} 4 \longleftarrow 14/3 \\ 3 \overline{) 14} \\ \underline{12} \\ 2 \longleftarrow 14\%3 \end{array}$$

“Rules”

- Use parentheses !
 - ✓ even when redundant
- Use whitespaces !

$(b * b - 4 * a * c) / (2 * a)$ 🙄

$((b * b) - (4 * a * c)) / (2 * a)$ 👍

5

Boolean expressions

- Expressions that evaluate to either **true** or **false**

- Can use comparison operators

== **<** **>** **>=** **<=** **!=**

- Can use logical operators

! **&&** **||**

6

Truth Tables

AND

Exp_1	Exp_2	Exp_1 && Exp_2
true	true	true
true	false	false
false	true	false
false	false	false

OR

Exp_1	Exp_2	Exp_1 Exp_2
true	true	true
true	false	true
false	true	true
false	false	false

NOT

Exp	!(Exp)
true	false
false	true

Comparison Operators

Math Symbol	English	C++ Notation	C++ Sample	Math Equivalent
=	equal to	==	<code>x + 7 == 2*y</code>	$x + 7 = 2y$
≠	not equal to	!=	<code>ans != 'n'</code>	$\text{ans} \neq 'n'$
<	less than	<	<code>count < m + 3</code>	$\text{count} < m + 3$
≤	less than or equal to	<=	<code>time <= limit</code>	$\text{time} \leq \text{limit}$
>	greater than	>	<code>time > limit</code>	$\text{time} > \text{limit}$
≥	greater than or equal to	>=	<code>age >= 21</code>	$\text{age} \geq 21$

Precedence Rules

The unary operators +, -, ++, --, and !.

The binary arithmetic operations *, /, %

The binary arithmetic operations +, -

The Boolean operations <, >, <=, >=

The Boolean operations ==, !=

The Boolean operations &&

The Boolean operations ||

*Highest precedence
(done first)*



*Lowest precedence
(done last)*

from: Problem Solving with C++, 10th Edition, Walter Savitch

9

What is the value of this expression?

x = 5

(x + 1) > 2 || (x + 1) < -3

Recommended style

((x + 1) > 2) || ((x + 1) < -3)

10

In C++ any non-zero value is **true**
and zero is **false**

11

What is the value of this expression?

false
(! 32 > 64)

(0 > 64)

false

12

What is the value of this expression?

true
(! 0 > 64)

(1 > 64)

false

13

What is the value of this expression?

```
a = 0;   b = 1;   c = 15;   d = 5;   e = 20;  
( !b && !!c ) || ( d == e ) || ( !a && ((d + e) % 10 == 0) );
```

14

Selection Statements if and switch

if statements

- Allow conditional execution of code
- General idea:

```
if (expression)  
    true statement  
else  
    false statement
```

16

The if statement (basic syntax)

```
if (expression)
    statementA

if (expressionA)
    statementA
else if (expressionB)
    statementB

if (expression)
    statementA
else
    statementB

if (expression)
    statementA
else if (expressionB)
    statementB
else if (expressionC)
    statementC
else
    statementN
```

17

Example

```
int value;

std::cout << "Enter a number: ";
std::cin >> value;

if (value > 0) {
    std::cout << "positive number" << std::endl;
} else if (value < 0) {
    std::cout << "negative number" << std::endl;
} else {
    std::cout << "zero" << std::endl;
}
```

18

Compound statements

```
if (expression) {
    statementA
    statementB
    statementC
    ...
} else {
    statementL
    statementM
    statementN
    ...
}
```

- ✓ Recommended to **always use braces**, even with single statements
- ✓ Develop a good and consistent programming style

19

Compound statements

```
2 #include <iostream>
3
4 int main( )
5 {
6     double fuelGaugeReading;
7
8     std::cout << "Enter fuel gauge reading: ";
9     std::cin >> fuelGaugeReading;
10
11     std::cout << "First with braces:\n";
12     if (fuelGaugeReading < 0.75)
13     {
14         if (fuelGaugeReading < 0.25)
15             std::cout << "Fuel very low. Caution!\n";
16     }
17     else
18     {
19         std::cout << "Fuel over 3/4. Dont stop now!\n";
20     }
21
22     std::cout << "Now without braces:\n";
23     if (fuelGaugeReading < 0.75)
24     if (fuelGaugeReading < 0.25)
25         std::cout << "Fuel very low. Caution!\n";
26     else
27         std::cout << "Fuel over 3/4. Don't stop now!\n";
28
29     return 0;
30 }
```

20

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;
}
```

from: Problem Solving with C++, 10th Edition, Walter Savitch

21

An *if-else* Statement within an *if* Statement

```
if (count > 0)
{
    if (score > 5)
    {
        cout << "count > 0 and score > 5\n";
    }
    else
    {
        cout << "count > 0 and score <= 5\n";
    }
}
```

from: Problem Solving with C++, 10th Edition, Walter Savitch

22

switch statements

- Allow conditional execution of code based on the value of an **integer** expression

- Basic syntax:

```
switch (expression) {
    case valueA:
        statementA
    case valueB:
        statementB
    .
    .
    case valueN:
        statementN
    default:
        statement
}
```

if expression equals to a value, control executes corresponding statement (can be a compound statement), then continue executing statements until **break** is encountered

23

switch statements

```
3  #include <iostream>
4  int main() {
5      int x = 2;
6      switch (x)
7      {
8          case 1:
9              std::cout << "Choice is 1 \n";
10             break;
11          case 2:
12              std::cout << "Choice is 2 \n";
13              break;
14          case 3:
15              std::cout << "Choice is 3 \n";
16              break;
17          default:
18              std::cout << "Choice other than 1, 2 and 3 \n";
19              break;
20      }
21      return 0;
22  }
```

24

switch statements

```
4 int main() {
5     int x = 2;
6     switch (x)
7     {
8     case 1:
9         std::cout << "Choice is 1 \n";
10        //break;
11    case 2:
12        std::cout << "Choice is 2 \n";
13        //break;
14    case 3:
15        std::cout << "Choice is 3 \n";
16        //break;
17    default:
18        std::cout << "Choice other than 1, 2 and 3 \n";
19        //break;
20    }
21    return 0;
22 }
```

25

A switch Statement (part 1 of 2)

```
//Program to illustrate the switch statement.
#include <iostream>
using namespace std;

int main()
{
    char grade;

    cout << "Enter your midterm grade and press Return: ";
    cin >> grade;

    switch (grade)
    {
        case 'A':
            cout << "Excellent. "
                << "You need not take the final.\n";
            break;
        case 'B':
            cout << "Very good. ";
            grade = 'A';
            cout << "Your midterm grade is now "
                << grade << endl;
            break;
        case 'C':
            cout << "Passing.\n";
            break;
        case 'D':
            cout << "Not good. Go study.\n";
            break;
        case 'F':
            cout << "Not good. "
                << "Go study.\n";
            break;
        default:
            cout << "That is not a possible grade.\n";
    }

    cout << "End of program.\n";
    return 0;
}
```

characters (ascii values) can also be used in switch statements

A switch Statement (part 2 of 2)

Sample Dialogue 1

Enter your midterm grade and press Return: A
Excellent. You need not take the final.
End of program.

Sample Dialogue 2

Enter your midterm grade and press Return: B
Very good. Your midterm grade is now A.
End of program.

Sample Dialogue 3

Enter your midterm grade and press Return: D
Not good. Go study.
End of program.

Sample Dialogue 4

Enter your midterm grade and press Return: E
That is not a possible grade.
End of program.

from: Problem Solving with C++, 10th Edition, Walter Savitch

26

Exercise

- Write a program in C++ (**on paper**) that:
 - reads the number of **hours**
 - calculates payment:
 - if number of hours no greater than 40, **payment** is calculated using the regular hourly rate of \$35
 - if overtime, **payment** is calculated using the regular hourly rate for the first 40 hours and the special rate of \$50 for the remaining hours
 - prints the calculated **payment**

27