1. Language Basics

A simple C++ program: oop_ex1.cpp
A simple Java program: oop_ex2.java
Note: the “main” in C++ and Java

1.1 Variables

A variable is an item of data names by an identifier.
A general variable declaration format:

```
type name;      //Declaration only
type name = value;      //Declaration and initialization
type name1, name2 = value2, name3; //Multiple declaration in a single line
```

where (1)type: see Primitive Data Types in C++ and Java
(2)name: see Identifier

Ex: Point out all the variable declarations oop_ex1.cpp and oop_ex2.java.

1.1.1 Primitive Data Types in C++ and Java

Type	C++ Keyword	Java Keyword
Boolean Type	bool	boolean
Character Type	char	char
	unsigned char	-
Integer Type	-	byte
	short	short
	int	int
	long	long
	unsigned short	-
	unsigned int	-
	unsigned long	-
Real Number Type	float	float
	double	double
	long double	-

```
Ex:     bool b = true;          //C++
        boolean bb = false;     //Java
        int x = 3, y = 777, z = -100;   //Multiple declaration in a single line
        double pi = 3.14159;
        char c = 'c';
```

Note: In addition to primitive data types, C++ has pointer types and reference types, and Java has reference types. These types are mostly for arrays and objects.

1.1.2 Identifier

A program refers to a variable's value by the variable's name.

In C++ and Java, all names are case-sensitive.

For a legal variable's name, the following must be true:

1. It must be a legal identifier. See references for detail.
2. It must not be a keyword.
3. It must be unique within its scope.

1.1.3 Scope

Local variable: A variable is declared inside a function (includes main) or a class.
Global variable: A variable is not declared within any { }.

The hiding effect: oop_ex3.cpp

1.1.4 const / final variable

const is the keyword in C++ for declaring variables whose value can't be changed.
final is the equivalent keyword in Java.

In C++, the declaration of a const/final variable must assign its value. In Java, *blank final* is allowed.

Ex: oop_ex4.cpp oop_ex5.java

1.2 Operators

An operator performs a function on one, two, or three operands.

Unary operators:

```
operator op          //prefix notation     //ex: -x
op operator          //postfix notation    //ex: y--
```

Binary operator:

```
op1 operator op2     //infix notation      //ex: a+b
```

Ternary operator

```
op1 ? op2 : op3      //infix notation
```

1.2.1 Arithmetic Operators

Binary Arithmetic Operators

Operator	Use	Description
+	op1 + op2	Also used to concatenate strings in Java
-	op1 - op2	
*	op1 * op2	
/	op1 / op2	
%	op1 % op2	Compute the remainder

Unary Arithmetic Operator

Operator	Use	Description
+	+op	
-	-op	

Shortcut Increment and Decrement Operator

Operator	Use	Description
++	op++	op = op+1; evaluate before increment
	++op	op = op+1; evaluate after increment
--	op--	op = op-1; evaluate before decrement
	--op	op = op-1; evaluate after decrement

1.2.2 Relational and Conditional Operators

The result of relational or conditional operation is either true(1) or false(0).

Relational Operators

Operator	Use	Description
>	op1 > op2	true if op1 *greater than* op2
>=	op1 >= op2	*greater than or equal to*
<	op1 < op2	*less than*
<=	op1 <= op2	*less than or equal to*
==	op1 == op2	*equal to*
!=	op1 != op2	*not equal to*

Conditional Operators

Operator	Use	Description
&&	op1 && op2	and
||	op1 || op2	or
!	!op	true->false, false->true

1.2.3 Assignment Operators

Assignment Operator

Operator	Use	Description
=	op1 = op2	Copy the value of op2 to op1

Shortcut Assignment Operator

Operator	Use	Description
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2

1.3 Control Flow Statements

Control Flow Statements

Statement type	Keywords
Looping	while, do-while, for
Decision Making	if-else, switch-case
Branching	break, continue, return
Excepting handling	try-catch-finally, throw (for Java)

Java Excepting handling:

```
Try{
    statement(s)
}catch(exceptiontype name){
    statement(s)
}finally{
    statement(s)
}
```

1.4 Simple I/O in C++ and Java

1.4.1 input from keyboard and output to screen in C++

To read in data from user's input on keyboard:

```
cin >> var_name;
cin >> var_name1 >> var_name2 >> var_name3 ……;
```

To display variable's value or string on screen (DOS window):

```
cout << var_name;
cout << var_name1 << "Output words or symbols" << var_name2 << endl;
```

Ex: oop_ex6.cpp

1.4.2 input from keyboard and output to screen in Java

To read in data from user's input on keyboard: (Must be done in a hard way, data conversion is often needed)

(1) `int var_name = System.in.read( );` `// read a char, then return its int code`

(2)
```
DataInputStream in = new DataInputStream(System.in);
String buffer = in.readLine();        //read a line of data as a string,
                                      //can convert it into int, double etc.
```

(3)
```
Scanner in = new Scanner(System.in);
int var_name = in.nextInt();          //use Scanner to read numerical data
                                      //import java.util.* is required
```

To display variable's value or string on screen (DOS window):

```
System.out.print(var_name);
System.out.println( var_name1 + "Output words or symbols" + var_name2);
```

Ex: oop_ex7.java oop_ex8.java

1.4.3 input/output from/to files in C++

To read in data from an input file:

```
ifstream fcin = ifstream("inputFileName.txt");
fcin >> var_name;
fcin >> var_name1 >> var_name2 >> var_name3 ……;
```

To write variable's value or string to an output file:

```
ofstream fcout = ofstream("outputFileName.txt");
fcout << var_name;
fcout << var_name1 << "Output words or symbols" << var_name2 << endl;
```

Ex: oop_ex9.cpp

1.4.4 input/output from/to files in Java

To read in data from an input file:

(1) FileReader in = new FileReader(new File("inputFileName.txt"));
var_name = in.read() // read a char, then return its int code

(2) FileReader inputFile = new FileReader("inputFileName.txt");
BufferedReader in = new BufferedReader(inputFile);
String buffer = in.readLine(); //read a line of data as a string,
//can convert it into int, double etc.

(3) FileReader inputFile = new FileReader("inputFileName.txt");
Scanner in = new Scanner(inputFile);
double var_name = in.nextDouble(); //use Scanner to read numerical data
//import java.util.* is required

To write variable's value or string to an output file:

(1) FileWriter out = new FileWriter(new File("outputFileName.txt "));
out.write(var_name); // write a char by its int value

(2) OutputStream outputFile = new FileOutputStream("ans.txt");
PrintWriter out = new PrintWriter(outputFile);
out.println(var_name);
out.close(); // this is required!

Ex: oop_ex10.java, oop_ex11.java, oop_ex12a.java, oop_ex12b.java

1.5 Functions(in C++) / Methods(in Java)

Please study the following topics about functions/methods.

1. (C++) Header files and the use of functions in C++ standard libraries:
   #include <xxxx.h>
2. (Java) The use of classes/interfaces in Java packages:
   import java.xxx.*;
3. (C++ & Java) Function/Method definition.
4. (C++) Function prototype.
5. (C++ & Java) Recursive function.
6. (C++) inline function.
7. (C++) default argument.

The following topics on functions/methods will be introduced later.

1. (C++/Java) Function/Method Overloading.
2. Argument passing in C++ & Java.