The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Week 2 - Lecture 2
Data Types, Variables and Operators

Edited by: Dr. Wooi Ping Cheah
Autumn 2022

University of Nottingham
UK | CHINA | MALAYSIA

Overview

- **Variables and Data types**
- printf()
- Selection statements
- Arithmetic operators
- Operator precedence

University of Nottingham
UK | CHINA | MALAYSIA

Naming Variables

- Begin with either a letter or the underscore

- Can contain uppercase letters, lowercase letters, digits and the underscore

- Case sensitive, e.g. Var != var != VaR

- Keywords cannot be used

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	for	return	typedef	
default	float	short	union	

University of Nottingham
UK | CHINA | MALAYSIA

Sometimes, Errors are confusing!

University of Nottingham
UK | CHINA | MALAYSIA

Naming Variables (2)

- Declare each variable on a separate line e.g.
int x; /* comment */
int y; /* comment */

- Unless the variables are self-explanatory and related e.g.
int year, month, day;

- Group related variables. Place unrelated variables even of the same type in separate lines
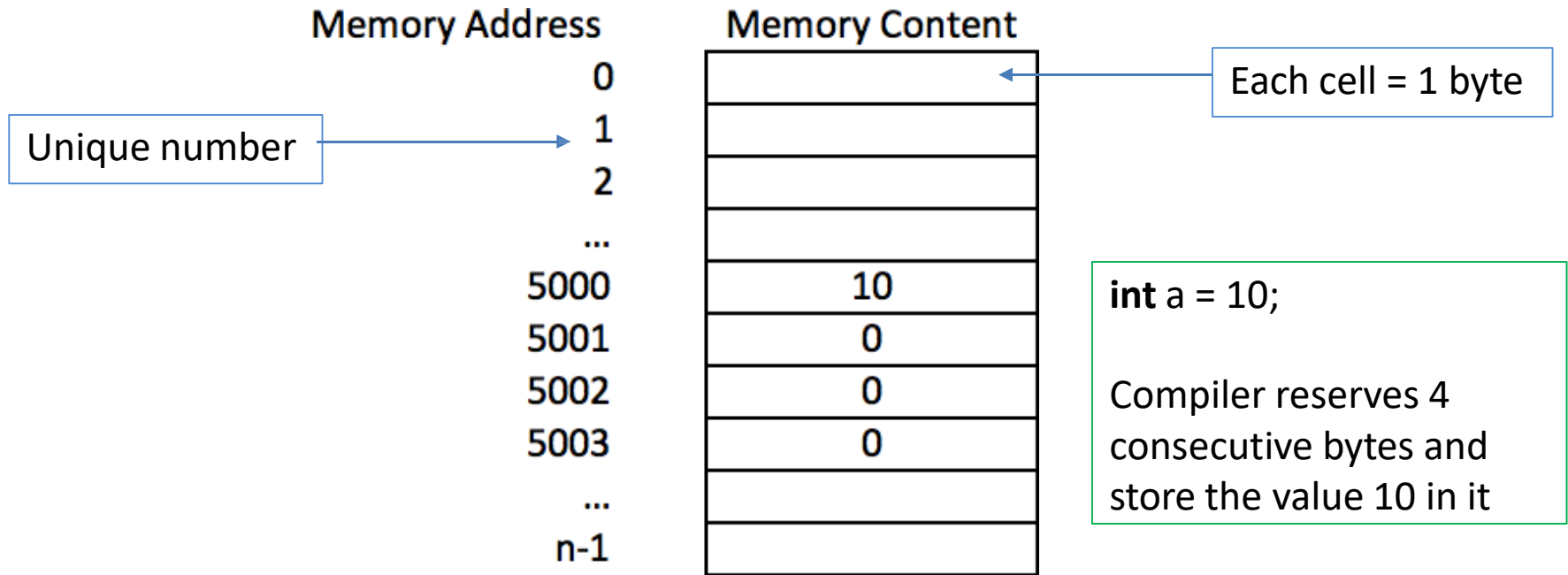int x, y, z;
int year, month, day;

Declaring Variables

- Declaration:

    data_type name_of_variable;

```
int  a;
float b;
int c, d, e;
```

Type	Size (Bytes)	Range (Min – Max)
char	1	-128 … 127
short	2	-32.768 … 32.767
int	4	-2.147.483.648…2.147.483.647
long	4	-2.147.483.648…2.147.483.647
float	4	Lowest positive value: $1.17*10^{-38}$
		Highest positive value: $3.4*10^{38}$
double	8	Lowest positive value: $2.2*10^{-308}$
		Highest positive value: $1.8*10^{308}$
long double	8, 10, 12, 16	
unsigned char	1	0 … 255
unsigned short	2	0 … 65535
unsigned int	4	0 … 4.294.967.295
unsigned long	4	0 … 4.294.967.295

University of Nottingham
UK | CHINA | MALAYSIA

Memory Layout

Memory Address	Memory Content
0	
1	
2	
...	
5000	10
5001	0
5002	0
5003	0
...	
n-1	

Unique number

Each cell = 1 byte

int a = 10;

Compiler reserves 4 consecutive bytes and store the value 10 in it

Var Name, Val and Mem Address

- **int** ID = 2017233;

    Variable name   Value

- &ID

    Memory address of ID

```
C:\Users\z2017233\Desktop>iteration
Current ID number is 0
Current ID number is 0060FF2C

Enter your ID number: 2017233

Current ID number is 2017233
Current ID number is 0060FF2C

Data Types, Variables and Operators
C:\Users\z2017233\Desktop>
```
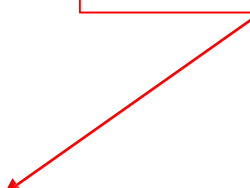
```c
2   #include <stdio.h>
3
4   int main(void)
5   {
6       int id = 0;
7
8       printf("Current ID number is %d\n", id);
9       printf("Current ID number is %p\n", &id);
10
11      printf("\n\nEnter your ID number: ");
12      scanf("%d", &id);
13
14      printf("\n\nCurrent ID number is %d\n", id);
15      printf("Current ID number is %p\n", &id);
16
17      return 0;
18  }
```

University of Nottingham
UK | CHINA | MALAYSIA

Assigning Values to Variables

- int a;
  a = 100;

- int a = 100;

- int a = 100, b = 200, c = 300;

- int a = 100, b = a + 100, c = b + 100;

> Assignments take place from right to left,
> … **but** comma take place left to right

University of **Nottingham**
UK | CHINA | MALAYSIA

Assigning Values to Variables (2)

```c
#include <stdio.h>

int main(void)
{
        int a = 0;
        int b = 0;
        int c = 0;

        a = 100, b = a + 100, c = b + 100;

        printf("a + b + c = %d\n", (a + b + c));

        a = 0;
        b = 0;
        c = 0;

        c = b + 100, b = a + 100, a = 100;

        printf("a + b + c = %d\n", (a + b + c));

        return 0;

}
```

Output:

a + b + c = 600
a + b + c = 300

Data Type Overflow in C

- Each data type has a maximum and a minimum limit.

- E.g. unsigned char is between 0 – 255.

```
65   unsigned char a = 250;
66   int b = (int)a;
67   int counter = 0;
68
69   printf("a is %d and b is %d\n", a, b);
70
71   while(counter < 10)
72   {
73       a = a + 1;
74       counter = counter + 1;
75       printf("%d + %d = %d\n", b, counter, a);
76   }
77   // NOTE: that 'a' doesn't go beyond 255
```

```
C:\prapa\teaching\2020_21\PGA>datatype
a is 250 and b is 250
250 + 1 = 251
250 + 2 = 252
250 + 3 = 253
250 + 4 = 254
250 + 5 = 255
250 + 6 = 0
250 + 7 = 1
250 + 8 = 2
250 + 9 = 3
250 + 10 = 4
```

University of
Nottingham
UK | CHINA | MALAYSIA

limits.h

Determines the properties of the various variable types.

Indeed, limits the values of various variable types like char, int and long.

Macro	Value	Description
CHAR_BIT	8	Defines the number of bits in a byte.
SCHAR_MIN	-128	Defines the minimum value for a signed char.
SCHAR_MAX	+127	Defines the maximum value for a signed char.
UCHAR_MAX	255	Defines the maximum value for an unsigned char.
CHAR_MIN	-128	Defines the minimum value for type char and its value will be equal to SCHAR_MIN if char represents negative values, otherwise zero.
CHAR_MAX	+127	Defines the value for type char and its value will be equal to SCHAR_MAX if char represents negative values, otherwise UCHAR_MAX.
MB_LEN_MAX	16	Defines the maximum number of bytes in a multi-byte character.
SHRT_MIN	-32768	Defines the minimum value for a short int.
SHRT_MAX	+32767	Defines the maximum value for a short int.
USHRT_MAX	65535	Defines the maximum value for an unsigned short int.
INT_MIN	-2147483648	Defines the minimum value for an int.
INT_MAX	+2147483647	Defines the maximum value for an int.
UINT_MAX	4294967295	Defines the maximum value for an unsigned int.
LONG_MIN	-9223372036854775808	Defines the minimum value for a long int.
LONG_MAX	+9223372036854775807	Defines the maximum value for a long int.
ULONG_MAX	18446744073709551615	Defines the maximum value for an unsigned long int.

Source: https://www.tutorialspoint.com/c_standard_library/limits_h.htm

University of Nottingham
UK | CHINA | MALAYSIA

Overview

- Variables and Data types
- **printf()**
- Selection statements
- Arithmetic operators
- Operator precedence

printf(): Escape Sequences

Escape Sequence	Action
\a	Make an audible beep
\b	Delete the last character (equivalent to using the Backspace key)
\n	Advance the cursor to the beginning of the next line (equivalent to using the Enter key)
\r	Move the cursor to the beginning of the current line (equivalent to a carriage return)
\t	Move the cursor to the next tab stop (equivalent to the Tab key)
\\	Display a single backslash (\ )
\"	Display double quotes (")

University of Nottingham
UK | CHINA | MALAYSIA

printf(): Conversion Specifications

Conversion Specifier	Meaning
c	Display the character that corresponds to an integer value.
d, i	Display a signed integer in decimal form.
u	Display an unsigned integer in decimal form.
f	Display a floating-point number in decimal form using a decimal point. The default precision is six digits after the decimal point.
s	Display a sequence of characters.
e, E	Display a floating-point number in scientific notation using an exponent. The exponent is preceded by the specifier.
g, G	Display a floating-point number either in decimal form (%f) or scientific notation (%e).
p	Display the value of a pointer variable.
x, X	Display an unsigned integer in hex form: %x displays lowercase letters (a-f), while %X displays uppercase letters (A-F).
o	Display an unsigned integer in octal.
%	Display the character %.

University of Nottingham
UK | CHINA | MALAYSIA

printf(): Optional Fields - Precision

```c
float a = 1.2365;
printf("Val = %f\n", a);
printf("Val = %.2f\n", a);
```

```
Val = 1.236500
Val = 1.24
```

University of
Nottingham
UK | CHINA | MALAYSIA

Overview

- Variables and Data types

- printf()

- **Selection statements**

- Arithmetic operators

- Operator precedence

University of
Nottingham
UK | CHINA | MALAYSIA

Selection Statements in C

- C provides three types of selection structures in the form of statements.

  - If, if-else, and nested if statements.

  - Conditional operator

  - Switch-case statement

University of **Nottingham**
UK | CHINA | MALAYSIA

If, if-else, and nested if statements

```
if(exp1)
    add1;
else if(exp2)
    add2;
else
    add3;
```

```
a = 2;
if(a > 3)
    printf("a is more than 3\n");
else if(a > 2)
    printf("a is more than 2\n");
else
    printf("a equals to 2\n");
```

University of Nottingham
UK | CHINA | MALAYSIA

If, if-else, and nested if statements (2)

- If the condition is true (i.e., the condition is met) the statement in the body of the if statement is executed.

- If the condition is false (i.e., the condition isn't met) the body statement is not executed.

- Conditions in if statements are formed by using the equality operators and relational operators.

- In C, a condition may actually be any expression that generates a zero (false) or nonzero (true) value.

University of Nottingham
UK | CHINA | MALAYSIA

If, if-else, and nested if statements (3)

Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

University of Nottingham
UK | CHINA | MALAYSIA

Example: Typhoon

- Control statements can increase efficiency of programs.

```c
48   #include <stdio.h>
49   #include <stdlib.h>
50
51   int main(int argc, char *argv[])
52   {
53       int wind = -1;
54
55       printf("Enter wind speed (km/h): ");
56       scanf("%d", &wind);
57
58       if(wind < 0)
59       {
60           printf("Impossible wind speed!\n");
61           exit(1);
62       }
63
64       if(wind < 63)
65       {
66           printf("Tropical Depression\n");
67           exit(1);
68       }
69       if(wind < 119)
70       {
71           printf("Tropical Storm\n");
72           exit(1);
73       }
74       if(wind < 241)
75       {
76           printf("Typhoon\n");
77           exit(1);
78       }
79
80       printf("Super Typhoon\n");
81
82   }
```

University of Nottingham
UK | CHINA | MALAYSIA

Conditional Operator

- C provides the conditional operator (?:) which is closely related to the if...else statement.

$$Exp?\ Value1:value2;$$

$$grade >= \mathbf{60}\ ?\ \mathbf{1}:\mathbf{0};$$

- The conditional operator is C's only ternary operator—it takes three operands.

University of **Nottingham**
UK | CHINA | MALAYSIA

Conditional Operator (2)

- The first operand is a condition.

- The second operand is the value for the entire conditional expression if the condition is true and the operand is the value for the entire conditional expression if the condition is false.
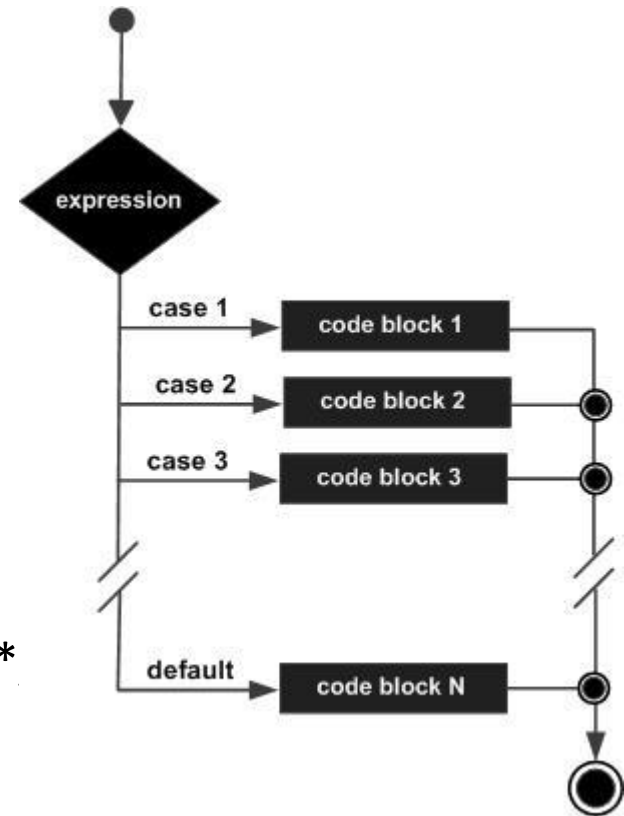
Switch-case statements

```
switch(expression) {

case constant-expression  :
    statement(s);
    break; /* optional */

case constant-expression  :
    statement(s);
    break; /* optional */

/* you can have any number of case statements *
default : /* Optional */
statement(s);
}
```

Switch-case statements (2)

```
switch(grade) {
    case 'A' :
      printf("Excellent!\n" );
      break;
    case 'B' :
    case 'C' :
      printf("Well done\n" );
      break;
    case 'D' :
      printf("You passed\n" );
      break;
    case 'F' :
      printf("Better try again\n" );
      break;
    default :
      printf("Invalid grade\n" );
}
```

Switch-case statements (3)

```
162    #include <stdio.h>
163
164    int main(void)
165    {
166        int A = 10;
167        int B = 20;
168        int C = 0;
169
170        // if-else statement
171        if(A > B)
172        {
173            C = A - B;
174        }
175        else
176        {
177            C = B - A;
178        }
179        printf("The difference between A and B is %d\n", C);
180
181        // conditional operator
182        (A > B) ? (C = A - B): (C = B - A);
183        printf("The difference between A and B is %d\n", C);
184
185        // switch statement
186        C = A - B;
187        switch((int)(C >= 0))
188        {
189            case 0:
190                printf("The difference between A and B is %d\n", (C * -1));
191                break;
192            case 1:
193                printf("The difference between A and B is %d\n", C);
194                break;
195            default:
196                printf("Error!!\n");
197                break;
198        }
199
200        return 0;
201    }
```

University of Nottingham
UK | CHINA | MALAYSIA

Overview

- Variables and Data types

- printf()

- Selection statements

- **Arithmetic operators**

- Operator precedence

University of
Nottingham
UK | CHINA | MALAYSIA

Arithmetic Operators

int A = 10, B = 20;

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Source: http://www.tutorialspoint.com/cprogramming/c_operators.htm

University of Nottingham
UK | CHINA | MALAYSIA

Assignment Operators

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
+=	c += 7	c = c + 7	10 to c
-=	d -= 4	d = d - 4	1 to d
*=	e *= 5	e = e * 5	20 to e
/=	f /= 3	f = f / 3	2 to f
%=	g %= 9	g = g % 9	3 to g

Source: http://www.tutorialspoint.com/cprogramming/c_operators.htm

University of Nottingham
UK | CHINA | MALAYSIA

Increment and Decrement Operators

Operator	Sample expression	Explanation
++	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	a++	Use the current value of a in the expression in which a resides, then increment **a** by 1.
--	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	b--	Use the current value of b in the expression in which **b** resides, then decrement b by 1.

Source: http://www.tutorialspoint.com/cprogramming/c_operators.htm

University of Nottingham
UK | CHINA | MALAYSIA

Increment and Decrement Operators (2)

- Pre-incrementing (pre-decrementing) a variable causes the variable to be incremented (decremented) by 1, then the new value of the variable is used in the expression in which it appears.

- Post-incrementing (post-decrementing) the variable causes the current value of the variable to be used in the expression in which it appears, then the variable value is incremented (decremented) by 1.

Source: http://www.tutorialspoint.com/cprogramming/c_operators.htm

University of **Nottingham**
UK | CHINA | MALAYSIA

Operator Precedence

Operator Category	Operators	Associativity
Primary	`()  []  ->  .  ++(postfix)  --(postfix)`	Left to right
Unary	`!   ~   ++(prefix)   --(prefix)   *(dereference)   & (address)  sizeof`	Right to left
Cast	`()`	Right to left
Multiplicative	`*(multiplication)  /  %`	Left to right
Additive	`+ -`	Left to right
Bitwise Shift	`<<  >>`	Left to right
Relational	`<  <=  >  >=`	Left to right
Equality	`==  !=`	Left to right
Bitwise AND	`&`	Left to right
Bitwise XOR	`^`	Left to right
Bitwise OR	`|`	Left to right
Logical AND	`&&`	Left to right
Logical OR	`||`	Left to right
Conditional	`?:`	Right to left
Assignment	`=  +=  -=  *=  /=  %=  &=  ^=  |=  <<=  >>=`	Right to left
Comma	`,`	Left to right

University of
Nottingham
UK | CHINA | MALAYSIA

Operator Precedence (2)

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)
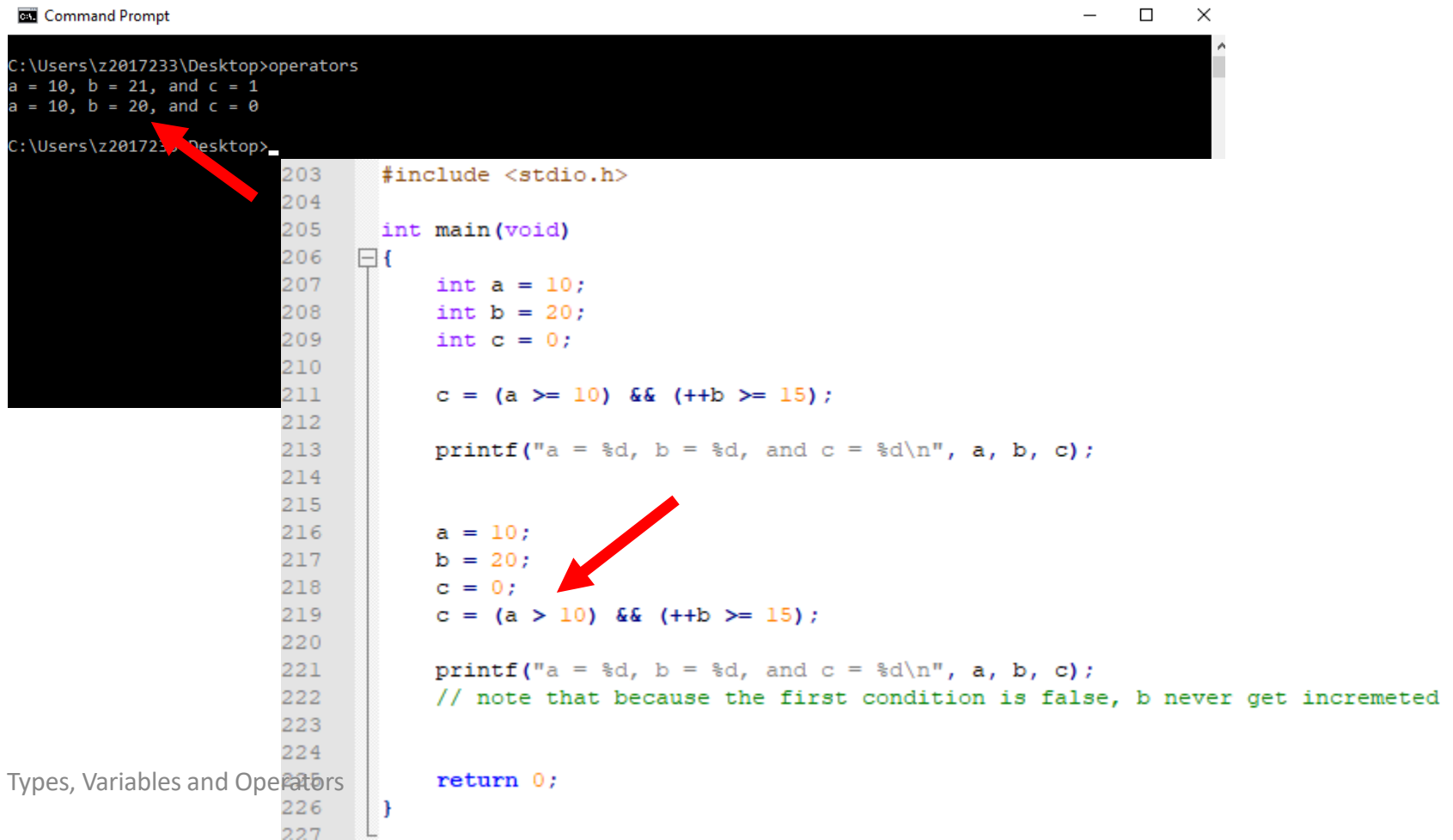
Operator Precedence (3)

- z = p * r % q + w / x − y;

  　6　　1　　2　　　4　　3　　5

- y = a * x * x + b * x + c;

  　6　　1　　2　　4　　3　　5

> The order in which C evaluates the operators.

Be Careful of Side Effect!!!

- Logical operators … &&, ||


```
203    #include <stdio.h>
204
205    int main(void)
206    {
207        int a = 10;
208        int b = 20;
209        int c = 0;
210
211        c = (a >= 10) && (++b >= 15);
212
213        printf("a = %d, b = %d, and c = %d\n", a, b, c);
214
215
216        a = 10;
217        b = 20;
218        c = 0;
219        c = (a > 10) && (++b >= 15);
220
221        printf("a = %d, b = %d, and c = %d\n", a, b, c);
222        // note that because the first condition is false, b never get incremeted
223
224
225        return 0;
226    }
227
```

Command Prompt output:
```
C:\Users\z2017233\Desktop>operators
a = 10, b = 21, and c = 1
a = 10, b = 20, and c = 0

C:\Users\z2017233\Desktop>
```

math.h

- Note rounding errors and the output type of any functions!

```
235    // example 1
236    int lhs = pow(x, 2) + pow(y, 2);
237    int rhs = pow(z, 2);
238
239    if(lhs == rhs)
240    {
241        printf("Right angled triangle\n");
242    }
243    else
244    {
245        printf("Not right angled, %d does not equal %d\n", lhs, rhs );
246    }
247
248    // example 2
249    if(z == sqrt(pow(x, 2) + pow(y, 2)))
250    {
251        printf("Right angled triangle\n");
252    }
253    else
254    {
255        printf("Not right angled, %d does not equal %d\n", lhs, rhs );
256    }
```

University of
Nottingham
UK | CHINA | MALAYSIA

Summary

- Variables and Data types

- printf()

- Selection statements

- Arithmetic operators

- Operator precedence