University of Glasgow

UESTC 1005 – Introductory Programming

Lecture 2 – Input, Variables

Dr Hasan T Abbas
Hasan.Abbas@glasgow.ac.uk

Fall 2019

Glasgow College – UESTC

Variables

Storage locations in the memory where we can store the data

- Variable types
  - Integer (int)
    - Example:
      1212
      -232
      0
  - Floating point (float)
    - Example:
      645.345
      0.00000001
      -273.15
  - Characters (char)
    - Example:
      A
      B
      £

Using Variables

```
// Volume of a sphere

// declare the variable
int radius;

// initialize the variable radius
radius = 1;

// calculate volume
float volume;
volume = 4.0/3.0*3.141516*radius*radius*radius;

// display the value
printf("Volume of %d radius sphere is %f", radius, volume);
```

WARNING – Notice!

```
volume = 4/3*3.141516*radius*radius*radius;
```

Will generate incorrect output!

Names of Variables - Identifiers

WARNING!

The names of the variables called identifiers can only begin with a letter or an underscore (_)

They can only contain letters, numbers and underscores

Valid Identifiers

```
Volume, volume, rad1us, _punk, _100real;
```

Invalid identifiers

```
10Volume, f(x), color-ful, void;
```

Keywords

We CANNOT some words called keywords as variable identifiers.

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

Operations on Variables

Arithmetic Operations (`*,/,+,-,%`)

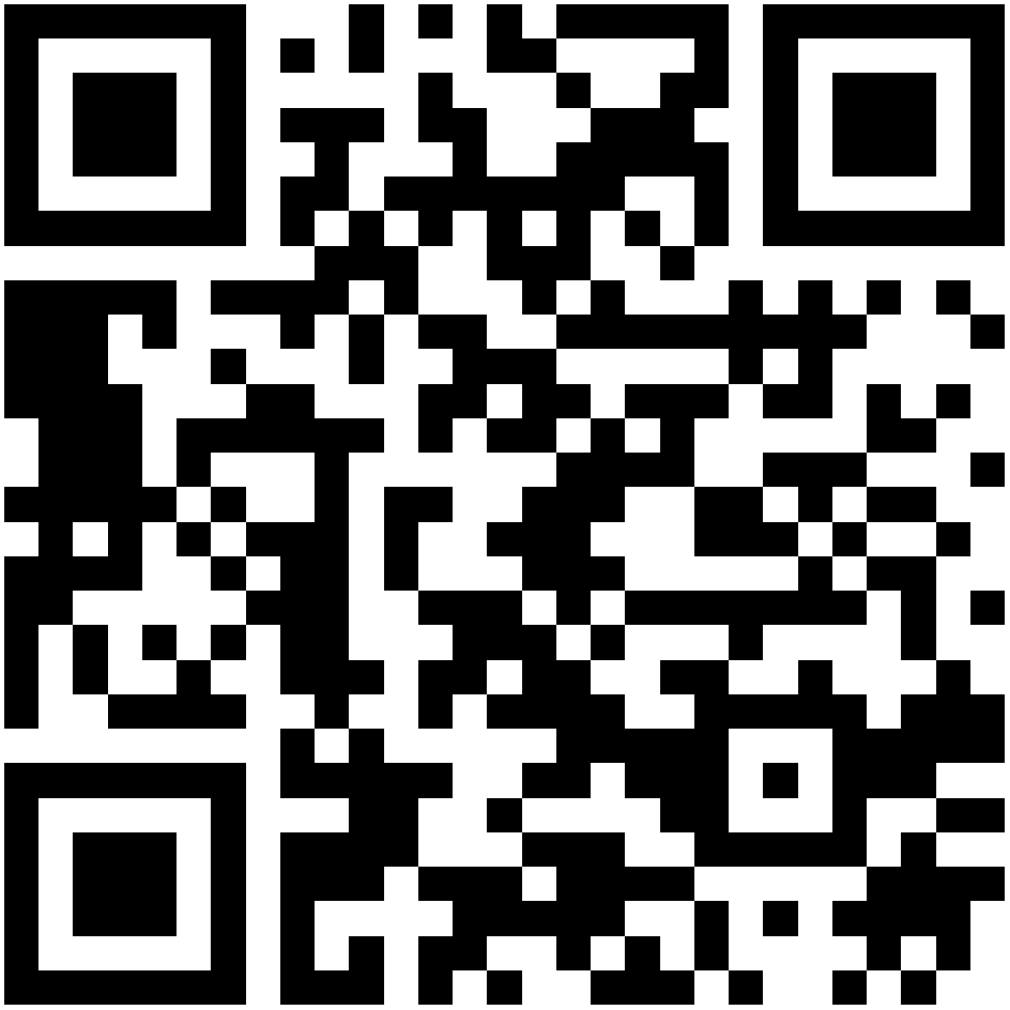
```
volume = 4/3 = 1;
remainder = 5%3 = 2;
```

For complex mathematical operations, we use parentheses.

- Mathematical operations follow a certain order
  1. ()
  2. *,/,%
  3. + and –

EXERCISE: what is the result of:

```
(3 + 8/3) – 12/3*5 + 10%3;
```



ROOM NAME: **AWJUQZW**

Notes on Operations on Variables

C does EXACTLY what we ask it to do.

We have to be careful.

```
printf("%f", (float) (5/2));
printf("%f", (float) 5/2);
```

```
The first line results in 2.000000

The second line results in 2.500000
```

Reading an Input

```
// Read an Input

// declare the variable
int radius;

// Read an integer value
scanf("%d", &radius);
```

- As we saw earlier, variables are locations in memory to store data
- `scanf( )` is a function to store data at a particular location (variable).
- %d → integer input; & → memory location

```
// Read a floating point value
scanf("%f",&radius);
```

Reading an Input – Interactive Way

```
// Read an Input

// declare the variable
int radius;

// Give a message to the user
printf("Enter a integer value for radius \n");

// Read an integer value
scanf("%d", &radius);
```

- Exercise
  - Try inputting a character rather than a number, e.g. 'a' instead of 1. Also try 2.3.
  - *The results will be abnormal*

Reading an Input – Dangers

```
// Read an Input

// declare the variable
float length, width;

// Give a message to the user
printf("Enter values for and length and width \n");

// Read an integer value
scanf("%f %f", &length, width);
```

- We should be extremely careful in using scanf() properly. It can give us unexpected result and the program may crash.
- `scanf("%f %f", &length, width);`
- `scanf("%d %f", &length, width);`
- `scanf("%f %f", &length, &width);`

Constants

- When some values are repeatedly used, it is better to give them names and create a constant, through macro definition
  - PI is 3.14159

```
#define PI 3.14159
```

- Convention is we only use UPPERCASE letters to define constants
- `#define` is another preprocessor directive

Constants - Example

```
#include<stdio.h>

// Another directive
#define INCH_CM 2.54

// A program to convert inches to centimeters
int main(){

    float measurement, new_units;

    measurement = 39.37;

    // Convert using constant
    new_units = measurement * INCH_CM;

    printf("%f inches are %f centimeters", measurement, new_units);

    return 0;
}
```

Logical Operators – Decision Making

Standard algebraic equality operator 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

Applying Conditions

Applying logic is one of the most important features of programming languages

What happens to my grades IF I complete all the labs.

Erin will only go to party IF Qin is going.

IF it is warm outside, THEN turn on the AC, ELSE keep it off.

```
int main(){

    int temperature = 35; // warm temperature ☹
    int ac_switch; // ON -> 1; OFF -> 0
    if (temp > 35)
    {
            ac_switch = 1; // Turn ON the air conditioner
            printf("It is HOT! ;-O");
    }
    return 0;
}
```

Next Lectures

Continue conditional statements

- `if() then elseif(), else`

Loop Environments

- `while() loop`
- `for() loop`