COMP0204: Introduction to Programming for Robotics and AI

Information Representation

Course lead: Dr Sophia Bano

Lecture by: Dr Yunda Yan

MEng Robotics and AI

UCL Computer Science

Recap (previous week)

Key Concepts

- Overview of C programming
- Setting up the C development environment
- Basic C syntax and structure
- Variables and data types
- Fundamental operations

Importance of C

- Versatile and foundational programming language
- Widely used in system programming, embedded systems, and application development

Recap (previous week)

Syntax & Structure

- Basic syntax, development and design of program

Variables & Data Types

- Explored how to declare and use variables to store data
- Discussed different data types available in C

Operations

- Introduced basic operations, including arithmetic, relational, logical, and bitwise operations

Type conversion, order of evaluation

This week

- Information representation – understanding how computer sees the data
- Design and development – understanding how to develop a program
- Control flows – if-else, for, while, switch

Information Representation

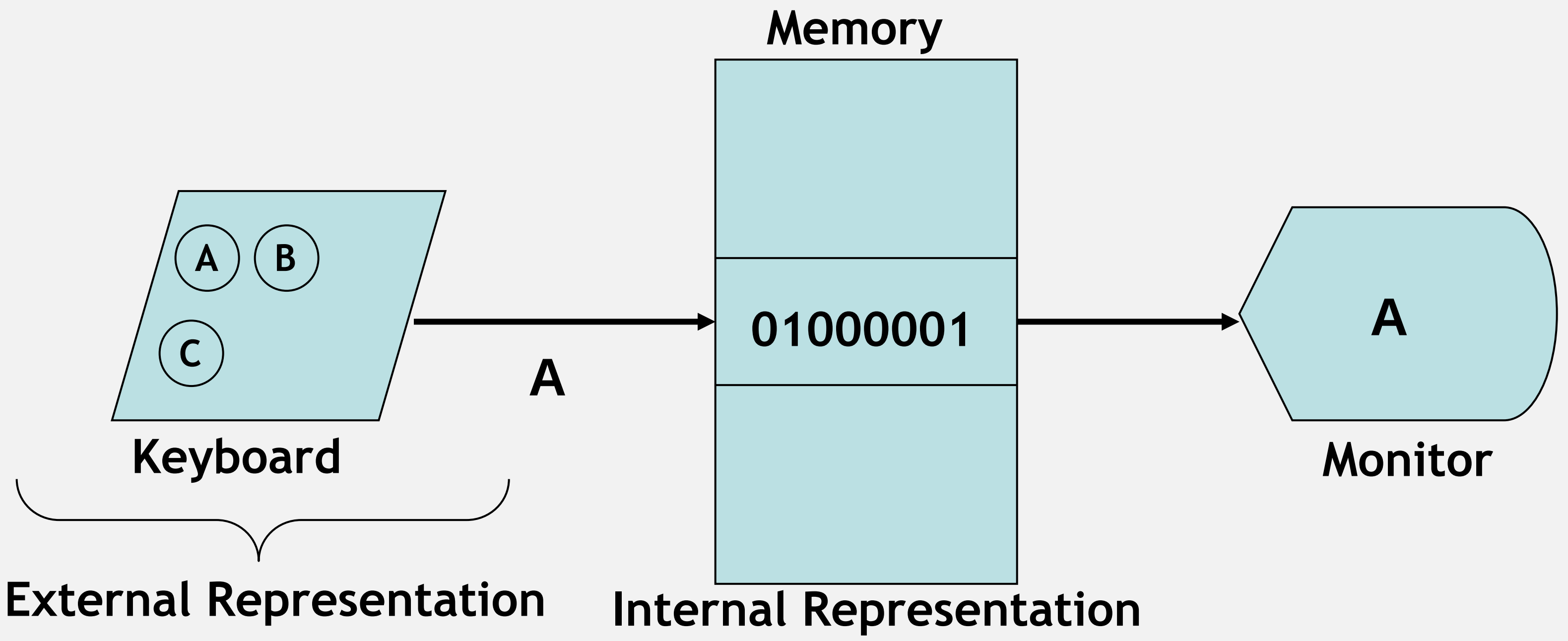
- There are two types of information representation.
  - External representation
  - Internal representation

- **External Representation** of information is the way that how the information is represent by the humans and the way it is entered by at a keyboard or displayed on a printer or screen.

- **Internal Representation** of information is the way it is stored in the memory of a computer or passed to any device of computer.

Information Representation

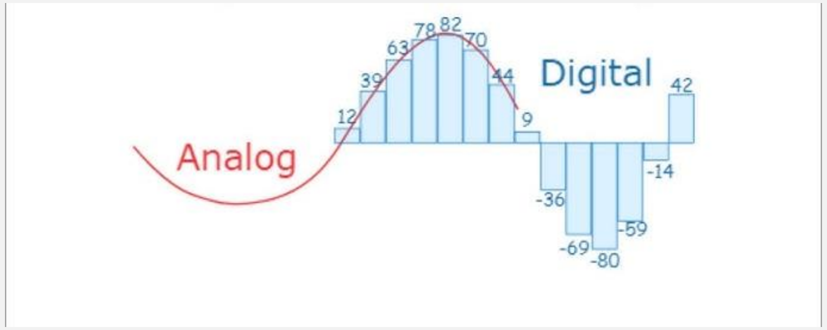
Externally, computers do use decimal digits, sign/magnitude notations, and the familiar 26-character alphabets. However, virtually every computer ever build stores data- **numbers, letters, graphics-internally** using the **binary numbering system**.



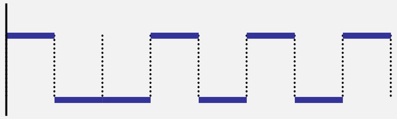
Information Representation

Computer use a binary systems

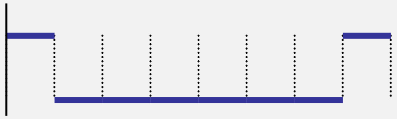
- Why binary?
  - Electronic bi-stable environment
    - on/off, high/low voltage
    - Bit: each bit can be either 0 or 1



- Reliability
  - With only 2 values, can be widely separated, therefore clearly differentiated
- Example:



1 0 0 1 0 1 0 1



1 0 0 0 0 0 0 1

Binary Representation in Computer System

- All information of diverse type is represented within computers in the form of **bit patterns**.
- e.g., text, numerical data, sound, and images
- One important aspect of computer design is to decide how information is converted ultimately to a bit pattern
- Writing software also frequently requires understanding how information is represented along with accuracies

Number Systems - Decimal Number System

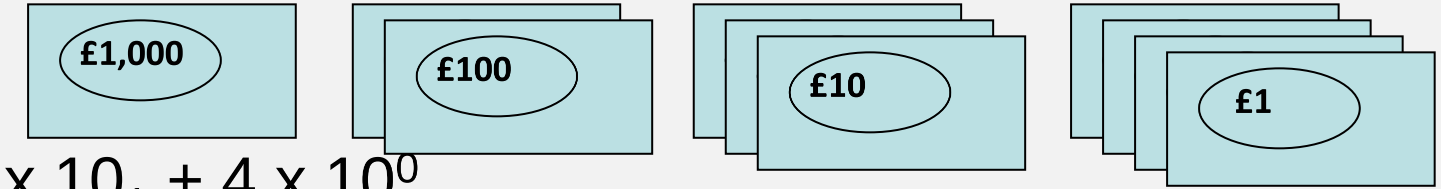
- Base is 10 or ‘D’ or ‘Dec’
- Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Each place is weighted by the power of 10

Example:

1234_{10} or 1234_{D}

$= 1 \times 10^{3} + 2 \times 10^{2} + 3 \times 10_{1} + 4 \times 10^{0}$

$= 1000 + 200 + 30 + 4$



Number Systems

- Number systems are different ways to represent numeric values.

- Three common number bases:
  - Binary
  - Decimal
  - Hexadecimal.

- Each base has its own symbols and rules.

Number Systems - Binary Number System

- Base is 2 or ‘b’ or ‘B’ or ‘Bin’
- Two symbols: 0 and 1
- Each place is weighted by the power of 2

Example: 1011_2 *or* 1011_B

$= 1 \text{ x } 2^3 + 0 \text{ x } 2^2 + 1 \text{ x } 2^1 + 1 \text{ x } 2^0$

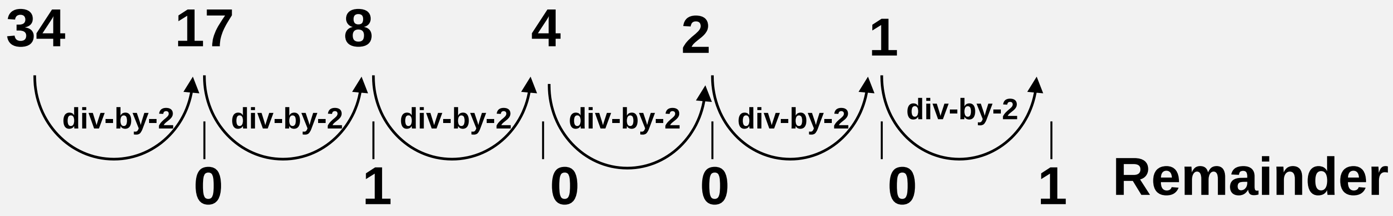
$= 8 + 0 + 2 + 1$

$= 11_{10}$

11 in decimal number system is 1011 in binary number system

Conversion from Decimal and Binary

- To represent **34_{10}** in the binary number system, we use the divide-by-2 technique repeatedly.



- Write the remainder from right to left :

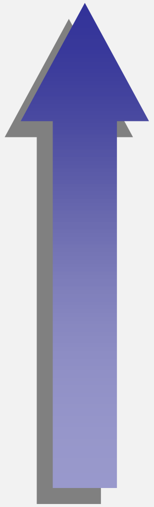
$$34_{10} \equiv 1 \text{ x } 2^5 + 0 \text{ x } 2^4 + 0 \text{ x } 2^3 + 0 \text{ x } 2^2 + 1\text{x } 2^1 + 0 \text{ x } 2^0$$

$$\equiv 100010_2$$

Conversion from Decimal and Binary

- Convert 165_{10} to binary (base 2)

165 / 2	= 82	rem 1
82 / 2	= 41	rem 0
41 / 2	= 20	rem 1
20 / 2	= 10	rem 0
10 / 2	= 5	rem 0
5 / 2	= 2	rem 1
2 / 2	= 1	rem 0
1 / 2	= 0	rem 1



$165_{10} = 10100101_2$

How many bytes are needed to represent 165?

Conversion from Binary to Decimal

- Convert 100010_2 is decimal

$$100010_2 \equiv 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$\equiv 32 + 0 + 0 + 0 + 2 + 0$$

$$\equiv 34_{10}$$

Number Systems - Hexadecimal Number System

- Hexadecimal Number System
  - Base = 16 or ‘H’ or ‘Hex’
  - 16 symbols:
    { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A(=10), B(=11), C(=12), D(=13), E(=14), F(=15)}

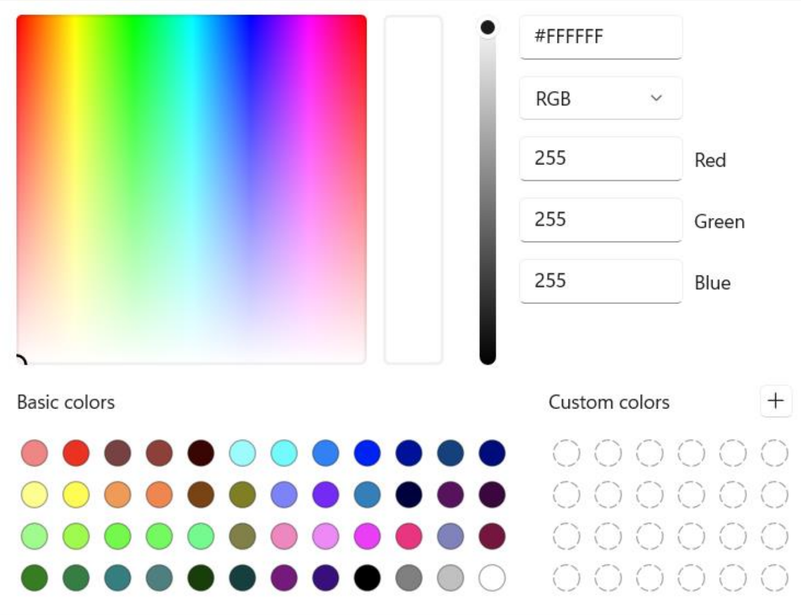
- Hexadecimal to Decimal

$(a_{n-1}a_{n-2}\ldots a_1a_0)_{\mathbf{16}} = (a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} + \ldots + a_1 \times 16^1 + a_0 \times 16^0)_{\mathbf{D}}$

<u>Example</u>: $(1C7)_{16} = (1 \times 16^2 + 12 \times 16^1 + 7 \times 16^0)_{10} = (256 + 192 + 7)_{10} = (455)_{10}$

Hexadecimal Number System

- Decimal to Hexadecimal Repeated division by 16
  - Similar in principle to generating binary codes



#FFFFFF

RGB

255 Red

255 Green

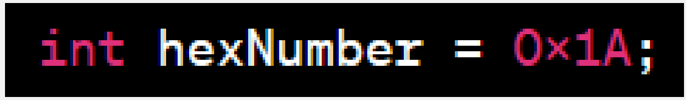
255 Blue

Example: $(829)_{10} = (?)_{16}$

Divide-by-16	Quotient	Remainder
829 / 16	51	13
51 / 16	3	3
3 / 16	0	3

Stop, since quotient = 0

Hence, $(829)_{10}$ = $(33D)_{16}$



```
int hexNumber = 0x1A;
```

How a variable is initialized as hexadecimal

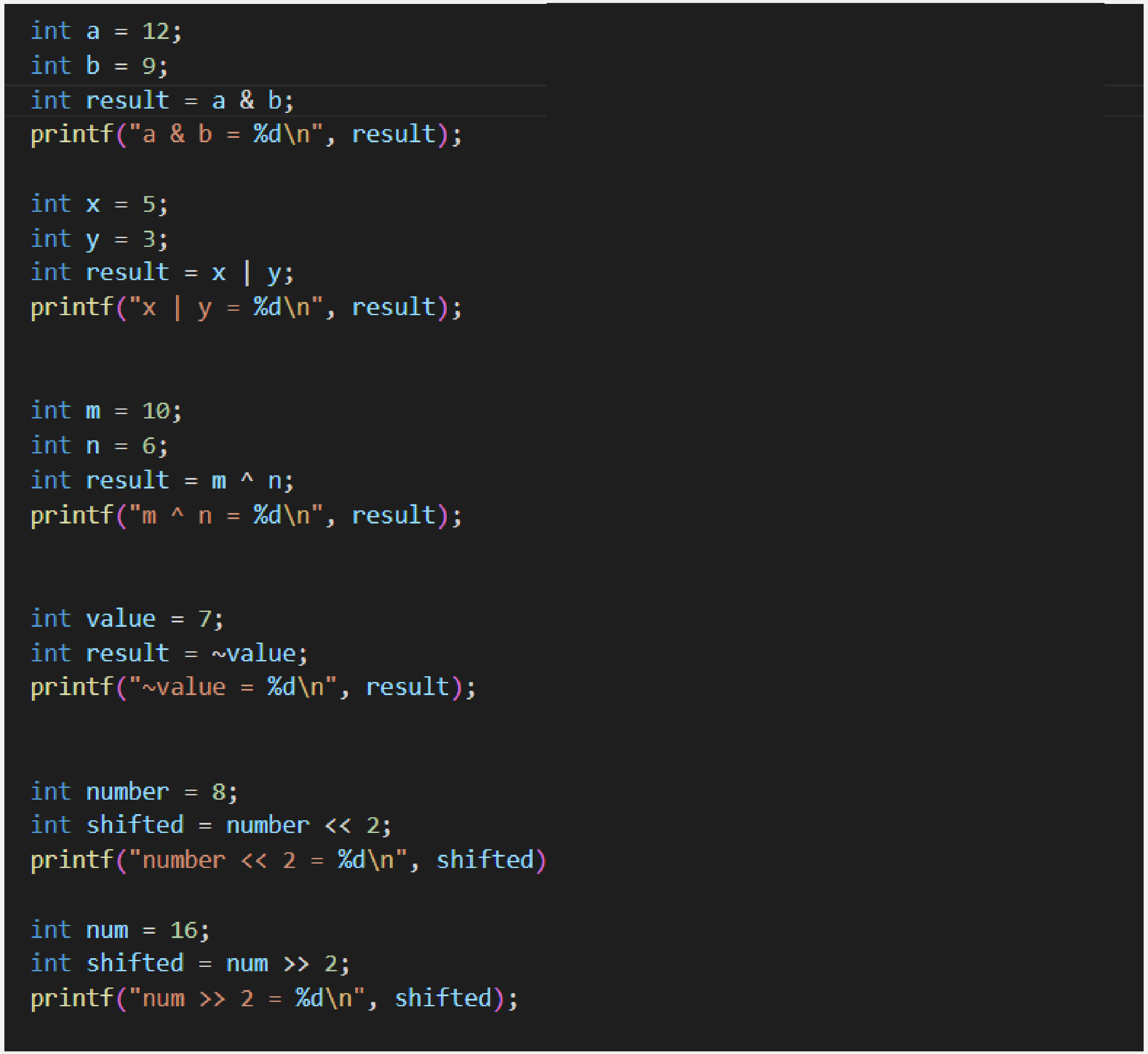
Bitwise operators

- Used for performing operations on individual bits of integers (usually ‘int’ and ‘char’ data types)
- Allow to manipulate the binary representation of integers

Operator	
Bitwise AND (&)	If both bits are 1, the result bit is 1; otherwise, it's 0
Bitwise OR (|)	If at least one of the bits is 1, the result bit is 1; otherwise, it's 0
Bitwise XOR (^)	If the bits are different (one is 0 and the other is 1), the result bit is 1; otherwise, it's 0
Bitwise NOT (~):	1s become 0s, and 0s become 1s
Bitwise Left Shift (<<)	Shifts the bits of an integer to the left by a specified number of positions
Bitwise Right Shift (>>):	Shifts the bits of an integer to the right by a specified number of position

Bitwise operators

- Examples



```c
int a = 12;
int b = 9;
int result = a & b;
printf("a & b = %d\n", result);

int x = 5;
int y = 3;
int result = x | y;
printf("x | y = %d\n", result);


int m = 10;
int n = 6;
int result = m ^ n;
printf("m ^ n = %d\n", result);


int value = 7;
int result = ~value;
printf("~value = %d\n", result);


int number = 8;
int shifted = number << 2;
printf("number << 2 = %d\n", shifted)

int num = 16;
int shifted = num >> 2;
printf("num >> 2 = %d\n", shifted);
```