Technical Informatics I

Arithmetic operations and math functions

by
Dr. Fatimah
Faculty of Mechanical Engineering
fatimahd@ump.edu.my

Arithmetic operations and math functions

- Aims
  - Introduce students to arithmetic operations, the header file `math.h` and math functions
- Expected Outcomes
  - Students are able to construct simple C programs that can implement arithmetic operations with appropriate operators and precedence
  - Students are able to construct simple C programs involving various math functions
- References
  - Harry H. Cheng, 2010. C for Engineers and Scientists: An Interpretive Approach, McGraw Hill

Content

- Arithmetic Operations and Precedence
- Math Functions
- Examples
- Conclusion

Arithmetic Operations

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

- Notes on the `%` operator:
  - The operands of the `%` operator should be of type `int` and will return the remainder.
  - If the value of the second operand is zero, the behavior is undefined.
  - For example: 9 % 3 = 0, 6 % 9 = 6, 10 % 3 = 1

Arithmetic Operations

- The order of data type is: `char` -> `int` -> `float` -> `double`.
  - Here char takes less memory while double takes the most memory
- You may convert a data type that occupies ***less memory*** to a data type that occupies ***more memory*** space without any loss.
  - However, the resultant data type depends on the operations and algorithms
  - For binary operations, such as +,-,/,*, the resultant data type will take the higher order data type of two operands.
    - The addition of two `double` will result in a `double`
    - The addition of an `int` and a `double` will result in a double.

Arithmetic Operations

- Example:

```c
#include<stdio.h>
int main(){
  int i = 10;  /*initialize i=10*/

  printf("5*i=%d\n",5*i); /*multiplication with an int*/
  printf("19/5=%d\n",19/5); /*division between 2 ints results in an int*/
  printf("19.0/5=%f\n",19.0/5); /*division of a double with an int results in a double*/
  printf("19/5.0=%f\n",19/5.0); /*division of an int with a double results in a double*/
  printf("19%%5=%d\n",19%5); /*remainder of 19 divide by 5*/
return 0;
}
```

```
>ch -u "MA15024.c"
5*i=50
19/5=3
19.0/5=3.800000
19/5.0=3.800000
19%5=4
>Exit code: 0
```

Arithmetic Operations

- The list of operators are shown on right.
- Operators at the higher level has precedence over operators at the lower level.

Operations	Associativity
::	Left to right
() []	Right to left
function name()	Left to right
. ->	Right to left
' ! ` ++ -- + - * &(type) sizeof	Left to right
* / % .* ./	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
^^	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= |= <<= >>=	Right to left
,	Right to left

Arithmetic Operations

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Example 1:

```
i = 2 + 3*4
```

Operations	Associativity
::	Left to right
() []	Right to left
function name()	Left to right
. ->	Right to left
' ! ` ++ -- + - * &(type) sizeof	Left to right
* / % .* ./	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
^^	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= |= <<= >>=	Right to left
,	Right to left

Arithmetic Operations

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Example 1:

```
i = 2 + 3*4
  = 2 + 12
```

Operations	Associativity
::	Left to right
() []	Right to left
function name()	Left to right
. ->	Right to left
` ! ` ++ -- + - * &(type) sizeof	Left to right
* / % .* ./	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
^^	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= |= <<= >>=	Right to left
,	Right to left

Arithmetic Operations

Example 1:

Followed by the operator +.

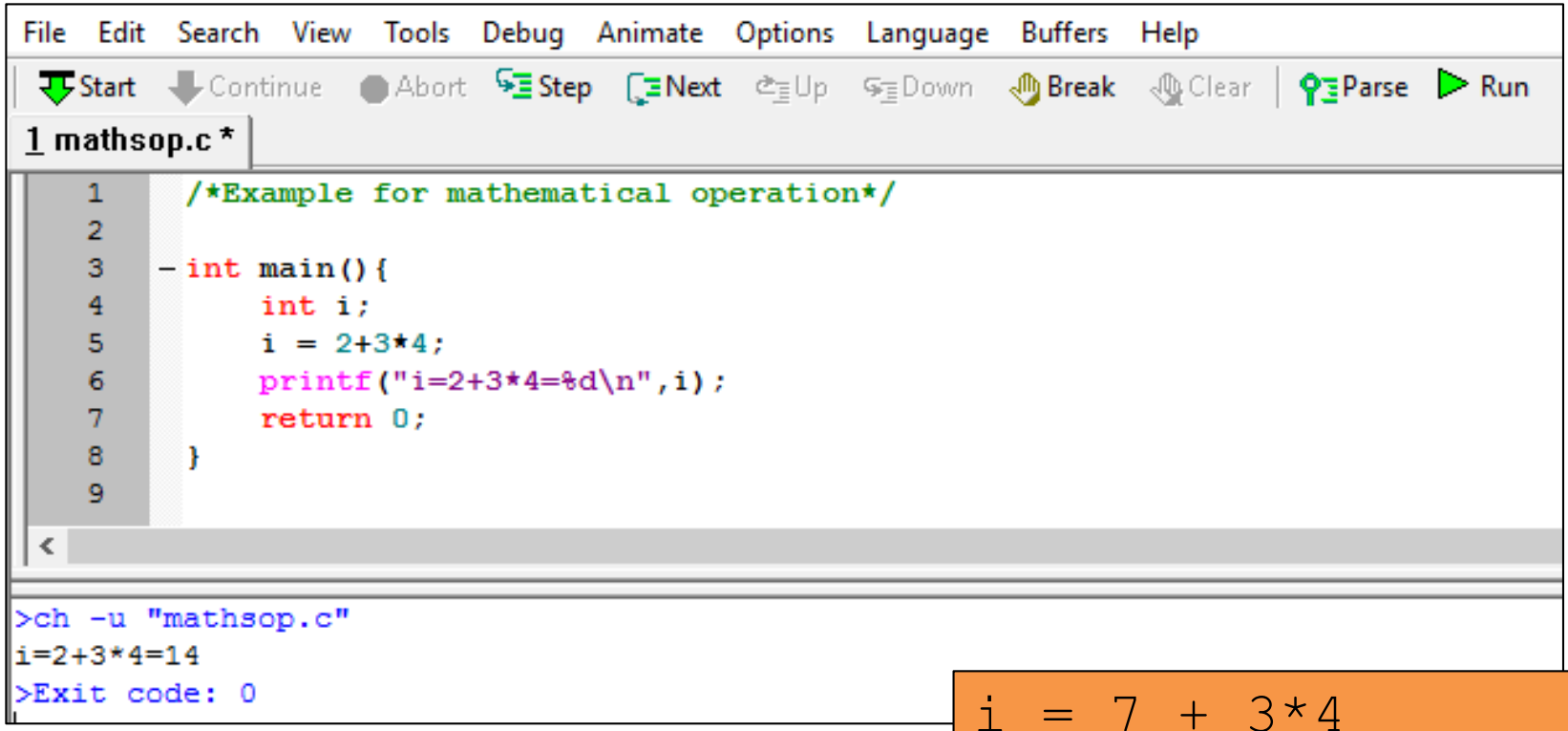
The operator = the lowest in the expression i = 2 + 3 *4:

```
i = 7 + 3*4
  = 7 + 12
  = 14
```

The order of the precedence for the operators here are: *, +, and =.

Operations	Associativity
::	Left to right
() []	Right to left
function name()	Left to right
. ->	Right to left
' ! ` ++ -- + - * &(type) sizeof	Left to right
* / % .* ./	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
^^	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= |= <<= >>=	Right to left
,	Right to left

Arithmetic Operations




```
i = 7 + 3*4
  = 7 + 12
  = 14
```

Header file `math.h`

- The `math.h` header defines various mathematical functions
- All the functions available in this library take `double` as an argument and returns `double` as the result.

Library functions in `math.h`

function	description
`cos(x)`	Returns the cosine of a radian angle x.
`sin(x)`	Returns the sine of a radian angle x.
`tan(x)`	Returns the tan of a radian angle x.
`log(x)`	Returns the natural logarithm (base-e logarithm) of **x**.
`log10(x)`	Returns the common logarithm (base-10 logarithm) of **x**.
`exp(double x)`	Returns the value of **e** raised to the xth power.
`pow(x, y)`	Returns x raised to the power of **y**
`sqrt(x)`	Returns the square root of **x**.
`cosh(x)`	Returns the hyperbolic cosine of a radian angle x.
`sinh(x)`	Returns the hyperbolic sine of a radian angle x.
`tanh(x)`	Returns the hyperbolic tan of a radian angle x.

*Please refer to https://en.wikibooks.org/wiki/C_Programming/C_Reference/math.h for the complete list

Math Functions

Example 1: Calculate the following:

$$p = 2^3 \quad \text{and} \quad \sqrt{x}$$

Write a program to solve for $p = 2^3$ and $\sqrt{x}$

Note: there is no exponential operator in C so you should use the mathematical function:

- `pow(x,y)` to calculate the exponential expression x^y
- `sqrt(x)` to calculate $\sqrt{x}$

These functions are declared inside the header file **math.h**

Math Functions

Example 1:

chdemos

```
1   /* File: powsqrt.c */
2   #include <stdio.h>
3   #include <math.h>  /* for pow() and sqrt() */
4
5   int main() {
6       double p, x = 2.0, y = 3.0;
7       p = pow(x,y);
8       printf("pow(2,3) = %f\n", p);
9       printf("sqrt(2) = %f\n", sqrt(x));
10      return 0;
11  }
12
```

```
>ch -u "MA15024.c"
pow(2,3) = 8.000000
sqrt(2) = 1.414214
>Exit code: 0
```

Math Functions

Example 1:

chdemos

```
1   /* File: powsqrt.c */
2   #include <stdio.h>
3   #include <math.h>  /* for pow() and sqrt() */
4
5   int main() {
6       double p, x = 2.0, y = 3.0;
7       p = pow(x,y);
8       printf("pow(2,3) = %f\n", p);
9       printf("sqrt(2) = %f\n", sqrt(x));
10      return 0;
11  }
12
```

```
>ch -u "MA15024.c"
pow(2,3) = 8.000000
sqrt(2) = 1.414214
>Exit code: 0
```

Math Functions

Example 2: Write a program to solve the roots of the quadratic function

$$x^2 - 5x + 6 = 0$$

Recall that the roots of the quadratic function,

$$ax^2 + bx + c = 0$$

Is given by:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Math Functions

Example 2:

```c
/* File: secondorder1.c */
#include <stdio.h>
#include <math.h>
int main() {

    double a = 1.0, b = -5.0, c = 6.0, x1, x2;
    x1 = (-b +sqrt(b*b-4*a*c))/(2*a);
    x2 = (-b -sqrt(b*b-4*a*c))/(2*a);
    printf("x1 = %f\n", x1);
    printf("x2 = %f\n", x2);

    return 0;
}
```

```
>ch -u "MA15024.c"
x1 = 3.000000
x2 = 2.000000
>Exit code: 0
```

Math Functions

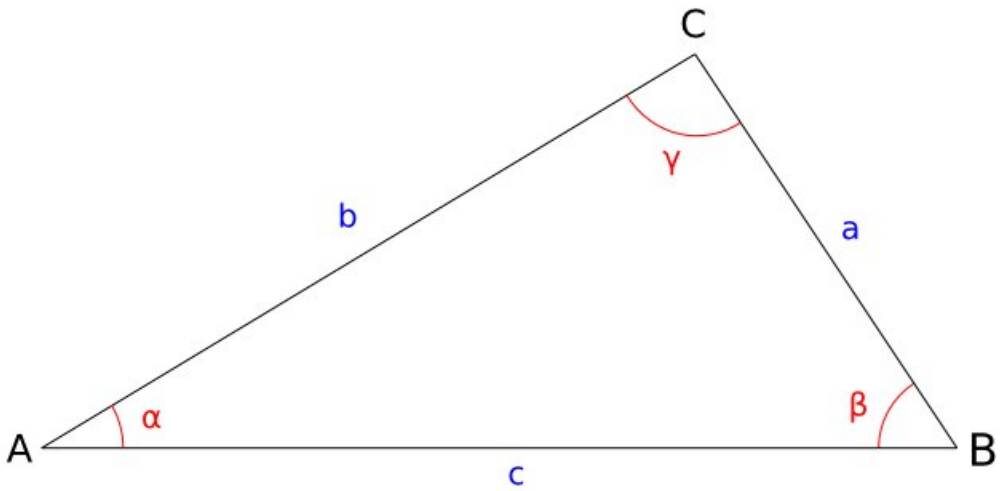
Example 2:

```
1   /* File: secondorder1.c */
2   #include <stdio.h>
3   #include <math.h>
4   int main() {
5
6       double a = 1.0, b = -5.0, c = 6.0, x1, x2;
7       x1 = (-b+sqrt(b*b-4*a*c))/(2*a);
8       x2 = (-b-sqrt(b*b-4*a*c))/(2*a);
9       printf("x1 = %f\n", x1);
10      printf("x2 = %f\n", x2);
11
12      return 0;
13  }
14
```

```
>ch -u "MA15024.c"
x1 = 3.000000
x2 = 2.000000
>Exit code: 0
```

Math Functions

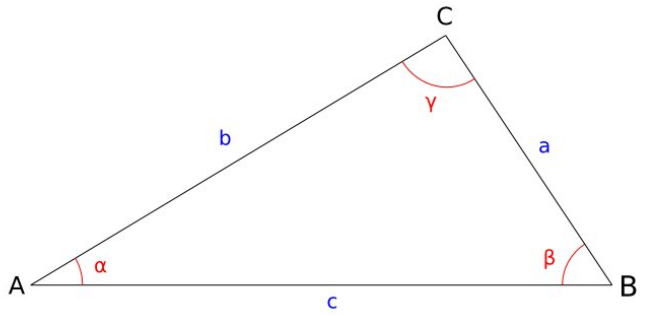
Example 2: Write a C program to calculate the side c, given: $a = 10\ cm, \alpha = 10°, \gamma = 60°$



$$\frac{a}{\sin(\alpha)} = \frac{b}{\sin(\beta)} = \frac{c}{\sin(\gamma)}$$

Math Functions

Example 2: You need to first rearrange the sin law:



$$c = \frac{a\sin(\gamma)}{\sin(\alpha)}$$

$$\frac{a}{\sin(\alpha)} = \frac{b}{\sin(\beta)} = \frac{c}{\sin(\gamma)}$$

Note:

1. Trigonometric functions such as for sine, cosine, `sin(x)`, `cos(x)`, `tan(x)` are declared in the header file `math.h`
2. Recall that these functions return a value of type `double`
3. The unit for the argument of trigonometric functions is in radians, not in degree. So you need to *convert* the values of ***α*** and ***γ***
   - To do this, we have defined a constant π using `#define`:

```
#defined PI 3.14159265359
```

Math Functions

Example 3:

```c
/* File: sinelaw.c */
#include <stdio.h>
#include <math.h>

#define PI 3.14159265358979323846

int main() {

    double c, a, alpha, gamma;
    a = 10; /*side a*/
    alpha=90*PI/180; /*convert from degree to radians*/
    gamma = 60*PI/180; /*convert from degree to radians*/

    c = a*sin(gamma)/sin(alpha);
    printf("c = %f\n",c);

    return 0;
}
```

```
>ch -u "lecture4test.c"
c = 8.660254
>Exit code: 0
```

Conclusion

- Conclusion #1
  - Be careful with precedence when dealing with operators

- Conclusion #2
  - In order to use math functions, you need to include the header file `math.h`
  - The math functions return a double
  - The units of the trigonometric functions in `math.h` is in radians. So if the units is in degree it *must* be converted to radians

Technical Informatics I

Lecture 3

Dr Fatimah