Programming fundamentals

Lecture 5: Data types, operators, conditional statements

Recap

- Formatting output: setfill, setw, setprecision

Agenda

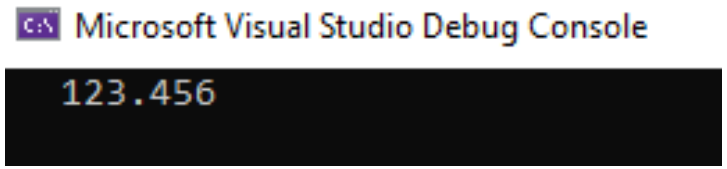
- Data types,
- operators, and
- conditional statements

Setf()

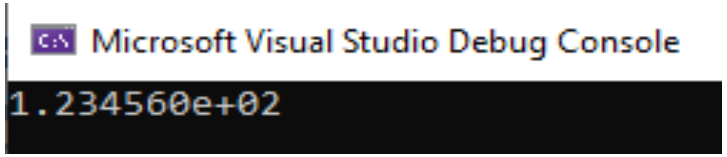
- cout.setf(ios**::argument);**
- Test the output by changing the **argument** with:

left	left-justify the output
right	right-justify the output
scientific	display floating point numbers in scientific ("E") notation
fixed	display floating point numbers in normal notation - no trailing zeroes and no scientific notation

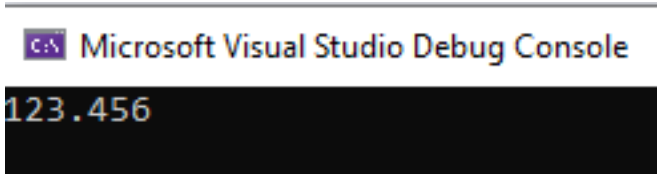
```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    cout.setf(ios::right);
    cout <<setw(10)<< 123.456 << endl;
    return 0;
}
```



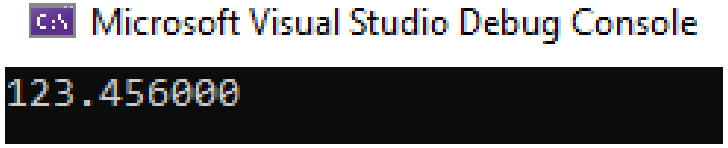

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    cout.setf(ios::scientific);
    cout <<setw(10)<< 123.456 << endl;
    return 0;
}
```




```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    cout.setf(ios::left);
    cout <<setw(10)<< 123.456 << endl;
    return 0;
}
```




```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    cout.setf(ios::fixed);
    cout <<setw(10)<< 123.456 << endl;
    return 0;
}
```




Punjab University College of Information Technology
PUCIT
Serve Humanity
UNIVERSITY OF THE PUNJAB

Data types

Integer data types

- Integer data types hold whole numbers. C++ has eight different integer data types.
- They differ based on how large of an integer they can store **(how many bytes are allocated for storage)** and whether they can store both positive and negative integers, or only nonnegative integers.

Data Type	**Size***	**Range**
int	4 bytes	-2,147,483,648 to +2,147,483,647
unsigned int	4 bytes	0 to +4,294,967,295
short	2 bytes	-32,768 to +32,767
unsigned short	2 bytes	0 to +65,535
long long	8 bytes	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
unsigned long long	8 bytes	0 to +8,446,744,073,709,551,615

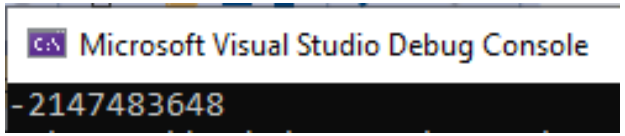
Integer: int

- It takes 4 bytes
- 4 bytes= 4x8=32 bits => 2^32= 4,294,967,296 i.e. **(0** to **4,294,967,295)**
- Integers consists of both negative and positive numbers. Thus we need to divide that number by 2.
- 4,294,967,296/2 =2,147,483,648
- Thus we have negative numbers in range from -1 to -2,147,483,648
- And positive numbers have range from 0 to 2,147,483,647 (not 2,147,483,648 because we start positive numbers from 0)

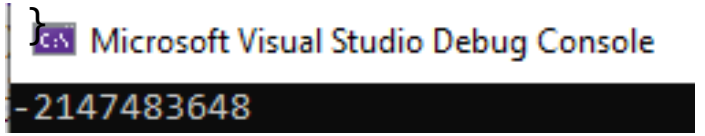
				MIN						MAX			
Input value	--	--	-2,147,483,649	-2,147,483,648	-2,147,483,647	--	0	--	2,147,483,646	2,147,483,647	2,147,483,648	--	--
Output on console	Max -2	Max -1	Max 2,147,483,647	-2,147,483,648	-2,147,483,647	-- -	0	-- -	2,147,483,646	2,147,483,647	Min -2,147,483,648	Min+1	Min+2

			MIN						MAX			
--	--	-2,147,483,649	-2,147,483,648	-2,147,483,647	--	0	--	2,147,483,646	2,147,483,647	2,147,483,648	--	--
Max-2	Max-1	Max 2,147,483,647	-2,147,483,648	-2,147,483,647	-- -	0	-- -	2,147,483,646	2,147,483,647	Min -2,147,483,648	Min+1	Min+2


```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    int x;
    x = -2147483647-1;
    cout << x;
    return 0;
```




```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
int x;
x = 2147483648;
cout << x;
return 0;
}
```



Checking the amount of memory allocated

- Exact size depends on architecture of computer (e.g. 32 and 64 bit systems)
- You can check the exact size using **sizeof** operator


```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int myIntVariable = 1000;
    // use sizeof with types
    cout << "bytes for short: " << sizeof(short) << endl;
    cout << "bytes for int: " << sizeof(int) << endl;
    // use sizeof with variable
    cout << "bytes for myIntVariable: " << sizeof(myIntVariable) << endl;
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
bytes for short: 2
bytes for int: 4
bytes for myIntVariable: 4
```

Float and double

Floating point data types are used to store real numbers (numbers that can have a fractional part). There are three C++ floating point data types, float, double, and long double.

Data Type	Size*
float	4 bytes
double	8 bytes
long double	8 bytes

```
#include <iostream>
using namespace std;
int main()
{
    float num = 4.5;
    cout << num;
    return 0;
}
```

char

- The char data type is used for single characters.
- Interpreted as integers, ASCII codes
- A char literal is a single character enclosed in single quotes.

```
#include <iostream>
using namespace std;
int main()
{
    char alphabet = 'a';
    cout << alphabet;
    return 0;
}
```

string

- **#include <string>** must be used

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

int main()
{
    string myName = “Ali";  // string literals are enclosed in double quotes

    cout << "My name is: " << myName << endl;
    return 0;
}
```

bool

- The data type bool is for Boolean variables. Boolean data are either true or false.
- 1 byte

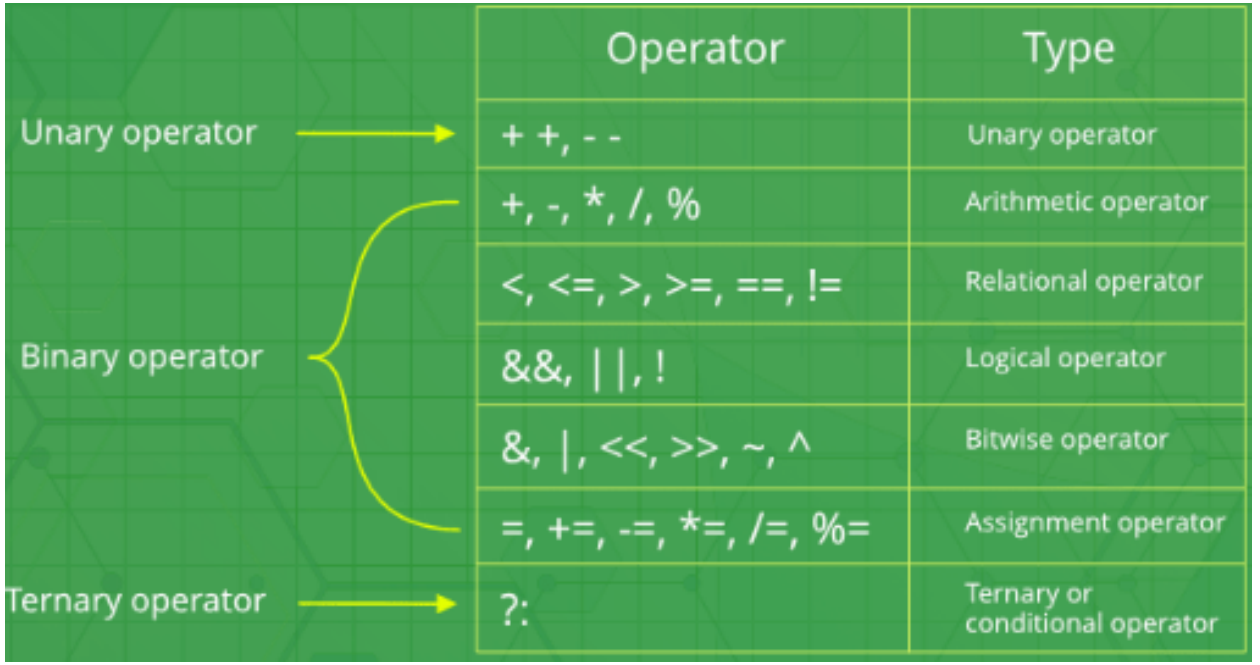
```
#include <iostream>
#include <string>
using namespace std;

int main()
{
bool myBooleanVariable = false;  // initialize as false

myBooleanVariable = true; // switch value to true

cout << "Value of my variable is: " << myBooleanVariable << endl;
return 0;
}
```


Punjab University College of Information Technology
PUCIT
Serve Humanity
UNIVERSITY OF THE PUNJAB

Operators

Operators

- An *operator* is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- *Arithmetic, assignment, relational* and *logical,* and *bitwise*



Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, ||, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

Arithmetic

++ and – (increment and decrement operators

x = x+1;
can be written as

++x; // prefix form

or as

x++; // postfix form

Operator's precedence

highest	++ – –
	– (unary minus)
	* / %
lowest	+ –

Operator(s)	Operation(s)
( )	Parentheses
*	Multiplication
/	Division
%	Modulus
+	Addition
–	Subtraction

e.g. solving expression

Algebra: $z = pr\%q + w/x - y$

C++:

```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

e.g. same operator used multiple times

Step 1. `y = 2 * 5 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`2 * 5 is 10`

Step 2. `y = 10 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`10 * 5 is 50`

Step 3. `y = 50 + 3 * 5 + 7;` *(Multiplication before addition)*

`3 * 5 is 15`

Step 4. `y = 50 + 15 + 7;` *(Leftmost addition)*

`50 + 15 is 65`

Step 5. `y = 65 + 7;` *(Last addition)*

`65 + 7 is 72`

Step 6. `y = 72` *(Last operation—place* `72` *in* `y`*)*

Relational and Logical Operators

Relational Operators	
Operator	**Meaning**
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
==	equal to
!=	not equal to

Logical Operators	
Operator	**Meaning**
&&	AND
||	OR
!	NOT

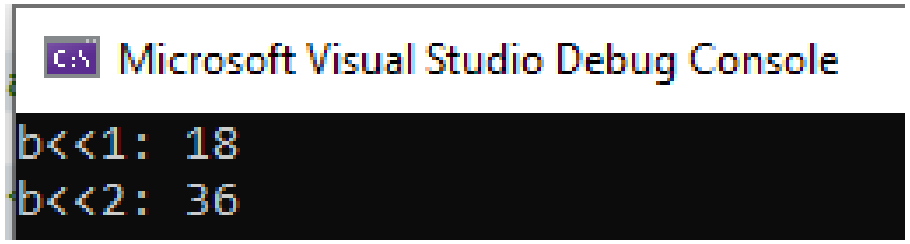
highest	!
	> >= < <=
	= = !=
	&&
lowest	||

Bitwise operators

- The **& (bitwise AND)** in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
- The **| (bitwise OR)** in C or C++ takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
- The **^ (bitwise XOR)** in C or C++ takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
- The **<< (left shift)** in C or C++ takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
- The **>> (right shift)** in C or C++ takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
- The **~ (bitwise NOT)** in C or C++ takes one number and inverts all bits of it

e.g

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int b = 9;
    cout << "b<<1: "<<(b<<1)<<endl;
    b = 9;
    cout <<"b<<2: "<< (b << 2)<<endl;
    return 0;
}
```



Conditional statements

- If
- If, else
- If, else if,…., else