



# Chapter 5

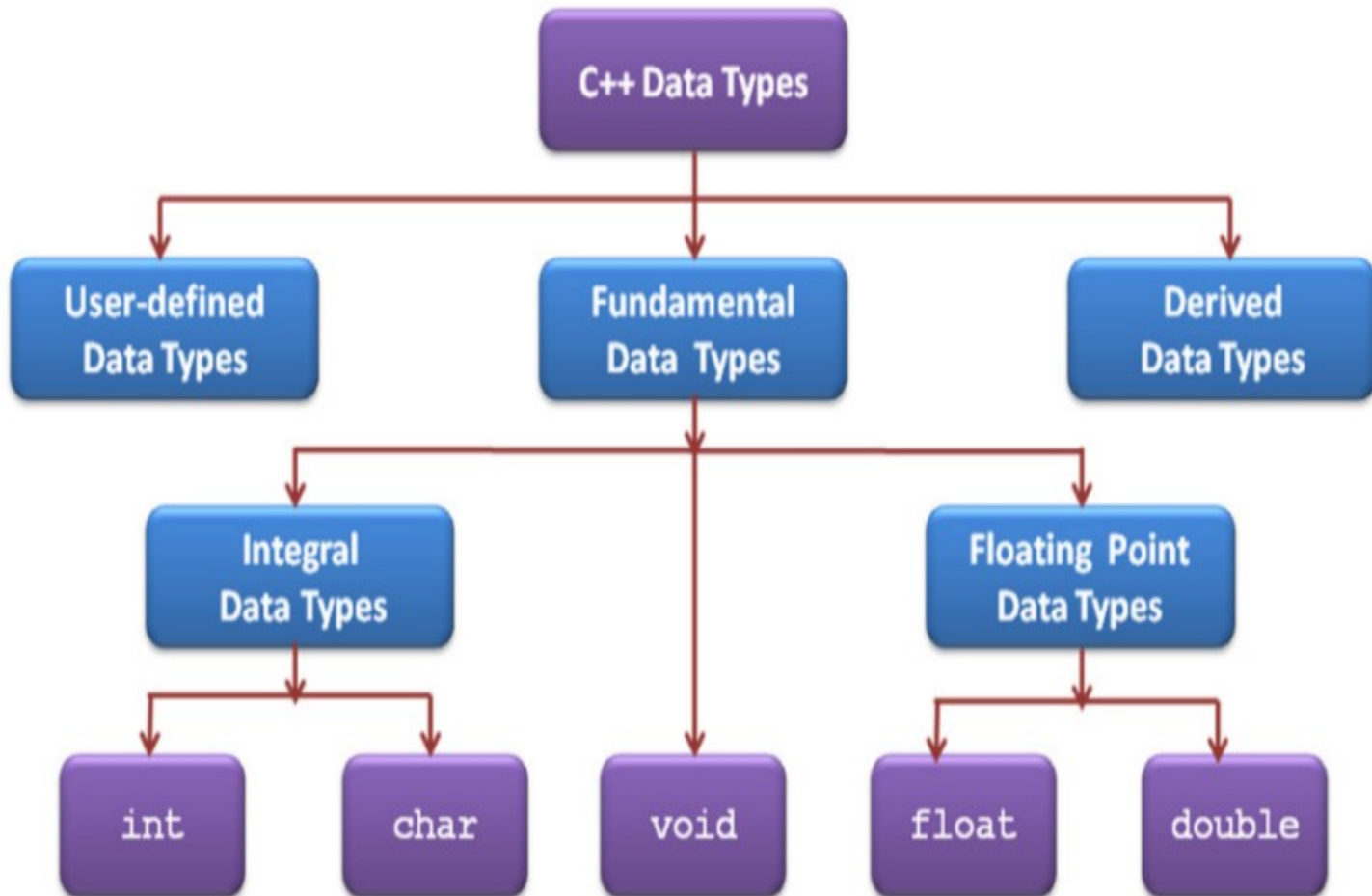
## DATA TYPES AND OPERATORS



# Data Types

- Data types are the means to identify the type of data and associated operations of handling it.
- 


# Classification of data types



Data type/ Keyword	Usage	Approx. Size in memory	Minimal Range
<b>int</b>	Integer	4 bytes	
<b>char</b>	To represent characters	1 Byte	-128 to + 127 0 to 255 -128 to + 127
<b>float</b>	To represent numbers with fractional part. It may be +ve or -ve	4 Bytes	$3.4 \times 10^{-38}$ to $3.4 \times 10^{38} - 1$ (7 digits precision)
<b>double</b>		8 Bytes	$1.7 \times 10^{-308}$ to $1.7 \times 10^{308} - 1$ (15 digits precision)
<b>void</b>		0 Bytes	



# Variables

- Variables are named storage locations whose values can be changed during program run.
- 

# Variables

- The syntax is:  
datatype Variablename;
- Examples:
- Int num;
- float p,q,root;
- int sum=10;
- const float pi = 3.14;
- In the third example the variable 'sum' is assigned with a value 10, which is called the variable initialization. The fourth statement declares pi as a constant with value 3.14, which cannot be altered during the program run.

# Constant :

- It makes a variable symbolic constant. The keyword `const` is used to make an object constant. The syntax is:
- `const Datatype Variablename = Value;`

# Operators

- Consider the expression or statement or operation
- $\text{Sum} = A + B * C$
- Here sum, A, B, C are operands, =, +, \* are operators
- The symbols that used in an operations are called operators.
- An operation requires operator and operands
- Operands are the data on which the specified task is carried out.



# Operators

- They are classified in to three according to its operand,
- Unary (with one operand)
- Binary (with two operands)
- Ternary (with three operands).
- Operators are again classified into I/O operators, Arithmetic operators, Increment/ Decrement operators, Relational operators, Logical operators, Conditional operators, Sizeof operators and Comma operators.

# Arithmetic Operators

- The symbols (or tokens) that trigger computations in the program are called arithmetic operators. They are

Operator	Example a=5, b=2
+ Addition	$5+2=7$
- Subtraction	$5-2=3$
* Multiplication	$5*2=10$
/ Division	$5/2=2.5$
% Remainder of the division	$5\%2=1$

# Relational Operators

- These operators compare two quantities, that is, they determine the relation between two data.
- The result of the operation will be either TRUE (1) or FALSE (0).
- There are six such operators and all of them are binary operators. They are: < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to), == (equal to) and != (not equal to).

# Relational Operators

Operator	Example a=5, b=3
$a > b$	True
$a < b$	False
$a \geq b$	True
$a \leq b$	False
$a \neq b$	True
$a == b$	False

## Logical Operators

- Logical operators are used to combine existing expressions. The operands as well as the result of these operators are Boolean values (TRUE and / or FALSE).
- The logical AND (&&), Logical OR (||) and logical NOT (!) are the operators.

## Logical Operators

Consider two relational operations with following output then,

Logical AND (&&)	Logical OR (  )	Logical NOT (!)
$T \&\& T = T$	$T    T = T$	
$T \&\& F = F$	$T    F = T$	$!T = F$
$F \&\& T = F$	$F    T = T$	$!F = T$
$F \&\& F = F$	$F    F = F$	

# Input/Output Operators

- The operators for input (cin) and output (cout) operations are >> (called extraction or get from operator) and << (called insertion or put to operator) respectively.
- cin >>a ;
- cin>>num1>>num2;
- cout<<"hello guys";
- cout << "Average of" <<a<< "and" <<b << "is" <<avg;
- The multiple use of input or output operators (>>or<<) in one statement is called cascading of I/O operators. But >> and << together should not appear in one statement.

# Assignment Operator (=)

- The values returned by the expressions can be stored in a variable using assignment operator (=)

Variablename=Expression;

Example:

A=5;

b=a;

**Compare a=5 and a==5**

-



# Expressions

- Operators and operands are combined to form expressions.
- An expression represents an operation and normally returns a value as the result of the operation.
- It may be an integer expression or a real expression or a logical expression.
- Eg.  $c = a + b$ ;

# Arithmetic Expression

- 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

# Arithmetic Expression (cont.)

- 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`

# Arithmetic Expression (cont.)

- 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!

# Arithmetic Expression (cont.)

- % 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 Expression (cont.)

## Integer Division

$$\begin{array}{r} 4 \\ 3 \overline{) 12} \\ \underline{12} \\ 0 \end{array}$$

←  $12/3$

←  $12\%3$

$$\begin{array}{r} 4 \\ 3 \overline{) 14} \\ \underline{12} \\ 2 \end{array}$$

←  $14/3$

←  $14\%3$

# Arithmetic Expression (cont.)

## Arithmetic Expressions

### Mathematical Formula

### C++ Expression

$$b^2 - 4ac$$

$$b*b - 4*a*c$$

$$x(y + z)$$

$$x*(y + z)$$

$$\frac{1}{x^2 + x + 3}$$

$$1/(x*x + x + 3)$$

$$\frac{a + b}{c - d}$$

$$(a + b)/(c - d)$$

# Arithmetic Expression (cont.)

- 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

$x + v * z$       (  $v$  is multiplied by  $z$  first)



# Arithmetic Expression (cont.)

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

# Relational / Boolean Expressions

- Relational Expressions
  - Use relational operators and operands of various types
  - Evaluate to some Boolean representation
  - It gives result either 1 or 0 (means true or false)
  - Operator symbols used
  - ( $\neq$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ,  $==$  etc)
  - Eg for Relational Expressions
  - $C = x > y$  ,  $x + y \neq 10$  ,  $X * Y == A + B$

# Logical Expression

- Used to combine two or more Relational Expressions
  - It gives result either 1 or 0 (means true or false)
  - Operator symbols used
  - AND (&&), OR (||) , Not (!)
- Eg
  - $C = x < 10 \ \&\& \ x > y$  ,
  - $x + y \neq 10 \ || \ X * Y == A + B$
  - $!(x == y) \ \&\& \ !(X + Y == 0)$

# Statements

- Statements are smallest executable units of a programming language
- 1. Declaration statement: used to declare a data type for a variable
- Eg: `int a;`
- `Float avg;`
- `Double score;`

# Assignment Statements

- used to assign a value to a variable
- Eg: `a=15;`
- `avg=12.5;`
- `Score=avg*avg;`
- `D=(a+b)*(a-b)`

# Input/Output Statement in

## C++ The Insertion Operator

(<<)

- To send output to the screen we use the insertion operator on the object cout
- Format: `cout << Expression;`
- The compiler figures out the type of the object and prints it out appropriately

```
cout << 5; // Outputs 5
```

```
cout << 4.1; // Outputs 4.1
```

```
cout << "String"; // Outputs String
```

```
cout << '\n'; // Outputs a newline
```

# Input/Output Statement in

## C++ The Extraction Operator

(>>)

- To get input from the keyboard we use the extraction operator and the object `cin`
- Format: `cin >> Variable;`
- No need for `&` in front of variable
- The compiler figures out the type of the variable and reads in the appropriate type

```
int X;
```

```
float Y;
```

```
cin >> X; // Reads in an integer
```

# Cascading I/O Operators

- We can combine different input/output operators in to one
- Eg: `cin>>a`
- `cin>>b;`
- `cin>c;`
- Can be written as `cin>>a>>b>>c;`
- as well as `cout<<a<<b<<c;`
- `cout<<"Result is"<<a+b+c;`