CS4002
Applied Programming

National University of Computer & Emerging Sciences
Islamabad, Pakistan

Ms. Umarah Qaseem

1

Input value

- Input Source?
  - Console
  - Data File
- Console Input
  - cin >>
  - In the header file “iostream”

2

Arithmetic operators

- Arithmetic calculations
  - *
    Multiplication
  - /
    Division
    Integer division truncates remainder
    7 / 5 evaluates to 1
  - %
    Modulus operator returns remainder
    7 % 5 evaluates to 2
  - + and −
    Addition and Subtraction

3

Operator precedence

- Operator precedence determines the order in which operations are performed within an expression. Operators with higher precedence are evaluated before those with lower precedence.

4

Operator precedence

- Rules of operator precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
()	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
*, /, or %	Multiplication Division Modulus	Evaluated second. If there are several, they re evaluated left to right.
+ or -	Addition Subtraction	Evaluated third. If there are several, they are evaluated left to right.
=	Assignment	Evaluated last, right to left

5

Arithmetic operators

- Priority of operators
  - `a = 5 + 7 % 2;`
  - we may doubt if it really means:
    - `a = 5 + (7 % 2)` with result 6 or
    - `a = (5 + 7) % 2` with result 0
  - Parentheses are included when one is not sure

6

Activity: Operator precedence

Take out your notebook!

- Given integer variables a, b, c, d, and e, where a = 1, b = 2, c = 3, d = 4,
- Evaluate the following expressions:

```
a + b - c + d
a * b / c
1 + a * b % c
a + d % b - c
e = b = d + c / b - a
```

7

Arithmetic operators

- Arithmetic Assignment Operators
  - `a = a + b;`
  - `a += b;`

```
void main(void)
{
        int number = 15;
        number +=10;
        cout << number << endl;
        number -=7;
        cout << number << endl;
        number *=2;
        cout << number << endl;
        number %=2;
        cout << number << endl;
}
```

```
25
18
36
0
```

8

Arithmetic operators

- Increment Operators (Unary Operator)
  - `count = count + 1;`
  - `count +=1;`
  - `count++;  OR ++count;`

```
int a = 5;
int b = 10;
int c = a * b++;
```

50

```
int a = 5;
int b = 10;
int c = a * ++b;
```

55

9

Arithmetic operators

- Decrement Operators (Unary Operator)
  - `count = count - 1;`
  - `count -=1;`
  - `count--;  OR --count;`

postfix prefix

10

Arithmetic operators

```
void main()
{
        int count = 10;

        cout << count << endl;
        cout << ++count << endl;
        cout << count << endl;
        cout << count++ << endl;
        cout << count << endl;
}
```

```
10
11
11
11
12
```

11

Relational operators

- To evaluate comparison between two expressions
  - Result : True / False

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	**>**	**x > y**	**x** is greater than **y**
<	**<**	**x < y**	**x** is less than **y**
≥	**>=**	**x >= y**	**x** is greater than or equal to **y**
≤	**<=**	**x <= y**	**x** is less than or equal to **y**
Equality operators			
=	**==**	**x == y**	**x** is equal to **y**
≠	**!=**	**x != y**	**x** is not equal to **y**

12

Relational operators

- Evaluate the following expressions and write answers in terms of true and false
- Examples
  - (7 == 5)would return false
  - (3 != 2) would return true
  - (6 >= 6)would return true
- If a=2, b=3 and c=6
  - (a*b >= c) would return true since it is (2*3 >= 6)
  - (b+4 > a*c) would return false since it is (3+4 > 2*6)
  - ((b=2) == a) would return true

13

Relational operators

- Examples
  - (7 == 5)would return
  - (3 != 2) would return
  - (6 >= 6)would return
- If a=2, b=3 and c=6
  - (a*b >= c) would return
  - (b+4 > a*c) would retur
  - ((b=2) == a) would retur

14

Relational operators - characters

- C++ allows character comparison using relational and equality operators.
- During comparison alphabetical order is followed. (ASCII: **American Standard Code for Information Interchange**).
- Examples
  - ‘a’ < ‘e’ // True

15

Equality (==) and Assg (=) Operators

- Common error : Does not cause syntax errors
- Example

```
if ( payCode == 4 )
   cout << "You get a bonus!";
```

- If == was replaced with =

```
if ( payCode = 4 )
   cout << "You get a bonus!";
```

  - PayCode set to 4 (no matter what it was before)
  - Statement is true (since 4 is non-zero)
  - Bonus given in every case

16

Logical operators

- **Logical expressions** - expressions that use conditional statements and logical operators.
  - && (And)
    - A && B is true if and only if both A and B are true
  - || (Or)
    - A || B is true if either A or B are true
  - ! (Not)
    - !(condition) is true if condition is false, and false if condition is true
    - This is called the **logical complement** or **negation**
    - Operator ! is equivalent to Boolean operation NOT

17

Logical operators

- Examples
  - (salary < 10000) || (dependants > 5)
  - (temperature > 40.0) && (humidity > 90)
  - !(temperature > 90.0)

18

Truth table (AND)

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

Example: (temperature > 40.0) && (humidity > 90)

19

Truth table (OR)

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

20

Truth table (NOT)

x	x’
0	1
1	0

Truth Table

21

Logical operators – Exercises - Solve

- Examples
  - ! (5 == 5)
  - ! (6 <= 4)
  - ! true
  - ! false
  - ((5 == 5) && (3 > 6))
  - ((5 == 5) || (3 > 6))

22

Logical operators – Exercises - Solve

- Examples
  - ! (5 == 5) returns false
  - ! (6 <= 4) returns true
  - ! true returns false.
  - ! false returns true.
  - ((5 == 5) && (3 > 6)) returns false (true && false)
  - ((5 == 5) || (3 > 6)) returns true (true || false).

23

Conditional operators

- `condition ? result1 : result2`
  - if condition is true the expression will return result1, if not it will return result2
- Examples
  - 7==5 ? 4 : 3 returns 3 since 7 is not equal to 5.
  - 7==5+2 ? 4 : 3 returns 4 since 7 is equal to 5+2
  - a>b ? a : b returns the greater one, a or b

25

Activity 1

- **`condition ? result1 : result2`**
- Task: Print orange if user enters 1 and print Red otherwise.
- Solution
  - int x = 0;
  - Cout<<“Enter Value”
  - Cin>>x;
  - x==1 ?cout<<“Orange” : cout<<“Red”

26

Bitwise Operators

- Bitwise Operators (&, |, ^, ~, <<, >>)

&	**AND**	Logical AND
|	**OR**	Logical OR
^	**XOR**	Logical exclusive OR
~	**NOT**	Complement to one (bit inversion)
<<	**SHL**	Shift Left
>>	**SHR**	Shift Right

27

Example Bitwise Operators

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

int main() {
    // declare variables
    int a = 12, b = 25;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a & b = " << (a & b) << endl;

    return 0;
}
```

```
12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

//Bitwise AND Operation of 12 and 25

     00001100
&    00011001
     ________
     00001000  = 8 (In decimal)
```

Output

```
a = 12
b = 25
a & b = 8
```

28

Type conversion

- Automatic Type Conversion
- Casting

Automatic Type Conversion

```
void main(void)
{
int number = 2;
float factor = 1.5;
double result = number *
factor;
cout << "Result is : " <<
result;
}
```

```
void main(void)
{
short x = 2;
int y = x;
}
```

29

Type conversion

Casting

```
void main(void)
{
short number = 30000;//-32768 to 32767
short result = (number * 10) / 10;
cout << "Result is : " << result;//Result Incorrect
number = 30000;
result = (long(number) * 10) / 10; //Casting
cout << "Result is : " << result;
}
```

30

Activity 2

Task: Convert a number from int to float, using type casting.

```
void main(void)
{
int number = 10;
float result ;
Result = number; //Casting Method 1
cout << "Result is : " << result;
result = (float(number); //Casting Method 2
cout << "Result is : " << result;
}
```

31

Operator precedence

Operator	Description	Associativity
( )	Parentheses and function calls (see Note 1)	left-to-right
+ - ! (*type*)	Unary plus/minus Logical negation Cast (change *type*)	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
=	Assignment	right-to-left

Note 1: Parentheses are also used to group sub-expressions to force a different precedence; such parenthetical expressions can be nested and are evaluated from inner to outer.

32

Example Activity: Solve. Print output

```
int x = 1, y = 2;
int result = y++ + ++x;
cout << result << endl;
cout << x << endl;
cout << y << endl;
```

- **x** gets the value 2.
- **y++** is postfix form so **y** is incremented **after** the execution of this statement
- **2+2 = 4**

- Unary operator has higher precedence and is evaluated right to left
- **Example:**
- **Initial value of a: 6**
- **Value of (-++a): -7**

33

Control Structures

34

34

Outline

- if statement
- Use of if else
- else-if Statement
- Switch statement

35

if, if else and else if Statements

36

Control Structures

- 3 control structures
  - Sequence structure
    - Programs executed sequentially by default
  - Selection structures
    - `if`, `if/else`, `switch`
  - Repetition structures
    - `for`, `while`, `do/while`

37

The If Selection Structure

- Selection structure
  - Choose among alternative courses of action
  - Pseudocode example:

    If student's grade is greater than or equal to 60
    Print "Passed"

    - If the condition is **`true`**
      Print statement executed, program continues to next statement
    - If the condition is **`false`**
      Print statement ignored, program continues

38

The If Selection Structure

- Translation into C++

  If student's grade is greater than or equal to 60
  Print "Passed"

```
if ( grade >= 60 )
   cout << "Passed";
```

39

`if/else` Selection Structure

- **`if`**
  - Performs action if condition true
- **`if/else`**
  - Different actions if conditions true or false
- Pseudocode

  if student's grade is greater than or equal to 60
  print "Passed"
  else
  print "Failed"
- C++ code

```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

40

Activity

- **Rewrite the following code through ternary operator.**
- Pseudocode

  if student's grade is greater than or equal to 60
  print "Passed"
  else
  print "Failed"
- C++ code

```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

41

The If Selection Structure

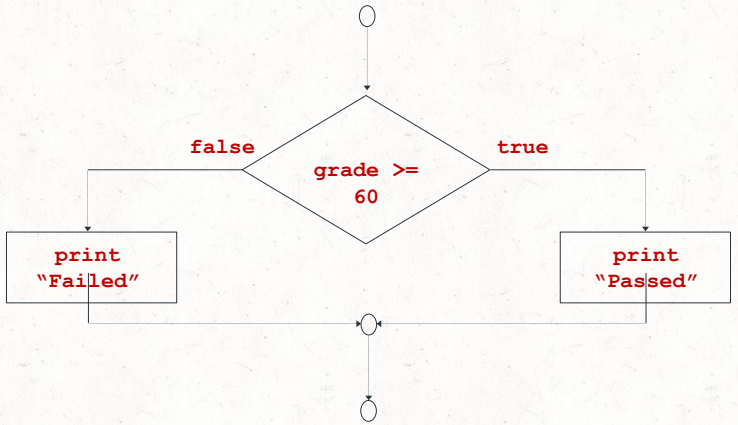
- Ternary conditional operator (**?:**)
  - Three arguments (condition, value if **true**, value if **false**)

- Code could be written:

```
cout << ( grade >= 60 ? "Passed" : "Failed" );
```

42

Understand through flowchart



43

Activity

- **Write code to print "positive" if a user enters a positive number. If user enters negative number than nothing is shown/printed.**

44

Example

```
// Program to print positive number entered by the user
// If user enters negative number, it is skipped

#include <iostream>
using namespace std;

int main()
{
      int number;
      cout << "Enter an integer: ";
      cin >> number;

      // checks if the number is positive
      if ( number > 0)
      {
            cout << "You entered a positive integer: " << number << endl;
      }

      cout << "This statement is always executed.";
      return 0;

}
```

Output 1

```
Enter an integer: 5
You entered a positive number: 5
This statement is always executed.
```

Output 2

```
Enter a number: -5
This statement is always executed.
```

45

The If Selection Structure

- Compound statement
  - Set of statements within a pair of braces
  - Without braces, always executed
- Block
  - Set of statements within braces

```
if ( grade >= 60 )
   cout << "Passed.\n";
else
         {
   cout << "Failed.\n";
   cout << "You must take this course again.\n";
}

cout << "You must take this course again.\n";
```

46

The If Selection Structure

- Nested **if/else** structures
  - One inside another, test for multiple cases
  - Once condition met, other statements skipped

```
if student's grade is greater than or equal to 90
   Print "A"
else
   if student's grade is greater than or equal to 80
            Print "B"
   else
      if student's grade is greater than or equal to 70
            Print "C"
      else
         if student's grade is greater than or equal to 60
            Print "D"
         else
            Print "F"
```

47

Example

The If Selection Structure

```
if ( grade >= 90 )        // 90 and above
   cout << "A";
else if ( grade >= 80 )   // 80-89
   cout << "B";
else if ( grade >= 70 )   // 70-79
   cout << "C";
else if ( grade >= 60 )   // 60-69
   cout << "D";
else                      // less than 60
   cout << "F";
```

48

The If Selection Structure

- You can assign an int type variable a non zero value for *true* or zero for *false*.
  - Example

```
even = (n%2 == 0);
if(even) { do something }
```

- Some people prefer following for better readability.

```
if(even == 0) { do something }
```

49

The If Selection Structure

- Beginning programmers sometime prefer to use a sequence of if statements rather than a single nested if statement

```
if (x > 0)
        num_pos = num_pos + 1;
if (x < 0)
        num_neg = num_neg + 1;
if (x == 0)
        num_zero = num_zero +1;
```

- This is less efficient because all three of the conditions are always tested.

50

The If Selection Structure

- Nested 'if' statements

```
if(x > 0)
        num_pos = num_pos + 1;
else if(x < 0)
        num_neg = num_neg + 1;
else
        num_zero = num_zero + 1;
```

- In the nested if statement, only the first condition is tested when x is positive.

51

The If Selection Structure

- Nested if statements can become quite complex. If there are more than three alternatives and indentation is not consistent, it may be difficult for you to determine the logical structure of the if statement.
- You can code the nested if as the multiple-alternative decision.

```
if ( condition_1 )
        statement_1
else if ( condition_2 )
        statement_2
        .
        .
        .
else if ( condition_n )
        statement_n
else
        statement_e
```

52

The If Selection Structure

- Order of Conditions
  - When more than one condition in a multiple-alternative decision is true, only the task following the first true condition executes.
  - Therefore, the order of the conditions can affect the outcome.

53

Switch Statement

54

Switch Statement in C++

- **Syntax**

```
switch (selector)
{ case L1: statements1; break;
  case L2: statements2; break;
   …
  default: statements_n;
}
```

55

Alternatives for switch

Multiple if

```
If ( grade >=90)
        cout<<“A”;
If ( grade >=80)
        cout<<“B”;
If ( grade >=70)
        cout<<“C”;
If ( grade > =60)
        cout<<“D”;
If ( grade<60)
        cout<<“F”;
```

Multiple if-else

```
if ( grade >= 90 )
   cout << "A";
else if ( grade >= 80 )
   cout << "B";
else if ( grade >= 70 )
   cout << "C";
else if ( grade >= 60 )
   cout << "D";
else
   cout << "F";
```

```
switch (grade)
{
case 90 :
cout <<"You Got A \n";   break;

case 80 :
cout <<"You Got B \n";   break;

case 70 :
cout <<"You Got C \n";   break;

case 60 :
cout <<"You Got D \n";   break;

default :
cout <<" Sorry , You Got F \n";
}
```

56

Example

```
char character;
cout << "Enter a character : ";
cin >> character;
switch (character)
{
case 'a':
            cout << " Australia " << endl;
            break;
case 'b':
cout << " Bangladesh " << endl;
break;
case 'c':
cout << " Chille " << endl;
break;
case 'd':
cout << " Denmark " << endl;
break;
case 'e':
cout << " England " << endl;
break;
case 'f':
cout << " France " << endl;
break;
case 'g':
cout << " Gana " << endl;
break;
default:
cout << " I dont know more countries " << endl;
}
```

```
char character;
cout << "Enter a character : ";
cin >> character;
switch (character)
{
case 'a':
case 'A':
cout << " Australia " << endl;
cout << " Afaghanistan " << endl;
break;
case 'b':
case 'B':
cout << " Bangladesh " << endl;
break;
case 'c':
case 'C':
cout << " Chille " << endl;
break;
case 'd':
case 'D':
cout << " Denmark " << endl;
break;
case 'e':
case 'E':
cout << " England " << endl;
break;
case 'f':
case 'F':
cout << " France " << endl;
break;
case 'g':
case 'G':
cout << " Gana " << endl;
break;
default:
cout << " I dont know more countries " << endl;
}
```

57

Activity

- Write code using switch which will find area of circle if a is pressed (entered by user) and circumference if c is pressed. Take the value of radius in a variable at the start of program.

58

Example

```
switch (character)
{
case 'a':
area = 3.14f * radius * radius;
cout << " Area = " << area << endl;
break;

case 'c':
circum = 2 * radius * 3.14f;
cout << " Circumference = " << circum << endl;
break;

default:   cout << " Invalid letter was read " << endl;
}
```

59

```
cout << "Enter the grade of student : "; cin >> character;
switch (character)
{
case  'A':
case   'a':
cout << "Excellent";
break;
case  'B':
case   'b':
cout << "Good";
break;
case  'C':
case   'c':
cout << "O.K";
break;
case  'D':
case  'd':
case  'F':
case  'f':
cout << "poor";
break;
default:    cout << "invalid letter grade";
}
```

60

Alternatives for switch

Multiple if

```
If ( grade >=90)
        cout<<"A";
If ( grade >=80)
        cout<<"B";
If ( grade >=70)
        cout<<"C";
If ( grade > =60)
        cout<<"D";
If ( grade<60)
        cout<<"F";
```

Multiple if-else

```
if ( grade >= 90 )
   cout << "A";
else if ( grade >= 80 )
   cout << "B";
else if ( grade >= 70 )
   cout << "C";
else if ( grade >= 60 )
   cout << "D";
else
   cout << "F";
```

```
switch (grade)
{
case 90 :
cout <<"You Got A \n";   break;

case 80 :
cout <<"You Got B \n";   break;

case 70 :
cout <<"You Got C \n";   break;

case 60 :
cout <<"You Got D \n";   break;

default :
cout <<" Sorry , You Got F \n";
}
```

61

What if there are no breaks?

- Without break, switch statements will execute the first statement for which the expression matches the case value AND then evaluate all other statements from that point on
- For example:

```
switch (expression) {
        case value1:
                statement1;

        case value2:
                statement2;

        default:
                default_statement;
}
```

- NOTE: **Every statement after the true case is executed**

62

Points to Remember – switch statement

- The expression followed by each **`case`** label must be a constant expression.
- No two **`case`** labels may have the same value.
- Two **`case`** labels may be associated with the same statements.
- The **`default`** label is not required.
- There can be only one **`default`** label, and it is usually last.

63

Logic Building Hour

64

64

65

Activity

- **Create calculator using switch.**

66

```
// Program to built a simple calculator using switch Statement

#include <iostream>
using namespace std;

int main()
{
      char o;
      float num1, num2;

      cout << "Enter an operator (+, -, *, /): ";
      cin >> o;

      cout << "Enter two operands: ";
      cin >> num1 >> num2;

      switch (o)
      {
            case '+':
                  cout << num1 << " + " << num2 << " = " << num1+num2;
                  break;
            case '-':
                  cout << num1 << " - " << num2 << " = " << num1-num2;
                  break;
            case '*':
                  cout << num1 << " * " << num2 << " = " << num1*num2;
                  break;
            case '/':
                  cout << num1 << " / " << num2 << " = " << num1/num2;
                  break;
            default:
                  // operator is doesn't match any case constant (+, -, *, /)
                  cout << "Error! operator is not correct";
                  break;
      }

      return 0;
```

Output

```
Enter an operator (+, -, *, /): +
-
Enter two operands: 2.3
4.5
2.3 - 4.5 = -2.2
```

67

Activity

- Write program to Reverse a Number. Your program should be able to print any number in reverse.
- E.g. int x = 1234 should be printed as 4321.
- Note: Take number from user.

68

Reverse a Number

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

int main()
{
        int num = 1234;

        cout << num % 10 << " ";    //(1234 % 10 = 4)
        num = num / 10;             //(1234 / 10 = 123)

        cout << num % 10 << " ";    //(123 % 10 = 3)
        num = num / 10;             //(123 / 10 = 12)

        cout << num % 10 << " ";    //(12 % 10 = 2)
        num = num / 10;             //(12 / 10 = 1)

        cout << num % 10 << " ";    //(1 % 10 = 1)
        num = num / 10;             //(1 / 10 = 0)
}
```

69

69