C++ Basics

Rahul Deodhar
@rahuldeodhar
www.rahuldeodhar.com
rahuldeodhar@gmail.com

Homework?

Topics for today

- Basic Input Output

Topics for today

- Variables
  - Data types
  - Identifiers
  - Declarations
    - Global and Local Declarations
  - Assignment
  - Uninitialized variables
- Class work!

Topics for today

- Operators
  - Input/Output Operators
  - Mathematical Operators
    - Multiple Operators
  - Increment and Decrement Operators
  - Boolean Operators
  - Other important operators / identifiers
- Classwork!

Topics for today

- Homework
  - Program
  - Others

Basic Input Output

Input Output

- Std::cin>>
- Std::cout>> “statement output”;
- Std::cout>>x; //variable output
- Namespace in brief
  - Other part after functions, classes etc.
  - Using Namespace std;
  - Cin>>x;
  - Cout>>y;

Special characters

\n	Adds newline cursor goes to start of next line
\t	Moves cursor by one tab
\r	Carriage return, cursor goes to start of same line
\a	Alert sound, using system speaker
\\	Enters backslash character
\”	Enters quotation mark

The datatype of the variable determines the operations indicated by the operator. This is called "Overloading" or "Operator overloading" specifically

Variables

Variables in C++

- C++ is a strongly-typed language, and requires every variable to be declared with its type before its first use.
- Variables have three parts
  - Data type
  - Identifier
  - Value

Default Datatypes

Data Type	Range	Examples	Comments
Bool	True / False	True /False	
char	one ASCII character	a, $, \n, \a	
Int			
Short	-32767 to 32767	1, 15, 10,500	2 bytes
Long (also Int)	-2147483647 to 214483647	500,000	4 bytes
Float	10^(-38) to 10^(38)	7 digits	4 bytes
Double	10^(-308) to 10^(308)	15 digits	8 bytes
Long Double	10^(-4932) to 10^(4932)	15 digits	10 bytes
String			

C++11 Hint

- You can use the type of an initializer as the type of a variable
  - auto x = 1; // 1 is an int, so x is an int
  - auto y = ‘c’; // ’c’ is a char, so y is a char
  - auto d = 1.2; // 1.2 is a double, so d is a double
  - auto s = ”Howdy”; // ”Howdy” is a string literal of type const char[]
    // so don’t do that until you know what it means!
  - auto sq = sqrt(2); // sq is the right type for the result of sqrt(2)
    // and you don’t have to remember what that is

Identifiers in C++

- Starts with a letter, contains letters, digits, and underscores (only)
  - x, number_of_elements, Fourier_transform, z2
  - Not names:
    - 12x
    - time$to$market
    - main line
  - Do not start names with underscores: _foo
    - those are reserved for implementation and systems entities
- Users can't define names that are taken as keyword
- Variable identifiers are case sensitive
  - Radius <> radius

Keywords in C++

- Some Keywords
  - alignas, alignof, and, and_eq, asm, auto, bitand, bitor, bool, break, case, catch, char, char16_t, char32_t, class, compl, const, constexpr, const_cast, continue, decltype, default, delete, do, double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, noexcept, not, not_eq, nullptr, operator, or, or_eq, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_assert, static_cast, struct, switch, template, this, thread_local, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, while, xor, xor_eq
- And others…
- Keywords can appear inside comments (inline / block)

Choose meaningful identifiers

- Abbreviations and acronyms can confuse people
  - mtbf, TLA, myw, nbv
- Short names can be meaningful
  - (only) when used conventionally:
    - x is a local variable
    - i is a loop index / counter
- Don't use overly long names
  - Ok:
    - partial_sum
      element_count
      staple_partition
  - Too long:
    - the_number_of_elements
      remaining_free_slots_in_the_symbol_table

Declarations

- Int x;
- Int x,y;
- Int x=5;
- Int x(25), y(32), z;
- Int x{25}, y{32};

Local and Global Declarations

- Global Declarations
  - Declared outside main()
  - AND
  - Right after header files
- Local Declarations
  - Within main()
    - Int konstant;
  - Within functions ()
    - Int Func_konstant;

Assignment

- Method 1
  - Int x;
  - x = 5; // Assignment
- Method 2
  - Int x=5;
  - Int x(5);
  - Int x {5};

Assignment - Important

- When the variables are declared, they have an undetermined value until they are assigned a value for the first time.

Type Compatibilities

- In general store values in variables of the same type
  - This is a type mismatch:
    ```
    int int_variable;
    int_variable = 2.99;
    ```
  - If your compiler allows this, int_variable will most likely contain the value 2, not 2.99

int ←→ double (part 1)

- Variables of type double should not be assigned
  to variables of type int

```
int int_variable;
double double_variable;
double_variable = 2.00;
 int_variable = double_variable;
```

  - If allowed, int_variable contains 2, not 2.00

int ←→ double (part 2)

- Integer values can normally be stored in variables of type double

```
double double_variable;
double_variable = 2;
```

  – double_variable will contain 2.0

char ← → int

- The following actions are possible but generally not recommended!
- It is possible to store char values in integer variables

```
int value = 'A';
```

  value will contain an integer representing 'A'

- It is possible to store int values in char variables

```
char letter = 65;
```

bool ← → int

- The following actions are possible but generally
  not recommended!
- Values of type bool can be assigned to int variables
  - True is stored as 1
  - False is stored as 0
- Values of type int can be assigned to bool variables
  - Any non-zero integer is stored as true
  - Zero is stored as false

Class Work - Variables

- What would be data type of following variables:
    - Pi / Area / length / breadth
    - Height / Weight / Roll no.
    - Name / Class room / Marks / Rank / Grade
    - Salary / Pan Card No. / Income tax due
    - Flight No. / Flight status / No. of Passengers / Seats available
    - Car Number / Driving License No. / Passport No.
- Advanced data types will be covered subsequently.

Operators

Types of operators

- Assignment Operator (=)
- Arithmetic operators (+, -, *, /, %,^)
- Compound assignment (+=, -=, *=, /=, %=)
- Increment and Decrement Operators (++, --)
- Relational and comparison operators (==, !=, >, <, >=, <=)
- Logical operators (!, &&, ||)
- Conditional ternary operator (?)
  - c = (a>b) ? a : b;
- Comma operator (,)
  - a = (b=3, b+2);
- Bitwise operators (&, |, ^, ~, <<, >>)
- Explicit typecasting operator
  - i = (int) f;

Precedence

- BODMAS

Arithmetic

- Arithmetic is performed with operators
  - + for addition
  - - for subtraction
  - * for multiplication
  - / for division

- Example: storing a product in the variable
  total_weight

```
 total_weight  =  one_weight *
number_of_bars;
```

Results of Operators

- Arithmetic operators can be used with any numeric type
- An operand is a number or variable used by the operator
- Result of an operator depends on the types of operands
  - If both operands are int, the result is int
  - If one or both operands are double, the result is double

Division of Doubles

- Division with at least one operator of type double produces the expected results.

```
double divisor, dividend, quotient;
divisor = 3;
dividend = 5;
quotient = dividend / divisor;
```

  - quotient = 1.6666…
  - Result is the same if either dividend or divisor is of type int

Division of Integers

- Be careful with the division operator!
  - int / int produces an integer result (true for variables or numeric constants)

    ```
    int dividend, divisor, quotient;
    dividend = 5;
    divisor = 3;
    quotient = dividend / divisor;
    ```

  - The value of quotient is 1, not 1.666…
  - Integer division does not round the result, the fractional part is discarded!

Integer Remainders

- % operator gives the remainder from integer division
- int dividend, divisor, remainder;
  dividend = 5;
  divisor = 3;
  remainder = dividend % divisor;

  The value of remainder is 2

Arithmetic Expressions

- Use spacing to make expressions readable
  - Which is easier to read?

```
x+y*z      or   x + y * z
```

- Precedence rules for operators are the same as used in your algebra classes
- Use parentheses to alter the order of operations

```
 x + y * z     ( y is multiplied by z first)
(x + y) * z   ( x and y are added first)
```

Operator Shorthand

- Some expressions occur so often that C++ contains to shorthand operators for them
- All arithmetic operators can be used this way
  - += count = count + 2; becomes
    count += 2;
  - *= bonus = bonus * 2; becomes
    bonus *= 2;
  - /= time = time / rush_factor; becomes
    time /= rush_factor;
  - %= remainder = remainder % (cnt1+ cnt2); becomes
    remainder %= (cnt1 + cnt2);

Assignment and increment

- Examples of Assignment
  - int a = 7; // a variable of type int called a
  - // initialized to the integer value 7
  - a = 9; // assignment: now change a's value to 9
  - a = a+a; // assignment: now double a's value
  - a += 2; // increment a's value by 2
  - ++a; // increment a's value (by 1)
  - - - a; //reduces value of a by 1
- Difference between ++a and a++;
  - Int a

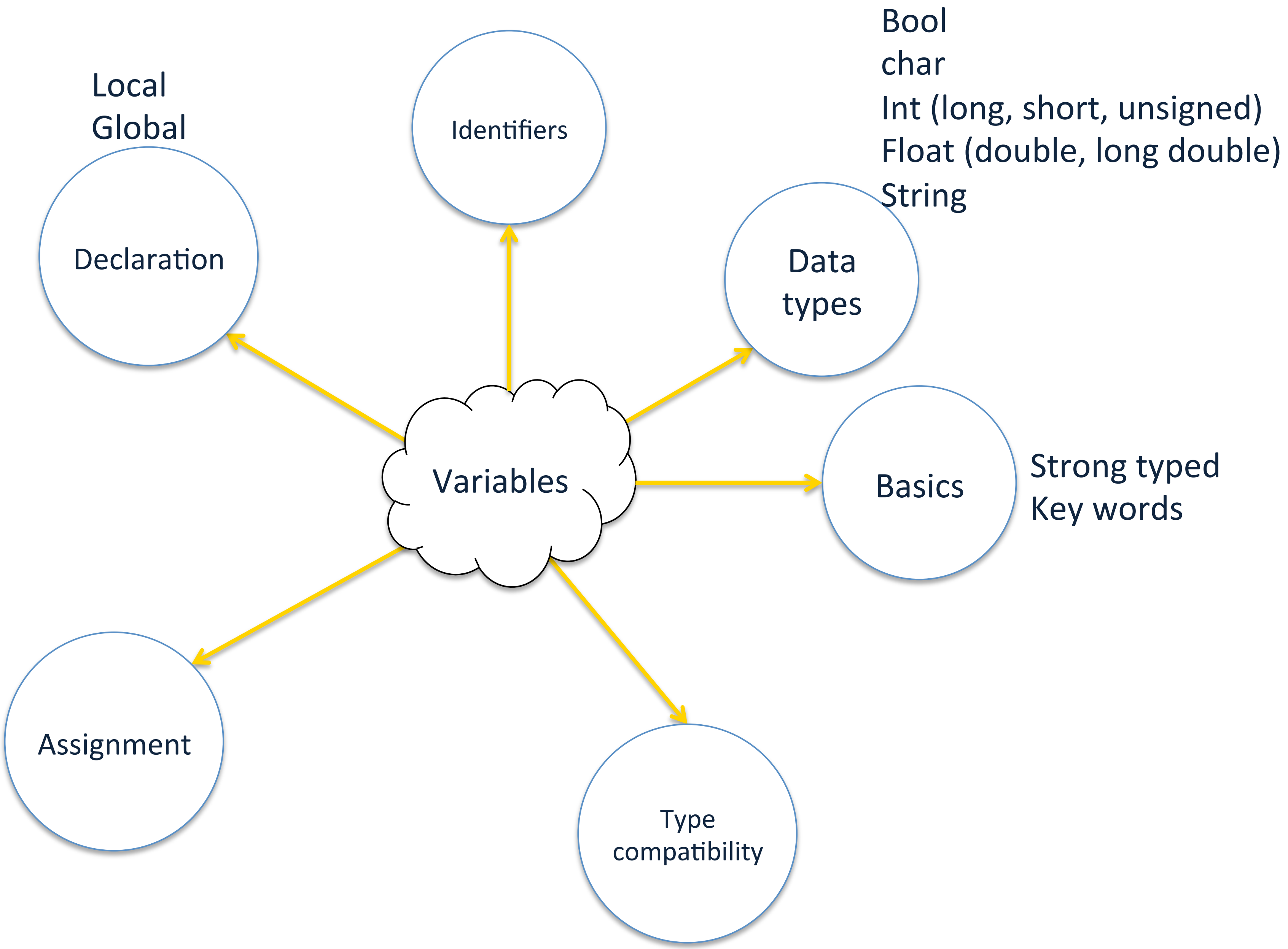
Operators & their function

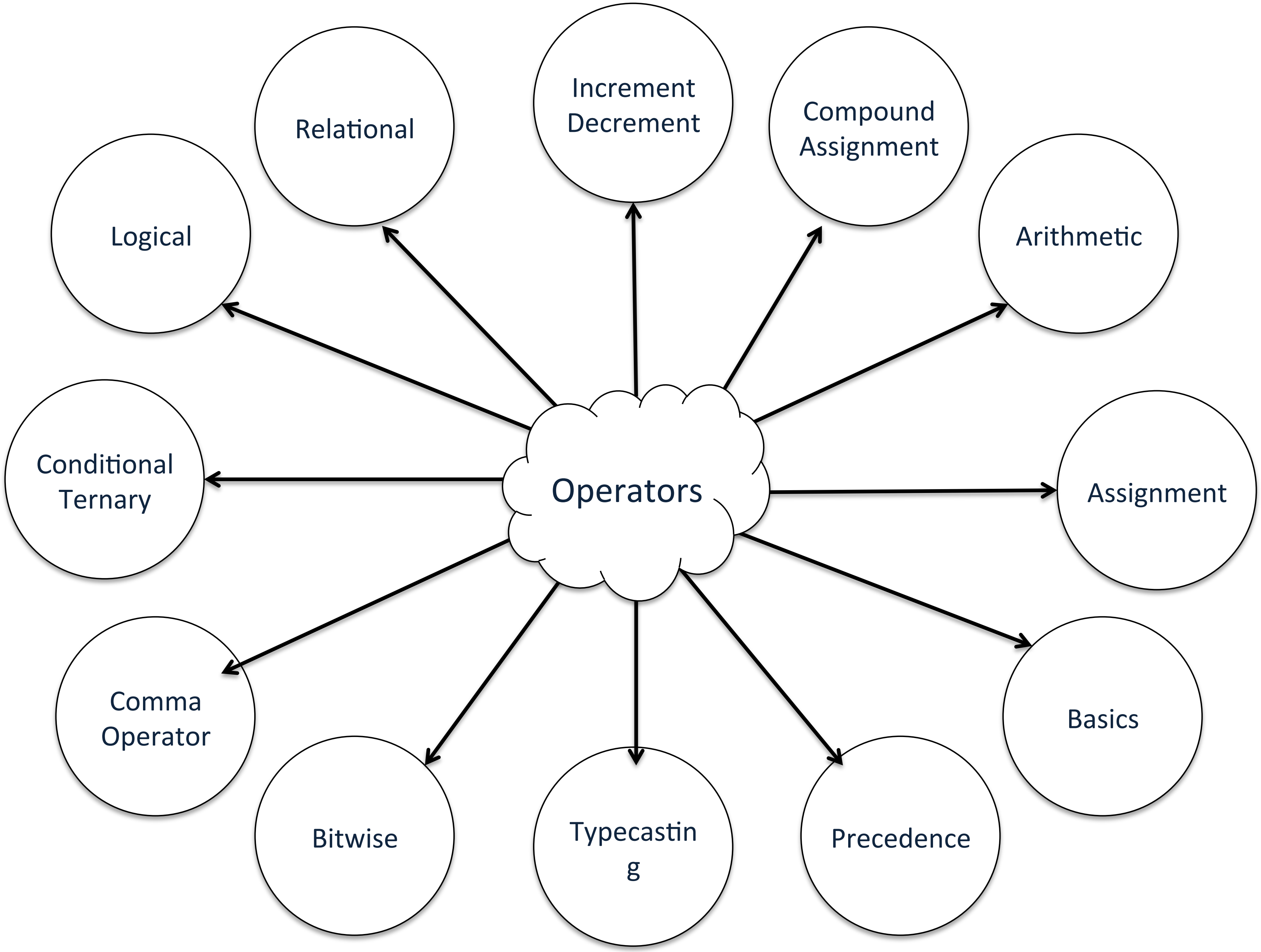
	String	Numbers
cin>>	Reads the word	Reads the number
cout<<	Writes the word	Writes the number
+	Concatenates	Adds
+=s OR +=n	Adds String "s" at the end	Increments number by "n"
++	Error	Increments by 1
-	Error	subtracts

Classwork

- ++/--
- String operators
- Solve equation

Revision


Local
Global
Declaration
Identifiers
Bool
char
Int (long, short, unsigned)
Float (double, long double)
String
Data types
Variables
Basics
Strong typed
Key words
Assignment
Type compatibility


Operators
Increment Decrement
Compound Assignment
Arithmetic
Assignment
Basics
Precedence
Typecastin g
Bitwise
Comma Operator
Conditional Ternary
Logical
Relational

Homework

- Hello World!
- Hello Mr. X! Have a great day!
- Circle
  - Area of circle
    - Input = radius
    - Input as a diameter
    - Input as choice = radius / diameter
  - Circumference of the circle
  - Global variable declarations
  - Others