Programming in C/C++

Patrick Ho
peiqistar@gmail.com

Day 2

Comparison Operators

Equal to	a == b
Not equal to	a != b
Greater than	a > b
Less than	a < b
Greater than or equal to	a >= b
Less than or equal to	a <= b

Return value is bool (true / false)

e.g.

```
bool ret = (7 != 7);
```

int ia=9; int ib =10;

```
bool ret = (ia > ib);
```

(ret=?)

int ca=10; int cb=10;

```
bool ret = (ca <= cb);
```

(ret=?)

Logical Operators

Logical negation (NOT)	`!a`
Logical AND	`a && b`
Logical OR	`a || b`

Return value is bool (true / false)
e.g.
ia=true;
bool ret = !ia; (ret=false)
ia=-1; ib=-3;
bool ret = (ia < 0) && (ib < 0); (ret=true)
bool ret = (ia < 0) || (ib < 0); (ret=true)

Note: AND is TRUE on both TRUE
OR is TRUE on either one TRUE

Bitwise Operators

Bitwise NOT	`~a`
Bitwise AND	`a & b`
Bitwise OR	`a | b`
Bitwise XOR	`a ^ b`
Bitwise left shift[note 1]	`a << b`
Bitwise right shift[note 1]	`a >> b`

XOR: diff is TRUE

Return value is same type value

Note: Remember << and >> by N
<< N is multiple 2 of power N
>> N is divide 2 of power N
e.g. int i = 2 << 2; // i=2*4= 8

Special Arithmetic Operators

Increment	Prefix	++a
	Suffix	a++
Decrement	Prefix	--a
	Suffix	a--

1. Value is plus + / minus – by 1 to itself (a = a+1;or a=a-1;)
2. Operation order is based on Pre or Suf in whole statement

e.g. A

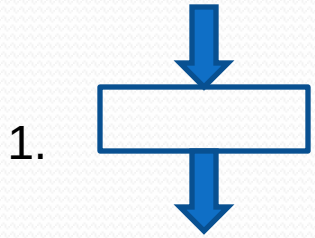
```
int a = 1;
int b = 1 + a++;
int b=1+a;
a++;
b = ?
 a = ?
```

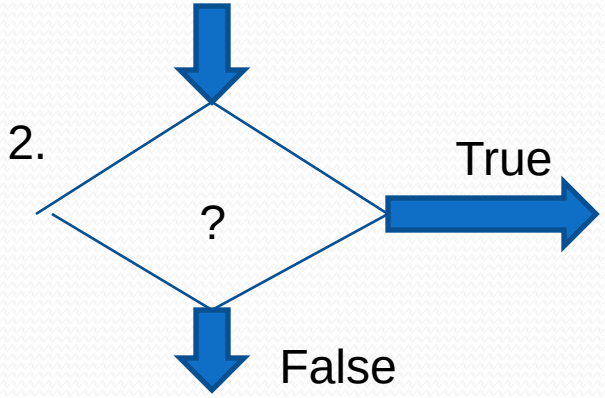
e.g. B

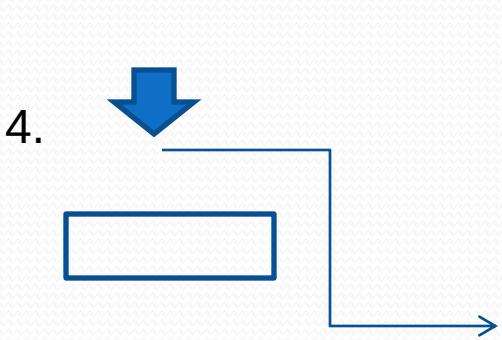
```
int a = 1;
int b = 1 + (++a);
a++;
int b=1+a;
// b = ?
// a = ?
```

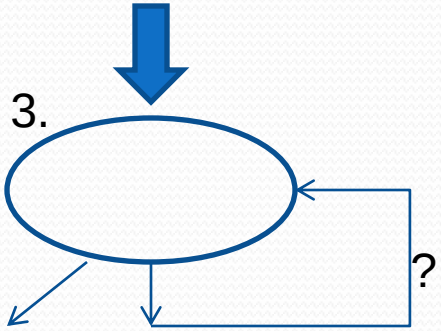
Programming Work Flow

1. Sequential Statements Flow
   Step by step moving forward
2. Choice Control Flow
3. Loops Control Flow
4. Jump Control Flow









Choice Control Flow (1)

perform different computations or actions depending on whether a Boolean condition is true or false.

1.if … (else) … statement

a. if (condition) {…} if (i==0) {printf("i=0\n");}
b. if (…) {…} else {…}
c. if (…) {…} else if (…) {…} else {…} // multi condition
d. () ? {} : {}; simple format for "if () {} else {} "

e.g.

```
bool ret = (i < 0) ? true : false;
```

Conditional statements can be and often are nested inside other conditional statements.

Problem Solving

Given any integer, convert it to positive integer.

Input: 100

Output: 100

Input: -100

Output: 100

1. Solving:

```
/*
Convert to positive integer
*/

#include <cstdio>

int main()
{
        int inputI;

        scanf("%d", &inputI);
        if (inputI < 0)
        {
           inputI *= -1; // inputI = -inputI;
        }

        printf ("%d\n", inputI);

        return 0;
}
```

Loops Control Flow (1)

1. for statement
   a. for (initial statement; condition statement; finish statement) {…}

   e.g. for (int i=0; i<10; i++) {printf("%d\n", i);}

   a. for (; condition statement; finish statement) { … }
   b. for (;;) { … } <dead loop>

   Sometimes there's continue/break statement inside it

Loops Control Flow (2)

2. while statement
   a. while (condition statement) {…}

   e.g. int cnt=0;

   ```
   while (cnt < 10){printf("%d\n", cnt);
           //if (cnt==4){    cnt += 2;    break;}
           cnt++;}
   ```

   a. while (true) {…} <dead loop>

3. Do-While statement
   a. do { … } while (condition statement)

   Sometimes there's continue/break statement inside it

Jumps Control Flow

1. continue statement
   a. Make flow go to finish statement(for)/condition statement(while) in current loop

2. break statement
   a. Make flow go out/finish current loop statement

3. goto statement
   a. Make flow jump to any where with label
   b. goto AnyPoint;
      AnyPoint: ...
   c. Not good flow control (not used except must)

Problem Solving

Multiplication Table

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Problem Solving

Given any integer,
calculate the sum of each multiple of 7 smaller than it.

Input: 10

Output: 7

Input: 15

Output: 21