

COMP2012H Honors Object-Oriented Programming and Data Structures Syntax Comparison between Java and C++: Basics and Program Flow Control

The purpose of this set of notes is to help you quickly transfer your basic knowledge of Java to that of C++. Please note that it is not a complete summary of our lecture notes. For all the C++ features discussed in COMP2012H, you have to carefully study the lecture notes on our course website.

In Java	In C++
Hello World Program	
<pre>/* * File: HelloWorld.java * A common program used to demo a new language */ public class HelloWorld { public static void main(String[] args) { System.out.println("Hello world"); } }</pre>	<pre>/* * File: hello_world.cpp * A common program used to demo a new language */ #include <iostream> using namespace std; int main() { cout << "Hello world" << endl; return 0; }</pre>
Executing a Java program	Executing a C++ program
<ol style="list-style-type: none">1. Compile the program: javac HelloWorld.java2. Execute the program: java HelloWorld	<ol style="list-style-type: none">1. Compile the program: g++ -o hello_world.out hello_world.cpp2. Execute the program: hello_world.out
Basic Output	
To print the word “abc” with a newline character: System.out.println("abc"); Or, System.out.print("abc\n");	To print the word “abc” with a newline character: cout << "abc" << endl; where endl means “end of the line” Or , cout << "abc\n";
Comments	
<ul style="list-style-type: none">• For one or more lines of comments: /* ... */• For one line of comment only: // ...	The same.
Using Packages/Libraries	
import java.io.*;	#include <iostream>

Statements	
<ul style="list-style-type: none">• Each statement ends in a semicolon “;”• Extra blanks, tabs, lines are ignored.• More than one statement can be on one line.• A statement may be spread over several lines.	The same. For example: cout << "Hello" << " world"; cout << "!" << endl;
For example: System.out.print("Hello" + " world"); System.out.println("!");	
Variables	
<ul style="list-style-type: none">• Primitive Data Types:<ul style="list-style-type: none">– Integer: <code>short</code>, <code>int</code>, <code>long</code> Examples of values: 0, 1, 100, -101, ...– Floating point: <code>float</code>, <code>double</code>, <code>long double</code>, etc. Examples of values: 0.5, -123.908232– Character: <code>char</code> Examples of values: 'A', 'a', 'B', 'b', ...– Boolean: <code>boolean</code> Examples of values: <code>true</code>, <code>false</code>• Variables have to be declared and defined. For examples: int num1; num1 = 100; double num2 = 0.05;	<ul style="list-style-type: none">• Basic Data Types:<ul style="list-style-type: none">– Integer: <code>short</code>, <code>int</code>, <code>long</code>, <code>long long</code>, etc. Examples of values: 0, 1, 100, -101, ...– Floating point: <code>float</code>, <code>double</code>, <code>long double</code>, etc. Examples of values: 0.5, -123.908232– Character: <code>char</code> Examples of values: 'A', 'a', 'B', 'b', ...– Boolean: <code>bool</code> Examples of values: <code>true</code>, <code>false</code>• Variables have to be declared and defined. For examples: int num1; num1 = 100; double num2 = 0.05;
if Statement	
The syntax of the if statements are the same in Java and C++: if (<bool-expr>) <stmt> if (<bool-expr>) { <stmt(s)> } if (<bool-expr>) <stmt> else <stmt> if (<bool-expr>) { <stmt(s)> } else { <stmt(s)> } if (<bool-expr>) { <stmt(s)> } else if (<bool-expr>) { <stmt(s)> } else { <stmt(s)> }	
For example, in Java: int x = -5; if (x > 0) { System.out.print("x is positive"); if (x % 2 == 1) System.out.println(" and odd."); else System.out.println(" and even."); } else if ((x < 0) && (x % 2 == 1)) System.out.println("x is negative and odd."); else if ((x < 0) && !(x % 2 == 1)) System.out.println("x is negative and even."); else System.out.println("x is zero.");	For example, in C++: int x = -5; if (x > 0) { cout << "x is positive"; if (x % 2) cout << " and odd." << endl; else cout << " and even." << endl; } else if ((x < 0) && (x % 2)) cout << "x is negative and odd." << endl; else if ((x < 0) && !(x % 2)) cout << "x is negative and even." << endl; else cout << "x is zero." << endl;

if-else Operator

The syntax of the if-else expressions are the same in Java and C++:

```
<condition> ? <result1> : <result2>
```

It means that if <condition> is true, the expression's value will be <result1>, otherwise it will be <result2>.

For example:

```
int x = 2, y = 3;
cout << ((x > y) ? x : y) << endl; // the output will be 3
```

while Loop

The syntax of the while statements are the same in Java and C++:

```
while (<bool-expr>) { <stmt(s)> }
do { <stmt(s)> } while (<bool-expr>);
```

for Loop

The syntax of the following for statements are the same in Java and C++:

```
for (<for-initialization>; <bool-expr>; <post-processing>) { <stmt(s)> }
```

break and continue

The syntax are the same in Java and C++:

In a **for** loop, **break** means to stop the whole loop; while **continue** means to skip the current execution.

Methods and Functions

The class methods in Java are equivalent to the class member functions in C++.

But C++ also has global functions which are similar to static class methods in Java.

For example,

```
/* File: Example.java
A Java program of the class Example
with two methods:
PrintNum() and AddOne()
*/
public class Example {
    public static void main(String args[])
    {
        PrintNum(10);
        PrintNum(AddOne(10));
    }

    public static void PrintNum(int num)
    {
        System.out.println("The number is " + num);
    }

    public static int AddOne(int num)
    {
        return (num + 1);
    }
}
```

For example,

```
/* File: function_example.cpp
A C++ program with two functions:
PrintNum() and AddOne()
*/
#include <iostream>
using namespace std;

void PrintNum(int num)
{
    cout << "The number is " << num << endl;
}

int AddOne(int num)
{
    return (num + 1);
}

int main()
{
    PrintNum(10);
    PrintNum(AddOne(10));
    return 0;
}
```

Operators

The following operators are the same in Java and C++:

		Symbol	Example	Output
Arithmetic Operators	Addition	+	1 + 2	3
	Subtraction	-	1 - 2	-1
	Multiplication	*	1 * 2	2
	Division	/	1.0 / 2	0.5
	Modulus (Remainder)	%	1 / 2	0
Assignment Operators	Modulus (Remainder)	%	9 % 4	1
	Assignment	=	x = y	
	Addition Assignment	+=	x += y	
	Subtraction Assignment	-=	x -= y	
	Multiplication Assignment	*=	x *= y	
Relational Operators	Division Assignment	/=	x /= y	
	And	&&	true && false	false
	Or		true false	true
	Not	!	!false	true
Comparison Operators	Larger than	>	20 > 10	true
	Larger than or equal to	>=	20 >= 10	true
	Smaller than	<	20 < 10	false
	Smaller than or equal to	<=	20 <= 10	false
	Equal to	==	20 == 10	false
	Not equal to	!=	20 != 10	true
Increment Operators	Post-increment	++	x = 1; y = 2; y = x++; cout << x << " " << y;	2 1
	Pre-increment	++	x = 1; y = 2; y = ++x; cout << x << " " << y;	2 2
Decrement Operators	Post-decrement	--	x = 1; y = 2; y = x--; cout << x << " " << y;	0 1
	Pre-decrement	--	x = 1; y = 2; y = --x; cout << x << " " << y;	0 0

References:

1. Cay Horstmann. (2012). C++ For Everyone. Second Edition. Wiley.
2. The Java Tutorial. Aug 2016. <https://docs.oracle.com/javase/tutorial/index.html>