



NO. 1

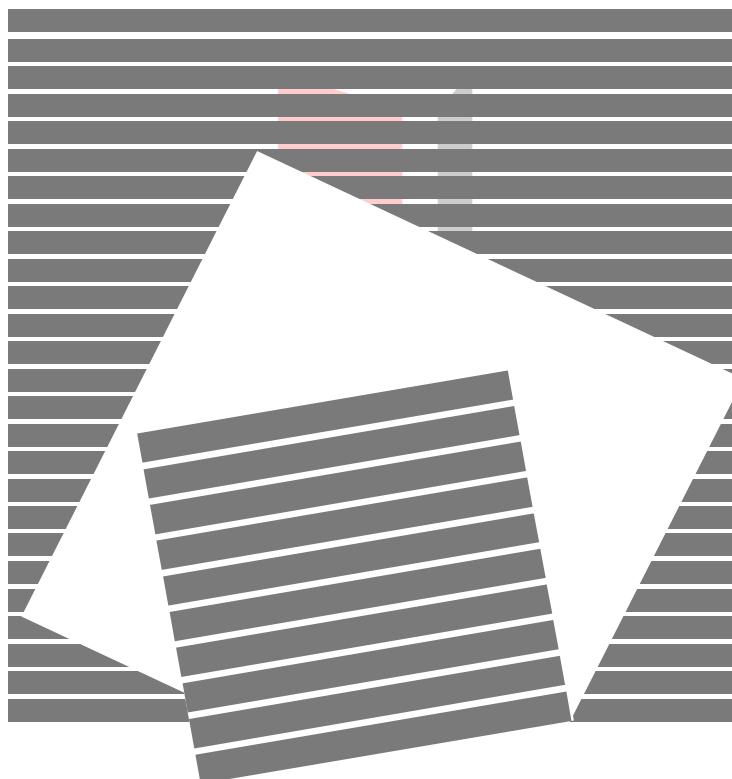
## Textbook for Software Design & Development Engineers

# COMPUTER BASICS

## AND

# COMPUTER SYSTEMS

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



Second Edition

REVISED AND UPDATED BY



Japan Information Processing Development Corporation  
Japan Information-Technology Engineers Examination Center

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



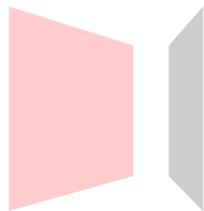
<http://www.vitec.org.vn>

Photos provided by:

I-O Data Device, Inc.  
Intel Corporation  
Uchida Yoko Co., Ltd.  
Canon Sales Co., Inc.  
Sharp Corp.  
Sumitomo Electric Industries, Ltd.  
Seiko Instruments Inc.  
Sekonic Co., Ltd.

Sony Corp.  
IBM Japan, Ltd.  
NEC Corp.  
HAL Corporation  
Fujitsu Ltd.  
Microsoft Co., Ltd.  
Ricoh Co., Ltd.

# Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



- Microsoft®, MS-DOS®, Microsoft® Windows® and Microsoft® Windows NT® are registered trademarks of Microsoft Corporation of the United States in the United States and other countries.
- The product names appearing in this textbook are trademarks or registered trademarks of the respective manufacturers.

<http://www.vitec.org.vn>

---

Textbook for Software Design & Development Engineers

## No. 1 COMPUTER BASICS AND COMPUTER SYSTEMS

---

First edition first printed October 1, 2001  
Second edition first printed August 1, 2002

Japan Information Processing Development Corporation

### Japan Information-Technology Engineers Examination Center

TIME 24 Building 19th Floor, 2-45 Aomi, Koto-ku, Tokyo 135-8073 JAPAN

©Japan Information Processing Development Corporation/Japan Information-Technology Engineers  
Examination Center 2001,2002

# ***Table of Contents***

---

## **1. Basic Theories of Information**

<b>Introduction</b>	<b>1-1</b>
<b>1.1 Data representation</b>	<b>1-2</b>
<b>1.1.1 Numeric conversion</b>	1-2
<b>1.1.2 Numeric representation</b>	1-16
<b>1.1.3 Operation and precision</b>	1-33
<b>1.1.4 Non-numeric value representation</b>	1-36
<b>1.2 Information and logic</b>	<b>1-40</b>
<b>1.2.1 Proposition logic</b>	1-40
<b>1.2.2 Logical operation</b>	1-41
<b>Exercises</b>	<b>1-44</b>

## **2. Data Structures**

<b>2.1 What is the data structure?</b>	<b>2-2</b>
<b>2.2 Basic data structure</b>	<b>2-3</b>
<b>2.2.1 Basic data type</b>	2-3
<b>2.2.2 Structured type</b>	2-4
<b>2.2.3 Abstract data type</b>	2-7
<b>2.3 Problem-oriented data structure</b>	<b>2-8</b>
<b>2.3.1 List structure</b>	2-8
<b>2.3.2 Stack</b>	2-11
<b>2.3.3 Queue</b>	2-12
<b>2.3.4 Tree Structure</b>	2-13
<b>2.3.5 Hash</b>	2-18
<b>2.3.6 Backus-Naur Form</b>	2-20
<b>2.3.7 EBNF (extended BNF)</b>	2-20
<b>Exercises</b>	<b>2-22</b>



## **3. Algorithms**

<b>Introduction</b>	<b>3-2</b>
<b>3.1 Basic of algorithms</b>	<b>3-2</b>

<b>3.1.1 What is an algorithm</b>	<b>3-2</b>
<b>3.1.2 Algorithm and the data structure</b>	<b>3-4</b>
<b>3.2 Various Algorithms</b>	<b>3-9</b>
<b>3.2.1 Search algorithm</b>	<b>3-9</b>
<b>3.2.2 Sort algorithm</b>	<b>3-14</b>
<b>3.2.3 Recursive algorithm</b>	<b>3-29</b>
<b>3.2.4 Character string processing</b>	<b>3-32</b>
<b>3.2.5 File processing</b>	<b>3-37</b>
<b>3.2.6 Drawing figures</b>	<b>3-44</b>
<b>3.2.7 Graph</b>	<b>3-48</b>
<b>3.2.8 Numeric calculation</b>	<b>3-53</b>
<b>3.2.9 Collation algorithm</b>	<b>3-62</b>
<b>3.2.10 Approximate and probability algorithms</b>	<b>3-65</b>
<b>3.2.11 Genetic algorithms</b>	<b>3-71</b>
<b>3.2.12 Data compression algorithms</b>	<b>3-81</b>
<b>3.2.13 Memory management algorithms</b>	<b>3-91</b>
<b>3.2.14 Compiler algorithms</b>	<b>3-94</b>
<b>3.2.15 Programming Languages</b>	<b>3-97</b>
<b>3.3 Evaluation of algorithms</b>	<b>3-115</b>
<b>3.3.1 Evaluation by computational complexity</b>	<b>3-115</b>
<b>3.3.2 Evaluation by validity</b>	<b>3-116</b>
<b>3.3.3 Evaluation by representation</b>	<b>3-116</b>
<b>3.4 How to design algorithms</b>	<b>3-117</b>
<b>3.5 Applications of algorithms</b>	<b>3-118</b>
<b>3.5.1 Routing</b>	<b>3-118</b>
<b>3.5.2 Search engines</b>	<b>3-119</b>
<b>3.5.3 Cryptography</b>	<b>3-122</b>
<b>3.5.4 Data mining</b>	<b>3-124</b>
<b>Exercises</b>	<b>3-129</b>

## **4. Hardware**

<b>Introduction</b>	<b>4-2</b>
<b>4.1 Information element</b>	<b>4-3</b>
<b>4.1.1 Integrated circuit</b>	<b>4-3</b>
<b>4.1.2 Semiconductor memory</b>	<b>4-3</b>
<b>4.2 Processor architecture</b>	<b>4-6</b>
<b>4.2.1 Processor structure and operation principles</b>	<b>4-6</b>
<b>4.2.2 Speed performance enhancement in processor</b>	<b>4-21</b>

<b>4.2.3 Operation mechanism</b>	<b>4-25</b>
<b>4.2.4 Multi-processor</b>	<b>4-30</b>
<b>4.2.5 Processor performance</b>	<b>4-31</b>
<b>4.3 Memory architecture</b>	<b>4-33</b>
<b>4.3.1 Memory types</b>	<b>4-33</b>
<b>4.3.2 Memory capacity and performance</b>	<b>4-34</b>
<b>4.3.3 Memory configuration</b>	<b>4-36</b>
<b>4.4 Auxiliary storage devices</b>	<b>4-38</b>
<b>4.4.1 Types and characteristics of auxiliary storage devices</b>	<b>4-38</b>
<b>4.4.2 RAID types and characteristics</b>	<b>4-53</b>
<b>4.5 Input/output architecture and devices</b>	<b>4-55</b>
<b>4.5.1 Input/output control method</b>	<b>4-55</b>
<b>4.5.2 Input/output interfaces</b>	<b>4-57</b>
<b>4.5.3 Types and characteristics of input/output devices</b>	<b>4-62</b>
<b>4.6 Computer types</b>	<b>4-77</b>
<b>4.6.1 Computer types</b>	<b>4-77</b>
<b>4.7 Wireless standards and mobile computing</b>	<b>4-82</b>
<b>4.7.1 Transmission methods</b>	<b>4-82</b>
<b>4.7.2 Bluetooth</b>	<b>4-85</b>
<b>4.7.3 IEEE802.11</b>	<b>4-86</b>
<b>4.7.4 Network Architecture</b>	<b>4-87</b>
<b>4.7.5 Security</b>	<b>4-89</b>
<b>4.7.6 Characteristics of mobile computing</b>	<b>4-90</b>
<b>4.7.7 Mobile computing infrastructure</b>	<b>4-91</b>
<b>4.7.8 Type of software</b>	<b>4-92</b>
<b>4.7.9 Mobile based protocols</b>	<b>4-95</b>
<b>Exercises</b>	<b>4-100</b>

## **5. Multimedia system**

<b>5.1 Multimedia System Design</b>	<b>5-2</b>
<b>5.1.1 Types of multimedia applications</b>	<b>5-3</b>
<b>5.1.2 Design Steps</b>	<b>5-4</b>
<b>5.1.3 Needs Assessment</b>	<b>5-5</b>
<b>5.1.4 Audience Assessment</b>	<b>5-5</b>
<b>5.1.5 Define Contents</b>	<b>5-6</b>
<b>5.1.6 Select Authoring and delivery System</b>	<b>5-7</b>
<b>5.1.7 Storyboard</b>	<b>5-14</b>
<b>5.1.8 Define the Interactions</b>	<b>5-15</b>

<b>5.1.9 Create the Scripts</b>	<b>5-18</b>
<b>5.1.10 Feedback</b>	<b>5-19</b>

<b>Answers to Exercises</b>	<b>6-1</b>
-----------------------------	------------

<b>Index</b>
--------------

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

# **1 Basic Theories of Information**

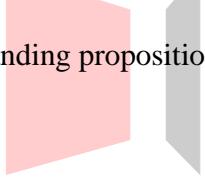
---

## **Chapter Objectives**

Understanding the computer mechanism of data representation and basic theories.

In particular, the binary system is an important subject to learn, indispensable for computer data representation. However, for people who are used to the decimal system, it is hard to become familiar with this representation, so it should be carefully studied

- 1 Understanding a computer's basic data units such as binary numbers, bits, bytes, words etc and their conversion from and to decimal and hexadecimal digits.
- 2 Understanding basic concepts of computer internal data representation, focusing on numeric data, character codes, etc.
- 3 Understanding proposition calculus and logical operations



**VITEC**

<http://www.vitec.org.vn>

# Introduction

---

In order to make a computer work, it is necessary to convert the information we use in daily life into a format that can be understood by the computer. For that reason, the way information is actually represented inside a computer as well as the way it is processed will be learned here.

## 1.1 Data representation

---

### 1.1.1 Numeric conversion

For a computer to do processing, it is first necessary to input into the memory the programs which are contents of a task or processing procedures. The binary system is what is used to represent this information.

While the binary system represents by means of the combination of "0" and "1", we ordinarily use the decimal system. Therefore, an important fundamental knowledge required by information processing engineers is to understand the relationship between binary and decimal numbers. This is the basic difference between computers and human beings as well as the point of contact between them.

Since the mechanism in which the computer operates is completely based on binary numbers, the relationship between binary and decimal numbers, as well as hexadecimal numbers combining binary numbers will be explained.

#### (1) Data representation unit and processing unit

##### ① Binary numbers

The internal structure of a computer is composed of an enormous number of electronic circuits.

Binary numbers represent two levels of status in the electronic circuits, as in:

- Whether the electric current passes through it or not

- Whether the voltage is high or low

For example, setting the status where the electric current passes through (the power is on) to "1" and the status where the electric current does not pass through (the power is off) to "0", then by replacing the computer status or data with numeric values their representation can be easily performed in an extremely convenient way.

The representation of decimal numbers from "0" to "10" using binary numbers is shown in Figure 1-1-1.

**Figure 1-1-1**  
Decimal numbers  
and binary numbers

	Decimal numbers	Binary numbers	
	0	0	
	1	1	
	2	10	A carry occurs
	3	11	
	4	100	A carry occurs
	5	101	
	6	110	
	7	111	
	8	1000	A carry occurs
	9	1001	
A carry occurs	10	1010	

As can be seen in this Figure, compared to the decimal system, a carry occurs more frequently in the binary system, but since besides "0" and "1", no other figure is used, it is the most powerful tool for the computer.

## ② Bits

A bit (binary digit) is 1 digit of the binary system represented by "0" or "1". A bit is the smallest unit that represent data inside the computer. 1 bit can represent only 2 values of data, "0" or "1", but 2 bits represent 4 different values.

- 00

- 01
- 10
- 11

However, in practice, the amount of information processed by a computer is so immense (there are 26 values in the English alphabet alone) that the two bits, 0 and 1, are insufficient for an information representation method.

### ③ Bytes

Compared to a bit which is the smallest unit that represent data inside the computer, a byte is a unit that represents 8 bits of 1 character or number. Considering that a byte is equal to 8 bits, the following is the information which can be represented with one byte, by the combination of "0" and "1".

- 00000000.
- 00000001
- 00000010
- :
- 11111101
- 11111110
- 11111111

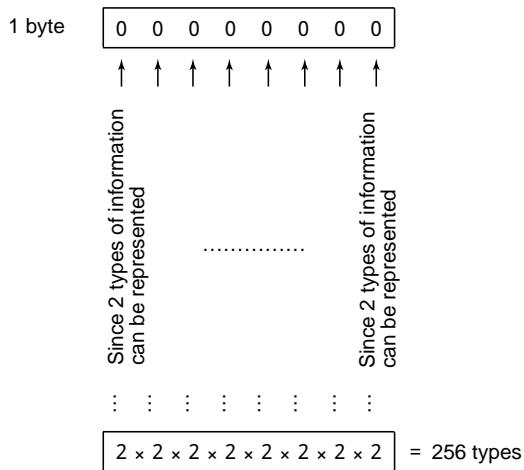


<http://www.vitec.org.vn>

The information represented by a string of "1's" and "0's" is called a bit pattern. Since 1 bit can be represented in two ways, the combination of 8 bit patterns into 1 byte enables the representation of  $2^8=256$  types of information. In other words, besides characters and figures, symbols such as "+" and "-" or special symbols such as "<" and ">" can also be represented with one byte.

**Figure 1-1-2**

Types of information  
that can be represented  
with one byte



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
However, since the number of kanji (Chinese characters) amounts to thousands, they cannot be represented with one byte. Therefore, two bytes are connected to get 16 bits, and one kanji is represented with two bytes. With 16 bits,  $2^{16} = 65,536$  kanji can be represented.

#### ④ Words

A bit is the smallest unit that represents data inside a computer and a byte is a unit that represents 1 character. However, if the computers internal operations were performed on the bit basis, the operation speed would be slow. For that reason the idea of processing using a unit called word was born.

Over 10 years ago, personal computers operated on words each consisting of 16 bits. Currently, mainstream PCs use words each consisting of 32 bits.

#### ⑤ Binary system and hexadecimal system

In information processing, the binary system is used to simplify the structure of the electronic circuits that make up a computer. However, for us, the meaning of a string of "0's" and "1's" is difficult to understand. In the decimal system, the numeric value "255" has 3 digits, but in the binary system the number of digits become 8. Therefore, in order to solve the problem of difficulty in identification and of a large number of digits hexadecimal system is used.

A hexadecimal number is a numeric value represented by 16 numerals, from "0" to "15". When it becomes 16, a carry occurs. However, since it cannot distinguish between the "10" before a carry has been generated, and the "10" after a carry has been generated, for purposes of convenience, in the hexadecimal system "10" is represented by the letter "A", "11" by "B", "12" by "C", "13" by "D", "14" by "E" and "15" by "F".

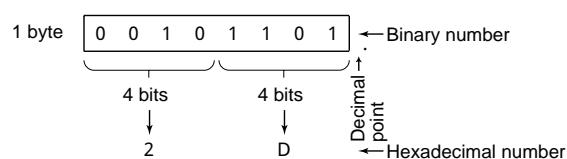
**Figure 1-1-3**  
Decimal numbers,  
binary numbers,  
and hexadecimal numbers

Decimal numbers	Binary numbers	Hexa-decimal numbers
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14

Figure 1-1-3 shows the notation of the numbers "0" to "20" of the decimal system in the binary system and the hexadecimal system.

Focusing on the relationship between the hexadecimal numbers and binary numbers in this table, it can be noted that 4 digits in the binary system correspond to 1 digit in the hexadecimal system. Therefore, binary numbers can be converted to hexadecimal numbers, by replacing each group of 4 bits with a hexadecimal digit, starting from the decimal point. (Figure 1-1-4)

**Figure 1-1-4**  
Binary, and hexadecimal  
counting systems



## (2) Representation of numeric data

By means of the combination of "0's" and "1's", characters are represented as codes. However, a

different representation method is used to process numeric data. Here, radix and radix conversion, the addition and subtraction of binary numbers and hexadecimal numbers, the representation of negative numbers, among other points considered basic for the representation of numeric data, will be explained.

## ① Radix and "weight"

### a. Decimal numbers "weight" and its meaning

When quantities are represented using decimal numbers, 10 types of numerals from "0" to "9" are combined. Each of them, from the lower digit in the ascendant order has a "weight" as  $10^0$ ,  $10^1$ ,  $10^2$ ,  $10^3$ ... (Figure 1-1-5).

For example, using the weight, a decimal number 1,234 would be represented as follows:  
 $1,234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$

**Figure 1-1-5**  
Weight of each digit of  
the decimal number 21998

2	1	9	9	8	Decimal number
Ten thousand	Thousands	Hundred	Ten	Unit	Name of each digit
$10^4$	$10^3$	$10^2$	$10^1$	$10^0$	Weight of each digit

In Figure 1-1-5, the weight of each digit is represented as  $10^0$ ,  $10^1$ ,  $10^2$ ,  $10^3$ , and  $10^4$ . This "10" is called "Radix" and the value placed at the upper right of 10 is called the "Exponent". The notation and meaning of the weight in the decimal system is explained below.

In  $10^0$ , the radix 10 is multiplied 0 times by 1, so it becomes 1, in  $10^1$ , the radix 10 is multiplied 0 times by itself, so it becomes 10.

Likewise, in  $10^2$ , 10 is multiplied 2 times by itself, so it becomes 100; in  $10^3$ , 10 is multiplied 3 times by itself, so it becomes 1000.

In this way, even when the number of digits increases. it can be easily represented by writing down in small numbers, to the upper right of 10, the numeric value that indicates the number of times the radix 10 is multiplied (exponent).

## b. Binary digits "weight" and its meaning

The radix of the decimal system is 10, and the radix of the binary system is 2. As in the decimal system, the weight of each digit in the binary system is shown in Figure 1-1-6.

Figure 1-1-6

Weight of each digit of the binary number 11111001110

1	1	1	1	1	0	0	1	1	1	0	Binary number
$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	Weight of each digit

The notation and meaning of the weight in the binary system is explained below.

In  $2^0$ , the radix 2 is multiplied 0 times by itself, so it becomes 1, in  $2^1$ , the radix 2 is multiplied only 1 time by itself, so it becomes 2. Likewise, in  $2^2$ , 2 is multiplied 2 times by itself, so it

becomes 4.

To verify that the decimal number 1,998 is represented as "11111001110" in the binary system, the weight of each of the digits represented by 1 in the binary representation should be added, as is shown below:

$$\begin{array}{ccccccccccccc} & 1 & & 1 & & 1 & & 1 & & 1 & & 0 & & \\ & \downarrow & & \\ & 2^{10} & + & 2^9 & + & 2^8 & + & 2^7 & + & 2^6 & + & 2^5 & + & 2^4 & + & 2^3 & + & 2^2 & + & 2^1 & + & 2^0 \\ & \downarrow & & \\ = 1,024 & + & 512 & + & 256 & + & 128 & + & 64 & + & & & & & & & & & & & & = 1,998 \end{array}$$

<http://www.vitec.org.vn>

## ② Auxiliary units and power representation

Since the amount of information processed by computers is immense, auxiliary units that represent big amounts are also used. Likewise, since computers operate at high speeds, auxiliary units that represent extremely small amounts are also needed to represent the performance.

Figure 1-1-7 shows the auxiliary units that represent large and small amounts as well as the exponent to which the radix is raised.

**Figure 1-1-7**  
Auxiliary units

	Unit symbol	Exponent notation	Remarks
Units that represent large amounts	T (giga)	$10^{12}$	$2^{40}$
	G (tera)	$10^9$	$2^{30}$
	M (mega)	$10^6$	$2^{20}$
	k (kilo)	$10^3$	$2^{10}$
Units that represent small amounts	m (milli)	$10^{-3}$	$\frac{1}{1,000}$
	$\mu$ (micro)	$10^{-6}$	$\frac{1}{1,000,000}$
	n (nano)	$10^{-9}$	$\frac{1}{1,000,000,000}$
	p (pico)	$10^{-12}$	$\frac{1}{1,000,000,000,000}$

It is important to note that, as indicated in the Remarks column in Figure 1-1-7, kilo is equal to  $10^3$ , but it is almost equal to  $2^{10}$ . In other words, the kilo we ordinary use is equal to 1,000, however, since the binary system is used in computing,  $2^{10}$  (1,024) is a kilo. Furthermore, if  $2^{10}$  and  $10^3$  are almost equal,  $10^6$  that is a mega, is almost equal to  $2^{20}$  and  $10^9$  a giga, is almost equal to  $2^{30}$ .

Therefore, when it is said that a computer memory capacity is 1 kilobyte, strictly speaking, 1 kilobyte does not mean 1,000 bytes, but 1,024 bytes.

### ③ Addition and subtraction of binary numbers

#### a. Addition

The following are the 4 basic additions of the binary system:

- $0 + 0 = 0$  (0 in the decimal system)
- $0 + 1 = 1$  (1 in the decimal system)
- $1 + 0 = 1$  (1 in the decimal system)
- $1 + 1 = 10$  (2 in the decimal system) ← Main characteristic of the binary system that differ from the decimal system

Among these additions, a carry is generated in  $1 + 1 = 10$

$$\begin{array}{r} \boxed{1} \leftarrow \text{Carry} \\ 1 \\ + \quad 1 \\ \hline 10 \end{array}$$

### Example

$$(11010)_2 + (1100)_2$$

$$\begin{array}{r} \boxed{1} \boxed{1} \leftarrow \text{Carry} \\ 11010 \\ + 1100 \\ \hline 100110 \end{array}$$

The result is  $(100110)_2$

### b. Subtraction

The following are the 4 basic subtractions of the binary system:

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

- $0 - 0 = 0$
- $0 - 1 = -1$
- $1 - 0 = 1$
- $1 - 1 = 0$

Among these subtractions, if the upper digit of 0 is 1 in  $0 - 1 = -1$ , a "borrow" is performed.

$$\begin{array}{r} \curvearrowleft \leftarrow \text{Borrow} \\ 10 \\ - 1 \\ \hline 1 \end{array}$$



### Example

<http://www.vitec.org.vn>

$$(10011)_2 - (1001)_2$$

$$\begin{array}{r} \curvearrowleft \leftarrow \text{Borrow} \\ 10011 \\ - 1001 \\ \hline 1010 \end{array}$$

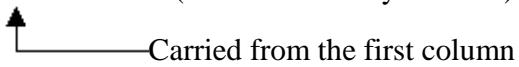
The result is  $(1010)_2$

### ④ Addition and subtraction of hexadecimal numbers

Basically the addition and subtraction of hexadecimal numbers is similar to that of decimal and binary numbers.

- a. Addition is performed starting at the lowest (first from the left) digit. When the addition result is higher than 16, a carry to the upper digit is performed.

$$\begin{array}{r}
 & \curvearrowleft \quad \leftarrow \text{Borrow} \\
 1 & 0 & 0 & 1 & 1 \\
 - & & & & \\
 \hline
 1 & 0 & 0 & 1 \\
 & & & \\
 \hline
 1 & 0 & 1 & 0
 \end{array}$$

- First digit:  $D + 7 =$  (In the decimal system:  $13 + 7 = 20 = 16$  (carried 1) + 4)  
The sum of the first column is 4 and 1 is carried to the second column.
- Second digit:  $1 + 8 + 1 =$  (In the decimal system:  $10 = A$ )  

- Third digit:  $A + B =$  (In the decimal system:  $10 + 11 = 21 = 16$  (carried 1) + 5)  
The sum of the third column is 5 and 1 is carried to the fourth column.

#### b. Subtraction

Subtraction is performed starting from the first column, and when the subtraction result is negative (minus), a borrow from the upper order column is performed.

$$\begin{array}{r}
 & \curvearrowleft \quad \leftarrow \text{Borrow} \\
 6 & D & 3 \\
 - & 1 & 7 & 4 \\
 \hline
 5 & 5 & F
 \end{array}
 \quad \left[ \begin{array}{r}
 & \curvearrowleft \quad \leftarrow 16 \\
 6 & 13 & 3 \\
 - & 1 & 7 & 4 \\
 \hline
 5 & 5 & 15
 \end{array} \right]$$

- First digit: Since  $3 - 4 = -1$ , a borrow is performed from D in the second digit (D becomes C).  $16$  (borrowed 1) +  $3 - 4 = F$  (In the decimal system:  $19 - 4 = 15$ )
- Second digit:  $C - 7 = 5$  (In the decimal system:  $12 - 7 = 5$ )
- Third digit:  $6 - 1 = 5$

The result is  $(55F)_{16}$

### (3) Radix conversion

In order to process numeric values in a computer, decimal numbers are converted into binary or hexadecimal numbers. However, since we ordinarily use decimal numbers, it would be difficult to

understand the meaning of the result if it were represented by binary or hexadecimal numbers.

Therefore, the conversion amongst decimal, binary and hexadecimal numbers is necessary. This operation is called radix conversion,

A concrete example of the conversion amongst the radixes of decimal, binary and hexadecimal numbers which are currently used the most, will be performed below. In order to avoid confusion, the respective radix will be written outside the parenthesis to distinguish between them. For example:

- Notation of binary numbers:  $(0101)_2$
  - Notation of decimal numbers:  $(123)_{10}$

### ① Conversion of decimal numbers into binary numbers

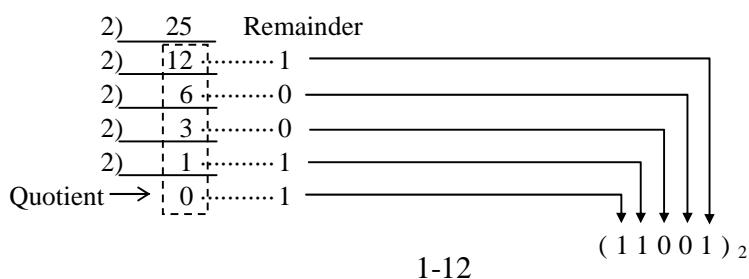
The method of conversion for translating decimal numbers into binary numbers differs depending on whether the decimal number is an integer or a fraction.

#### a. Conversion of decimal numbers

The decimal integer is divided into 2, and the quotient and remainder are obtained. The resulting quotient is divided into 2 again, and the quotient and remainder are obtained. This operation is repeated until the quotient becomes 0.

Since a decimal integer is divided into 2, when the decimal integer is an even number the remainder will be "0," when it is an odd number the remainder will be "1". The binary digit is obtained by placing the remainder (s) in the reverse order.

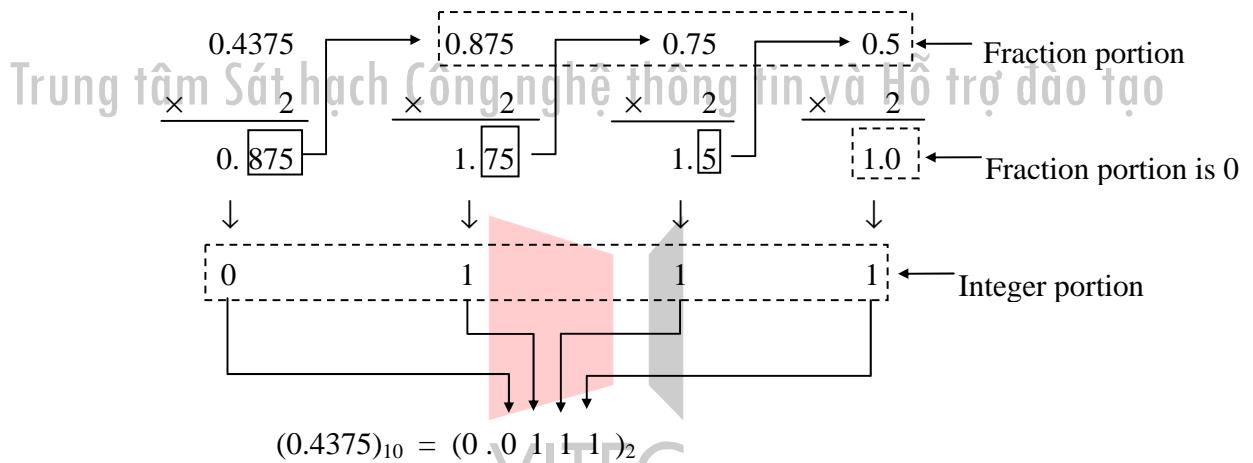
**Example** (25)<sub>10</sub>



### b. Conversion of decimal fractions

The decimal fraction is multiplied by 2, the integer and fraction portion of the product are separated, and the integer section is extracted. Since the integer portion is the product of the multiplication of the fraction portion by 2, it will always be "0" or "1". Next, setting aside the integer portion, only the fraction portion is multiplied by 2. This operation is repeated until the fraction portion becomes 0. The binary digit is obtained by placing the integer portions extracted in the order they were extracted.

**Example**  $(0.4375)_{10}$



It should be noted that when decimal fractions are converted into binary fractions, most of the times, the conversion is not finished, since no matter how many times the fraction portion is multiplied by 2, it will not become 0. In other words, the above mentioned example is that of a special decimal fraction, but most of the decimal fractions become infinite binary fractions.

The verification of the kind of numeric values which correspond to special decimal fractions is performed below. For example, the result of the conversion of the binary fraction 0.11111 into a decimal fraction is as follows:

$$\begin{array}{ccccccccc}
 0. & 1 & 1 & 1 & 1 & 1 & & \leftarrow \text{Binary fractions} \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \\
 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} & 2^{-5} & & & \leftarrow \text{Weight} \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & & \\
 0.5 & + & 0.25 & + & 0.125 & + 0.0625 + 0.03125 = 0.96875 & & \leftarrow \text{Decimal fractions}
 \end{array}$$

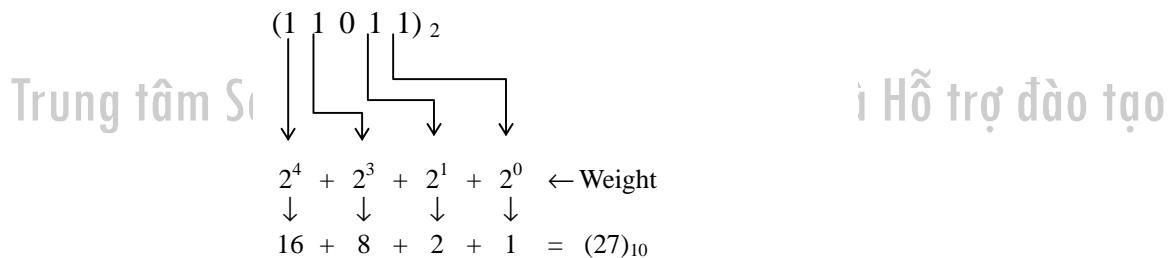
From this example, it can be understood that besides the decimal fractions that are equal to the weight of each digit (0.5, 0.25, 0.125, .... etc,) or the decimal fractions that result from their combination, all other decimal fractions become infinite binary fractions.

## ② Conversion of binary numbers into decimal numbers

The conversion into decimal numbers is performed by adding up the weights of each of the "1" digits of the binary bit string.

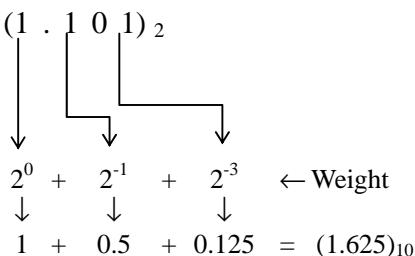
### a. Conversion of binary integers

**Example**  $(11011)_2$



### b. Conversion of binary fractions

**Example**  $(1.101)_2$



## ③ Conversion of binary numbers into hexadecimal numbers

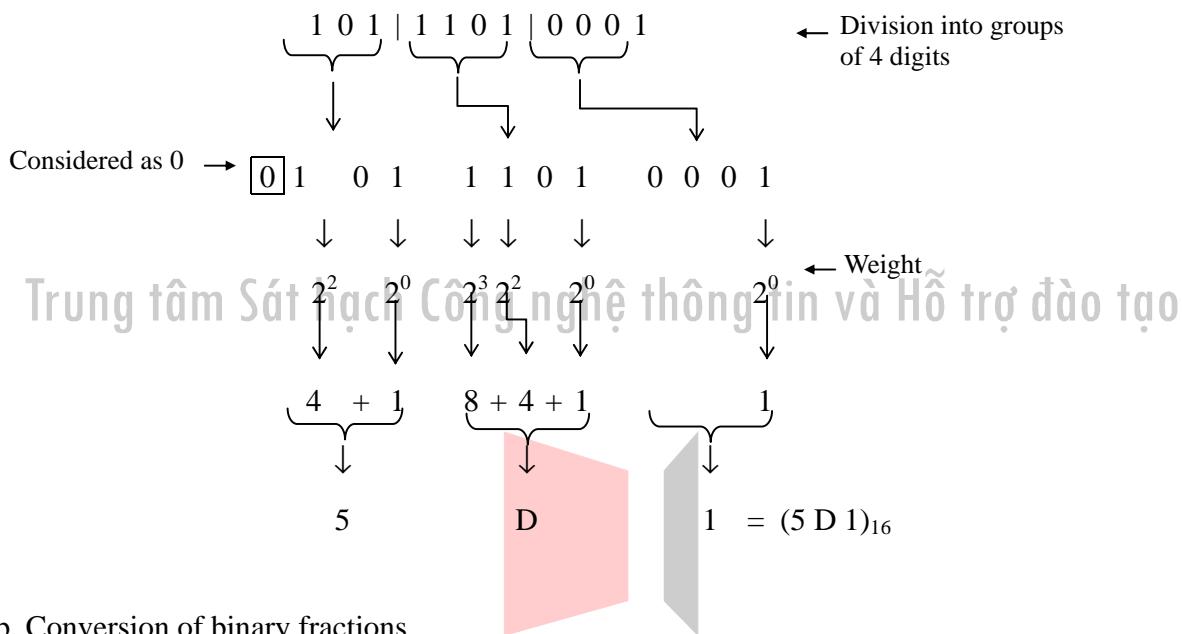
Since 4-bit binary strings are equivalent to 1 hexadecimal digit, in binary integers, the binary number is divided into groups of 4 digits starting from the least significant digit. In binary fractions, the binary number is divided into groups of 4 digits starting from the decimal point. Then, the conversion is performed by adding up the weights of each of the binary digits displayed as "1," in

each group of 4 bits.

In the event that there is a bit string with less than 4 digits, the necessary number of "0's" is added and the string is considered as a 4 bit string.

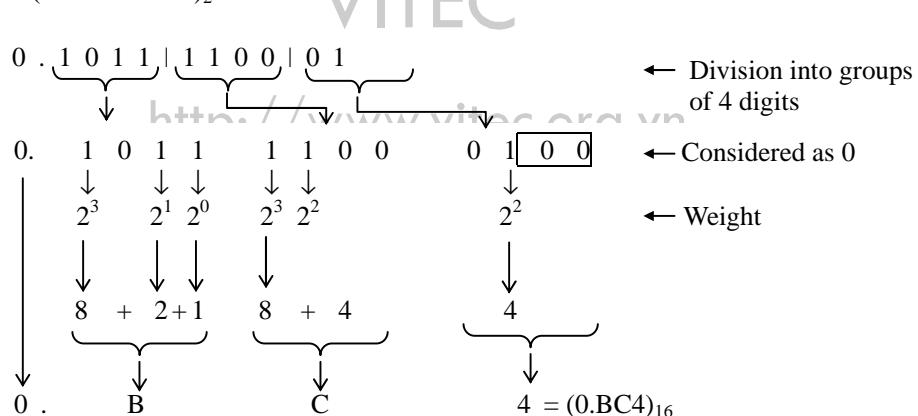
#### a. Conversion of binary integers

**Example**  $(10111010001)_2$



#### b. Conversion of binary fractions

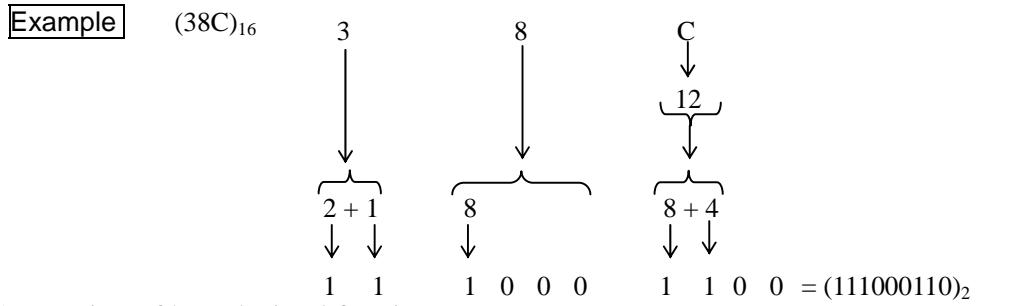
**Example**  $(0.101110001)_2$



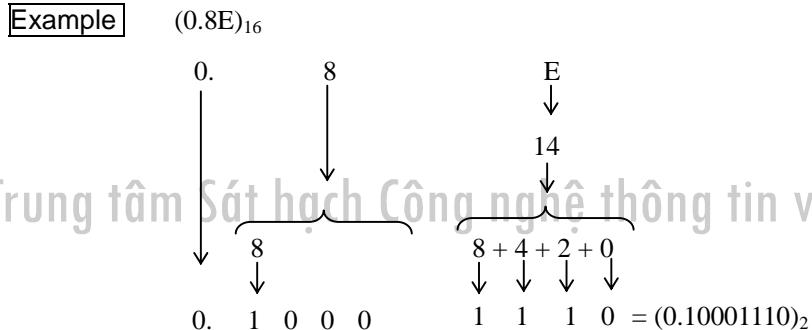
#### ④ Conversion of hexadecimal numbers into binary numbers

Hexadecimal numbers are converted into binary numbers by performing the reverse procedure. In other words, 1 digit of the hexadecimal number is represented with a 4 digit binary number.

a. Conversion of hexadecimal integers



b. Conversion of hexadecimal fractions

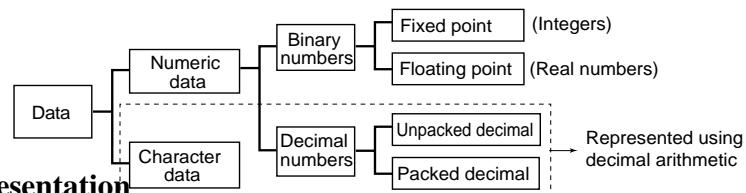


⑤Conversion from decimal numbers into hexadecimal numbers and from hexadecimal numbers into decimal numbers To convert them into binary numbers, decimal numbers are divided into 2, to convert them into hexadecimal numbers, and then they are divided into 16. Likewise, hexadecimal numbers are converted into decimal numbers by adding up the exponents whose radices are 16. It should be noted that due to the general unfamiliarity with the notation of hexadecimal numbers, ordinarily hexadecimal numbers are first converted into binary numbers to convert them into decimal numbers.

### 1.1.2 Numeric representation

In the computer, originally invented as a calculating machine, amongst other aspects involving the management of the data subject for processing, the precision and the easiness with which calculations can be performed have also been worked out. The representation format suitable for each type of data is explained here.

**Figure 1-1-8**  
Data representation format



### (1) Decimal digit representation

#### ① Binary coded decimal code

A format of character data and decimal numbers, there is a representation method called binary coded decimal code (BCD code) that, using 4 bit binary digits that correspond to the numbers 0 to 9 of the decimal system, represents the numeric value of each digit.

**Figure 1-1-9** Binary-coded decimal code

Decimal number	Binary number	Binary-coded decimal code
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	0 1 0 1
6	0 1 1 0	0 1 1 0
7	0 1 1 1	0 1 1 1
8	1 0 0 0	1 0 0 0
9	1 0 0 1	1 0 0 1
10	1 0 1 0	0 0 0 1 0 0 0 0
11	1 0 1 1	0 0 0 1 0 0 0 1
:	:	

VITEC

Since the binary-coded decimal code is not a numeric value but a code, there are only 10 patterns, and the notation is performed by arranging the patterns for each digit.

For example, the representation of the decimal number "789" using the binary coded decimal would be as follows:

↓            ↓            ↓  
 { 0 1 1 1 }   { 1 0 0 0 }   { 1 0 0 1 }      (011110001001)<sub>2</sub>

In this way, as the number of digits of a decimal number increases, the length of the binary coded decimal code increases as well (a group of 4 bits is added for each digit). This format is called variable length format.

The same value as the binary coded decimal code has also been set to the least significant 4 bits of the numeric characters of the EBCDIC, JIS II and other codes.

The binary coded decimal code is mainly used for the numeric representation of office calculations, and according to the memory format of the computer, it is divided into unpack decimal format and packed decimal format. And, since character codes as well as the unpacked decimal format or packed decimal format are represented by the use of the binary coded decimal code, they are automatically processed using the decimal arithmetic system of the computer. It is not necessary for the user to be aware of this process.

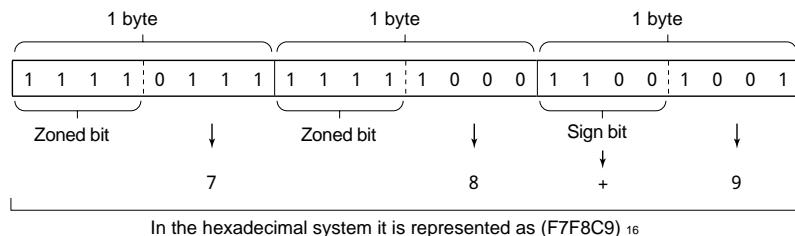
## ② Unpacked decimal code

When representing signed decimals, the unpacked decimal format uses 1 byte for each digit of the decimal number. The unpacked decimal format represents the values from 0 to 9 in the least significant 4 bits of 1 byte, and in the most significant 4 bits, which are called zoned bits, in the case of the EBCDIC code used in high end mainframe machines, where ordinarily  $(1111)_2$  is stored. However, in the zoned bits of the least significant digit, the 4 bits that represent the sign are stored, in both the case of 0 and positive numbers,  $(1100)_2$ , and in the case of negative numbers,  $(1101)_2$ . In the JIS code used for data transmission as well as in the low end machines,  $(0011)_2$  is stored in the zoned bits. The unpacked decimal format is also called zoned decimal format.

The bit pattern of the representation of the decimal numbers +789 and -789 in the unpacked decimal format is shown in Figure 1-1-10.

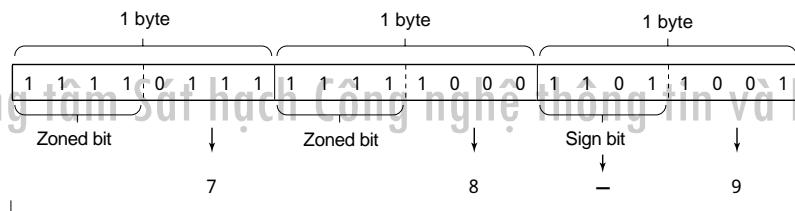
**Figure 1-1-10** Unpacked decimal format

<For +789>



In the hexadecimal system it is represented as  $(F7F8C9)_{16}$

<For -789>



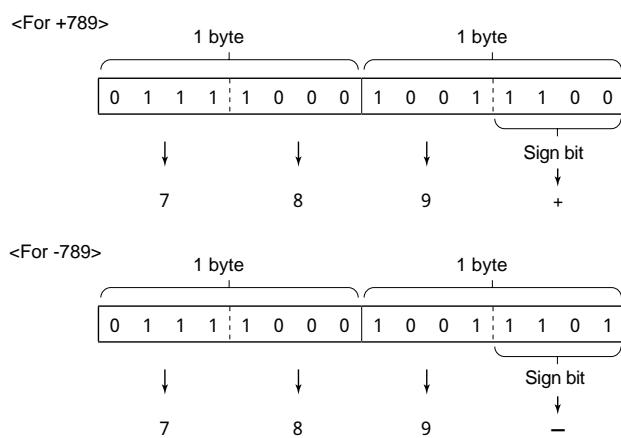
In the hexadecimal system it is represented as  $(F7F8D9)_{16}$

In the unpacked decimal format, excepting the least significant byte, only half of a byte is used. This was considered a waste of resources. This defect was eliminated by the packed decimal format.

### ③ Packed decimal code

In the packed decimal format, 1 byte represents a numeric value of 2 digits and the least significant 4 bits represent the sign. The bit pattern of the sign is the same as that of the unpacked decimal format,  $(1100)_2$ . Figure 1-1-11 shows the bit pattern of the packed decimal format.

**Figure 1-1-11**  
Packed decimal format



Compared to the unpacked decimal format, the packed decimal format has the following advantages:

- A numeric value can be represented by fewer bytes
- The conversion into the binary system is easy.

## (2) Binary representation

### ① Representation of negative integers

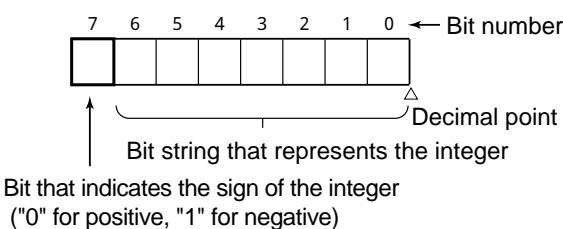
The typical examples of methods to represent negative integers are mentioned below:

- Absolute value representation
- Complement representation

#### a. Absolute value representation of negative integers

As is shown in Figure 1-1-12, in the absolute value representation of negative integers, the first bit represents the sign and the 7 other bits represent the numeric value (absolute).

**Figure 1-1-12**  
Absolute value representation of negative integers



For example, in  $(00001100)_2$ , since the sign bit at the top is 0, it is a positive number. Likewise,

since the 7 other bits, which are the absolute value of the numeric value, are  $(0001100)_2 = 2^2 + 2^3 = (12)_{10}$ , the decimal number 12 (positive number) is represented.

On the other hand, since in  $(10001100)_2$  the sign bit at the top is 1, it is a negative number. The decimal number -12 =(negative number) is represented. However, since in this representation method the numeric value 0 can be represented in two ways, as 00000000 (positive zero) or as 10000000 (negative zero), the operation becomes complicated, and for that reason it is not widely used.

It should be noted that when the absolute value representation of negative numbers is performed with 8 bits, the range of numeric values that can be represented is as follows (in decimal digits):

-127 to 127  
b. Complement representation of negative integers

The complement is the number that indicates the quantity by which a numeric value falls short of a specific numeric value. There are 2 types of radix complements, the radix complement and the reduced radix complement.

#### ● Decimal complement

There are two types of decimal complements, "10's complement" and "9's complement." For example, the 9's complement of a given numeric value will be the result of the subtraction of each of the digits of this numeric value from 9. Likewise, the 10's complement of a given numeric value will be the result of the subtraction of each of the digits of this numeric value from 10 As a result, the 10's complement is the result of the addition of 1 to the 9's complement.

**Example** "9's complement" of  $(123)_{10}$

$$\begin{array}{r} 999 \\ - 123 \\ \hline 876 \end{array}$$

**Example** "10's complement" of  $(123)_{10}$

$$\begin{array}{r} 1000 \\ - 123 \\ \hline 877 \end{array} \quad (=999+1)$$

## ● Binary complement

There are two types of binary complements, "1's complement" and "2's complement".

### ● 1's complement

The "1's complement" of a give numeric value will be the result of the subtraction of each of the digits of this numeric value from 1, as a result, all the "0" and "1" bits of the original bit strings are switched.

For example, the "1's complement" of the bit string  $(10110011)_2$  is shown below:

$$\begin{array}{r} 10110011 \\ \downarrow \leftarrow \text{All the "0" and "1" bits of the original bit string are switched} \\ 01001100 \leftarrow \text{"1's complement"} \end{array}$$

### ● 2's complement

"2's complement" is the "1's complement" bit string plus 1. Therefore, the "2's complement" of the bit string  $(10110011)_2$  is obtained as follows:

$$\begin{array}{r} 10110011 \\ \downarrow \leftarrow \text{All the "0" and "1" bits of the original bit string are switched} \\ 01001100 \leftarrow \text{"1's complement"} \\ + 1 \leftarrow 1 \text{ is added} \\ \hline 01001101 \leftarrow \text{"2's complement"} \end{array}$$

## ● "1's complement" and "2's complement" representation of negative integers

### ◆ "1's complement" representation of negative integers

◆ Sign bit: 0 for positive, 1 for negative, and both ,+0 and -0, for 0

◆ Numeric value: "1's complement"

For example, the "1's complement" representation of the decimal number -126 will be as follows:

$$\begin{array}{r} 01111110 \leftarrow +126 \\ \downarrow \leftarrow \text{Sign} \rightarrow \downarrow \leftarrow \text{All the "0" and "1" bits of the original bit string are switched} \\ 10000001 \leftarrow -126 \end{array}$$

### ◆ "2's complement" representation of negative integers

◆ Sign bit: 0 for positive and 0, 1 for negative

◆ Numeric value: "2's complement"

For example, the "2's complement" representation of the decimal number -126 will be as follows:

Sign →  $\begin{array}{r} 0 \\ \downarrow \\ 1 \end{array}$   $\begin{array}{r} 1111110 \\ \downarrow \\ 10000001 \end{array}$  ← All the "0" and "1" bits of the original bit string are switched  
 $\begin{array}{r} + \\ \hline 10000001 \end{array}$  ← 1 is added  
 $\hline 10000010$  ← -126

As can be observed, even for the same number the bit strings of the "1's complement" and the "2's complement" differ. Figure 1-1-13 shows a comparison of the range of numeric values which can be represented with 3 bits in the "1's complement" and the "2's complement" representation. From this figure it can be noted that the range of representable numeric values with the "2's complement" is wider. Likewise, as in the absolute value representation of negative integers, and in the representation of negative numbers using the "1's complement", 0 can be represented both, as +0 and as -0, so the operation becomes complicated. For that reason, a great number of today's computers have adopted the 2's complement method.

Figure 1-1-14 shows the range of representable numeric values when an n-bit binary number is represented with the "1's complement" and "2's complement".

**Figure 1-1-13**

"1's complement"  
and "2's complement"

"1's complement"				"2's complement"			
0	1	1	= 3	0	1	1	= 3
0	1	0	= 2	0	1	0	= 2
0	0	1	= 1	0	0	1	= 1
0	0	0	= 0	0	0	0	= 0
<hr/>				<hr/>			
1	1	1	= -0	1	1	1	= -1
1	1	0	= -1	1	1	0	= -2
1	0	1	= -2	1	0	1	= -3
1	0	0	= -3	1	0	0	= -4

**Figure 1-1-14**

Range of representable numeric values with "1's complement" and "2's complement"

- Range of representable numeric values when an n-bit binary number is represented using the "1's complement" method  
 $-(2^{n-1} - 1) \dots 2^{n-1} - 1$
  - Range of representable numeric values when an n-bit binary number is represented using the "2's complement" method  
 $-2^{n-1} \dots 2^{n-1} - 1$

Another important reason for the adoption of the 2's complement method is illustrated by the following example.

$$\begin{array}{r}
 01100100 \\
 + 10100110 \\
 \hline
 100001010
 \end{array}
 \quad (\text{Decimal digit } 10)$$

↑

Since the bit number is 8, the 9th digit resulting from the carry is ignored.

Therefore, the reason why negative numbers are represented using the 2's complement method in computing is that subtractions can be performed as additions. In other words, since subtractions can be performed with addition circuits, the subtraction circuits are unnecessary, simplifying the hardware structure.

## ② Fixed point

### a. Integer representation

The fixed point is a data representation format used mainly when integer type data is processed (Figure 1-1-15). Therefore, the fixed point format is also called an integer type.

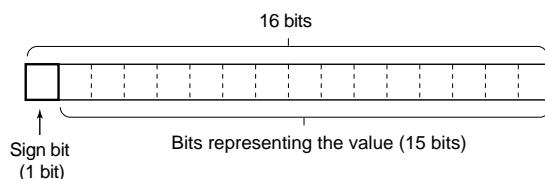
In the unpacked decimal format or packed decimal format, depending on the digit number of the decimal number, the number of bytes changes, but in the fixed point format one word is represented in a fixed length such as 16 bits and 32 bits.

For that reason, if there is an attempt to represent a numeric value that exceeds the fixed length a problem called overflow will occur.

**Figure 1-1-15**

Fixed point

<When 1 word is represented by 16 bits>



Since in the fixed point format in Figure 1-1-15, where a value is represented with 15 bits, if a negative number is represented using the 2's complement" representation, the range of

representable numeric values in the decimal system is as follows:

$$-2^{15} \text{ to } 2^{15} - 1 = -32,768 \text{ to } 32,767$$

Likewise, if one word is composed of n bits, and a negative number is represented using the "2's complement" representation, the range of representable numeric values in the decimal system is as follows:

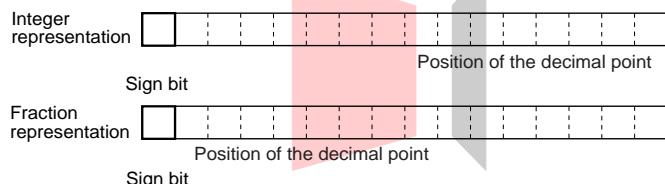
$$-2^{n-1} \text{ to } 2^{n-1} - 1$$

#### b. Fraction representation

When integers are represented, the position of the decimal point is considered to be on the right-hand side of the least significant bit.

When fractions are represented, the position of the decimal point is considered to be immediately preceded by the sign bit.

Figure 1-1-16 Representation format of integers and fractions



### ③ Floating point

While the fixed point format represents integer type data, the floating point format is used to represent real number type data. In ordinary mainframe computers, a maximum of 18 digit decimal numbers only can be represented. With 18 digits, there should be almost no problem in our daily life.

However, in a world where complicated calculations such as the ones mentioned below are required, correct results cannot be achieved with integer type data alone.

- Fluid mechanics calculations required for airplane design
- Calculations for weather forecasts
- Space flight planning and control

- Ballistic calculation
- CAD (Computer Aided Design)

For scientific and engineering fields requiring this kind of complicated calculation, the floating point format is used. Here, "complicated" means not only the calculation process itself is complicated, but also the either extremely large or small size of data is processed representation:

$15 \times 10^8$   
In the floating point format it would be written as  $0.15 \times 10^{10}$ .

$\underline{0.15} \times 10^{10}$   
↑

This part is represented as smaller than 1

The name of each of the numbers is shown below.

$0.15 \times 10^{10}$  ← Exponent  
↑ ↑

Mantissa Radix

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Here, to make it easier to understand, the decimal system is used, but in the computer, the binary system is used.

The floating point representation format varies depending on the computer. This is roughly classified into the format used in mainframe computers and that defined by the IEEE (Institute of Electrical and Electronics Engineering).

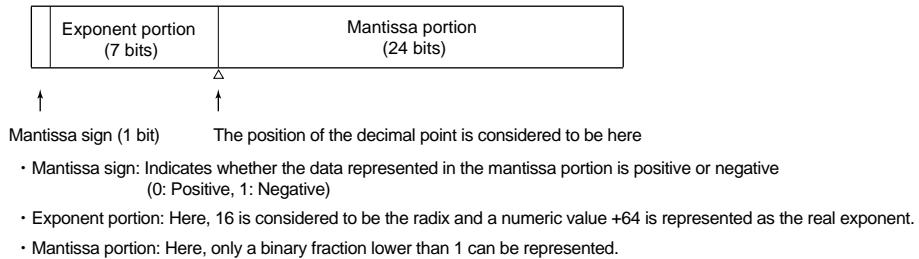
#### a. Floating point representation format in mainframe computers

The floating point representation format used in general purpose computers is shown in Figure 1-1-17.

<http://www.vitec.org.vn>

This format was adopted in the first general purpose computer in the world the "IBM System/360" and it was called Excess 64.

**Figure 1-1-17** Floating point representation format in general-purpose computers



### ● Exponent portion

The exponent portion has 7 bits, and the range of numeric values representable in the binary system is  $(0000000)_2$  to  $(1111111)_2$ , which in the decimal system is 0 to 127. However a numeric value 64 times larger than the real exponent is represented. For that reason, the real exponent is equivalent to -64 to +63.

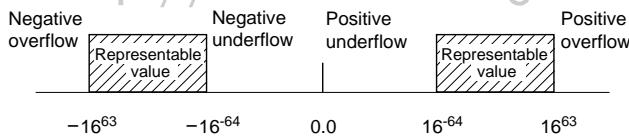
Likewise, since the radix is considered to be 16, the numeric values that can be represented with the exponent portion range between

$$16^{-64} \text{ to } 16^{63}$$

Then, including the sign bit, the range of numeric values that can be represented with the exponent portion is shown in Figure 1-1-18.

**Figure 1-1-18** Range of numeric values that can be represented with the exponent portion

<http://www.vitec.org.vn>

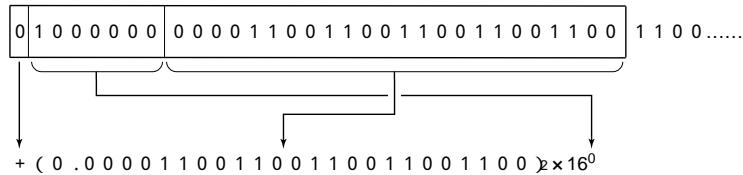


### ● Mantissa portion

When the decimal fraction 0.05 is converted into a binary fraction, it becomes a repeating binary fraction.  $(0.000011001100110011001100\dots)_2$

Figure 1-1-19 shows the representation of this fraction when the floating point format is used.

**Figure 1-1-19** Representation of the binary fraction ( $0.0000110011001100110011001100\dots$ )<sub>2</sub>



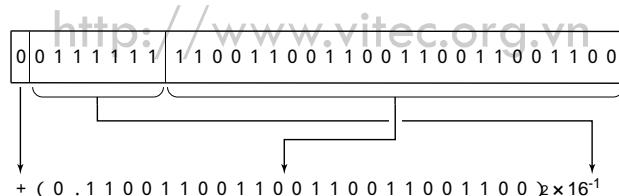
Since the mantissa portion has 24 bits, in this case, the decimal fraction 0.05 will not be represented correctly. (The error that occurs in this case is called a rounding error)

However, if we look at the bit pattern of the mantissa portion, we can see that the 4 top bits are 0, if we then extract these 4 bits and shift the remaining bits to the left, 4 rounded bits can be represented. Here, as a result of shifting the mantissa portion 4 bits to the left, the original value of the mantissa portion was increased by  $2^4 = 16$ . In order to cancel this increase it is necessary to divide it into 16 ( $\times 16^{-1}$ ). Since the radix is 16, the value of the exponent portion can be set to -1. The technique, used in this way, in order to reduce the rounding error to its minimum as well as to maximize precision is called normalization.

Furthermore, as a result of this normalization technique, the bit strings that represent a value are standardized. This operation is performed automatically by the hardware.

The result of the normalization of the binary fraction representation in Figure 1-1-19 is shown in Figure 1-1-20.

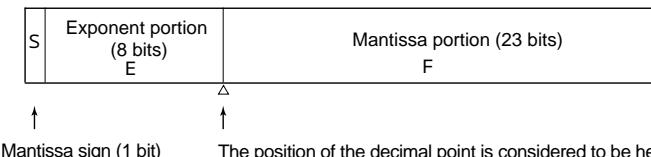
**Figure 1-1-20**  
Normalization



### b. IEEE Floating point representation format

The floating point representation format according to an IEEE standard is shown in Figure 1-1-21.

Figure 1-1-21 IEEE floating point representation format



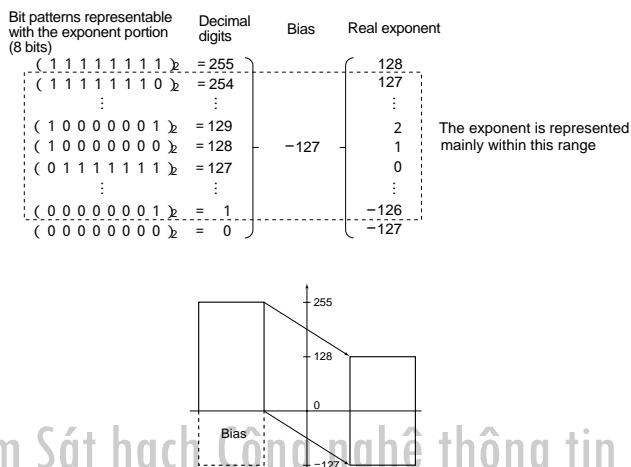
Mantissa sign (1 bit)      The position of the decimal point is considered to be here

- Mantissa sign: Indicates whether the data represented in the mantissa portion is positive or negative (0: Positive, 1: Negative)
- Exponent portion: 2 is considered to be the radix, and a value resulting from the addition of 127 to the value of the original exponent portion is represented.  
However, When: E = 255 , F = 0 : Not a number  
When: E = 255 , F = 0 : (-1)<sup>S</sup>  
When: E = 0 , F = 0 : (-1)<sup>S</sup> 2<sup>-126</sup> (0.F)  
When: E = 0 , F = 0 : (-1)<sup>S</sup> 0 (Zero)
- Mantissa portion: Here, only a binary fraction lower than 1 can be represented.
- Value represented using the floating point format:  $(-1)^S \times 2^E \times (1 + F)$

The differences from the general purpose computer floating point representation format are as follows:

- The exponent portion has 8 bits, and a value resulting from the addition of 127 to the high value of the original exponent portion is represented. This addition to the original value is called bias. (Figure 1-1-22).
- The mantissa portion has 23 bits and a binary fraction equivalent to the mantissa -1 is registered. In other words, 1 is considered to be omitted.
- The radix of the exponent portion is 2.

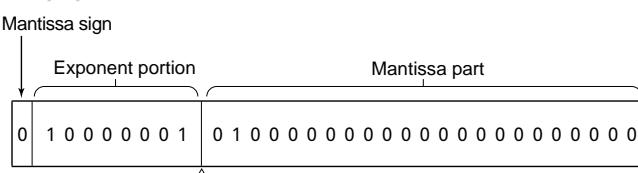
**Figure 1-1-22** Representation of the exponent portion



For example, if the decimal number 5 is represented using this format, it will be represented as in Figure 1-1-23.

- Since it is positive, the mantissa sign will be 0.
- If  $(5)_{10} = (101)_2 = (101)_2 \times 2^2$ , then, the exponent portion will be  $(1.01)_2 - (1)_2 = (0.01)_2$
- If  $(101)_2$  is shifted 2 bits to the right, it becomes  $(1.01)_2$ ,  $2^{-2}$  times the former value. In order to normalize it, 2 is added to the exponent 0, which becomes 2. As a result, since the exponent portion is  $2 + 127 = 129$ , the representation will be  $(10000001)_2$ .

**Figure 1-1-23**  
Representation of  
the decimal number 5



### c. Shift operation

The mechanism of the shift operation performed at the normalization is explained below.

In the binary system, each digit has a weight which is a power of 2. This is called positional weight.

Therefore, even though the same number 1 is represented, its meaning is different to that of the 1 positioned in the second digit and the 1 positioned in the third digit.

$$\begin{array}{l} \text{1 positioned in the second digit } (10)_2 \rightarrow 2^1 = 2 \\ \text{1 positioned in the third digit } (100)_2 \rightarrow 2^2 = 4 \end{array}$$

In the shift operation, by moving the position of 1 to the left (or to the right), the multiplication and division of numeric values can be easily performed.

The conversion into decimal numbers  $(100101)_2$  and  $(1001010)_2$ , resulting from shifting the first value 1 bit to the left, would be as follows:

Weight of each digit:       $2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

$(100101)_2$	$\rightarrow$	
$(1001010)_2$	$\rightarrow$	$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 = 64 + 8 + 2 = (74)_{10}$

Through this conversion, it should be noted that the 1 that represented  $2^5$  before shifting, now represents  $2^6$ , the 1 that represented  $2^2$  now represents  $2^3$ , and the 1 that represented  $2^0$  now represents  $2^1$ . In other words, after the shift the value of each of the 1s was doubled, and the result of the conversion into decimal system was also doubled from  $(37)_{10}$ , to  $(74)_{10}$ .

In short, the above mentioned result shows that by shifting a binary digit 1 to the left, its value is doubled. Following this approach, the shift operation can be summarized by the following rules:

- When a binary number is shifted n bits to the left, its former value is increased  $2^n$  times.
- When a binary number is shifted n bits to the right, its former value is decreased  $2^{-n}$  times. (The former value is divided by  $2^n$ )

### ◆ Arithmetic shift

The arithmetic shift is the shift operation used to calculate numeric values. It is used in the fixed point format that represents negative numbers using the "2's complement" representation.

#### [Arithmetic shift rules]

- The sign bit is not shifted
- The bit shifted out is lost.

- The bit to be filled into the bit position vacated as a result of the shift is:  
For left shifts: 0  
For right shifts: Same as the sign bit

**Example** Calculation of  $(22)_{10} \times 4$  using the arithmetic shift

- Represent  $(22)_{10}$  using the fixed point format (8 bits)

$$(22)_{10} = (10110)_2$$



0	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

- Shift 2 bits to the left to increase it by 4 ( $= 2^2$ ).

0	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

0	1	0	1	1	0	0	0
---	---	---	---	---	---	---	---

0 0

0s are put into the vacated bit positions

The bits shifted out are lost

### ● Logical shift

The logical shift is the shift operation used in change the bit position. The big difference from the arithmetic shift is that the sign bit is not treated differently.

{Logical shift rules]

- The sign bit is also shifted
- The bit shifted out is lost
- The bit to be filled into the bit position vacated as a result of the shift is 0.

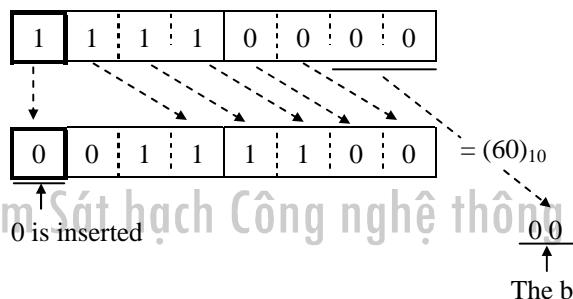
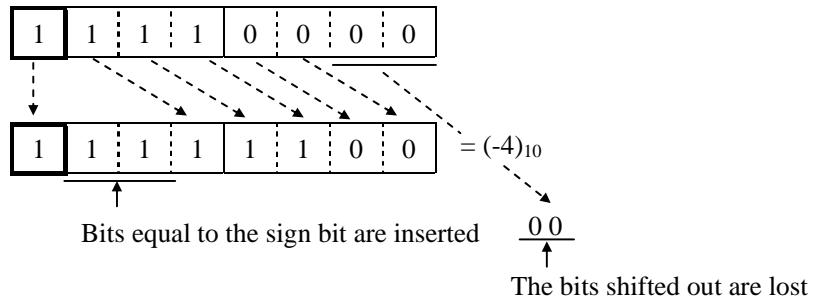
**Example**

After arithmetically and logically shifting  $(-16)_{10}$  2 bits to the right, convert each of the results into decimal digits.

- Represent  $(-16)_{10}$  using the fixed point format (8 bits).

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

② Arithmetically shift 2 bits to the right



### 1.1.3 Operation and precision

Since the storage capacity of a computer is limited, not all the numeric values we use can be represented correctly. In other words, a value represented in a computer is an approximate representation. As was mentioned above, in commercial data processing, operations are performed using decimal representations, while both the internal representation as well as the operation of scientific and engineering calculations, are performed using the binary representation. For that reason, the difference between the numeric value represented internally and the true value becomes a problem.

#### (1) Precision of the numeric value representation

The precision of a number is the range of its error, and so "high precision" means "small error". Focusing only on the integer part, if enough digits are available, no error occurs. However, the fraction part is not so simple; since many decimal fractions cannot be completely represented with the binary fractions containing a finite number of digits.

## ① Single precision

In scientific and engineering calculations, numerical values are represented by binary digits in word units. The word length depends on the hardware. For example, when 1 word = 16 bits, in general terms, The format in which 1 numeric value is represented with one word .is called single precision. The range of numeric value representable with 16bits, in case of an integer without a sign , is indicated below.

$$\text{Minimum value} = (0000\ 0000\ 0000\ 0000)_2 = 0$$

$$\text{Maximum value} = (1111\ 1111\ 1111\ 1111)_2 = 65,535$$

In other words, values higher than 65,535 cannot be represented.

Likewise, the range of numeric values representable with 16 bits , in the case of a fraction without a sign is indicated below.

$$\text{Minimum value} = (0000\ 0000\ 0000\ 0000)_2 = 2^{-16} \equiv 0.0000152587890625000$$

$$\text{Maximum value} = (1111\ 1111\ 1111\ 1111)_2 = 1 - 2^{-16} \equiv 0.9999847412109370000$$

In this case, values lower than 0.0000152587890625, and values higher than 0.999984741210937 can't be represented.

## ② Double precision

In order to widen the range of representable numeric values, the number of digits is increased. By representing 1 numeric value with 2 words, in comparison to the representation with 1 word, the range of representable numeric values becomes much wider. This format is called double precision.

If 1 word = 16 bits, 1 numeric value is represented with twice as many bits, 32 bits.

$$\text{Minimum value} = (0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000)_2 = 0$$

$$\text{Maximum value} = (1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111)_2 = 4,294,967,295$$

In other words, values up to 4,294,967,295 can be represented.

Likewise, the range of numeric values representable with 32 bits , in the case of a fraction without a sign is indicated below.

Minimum value = ( 0000 0000 0000 0000 0000 0000 0000 0000 )<sub>2</sub> =  $2^{-32} \equiv$   
0.0000000023283064365387

Maximum value = ( 1111 1111 1111 1111 1111 1111 1111 1111 )<sub>2</sub> =  $1 - 2^{-32} \equiv$   
0.9999999976716900000000f

## (2) Operation precision

### ① Precision of fixed point representation

The range of representable numeric values with the fixed point representation depends on the computer hardware. Depending on the number of bits in a one word, the range of representable numeric values differs. The step size of the integer part is always 1, regardless of the number of bits, and only the maximum value changes. However, in the step size of the fraction part the larger the number of bits assigned, the smaller the step size becomes and the error is also reduced.

### ② Precision of floating point representation

#### a. Overflow and underflow

When multiplication of extremely large values or extremely small values is performed, there are cases where the operation results exceed the range of numeric values that can be represented with the exponent portion. The condition that occurs when the product is higher is than the maximum value that can be represented with the exponent portion, is called overflow (Maximum absolute value < Overflow). The condition that occurs when the product is lower is than the minimum absolute value is called underflow (0 < Underflow < Minimum absolute value).

#### b. Cancellation

When subtraction of two floating point numbers of almost equal values is performed, since the result becomes extremely small, it is left out of the range of numeric values which can be represented. This condition is called cancellation.

#### c. Loss of information

When two values represented by using the floating point format are added, the exponents must coincide. Generally, exponents are adjusted to the largest value.

. When an addition of an extremely small value and an extremely large value is performed, since exponents must be adjusted to the exponent of the largest value, the mantissa portion of the small value is shifted largely to the right. As a consequence of this shift, the information that should have been represented is lost. This condition is called loss of information.

In order to avoid this kind of error, it is necessary to think out strategies such as changing the order of the operations, etc. These strategies are worked out by the user.

#### **1.1.4 Non-numeric value representation**

When using a computer, in order to input numerals and characters (alphabetical characters, symbols, etc.) Input devices such as keyboards, are used. Inside the computer, in order to represent characters using binary digits a concept called code is used.  
Presently, different character codes are used depending on the computer. Here, the codes widely used around the world and in Japan will be explained.

##### **(1) Character representation**

The character codes widely used worldwide basically represent 1 character with 8 bits, that is, 1 byte. The character codes used in the information processing field are sometimes called codes for information interchange.

By typing on a keyboard, these character codes are input in the computer as 1 byte codes.

The following keys are found in the keyboard of a personal computer, etc.

- Numeric keys: 10 types
- Character keys: Alphabet: (Uppercase: A to Z and lowercase: a to z) 52 types
- Symbolic types: 40 types
- Control character keys: 34 types (Space key, etc.)

To assign the unique bit pattern corresponding to these 136 types of characters and symbols, 256 types of bit patterns that can be represented with 8 bits are required.

##### **(2) Character codes**

The main character codes are listed below.

## ① ASCII (American Standard Code for Information Interchange) code

The ASCII code was established by the U.S. standards institution, ANSI (American National Standards Institute) in 1962. Character code of 8 bits composed by the code bit representing the alphabet, numeric characters, etc. (7 bits) and the parity bit used to detect errors. It is used in personal computers and in data transmission.

## ② ISO (International Organization for Standardization) code

The ISO code is a 7 bit character code that was established by the International Organization for Standardization (ISO) in 1967 based on the ASCII code. It is the base of the character codes used in all countries of the world.

## ③ JIS (Japan Industrial Standards) code

The JIS code was established as JIS X 0201 by adding the Romaji, hiragana and other characters peculiar to the Japanese language to the ISO code. The "JIS 7 bit code" used to represent Romaji and the "JIS 8-bit code" used to represent katakana as well as the "JIS kanji code," that represents 1 character with 2 bytes (16 bits), and is used to represent hiragana and kanji, exists.

## ④ EBCDIC (Extended Binary Coded Decimal Interchange Code)

The EBCDIC is a character code developed by IBM. Compared to the above mentioned character codes, which were established to be used as standards, the EBCDIC code was developed for IBM computers. Since IBM held the greatest share of the computer market when the third generation computers, in which this code was developed, were launched, other companies developed their computers according to this character code, and as a result it became a standard character code. Standards like this, resulting from the existence of a large number of users, are called *de facto* standards.

## ⑤ Shift JIS code

As was mentioned above, to represent kanji, the JIS kanji code represents 1 word with 2 bytes.

Figure 1-1-24 JIS X 0201 code table

The table shows the JIS X 0201 code table with bit numbers (b8 to b1) on the left. It is divided into 'Low-order bits' and 'High-order bits'. The High-order bits section contains 16 rows of binary values from 0 to F. The Low-order bits section contains 16 rows of binary values from 0 to F, followed by their corresponding control codes (NUL, TCx, DCx, etc.) and their meanings in English and Japanese. A vertical bracket on the right indicates the 'National character code portion' for each row.

b8	b7	b6	b5	b4	b3	b2	b1	RDW	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0		0	0	1	1	1	1	1	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1		1	1	0	0	1	1	1	0	0	1	1	0	0	1	1
0	0	0	0	0	1	0	1		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	0	0	0	1	1	2	3		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	NUL	TC7(DLE)	SP	0	@	P	`	p	:	:	—	タ	ミ	：	：	：	：	：
0	0	0	0	1	1	TC1(SOH)	DC1	!	1	A	Q	a	q	:	:	。	ア	チ	ム	：	：	：	：
0	0	1	0	2	2	TC2(STX)	DC2	2	B	R	b	r	:	:	「	イ	ツ	メ	：	：	：	：	
0	0	1	1	3	3	TC3(ETX)	DC3	#	3	C	S	c	s	:	:	」	ウ	テ	モ	：	：	：	：
0	1	0	0	4	4	TC4(EOT)	DC4	\$	4	D	T	d	t	:	:	、	エ	ト	ヤ	：	：	：	：
0	1	0	1	5	5	TC5(ENQ)	TC8(NAK)	%	5	E	U	e	u	:	:	・	オ	ナ	ユ	：	：	：	：
0	1	1	0	6	6	TC6(ACK)	TC9(SYN)	&	6	F	V	f	v	:	:	ヲ	カ	ニ	ヨ	：	：	：	：
0	1	1	1	7	7	BEL	TC10(ETB)	'	7	G	W	g	w	:	:	ア	キ	ヌ	ラ	：	：	：	：
1	0	0	0	8	8	FE0(BS)	CAN	(	8	H	X	h	x	:	:	イ	ク	ネ	リ	：	：	：	：
1	0	0	1	9	9	FE1(HT)	EM	)	9	I	Y	i	y	:	:	ウ	ケ	ノ	ル	：	：	：	：
1	0	1	0	10	10	FE2(LF)	SUB	*	:	J	Z	j	z	:	:	エ	コ	ハ	レ	：	：	：	：
1	0	1	1	11	11	FE3(VT)	ESC	+	:	K	[	k	{	:	:	オ	サ	ヒ	ロ	：	：	：	：
1	1	0	0	12	12	FE4(FF)	IS4(FS)	,	<	L	¥	l	]	:	:	ヤ	シ	フ	ワ	：	：	：	：
1	1	0	1	13	13	FE5(CR)	IS5(GS)	-	=	M	]	m	}	:	:	ユ	ス	ヘ	ン	：	：	：	：
1	1	1	0	14	14	SO	IS2(RS)	.	>	N	^	n		:	:	ヨ	セ	ホ	：	：	：	：	：
1	1	1	1	15	15	SI	IS1(US)	/	?	O	_	o	DEL	:	:	ツ	ソ	マ	。	：	：	：	：

There are times when the JIS code and the JIS kanji code are mixed. When 1-byte codes and 2-byte codes are mixed, their interpretation can't be performed. Therefore, a special escape code is added to the front and back of the kanji code string. For example, a bit string of 5 kanji becomes 10 bytes + 2 bytes. When data is missed during data transmission, etc. recovery is difficult.

In order to find a solution to this defect, the shift JIS code that converts the characters defined in the JIS Kanji code into another code system was created. The first byte of the shift JIS uses a code that is not used in the JIS (1 byte) code, and, at the same time, by avoiding control character codes in the second character, 1 byte codes and 2 byte codes can be mixed without using special escape codes.

## ⑦ Unicode

The Unicode is a 2 byte code system unified to all the countries, which was proposed and designed by Apple Computer, IBM, Microsoft, and other U.S. companies in order to smooth the exchange of data amongst personal computers. This code was adopted by ISO as an international standard draft.

### **(3) Audio representation**

As has been said, the current information technology provides multimedia support, and the data subject to processing is not limited to character data or numerical data. It also covers many kinds of information that are used in our daily life. One of the components which compose multimedia is audio.

The human voice is produced when the airflow generated in the lungs changes, vibrates and resonates due to a great number of organs such as the tongue, lips, teeth, jaw, nasal cavity and vocal cords. Since audio data has a complicated analog waveform, audio analysis is performed using a numeric formula and once it is converted into digital codes it is processed in the computer. Word processors that accept audio input and speaker recognition are examples of its recent applications.

### **(4) Image representation**

In order to support current multimedia, not only audio but also image data must be processed.

Inside the computer, image data is processed as a set of dots. For that reason, the registration of the status of each of the dots that compose an image, is the registration of the image data itself. the easiest approach is to register two states, black and white, for each of the dots that compose an image. In this case, 1 bit is used to register the information of each dot. Today most of the image data is colored, so this method does not solve all problems. Therefore, the representation method that combines the basic colors in each dot is used. Amongst computer screens, there are a great number of systems that combine the three primary colors (Red, green and blue) in 256 levels respectively and represent approximately 16,000,000 colors. In this case, since 8 bits are needed for 1 color, in order to register the information of 1 dot, 24 bits is used.

## 1.2 Information and logic

---

### 1.2.1 Proposition logic

Operations which can be processed in a computer are not limited to arithmetic formulas. By assigning a value to a sentence, sentence operations can be performed. For example, in logical mathematics, the sentences represented as "The wind is blowing," "It is raining" "x=5" and "y=2" are called propositions.

Values of "truth" or "lie" in other words, "true" and "false" can be assigned to these propositions.

However, one proposition will always be either "true" or "false." The same proposition can't be "true" at the same time it is "false." "The wind is blowing, and it is raining" is possible, but "The wind is blowing, there is no wind" is impossible.

These propositions are represented by p,q,r,... and other letters, and though the combination of their logical significance new synthetic propositions can be created. Each proposition relation is made clear, through logical operation by proposition logic. Whether a symbolic proposition is true or false is determined by the truth table. An example of this table is shown in Figure 1-2-1.

This truth tables shows:



- The proposition "The wind is not blowing" is false when the proposition 1, "The wind is blowing" is true, it is true when the proposition 1 is false.
- The proposition "The wind is blowing or it is raining" is true when both, the proposition 1, "The wind is blowing" and the proposition 2, "It is raining," are true or when either of the two is true. When both of them are false, it is false.

**Figure 1-2-1** Truth table

Proposition 1 The wind is blowing	Proposition 2 It is raining	The wind is not blowing	The wind is blowing and it is raining	The wind is blowing or it is raining	If the wind blows it rains
True Wind	True Rain	False Lie	True Truth	True Truth	True Truth
True Wind	False No rain	False Lie	False Lie	True Truth	False Lie
False No wind	True Rain	True Truth	False Lie	True Truth	False Lie
False No wind	False No rain	True Truth	False Lie	False Lie	False Lie

## 1.2.2 Logical operation

Since the expression of the logical significance with words becomes lengthy and it is not suitable for computer operations, logical relations are represented with symbols. The symbols that represent these propositional operations (or logical operations) are called logical symbols or logical connectors. The main logical symbols used in information processing are NOT, AND, OR, exclusive OR, etc. Their meanings are explained below.

Each logical operation will be explained using the examples shown in Figure 1-2-1 and proposition 1: "The wind is blowing," which will be denoted as p, and proposition 2 "It is raining" as q.

### (1) Negation

By negating the proposition "The wind is blowing," a new proposition, "The wind is not blowing," can be created. In this case, the logical symbol " $\neg$  (NOT)" is used and it is represented as " $\neg p$ ".

**Figure 1-2-2**

Truth table for negation

p	$\neg p$
T	F
F	T

T(rue) : True  
F(ales) : False

### (2) Logical product

When two propositions are connected with the conjunction "AND" as in "The wind is blowing and it is raining," both "The wind is blowing" and "It is raining" are expressed simultaneously.

The connection of the two propositions p and q with the conjunction "AND" is called logical product. In this case, the logical symbol " $\wedge$  (AND)" is used and it is represented as " $p \wedge q$ ".

The truth table is shown in Figure 1-2-3, and the result is true only when p and q are both true.

Figure 1-2-3

Truth table for  
the logical product

p	q	p $\wedge$ q
T	T	T
T	F	F
F	T	F
F	F	F

### (3) Logical sum

When two propositions are connected with the conjunction "OR" as in "The wind is blowing or it is raining," either "The wind is blowing" or "It is raining" is expressed.

The connection of the two propositions p and q with the conjunction "OR" is called logical sum. In this case, the logical symbol "v (OR)" is used and it is represented as "pvq." The result is true only when p and q are both true.

Figure 1-2-4

Truth table for  
the logical sum

p	q	p $\vee$ q
T	T	T
T	F	T
F	T	T
F	F	F

### (4) Exclusive OR

In the logical sum mentioned above, "The wind is blowing or it is raining," either "The wind is blowing" or "It is raining" is expressed. This logical sum is true when "The wind is blowing and it is also raining," or in other words, when both propositions are true. Ordinarily, the word "or" is used in many cases to express exclusive meanings as "either of the two". The exclusive OR is used to support these cases.

<http://www.vitec.org.vn>

In the case of the exclusive OR, the logical symbol " $\vee\!\!v$  (EOR)" is used and it is represented as "pvq." The result is true only when p or q, either of the two, is true. Therefore, the result is false when p and q are both true or false. This logical operation is frequently used in programming.

Figure 1-2-5

Truth table for  
the exclusive OR

p	q	p $\vee\!\!v$ q
T	T	F
T	F	T
F	T	T
F	F	F

### (5) Negative AND (NAND)

It is the negation of the above mentioned logical product. It is represented as " $\neg(p \wedge q)$ ." This

logical operation is frequently used in the design of logical circuits.

### (6) Negative Logical sum (NOR)

It is the negation of the above mentioned logical sum. It is represented as " $\neg(p \vee q)$ ".

Figure 1-2-6 puts the six logical operations mentioned above together.

**Figure 1-2-6** Truth table for the logical operations NOT, AND, OR, EOR, NAND and NOR (Summary)

p	q	NOT p	p AND q	p OR q	p EOR q	p NAND q	p NOR q
T	T	F	T	T	F	F	F
T	F	F	F	T	T	T	F
F	T	T	F	T	T	T	F
F	F	T	F	F	F	T	T

### (7) Logical expression laws

Logical symbols	Meaning	Symbols	Notation example
Negation	NOT	$\neg$	$\neg X$
Logical product	AND	$\cdot$	$X \cdot Y$
Logical sum	OR	$+$	$X + Y$
Exclusive OR	EOR	$\overline{+}$	$X \overline{+} Y$

$$\begin{aligned}
 X \oplus Y &= (\bar{X} \cdot Y) + (X \cdot \bar{Y}) \\
 &= ((\bar{X} \cdot Y) + X) \cdot ((\bar{X} \cdot Y) + \bar{Y}) \quad \dots\dots\dots \text{Distribution law} \\
 &= ((X + \bar{X}) \cdot (X + Y)) \cdot ((\bar{Y} + \bar{X}) \cdot (\bar{Y} + Y)) \quad \dots\dots\dots \text{Switching law and distribution law} \\
 &= (1 \cdot (X + Y)) \cdot ((\bar{Y} + \bar{X}) \cdot 1) \quad \dots\dots\dots \text{Logical sum law} \\
 &= (X + Y) \cdot (\bar{X} + \bar{Y}) \quad \dots\dots\dots \text{Logical product law}
 \end{aligned}$$

- Logical product law:  $X \cdot X = X, X \cdot \bar{X} = 0, X \cdot 0 = 0, X \cdot 1 = X$
- Logical sum law:  $X + X = X, X + \bar{X} = 1, X + 0 = X, X + 1 = 1$
- Exclusive OR law:  $X \oplus X = 0, X \oplus \bar{X} = 1, X \oplus 0 = X, X \oplus 1 = \bar{X}$
- Commutative law:  $X + Y = Y + X, X \cdot Y = Y \cdot X$
- Associative law:  $X + (Y + Z) = (X + Y) + Z, X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
- Distributive law:  $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$   
 $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$
- Absorptive law:  $X + (X \cdot Y) = X, X \cdot (X + Y) = X$
- Restoring law:  $\overline{\overline{X}} = X$
- De Morgan's law:  $\overline{X + Y} = \bar{X} \cdot \bar{Y}, \overline{X \cdot Y} = \bar{X} + \bar{Y}$

---

## Exercises for No.1 Chapter 1 (Basic Theories of Information)

**Q1** Which of the following represents correctly the size relation amongst the following 4 prefix symbols that represent integer exponents of 10: G (giga), k (kilo), M (mega) and T (tera)?

- A.  $G < k < M < T$       B.  $k < G < T < M$       C.  $k < M < G < T$   
D.  $M < G < k < T$       E.  $M < T < G < k$

**Q2** Which of these values corresponds to 1 picosecond?

- A. 1 nanosecond  $\times 1,000$   
B. 1 microsecond / 1,000,000  
C.  $2^{-12}$  seconds  
D.  $10^{-10}$  seconds  
E.  $10^{-11}$  seconds

**Q3** Given the binary digits A and B, where

$$A = 01010101 \quad B = 01100110$$

which of these values corresponds to the result of the operation  $A + B$ ?

- A. 01101010      B. 01111010      C. 10011010  
D. 10111011      E. 11010101

**Q4** Which of these values is the correct result of the subtraction of the hexadecimal numbers DD and 1F "00-1F"?

- A. AF      B. BE      C. CE      D. EC      E. FC

**Q5** Which of these values represents in decimal numbers the result of the addition of the binary numbers 1.1011 and 1.1101?

- A. 3.1      B. 3.375      C. 3.5      D. 3.8      E. 3.9375

**Q6** Which is the decimal number that can be represented without error in the binary floating point representation?

- A. 0.2      B. 0.3      C. 0.4      D. 0.5

**Q7** Which of these values represent the hexadecimal fraction 0.248 in decimal fractions?

- A.  $\frac{31}{32}$       B.  $\frac{31}{125}$       C.  $\frac{31}{512}$       D.  $\frac{73}{512}$

**Q8** Which is the correct bit pattern of the decimal number +432 when it is represented in the packed decimal format? Note that the sign is represented by the last 4 bits, and that "1100" represents positive numbers while "1101" represents negative numbers.

- A. 0000 0001 1011 0000  
B. 0000 0001 1011 1100  
C. 0001 1011 0000 1100  
D. 0100 0011 0010 1100  
E. 0100 0011 0010 1101

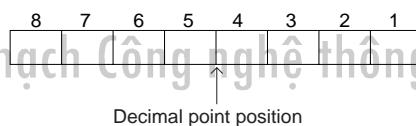
**Q9** What is the range of the integer representable with n bits using the fixed point format that represents a negative number with 2's complement representation? Here, the decimal point position is on the right side of the least significant bit (LSB).

- A.  $-2^n$  to  $2^{n-1}$
- B.  $-2^{n-1}$  to  $2^{n-1}$
- C.  $-2^{n-1}$  to  $2^{n-1} - 1$
- D.  $-2^{n-1} - 1$  to  $2^{n-1}$

**Q10** Which of the following is the reason why in many computers the complement representation is used to simplify the operation circuit?

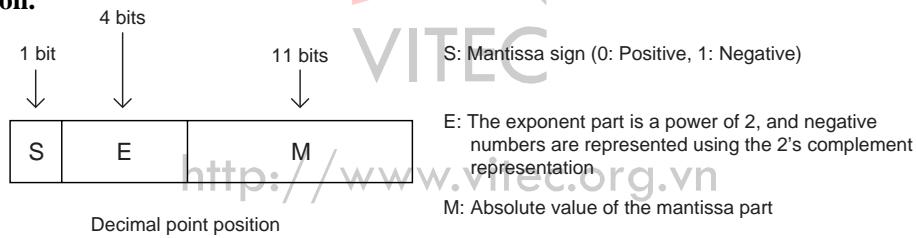
- A. Additions can be processed as subtractions.
- B. Subtractions can be processed as additions.
- C. Multiplications can be processed by the combination of additions.
- D. Divisions can be processed by the combination of subtraction.

**Q11** Which of these values corresponds to the representation of the decimal number -5.625 in binary number using the 8-bit fixed point format? Here, the position of the decimal point is between the fourth and fifth bits, and negative numbers are represented using the 2's complement representation.



- A. 01001100
- B. 10100101
- C. 10100110
- D. 11010011

**Q12** Which is the normalized representation of the decimal number 0.375? Here, the numeric value is represented using the 16-bit floating point format and the format is indicated in the figure. The normalization performed is an operation that adjusts the exponent portion in order to eliminate the 0s of higher digit rather than the significant values of the mantissa portion.



- |   |                                       |
|---|---------------------------------------|
| A | 0   0 0 0 0   0 1 1 0 0 0 0 0 0 0 0 0 |
| B | 0   1 0 0 0   0 0 0 0 0 0 0 0 0 0 1 1 |
| C | 0   1 1 1 1   1 1 0 0 0 0 0 0 0 0 0 0 |
| D | 1   0 0 0 1   1 1 0 0 0 0 0 0 0 0 0 0 |

**Q13** Which is the value that corresponds to the result of logically shifting the hexadecimal number ABCD two bits to the right?

- A. 2AF3
- B. 6AF3
- C. AF34
- D. EAF3

**Q14** The multiplication of binary number can be performed through the shift operation (digit shift), and addition. To increase the binary digit m by  $2^n$  it is necessary to shift m n bits to the left.

For example,  $m \times 19$  can be obtained through the following operation:

(Value of the result of shifting m one bit to the left) + (Value of the result of shifting m one bit to the left) + m  
Which is the value of a?

- A. 2      B. 3      C. 4      D. 5

**Q15** The decimal number -100 is registered using the 2's complement representation in a 8-bit register. Which of these values represent, in decimal numbers, the result of arithmetically shifting this registration three bits to the right?

- A. -33      B. -13      C. -12      D. 19

**Q16** Which of the following descriptions of the rounding error is correct?

- A. It is an error generated when an operation result exceeds the maximum value that can be processed by the computer.  
B. Due to the limited number of digits in the number representation, it is an error generated as a result of rounding off, rounding up, or omitting portions smaller than the least significant digit.  
C. It is an error generated due to the loss of the most significant values in the subtraction operation of numeric values, whose absolute values are almost equal.  
D. It is an error generated due to the loss of the least significant values of the mantissa portion of the numeric value, with the lower exponent value in the subtraction operation of floating point numbers.

**Q17** What is the minimum number of digits required for representing uniquely with the same number of bits of the uppercase alphabetic characters (A to Z) and the numeric characters (0 to 9)?

- A. 5      B. 6      C. 7      D. 8

**Q18** The following truth table shows the operation results of the logical operation "x  $\star$  y." Which of these expressions is equivalent to this operation?

Truth table

x	y	x $\star$ y
True	True	False
True	False	False
False	True	True
False	False	False

- A. x AND (NOT y)      B. x OR (NOT y)  
C. (NOT x) AND y      D. (NOT x) AND (NOT y)  
E. (NOT x) OR (NOT y)

**Q19** Which of the following expressions is equivalent to the logical expression  $\overline{(A + B) \cdot C}$ ? Here, " $\cdot$ " represents the logical product (AND), "+" the logical sum (OR), and  $\overline{A}$  is the negation of A (NOT).

- A.  $(A \cdot B) + \overline{C}$       B.  $A \cdot B \cdot \overline{C}$   
C.  $\overline{A} + \overline{B} + \overline{C}$       D.  $(\overline{A} \cdot \overline{B}) + \overline{C}$

## **2 Data Structures**

---

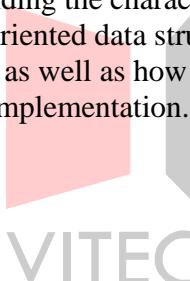
### **Chapter Objectives**

Choosing the most appropriate data structure and data description procedure is the key to creating efficient, easy-to-understand program.

This chapter describes various data structures you must understand as the first step to learning programming

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

- 1 Understanding how various data structures can be classified
- 2 Understanding the most commonly used basic data types and data arrays
- 3 Understanding the characteristics and mechanisms of problem oriented data structures used to solve specific problems, as well as how to use a basic data structure for program implementation.



<http://www.vitec.org.vn>

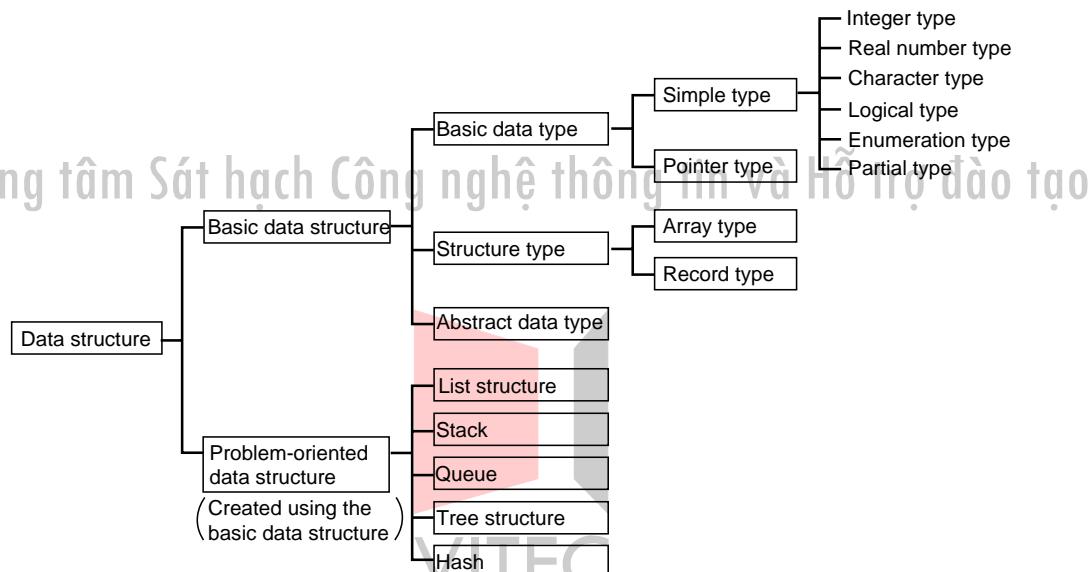
## 2.1 What is the data structure?

---

A set of the same kind of data processed by a computer is called a "data type". In the program design stage, the way data should be represented and programmed into a computer must be examined carefully, so that the most appropriate data type can be chosen. A data type that is represented and programmed is called a "data structure".

Figure 2-1-1 shows the Classification of data structures

Figure 2-1-1 Classification of data structures



The basic data structure can be represented in almost all programming languages. The problem oriented data structure is a data structure that can be effectively used to solve specific problems. These are some problem oriented data structures that cannot be represented in programming languages. In that case, the basic data structure is used.

## 2.2 Basic data structure

---

### 2.2.1 Basic data type

The basic data type is a set of individual data and is frequently used to create a program. It is classified into simple and pointer types.

#### (1) Simple type

The simple type is the most basic data type. When using the simple type for programming, the data type is usually declared according to the syntax rule of a language.

##### ① Integer type

The integer type represents integers and is represented inside a computer as binary numbers of fixed-point numbers that have no significant digits below the decimal point. A maximum or minimum value of the integer type is the unit of data that a computer can process at one time, and it is determined by the length of one word.

##### ② Real number type

The real number type represents real numbers. It is used to represent fixed-point and floating point numbers.

##### ③ Character type

The character type represents alphabets, numerals and symbols as characters. A character code is expressed as a binary number inside a computer.

##### ④ Logical type

The logical type is used to perform logical operations, such as AND, OR and NOT operations.

##### ⑤ Enumeration type

The enumeration type is defined as a data type that enumerates all possible values of variables. In the case of the enumeration type, integers can be named.

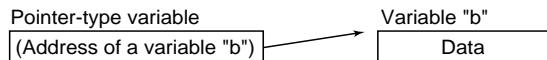
##### ⑥ Partial type

The partial type is used to specify an original-value subset by constraining existing data types. the data type that has upper and lower limits specified as constraints is called the partial range type.

## (1) Pointer type

The pointer type has an address allocated in a main memory unit. It is used to refer to variables, file records or functions. It is used for Pascal and C but not for FORTRAN and COBOL.

Figure 2-2-1 Image of the pointer type



## 2.2.2 Structured type

A data structure that contains a data structure or any defined data types as its elements (data) is called the structured type. The structured type is classified into the array type and record type.

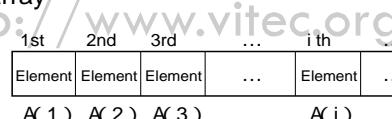
### (1) Array type

An array is called a table. The array type is a data structure that contains data of the same type and size. Each individual data is called an array element, a table element or an element. How arrays are described or how data is arranged varies, depending on the programming language used.

#### ① One-dimensional array

A one-dimensional array has a data structure in which data is arrayed in a line. To specify an element in this array, first enter a left parenthesis ( or left bracket [ after the name of an array, then enter a subscript and a right parenthesis ) or right bracket ]. A subscript, also called an index, indicates the ordinal number from the top of an array where a specified element is located. An array "A" having the number of elements denoted by "i" is expressed as A(i).

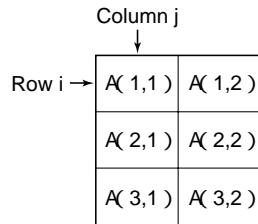
Figure 2-2-2 One-dimensional array



#### ② Two-dimensional array

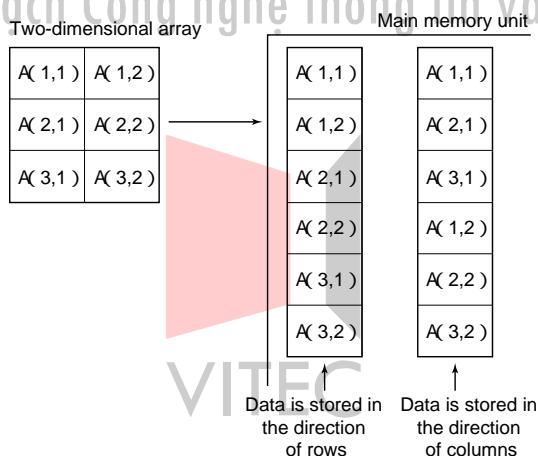
A data structure in which data is lined up in both vertical and horizontal directions is called a two-dimensional array. Data in a vertical direction is called a row and data in the horizontal direction is called a column. To specify an element in this array, two subscripts become necessary: one subscript showing in an ordinal number in a vertical direction (in what column) where a specified element is located and the other subscript showing in what ordinal number in a horizontal direction (in what row) it is located. An array "A" located in the "i" column and "j" row, for example, can be expressed as A(i,j).

**Figure 2-2-3** Two-dimensional array (with three rows and two columns)



When a two-dimension array is stored in a main memory unit, it takes the form of a one-dimensional array. The two-dimension array shown in Figure 2-2-3 takes the form of a one-dimensional array having six elements. As shown in Figure 2-2-4, data is stored sequentially in either the direction of columns or the direction of rows. The direction in which data is stored varies, depending on the compiler of the programming language used. In general, data is stored vertically when FORTRAN is used and horizontally when COBOL is used.

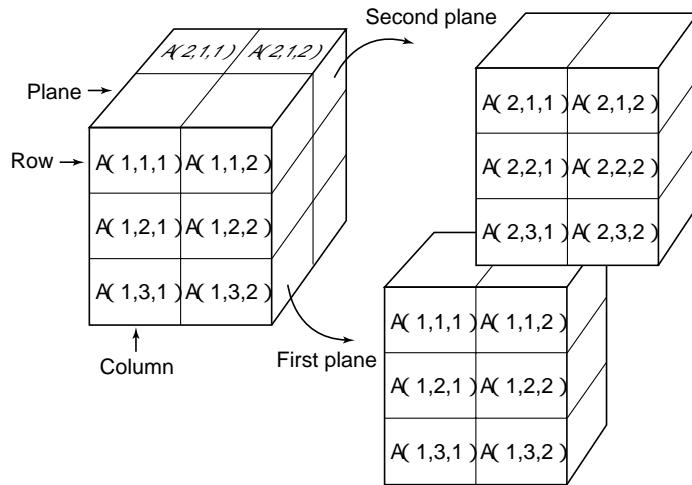
**Figure 2-2-4** How data of a two-dimensional array is stored in a main memory unit



### ③Three-dimensional array

A three dimensional array has a data structure in which more than one-dimensional is contained. It has a three-dimensional structure comprising planes, columns and rows as elements. By developing a three-dimensional array into a two-dimensional one, a three dimensional array can be handled in the same way as a two-dimensional array.

**Figure 2-2-5** Developing a three-dimensional array into a two-dimensional array



## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Multidimensional arrays such as four-, five- or n- dimensional arrays can be defined. However, there may be certain limitations to the number of definable dimensions depending on the type of programming language or compiler.

Arrays can be classified into static and dynamic arrays according to the method used to secure an area.

- Static array: An array for which a required area is determined by a program
- Dynamic array: An array for which a required area is determined after a subscript is used for arraying is provided with an expression and the expression is evaluated during execution of a program.

### (2) Record type

<http://www.vitec.org.vn>

Although structured type data is superior in the ease of element reference and operation, it has a drawback in that it can handle only data of the same type. Therefore, data that contains data of different types must take the form of record-type data. This record type is also called a structure.

**Figure 2-2-6** Record type

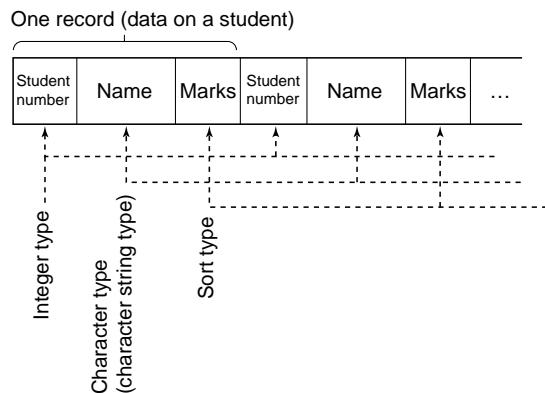
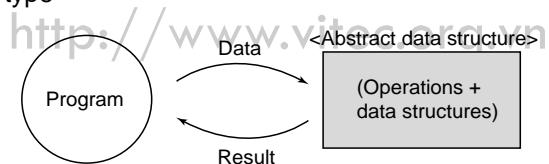


Figure 2-2-6 shows the data structure of the record type. One record contains a student number (integer type), a name (character type) and marks (integer type). One record type data contains a set of records of this format. Although one-dimensional record type data can be handled in the same way as a one-dimensional array, each data must be named for identification since each element contains more than one data.

### 2.2.3 Abstract data type

Data that contains a certain data structure and type of operations is called an abstract data type. To access this type of data, you do not need to be aware of its internal structure. All data are hidden except the data that you can access for reference, insertion or deletion. This is called information hiding. Hiding information or hiding data on the level of data types is called data encapsulation.

**Figure 2-2-7** Abstract data type



## 2.3 Problem-oriented data structure

Various problem-oriented data structures can be designed using the array type, pointer type and other basic data structures.

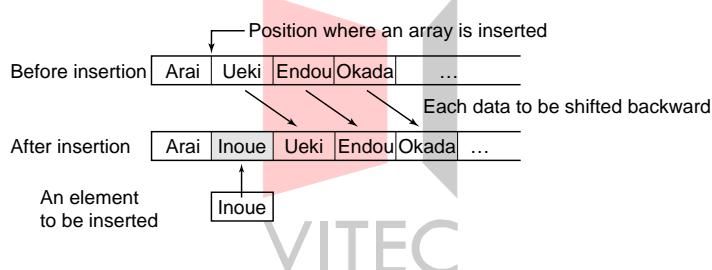
### 2.3.1 List structure

Unlike the basic data type that handles individual data, a list structure allows data to be linked to each other and handled in one lump. Data arranged according to this list structure is called a list.

#### (1) List structure and cells

Using a subscript for each element in an array, quick access to any element is possible. Also, a change to data can be made easily. If you insert one piece of data somewhere in arrays, you must shift each of all pieces of data subsequent to the inserted element backward. If you delete one piece of data in arrays, you must likewise shift each of all pieces of data subsequent to the deleted piece forward.

Figure 2-3-1 Inserting an array element



Unlike the array type structure, the list structure allows data element to be inserted or deleted easily. The list structure is similar to the array structure in that data elements of the same type are sequentially lined up.

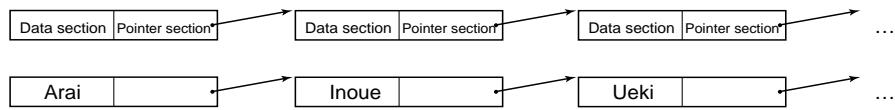
The array type requires that the logical arrangement of elements is the same with the physical arrangement of elements stored in a main memory unit. In the case of the list structure, the logical arrangement does not need to match the physical arrangement.

The list contains cells and each cell consists of the following:

- Data section that contains data element
- Pointer section that contains an address

Therefore, the data section of a cell has the same structure as that of stored data and the pointer section of a cell has a pointer type data structure. This means that cells represent record-type (structure) data that contain elements of different data structures. The list contains cell addresses in the pointer section and one cell is linked to another via a pointer.

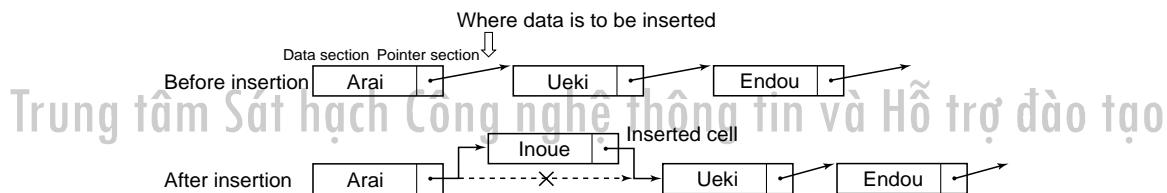
**Figure 2-3-2** List structure



## (2) Inserting data into a list

To insert data into a list, all you have to do is to replace pointers preceding and subsequent to data to be inserted. Because you do not have to shift elements as in the case of array-type data, you can insert data easily and quickly. (See Figure 2-3-3)

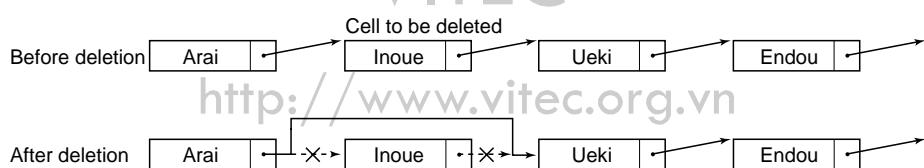
**Figure 2-3-3** Inserting data into a list



## (3) Deleting data from a list

To delete data from a list, all you have to do is to replace pointers as you do to insert data into a list. Cells that contain data (Inoue) remain in a memory area as garbage after data is deleted, as shown in Figure 2-3-4.

**Figure 2-3-4** Deleting data from a list



Although the list structure allows data to be inserted or deleted by simply replacing pointers, it has the drawback that you must track each pointer one by one from the top if you want to access specific data.

## (4) Types of list structures

Representative list structures include:

- Uni-directional list

- Bi-directional list

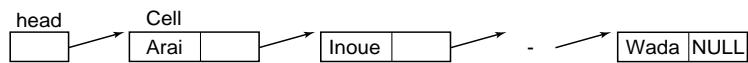
- Ring list

because these lists are represented in the form of a straight line, they are generically called linear lists.

### ①Uni-directional list

The uni-directional list is also called a one-way list. The pointer section of a cell contains the address of a cell in which the next data is stored. By tracking these addresses one by one, you can perform a search on data.

**Figure 2-3-5** Uni-directional list

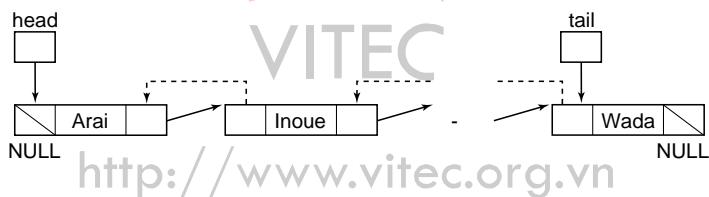


The first pointer is called a root or head. Because the pointer section of the last cell does not have any address in which data can be stored. NULL (Numerical value of zero) or \ is entered in this section.

### ②Bi-directional list

The bi-directional list has two pointer sections ( $\diamond$  and  $\blacklozenge$ ) which contain cell addresses as shown in Figure 2-3-6.

**Figure 2-3-6** Bi-directional list



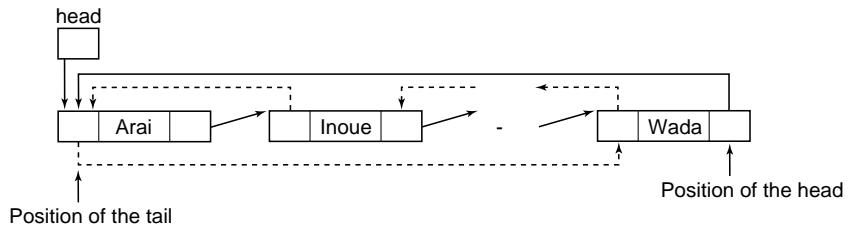
The pointer sections  $\diamond$  and  $\blacklozenge$  shown in Figure 2-3-6 contains the address of the subsequent cell and the address of the preceding cell respectively. The address of the last cell is contained in the pointer tail. In the case of the bi-directional list, data can be tracked from either the head or the tail of cells.

### ③Ring list

A bi-directional list containing NULL in the first cell is called a ring list. The pointer section of this first cell and the pointer section containing NULL of the last cell contain addresses of one another, thus data is in the form of a ring. Data can be searched in the same way as in the bi-

directional list.

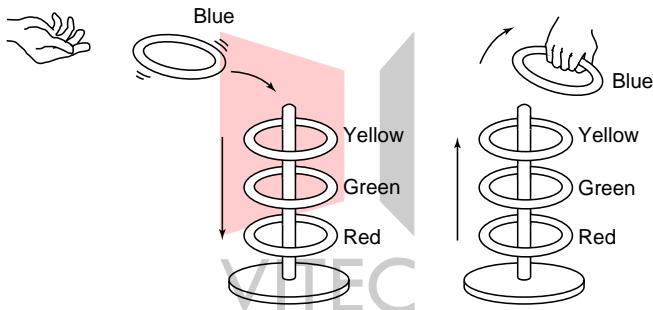
Figure 2-3-7 Ring list



### 2.3.2 Stack

A stack is a data structure designed based on a one-dimensional array. Last stored data is read first. It is likened to the game of quoits.

Figure 2-3-8 Quoits

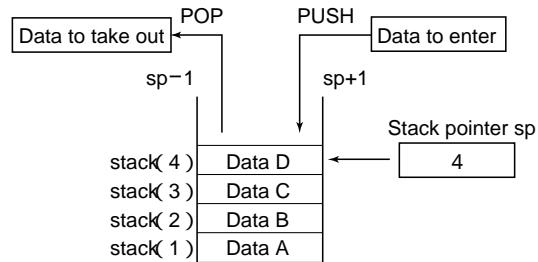


Quoits are thrown in the order of red, green, yellow and blue (data input). They are retrieved one by one (data output) in reverse order of throwing, i.e. blue, yellow, green and red. That is, blue quoit last thrown is first retrieved.

This type of data structure which can be likened to the game of quoits is called a stack. This system is termed the last-in first-out (LIFO) system. Storing data in a stack is called "push down (PUSH)" and taking data from a stack is called "pop up (POP)". A variable that controls pushing down and popping up of data is called a stack pointer.

To push down data into a stack, set the stack pointer "sp" to +1 and store data in array elements indicated by "sp". To pop up data from a stack, take out data stored in array elements indicated by "sp" and set the stack pointer to sp-1.

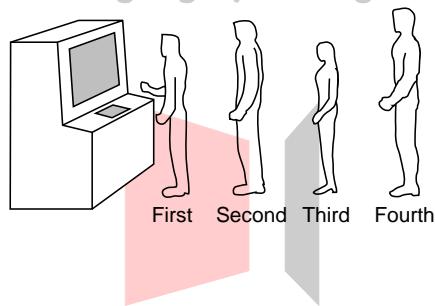
**Figure 2-3-9** Stack structure



### 2.3.3 Queue

A queue is a data structure designed based on one-dimensional arrays. Data first stored is first read. It is likened to a queue of people who are waiting in front of a cash dispenser machine of a bank.

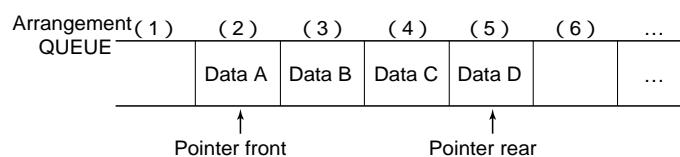
**Figure 2-3-10** Queue



A data structure that allows customers to be served on the first-come first served basis is called a queue.

This system is called the first-in first-out (FIFO) system. Two pointers that indicate the head and tail of a queue are required for queue control. Pointers that indicate the head and tail of a queue are represented as a front variable and a rear variable respectively. (See Figure 2-3-11.)

**Figure 2-3-11** Queue structure



<Queue operation procedure>

1. To store data in a queue (enqueue), set a rear variable of a pointer indicating the tail to +1 and store data
2. To take out data from a queue (dequeue), take out data and set a front variable of a pointer

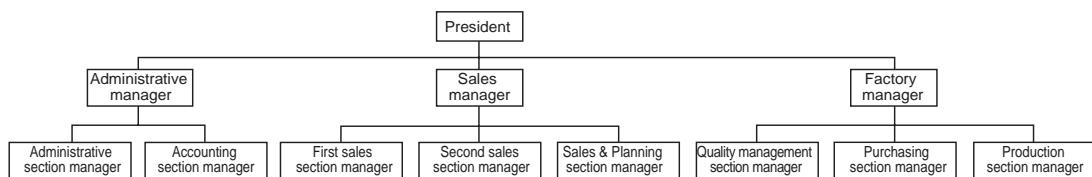
indicating the head to +1.

### 2.3.4 Tree structure

The tree structure is one of very useful data structures since it can control complex data better than array- or list type data structures.

Because it has a hierarchical structure like a company organization or family tree, it is suited for the type of work involving step-by-step classification.

Figure 2-3-12 Organization chart



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
Although each cell is linearly correlated in the list structure, data is correlated while it branches off in the tree structure.

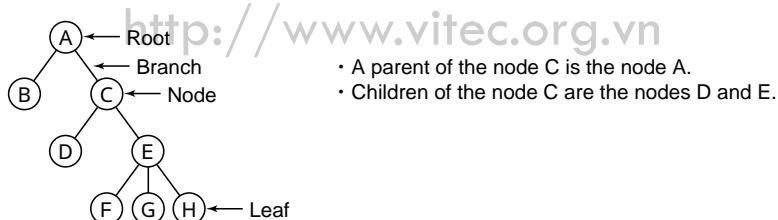
The tree structure consists of elements shown below:

- Node: Correspond to data
- Branch: Connecting one node to another
- Root: Highest-order node that has no parent
- Child: Node that branches off under a node
- Parent: Original data before it begins to branch
- Leaf: Lowest-order node that has no child

Figure 2-3-13 shows the tree structure.

VITEC

Figure 2-3-13 Tree structure

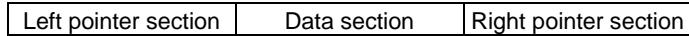


#### (1) Binary tree

If the number of branches branching off from a node is "n" or less, that is, if the number of children is 0 to "n", such a tree structure is called an N-ary tree. An N-ary tree that has two branches ( $n=2$ ), that is, an N-ary tree that has no child, one child or two children is called a binary tree, which is the most commonly used data structure. On the other hand, a tree structure that has three or more branches ( $n>2$ ) is called a multiway tree.

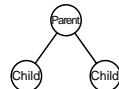
Binary-tree data consists of one data section and two pointer sections. The left pointer section shows the position of a node that extends to the left and branches off while the right pointer section shows the position of a node that extends to the right and branches off.

**Figure 2-3-14** Binary-tree data structure



The binary tree has a parent-child hierarchical structure, as shown in Figure 2-3-15.

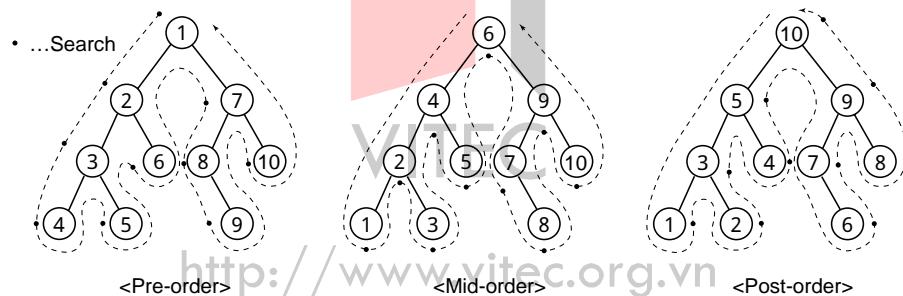
**Figure 2-3-15** Basic binary-tree structure



<How to perform a search on binary tree data>

To find specific data in binary tree data, each node must be tracked one by one. There are three methods to perform a search on binary tree data. As it is difficult to explain them in words, check Figure 2-3-16 and see how data is searched using each method.

**Figure 2-3-16** How to perform a search on binary-tree data

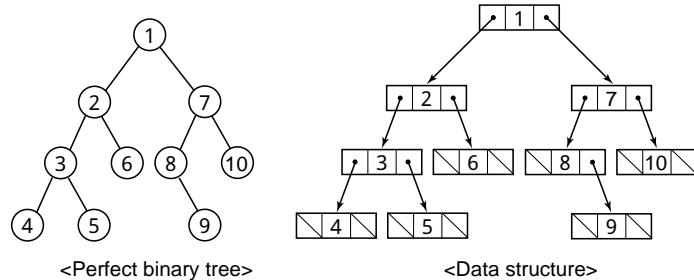


- Pre-order: With a root as a starting point, the left side of each node is tracked in a sequentially.
- Mid-order: With a leaf at the bottom left as a starting point, the underside of each node is tracked in a sequential way
- Post order: With a leaf at the bottom as a starting point, the right side of each node is tracked in a sequential way.

## (2) Perfect binary tree

If a binary tree is constructed in such a way that the number of branches from a root to each leaf along one branch is equal to or different by one from that of branches from a root to each leaf along another branch (or if the height from a root to each leaf along one branch is equal to or different by one from that from a root to each leaf along another branch), it is called a perfect binary tree. Figure 2-3-17 shows a perfect binary tree that has ten nodes.

**Figure 2-3-17** Perfect binary tree



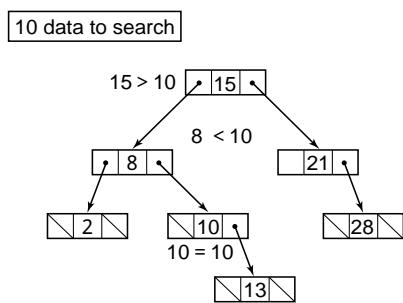
### (3) Binary search tree

A binary search tree is used as a variant of a binary tree. In the case of a binary tree, a descendent to the left is smaller than a parent and a descendent to the right is larger than a parent.

The algorithm of a binary search tree is as follows:

1. A root is a point where a search starts
2. Binary tree data is compared with data to search
3. If binary tree data = data to search, a search is successful (completed)
4. If binary tree data > data to search, nodes to the left of a binary tree are searched and compared
5. If binary tree data < data to search, nodes to the right of a binary tree are searched and compared
6. If no child is found, a search is unsuccessful (no data is found)

**Figure 2-3-18** Performing a search on data in a binary search tree



### (4) Balanced tree

If data is added or deleted in a tree structure, leaves randomly develop and the operation

efficiency will decrease. A tree structure capable of reorganizing itself is called a balanced tree. Representative balanced trees are a B-tree and a heap.

### ①B-tree

A B-tree is a further developed version of a binary tree:

- End leaves are removed and each node has a number of nodes specified as "m."
- Each node has a maximum number of data specified as "m-1"
- All leaves are on the same level.
- Data contained in each node is arranged in a queue.

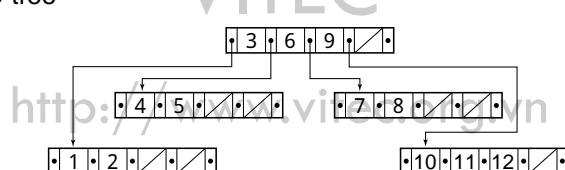
Because the above characteristics can increase the memory and calculation efficiency, the name "B-tree" means a well-balanced tree.

#### a. Characteristics of a B-tree

- To increase the memory area usage, the number of pointers that each node has is set at  $m/2$  or more and  $m$  or less. The number of pointers along a route, however, is set at 2 or more.
- Each time data is deleted or added, splitting and concatenation are automatically executed so that the number of elements of a node become  $m/2$  or more or  $m$  or less. This allows leaves to be always maintained on the same level.
- A search starts from the root
- Addition or deletion of data starts from a leaf

Figure 2-3-19 shows a quint-B-tree with elements | 7, 6, 10, 2, 5, 12, 4, 9, 8, 11, 1, 3 |

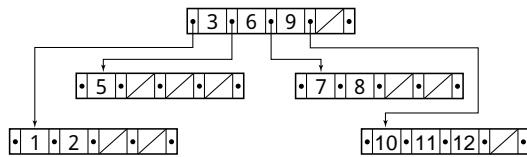
Figure 2-3-19 Quint-B-tree



#### b. Deleting data

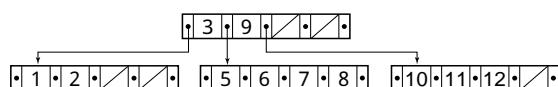
Figure 2-3-20 shows a case where the data "4" is deleted from a B-tree shown in Figure 2-3-19. If "4" is deleted, the number of deleted elements of a node becomes one and the requirements for a B-tree cannot be met.

Figure 2-3-20 Wrong B-tree



The node from which "4" is deleted is merged with an adjacent node either to the right or the left. Because the pointers of a parent must also be reorganized, "6" is moved to a lower order node and each node can be reorganized, as shown in Figure 2-3-21.

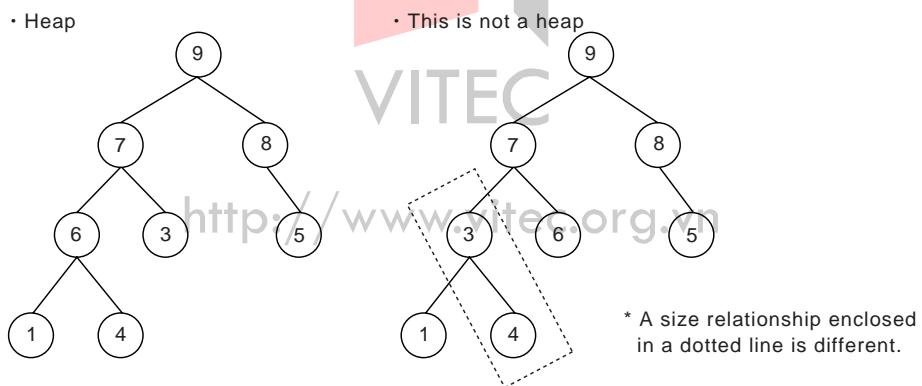
Figure 2-3-21 Correct B-tree



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
②Heap

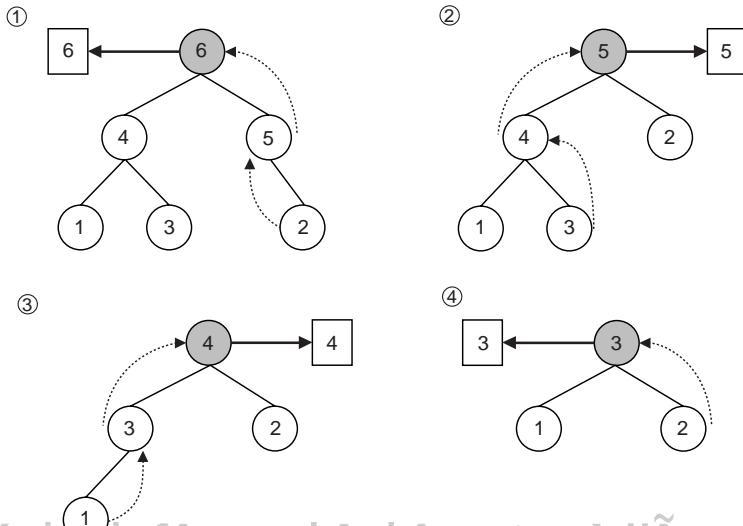
A perfect binary tree that has a certain size relationship between a parent node and a child node is called a heap. A heap is different from a binary search tree in that a heap does not have a certain size relationship between brothers.

Figure 2-3-22 Example of a heap



Because a heap has the structure of a perfect binary tree, it is a kind of organizable, balanced tree. In the case of a heap, maximum values (or minimum values) of all data are recorded in a root. Using this characteristic, data can be sorted by taking out data from a root in a sequential way.

**Figure 2-3-23** Sorting data using a heap



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

## 2.3.5 Hash

A hash is a way of using an array-type data structure. Using a hash, you can directly access specific data using a key without accessing recorded data one by one.

### (1) Converting to a hash address

To convert a key to a hash address (subscript), a hash function is used. Representative hash functions are

- Division method
- Registration method
- Base conversion method

<http://www.vitec.org.vn>

#### ① Division method

A key is divided by a certain value (a prime number closest to the number of array elements is usually used) and the remainder is used as the address (subscript) of a key.

#### Example

Key: 1234 and the number of array elements: 100

$$1234 \div 97 \text{ (a prime number closest to 100)} = 12$$

70: Address (subscript)

#### ② Registration method

A key is decomposed according to a certain rule and the total is used as the address (subscript) of a key. used) and the remainder is used as the address (subscript) of a key.

### Example

Key: 1234

1234 is decomposed into 12 and 34 and both numbers are totaled.

$$12 + 34 = 46 \quad : \text{Address (subscript)}$$

### ③ Base conversion method

A key is normally represented in decimal numbers. This key is converted into a radix other than decimal numbers (tertiary numbers, for example) and the result is used as the address (subscript) of a key.

### Example

Key: 1234

$$1 \times 3^3 + 2 \times 3^2 + 3 \times 3^1 + 4 \times 3^0 = 27 + 18 + 9 + 4 = 58 : \text{Address (subscript)}$$

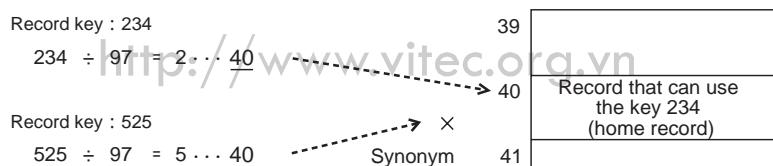
- The value of each digit of a tertiary numbers is 3 or smaller. In this case, however, this fact does not need to be considered.

## (2) Synonym

When a key is converted to an address using a hash function, different keys can be converted to the same address. This is called a synonym (collision) (see Figure 2-3-24). A record that can use the converted address is called a home record and a record that cannot use it is called a synonym record.

Figure 2-3-24 Occurrence of a synonym

- If a key is converted to 97 using the division method



### ① Sequential method

Using a sequential method, a synonym record is stored in free space that is located close to the desired address. There is the possibility that synonyms may occur in a chain reaction.

### ② Chain method

Using a chain method, a separate memory is established and synonym records are stored in this area.

In this case, it is necessary to provide a pointer that shows the address where synonym records are stored.

### 2.3.6 Backus-Naur Form

This notation also known as the BNF notation can be used to describe the structure.

The following symbols are used.

<, >, ::=, |

Variables are enclosed between < and >.

The symbol --> is replaced with ::=.

The symbol | is used to separate alternatives.

Terminals are represented by themselves or are written in a type face different from the symbols of the BNF.

Arithmetic expressions in BNF notation is shown below

```
<Expression> ::= <Identifier> | <Number> |
<Expression> <Op> <Expression> |
( <Expression> )
<Op> ::= + | - | * | /
<Identifier> ::= <Letter>
<Identifier> ::= <Identifier> <Letter>
<Number> ::= <Digit>
<Number> ::= <Number> <Digit>
<Letter> ::= A | ... | Z
<Digit> ::= 0 | ... | 9
```

### 2.3.7 EBNF (extended BNF)

Extensions to the original BNF notation have been suggested improve the readability.

One extension is to write the names of the variables in italics and without < and >.

In addition, the EBNF (extended BNF) is a combination of the BNF and the notation of regular expressions.

An EBNF production rule is of the form  $N ::= E$ , where  $N$  is a nonterminal symbol and  $E$  is an extended regular expression.

Like ordinary regular expressions,  $E$  may contain '|', '\*', and parentheses for grouping but unlike ordinary regular expressions,

it may contain variable symbols as well as terminal symbols.

Some additional extensions include the use of braces, { $E$ }, or ellipses,  $E\dots$ , to indicate zero or more repetitions of an item and brackets,

[ $E$ ], to indicate an optional item.

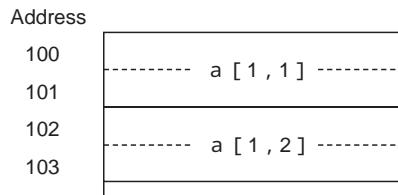
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

## Exercises for No.1 Chapter 2 (Data Structures)

**Q1** When storing a two-dimensional array "a" with ten rows and ten columns in continuous memory space in the direction of rows, what is the address where a [5, 6] is stored? In this question, the address is represented in decimal numbers.



- a. 145      b. 185      c. 190      d. 208      e. 212

**Q2** The figure below is a uni-directional list. Tokyo is the head of this list and the pointer contains addresses of data shown below. Nagoya is the tail of the list and the pointer contains 0. Choose one correct way to insert Shizuoka placed at address 150 between Atami and Hamamatsu.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Pointer for the head      Address      Data      Pointer

10	Tokyo	50
30	Nagoya	0
50	Shin Yokohama	90
70	Hamamatsu	30
90	Atami	70
150	Shizuoka	

- a. The pointer for Shizuoka to be set to 50 and that for Hamamatsu to be set to 150  
b. The pointer for Shizuoka to be set to 70 and that for Atami to be set to 150  
c. The pointer for Shizuoka to be set to 90 and that for Hamamatsu to be set to 150  
d. The pointer for Shizuoka to be set to 150 and that for Atami to be set to 90

**Q3** What is the data structure suitable for the FIFO (first-in first-out) operation?

- a. Binary tree      b. Queue      c. Stack      d. Heap

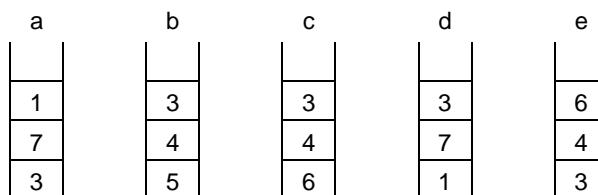
**Q4** Two stack operations are defined as follows:

PUSH n: Data (integer "n") is pushed into a stack.

POP: Data is popped out of a stack.

If a stack operation is performed on an empty stack according to the procedure shown below, what result can be obtained?

PUSH 1 → PUSH 5 → POP → PUSH 7 → PUSH 6 → PUSH 4 → POP → POP → PUSH 3



**Q5** Figure 2 is the array representation of a binary tree shown in Figure 1. What value should be put into the space "a"?

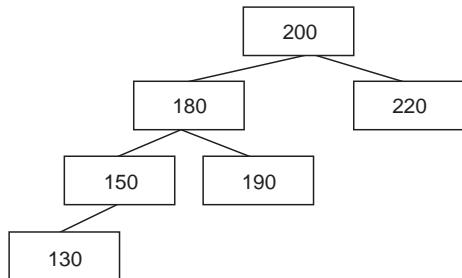


Figure 1 Binary tree

Subscript	Value	Pointer 1	Pointer 2
1	200	3	2
2	220	0	0
3	180	5	a
4	190	0	0
5	150	6	0
6	130	0	0

Figure 2  
Array representation of a binary tree

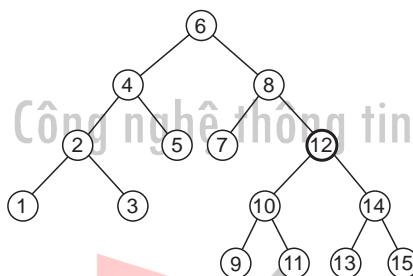
a. 2

b. 3

c. 4

d. 5

**Q6** When the element 12 is deleted from the binary search tree shown below, which element should be moved to the point where the element was deleted to reorganize a binary search tree?



a. 9

b. 10

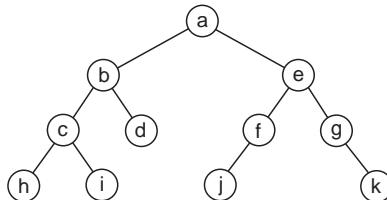
c. 13

d. 14

**Q7** If two or less than two branches branch off from each node of a tree, such a tree is called a binary tree. A binary tree consists of one node, a left tree and a right tree. There are three methods of performing a search on this tree:

- (1) Pre-order: A search is performed in the order of a node, a left tree and a right tree.
- (2) Mid-order: A search is performed in the order of a left tree, a node and a right tree.
- (3) Post-order: A search is performed in the order of a left tree, a right tree and a node.

If a search is performed using the post-order method, which can be output as the value of a node?



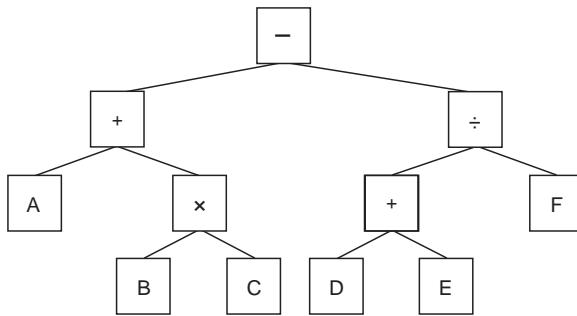
a. abchidefjgk

b. abechidfjgk

c. hcibdajfegk

d. hicdbjfkgae

**Q8** A binary tree shown below can be represented using an arithmetic expression. Which expression is correct?

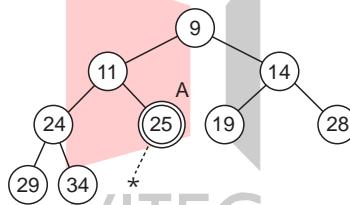


- a.  $A + B \times C + (D + E) \div F$   
 b.  $A + B \times C - (D + E) \div F$   
 c.  $A + B \times C - D + E \div F$   
 d.  $A \times B + C + (D - E) \div F$   
 e.  $A \times B + C - D + E \div F$

**Q9** Regarding storing data using a B-tree, choose one correct comment from the following:

- a. Split and merge nodes to allow the hierarchical depth to become the same.  
 b. Identify the position where data is to be stored by using a certain function and key value.  
 c. Only sequential access to head data and subsequent data is possible.  
 d. There are a directory and a member. A member is a sequentially organized file.

**Q10** There is a heap with the value of a parent's node smaller than that of a child's node. To insert data into this heap, you can add an element to the backmost part and repeat exchanging a parent with a child when the element is smaller than a parent. When element 7 is added to the position \* of the next heap, which element comes to the position A?



- a. 7      b. 9      c. 11      d. 24      e. 25

**Q11** A five-digit number ( $a_1 a_2 a_3 a_4 a_5$ ) must be stored in an array using the hash method. If the hash function is defined as  $\text{mod}(a_1 + a_2 + a_3 + a_4 + a_5, 13)$  and if the five-digit number is to be stored in an array element located in the position that matches a calculated hash value, in which position will 54321 be placed in an array shown below? Here, the  $\text{mod}(x, 13)$  value is the remainder when  $x$  is divided by 13.

Position	Array
0	
1	
2	
...	
11	
12	

- a. 1      b. 2      c. 7      d. 11

**Q12** Five data with key values distributed uniformly and randomly in the range of 1 to 1,000,000 must be registered in a hash table of 10 in size. What is the approximate probability that

**collisions occur? Here, the remainder obtained when a key value is divided by the size of a hash table is used as a hash value.**

- a. 0.2
- b. 0.5
- c. 0.7
- d. 0.9

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

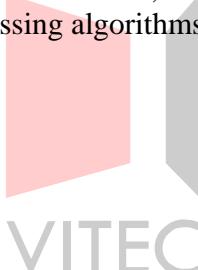
## **3 Algorithms**

---

### **Chapter Objectives**

The basis of programming is algorithm design. In designing the logical structure of a program, it is important to use the most appropriate algorithm. Also, in choosing a program from a library, what algorithm to use is one of the most important criteria for choosing the most appropriate program. This chapter describes the basics of algorithm design and representative algorithms.

- 1 Understanding the basics of algorithms, i.e. algorithm definitions, design, relationships with data structures representative methods, etc
- 2 Understanding the characteristic and meanings of representative search, sort, character string processing and file processing algorithms.



<http://www.vitec.org.vn>

# Introduction

---

Using an efficient, easy to understand algorithm enables one to increase the execution speed and decrease the number of hidden bugs. An algorithm is one of the critical factors that determine the system performance. Also, the quality of the algorithm is used as criteria for choosing parts from a library.

This chapter describes the basics of algorithm used for module logic design and the algorithms used to solve typical problems. Choosing a well-known or generally used algorithm without fully analyzing given problems should be avoided. An algorithm that is the most appropriate for characteristics of problems must be chosen.

Evaluating algorithms themselves is also an important task. Data obtained by comparing plural algorithms on the basis of objective figures will greatly help you choose the most appropriate algorithm.

In this chapter, you learn the basics of algorithms so that you can design an easy to understand module.

## 3.1 Basic of algorithms

---

This section explains the following:

- Definition of an algorithm
- Relationships between an algorithm and a data source

### 3.1.1 What is an algorithm?

#### (1) Definition of an algorithm

An algorithm is defined in the Japanese Industrial Standard (JIS) as follows:

A set of a limited number of clearly defined rules that are applied a limited number of times to solve problems.

This means that an algorithm is a set of rules (procedures) established to solve problems. To solve one problem, there are several routes that we can take. As a simple example, when we calculate a value Y times as large as X, we can take two approaches:

- Multiply X by Y
- Adding X Y times

In determining an algorithm, it is important to not only think out a procedure for solving a problem but also design and adopt an algorithm that can solve a problem effectively and efficiently.

## (2) Algorithm and programming

An algorithm and programming are two sides of the same coin. Programming is also describing data and algorithms in a programming language so that a computer can execute its given tasks. In general, a program consists of algorithms and data and an algorithm consists of logic and control.

Programming can be classified into four different types according to how algorithms are viewed.

- Procedural programming
- Functional programming
- Logic programming
- Object programming

The most appropriate type of programming must be chosen with consideration of the characteristics of a problem.

### ① Procedural programming

Procedural programming is the most commonly used type of programming. This category includes the following programming languages, FORTRAN, COBOL, PL/I, Pascal, C, etc.

Characteristic

- A program is divided into modules to make a large, complicated program easy to understand.
- Coding is performed for each module
- Structured theorems are introduced to programming (to be explained later)
- Structured figures are used to compile a program

Because procedural programming is procedural (control)-oriented, they are the following restrictions:

- In addition to a procedure (control), variables (variable names, types, sizes, etc) must be declared.
- Instructions are executed one by one in a sequential manner (parallel processing cannot be done).
- Comparisons and calculations must be made to solve a problem.

### ② Functional programming

Functional programming is functional-oriented programming. It is used in the field of artificial intelligence (AI), basic calculation theories and other research tasks. LISP, among others, is a functional programming language.

Characteristic

- Unlike sequential execution type programming, expressions are built by nesting and they are replaced with results of calculations to execute a program.
- Recursive calls can be described easily
- The level of affinity with parallel processing is high
- A calculation process or procedure does not need to be considered.

### ③Logic programming

A predicate based on facts and inferences is the base of logic programming. Prolog is an example of a logic programming language.

#### Characteristic

- Because facts are described based on the predicate logic, the process of programming is made easy.
- Inferences and logical calculations can be made easily.

### ④Object programming

In object programming, a system is considered a group of objects. Languages include Smalltalk, C++, Java and others.

## 3.1.2 Algorithm and the data structure

An algorithm and the data structure are closely related to one another. The data structure determines the framework of an algorithm to a certain extent.

### (1) Data structure

The data structure is defined as follows:

A procedure that a program follows to store data and perform given tasks.

#### ①Basic data structure

Storing data means storing data in a main memory unit. In storing data in a main memory unit, the data type (data type, size, etc) must be declared. The most basic data structure unit used to declare the data type is called a basic data structure. In performing this step of data type declaration, data is manipulated using the name of data whose data type has been declared.

(See Figure 3-1-1.)

#### ②Problem oriented data structure

The problem oriented data structure built by combining basic data structures contains one or more of the following:

- List
- Stack
- Queue
- Tree structure

These elements are determined by programming (designed algorithms). If data is processed in the order it is input, a queue is used. If data is processed in the reverse order of input, a stack is used. Therefore, the contents of the problem oriented data structure are determined by algorithms to a certain extent.

**Figure 3-1-1** Data definition and procedure division in a COBOL program

```

DATA DIVISION.
FILE SECTION.
FD TOGETU-FILE.
01 T-REC.
  02 T-HIZUKE PIC X(06).
    88 T-ENDF VALUE HIGH-VALUE.
  02 FILLER PIC X(34).

FD RUISEKI-FILE.
01 R-REC.
  02 R-HIZUKE PIC X(06).
    88 R-ENDF VALUE HIGH-VALUE.
  02 FILLER PIC X(34).

FD N-RUISEKI-FILE.
01 N-REC PIC X(40).

WORKING-STORAGE SECTION.
01 T-COUNT PIC 9(05).
01 R-COUNT PIC 9(05).
01 N-COUNT PIC 9(05).
* PROCEDURE DIVISION.

HEIGOU-SYORI.
  OPEN INPUT TOGETU-FILE RUISEKI-FILE
        OUTPUT N-RUISEKI-FILE.
  INITIALIZE T-COUNT R-COUNT N-COUNT.
  PERFORM T-YOMIKOMI.
  PERFORM R-YOMIKOMI.
  PERFORM UNTIL T-ENDF AND R-ENDF
    IF T-HIZUKE < R-HIZUKE
      THEN WRITE N-REC FROM T-REC
          PERFORM T-YOMIKOMI
      ELSE WRITE N-REC FROM R-REC
          PERFORM R-YOMIKOMI
    END-IF
    COMPUTE N-COUNT = N-COUNT + 1
  END-PERFORM.
  DISPLAY T-COUNT R-COUNT N-COUNT.
  CLOSE TOGETU-FILE RUISEKI-FILE N-RUISEKI-FILE.
  STOP RUN.

T-YOMIKOMI.
  READ TOGETU-FILE
    AT END MOVE HIGH-VALUE TO T-HIZUKE
    NOT AT END COMPUTE T-COUNT = T-COUNT + 1
  END-READ.

R-YOMIKOMI.
  READ RUISEKI-FILE
    AT END MOVE HIGH-VALUE TO R-HIZUKE
    NOT AT END COMPUTE R-COUNT = R-COUNT + 1
  END-READ.

```

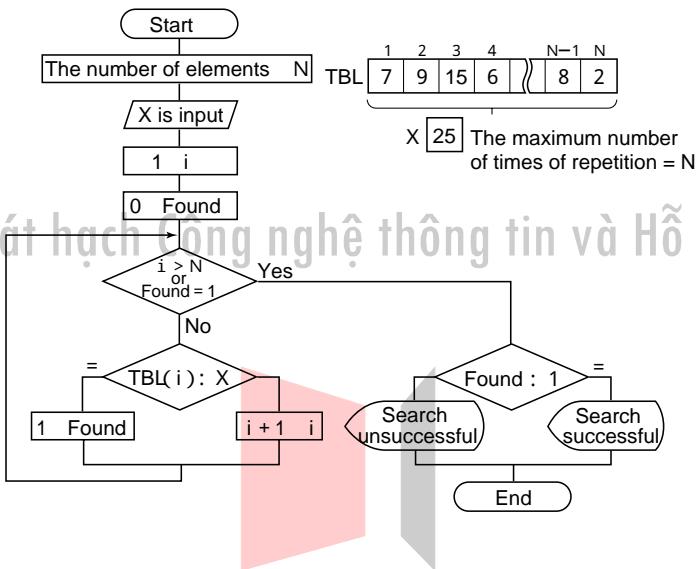
## (2) Relationships between an algorithm and then data structure

The relationships between an algorithm and the data structure can be described as follows:

### ① Array processing

Take a linear search algorithm (details explained in Section 3.2.1) shown in Figure 3-1-2 for example.

Figure 3-1-2 The linear search algorithm and the data structure

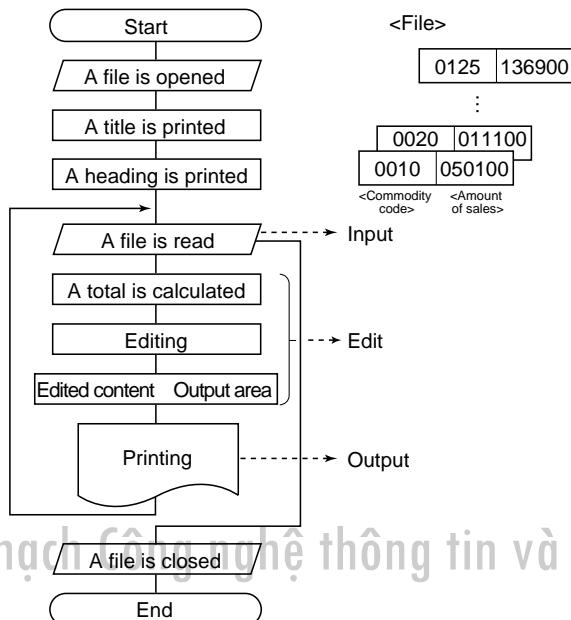


The linear search is most frequently used to search data. In performing the linear search on data, data is searched while subscripts in an array are being incremented. The maximum number of times the search is repeated is the size of an array. That is, if an array structure is used, the procedure and the number of times of repetition are determined.

### ② File processing

Take an algorithm for reading and printing a file shown in Figure 3-1-3 for example. (Details are explained in section 3.2.5)

**Figure 3-1-3** Algorithm for processing a file and the data structure



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

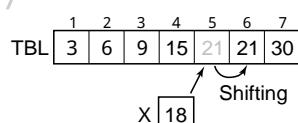
The process of reading, editing and printing a file is repeated until there is no data in a file.

Because a file must be accessed in each process cycle, this task is executed in the form of a loop.

### ③List processing

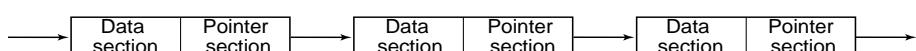
In processing an array structure, data can be searched and updated smoothly but it takes time to insert or delete data. Because data is arranged in queues, inserting or deleting data is inevitably accompanied by the shifting of related data backward or forward.

**Figure 3-1-4** Inserting data into an array



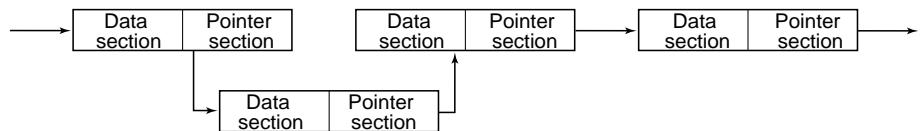
Using the list structure, inserting or deleting data is easy. Although data is arranged in an orderly manner in the list structure, it is arranged so in terms of logic since it does not need to be physically arranged in sequential order.

**Figure 3-1-5** List structure



In the case of the list structure, data does not need to be shifted as shown in Figure 3-1-6; data can be inserted or deleted by simply manipulating a pointer.

Figure 3-1-6 Inserting data into the list structure



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

## 3.2 Various algorithms

An algorithm should be thought out to solve each specific problem. If an algorithm that can solve similar problems is already designed and made available, using such an algorithm will enable you to create a better algorithm in an efficient manner.

This section describes the representative algorithms that have until now been developed in relation to each type of problem.

### 3.2.1 Search algorithm

A table search is the method of performing a search on tables and files stored in a memory unit to find the elements that meet specified requirements.

This section describes two table search methods: linear (or sequential) and binary search methods.

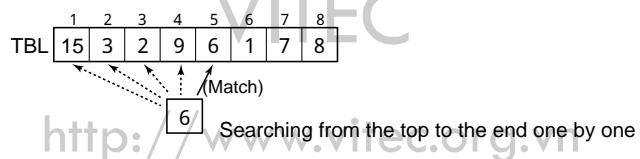
#### (1) Linear (or sequential) search method

A linear (or sequential) search method is a very simple search method of searching from the top of a table in a sequential manner.

##### ①Exhaustive search method

An exhaustive search method is the simplest method of combing a table from the top to the end.

Figure 3-2-1 Image of the exhaustive search method



Data retrieval is collated with each data in a table. The algorithm is designed such that the search is successful if there is data that matches data to retrieve and it is unsuccessful if there is no data that matched data to retrieve.

The search procedure ends when the first successful match occurs. Therefore, this search method is inappropriate for counting the number of data that matches data to retrieve.

##### ②Sentinel search method

**Figure 3-2-2** Sentinel search method

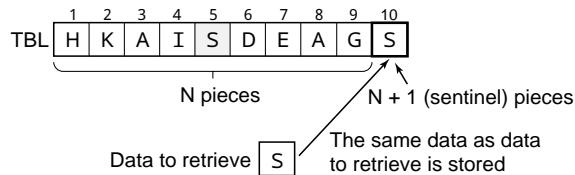
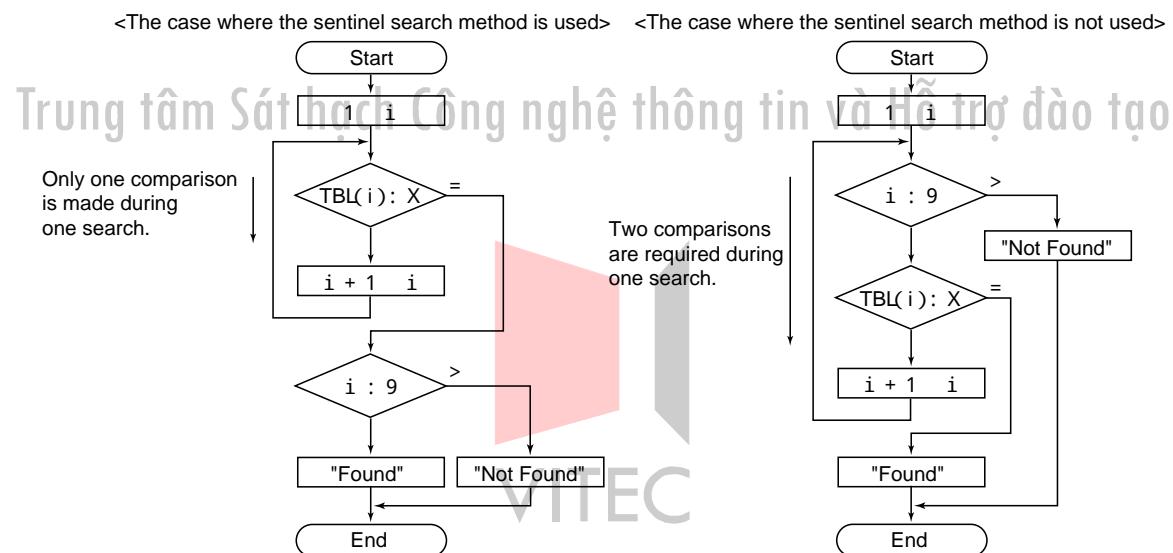


Figure 3-2-3 shows a comparison of the case where the sentinel search method is used and the case where it is not used.

**Figure 3-2-3** A comparison of the case where the sentinel search method is used and the case where it is not used



<The case where the sentinel search method is used>

In the first round of data collation, data to be retrieved is authenticated. In the second round, it is determined whether there is data that matches data to retrieve. That is, if the number of data is N, comparison is made only (N + 1) times.

<The case where the sentinel search method is not used>

In one round of data collation, the authenticity of data to retrieve as well as whether data to retrieve ends must be determined. That is, a comparison is made (N x 2) times.

<Maximum number of times a comparison is made if the number of elements is N>

- (N + 1) times if the sentinel search method is used

-  $(N \times 2)$  times if the exhaustive search method is used

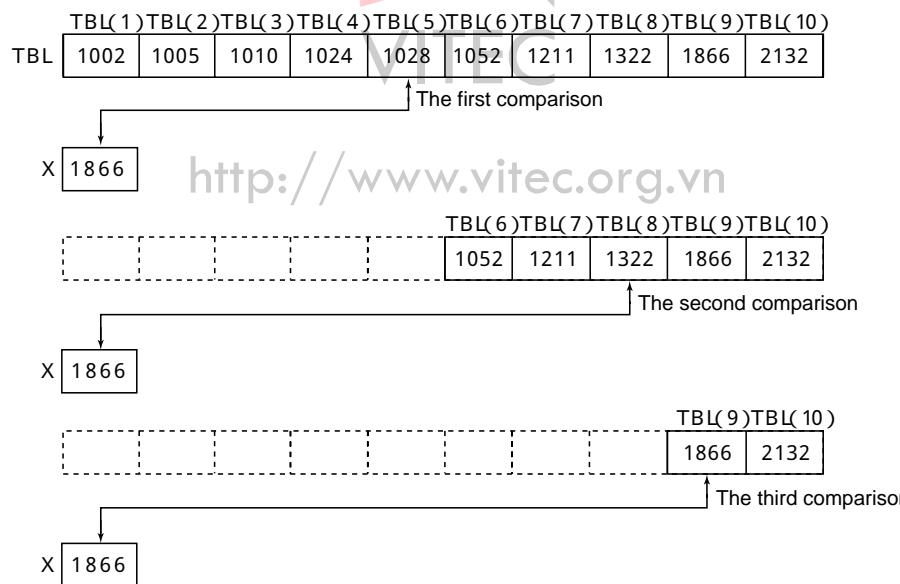
## (2) Binary search method

A binary search method is the method of narrowing down target data while successively dividing a search area into two parts. The number of comparisons can be greatly decreased compared with linear search method and the search efficiency can equally improve. This search method, however, requires that elements are arranged in ascending or descending order.

Figure 3-2-4 shows the algorithm used for the binary search method.

Figure 3-2-4 Algorithm for the binary search method

- Step 1: A total of the value of a subscript representing the top of a table and that of a subscript representing the end of a table is divided by 2.
- Step 2: The elements having the value obtained in step 1 as a subscript are compared with a target element.
- Step 3: If there is an element that matches a target element, the search is successful.
- Step 4: If the value of a target element is smaller than that of an element in a table, 1 is subtracted from the current subscript and the value is used as a subscript for representing the end of a table.  
If the value of a target element is larger than that of an element in a table, 1 is added to the current subscript and the value is used as a subscript for representing the top of a table.
- Step 5: Step 1 through step 4 is repeated. If an element matching a target element cannot be found at the point where the value of a subscript representing the top of a table becomes larger than that of a subscript representing the end of a table, the search is unsuccessful. This completes the search.



Because elements are arranged in ascending or descending order, data smaller than reference data does not need to be searched if data being searched is larger than reference data. Therefore, the number of data to search can be halved after the first search - a great advantage in the search efficiency.

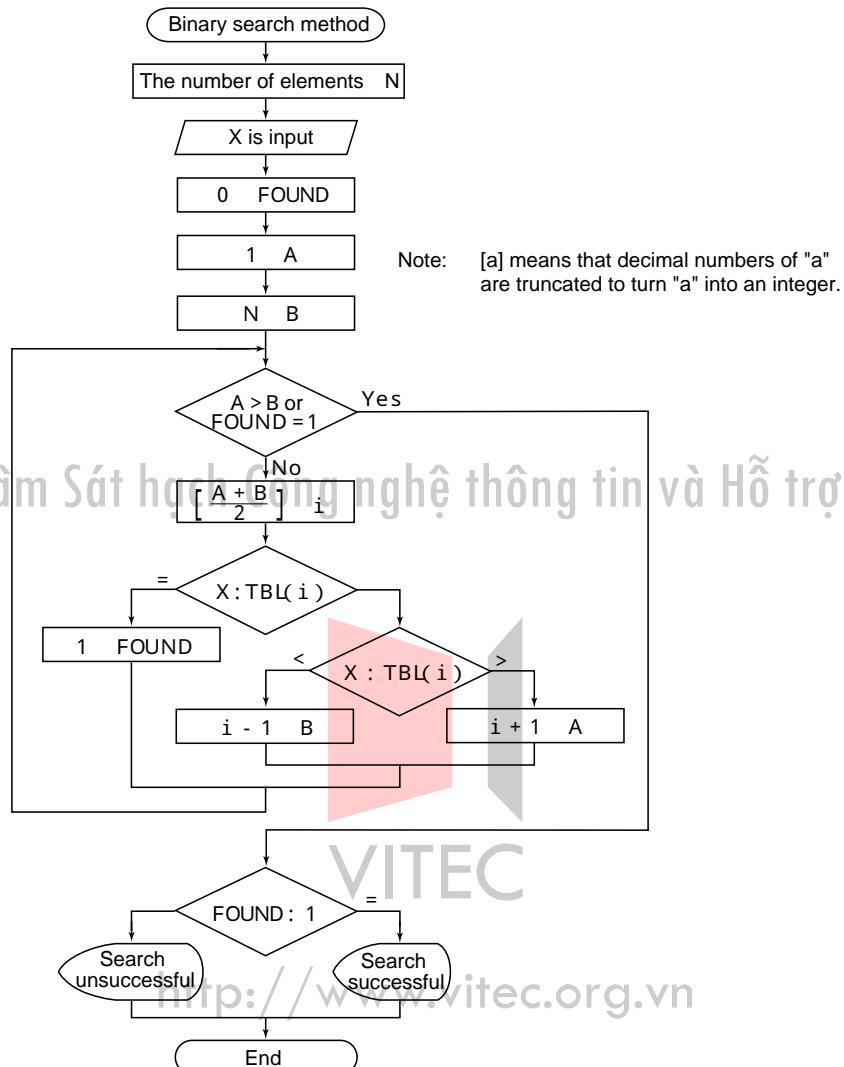
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Figure 3-2-5 shows a flowchart of the binary search method.

**Figure 3-2-5** Flowchart of the binary search method



<Maximum and average number of times a comparison is made: If the number of elements is N>

- Average number of times =  $\lceil \log_2 N \rceil$
- Maximum number of times =  $\lceil \log_2 N \rceil + 1$

([ ] is a Gaussian symbol and decimal numbers of the value shown in this symbol are truncated)

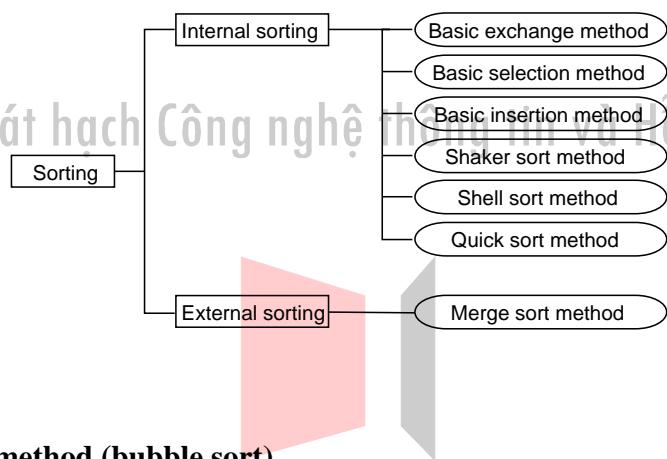
### 3.2.2 Sort algorithm

Sorting data is reorganizing data in a specified order. Sorting data in the order of small to large values is called ascending order sorting, and sorting data vice versa is called descending order sorting.

If data is sorted or organized in a certain specific order (amounts of sales, commodity codes, etc), the efficiency of the work of data processing can be increased. Therefore, the sort algorithm is one of the most commonly used. Using this sort algorithm, one can sort numbers and characters. This type of sorting is possible because characters are represented as character codes (internal codes) in a computer.

Figure 3-2-6 shows various sort methods.

Figure 3-2-6 Sort methods



#### (1) Basic exchange method (bubble sort)

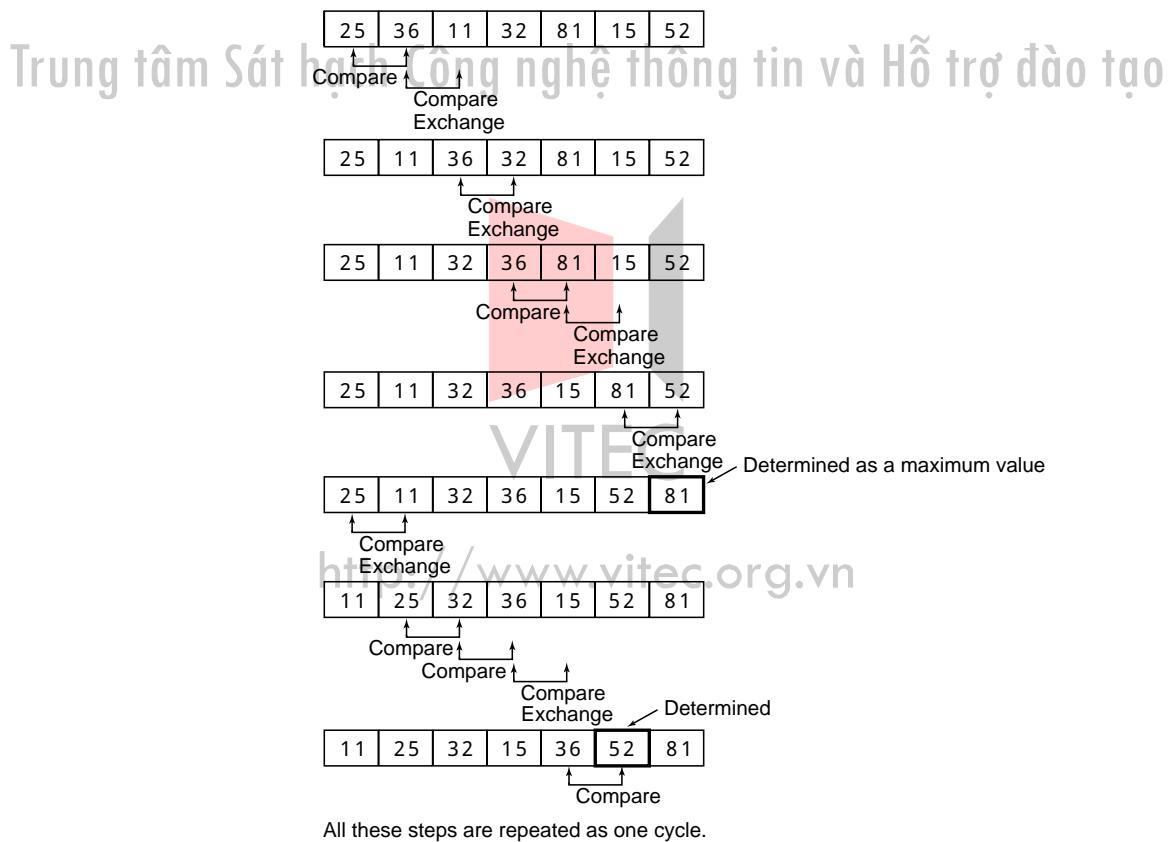
The basic exchange method (bubble sort) is used to compare a pair of data sequentially from the head of an array. If the sequence is wrong, data is exchanged. When all data items in one array are compared, the routine returns to the head of an array and it is repeated until there are no remaining data items to exchange.

This method is the simplest, best known sort method. The name "bubble sort" was given since the movement of maximum or minimum data looks like bubbles popping out on the surface of the water.

Figure 3-2-7 shows the algorithm of the basic exchange method.

Figure 3-2-7 Steps of the basic exchange method

Algorithm for sorting data in the ascending order  
Step 1: The first and second elements in a table are compared.  
Step 2: If the first element is larger than the second, the first is exchanged with the second.  
Step 3: If the second element is larger than the first, no exchange occurs.  
Step 4: The second and third elements are compared and steps 2 and 3 are repeated.  
Step 5: This routine operation is repeated to the last element in a table. As it reaches the last element, a maximum value is stored in the last element in that table.  
Step 6: Steps 1 through 4 and step 5 are executed until the operation reaches the last element but one.  
Step 7: Steps 1 through 6 are repeated until only the first and second elements in a table remain.  
This completes data sorting.



#### <Characteristics>

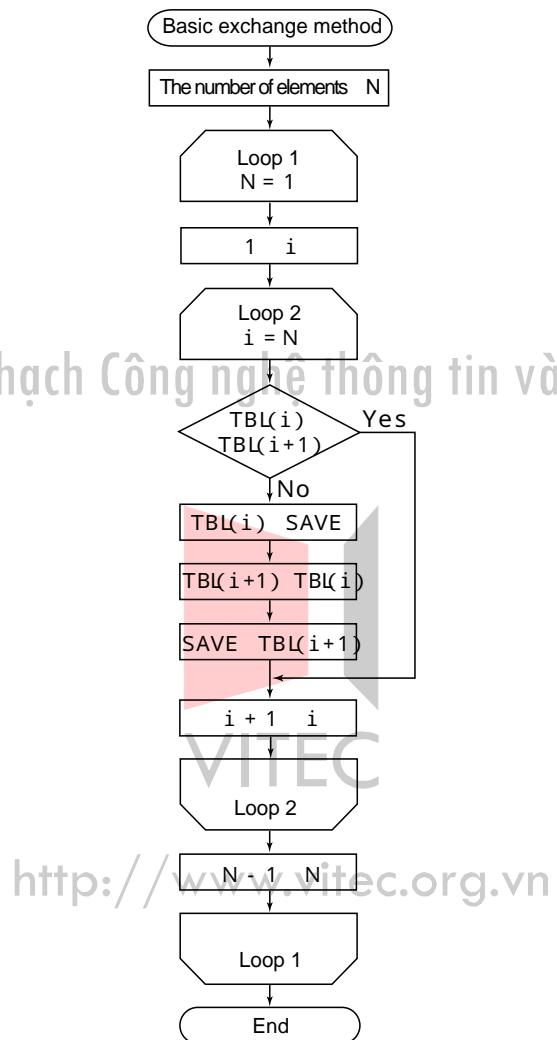
- One of the simplest sort method
- the efficiency is low since data item are unconditionally compared even if they are sorted correctly. If the volume of data is large, the process is time-consuming.

<Computational complexity>

Maximum computational complexity:  $O(n^2)$

Average computational complexity:  $O(n^2)$

Figure 3-2-8 Flowchart of the basic exchange method



## (2) Basic selection method

In the sort algorithm of the basic selection method, a data item with the smallest (or the largest) value is first selected from all the data items and it is exchanged with the data item at the head of an array, then the same routine is repeatedly executed on all the remaining data items. When the correct data item enters the last position but one, data sorting is completed.

Because this method allows a remaining element with the smallest or largest value to be selected, it is called the basic selection method.

Figure 3-2-9 shows the algorithm of the basic selection method.

**Figure 3-2-9** Steps of the basic selection method

Steps of the algorithm for sorting data in the ascending order  
Step 1: Data item with the smallest value is detected in data items stored in a table.  
Step 2: Data item with the smallest value detected is exchanged with the first data item in a table (space into which the data item with the smallest value can be temporarily saved is required).  
Step 3: Data item with the smallest value is detected in the second to the last data item in a table.  
Step 4: Data item with the smallest value detected is exchanged with the second data item in a table.  
Step 5: The same operation is repeatedly performed until the last data item but one is reached. When the last data item but one is reached, data sorting is completed.

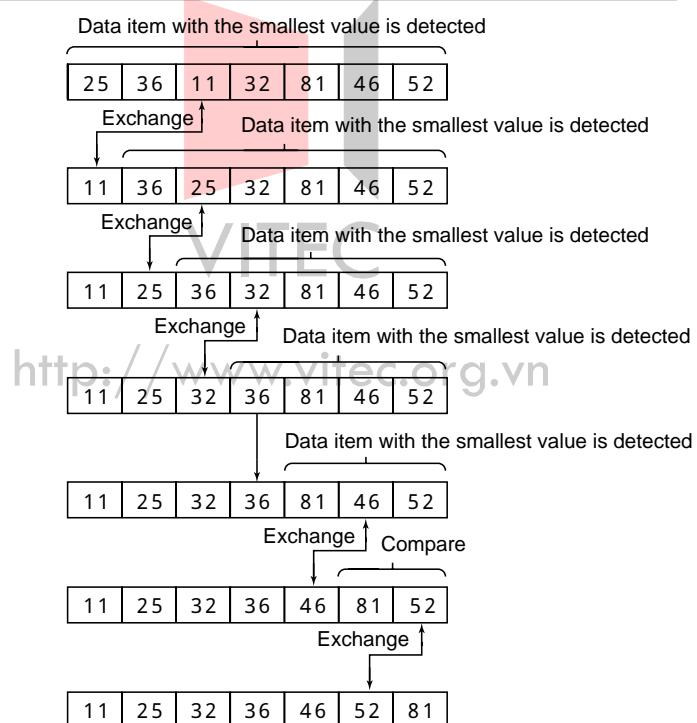
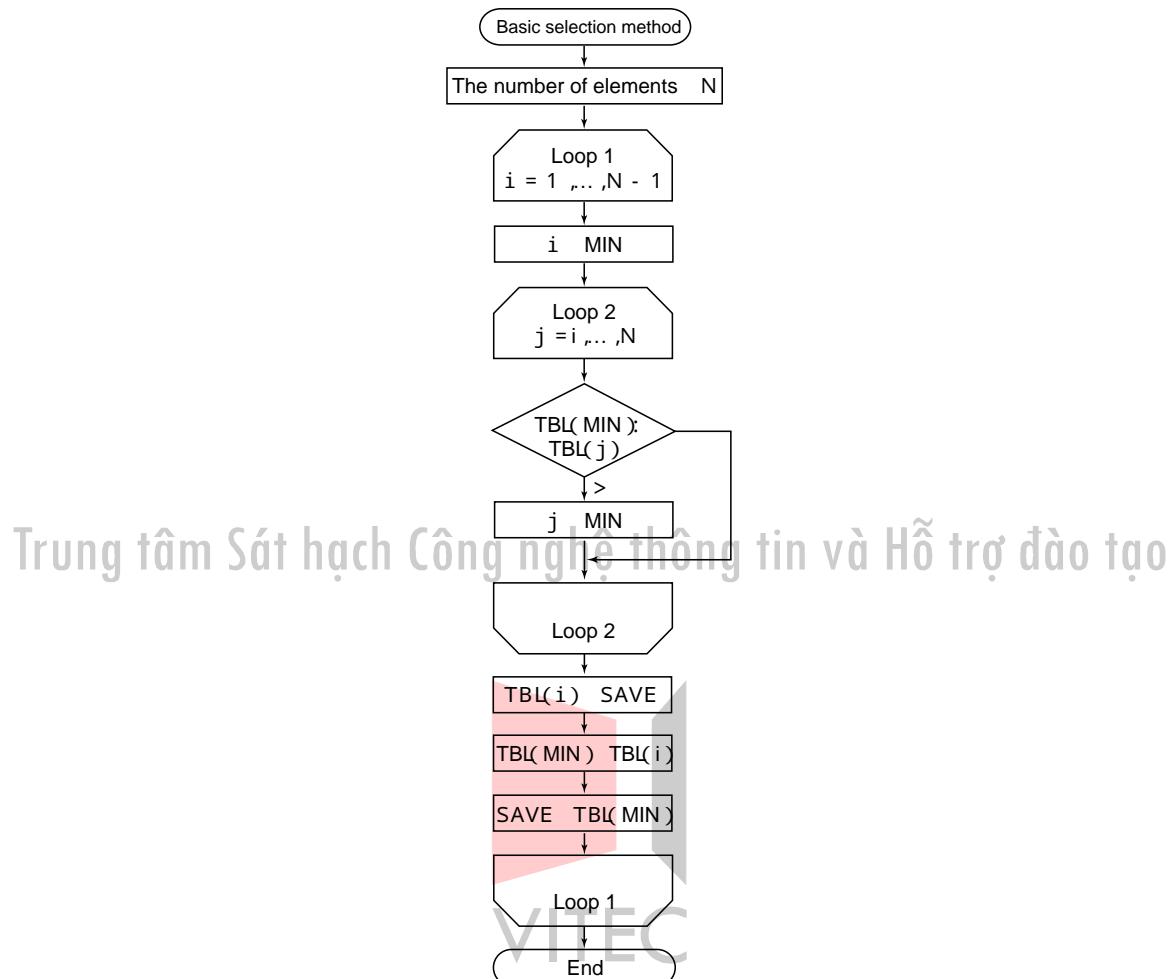


Figure 3-2-10 Flowchart of the basic selection method



### (3) Basic insertion method

In the algorithm of the basic insertion method, while data items are being sorted, an unsorted data item is inserted into a proper position in the sequence of sorted data items. Figure 3-2-11 shows the algorithm of the basic insertion method.

**Figure 3-2-11** Steps of the basic insertion method

Algorithm for sorting data in the ascending order

Step 1: The first and second elements in a table are compared.

Step 2: If the first element is smaller than the second, nothing is done.

Step 3: If the second element is smaller than the first, the first element is exchanged with the second. At this point, the first and second elements are in the correct order.

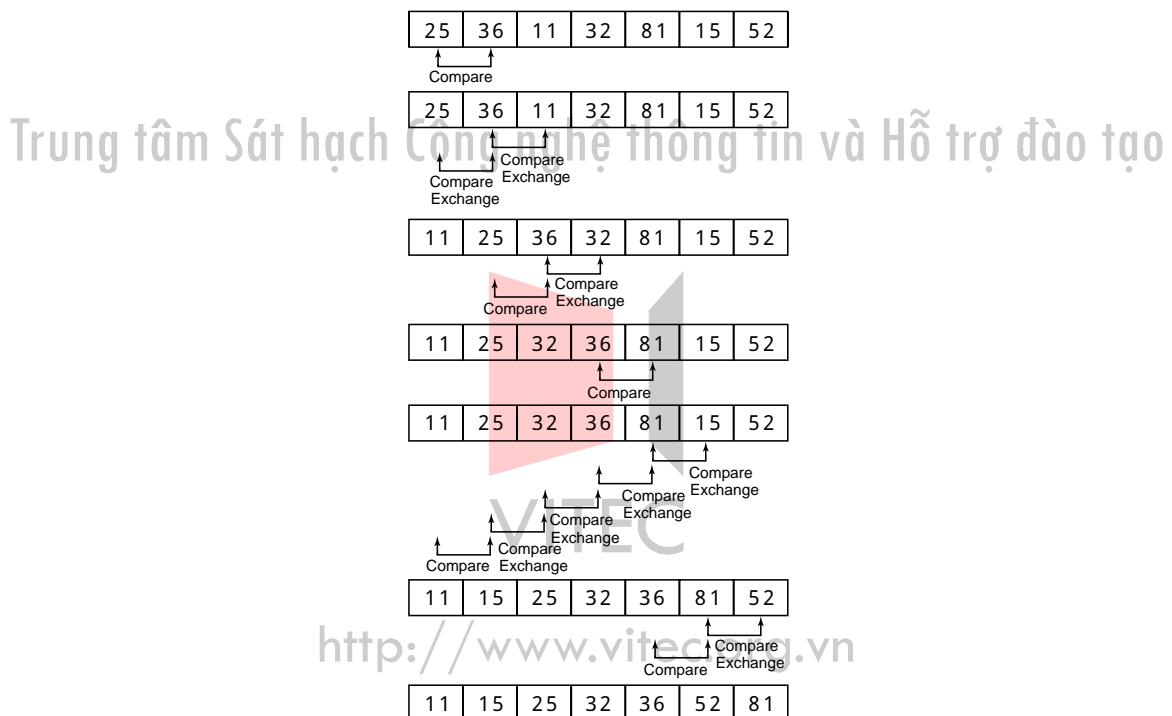
Step 4: The second and third elements are compared.

Step 5: If the second element is smaller than the third, nothing is done.

Step 6: If the third element is smaller than the second, the second element is exchanged with the third. Then this element is compared with the preceding element according to steps 2 and 3. These steps are repeated till it is placed in the right position.

Step 7: Steps 4, 5 and 6 are repeated until the last element in the table is inserted into the correct position.

This completes data sorting.

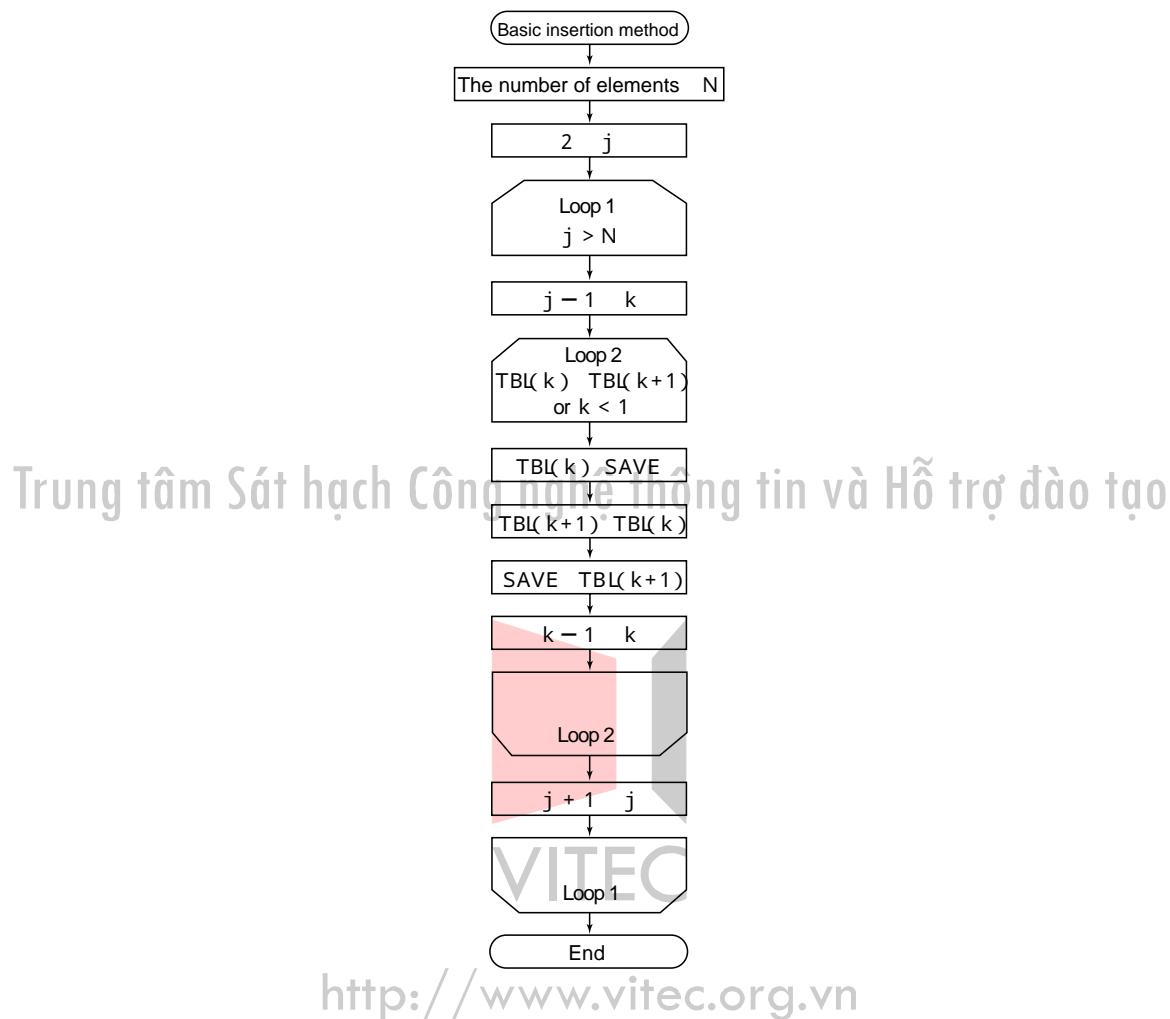


#### <Characteristics>

- One of the simplest sort methods, like the basic exchange method
- Because preceding data items have been sorted, comparison and insertion speeds are fast.
- If the volume of data is large, the process is time-consuming.

Figure 3-2-12 shows the flowchart of the basic insertion method.

Figure 3-2-12 Flowchart of the basic insertion method



#### (4) Shaker sort method

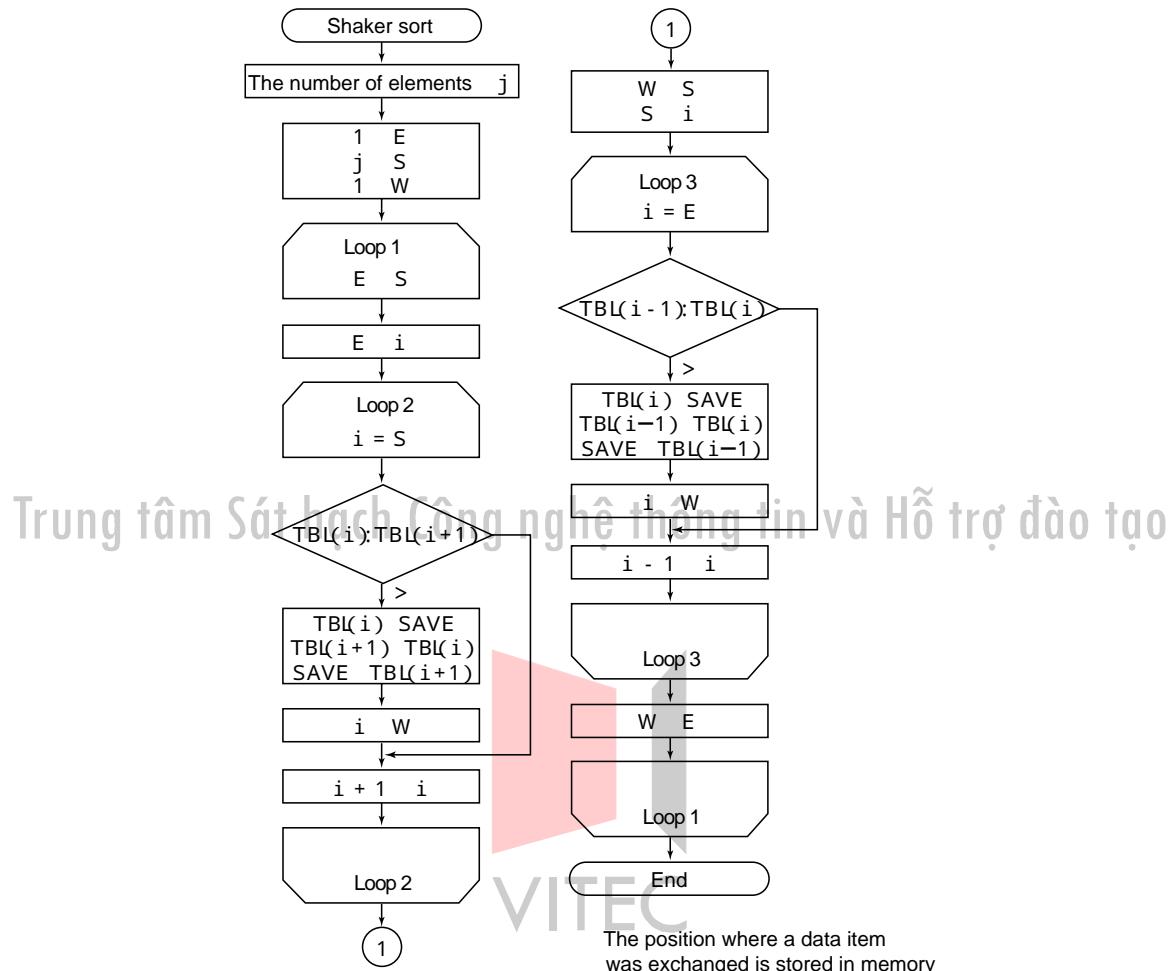
The algorithm of the Shaker sort method is basically the same as that of the basic exchange method (bubble sort). In the algorithm if the basic exchange method, data items are compared from left to right and are sorted in such a way that a maximum (minimum) value is set at the rightmost position. In the algorithm of the Shaker sort method, however, data items are first compared from left to right, then, after a maximum (minimum) value is set at the rightmost position, data items are compared from right to left and a minimum (maximum) value is set at the leftmost position; this operation is repeatedly performed to sort data.

<Characteristics>

- If the volume of data is large, the process is time-consuming.

Figure 3-2-13 shows the flowchart of the Shaker sort method.

Figure 3-2-13 Flowchart of the Shaker sort method



<http://www.vitec.org.vn>

## (5) Shell sort method

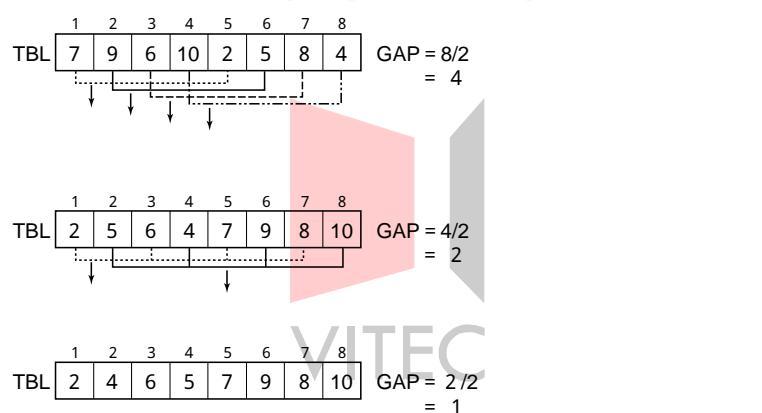
The Shell sort method is an extended version of the basic insert method. Two items of data located away from each other at a certain interval are picked out of a row of data and sorting is executed while the picked data items are compared with each other. The interval is called a gap. The gap is set large at the start of sorting; as sorting proceeds, it is gradually made smaller and finally set to 1. There are various ways of determining this gap. If the number of data is n, a simple way of determining the gap is  $[n/2]$ ,  $[n/4]$ , ...1.

When the gap is finally set to 1, sorting is executed in exactly the same way as the basic insertion method.

Using the basic insertion method, adjacent items are compared and exchanged and, therefore, the execution speed is slow. Using the Shell sort method, pieces of data that are away from each other and located in different positions are quickly exchanged so that data items sorted in wrong positions can be resorted to correct positions in the earliest stages of the sorting operation. As sorting proceeds, the gap between data items to be compared is narrowed.

Figure 3-2-14 shows the algorithm of the Shell sort method.

Figure 3-2-14 Steps of the Shell sort method

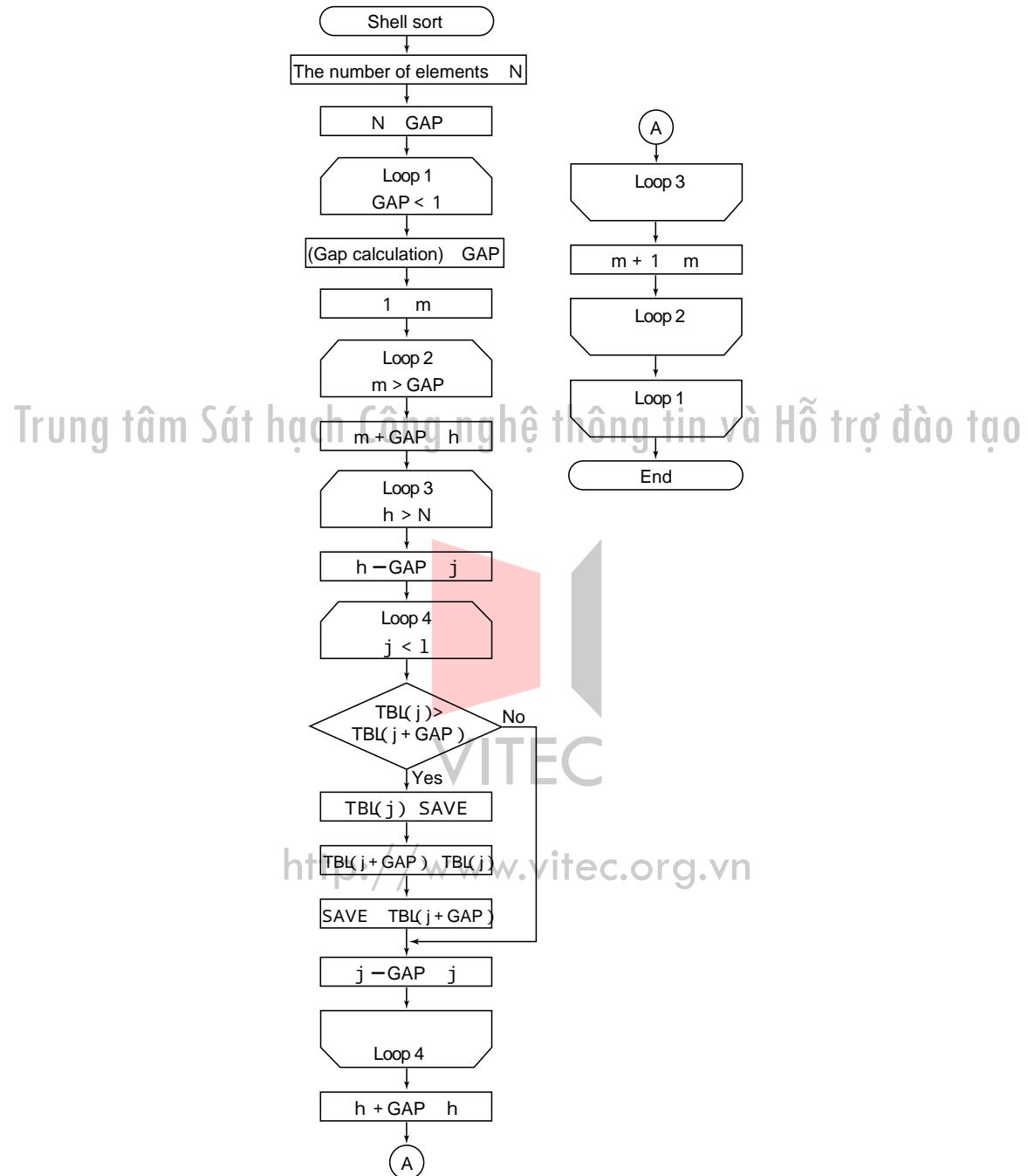


### <Characteristics>

- If part of the data is already sorted, sorting can be completed very quickly.
- A high-speed sorting method that uses the basic insertion method.

Figure 3-2-15 shows the algorithm of the Shell sort method.

Figure 3-2-15 Flowchart of the Shell sort method



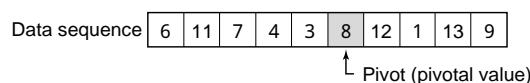
## (6) Quick sort method

The quick sort method was designed by Hoare. It is presently the fastest sort method using the recursive call method.

<To sort data in the ascending order>

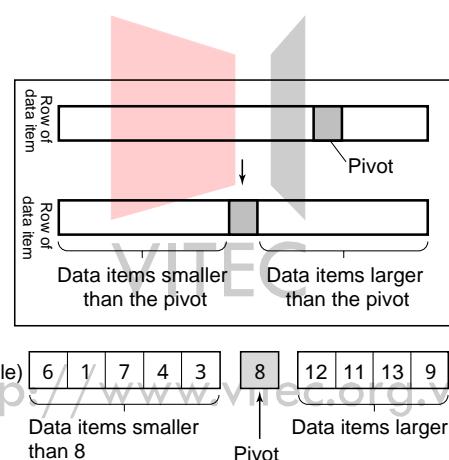
1. A reference value (pivot or pivotal value) is chosen from the data to be sorted. Although various values are used as reference values, a median or an intermediate value of three elements (right, center and left elements) is usually used. We use a median value in the example of sorting shown below.

Figure 3-2-16 Pivot (pivotal value)



- Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo
2. Data items smaller than the pivot are moved to the left of the pivot while data items larger than the pivot are moved to the right of it. All the data items are thus divided into two sections.

Figure 3-2-17 Moving data



3. A pivot is chosen from each section of data items so that the section is further divided into two sections.
4. This dividing operation is repeatedly performed until only one element remains.

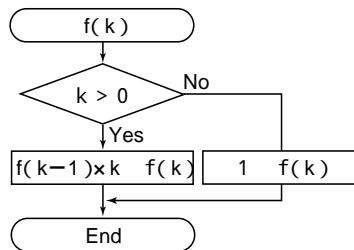
Figure 3-2-18 Data after sorting is completed

1	3	4	6	7	8	9	11	12	13
---	---	---	---	---	---	---	----	----	----

A method of dividing a large problem into small problems and solving each small problem individually, like the quick sort method, is called the divide-and-conquer method.

In performing step 3 and 4 above, the recursive call method of calling the routine itself is generally used. This recursive call method is often used to calculate factorials. (See Figure 3-2-19)

Figure 3-2-19 Algorithm for calculating factorials



With regard to computational complexity, if an ideal case in which a pivot that can divide data into two sections can always be chosen, the number of comparisons will be very close to  $O(n \log n)$ . If a maximum (minimum) value in a row of data is always chosen as a pivot, the number of comparison will become the worst,  $O(n^2)$ . Although computational complexity usually indicate maximum computational complexity, a case like this may happen in which average computational complexity becomes an important factor.

After data items are divided into two sections, data items in each section to be subjected to a recursive processing time is  $O(\log n)$  if parallel processing is executed.

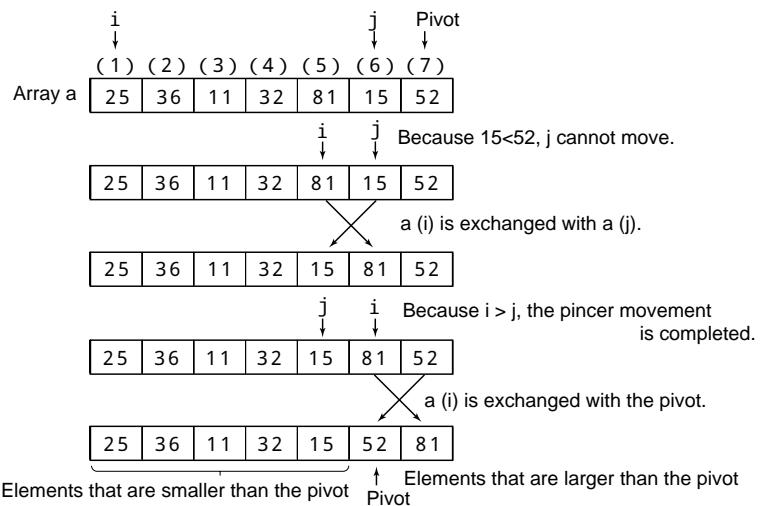
Figure 3-2-20 shows the algorithm for performing quick sorting with a pivotal set at the rightmost position of an array.

Specifically, two pointers are used and the routine proceeds from both ends to the center of a row of data. Pointer moving from the left end to the right is "i" and one moving from the right end to the left is "j".

<http://www.vitec.org.vn>

**Figure 3-2-20** Algorithm of the quick sort method

Step 1: As a preparatory step, pointers i and j must be initialized.  
 - i is at the left end, that is 1.  
 - j is at the last position but a (7) of the pivot, that is 6.  
 Step 2-1: i is moved to the right until an element larger than the pivot is detected.  
 Step 2-2: j is moved to the left until an element smaller than the pivot is detected.  
 Step 2-3: a (i) is exchanged with a (j).  
 Step 2-4: Steps 2-1, 2-2 and 2-3 are repeated until i = j.  
 Step 2-5: a (i) is exchanged with the pivot at the right end.



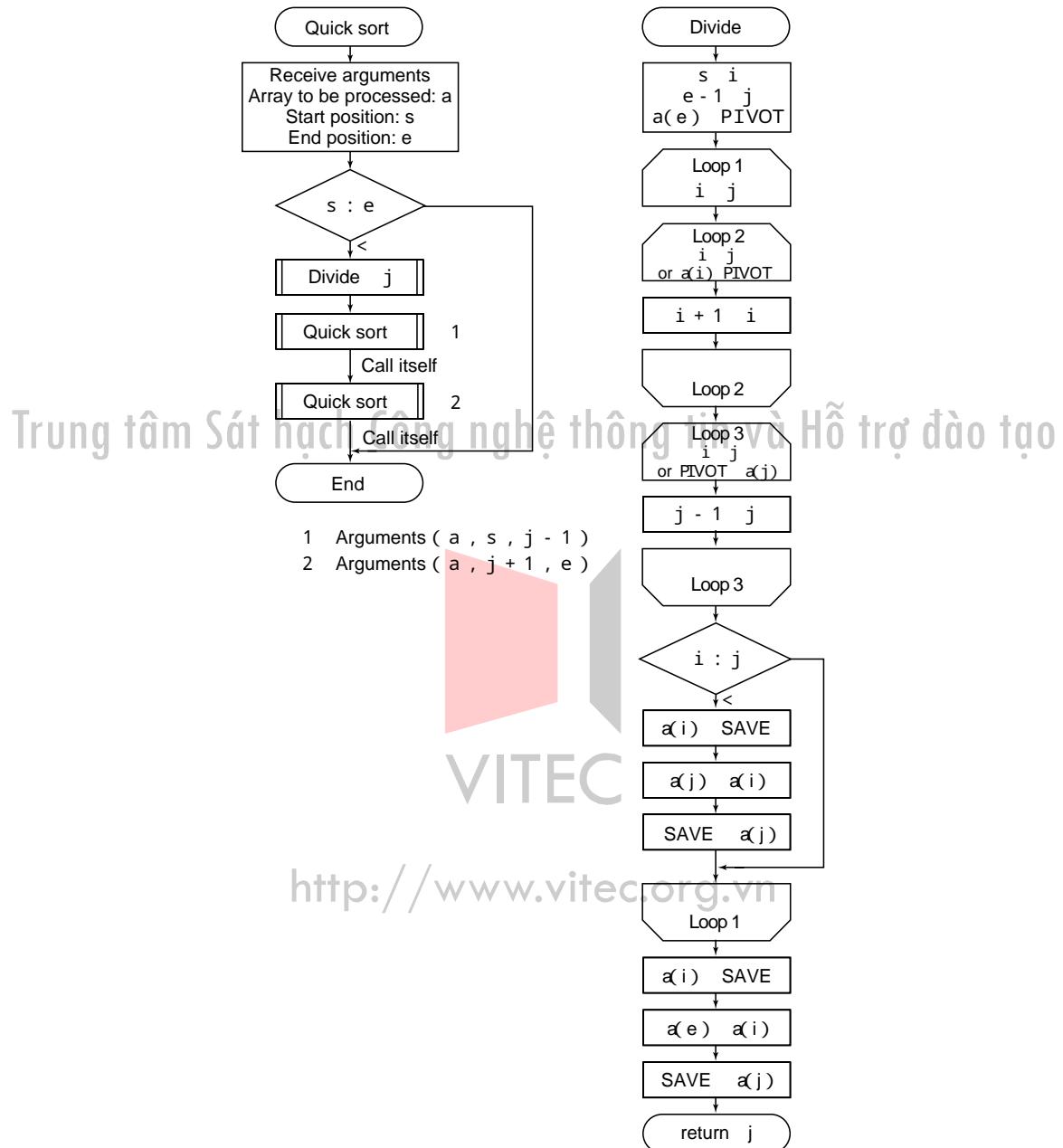
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Figure 3-2-21 shows the flowchart of the quick sort method.

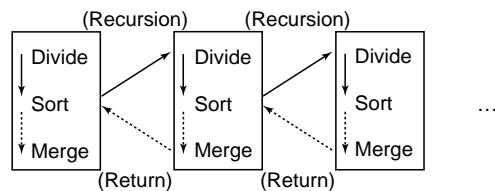
Figure 3-2-21 Algorithm of the quick sort method



## (7) Merge sort method

In the algorithm of the merge sort method, all the data items are divided into sections and sorting is executed in each divided section. The sorted sections are merged into a sorted sequence. (See Figure 3-2-22). Merge means comparing data items in two sorted sequences from the head and creating one sequence of sorted data by repeatedly taking out smaller data items in a sequential manner. If the number of data items is  $2n$ , repeating the merge by  $n$  times will complete sorting. If it is not  $2n$ , some adjustment is necessary.

Figure 3-2-22 Conceptual diagram of merge sort



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
<Characteristics>

- The recursive call and divide-and-conquer methods are used, as in the case of the quick sort method.
- Because sequences of data items are sequentially accessed and sorted, the merge sort algorithm is used for external sorting, for example, storing data on magnetic tapes.

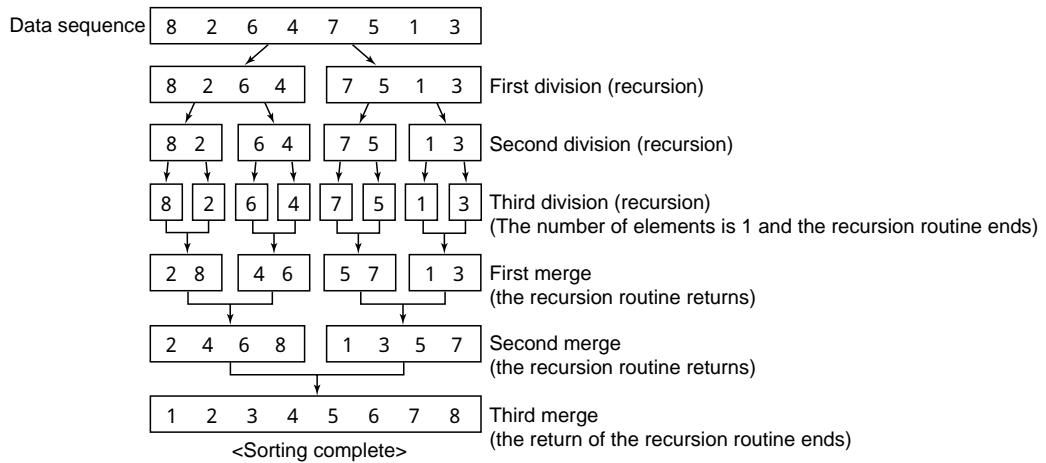
<Procedure>

- Data is divided into two sections and each divided section is further subdivided until there is only one remaining element in a sequence of data items.
- After a sequence of data items is divided, divided sections are sequentially merged.

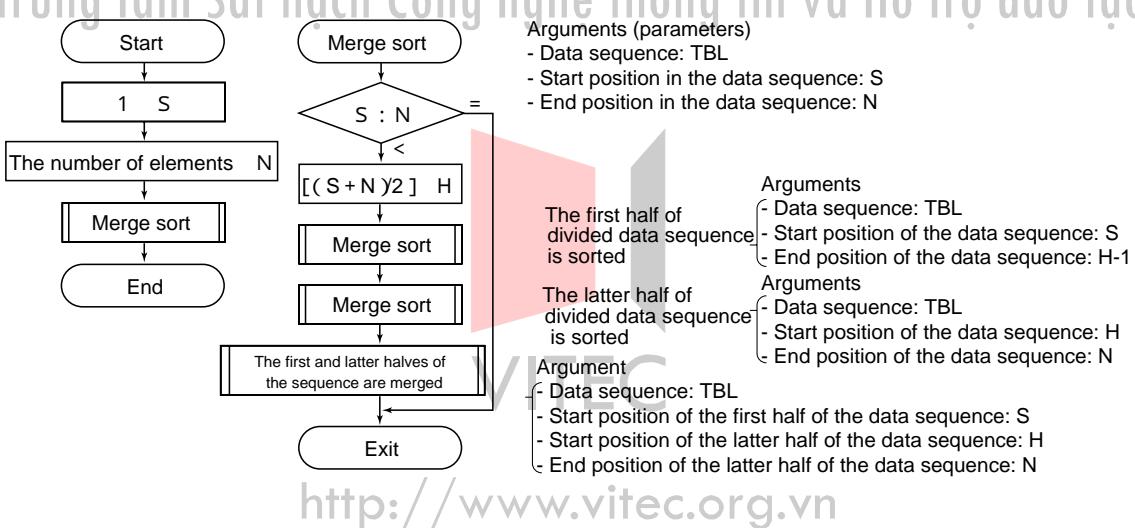
Figures 3-2-23 and 3-2-24 shows the status of data sequences during merge sort operations and the flowchart of the merge sort method.

<http://www.vitec.org.vn>

**Figure 3-2-23** Status of data sequences during merge sort operations



**Figure 3-2-24** Flowchart of the merge sort method



### 3.2.3 Recursive algorithm

A recursive call is calling a certain routine during data processing. The algorithm designed using a recursive call is the recursive call algorithm. The quick sort and merge algorithms are also recursive algorithms.

This section takes up the "eight-queen question" as an example for explaining how the recursive algorithm works.

#### (1) Eight-queen question

The eight-queen question is a question of how eight queens can be arranged on a chess board (8 x 8 grids) to prevent pieces from being taken by the opponent. A queen is a chess piece, and it can take a piece that is located on vertical, horizontal or oblique straight lines. In the answer,

therefore, queens must be positioned in such a way that they are not located on the same straight lines. Figure 3-2-25 shows one of the answers.

Figure 3-2-25 Example of an answer to the eight-queen question

	1	2	3	4	5	6	7	8
1	Q							
2							Q	
3					Q			
4								Q
5	Q							
6			Q					
7					Q			
8			Q					

Q : Queen

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

In solving the question, the following four arrangements are used:

q [i]: The position where a queen is placed in the i-th column (i=1 to 8)

In the above example, q = {1,2,3,4,5,6,7,8}

x [j]: To indicate whether or not they is a queen on the j-th row (j=1 to 8)

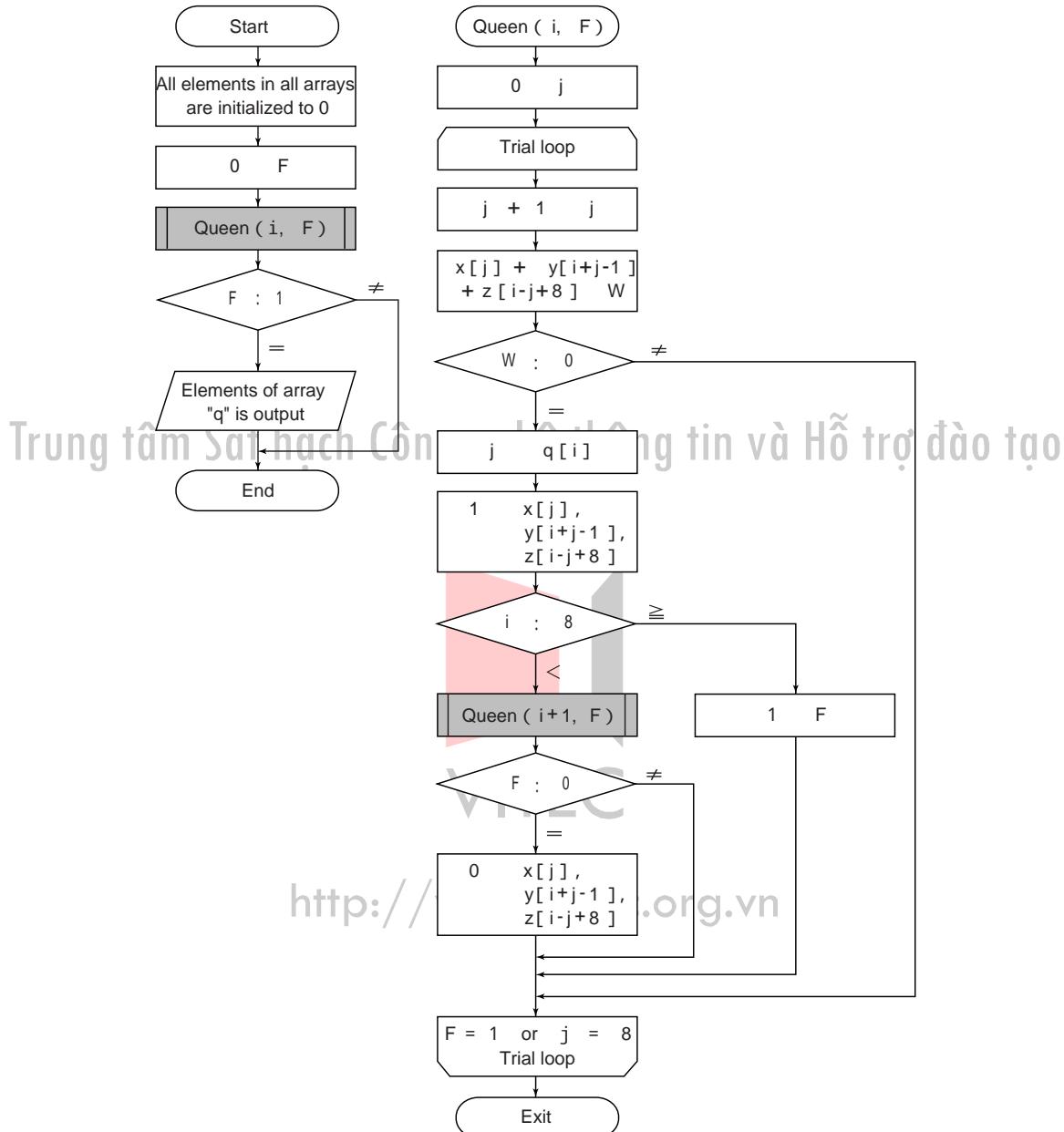
y [k]: To indicate whether or not they is a queen on the k-th left- to- right upward diagonal line.  
On the same left- to- right upward diagonal line,  $i + j$  is always the same. this can be used to obtain a subscript "k" with  $i+j+1$  ( $k=1$  to 15).

Z [l]: To indicate whether or not they is a queen on the l-th left- to- right downward diagonal line.  
On the same left- to- right downward diagonal line,  $i + j$  is always the same. this can be used to obtain a subscript "l" with  $i+j+8$  ( $k=1$  to 15).

\* For array x,y and z, subscripts 0 and 1 mean 'there is no queen' and 'there is a queen' respectively.

Figure 3-2-26 shows the flowchart of the algorithm for solving this question.

**Figure 3-2-26** Flowchart of the eight-queen question



### 3.2.4 Character string processing

Characters are searched, inserted, deleted, exchanged or compressed. This section describes character string search and compression.

#### (1) Character string search

To search character strings, the algorithm for searching a certain character string pattern and locating it in character strings is used.

① Simple collation

Text is compared character by character sequentially from the head. When a character string on which a search should be performed is detected, the position where it is located is set as a return value. In principle, this comparison is repeatedly executed until the first character of a character string to search matches a character in character strings in text. If there is a match, each remaining character of the matched character string is compared with each of a character string to search.

Figure 3-2-27 Image of the search

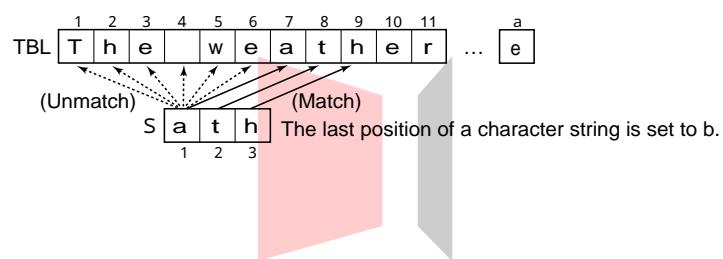
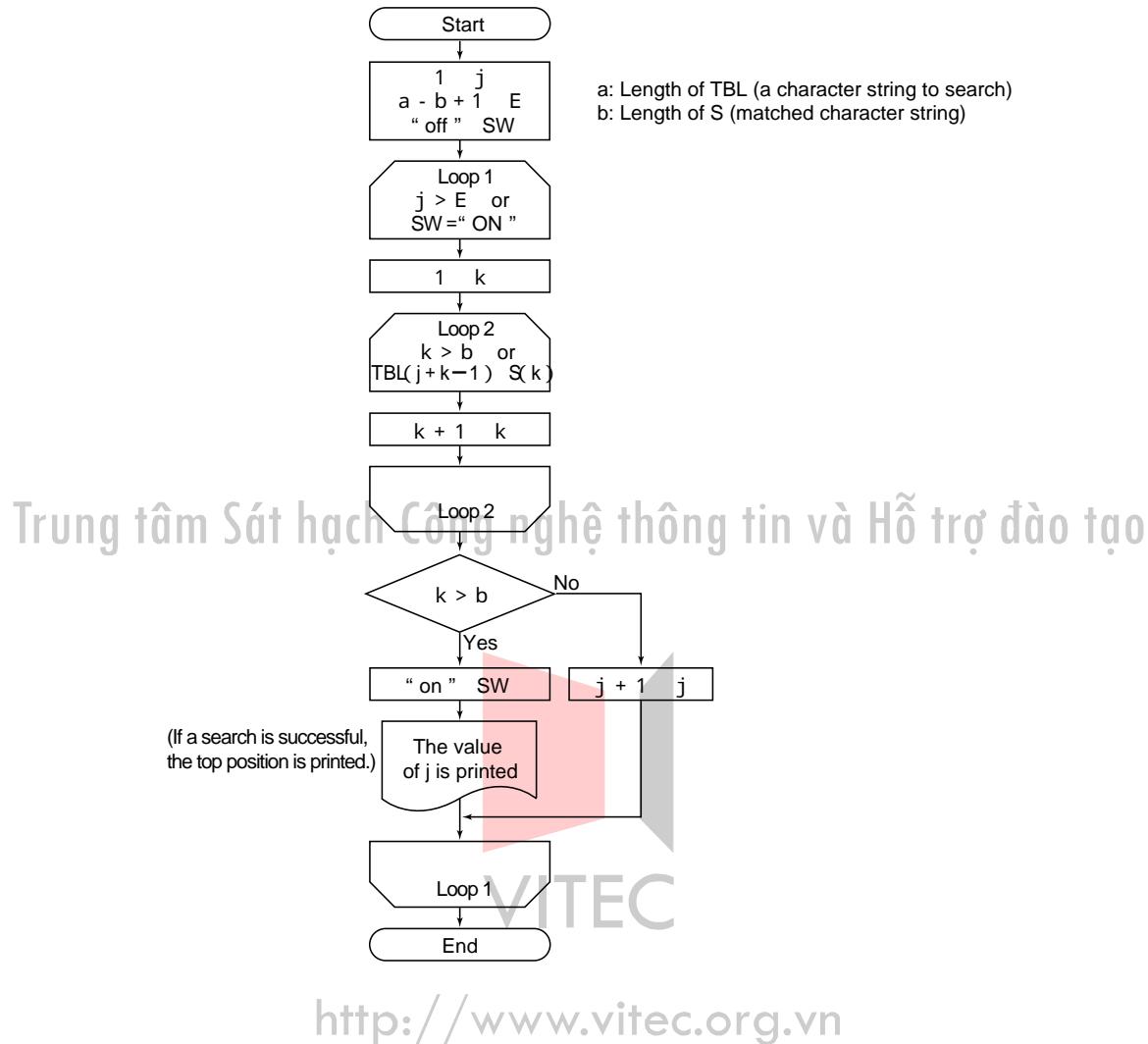


Figure 3-2-28 Flowchart of the simple collation



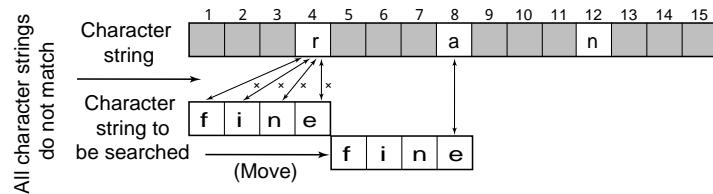
## ② Boyer-Moore method (BM method)

In the Boyer-Moore algorithm, data is collated while characters in the text are skipped. It has a complexity of  $O(m)$  where  $m$  is the length of the pattern .In this section, only the basic scheme of this algorithm is explained.

### a. If there is no character string to search

Figure 3-2-29 shows a comparison of the tail of a character string to search and text patterns.

**Figure 3-2-29** BM method (case 1)

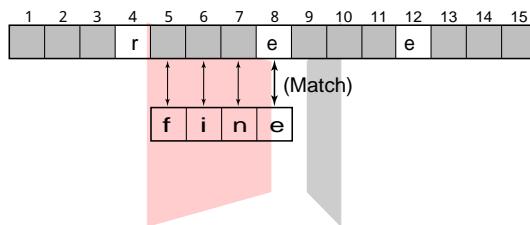


In this case, a character at the tail, and all other characters in the first text portion do not match any character of a character string to be searched. Therefore, a search point is moved by the length of a character string to be searched to allow the next search to start from that point.

b. If there is a match with a character at the tail of a character string to be searched

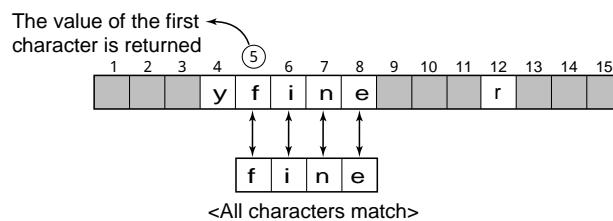
Figure 3-2-30 shows the case in which a character at the tail of a character string to be searched is compared with the text pattern, and a match is found.

**Figure 3-2-30** BM method (case 2)



Because a character at the tail of a character string matches a character in the text, characters that precede the matched character must be compared. If all characters match, the value of a subscript of the first character of that matched text pattern is returned.

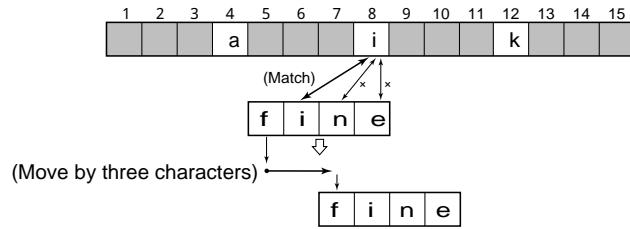
**Figure 3-2-31** Search successful



c. If there is a match with one character somewhere in a character string but unmatch with a character at the tail of a character string

In the case shown in Figure 3-2-32 a character string to be searched can simply be moved. What matters is the distance of movement.

**Figure 3-2-32** BM method (case 3)



The distance of movement is determined by how characters in a character string to be searched are arranged as shown in Figure 3-2-33.

**Figure 3-2-33** Distance of movement

Character	f	i	n	e	others
Distance of movement	3	2	1	0	4

By storing this distance of movement in an array, a character string to be searched can be moved to a correct position.

### ③Knuth-Morris-Pratt Algorithm

The matching of the pattern is done from left to right until it hits a mismatched. Information about the matched character(s) from the left of the text to the character preceding the mismatched character is utilized to shift the pattern to the right of the text by some distance.

The Knuth-Morris-Pratt (KMP) algorithm finds an occurrence and all occurrences of a pattern of length  $m$  within a text in with a complexity of  $O(m)$ .

<http://www.vitec.org.vn>

## (2) Character string compression

Making a character string short by replacing consecutive characters or blanks with the number of characters is called character string compression.

This section explains the algorithm for compressing blank characters (spaces).

In the case of the example shown in Figure 3-2-34, character strings are searched from the head, and if there are three consecutive blank characters, they are replaced with special characters (#) and the number of blank characters.

**Figure 3-2-34** Image of the character string compression

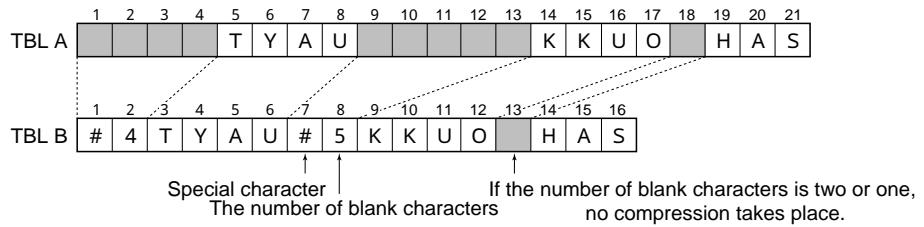
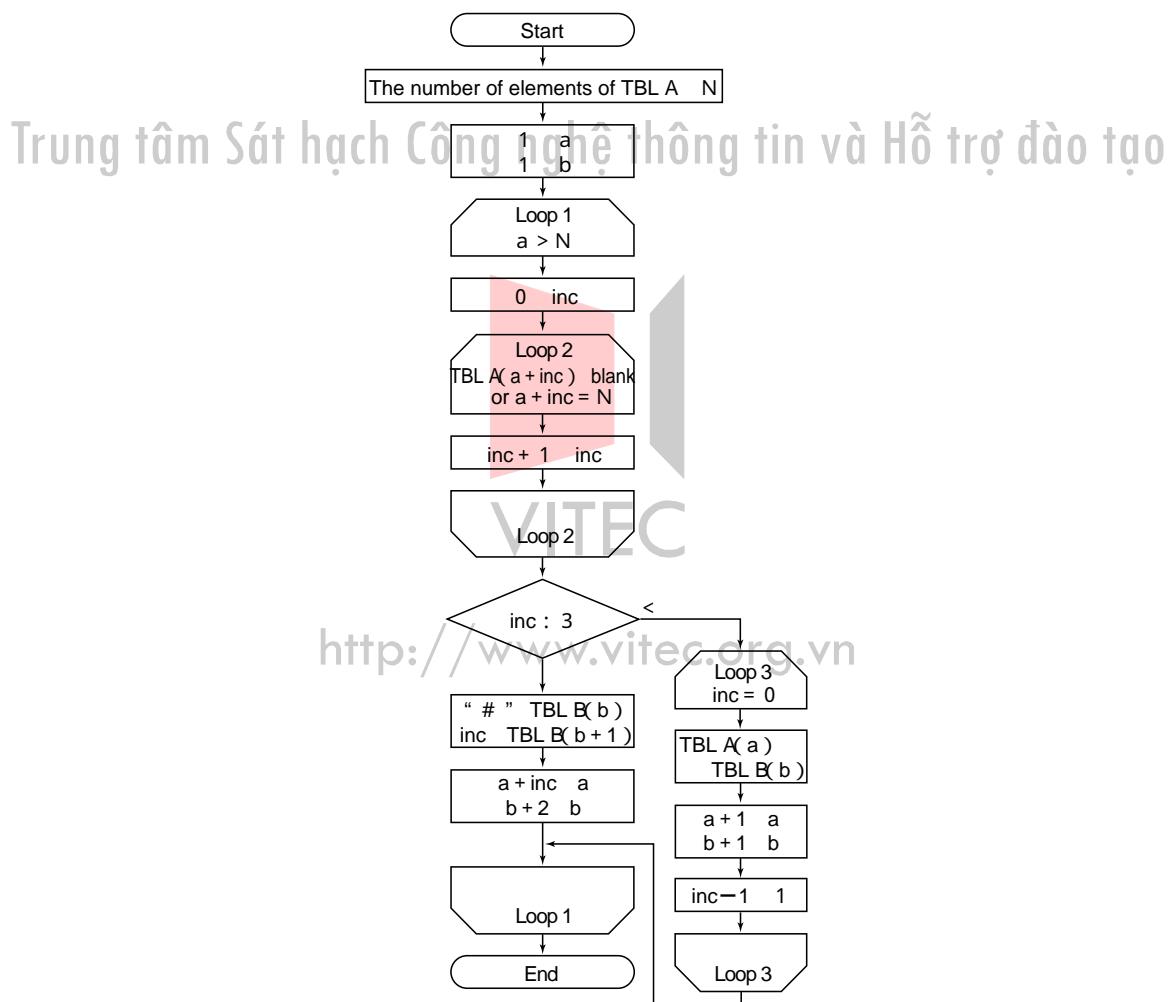


Figure 3-2-35 shows the flowchart of the character string compression.

**Figure 3-2-35** Flowchart of the character string compression



### 3.2.5 File Processing

Various files are used to perform paperwork tasks. The characteristic of file processing is processing each record one by one.

A typical file processing algorithm is as follows:

#### Example

Preparatory processing: Opening files, clearing counters, etc.

Main processing: Calculations, editing, outputting, etc.

End processing: Closing files, etc.

This section describes one algorithm for processing files of the same type and another algorithm for processing files of different types.

#### (1) Group control in processing files of the same type

Group control (group total) is to process records with the same key as one lump. In calculating a total of sales for each customer code (key=customer code) or an average mark for each class (key=class code), for example, group control is an indispensable processing routine.

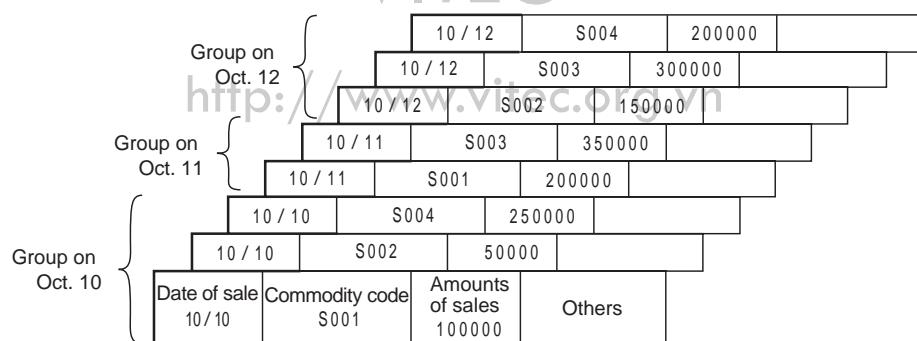
#### Example

Algorithm for reading sales files and printing a detailed listing of sales and total amounts of sales on each day

① Sales file layout

Figure 2-2-36 shows the layout of a sales file.

Figure 3-2-36 | Layout of a sales file



## ②Output format (a detailed listing of sales)

Figure 3-2-37 shows the format used to print a detailed listing of sales and total amounts of sales on each day.

Figure 3-2-37 Format used to output a detailed list of sales

Date of sale	Commodity code	Amounts of sales	
10 / 10	S001	100,000	Detailed listing on Oct. 10
10 / 10	S002	50,000	
10 / 10	S004	250,000	
Total amounts of sales		400,000	..... Group total on Oct. 10
10 / 11	S001	200,000	Detailed listing on Oct. 11
10 / 11	S003	350,000	
Total amounts of sales		550,000	
10 / 12	S002	150,000	Detailed listing on Oct. 12
10 / 12	S003	300,000	
10 / 12	S004	200,000	
Total amounts of sales			..... Group total on Oct. 12
Gross amounts of sales			..... Gross total of amounts on sales files

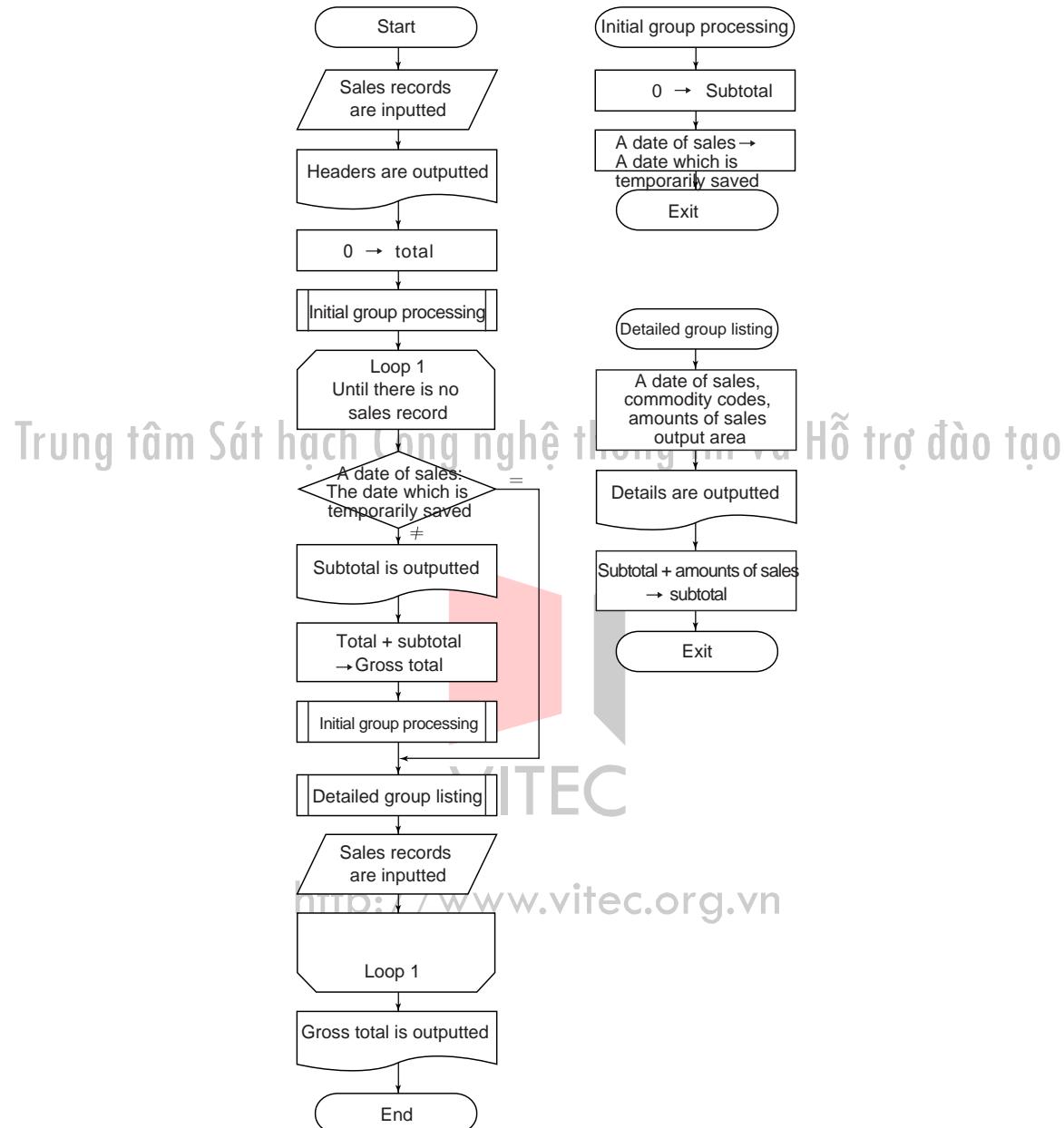
## ③Flowchart and the detailed scheme

To obtain group totals, a division between groups must be discerned. For this particular subject, a point where a date of sale changes becomes a division. Therefore, it is necessary to determine whether or not a date of sales in a newly loaded record matches that in the most recently processed records. To do this, a key (a date of sales) is temporarily saved before the first record in a group is processed, and it is compared with a key (a date of sales) of the past records.

<http://www.vitec.org.vn>

Figure 3-2-38 shows the flowchart of the group control.

Figure 3-2-38 Flowchart of the group control



## (2) Updating more than one file

If more than one file is compared using the same criteria by matching, all files must be stored in the sequence of keys, as in the case of the group control.

File processing tasks are as follows:

- Merging files
- Matching files
- Updating files
- Maintaining files

This section describes the task of file updating. File updating is to update master files based on data contained in transaction files. If a master file is a sequential file, it must be completely re-created. If a master file can be accessed randomly, a new master file does not need to be created since records can be retrieved using keys, and updated. Here, we deal with the updating procedure if a master file is a sequential file.

### ① 1:1 updating

1:1 updating means updating to be executed if each one record in a master file matches each one record in a transaction file. (See Figure 3-2-39)

Multiple files are read and processing routines are determined by comparing sizes of each key as follows:

Size of keys

- TR key = MS key Data is updated.
- TR key > MS key Data is copied (data is written into a new master file).
- TR key < MS key Data is added to a new master file (It may be the case where this status is considered an error and the error routine takes place).

VITEC

<http://www.vitec.org.vn>

Figure 3-2-39 Image of the 1:1 updating

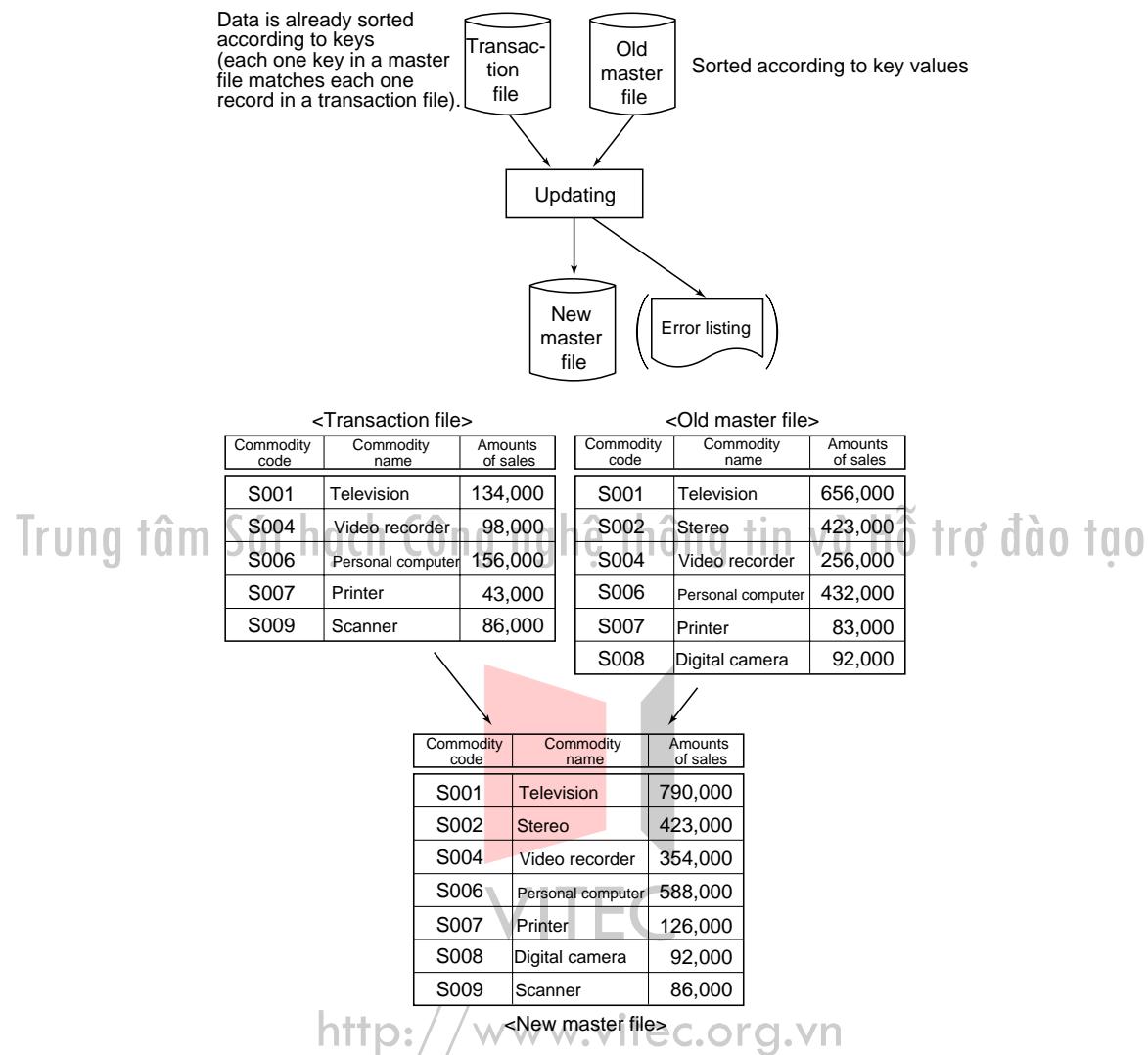
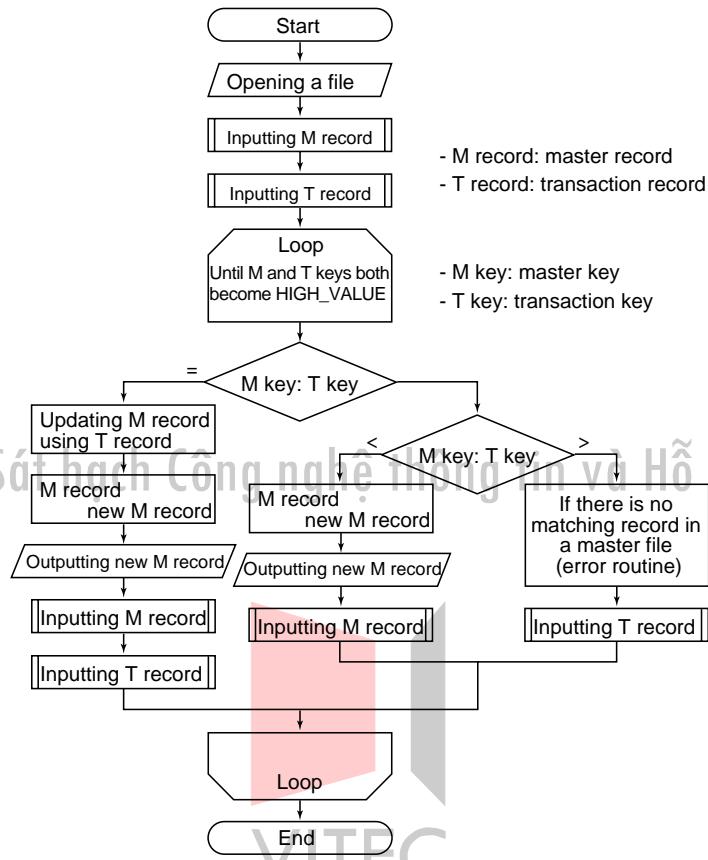


Figure 3-2-40 shows the flowchart of the 1:1 updating.

Figure 3-2-40 Flowchart of the 1:1 updating



<http://www.vitec.org.vn>

## ② 1:n updating

1:n updating is the updating routine used if one record in a master file matches more than one record in a transaction file. (See Figure 3-2-41)

In principle, multiple files are read, as in the case of the 1:1 updating, and processing routines are determined by comparing sizes of keys as follows:

### Sizes of keys

- TR key = MS key Data is totaled and the results are stored in a work area.
- TR key > MS key Data is updated.
- TR key < MS key Data is added to a new master file (It may be the case where this is considered an error and the error routine takes place)

Figure 3-2-41 Image of the 1:n updating

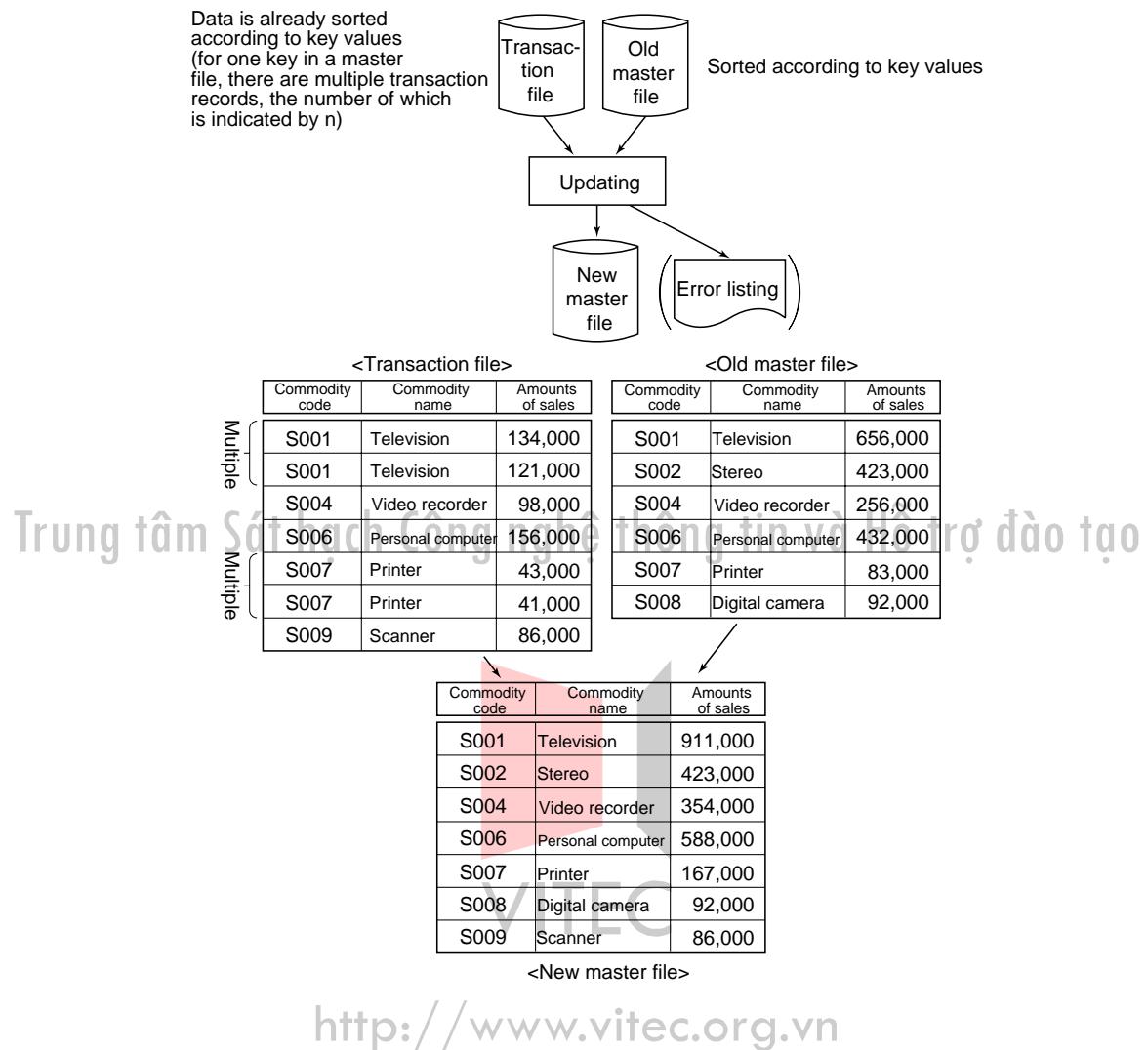
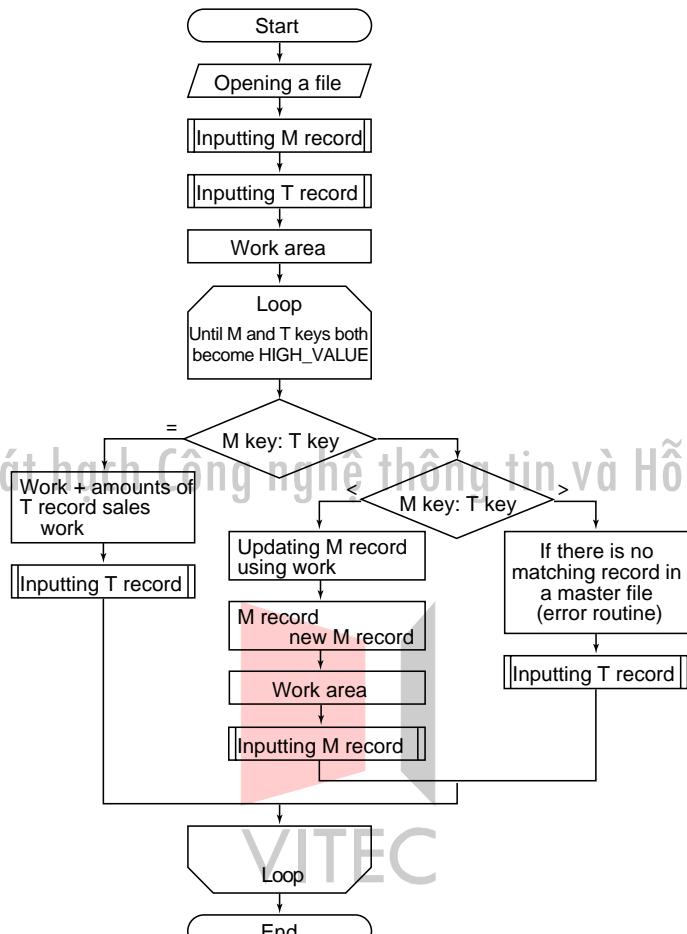


Figure 3-2-42 shows the flowchart of the 1:n updating.

Figure 3-2-42 Flowchart of the 1:n updating

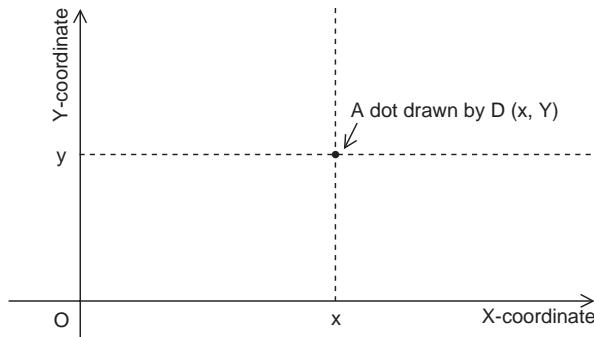


### 3.2.6 Drawing figures

In the present world of computers, CAD, CG and other figure drawing technologies are used. In an algorithm for drawing figures, a figure is treated as a set of dots. How a simple straight line and a circle can be drawn is explained in this section.

A function D (x,y) is used to draw a dot at an intersection point where a line along the x-coordinate meets a line along the y-coordinate, as shown in Figure 3-2-43.

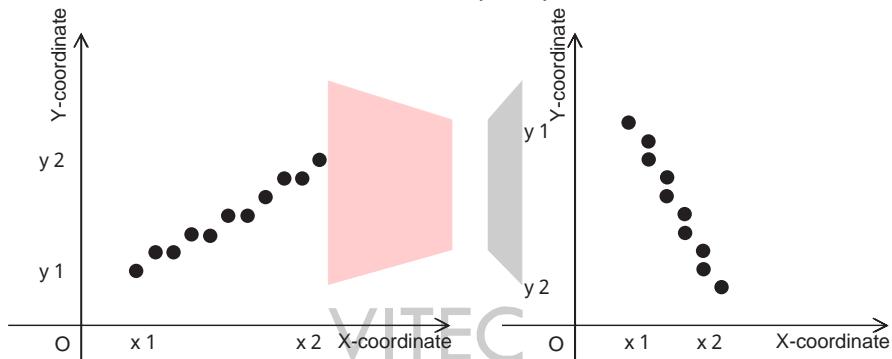
**Figure 3-2-43** How the function D (x, y) works



### (1) Drawing a straight line

Figure 3-2-44 shows the routine of straight line drawing.

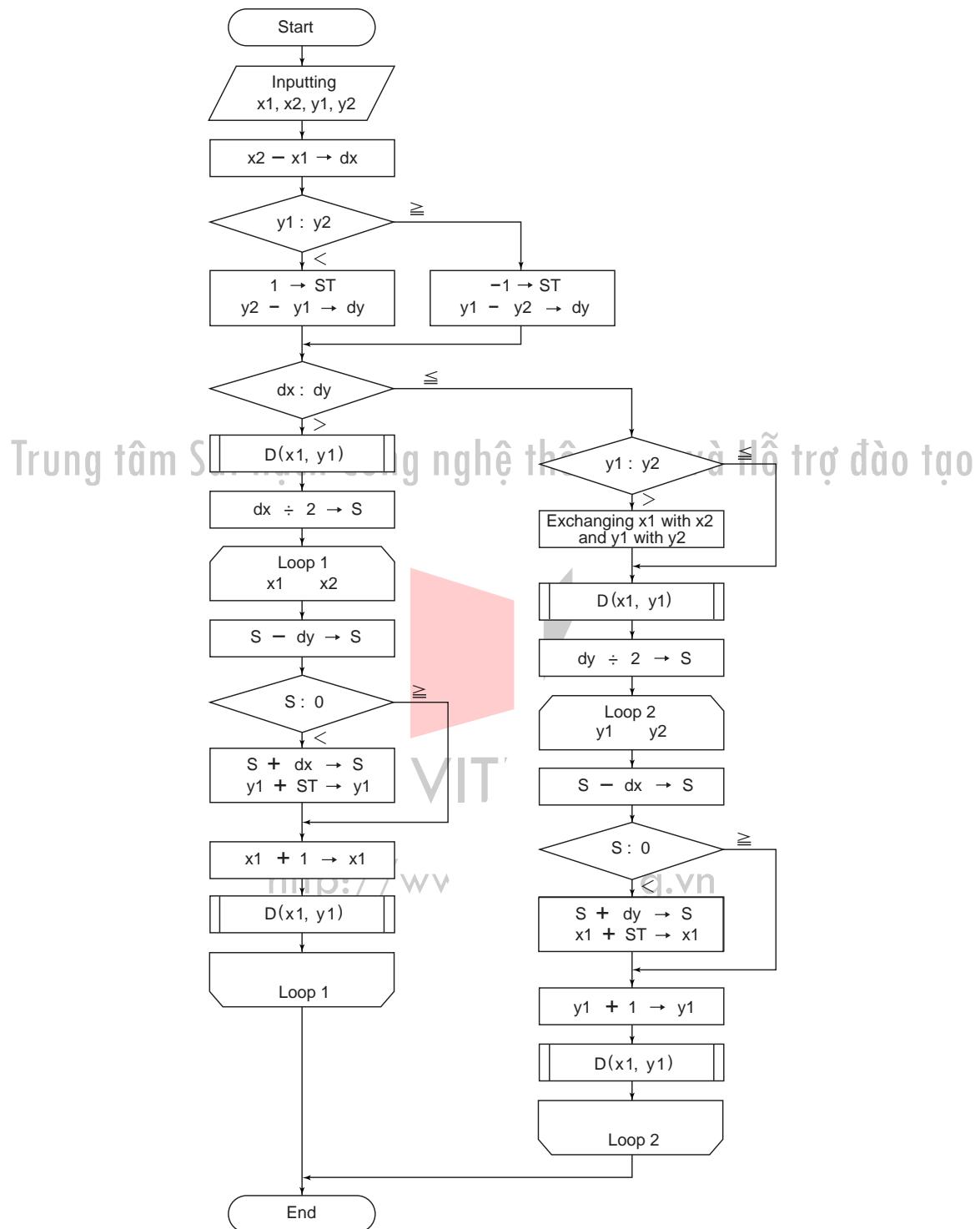
**Figure 3-2-44** Examples of straight line drawing  
[  $y_1 < y_2$  ]      [  $y_1 > y_2$  ]



<http://www.vitec.org.vn>

The flowchart shown in Figure 3-2-45 is an algorithm for drawing a straight line that connects two given points in the coordinates, i.e.  $(x_1, y_1), (x_2, y_2) \mid 0 < x_1 \leq x_2, 0 < y_1, 0 < y_2 \mid$ . Although this algorithm can draw straight lines running obliquely in upper or lower right directions by drawing dots on an integer coordinate, it is designed to allow lines to look like straight lines. Also, multiply-divide operations are kept to a minimum and add-subtract operations are mostly used to increase the processing speed.

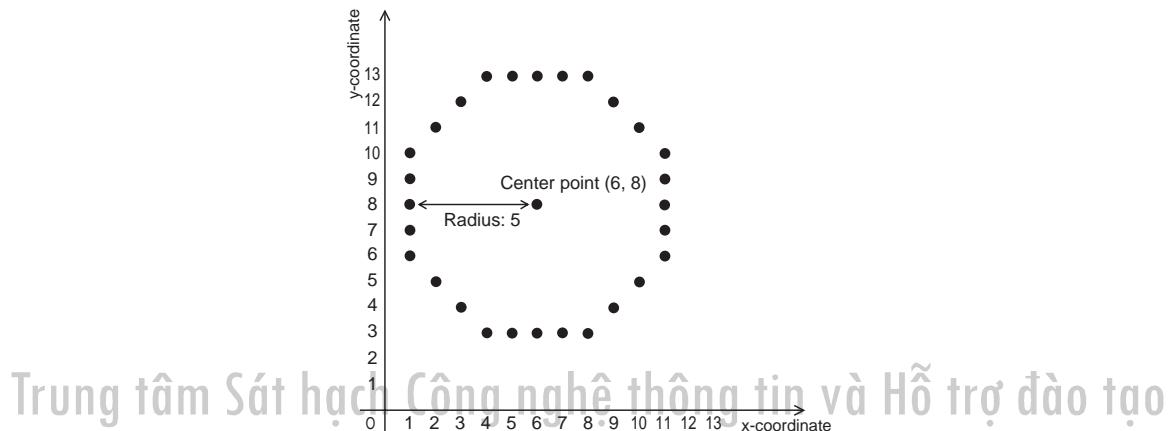
Figure 3-2-45 Flowchart of straight line drawing



## (2) Drawing a circle

Figure 3-2-46 shows how a circle is drawn.

Figure 3-2-46 How a circle is drawn

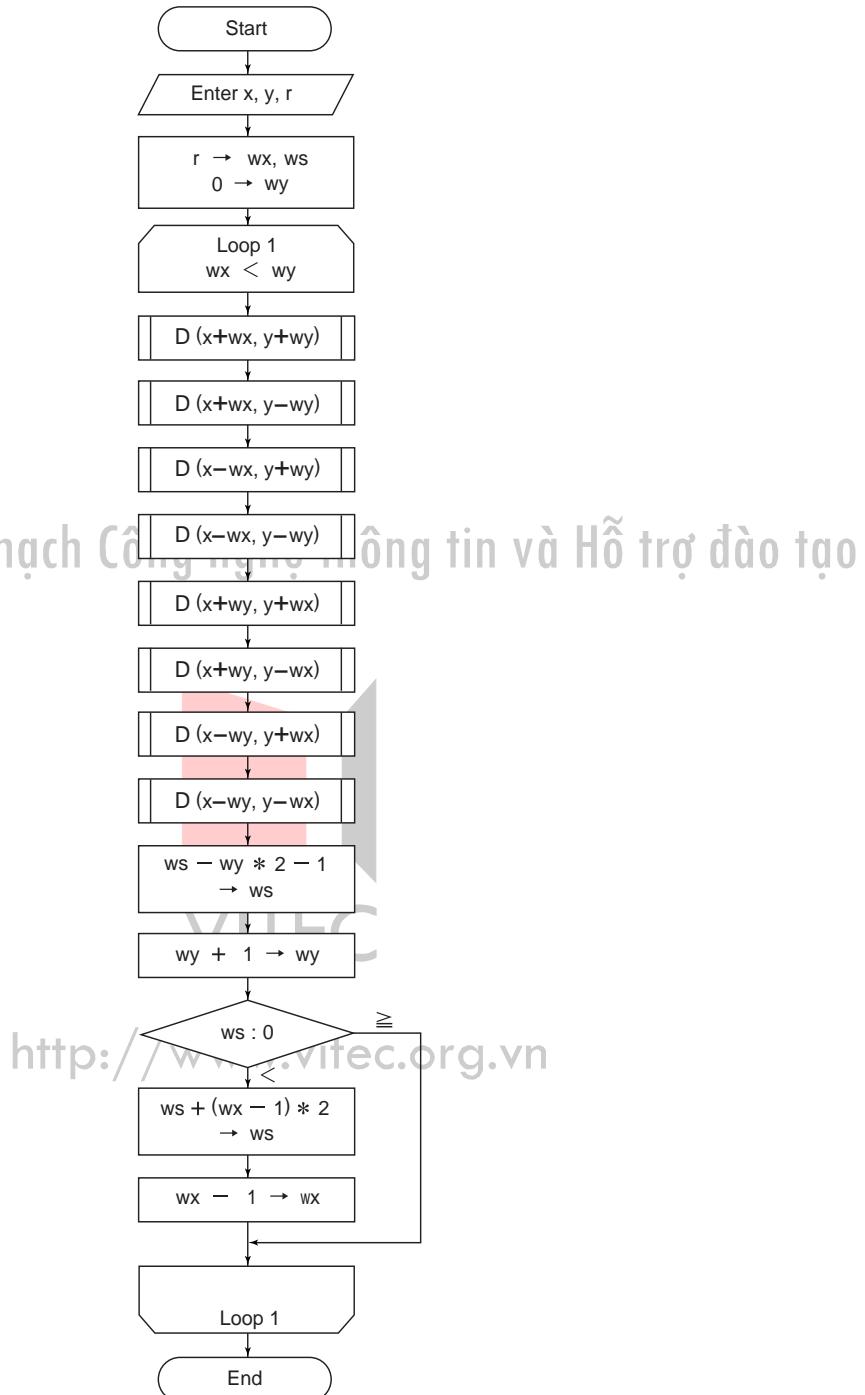


The flowchart in Figure 3-2-47 shows the algorithm for drawing a circle based on the coordinate  $(x, y)$  of a given center point and the radius  $r$ . To draw this circle, a trigonometric function that needs a long execution time is not used.

VITEC

<http://www.vitec.org.vn>

Figure 3-2-47 Flowchart for drawing a circle



### 3.2.7 Graph

The graph that we discuss in this section is one made up of arcs formed by connecting more than

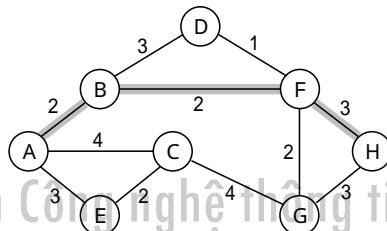
one point. Basically an undirected graph is assumed. A directed graph can also be used, depending on the type of problem to be solved.

The shortest path problem is described here as one of the representative graph problems.

### (1) Shortest path problem

As shown in Figure 3-2-48, there are some routes and nodes, and the shortest route that runs from point A to point H must be found. Because the numbers along each route show a distance, you can see at once that the thick line is the shortest route.

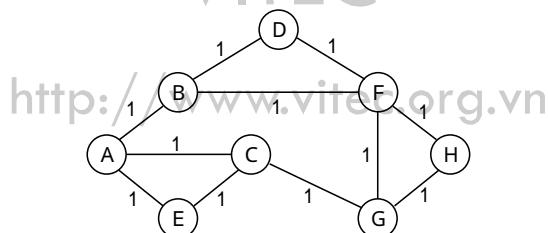
[Figure 3-2-48] Map (the thick line is the shortest route)



The graph shown in Figure 3-2-48 that has numbers along each route to show a route distance is called a weighted graph. Although we can discern the shortest route by sight, an algorithm must be used to enable a computer to find the shortest route through calculations.

We will here describe three route search methods: the depth-first and the breadth first search method which assume that the distance of each route is 1, and the Dijkstra's search method which is a major method for finding the shortest route.

[Figure 3-2-49] Graph in which the distance of all routes is set to 1



#### ① Depth-first search method

Using the depth-first search method, one route is chosen as a start point and a search starts to find a route to a destination using the stack.

<Procedure> (Figure 3-2-50)

1. A node at the start point is put into the stack.
2. A node is picked from the stack. Nodes adjacent to a picked node are checked and those that have never before being pushed into the stack are chosen. The chosen nodes are pushed into the

stack.

3. When the chosen nodes are pushed into the stack, the node picked in step 2 above is stored in the array.

4. Steps 1, 2 and 3 are repeatedly executed until a target node is put into the stack or the stack becomes empty.

Figure 3-2-50 | The state of the stack

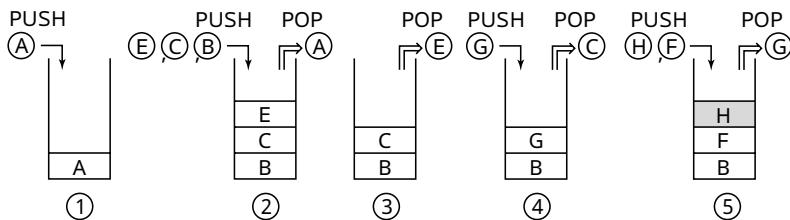


Figure 3-2-51 | The state of the array



Before the chosen nodes are pushed into the stack, the node picked in step 2 above must be stored in the array. For example, if **(A)** is picked and **(B)**, **(C)** and **(E)** are pushed into the stack, A must be stored in the elements of B, C and E. After a target node has been put into the stack, nodes are traced one by one sequentially to find the shortest route. (The same routine is executed in the case shown in Figure 2-2-53.)

#### <Results>

A search executed with depth-first search results in the route A-C-G-H shown in figure 3-2-51. Although the above Figures show the basic scheme of the search execution routine, what happens during an actual search is more complex. That is, assuming that the shortest route cannot be found, different routes are repeatedly searched, that is, as the search routine reaches the step before the step of putting H, shown in Figure 3-2-50, step ⑤ into the stack, it leaps back to step 1 and repeats all the steps.

This routine is repeated until the stack becomes empty so that all routes are counted to find the shortest route.

#### ② Breadth-first search method

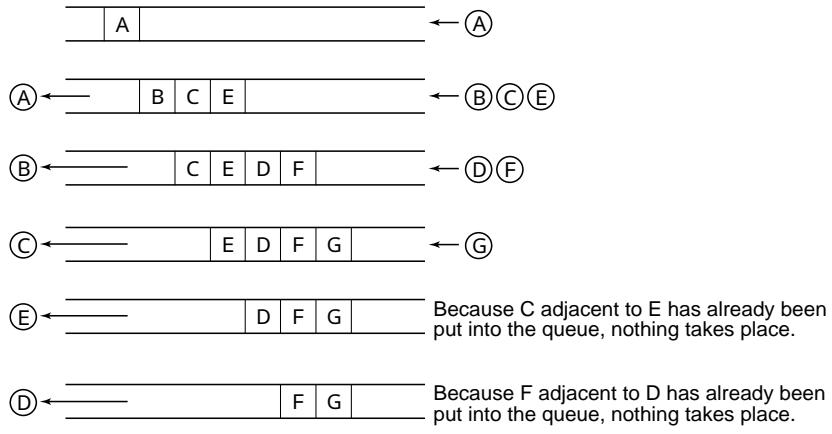
Using the breadth-first search method, all possible routes that can lead to a destination are searched using the queue.

#### <Procedure> (Figure 3-2-52)

1. A node at the start point is put into the queue.
2. One node is picked from the queue. Nodes adjacent to the picked node are checked and those that have never before being put into the queue are chosen. The chosen nodes are put into the queue.
3. The node picked in step 2 is stored in the array.
4. Above steps 1, 2 and 3 are repeatedly executed until a target node is reached or the queue

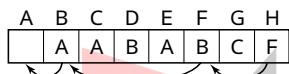
becomes empty.

**Figure 3-2-52** The state of the queue



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

**Figure 3-2-53** The state of the array



Before a node is put into the queue, the node picked in step 2 is stored in the array. For example, if A is picked and B, C and E are put into the queue, ① is stored in the elements of ②, ③ and ④.

A search executed using the breadth-first search results in the shortest route A-B-F-H shown in Figure 3-2-53. All nodes are searched one by one, as shown above, while calculations are being made to find the best route.

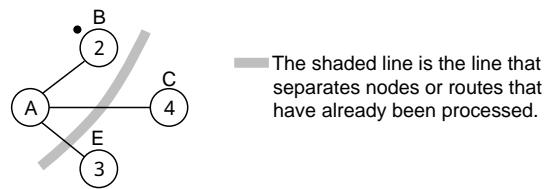
### ③ Dijkstra's search method

The Dijkstra's search method is a method to which the breadth-first search method is applied.

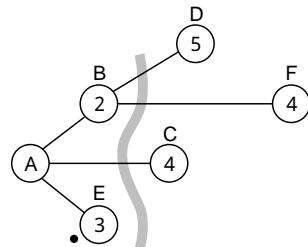
<Procedures> (Figure 3-2-54)

1. The distance to each node adjacent to the start node is measured and the node located at the shortest distance is chosen (this chosen node is defined as X).
2. The distance to each node adjacent to X from the start node as well as the distance to nodes except X from the start node are measured and the node located at the shortest distance is chosen.
3. Step 2 is repeatedly performed on all nodes until a target node is reached.
4. By adding the distance attached to each node, the shortest route can be established.

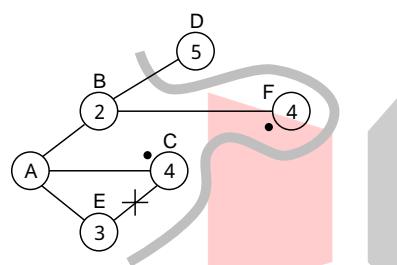
**Figure 3-2-54** Dijkstra search method



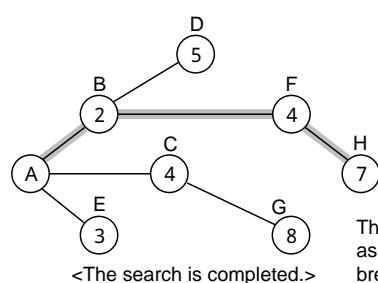
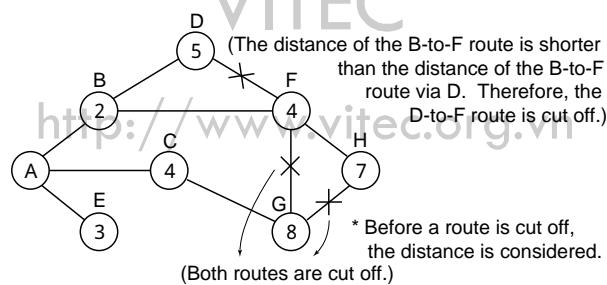
The distance from the start node to each adjacent node is compared and the node located at the shortest distance is marked.



The distance to a node located on the A-to-B route is compared with the distance of the node adjacent to A and the node at the shortest distance is marked.



C adjacent to E located at the shortest distance can be reached via A. Because the distance from A to C is shorter than the distance from A to C via E, the E-to-C route is cut off.



### 3.2.8 Numeric calculation

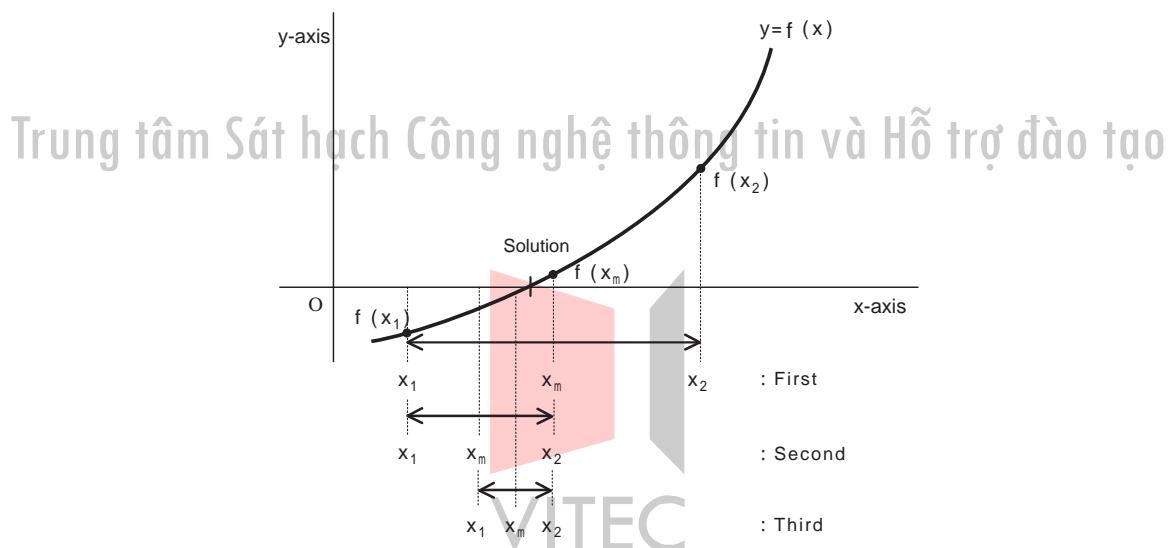
#### (1) Solution of algebraic equations

##### ① Bisection method

The bisection method is the simplest solution of higher degree equations. A commonly used higher-degree equation is defined as  $y = f(x)$ , and various values are assigned to  $x$ . The line is plotted by assigning values into  $x$ . the value of  $x$  when the line intersects  $y = 0$  ( $x$ -axis) becomes the root of this equation.

Figure 3-2-55 is a graph plotted based on  $y = f(x)$ .

Figure 3-2-55 Graph plotted based on  $y = f(x)$



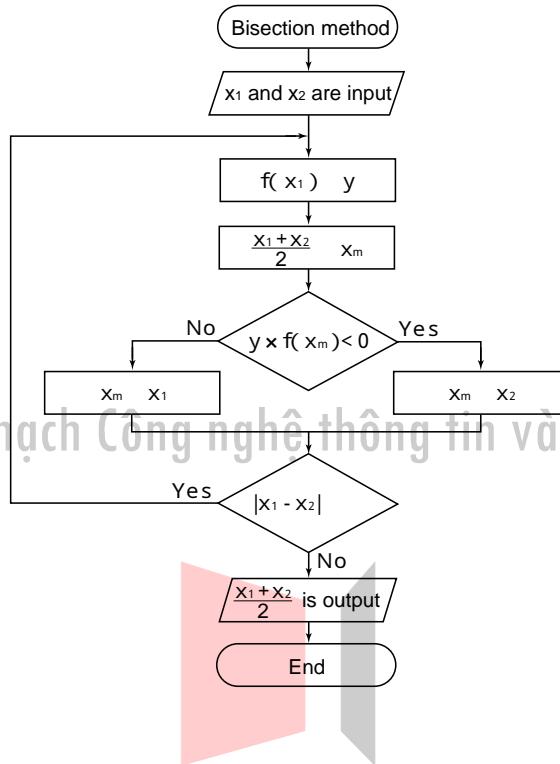
$f(x_1) \cdot f(x_2) < 0$  shows that there is at least one solution within the  $[x_1, x_2]$  region in which the line intersects the  $x$ -axis. Using the bisection method, this  $[x_1, x_2]$  region is divided into two sections. To obtain an approximate solution, the routine of dividing a region into two sections is repeatedly executed to choose a section (indicated by  $\langle - \rangle$ ) where there is a solution.

Figure 3-2-56 Algorithm of the bisection method

- Step 1 The point  $x_m$  that divides a region into two sections is calculated.
- Step 2 Each of the divided sections  $[x_1, x_m]$  and  $[x_m, x_2]$  is checked to determine in which section there is a solution. If  $f(x_1) \cdot f(x_m) < 0$ , there is a solution in the left section. If not, there is a solution in the right section.
- Step 3 If  $f(x_1) \cdot f(x_m) < 0$ ,  $x_m \leftarrow x_2$ . If not,  $x_m \leftarrow x_1$ .
- Step 4 A judgment made -- whether the section length  $|x_1 - x_2|$  is smaller than the convergence judgment value -- or not -- is used to determine whether the approximate solution has been obtained or not. If  $|x_1 - x_2|$  is larger than , the routine goes back to step 1 and repeats steps 1 through 4.

Figure 3-2-57 shows the flowchart of the algorithm for obtaining the solution of  $f(x) = 0$  using the bisection method.

**Figure 3-2-57** Flowchart of the algorithm for obtaining the solution of  $f(x) = 0$  using the bisection method



## ②Newton's method

Newton's method assumes that an approximate solution of a higher degree equation is already known. The approximate solution is repeatedly corrected to obtain a true solution. Newton's method is superior to other methods in that the convergence speed is faster and both real and imaginary solutions can be obtained.

Figure 3-2-58 is a graph plotted using Newton's method.

Figure 3-2-58 Newton's method

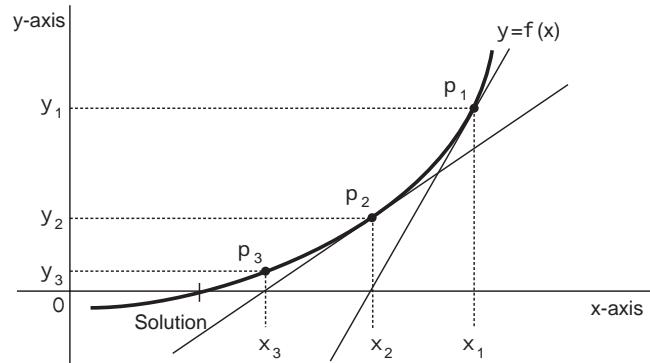


Figure 3-2-59 Algorithm of Newton's method

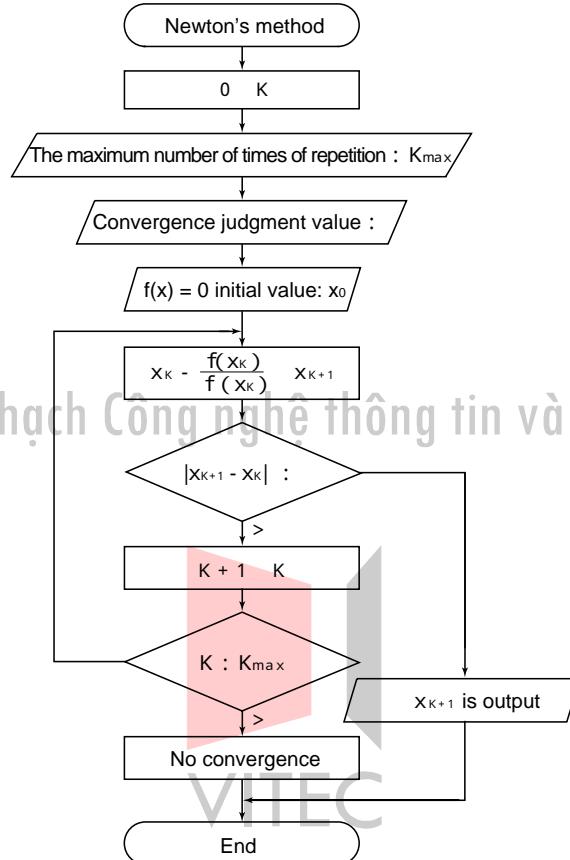
- Step 1 A tangential line  $y = f(x)$  at point  $p_1$  of the coordinate  $x_1, y_1$  is drawn and point  $x_2$  where the tangential line intersects x-axis is obtained.
- Step 2 A tangential line  $y = f(x)$  at point  $p_2$  of the coordinate  $x_2, y_2$  is likewise drawn and point  $x_1$  where the tangential line intersects x-axis is obtained. As this step is repeatedly executed the tangential line moves closer to a solution.
- Step 3 The difference between adjacent approximate values obtained in step 2 is compared with a predetermined convergence judgment value precision. Steps 1 and 2 are repeatedly executed until this difference becomes smaller than the predetermined convergence judgment value.

Because the equation for a tangential line at point  $p_1$  is  $y - f(x_1) = f'(x_1)(x - x_1)$ , point  $x_2$  where a tangential line intersects x-axis can be obtained using the following equations:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (i = 0, 1, 2, \dots)$$

Figure 3-2-60 shows the flowchart of the algorithm for obtaining the solution of  $f(x)=0$  using Newton's method.

**Figure 3-2-60** Flowchart of the algorithm for obtaining the solution of  $f(x) = 0$  using Newton's method



<http://www.vitec.org.vn>

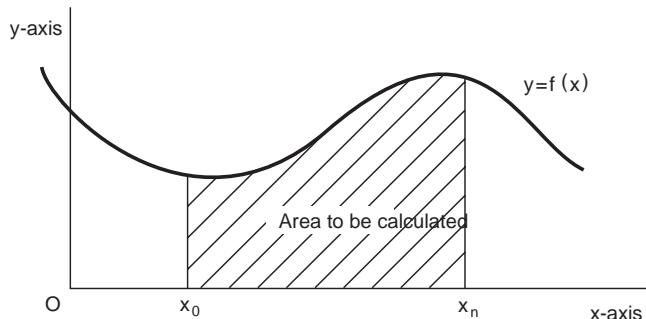
## (2) Numerical Integration

Numerical integration methods are used to calculate the area of a figure enclosed by curved lines. This section describes the trapezoidal rule and Simpson's method of all commonly used numerical integration methods.

## ① Trapezoidal rule

Figure 3-2-61 shows the basic concept of the trapezoidal rule.

Figure 3-2-61 Basic concept of the trapezoidal rule



Using the trapezoidal rule, an area is divided into multiple trapezoids to obtain an area.

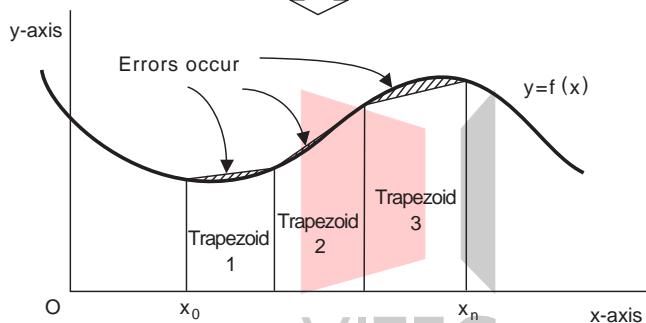


Figure 3-2-62 shows the procedure for calculating an area enclosed by the curved line  $y = f(x)$ , a region along the x-axis and a portion enclosed by the x-axis.

Figure 3-2-62 Procedure for calculating an area using the trapezoidal rule

- Step 1 Vertical lines are drawn in an area at regular intervals to divide it into parts.
- Step 2 Assuming that the curved line of each divided area is a straight line, each divided area is regarded as a trapezoid.
- Step 3 Each divided area is regarded as a trapezoid and each area is calculated.  
An area of one trapezoid = (upper base + lower base)  $\times$  height  $\div$  2
- Step 4 Areas of all trapezoids are totaled to obtain an area.

If an area is calculated using the trapezoidal rule, errors occur with respect to shaded portions, as shown in Figure 3-2-61, sic a true area is different from a trapezoidal area. To reduce errors, the number of trapezoids is increased.

Figure 3-2-63 shows how an area is divided into trapezoids.

**Figure 3-2-63** An area divided according to the trapezoidal rule

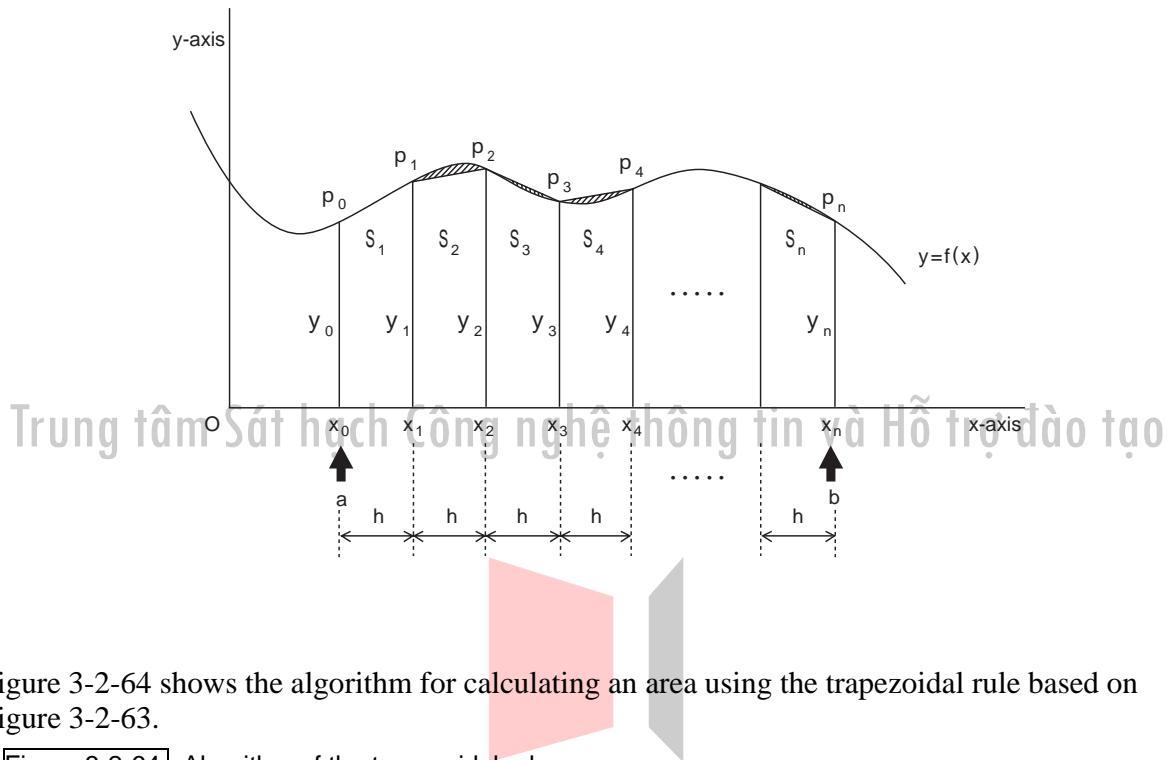


Figure 3-2-64 shows the algorithm for calculating an area using the trapezoidal rule based on Figure 3-2-63.

**Figure 3-2-64** Algorithm of the trapezoidal rule

Step 1 The portion where integration takes place, i.e.,  $[a, b]$  of  $y = f(x)$ , is divided into parts, the number of which is defined as  $n$ .  
 Step 2 Points where lines set at regular intervals intersect x-axis are defined as  $x_0, x_1, x_2, \dots, x_n$  from left to right.  
 Step 3 The values of functions at above intersection points are defined as  $y_0, y_1, y_2, \dots, y_n$ .  
 Step 4 Points where lines set at regular intervals intersect curved lines are defined as  $p_0, p_1, p_2, \dots, p_n$ . With  $x, y$  and  $p$  points connected, a trapezoid is formed.  
 Step 5 Providing that an area of each trapezoid is  $S_1, S_2, S_3, \dots, S_n$ ,

$$S_1 = \frac{h}{2} (y_0 + y_1)$$

$$S_2 = \frac{h}{2} (y_1 + y_2)$$

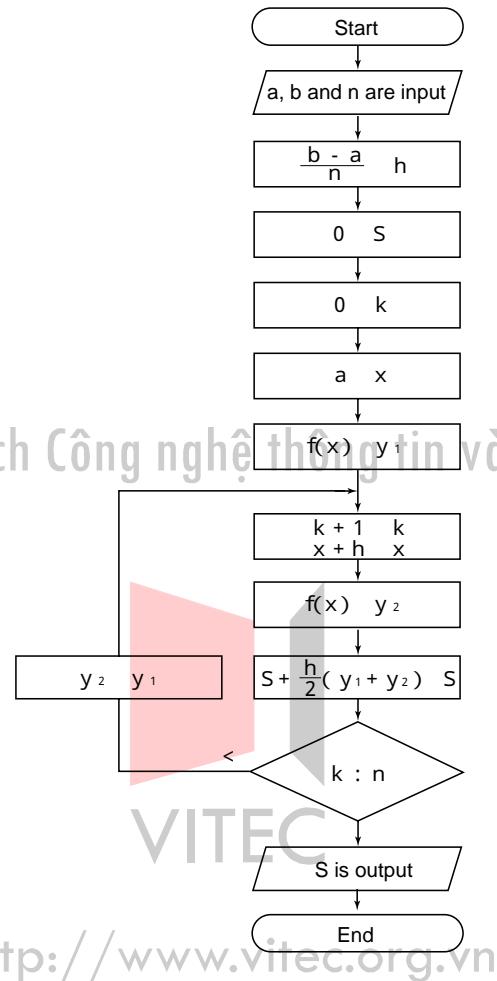
$$\vdots$$

$$S_n = \frac{h}{2} (y_{n-1} + y_n)$$

Each area is calculated using the above equations.  
 Step 6 An area enclosed by the curved line can be obtained by totaling areas of each divided area:  $S = S_1 + S_2 + S_3 + \dots + S_n$

Figure 3-2-65 shows the flowchart of the algorithm for calculating an area using a computer and the trapezoidal rule.

**Figure 3-2-65** Flowchart of the approximate area calculation using the trapezoidal rule

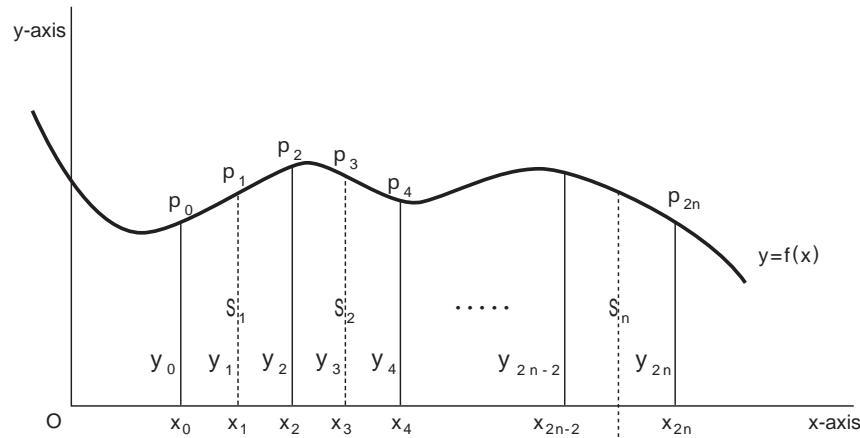


## ② Simpson's method

Using the trapezoidal rule, a curved line is divided at regular intervals and intersection points on the curved line are connected to form trapezoids. Areas of each trapezoid are total to obtain a whole area. Although errors can be reduced by increasing the number of trapezoids, it still holds that the greater the number of trapezoids, the longer it takes to execute totaling. Furthermore, because the method is based on the scheme wherein an area enclosed by three straight lines and one curved line is regarded as a trapezoid, there is concern over the accuracy of the result obtained.

As a solution, Simpson's method is used as shown in Figure 3-2-66. Using this method, a curved line is approximated to an easy-to-handle parabola to calculate an area.

Figure 3-2-66 Area divided using Simpson's method



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

To calculate an area  $S_1$  enclosed by  $y=f(x)$ ,  $x=x_0$ ,  $x=x_2$  and  $x$ -axis shown in Figure 3-2-66 using Simpson's method,  $f(x)$  is considered a parabola that runs through  $p_0$ ,  $p_1$  and  $p_2$ , and  $S_1$  is approximated as follows:

$$S_1 = \frac{(y_0 + 4y_1 + y_2)h}{3}$$

This method is quite different from the trapezoidal rule in that an area is equally divided into  $2n$  (equal, even-number division), not into  $n$ .

Figure 3-2-67 shows the algorithm for calculating an area shown in Figure 3-2-66 using Simpson's method.

Figure 3-2-67 Algorithm for calculating an area using Simpson's method

Step 1 The section  $[a, b]$  of the function  $y = f(x)$  where integration takes place is equally divided into  $2n$  (equal, even-number division). H

Step 2 Points where dividing lines intersect x-axis are defined from the left as  $x_0, x_1, x_2, \dots, x_{2n}$ .

Step 3 Function values at each intersection point are defined as  $y_0, y_1, y_2, \dots, y_{2n}$ .

Step 4 Points where dividing lines intersect a curved line are defined as  $p_0, p_1, p_2, \dots, p_{2n}$ .

Step 5 Three points  $p_{2i-2}(x_{2i-2}, y_{2i-2})$ ,  $p_{2i-1}(x_{2i-1}, y_{2i-1})$  and  $p_{2i}(x_{2i}, y_{2i})$  in two sections are approximated to a parabola to calculate an area  $S_i$ .

$$S_1 = \frac{(y_0 + 4y_1 + y_2)h}{3}$$

$$S_2 = \frac{(y_2 + 4y_3 + y_4)h}{3}$$

⋮

$$S_n = \frac{(y_{2n-2} + 4y_{2n-1} + y_{2n})h}{3}$$

Step 6 Areas  $S_i$  in each section are totaled to obtain an area  $S$  enclosed by a curved line.

$$S = S_1 + S_2 + \dots + S_n$$

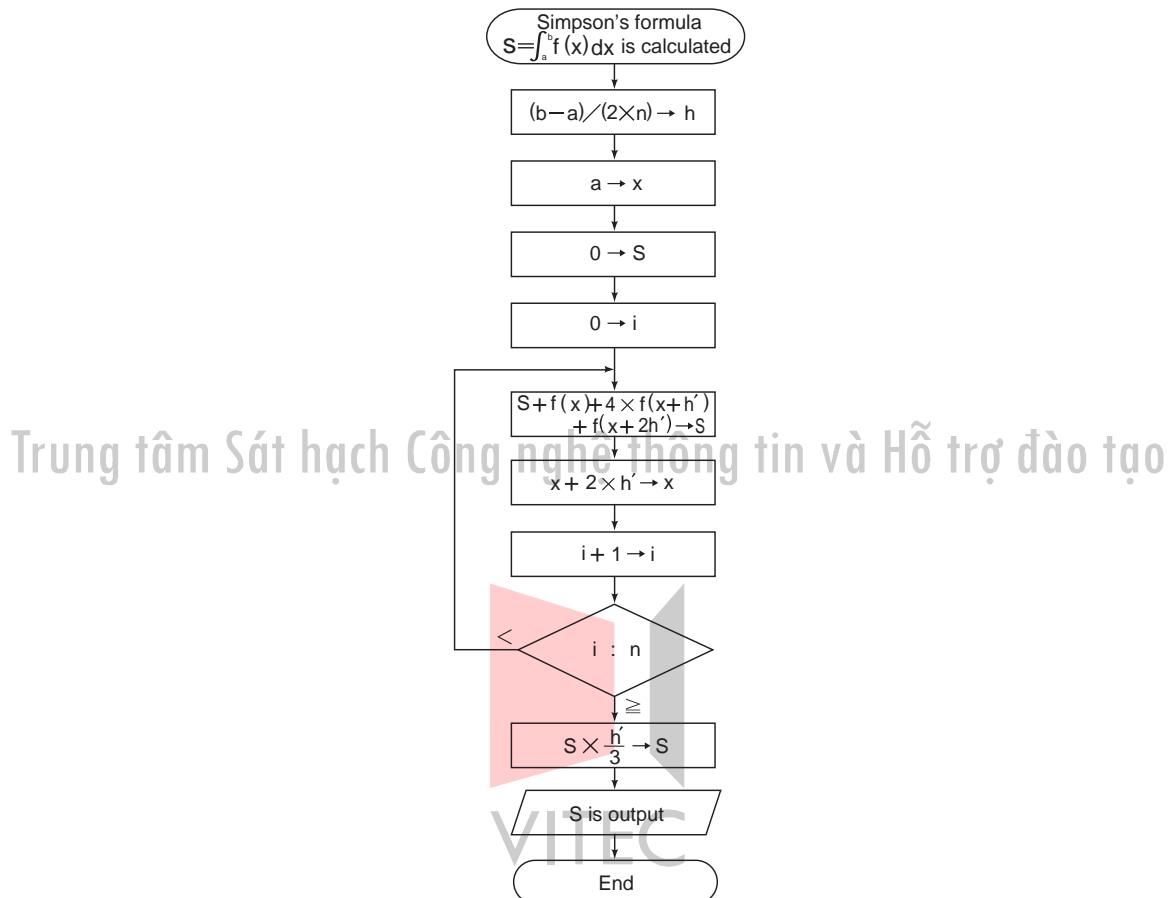
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Figure 3-2-68 shows the flowchart of the algorithm for calculating an area using a computer and the Simpson's method.

**Figure 3-2-68** Flowchart of the area approximation calculation using Simpson's method



<http://www.vitec.org.vn>

### 3.2.9 Collation algorithm

In the collation algorithm, values stored in the array are compared to obtain a solution. A search of character strings described under Section 2.2.4, Character string processing, also uses one of the collation algorithms.

This section describes the stable marriage problem to explain one of the representative collation algorithms.

#### (1) Stable marriage problem

In solving the stable marriage problem, stable pairs of men and women are determined. Stable means a state in which men and women feel more fondness for their present partners than others. Specifically, supposing that there are three men and three women, men are named A, B

and C and women are named a, b, and c. the levels of fondness (high to low levels in the order 1,2 and 3) are shown in the table below:

<Men>

	1	2	3
A	b	a	c
B	c	a	b
C	c	b	a

<Women>

	1	2	3
a	A	C	B
b	C	B	A
c	A	B	C

If they are paired as

$$P = \{ (A, a), (B, b), (C, c) \}$$

A feels fonder of b than a, who is the present partner. b feels fonder of B, who is the present partner, than A.

Therefore, there should be no problems. B feels fonder of a than b, who is the present partner. Because a feels fonder of A, who is the present partner, than B, there should be no problem.

However, c feels fonder of B than C who is the present partner. This state is called an unstable state. By changing partners, they can be paired as

$$P = \{ (A, a), (B, c), (C, b) \}.$$

This state of pairing can be analyzed as follows:

① A feels fonder of b than a, who is the present partner.  $\rightarrow$  b feels fonder of C, who is the present partner, than A.

② B feels fonder of c who is the present partner.

③ C feels fonder of c than b, who is the present partner.  $\rightarrow$  c feels fonder of B, who is the present partner, than C.

④ a feels fondest of A, who is the present partner.

⑤ b feels fondest of C, who is the present partner.

⑥ c feels fonder of A than B, who is the present partner.  $\rightarrow$  A feels fonder of a, who is the present partner, than c.

In this pairing, no pairs have an unstable factor. This result is called stable or stable matching.

Stable matching may not necessarily be determined uniquely. There may be a case where stable matching cannot be achieved using the approach described above. The stable marriage problem to be described in the following is designed to achieve stable matching by specifying the conditions for determining pairs:

[Stable marriage problem]

M Men and N women are looking for stable pairs

- Men's and women's levels of fondness are preset in array M and F (the number of elements: N+1).
- As levels of fondness, element numbers 1 through 5 (high to low) are assigned to the numbers of each partner

**Example**

If there are five men and women

[Combination M: Levels of fondness seen from the side of the men]

	1	2	3	4	5	6(PT)
M (1)	2	1	4	3	5	0
M (2)	1	3	4	2	5	0
M (3)	1	4	5	2	3	0
M (4)	5	4	1	3	2	0
M (5)	4	2	3	1	5	0

\* A partner is entered in PT.  
0 is set here as the initial value

[Combination M: Levels of fondness seen from the side of the women]

	1	2	3	4	5	6(PT)
F (1)	3	1	5	2	4	0
F (2)	2	1	4	3	5	0
F (3)	3	2	5	4	1	0
F (4)	5	2	3	3	4	0
F (5)	5	4	1	1	3	0

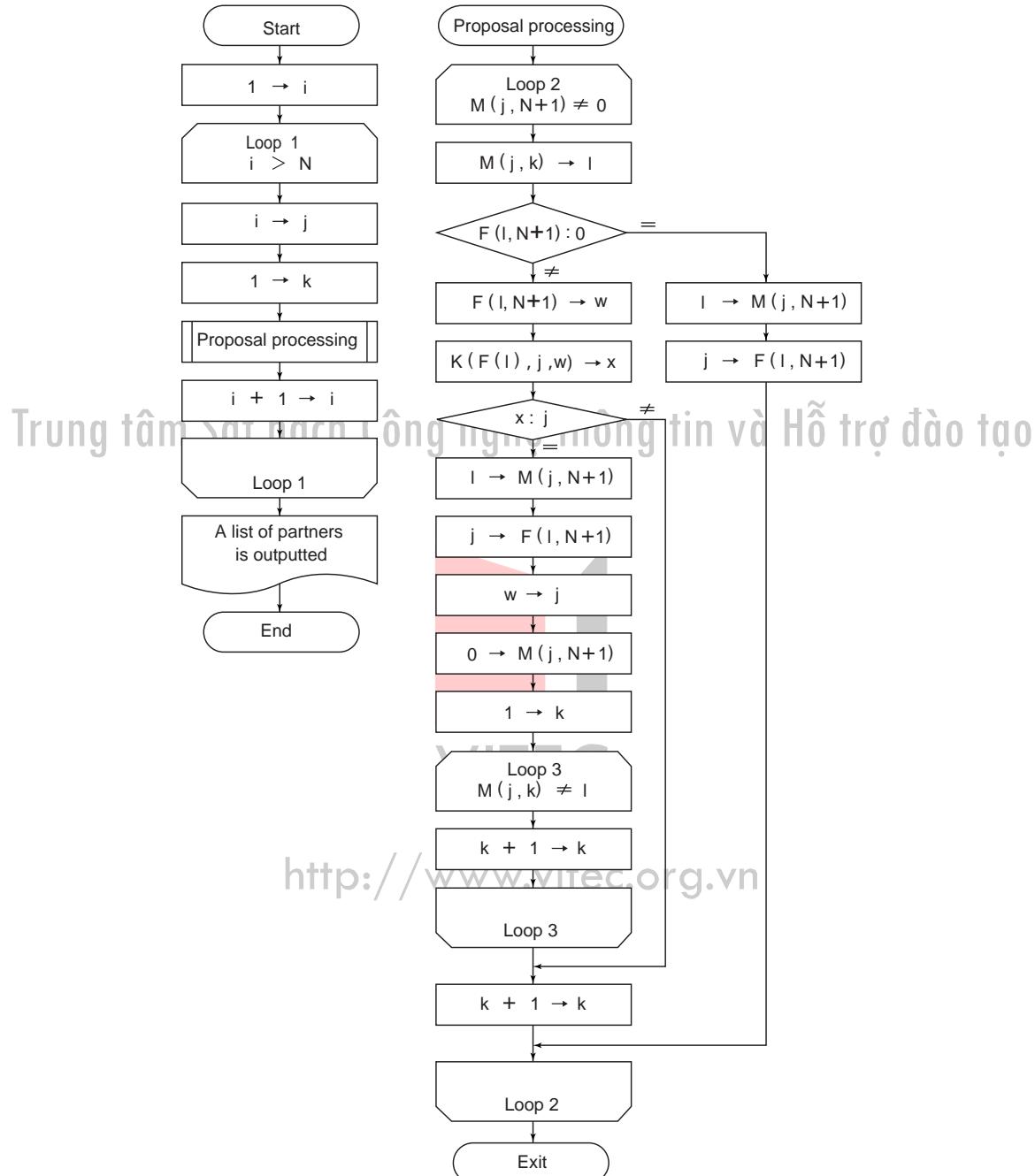
\* A partner is entered in PT.  
0 is set here as the initial value

- Proposals are made in the order of M (1), M (2).... M (N).
- Men make proposals to women in high-to-low order of fondness. They repeatedly make proposals until they obtain OKs. It may happen, however, that the OKs thus obtained are canceled.
- Women give OK or NO on the following criteria:
  - ① If one receives a proposal for the first time, she gives an OK and secures him as a partner.
  - ② If she already has a man, she compares her fondness for the present partner and that for the one who has made the proposal.
- If she feels fonder of the present partner, she gives a NO.
- If she feels fonder of a man who has made a proposal, she cancels the present partner and secures the man who has made the proposal as a partner (OK).
- A man who obtained an OK but was canceled later begins making proposals to women in high-to-low order of fondness. He repeatedly makes proposals until he obtains an OK.

In the flowchart shown in Figure 3-2-69, the function K (p1, p2, p3) is used to compare levels of fondness. Function K (p1, p2, p3): This function is for returning either p2 or p3 in the array element p1, whichever is higher in the level of fondness. K ( F(2), 1, 3) -> 1

Figure 3-2-69 shows the flowchart of the algorithm of the stable marriage problem.

**Figure 3-2-69** Flowchart of the algorithm of the stable marriage problem



### 3.2.10 Approximate and probability algorithms

Algorithms are generally used to obtain true values or solutions. There are problems that require a very long time to solve, and there are problems for which no algorithm is available. In this case,

algorithms for obtaining the solutions whose errors or whose possibility of containing mistakes are very small are used.

### (1) Approximation algorithms

Approximation algorithms are used to obtain an approximate solution in those cases where obtaining a true solution to a problem is impossible or it takes a very long time. Both Newton's method and Simpson's method described in Section 2.2.8 fall under the approximation algorithm.

[Knapsack problem]

You have  $n$  goods which are different in weight and value. You want to sell them in a town but you cannot put all of them into your knapsack. Find what combination of goods maximizes their value.

We apply the following values to explain how the algorithm works:

The number of goods (items) : 5

Weight of each item : {2,3,5,7,8} which apply to item 1, item 2,..... item 5 in that order

Value of each item(10,000 yen) : {2,4,8,9,10} which apply to item 1, item 2,..... item 5 in that order

Capacity of the knapsack : 10 kg

One way to solve this problem is to consider each combination of goods that would bring the total weight to 10 kg and compare the total value of goods in each combination. For this particular problem, the total value becomes highest if goods 1,2 and 3 are packed into the knapsack, as shown in the table below:

Goods chosen	Total weight	Total value
Goods 1,2 and 3	$2 + 3 + 5 = 10$	$2 + 4 + 8 = 14$
Goods 1 and 5	$2 + 8 = 10$	$2 + 10 = 12$
Goods 2 and 4	$3 + 7 = 10$	$4 + 9 = 13$

Maximum

Making the total weight equal to the knapsack weight is not always the best solution. For example, if there were a sixth item, 9 kg in weight and ₫150,000 in values, the maximum value could be obtained by packing this item alone into the knapsack. To find the best solution, all possible combinations of goods must be considered. As goods increase in number, the number of combinations becomes enormous, and it takes a great deal of time to find the solution. As a solution to this, the approximation algorithm can be used.

The knapsack problem can be formulated as follows:

There are  $n$  positive integers  $a_1, a_2, a_3, \dots, a_n$  and  $b_1, b_2, b_3, \dots, b_n$  and a positive integer  $c$ . Find a combination of  $x_1, x_2, x_3, \dots, x_n$  that maximizes the total sum of  $b_i x_i$  ( $i=1-n$ ) where  $x_i = \{0,1\}$  and the total

sum of  $a_i x_i$  ( $i=1-n$ ) is equal to or smaller than  $c$ .

If this formula is considered in analogy to the problem previously discussed,  $a$  is the weight of goods,  $b$  is the value of goods, and  $c$  is the capacity of the knapsack.  $x$  is whether the good is packed into the knapsack or not. 0 means that the good is not packed and 1 means that the good is packed. Therefore, the knapsack problem and the solution can be expressed as follows:

$$a = |2, 3, 5, 7, 8|$$

$$b = |2, 4, 8, 9, 10|$$

$$c = 10$$

$$x = |1, 1, 1, 0, 0|$$

Using this formula, a search is made the  $2^n$  times to check all possible combinations of 0,1 in array  $x$  if the approximation algorithm is not used.

Using the approximation algorithm, however, unit values of all goods are first identified. A unit value means the value per weight and it is given by value  $b_i / \text{weight}(a_i)$

Unit values  $k = |2/2, 4/3, 8/5, 9/7, 10/8|$

Trung tâm Sứ huých Công nghệ thông tin và Hỗ trợ đào tạo

After all unit values are identified, goods are packed into the knapsack in the order of high to low unit values. Because goods cannot be divided, goods that exceed the unused capacity of the knapsack cannot be packed.

① Item 3 with the highest unit value is packed: Unused capacity =  $10 - 5 = 5$

② Item 2 with the second highest unit value is packed: Unused capacity =  $5 - 3.33 = 1.67$

③ Item 4 with a unit value less than that of item 2 cannot be packed : Unused capacity < the weight of item 4

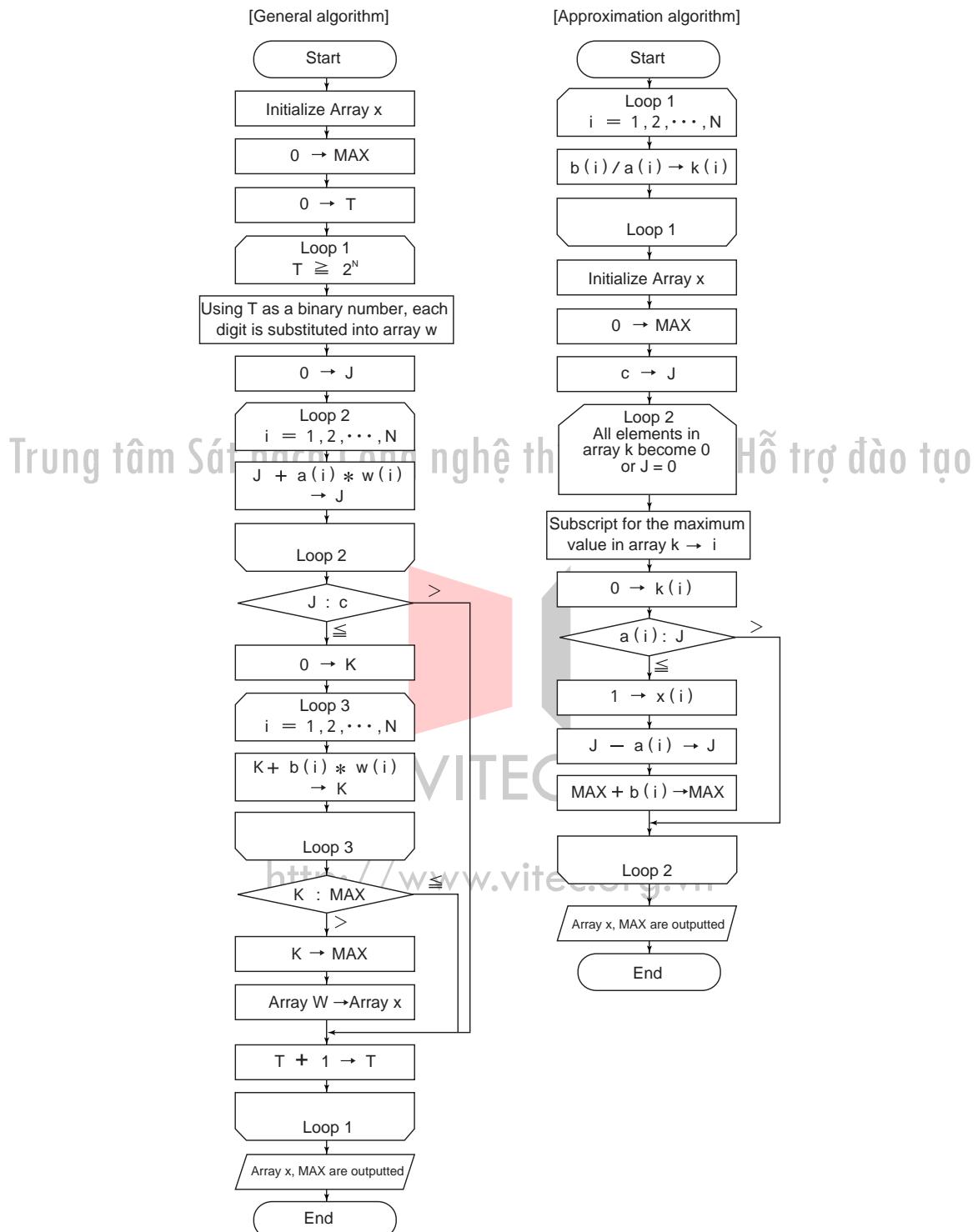
④ Item 5 with a unit value less than that of item 4 cannot be packed : Unused capacity < the weight of item 5

⑤ Item 1 with a unit value less than that of item 5 is packed: Unused capacity =  $2 - 2 = 0$

This way to obtain a solution does necessarily give the best solution. If values are defined as  $\{2,4,8,11, 10\}$ , the approximation algorithm gives the solution: (item1, item2, item3) = 140,000, though we already have the best solution: (item2, item 4) = 150,000. The approximation algorithm, however, is used in many different fields such as a method of quickly obtaining an approximate value very close to the best solution.

Figure 3-2-70 shows the flowchart of the algorithm for solving the knapsack problem.

Figure 3-2-70 Flowchart of the algorithm for solving the knapsack problem



## (2) Probability algorithm

The probability algorithm uses a probabilistic approach to find a solution using random numbers. A probabilistic approach is one in which the appropriateness of a solution is examined in terms of probability, or that a solution is found based on the probability of occurrence of a certain event. This section describes the primarity test problem as an algorithm for examining the appropriateness of a solution in terms of probability, and the case of obtaining a circular consistent  $\pi$  as an algorithm for finding a solution based on the probability of occurrence of a certain event.

### ① Primarily test problem

Primarity test problem is a process of checking given  $N (= 2^s d + 1, d$  is an odd number) and determining whether it is a prime number of not. To solve the primarity test problem, give  $N$  is divided using integers  $2, 3, \dots, \sqrt{N}$ , and it is considered a prime number if it is not divisible without a remainder by any of the integer. Although a correct solution can be obtained using this approach, a much longer time period is required to arrive at a solution if the value of  $N$  is large. As a solution, Rabin's algorithm, designed with the following theorem is used.

#### [Theorem of prime numbers]

If  $N (= 2^s d + 1, d$  is an odd number) is a prime number, either of two conditions shown below stand for the arbitrarily chosen positive integer,  $a (> 1)$

- $a^d \equiv 1 \pmod{N}$
- $a^{2kd} \equiv -1 \pmod{N}$  where  $0 \leq k \leq s-1$

If the arbitrarily chosen integer,  $a$ , does not meet the above conditions,  $N$  is considered a composite number, not a prime number. The probability that the arbitrarily chosen integer,  $a$ , for a composite number  $N$  can meet the above conditions is equal to or lower than  $1/4$ . Therefore, if the arbitrarily chose integer,  $a$ , meets the above conditions, the probability of the correctness of the judgement that  $N$  is a prime number is equal to or higher than  $3/4$ .

	Probability that the conditions can be met	Probability that the conditions cannot be met
Prime number $N$	100%	0%
Composite number $N$	25% or lower	75% or higher

If this algorithm is used, the possibility remains that the solution given by the algorithm is incorrect. This type of algorithm is called the probability algorithm with bounded errors. For the above particular example, since the probability that errors will occur is sufficiently low, the solution given by the algorithm should be considered dependable. In addition to the probability algorithm with bounded errors, the probability algorithm with no errors is sometimes used.

Although this algorithm can theoretically assure the correctness of a given solution, it sometimes takes too long to execute the algorithm process, or it ends up giving no definite solution.

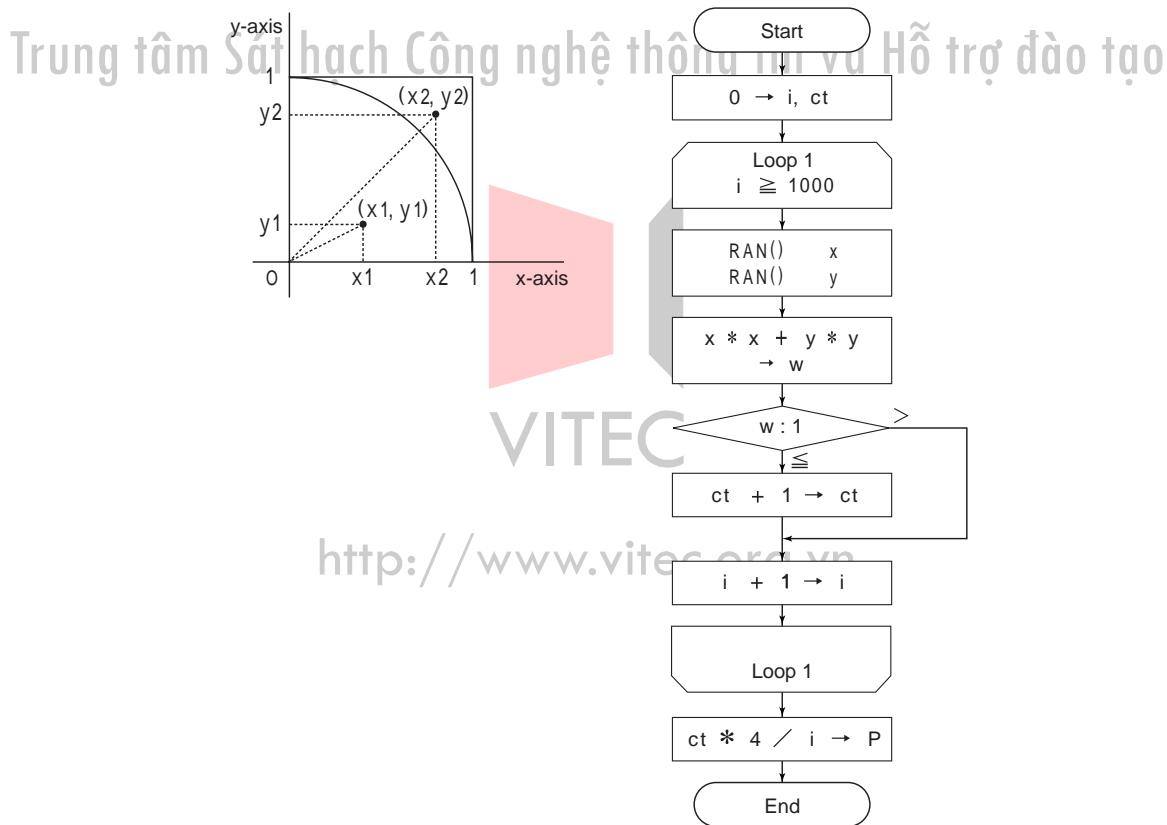
A probability algorithm with no errors is the probabilistic quick sort algorithm. This algorithm designed to improve efficiency by randomly rearranging input data using random numbers.

## ② How to find a circular constant $\pi$

Figure 3-2-71 shows a circle of 1 in radius. The area enclosed by the axes and arc is a fourth of that of the complete circle having a radius of 1. Therefore, it is  $\pi/4$  ( $= 1 \times 1 \times \pi/4$ ). On the other hand, the area of a square enclosed by four lines,  $x = 0$ ,  $x = 1$ ,  $y = 0$ , and  $y = 1$  is 1. If points on this square are designated in a random manner, the probability that these designated points is inside the circle is  $\pi/4$ .

In the flowchart shown in Figure 3-2-71, 1,000 points are generated using the function RAN() that can generate random numbers between 0 and 1. Whether generated points are inside the circle or not is determining by measuring the straight distance from (0,0) to each point. To simplify the calculation, the root calculation is omitted. In the case of the example shown in Figure 3-2-71, it is judged that  $(x_1, y_1)$  is inside if  $(x_1^2 + y_1^2) \leq 1$  and that  $(x_2, y_2)$  is outside if  $(x_2^2 + y_2^2) > 1$ .

**Figure 3-2-71** Flowchart of the algorithm for finding a circular constant



Because a circular constant obtained in this way sometimes contains errors resulting from the characteristics inherent in methods used to generate random numbers, it is generally used in the representation containing a standard error (solution  $\pm$  standard errors).

An algorithm like this one, that uses random numbers to solve mathematical problems, is called

the Monte Carlo method.

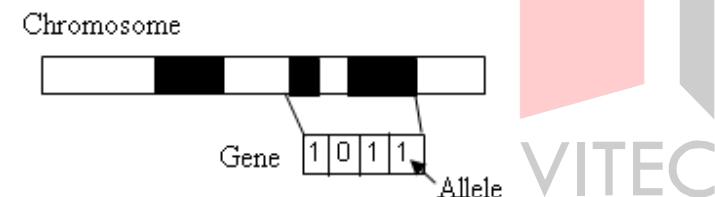
### 3.2.11 Genetic algorithms

**Genetic algorithms** are a part of **evolutionary computing**, which is a rapidly growing area of artificial intelligence. Genetic algorithms are inspired by Darwin's theory about evolution. A solution to a problem solved by genetic algorithms is evolved.

Genetic algorithms were formally introduced in the United States in the 1970s by John Holland at University of Michigan. The continuing price/performance improvements of computational systems have made them attractive for some types of optimization. In particular, genetic algorithms work very well on mixed (continuous *and* discrete), combinatorial problems. They are less susceptible to getting 'stuck' at local optima than gradient search methods. But they tend to be computationally expensive.

The three most important aspects of using genetic algorithms are:

- (1) definition of the objective function,
- (2) definition and implementation of the genetic representation,
- (3) definition and implementation of the genetic operators.



To use a genetic algorithm, you must encode solutions to your problem in a structure that can be stored in the computer. This object is a genome (or chromosome). The genetic algorithm creates a population of genomes then applies crossover and mutation to the individuals in the population to generate new individuals.

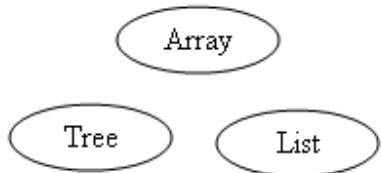
It uses various selection criteria so that it picks the best individuals for mating (and subsequent crossover).

Your objective function determines how 'good' each individual is.

Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the

best solution) is satisfied.

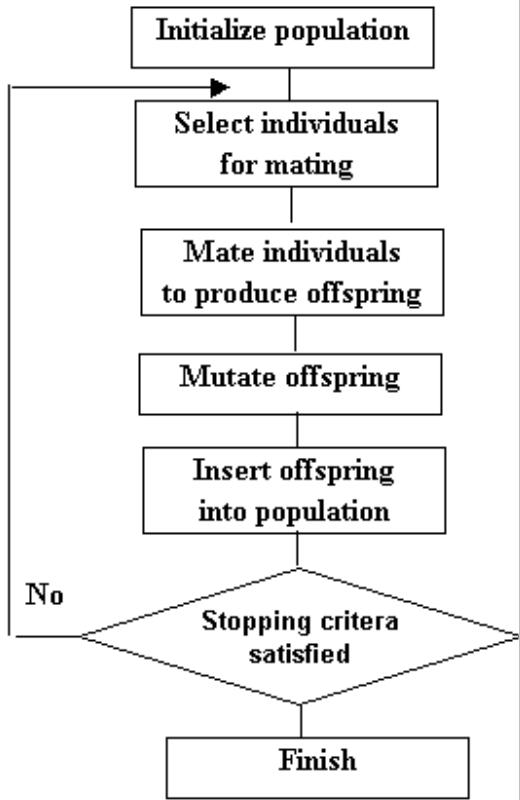


You can use any representation for the individual genomes in the genetic algorithm. Holland worked primarily with strings of bits, but you can use arrays, trees, lists, or any other object. But you must define genetic operators (initialization, mutation, crossover, comparison) for any representation that you decide to use.

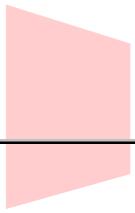
## Genetic Algorithm

1. **[Start]** Generate random population of  $n$  chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness  $f(x)$  of each chromosome  $x$  in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
  1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
  2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents. <http://www.vitec.org.vn>
  3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
  4. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step 2

- Define the representation  
 - minimal complete representation
- Define the operators  
 - initialization, crossover, mutation
- Define objective function  
 - feasibility and optimality
- Tune the algorithm  
 - selection, replacement



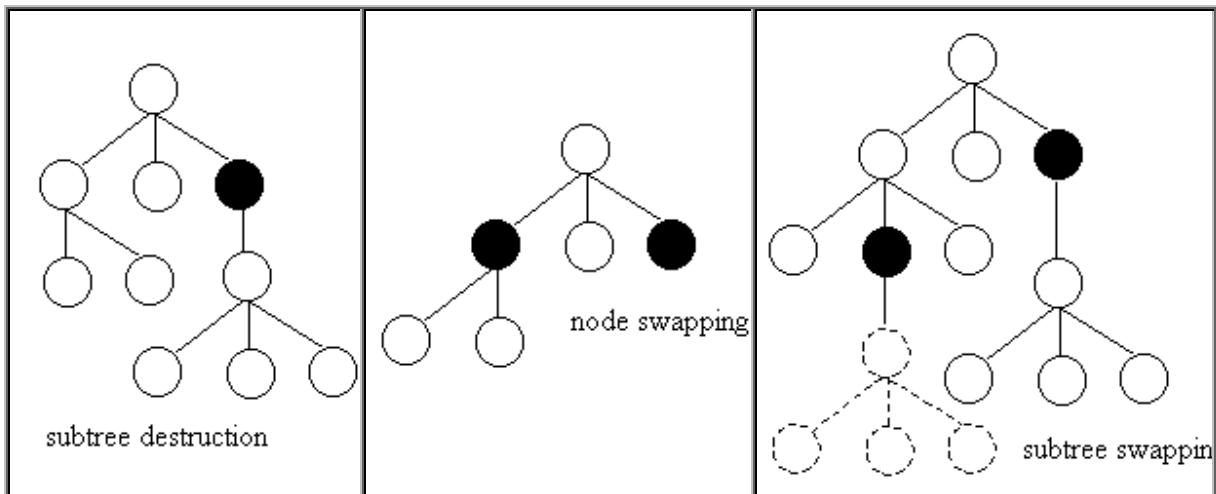
Trung tâm Sát hạch Công nghệ



VITEC

<http://www.vitec.org.vn>

## Mutation

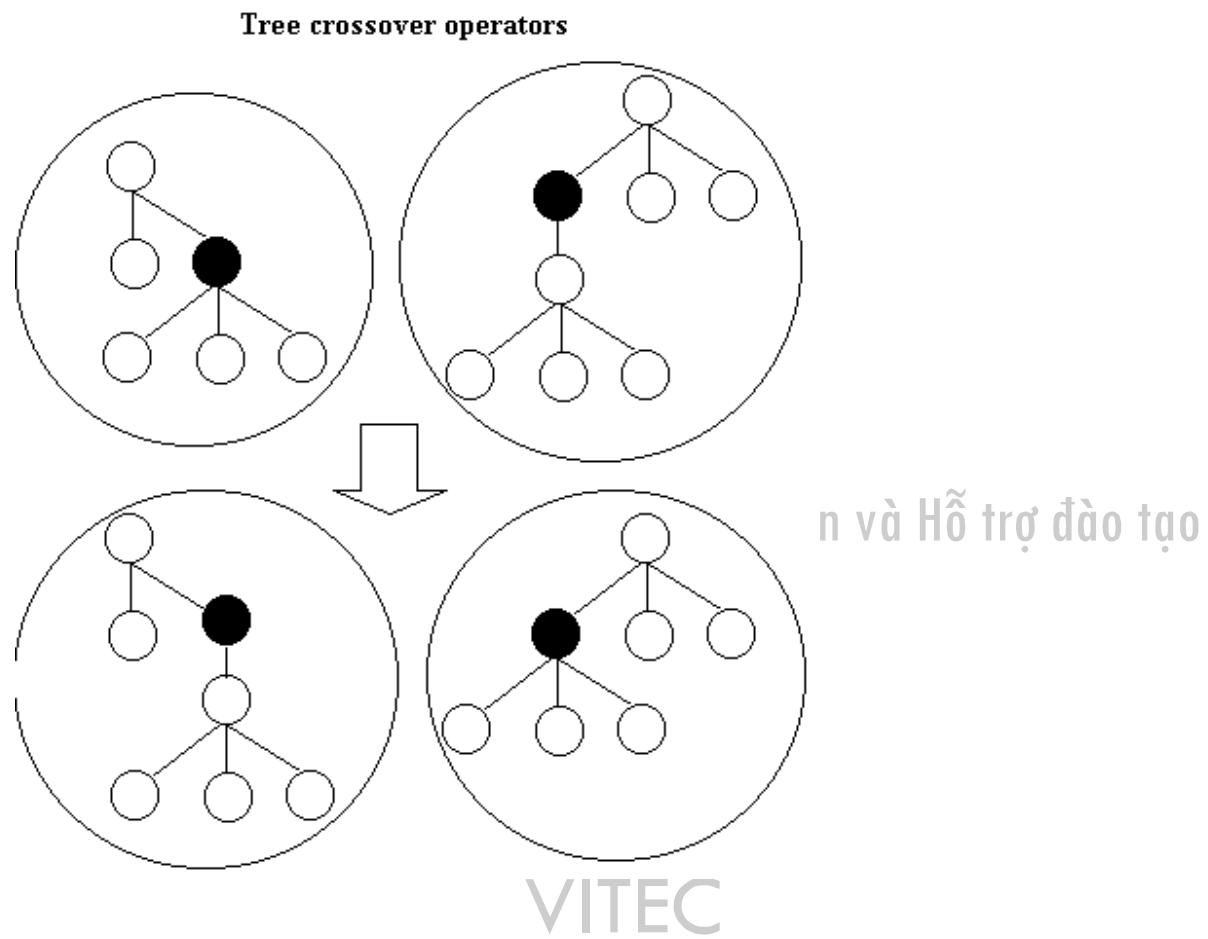


These are some sample tree mutation operators. You can use more than one operator during an evolution. The mutation operator introduces a certain amount of randomness to the search. It can help the search find solutions that crossover alone might not encounter.



<http://www.vitec.org.vn>

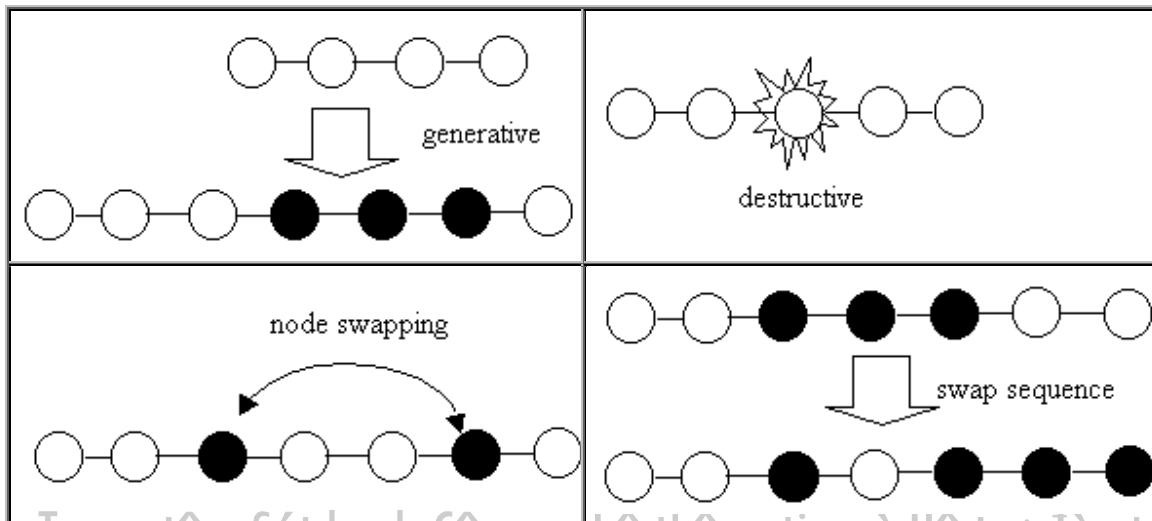
## Tree cross over operations



These are some sample tree crossover operators.

Typically crossover is defined so that two individuals (the parents) combine to produce two more individuals (the children). But you can define asexual crossover or single-child crossover as well. The primary purpose of the crossover operator is to get genetic material from the previous generation to the subsequent generation.

## Mutation operator



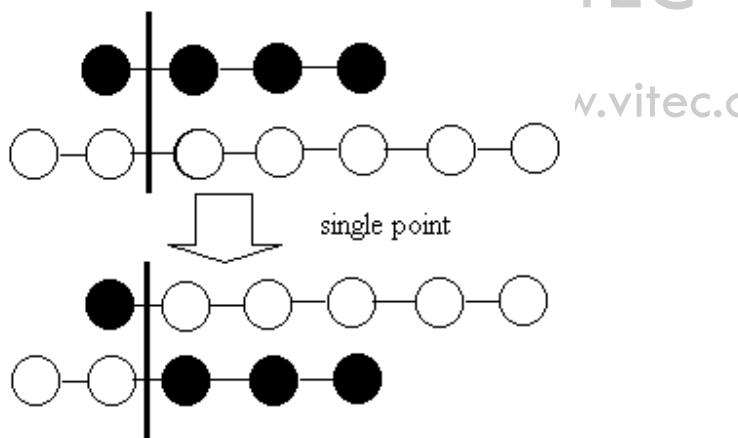
These are some sample list mutation operators. Notice that lists may be fixed or variable length. Also common are order-based lists in which the sequence is important and nodes cannot be duplicated during the genetic operations. You can use more than one operator during an evolution.

The mutation operator introduces a certain amount of randomness to the search. It can help the search find solutions that crossover alone might not encounter.

## List crossover operators

VITEC

[vitec.org.vn](http://vitec.org.vn)

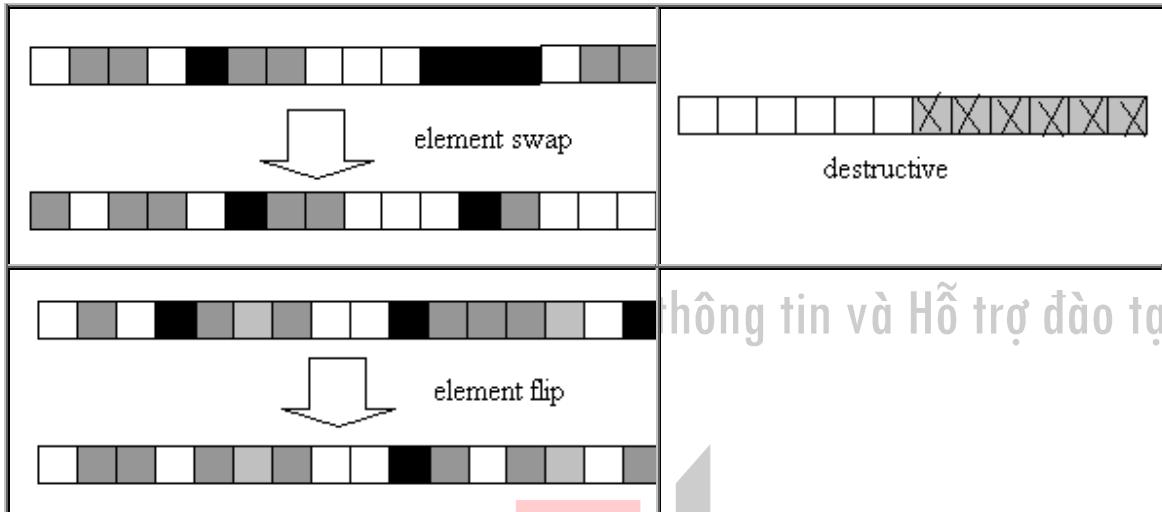


Typically crossover is defined so that two individuals (the parents) combine to produce two more individuals (the children). But you can define asexual crossover or single-child crossover as well. The primary purpose of the crossover operator is to get genetic material from the previous generation to the subsequent generation.

Notice that lists may be fixed or variable length. Also common are order-based lists in which the sequence is important and nodes cannot be duplicated during the genetic operations. You can use more than one operator during an evolution.

The mutation operator introduces a certain amount of randomness to the search. It can help the search find solutions that crossover alone might not encounter.

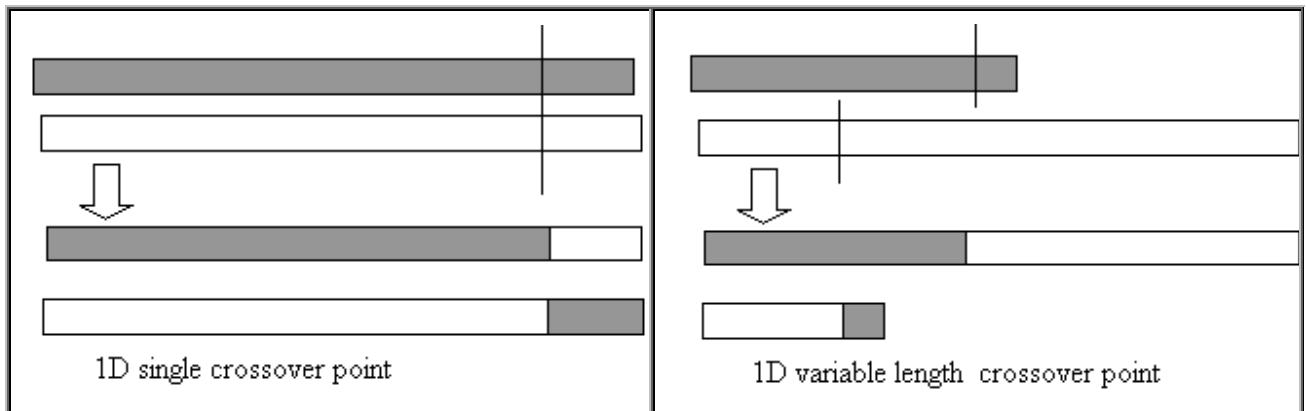
### Array mutation operators



The lists may be fixed or variable length. Also common are order-based lists in which the sequence is important and nodes cannot be duplicated during the genetic operations. You can use more than one operator during an evolution.

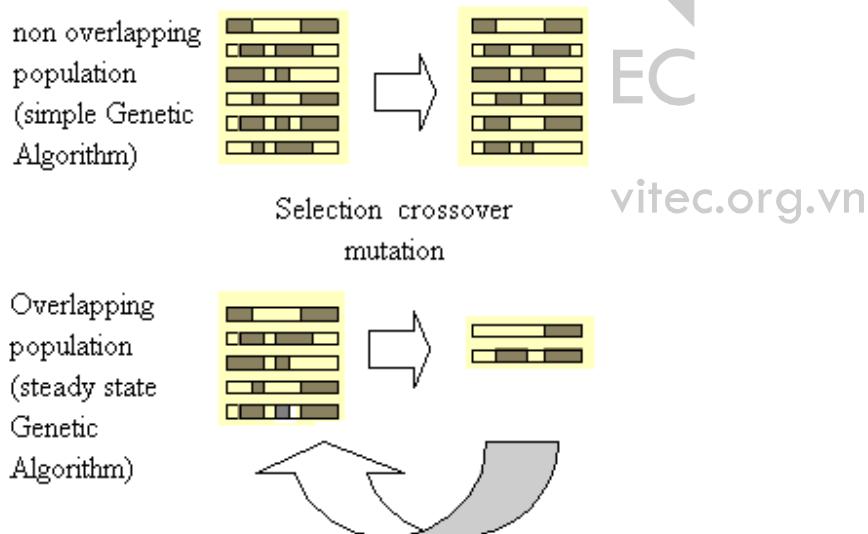
The mutation operator introduces a certain amount of randomness to the search. It can help the search find solutions that crossover alone might not encounter.

## Array crossover operator



The crossover is normally defined such that two individuals (the parents) combine to produce two more individuals (the children). But you can define asexual crossover or single-child crossover as well. The primary purpose of the crossover operator is to get genetic material from the previous generation to the subsequent generation.

Notice that arrays may be fixed or variable length. They may also be 2 or 3 dimensional (or more). Also common are order-based arrays in which the sequence is important and nodes cannot be duplicated during the genetic operations. You can use more than one operator during an evolution.



Two of the most common genetic algorithm implementations are 'simple' and 'steady state'. The simple genetic algorithm is described by Goldberg in his 1989 book, *Genetic Algorithms in*

Search and Optimization. It is a *generational* algorithm in which the entire population is replaced each generation.

The steady state genetic algorithm is used by the Genitor program. In this algorithm, only a few individuals are replaced each 'generation'. This type of replacement is often referred to as overlapping populations.

### Scaling schemes

- rank (no scaling)
- linear scaling ("fitness")
- sigma truncation scaling
- sharing (similarity scaling)

Often the objective scores must be transformed in order to help the genetic algorithm maintain diversity or differentiate between very similar individuals. The transformation from raw *objective* scores to scaled *fitness* scores is called scaling.

There are many different scaling algorithms. Some of the most common are linear (fitness proportionate) scaling, sigma truncation scaling, and sharing. Linear scaling transforms the objective score based on a linear relationship using the maximum and minimum scores in the population as the transformation metric. Sigma truncation scaling uses the population's standard deviation to do a similar transformation, but it truncates to zero the poor performers. Sharing degrades the score of individuals that are similar to other individuals in the population.

### Replacement schemes

- replace worst
- replace best
- replace parent
- replace random
- replace most similar

VITEC  
<http://www.vitec.org.vn>

Replacement schemes are used by genetic algorithms with overlapping populations to determine how the new individuals will be assimilated into the population. Replace-worst and replace most-similar are the only really useful replacement schemes. Sometimes replace-parent can be effective, but usually when the parents are similar to the offspring and this is just replace-most-similar.

### Selection schemes

- Rank selection
- roulette wheel selection
- tournament selection
- uniform stochastic sampling
- similarity based selection

The selection method determines how individuals are chosen for mating. If you use a selection method that picks only the best individual, then the population will quickly converge to that individual. So the selector should be biased toward better individuals, but should also pick some that aren't quite as good (but hopefully have some good genetic material in them).

Some of the more common methods include roulette wheel selection (the likelihood of picking an individual is proportional to the individual's score), tournament selection (a number of individuals are picked using roulette wheel selection, then the best of these is (are) chosen for mating), and rank selection (pick the best individual every time). Threshold selection can also be effective.

### Premature convergence

Often the crossover operator and selection method are too effective and they end up driving the genetic algorithm to create a population of individuals that are almost exactly the same. When the population consists of similar individuals, the likelihood of finding new solutions typically decreases.

On one hand, you want the genetic algorithm to find good individuals, but on the other you want it to maintain diversity. Two of the most common methods for maintaining diversity are DeJong-style crowding (named after Kenneth DeJong whose PhD thesis first explored the issue) and Goldberg-style fitness scaling.

DeJong-style crowding is basically the same thing as replace-most-similar replacement scheme. When new offspring are created, they replace the individuals in the population that are most similar to them.

Fitness scaling derates the objective score of individuals that are less unique than others in the population. By derating the scores of similar individuals, less similar individuals are more likely to be selected for mating.

### Areas of application

<http://www.vitec.org.vn>

The following shows the different areas where genetic algorithms are applied.

- scheduling/planning
- packing (2D and 3D)
- music generation
- plant growth
- catalog search
- structural topology
- assembly sequencing

### Conclusion

In general, genetic algorithms are better than gradient search methods if your search space has

many local optima. Since the genetic algorithm traverses the search space using the genotype rather than the phenotype, it is less likely to get stuck on a local high or low.

### 3.2.12 Data compression algorithms

#### RLE (Run-length encoding)

Run-length encoding (RLE) is a very simple form of data compression encoding. It is based on simple principle of encoding data. This principle is to every stream which is formed of the same data values (repeating values are called a run) i.e sequence of repeated data values is replaced with count number and a single value. This intuitive principle works best on certain data types in which sequences of repeated data values can be noticed; RLE is usually applied to the files that contain large number of consecutive occurrences of the same byte pattern

RLE may be used on any kind of data regardless of its content, but data which is being compressed by RLE determines how good compression ratio will be achieved. So RLE is used on text files which contain multiple spaces for indentation and formatting paragraphs, tables and charts. Digitized signals also consist of unchanged streams so such signals can also be compressed by RLE. A good example of such signal is monochrome images the run of characters is replaced with the number of the same characters and a single character.

Example:

before: R T A A A A S D E E E E  
after RLE compression: R T \*4A S D \*5E

The format of the compressed data is

Control character, Count, Character

VITEC

#### Arithmetic coding

The main problem of lossless compression is to decompose a data set into a sequence of events. Those events are to be encoded using as few bits as possible. The idea is to give a shorter codeword to more probable events. The data set could be compressed whenever some events are more probable than other.

The Arithmetic coding with accurate probability of events gives an optimal compression.

#### Algorithm

With given probabilities of events, the algorithm works in these three steps.

1. The coder starts with a “current interval” [H, L) set to [0,1).
2. For each events int the input file/file, the coder performs an (a) and (b) step.

- a. the coder subdivides current interval into subintervals, one for each event.
- b. The size of a subinterval is proportion to the probability that the event will be the next event in the field/file. the code selects the subinterval corresponding to the event that actually occurs next and makes it the new current interval
- 3. The output is the last current interval. Better to say, the output are so many bits that accurately distinguish the last current interval from all other final intervals.

The size of final interval is equal to the product of the probabilities of all individual events that occurs in the input file.

The description of algorithm in pseudocode for implementation

*set Low to 0*

*set High to 1*

*while there are input symbols do*  
*take a symbol*

*CodeRange = High - Low*

*High = Low + CodeRange \* HighRange(symbol)*

*Low = Low + CodeRange \* Low Range(symbol)*

*end of while*

*output Low*

The low and high parts of subintervals are subintervals of interval [0,1]. The size of each subinterval depends on the probability that this symbol occurs in the input field/file.

<http://www.vitec.org.vn>

### Example 1.

Table 1.

event	probability of event
a1	2/3
b1	1/3
a2	1

b2	1
a3	3/5
b3	2/5

The input field is given by these symbols: b1a2b3. The coding is given in table 2.

Action	Subinterval
Begin	[0,1)
subdivide current interval into new intervals of 2/3 and 1/3 of current interval	[0, 2/3), [2/3, 1)
new current interval (event is b1):	[2/3, 1)
subdivide current interval into new intervals of 1 and 1 of current interval	[2/3, 5/6), [5/6, 1)
new current interval (event is a2):	[2/3, 5/6)
subdivide current interval into new intervals of 1 and 1 of current interval	[2/3, 23/30), [23/30, 5/6)
new current interval (event is b3):	[23/30, 5/6)

The final interval is [23/30, 5/6). The size of this event is 1/15. That is the product of the sizes of probabilities events in the input filed

$$p_{b_1} * p_{a_2} * p_{b_3} = \frac{1}{3} * \frac{1}{2} * \frac{2}{5} = \frac{1}{15}.$$

### **Example 2.**

The message to be encoded is “ARITHMETIC”. There are ten symbols in the message. The probability distribution is given in the table 3.

Symbol	Subinterval
A	1/10
C	1/10
E	1/10
H	1/10
I	2/10
M	1/10
R	1/10
T	2/10

Each character is assigned the portion of the starting interval [0,1). The size of interval corresponds to symbol's probability of appearance.

Symbol	Subinterval
A	0,00 – 0,10
C	0,10 – 0,20
E	0,20 – 0,30
H	0,30 – 0,40
I	0,40 – 0,60
M	0,60 – 0,70
R	0,70 – 0,80
T	0,80 – 1,00

The coding is performed in the table 5. Note, the most significant portion of a coded message belongs to the first symbol to be encoded. In this case, the first symbol is “A” which owns range [0, 0,1).

Symbol	LowRange	HighRange
	0,0	1,0
A	0,0	0,1
R	0,07	0,08
I	0,074	0,076
T	0,0756	0,076
H	0,07572	0,07576
M	0,07596	0,07600
E	0,075968	0,075972
T	0,0759712	0,075972
I	0,07597152	0,07597168
C	0,075971536	0,075971552

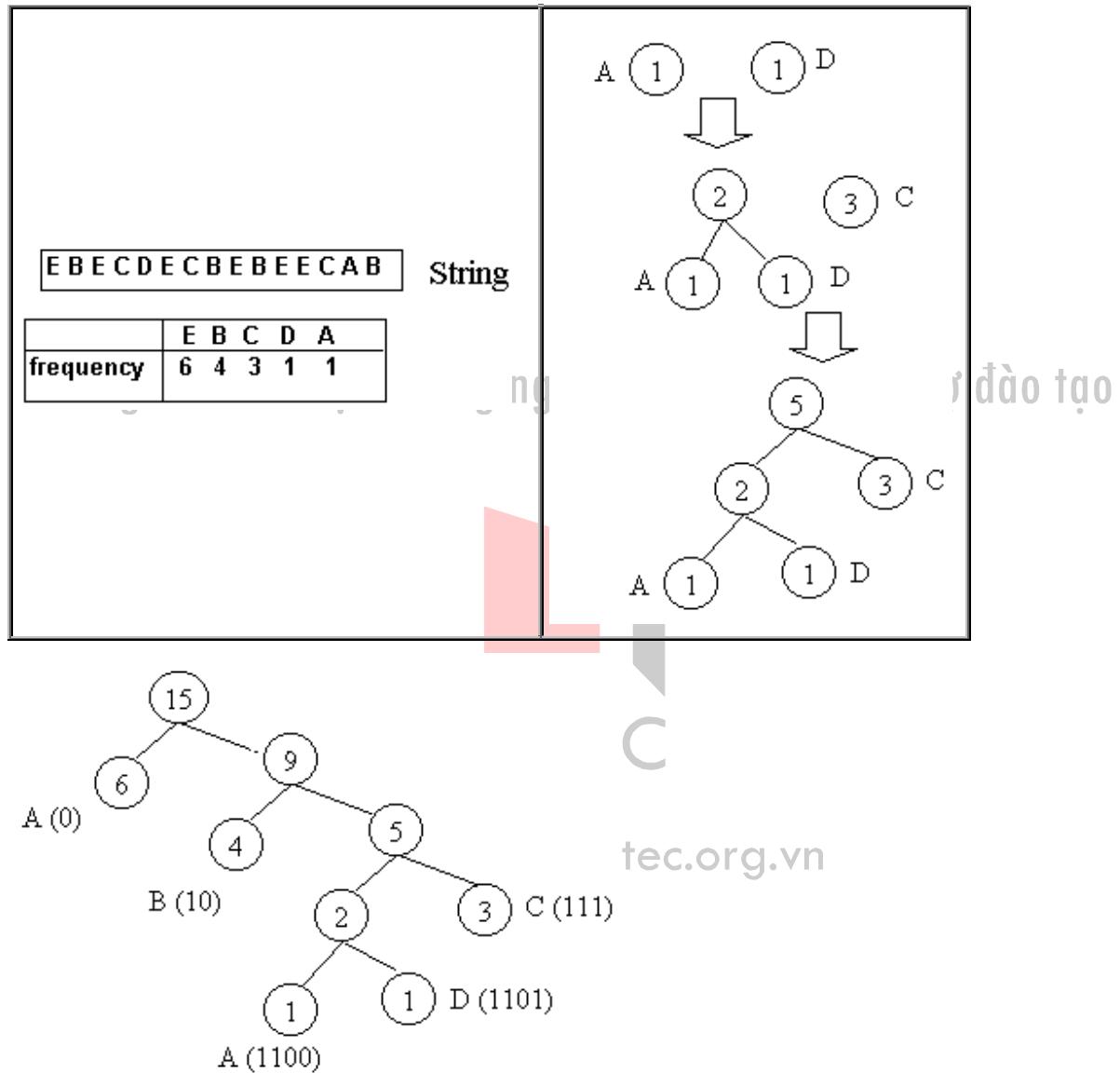
The output number is 0,075971536.

### Huffman compression

Huffman compression, named after D. A. Huffman It handles the problem of data compression by examining the input files and seeing which characters occur the most frequently. The Huffman algorithm then assigns bit codes to the individual characters, using short bit codes for the most common characters and longer bit codes for less common characters. At the front end of the output file, a header is written, giving the bit equivalents for each character. The bit codes for each input character are then written to the output file. The uncompression algorithm reads the encoded file and reverses the process, translating bit codes back into the original characters according to the translation information in the header.

Huffman's approach uses a binary tree to convert the individual characters to codes. The idea is

simple. Consider a binary tree in which the data items reside only in the leaves (or external nodes). The path from the root of the tree to a given leaf can be taken as the binary code for the data in that leaf by recording descent to the left as a 0, and descent to the right as 1.



### Comparison with other algorithms

The Huffman encoding algorithm, when compared to other image compression algorithms, is quite good. Run length encoding is probably easier to implement, but its compression ratios are only good for certain types of files (like images) where long sequences of repeated data are

present. Even when working with a favorable format (an image), the compression ratios achieved by run length encoding are very limited.

Entropy coding is probably harder than Huffman to implement. A nice feature of entropy encoding is that the dictionary used to encode the data does not need to be stored in the output file, thereby saving space. Unfortunately, this type of compression is limited by the size of its data dictionary. It becomes impractical to use entropy encoding if the dictionary's size is rather large (the largest practical size is approximately 256). A well known use of entropy encoding is the Lempel/Ziv/Welch algorithm, which is the basis of the Graphics Interchange Format (GIF) image compression algorithm.

Area coding is an improved form of run length coding. It is primarily used to compress images, since it's algorithm (group the image into rectangular regions containing similar or identical image characteristics) naturally represents the two dimensions of an image. Although area coding can achieve high compression ratios, it is non-linear and therefore cannot be implemented at a hardware level. For that reason, area coding is not highly competitive with the previously discussed image compression algorithms.

### **JPEG Compression**

The JPEG standard defines four variations for compression of images. These are known as follows Sequential DCT based mode: This is the JPEG baseline format.

Sequential lossless mode: This format is defined for applications in which it would be desirable to have no image detail lost in the compression process. This is not widely used as the compression statistics for this method is modest compared to the other styles.

Progressive DCT based mode: This format is very similar to the Baseline mode, however instead of processing information in one scan of the image, this mode processes in a series of scans, each one better defining the image from the last scan that was processed.

Hierarchical mode: This format allows a number of different resolution images to be held within one file. This means that only one image file needs to be made for a number of resolutions it could be viewed at. Again it uses more than one scan to define the image, the decoder simply uses scans in order until it has reached the resolution it requires.

JPEG Baseline is the defacto standard that is used, and is the version for which this project is based. The reasons for this are that is the simplest algorithm as well as being the most widely supported because the JPEG standard specifies that all JPEG decoders MUST support the baseline version, even if it uses some of the other features within the JPEG standard.

### **Entropy Coding**

Entropy Coding is a form of compression that relies on developing "code" words in a library for the input values or "symbols".

A very simple implementation of entropy coding would be to describe four kinds of weather by a 2 bit binary code as follows.

Symbol	Bit Code
Clear	00
Cloudy	01
Rainy	10
Snowy	11

This would allow a coding algorithm to describe the weather with two bits of information per sample. This is a "fixed length" code. Fixed length codes are best applicable to applications where each of the symbols are equally likely, that is to say where each of the weather types is 25% likely in the above example.

In most real world applications this is not the case, each of the symbols actually have different probabilities of occurring. This leads to the use of "variable length" codes. If an event is most likely to happen more often than the other events, then it makes sense to use a code word that is actually shorter, or requires less information, than the other event codes.

Using the weather example again, say that each event has a probability as follows, we can generate variable length codes

Symbol	% Prob	Bit Code
Clear	50	0
Cloudy	25	10
Rainy	12.5	110
Snowy	12.5	111

Now we can determine the average bit length for a code per symbol.

As can be seen, an average of 1.75 bit/symbol is more efficient than the 2 bits/symbol that the fixed length code produced.

The codes produced for this example are found using Huffman coding, which is one of the two options that the JPEG standard allows for. The other is known as Arithmetic Coding. This project uses Huffman Coding for the following reasons.

Huffman Coding is simpler.

Arithmetic Coding is patented and hence has restrictions on its use.

## MPEG Compression Algorithm

### MPEG Compression Techniques

MPEG compression is accomplished by four basic techniques: pre-processing, temporal prediction, motion compensation, and quantization coding. Pre-processing filters out non essential visual information from the video signal, information that is difficult to encode but not an important component of human visual perception. Motion compensation takes advantage of the fact that video sequences are most often highly correlated in time: each frame in any given sequence may be similar to the preceding AND following frames.

The compressed digital video encoder scans subsections within each frame, called macro blocks, and identifies which ones will not change position from one frame to the next. The encoder also identifies predictor macro blocks while noting their position and direction of motion. Only the relatively small difference, called the motion compensated residual, between each predictor block and the affected current block is transmitted to the receiver. The receiver/decoder stores the information that does not change from frame to frame it's buffer memory and uses it periodically to fill in the blanks, so to speak.

A mathematical algorithm called the Discrete Cosine Transform (DCT) reorganizes the residual difference between frames from a spatial domain into an equivalent series of coefficient numbers in a frequency domain that can be more quickly transmitted. Quantization coding converts these sets of coefficient numbers into even more compact representative numbers. The encoder refers to an internal index or code book of possible representative numbers from which it selects the code word that best matches each set of coefficients. Quantization coding also rounds off all coefficient values, within a certain range of limits, to the same value. Although this results in an approximation of the original signal, it is close enough to the original to be acceptable for most viewing applications.

The MPEG video compression algorithm employs two basic techniques: block-based motion compensation for the reduction of the temporal redundancy, and transform domain (DCT) coding for the reduction of spatial redundancy. The motion compensation technique is applied both in the forward (causal) and backward (non-causal) direction. The remaining signal (prediction error) is coded using the transform-based technique. The motion predictors, called motion vectors, are transmitted together with the spatial information.

The MPEG-2 standard uses the same set of algorithms as MPEG-1, and has an additional support for interlaced video sources and scalability options. Although there are minor differences in the syntax, the MPEG-2 standard is conceptually a super-set of MPEG-1, and therefore we will

describe both as the MPEG standard when the distinction is not necessary.

Motion-compensated prediction assumes that the current picture can be locally modeled as a translation of the pictures of some previous time. In the MPEG standard, each picture is divided into blocks of 16 x 16 pixels, called a macroblock. Each macroblock is predicted from the previous or future frame, by estimating the amount of the motion in the macroblock during the frame time interval. The MPEG syntax specifies how to represent the motion information for each macroblock

## MPEG-2

One of the biggest motivations for creating the new MPEG-2 standard was to introduce the support for interlaced video sources. Since the MPEG-1 standard was targeted at the bit rate of around 1.5 Mbits/s, it was assumed that the source video signal will be digitized at around 352 x 240 for 60 Hz systems (e.g., in U.S.) and 352 x 288 for 50 Hz systems (e.g., in Europe). The standard video signals carry twice the scan lines as the above sampling rates, with an interlaced scanning order. Therefore, the simplest way of creating a half-size digital picture was simply sampling only one field from each frame. The other field was always discarded. Since only one field from every frame is used, these sampled fields form a progressively-scanned video sequence. The MPEG-1 therefore addressed the coding parameters and algorithms for progressively-scanned sequences only. However, it should be noted that the syntax of the MPEG-1 standard does not constrain the bit rate or the picture size to any such values.

As MPEG-2 is targeted for coding broadcast-quality video signals, it is necessary to digitize the source video at its full bandwidth, resulting in both even and odd field pictures in the sequence. Since these two fields are separated by a time interval, coding the sequence using the MPEG-1 algorithm does not produce good quality pictures as MPEG-1 assumes that there is no time difference between successive lines in the picture. The MPEG-2 standard provides a means of coding interlaced pictures by including two field-based coding techniques: field-based prediction and field-based DCT.

In MPEG-2, the term picture refers to either a frame or a field. Therefore, a coded representation of a picture may be reconstructed to a frame or a field. During the encoding process, the encoder has a choice of coding a frame as one frame picture or two field pictures. If the encoder decides to code the frame as field pictures, each field is coded independently of the other, i.e., two fields are coded as if they were two different pictures, each with one-half the vertical size of a frame.

In frame pictures, each macroblock can be predicted (using motion compensation) on a frame or field basis. The frame-based prediction uses one motion vector per direction (forward or backward) to describe the motion relative to the reference frame. In contrast, field-based prediction uses two motion vectors, one from an even field and the other from an odd field. Therefore, there can be up to four vectors (two per direction, and forward and backward directions) per macroblock. In field pictures, the prediction is always field-based, but the prediction may be relative to either an even or odd reference field.

### 3.2.13 Memory management algorithms

#### Fixed Partitioning

- Partition main memory into a set of non overlapping regions called partitions
- Partitions can be of equal or unequal sizes
- any process whose size is less than or equal to a partition size can be loaded into the partition
- if all partitions are occupied, the operating system can swap a process out of a partition
- a program may be too large to fit in a partition. The programmer must then design the program with overlays
  - when the module needed is not present the user program must load that module into the program's partition, overlaying whatever program or data are there
- Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This is called internal fragmentation.
- Unequal-size partitions lessens these problems but they still remain...
- Equal-size partitions was used in early IBM's OS/MFT (Multiprogramming with a Fixed number of Tasks)

#### Placement Algorithm with Partitions

VITEC

- Equal-size partitions
  - If there is an available partition, a process can be loaded into that partition because all partitions are of equal size, it does not matter which partition is used
  - If all partitions are occupied by blocked processes, choose one process to swap out to make room for the new process
- Unequal-size partitions: use of multiple queues
  - assign each process to the smallest partition within which it will fit
  - A queue for each partition size
  - tries to minimize internal fragmentation
  - Problem: some queues will be empty if no processes within a size range is present

- Unequal-size partitions: use of a single queue
  - When its time to load a process into main memory the smallest available partition that will hold the process is selected
  - increases the level of multiprogramming at the expense of internal fragmentation

### **Dynamic Partitioning**

- Partitions are of variable length and number
- Each process is allocated exactly as much memory as it requires
- Eventually holes are formed in main memory. This is called external fragmentation
- Must use compaction to shift processes so they are contiguous and all free memory is in one block

• Used in IBM's OS/MVT (Multiprogramming with a Variable number of Tasks)

### **Dynamic Partitioning: an example**

- A hole of 64K is left after loading 3 processes: not enough room for another process
- Eventually each process is blocked. The OS swaps out process 2 to bring in process 4
- another hole of 96K is created
- Eventually each process is blocked. The OS swaps out process 1 to bring in again process 2 and another hole of 96K is created...
- Compaction would produce a single hole of 256K

### **Placement Algorithm**

- Used to decide which free block to allocate to a process
- Goal: to reduce usage of compaction (time consuming)
- Possible algorithms:
  - Best-fit: choose smallest hole
  - First-fit: choose first hole from beginning
  - Next-fit: choose first hole from last placement

### **Placement Algorithm: comments**

- Next-fit often leads to allocation of the largest block at the end of memory
- First-fit favors allocation near the beginning: tends to create less fragmentation than Next-fit
- Best-fit searches for smallest block: the fragment left behind is small as possible
  - main memory quickly forms holes too small to hold any process: compaction generally needs to be done more often

### **Replacement Algorithm**

- When all processes in main memory are blocked, the OS must choose which process to replace
  - A process must be swapped out (to a Blocked-Suspend state) and be replaced by a new process or a process from the Ready-Suspend queue
- We will discuss later such algorithms for memory management schemes using virtual memory

### **Buddy System**

- A reasonable compromise to overcome disadvantages of both fixed and variable partitioning schemes
- A modified form is used in Unix SVR4 for kernel memory allocation
- Memory blocks are available in size of  $2^K$  where  $L \leq K \leq U$  and where
  - $2^L$  = smallest size of block allocatable
  - $2^U$  = largest size of block allocatable (generally, the entire memory available)
- We start with the entire block of size  $2^U$
- When a request of size  $S$  is made:
  - If  $2^{U-1} < S \leq 2^U$  then allocate the entire block of size  $2^U$
  - Else, split this block into two buddies, each of size  $2^{U-1}$
  - If  $2^{U-2} < S \leq 2^{U-1}$  then allocate one of the 2 buddies
  - Otherwise one of the 2 buddies is split again
- This process is repeated until the smallest block greater or equal to  $S$  is generated
- Two buddies are coalesced whenever both of them become unallocated

- The OS maintains several lists of holes
  - the i-list is the list of holes of size  $2^i$
  - whenever a pair of buddies in the i-list occur, they are removed from that list and coalesced into a single hole in the  $(i+1)$ -list
- Presented with a request for an allocation of size  $k$  such that  $2^{i-1} < k \leq 2^i$ :
  - the i-list is first examined
  - if the i-list is empty, the  $(i+1)$ -list is then examined...

### Buddy Systems: remarks

- On average, internal fragmentation is 25%
  - each memory block is at least 50% occupied
- Programs are not moved in memory
  - simplifies memory management
- Mostly efficient when the size  $M$  of memory used by the Buddy System is a power of 2
  - $M = 2^U$  "bytes" where  $U$  is an integer
  - then the size of each block is a power of 2
  - the smallest block is of size 1
  - Ex: if  $M = 10$ , then the smallest block would be of size 5

### 3.2.14 Compiler algorithms

A **compiler** is a translator whose source language is a high-level language and whose object language is close to the machine language of an actual computer. The typical compiler consists of an analysis phase and a synthesis phase.

In contrast with compilers an **interpreter** is a program which simulates the execution of programs written in a source language. Interpreters may be used either at the source program level or an interpreter may be used to interpret an object code for an idealized machine. This is the case when a compiler generates code for an idealized machine whose architecture more closely resembles the source code.

There are several other types of translators that are often used in conjunction with a compiler to facilitate the execution of programs. An **assembler** is a translator whose source language (an

assembly language) represents a one-to-one transliteration of the object machine code. Some compilers generate assembly code which is then assembled into machine code by an assembler. A **loader** is a translator whose source and object languages are machine language. The source language programs contain tables of data specifying points in the program which must be modified if the program is to be executed. A **link editor** takes collections of executable programs and links them together for actual execution. A **preprocessor** is a translator whose source language is an extended form of some high-level language and whose object language is the standard form of the high-level language.

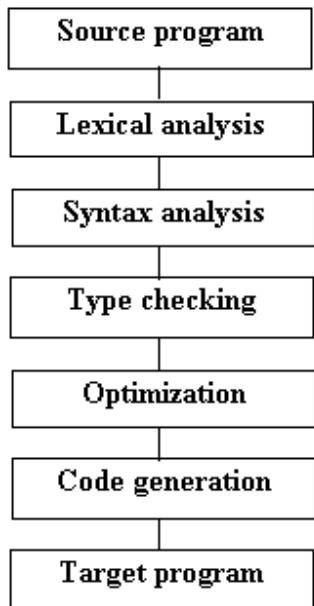
The typical compiler consists of several phases

- 1) Lexical analysis
- 2) Syntax analysis
- 3) Type checking
- 4) Optimization
- 5) Code generation

### **Parser**

A parser is a program that converts a list of input tokens, usually characters, into a value of the appropriate type. A simple example might be a function to find the integer value represented by a string of digits. A more complex example might be to translate programs written in a particular concrete syntax into a suitable abstract syntax as the first stage in the implementation of a compiler or interpreter. There are two common ways to write a parser in a functional language:

<http://www.vitec.org.vn>



## Mạch Công nghệ thông tin và Hỗ trợ đào tạo

The main task of a compiler is to map programs in a give source language into a target language.

The source language is a programming language and the target language is machine code.

It also detects if the source program conforms to the rules of language

A language is formally defined by a set of terminal symbols

a set of non terminal symbols

VITEC

a set of syntactic rules

a start symbol

<http://www.vitec.org.vn>

### Language

A language is a set of strings

Each string has a structure that is described by a tree.

Structure rules for sentences are defined by a grammar.

A grammar defines the sets of the sequences of terminal symbols which can be derived from the start symbol by successive application of productions

### Lexical analysis

The *lexical phase* (scanner) groups characters into lexical units or tokens. The input to the lexical

phase is a character stream. The output is a stream of tokens. Regular expressions are used to define the tokens recognized by a scanner (or lexical analyzer). The scanner is implemented as a finite state machine.

### Syntax analysis

The *parser* groups tokens into syntactical units. The output of the parser is a parse tree representation of the program. Context-free grammars are used to define the program structure recognized by a parser. The parser is implemented as a push-down automaton.

### Type checking

The *contextual analysis phase* analyzes the parse tree for context-sensitive information often called the *static semantics*. The output of the contextual analysis phase is an annotated parse tree. Attribute grammars are used to describe the static semantics of a program.

### Optimizer

The *optimizer* applies semantics preserving transformation to the annotated parse tree to simplify the structure of the tree and to facilitate the generation of more efficient code.

### Code generation

The *code generator* transforms the simplified annotated parse tree into object code using rules which denote the semantics of the source language.

## 3.2.15 Programming Languages

### Purpose of studying programming languages

The following shows why programming languages are studied

- 1) increase our capacity to express ideas (both programming and in general)
- 2) increase ability to select and learn a PL
- 3) better understanding of implementation issues in programming
- 4) increase ability to design new Programming languages
- 5) better understanding of computational theory

## **Language evaluation criteria**

### **Readability**

Simplicity of instructions  
Orthogonality of instructions  
Control statements  
Data Types and Structures  
Syntactic Issues -- identifier rules, reserved or special words, form and meaning

### **Writability**

Flexibility and availability of control structures, data types  
Simplicity and Orthogonality  
Support for Abstraction  
Expressitivity

### **Reliability** Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Readability and Writability  
Type Checking  
Exception Handling  
Aliasing

### **Cost**

Training/Learning Curve  
Time to write programs  
Compilation Speed/Efficiency  
Execution Speed/Efficiency  
Maintenance



### **Tradeoffs**

<http://www.vitec.org.vn>

Readability vs. Flexibility  
Cost vs. Expressitivity  
Type Checking vs. Abstraction and Flexibility

### **Design effects**

- 1) Computer Architecture - usually will concentrate on Von Neumann architectures
  - 2) Other architectures such as parallel processing (MIMD or SIMD) or pipelining might require very different languages
  - 3) Program Methodologies
- Procedural vs. Functional vs. Logical vs. Object-oriented

## **Implementation**

### 1. Compilation

- Translate high-level program to machine code
- Slow translation
- Fast execution

### 2. Pure interpretation

- No translation
- Slow execution
- Becoming rare

### 3. Hybrid implementation systems

- Small translation cost
- Medium execution speed

## Syntax and semantics

Syntax - form or structure of expressions or statements for a given language

Semantics - meaning of the expressions

Language - group of words that can be combined and the rules for combining those words

Lexeme - lowest level syntactic unit in the language

## **Grammar**

Language Recognizer - given a sentence, is it in the given language?

Language Generator - given a language, create legal and meaningful sentences

We can build a language recognizer if we already have a language generator

Grammar - a description of a language - can be used for generation

## **Classification of languages**

Regular

<http://www.vitec.org.vn>

Context-Free

Context-Sensitive

Recursively Enumerable

Context-Free grammars include those which can be generated from a language generator (grammar)

These include natural languages and programming languages

## **BNF (Backus Normal Form)**

Equivalent to a context-free language

Used to specify the grammar of a language and can then be used for language generation or recognition

Contains terminals, non-terminals and rules which map non-terminals into expressions of other non-terminals and terminals

## **BNF Grammar**

Mathematically, a BNF Grammar is given as  $G=\{\text{alphabet}, \text{rules}, \langle\text{start}\rangle\}$   
alphabet -- those symbols used in the rules (both terminals and non-terminals)  
rules -- map from a non-terminal to other elements in the alphabet  
 $\langle\text{start}\rangle$  -- a non-terminal which must be on at least 1 rule's left hand side

Example of grammar rules

```
<stmt> -> <single_stmt>
| begin <statelist> end
```

Notice the use of both terminals and non-terminals on the right side

Recursion is used as necessary

```
<ident_list> -> ident
```

```
| ident, <ident_list>
```

## **Parse Trees**

A hierarchical structure displaying the derivation of an expression in some grammar

Leaf nodes are terminals, non-leaf nodes are non-terminals

Parser - takes a sentence and breaks it into its component parts, deriving a parse tree. If the parser cannot generate a parse tree, then the sentence is not legal

## **Extended BNF Grammar**

3 common extensions to BNF:

[ ] - used to denote optional elements (saves some space so that we don't have to enumerate options as separate possibilities)

{ } - used to indicate 0 or more instances

( ) - for a list of choices

These extensions are added to a BNF Grammar for convenience allowing us to shorten the grammar

## **Functional programming**

Functional programming is a style of programming that emphasizes the evaluation of expressions, rather than execution of commands. The expressions in these languages are formed by using functions to combine basic values. A functional language is a language that supports and encourages programming in a functional style.

### **Strictness**

- In a strict language, the arguments to a function are always evaluated before it is invoked.

As a result, if the evaluation of an expression  $\text{exp}$  does not terminate properly (for example, because it generates a run-time error or enters an infinite loop), then neither will an expression of the form  $f(\text{exp})$ . ML and Scheme are both examples of this.

- In a non-strict language, the arguments to a function are not evaluated until their values are actually required. For example, evaluating an expression of the form  $f(\text{exp})$  may still terminate properly, even if evaluation of  $\text{exp}$  would not, if the value of the parameter is not used in the body of  $f$ . Miranda and Haskell are examples of this approach.

## Declarative Programming Languages

Most declarative programming languages stem from work in artificial intelligence and automated theorem proving, areas where the need for a higher level of abstraction and a clear semantic model of programs is obvious.

### Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

The basic property of a declarative programming language is that a program is a theory in some suitable logic. This property immediately gives a precise meaning to programs written in the language. From a programmers point of the basic property is that programming is lifted to a higher level of abstraction. At this higher level of abstraction the programmer can concentrate on stating *what* is to be computed, not necessarily *how* it is to be computed. In Kowalski's terms where **algorithm = logic + control**, the programmer gives the logic but not necessarily the control.

Declarative languages differ from imperative languages in that they are based on an abstract formalism, thus divorcing their semantics from the machine on which they are run. The statements in such a language are declarative since they can be understood without reference to machine level terms, such as side-effects. For this reason, programs in a declarative language can act as a specification, since they can be regarded as the formal description of a problem. Other advantages of such languages are that programs can be developed and tested bit by bit, and that they can be systematically synthesized or transformed.

LISP [McCarthy 1965a], arguably the first implemented declarative language, was based on the abstract formalism of the lambda calculus.

## Logic Programming

For practical applications there exists one logic programming in use: Prolog. Prolog is used for a wide variety of applications in artificial intelligence, knowledge based systems, and natural language processing. Programs are evaluated by proving queries with respect to the given program.

From a programming point of view Prolog provides two features not present in functional

languages, built-in search and the ability to compute with partial information. This is in contrast to functional languages where computations always are directed, require all arguments to be known and give exactly one answer

Accordingly, Prolog and most other logic programming languages only provide declarative programming in the weak sense where the programmer also need to provide some control information to get an efficient program from the declarative problem description. The method used by many Prolog programmers is rather ad hoc: First the problem is expressed in a nice declarative fashion as a number of predicate definitions. Then the program is tested and found slow. To remedy the problem the original program is rewritten by adding control information (cuts) as annotations in the original code and doing some other changes like changing the order of arguments to predicates. The resulting program is likely to be quite different from the original one -- and less declarative

### Logical programming characteristics

Symbolic Logic - propositions which evaluate to either True or False

Relationships - rules and axioms

Methods - Resolution and Unification

These three ideas form the basis for logical (declarative) programming which is very different from imperative and functional programming

Prolog is an example of such a PL which has often been used in AI

sunny -- it is sunny outside

Monday -- today is Monday

man(jake) -- jake is a man

like(bob, redheads) -- bob likes redheads (or possibly redheads like bob)

dog(spot) -- spot is a dog

poodle(spot) -- spot is a poodle

dog(x) -- true if x is a dog

<http://www.vitec.org.vn>

In logical programming, rules are in consequence form (spot is a dog if spot is a poodle)

dog(x) if poodle(x)

mortal(x) if man(x)

Given a series of propositions, predicates and rules in clause form, resolution can prove things are true by contradiction

Introduce the negation of what you are trying to prove

Show that with the negation, a contradiction arises with what we already know to be true

While Prolog uses predicate calculus form, it does not actually use resolution, instead it uses backward chaining:

We know A, B, C and E are true

A and B and C --> D

D and E --> F

G and H --> F

Want to prove F, how? Backward chaining

To prove F: prove G and H, or D and E. We know E, so now prove D by proving A...

Translate all statements into clause form:

clauses are propositions that are composed of a disjunction of lesser terms, or an implication with disjunctions on the left side and conjunctions on the right:

A1 or A2 or A3 ... :- B1 and B2 and B3 ...

with no existential quantifiers and all universal qualifiers replaced by variables

distribute negations using demorgan's law converting and's to or's and or's to and's  
separate "and" clauses into independent clauses

## Unification

Rules can be represented using variables instead of actual values:

mortal(Socrates) :- man(Socrates) is better represented as mortal(x) :- man(x)

In resolution, if we have a rule with foo(x) and a predicate foo(fred) then we can unify x to fred during the course of this proof

If unification fails, we can try to unify the variable to another value (x to harry)

## Logic programming

Programs are a series of declarations - statements, propositions, rules

No assignment statements, no control statements

The only control strategies are resolution and unification - these are built-into the language and the programmer does not do anything about it

Introduce a term of what you want to prove and backward chaining/unification do the rest

## Prolog

Developed in Edinburgh & Marseille in the early 70's

Programs - database of facts (predicate statements such as father(fred, frank) and atomic statements such as Tuesday) and rules (conjunction of propositions)

Proof is created by introducing what is to be proven and using backward chaining through the rules

Uses of logic

Database Management Systems

Expert Systems - particularly those dealing with T/F worlds (math, simple logic)  
Natural Language Processing and logic-based semantic networks  
Educational/Tutorial systems - teaching logic

## Functional Programming

The variables and memory usage is less visible making programming easier (at least in some situations)

It has simpler syntactic structures to deal with (since everything is a list)

The concurrency is easier to design and implement

The interpreted nature makes large systems easier to build and exploits recursion as much as possible, more so than imperative languages

## LISP

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Early in AI research, there was a need for symbolic computing

A functional approach was taken

Rather than a series of imperative statements executed in order, all instructions are functions which return a value which is then used in another function - this method was chosen due to the strong tie between the work being performed and mathematical functions

Lisp began as a purely functional language although it picked up elements from other PLs

Mathematical functions

Maps from a domain set to a range set

Written in prefix notation (+ 5 7)

Lambda expression - an unnamed function in which the parameter is specified

e.g. (lambda (x) (\* x x)) (5) --> 25

Functional forms - can take functions as parameters (this allows for functions within functions)

Construction - use a list of functions as parameters

Apply to all - applies a function to a list of parameters returning a list

Functional composition

$$= 3*(x+2)$$

$$f(x) = x + 2 \quad g(x) = 3*x$$

$$h = f \circ g \text{ or } h(x) = f(g(x))$$

Construction - functions as parameters applied to a single value returning a list of values as in [f, g](5) = (7, 10)

Apply-to-all - one function with a list of parameters returning a list as in apply(f, (2, 5, 12)) = (4, 7, 14)

## LISP

A typeless language

Data Structures: atoms and lists

Atom - any literal (char, string, number)

List - nil or ( x ) where x is an atom, a series of atoms, or a list, or any combination of these

Example of a list (A (B C D) E F)

List structures make use of pointers where each item has an info field and a next field where info might point to an atom or another list

All program code composed of functions

### Original LISP

The original idea was to make Lisp a universal Turing Machine where both data and instructions would be treated in the same way -- therefore everything is a list including programs

Lisp's primary function is EVAL which evaluates a list or atom at run-time

Since Lisp code is represented as lists, Lisp is interpreted

### LISP functions

As stated earlier, lambda functions are nameless functions

Lisp programs can be composed of lambda functions which are executed at the command line

Lisp also has a function for defining and binding functions (that is, there are facilities for naming functions so that you can use those by calling them from other functions)

### Application

Lisp was used extensively for research in Expert Systems, Natural Language Understanding, Machine Learning, Knowledge Representation, Speech/Vision Understanding, Robotics, Tutorials

Mostly used in AI research

Natural Language Understanding (easy parsing partially due to recursive nature)

Expert Systems (easy rule format)

Knowledge Representation (symbolic capabilities)

Machine Learning (dynamic storage)

etc...

Also used to teach functional programming and mathematically oriented programming

Used to implement EMACS, MACSYMA and even some operating systems

### Object oriented programming

Objects first introduced in Simula 67, a language mostly applied for simulations Developed more fully in Smalltalk which brought objects into the public focus C++ is a descendent of C and

Smalltalk CLOS implemented to provide Common Lisp with full Object capabilities

Ada 95 also has full Object capabilities Java is a new generation of OOPL, safer, easier?

## Objects

These are the characteristics of objects

- 1) Organized hierarchically with inheritance
- 2) Structures can be easily modified
- 3) Message-Passing rather than procedure calls
- 4) Encapsulation enforced (unlike ADTs where some PLs do not enforce this)
- 5) Generic nature of methods to allow for different typed objects to use the same named operations

## Inheritance

Solution to the modification problem by inheriting what was implemented for prior (parent) classes Derived class - subclass, inherits from another class which is the superclass or parent class. In some languages (C++), what is to be inherited can be dictated by the programmer

Some languages (C++) have multiple inheritance, others have single inheritance

## Polymorphism

Polymorphic variables can reference any subclass of a given class

Dynamic type binding of variable is done when method is called This allows for flexibility in system construction and maintenance so that variables do not have to be bound to specific types, types can change as new objects/subclasses are added or changed

## Message passing

Model which resembles the world with objects communicating via messages

Objects have their own local memory and processes (methods)

An object's method(s) is activated as a response to a message from another object

Return response is another message

Messages can have parameters to allow for more specific communication between objects

## **Types of OO languages**

Pure OOP - all computation done by message passing between objects (e.g. Smalltalk)

Hybrid OOP - objects make up part of the language in which computation is done by message passing and procedure/function calls (e.g. C++, Java,

CLOS/Common Lisp)

Objects can be statically or dynamically bound and are typically referenced by pointers

Classes - object abstractions organized hierarchy.

Instances - objects created from classes

## **Design**

Exclusivity of Objects - are all types classes or are there other types (pure vs. hybrid)

Subclass relationships - is a subclass a descendant of the class (an is-a relationship), or a different relationship

Inheritance and information hiding - how are these provided and enforced?

Type checking and polymorphism - is this done?

Single vs. multiple inheritance - which is allowable?

Allocation and Deallocation of Objects - when and how

Dynamic vs. Static Binding of Objects

<http://www.vitec.org.vn>

## **Smalltalk**

Programs are mostly objects and messages

Code is embedded into the objects

Variables are local (private) or global (shared) - all vars are pointed too by ptrs

Messages - 2 parts - object to receive the message and the message itself (method name) with possible params (also pointers)

## **Evaluation of Smalltalk**

Small language, simple syntax

Mathematically, very limited  
Less efficient than imperative languages  
Because of dynamic type checking, errors are often not caught until run-time  
Useful for games and graphics  
First real OO language, pioneered many of the ideas now embedded in OOP

## Java

All non-primitive types are objects, this includes:

Strings  
Arrays  
Exceptions

Java has no records (can be created as objects)

Some methods are available to all classes, these are called basic methods, and include

constructors, mutators and accessors All classes are descendants of the class OBJECT, all classes must have a parent

## Java packages

Two forms of encapsulation in Java - classes (logical encapsulations) and packages (physical encapsulations)

Similar classes are grouped into packages

Built-in class packages include java.applet, java.io, java.awt, java.lang and java.util

A class C in package P is specified as P.C

To use a package, it must be imported as in

import Name.Class; to use Name's Class or

import Name.\*; to use everything

## Java Classes and Inheritance

Java uses single inheritance but multiple inheritance can be simulated using the virtual class called Interface

Java does not support generic methods but this can be done by using object inheritance

Java allows for abstract classes and methods where derived types of these abstract classes must implement the abstract methods

(i.e. abstract classes outline what the class will do but the programmer must then finish the

implementation)

This - keyword to refer to the current object (used as a self-assignment, alias or to call the constructor of the same class)

Instanceof - a run-time test to determine if an expression is an instance of a class

Static - allows a field in a class to become global and shared

## Java Applets

Code placed into WWW documents to be executed by the Java interpreter

All applets are member functions of a class defined as a subclass of the built-in class Applet

Applet class has many previously defined functions common in WWW processes

Java objects are very similar to C++ and allow for similar power in programming

Unlike C++, all Java objects are explicit heap-dynamic, allocated with the new command and deallocated automatically by the garbage collector

Unlike C++, all Java objects are referenced by a reference variable, which is like a pointer, but safer

## Control structures

Programs have 4 basic control structures:

- 1) Sequential
- 2) Selection
- 3) Repetition
- 4) Unconditional Branching (which may either have a subsequent return or no return)

## Compound statements

<http://www.vitec.org.vn>

Compound statements make control statements easier to design so that groups of instructions can be formed inside of a selection, repetition or branch

ALGOL 60 - first PL to introduce blocks using begin-end statements

C, C++, Java - uses { } as delimiters

## Selection statements

Gives a program the ability to choose which instruction(s) to next execute based on conditions  
Essential for every language

Types:

- 1-Way Selection
- 2-Way Selection

## 3-Way Selection Multiple Selection

### One way selection

If-Then statement without an Else clause

Based on condition, either execute Then clause or skip Then clause

FORTRAN's IF statement was a One-Way (no Else clause, no nesting)

### Two way selection

Design Issues:

What is the form and type of expression that controls the statement?

Can a single stmt or compound stmts be used

How is nesting enabled?

### N way selectors

Case statement: compare variable against a list of options, if matches then execute the statement and end else try next list of options. In Pascal and Modula-2, variable can be any ordinal type, in others, it must be an integer

C - uses a Switch statement where, if one list matches, the statement is executed but the remainder of the list is checked rather than ending unless a break statement is included

### Repetition statements

Every Programming language has included some form of repetition

How is repetition controlled? Should testing of whether to repeat occur before or after the loop body is executed? (pre vs. post test)

Where should the control mechanism appear in the loop?

### Iteration (Counter controlled loops)

Control is maintained by using a loop variable which is started at the initial value, compared to the terminal value and incremented each time through the loop by a step size.

These types of loops are often supported directly in computer hardware (e.g. automatic incrementing of loop variable)

### Design

What is the Type and Scope of this loop variable?

What value does the loop variable have after termination if any?

Can the loop variable be changed during the loop?

Should the test for completion be pre- or post-test?

Should the initial and finding values be tested once or each time through?

### **Logically controlled loops**

For situations where the number of repetitions is not known before the loop proceeds -- these loops use some testable condition

Pretest vs. posttest

Is this type of statement separate from a special kind of counter-controlled loop?

### **Pre-test vs. Post test loops**

For situations where the number of repetitions is not known before the loop proceeds -- these loops use some testable condition

Pretest vs. posttest

Is this type of statement separate from a special kind of counter-controlled loop?

### **Exiting loops**

Should exiting only be permitted at the end when the test returns false? Or can premature exiting (and returning) be permitted?

Ada has loop ... end loop -- a conditionless loop (infinite loop unless a branch is used)

C and Modula-2 have unconditional exit statements (break, exit)

Multiple exits harm readability

### **Problems with Unconditional Branching**

Can make programs unreadable

Creates problems with maintenance thus harming reliability, especially of very large programs

The goto statement is too primitive -- “it is an invitation to make a mess of one’s program”

### **Expressions and statements**

One of the original intentions of computer programs was the evaluation of mathematical expressions

PLs would inherit much from math conventions

Expressions: constructed from mathematical operators, operands, parents and functions

### **Operators**

Unary - 1 operand as in C’s I++

Binary - 2 operands (the usual form)

Ternary - 3 operands, as in C's conditional

Expressions requiring fetching the operator followed by operands, then applying the operator to those operands

Precedence - the order that the operators are evaluated

## Associativity

If two or more operators are of the same precedence, then they are usually applied in a Left-to-Right manner

$$A + B + C = (A + B) + C$$

\*\* (exponent) is usually Right-to-Left

$$A ** B ** C = A ** (B ** C)$$

In Ada, \*\* is not associative which means that you must parenthesize the expression to either  $(A ** B) ** C$  or  $A ** (B ** C)$

## Side effects

When a function changes a parameter which is passed back or changes a global variable

Consider the expression  $A + \text{Fun}(A)$ , if  $\text{Fun}(A)$  does not alter  $A$ , then the order of evaluation is unimportant, but if

$\text{Fun}$  changes  $A$ , then  $A + \text{Fun}(A) \neq \text{Fun}(A) + A$

In Pascal, side effects can occur using global variables or var params in functions

Side effects can be disallowed (no var params, no global vars) or evaluate all expressions L-R putting functions at the end of the expression

type conversion

<http://www.vitec.org.vn>

Narrowing - going from a larger type to smaller type (e.g. integer to short int)

Widening - from smaller type to larger

Widening - almost always safe, narrowing - rarely safe (lose information)

Coercion - implicit type conversion initiated by the compiler due to the use of an arithmetic operator (e.g. dividing 2 integers in Pascal yields a real)

Relational and Boolean expressions

Evaluate to True/False rather than numeric

Relational Ops always have lower precedence than arithmetic so that the relational comparison is performed after any arithmetic operations (ex.  $a+1 > b-2$ )

Boolean operators AND, OR, NOT - usually have lower precedence than Relational Ops except possibly NOT (highest precedence in Ada, lowest in FORTRAN)

Assignment statement

<target\_var> <assignment\_op> <expr>

FORTRAN, BASIC, PL/I, C, C++ all use =

This can be confusing where = also stands for equivalence, consider A = B = C in PL/I  
ALGOL 60, Pascal, Modula-2, Ada use :=

In Java, lhs=rhs means that the lhs variable is set equal to the value of rhs if they are primitive types , otherwise, the pointer lhs is set to point to the item pointed to by rhs

**Phase Structure Grammar** is a set of things like NOUN-PHRASE and CONDITION

we use them in our definition of the structure of a valid sentence in the language, but they do not appear in the actual strings of the language these are the non-terminals of the grammar

### Context-free grammar

An important step in the automatic processing of natural or formal languages consists in breaking down the input stream of words or characters into sentences in the relevant language and then analyzing those sentences syntactically.

This can be done for each language on the basis of a grammar which defines:

The set of words (tokens) in the language.

These are identified in the grammar by means of terminal symbols (terminals).

The various categories of token sequences, i.e. phrases which occur when sentences are formed and which are part of the input stream.

These are indicated by non-terminal symbols (non-terminals).

The internal structure of phrases: A set of rules called production rules is provided for this purpose.

A start symbol from the set of non-terminals:

Whereas a grammar of this sort must be constructed inductively for natural languages (this is usually only partly successful), a formal language (command language, programming language) is often a set of sentences generated deductively with a predefined grammar.

A Context Free Grammar (CFG) is a set of recursive rewriting rules (or productions) used to generate patterns of strings.

A Context Free Grammar consists of the following components:

- A set of terminal symbols, which are the characters of the alphabet that appear in the strings

generated by the grammar.

- A set of non-terminal symbols, which are placeholders for patterns of terminal symbols that can be generated by the non-terminal symbols.
- A set of productions, which are rules for replacing (or rewriting) non-terminal symbols (on the left side of the production) in a string with other non-terminal or terminal symbols (on the right side of the production).
- A start symbol, which is a special non-terminal symbol that appears in the initial string generated by the grammar.

To generate a string of terminal symbols from a CFG, we:

- Begin with a string consisting of the start symbol;

· Apply one of the productions with the start symbol on the left hand side, replacing the start symbol with the right hand side of the production;

· Repeat the process of selecting non-terminal symbols in the string, and replacing them with the right hand side of some corresponding production,

until all non-terminals have been replaced by terminal symbols.

A) Context Free Grammar for if.....else.....if statement

<start> ---à if ( <expr> ) <stmt> else <stmt> if ( <expr> ) <stmt>

<stmt> = Statement

<expr> = Expression

Non-terminal symbols are <stmt> and <expr>

Terminal symbols are if and else

<start> is the Start symbol

VITEC

### 3.3 Evaluation of algorithms

---

Algorithms should be evaluated and selected based on certain criteria. If the most appropriate algorithm can be found to solve the problem, the work of programming can be done efficiently and a high degree program can be created.

This section describes the following three algorithm evaluation criteria:

- Evaluation in terms of computational complexity (a criterion for evaluating efficiency)
- Evaluation in terms of validity (a criterion for evaluating reliability)
- Evaluation in terms of representation (a criterion for evaluating elimination of redundancy and improvement of processing speed)

#### 3.3.1 Evaluation by computational complexity

Computational complexity, which is also called computational measure, quantitatively shows the workload needed to perform calculations.

Computational complexity is a measure used to mathematically clarify how much time and memory area a computer requires to perform calculations.

Computational complexity is expressed in  $O$  (order).

Example: If an algorithm deals with data in such a way that the data increases twofold, threefold, fourfold,..... etc, the execution time likewise becomes twofold, threefold, fourfold,..... etc, computational complexity of this algorithm is  $O(n)$ .

##### (1) Types of computational complexity

There are two types of computational complexity

- Time complexity measure: The maximum time that an algorithm needs to process all data
- Space complexity measure: The maximum area that an algorithm needs to process all data

In general, computational complexity assume the worst case. Computational complexity discussed in this section also assumes the worst case, if not otherwise indicated. If average computational complexity is  $O(n \log n)$  and if maximum computational complexity is  $O(n^2)$ , as in the case of quick sorting, average computational complexity should be considered more important.

Computational complexity of an algorithm is represented as data sizes (for example, a square of a size  $n$ ) are not restricted by hardware or software limitations.

##### (2) Computation example

We now take up linear search (sequential search) and binary search algorithms as examples.

Assuming that the data size is  $n$ , the number of times of comparison and computational complexity are as follows:

### ①Linear search (sequential search)

- Minimum number of times of comparison: 1 (computational complexities:  $O(1)$ )
- Average number of times of comparison:  $n/2$  (computational complexities:  $O(n)$ )
- Maximum number of times of comparison:  $n$  (computational complexities:  $O(n)$ )

### ②Binary search

- Minimum number of times of comparison: 1 (computational complexities:  $O(1)$ )
- Average number of times of comparison:  $\lceil \log_2 n \rceil$  (computational complexities:  $O(\lceil \log_2 n \rceil)$ )
- Maximum number of times of comparison:  $\lceil \log_2 n \rceil + 1$  (computational complexities:  $O(\lceil \log_2 n \rceil)$ )

## 3.3.2 Evaluation by validity

The validity of an algorithm is a criterion for making a judgment about whether or not an algorithm satisfies program specifications (module design document, etc)

The validity can be further divided into three categories:

- Partial validity: Whether segments or functions satisfy specifications or not
- Terminability: Whether a program can terminate after a specified number of execution steps, as defined by an algorithm: because an infinite loop must not occur
- Total validity: Whether the whole algorithm satisfies specifications or not

Although various methods are used to verify the validity of an algorithm, it is difficult to verify it completely. If the range of data is specified, the validity can be verified easily by detecting trouble that may occur when data outside the specified range is inputted.

## 3.3.3 Evaluation by representation

Algorithms must be evaluated in terms of not only computational complexity and validity, but also algorithm representation, which is called the elaboration of algorithm representation.

### Example

<http://www.vitec.org.vn>

- The same is repeatedly executed: If repeated steps are defined as a subroutine, they can be described as one single step so that the flow of steps in an algorithm can be presented in a simple easy to understand manner.
- It is necessary to increase the speed of an algorithm: Attention should be paid to steps that are repeated most frequently, and ways to increase the speed should be studied (if quick sorting is used, it maybe necessary to quit the sue of recursive calls and to work out an alternative)

## 3.4 How to design algorithms

---

Based on data obtained by analyzing a problem and the results described in the previous section, an algorithm is designed with the most attention given to how a solution can be had with the highest level of efficiency. There are several methods used to design an algorithm.

Representative methods are:

- Dynamic programming method
- Greedy algorithm method
- Reduction method

### (1) Dynamic programming method

In the dynamic programming method, one problem is considered to be a set of more than one sub-problem.

Steps to follow are:

1. A problem is divided into some sub-problems.
2. A solution to each sub-problem is found.
3. Some sub-problems are put together to make a slightly larger partial problem.
4. Above steps 2 and 3 are repeated until a solution method to the original problem is found.

### (2) Greedy algorithm method

In the greedy algorithm method, which is used to find a solution to an optimization problem, the values of variables are changed by degrees depending on each present condition. For example, to find how the number of coins can be decreased to a minimum to pay a specified amount, the number of large to small value coins is determined using the greedy algorithm method.

#### Example

To pay ₫574,

One 500-yen coin - ₫74 remains

One 50-yen coin - ₫24 remains

Two ten-yen coins - ₫4 remains

Four 1-yen coins - Nothing remains

### (3) Reduction method

In the reduction method, an algorithm originally designed has complexity of  $O(n)$  is subjected to a reduction process of time length  $cn$ , so that it can be reduced to a smaller size to create a new algorithm of  $O(n/x)$ . If  $O(n) > cn + O(n/x)$  cannot be satisfied, this algorithm is meaningless. Therefore, if  $n$  is very small, this algorithm cannot produce the expected results.

## 3.5 Applications of algorithms

Algorithms find usage in many areas. These include

- 1) Routing
- 2) Search engines
- 3) Cryptography
- 4) Data mining

### 3.5.1 Routing

- 1) Static routing

The shortest path to the destination is determined.

Dijkstra's algorithm is used to explore the network wavelike starting at node A and determine at every visited node the shortest path to A and the next node on this path.

Move the wave further at the (border-) node having the shortest distance to A.

- 2) Dynamic routing

Distance vector routing:

Every router sends the packet over the link, using a routing table that belongs to the shortest path.



- 1) Every router maintains distance table with 2 entries per destination:
  - shortest known (estimated) distance to destination (#routers/hops, delay, throughput,...)
  - link corresponding to this distance.
- 2) Every router knows (an estimate of) the distance to its direct neighbors.
- 3) Every router sends regularly its current distance vector to each direct neighbor (distances only, not links).
- 4) On receipt of a distance table from neighbor X the router checks, if it should correct its own entries:
  - increase entry, if all neighbors report larger distances

- decrease entry, if a neighbor reports a smaller distance

### **Link-State Routing:**

Compute shortest paths based on regularly exchanged local knowledge of the topology.

Every Router must

1. Detect its neighbors and their network addresses.
2. Measure the delay (cost) to the neighbors.
3. Compile the results of (1) and (2) into a suitable packet.
4. Send this packet to all other routers.
5. Compute the shortest path to all other routers.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### **3.5.2 Search engines**

Search engines utilize various algorithms for the search and retrieval.

The algorithms are used for:

- 1) pattern matching in textual documents
- 2) constructing an index
- 3) analyzing a search phrase (query)
- 4) measuring the relevance of retrieved documents

<http://www.vitec.org.vn>

#### **Types of search engines**

Search engines can be classified as

- 1) "spiders" or "crawlers"

These constantly visit web sites on the Internet in order to create catalogs of web pages.

The catalogs are created automatically and updated constantly without human interference.

- 2) Directories

These created catalogs of web pages manually. The sites must be submitted, then they are assigned to an appropriate category. Directories can often provide better results than search engines e.g. Yahoo

### **Mechanism of the search engine**

Search engines have three major elements:

- 1) the spider (also called crawler)

This visits a web page, reads (and analyses) it, and then follows links to other pages within the site.

- 2) It returns to the site on a regular basis, such as every month or two, to look for changes.

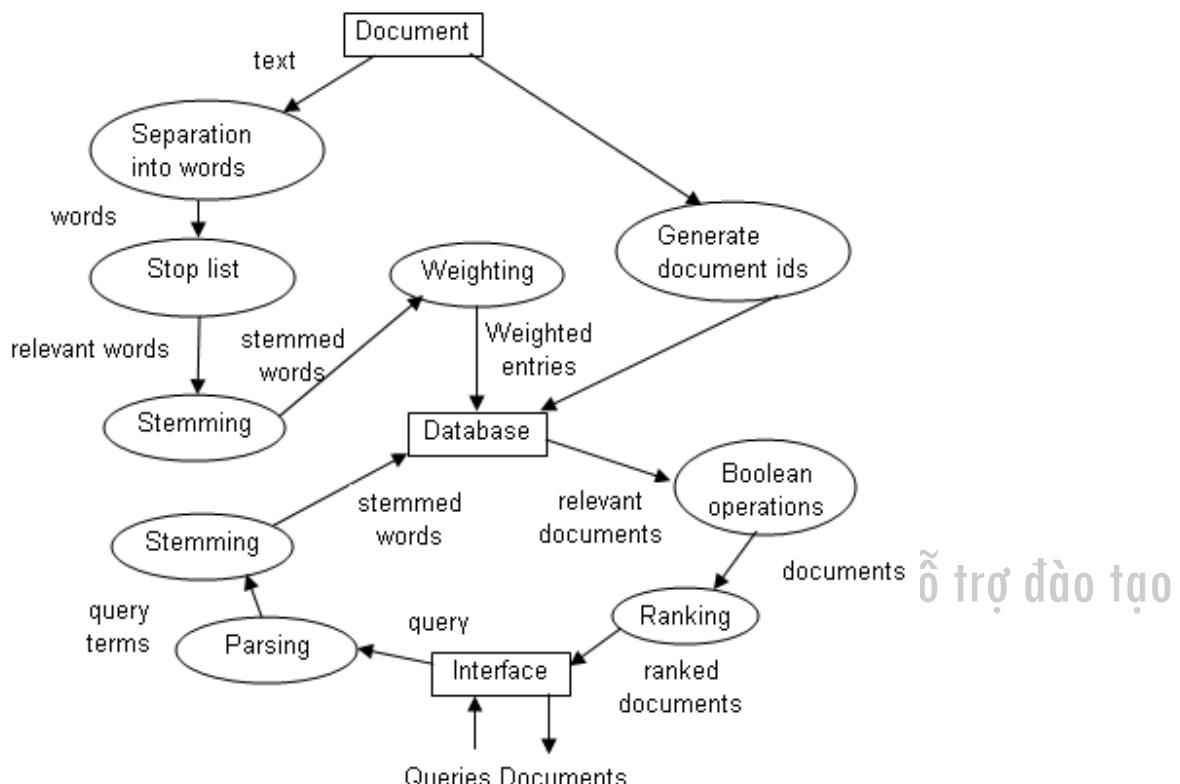
- 3) The index (also called catalog)

It contains information on every web page that the spider finds. It is updated, whenever web page changes are detected. Only the pages listed in the index are available for searching. It uses efficient data structures to support searching.

The search engine software analyses a query, searches the index for documents matching the query, ranks the matching documents in order of relevance.

VITEC

<http://www.vitec.org.vn>



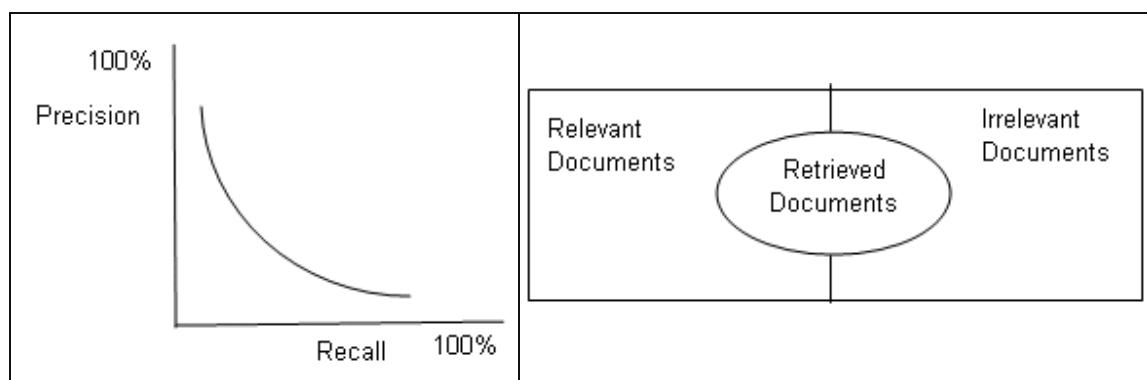
Criteria for assessing the quality of the search engine

Recall = No of relevant retrieved documents

$$\text{Recall} = \frac{\text{No of relevant documents}}{\text{No of relevant documents}}$$

Precision = No of relevant retrieved documents

$$\text{Precision} = \frac{\text{No of relevant documents}}{\text{No of retrieved documents}}$$



### 3.5.3 Cryptography

There are 2 kinds in use today

#### Symmetric key algorithms

This method uses the same key to encrypt and decrypt the message. DES (Data Encryption Standard) is an example of this. Symmetric keys are faster than public key algorithms. In terms of security, the encryption key must be exchanged securely.

Symmetric key Algorithms

Name	Description
DES	Data Encryption Standard It uses a 56 bit key
DESX	This is a modification of the DES to improve the security
Triple-DES	This executes the DES encryption 3 times
IDEA	International Data Encryption Algorithm This uses a 128 bit key
RC2	The key length is usually limited to 40 bits for software
RC4	This is a stream cipher
RC5	This allows a user defined key length, data block size and number of encryption rounds

#### Public key algorithms <http://www.vitec.org.vn>

A separate set of keys is used for encryption and decryption. The encryption key is known as the public key. The decryption key is known as the private or secret key. This means the public key can be freely published. Using this public key, a message can be sent securely to the other party. Only the party holding the secret key can decrypt the message.

Public key algorithms are also used for creating digital signatures on the data. The secret key is used to create the digital signature and the public key used to verify it.

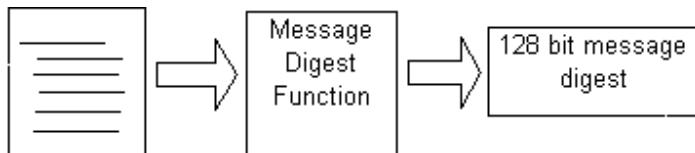
Name	Description
Diffie-Hellman key exchange	This is not encryption system but a method to exchange the key over a public communication
RSA	This is used for encrypting as well as for digital signatures. The key may be any length

### **Hybrid public/private cryptosystems**

Public key cryptography is used to create a random session key. This is then used as a base for the symmetric key algorithm. A session key means it is not reused after the session.

### **Message Digest Functions**

This produces a random pattern of bits. This uses a function to produce a number usually between 128 to 256 bits. Message digest functions have the following properties.



- i) The input affects every bit of the function.
- ii) If any bit of the input is changed, there is a 50% chance the output bit will also change.
- iii) It is almost impossible to produce another file with the same digest value.

Some common message digest functions are given below

Message Digest Function	Description
HMAC	Hashed Message Authentication Code This uses the function together with a secret key to create a secret message authentication code
MD2	Message Digest #2 The security is highest among the series developed by Ronald Rivest but takes the longest to compute. It produces a 128 bit digest
MD4	Message Digest #4 This is a faster alternative to the #2
MD5	Message Digest #5 A more secure version of the #4

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### PGP (Pretty Good Privacy)

This is available for free and used to encrypt and decrypt messages.

Every user has her own pass phrase (maximally 253 characters).

Using MD5 PGP generates from this a 128 Bit key.

This key is used to encrypt the private RSA-key.

VITEC

### 3.5.4 Data mining

Cluster detection

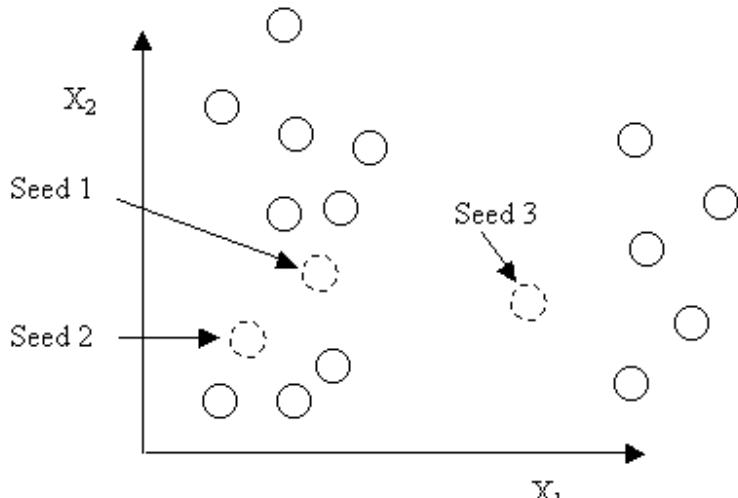
<http://www.vitec.org.vn>

k-means

The algorithm divides the data into a set of k clusters. Each record is mapped to a point.

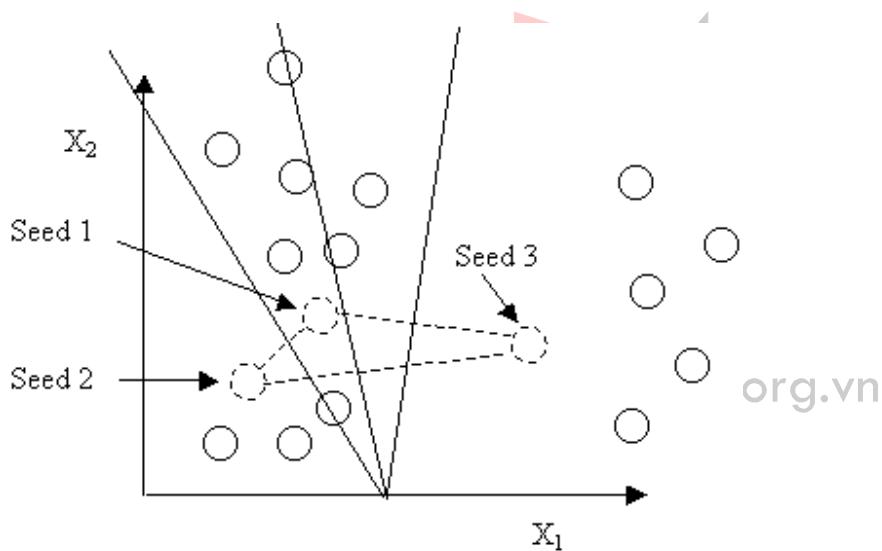
Geometry is used to obtain the mean distance. The number of dimensions is the same as the number of fields in the record. Each data must be normalized and the fields must be converted into numbers.

For each set of clusters, k data points are used as seeds.

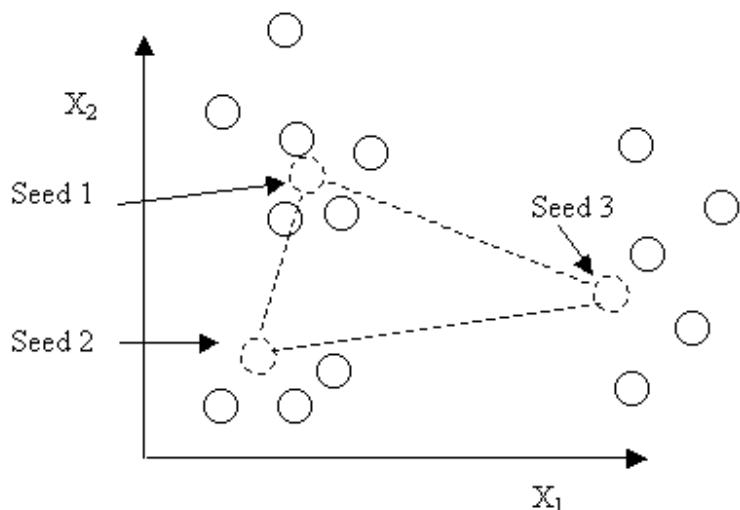


Trung tam sat nacn Cong nghe thong tin va Hỗ trợ đào tạo  
Each record is then assigned to the cluster whose centroid that is nearest. Boundaries

between clusters are drawn by utilizing the mid point between the seed points.



The centroid of each cluster is then calculated by finding the average of all the points in the cluster. Each record is then re-assigned to the clusters created. This continues until the boundary stops changing.



Trung tâmSAT nãcn Công nghệ thông tin và Hỗ trợ đào tạo  
Cluster detection is used when there is a possible set of natural groupings representing features in common. This is used when there are many competing patterns and it is difficult to spot a single pattern. Complexity is reduced by creating clusters and other data mining methods can be applied to the results.

### **Decision trees**

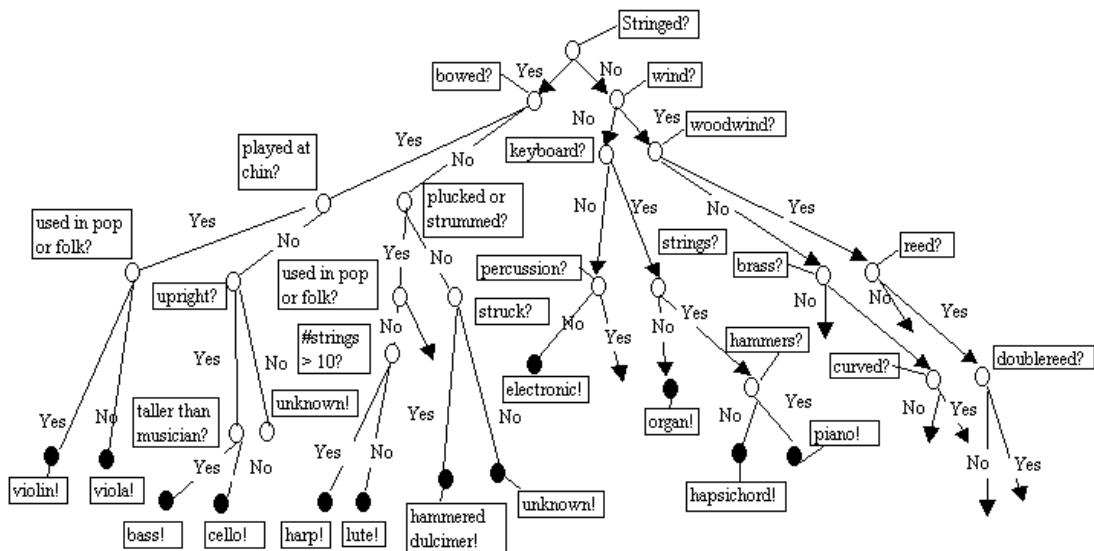
#### **Classification tree**

This means it aids in classifying the record to a given class.

The Class probability is the confidence that a record is in a given class.

#### **Trees grow in many forms**

An example of a binary tree of non uniform depth is shown. Each node has 2 children and the depth from root to leaf is variable. Decision trees don't necessarily have to be binary.



## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### Use of rules

The effectiveness of a decision tree is measured by applying it to a collection of unseen records and determining the percentage classified correctly. Each path through the tree represents a rule. The predictive ability of the tree can be improved by pruning back some of the weaker branches.

At each node, the following is measured

VITEC

- 1) Number of records entering the node
- 2) The way the records are classified if it is a leaf node
- 3) The percentage of records classified correctly at this node

Decision tree building algorithms start by trying to determine the best way to split the data between the different categories. At each level of the tree, the subsets are split further.

### Pruning the tree

This is the process of removing leaves and branches to improve the performance of the decision tree.

### **Bonsai technique**

This applies tests at each node to determine if the further split is useful. It may a simple rule like a minimum number of records at the node to a statistical test to for the significance of the proposed split. The danger with the bonsai technique is they rely only on information in the training set and may lead to the memorization of the training data and overfitting. A rule of thumb is setting a large minimum node size leads to well behaved trees.

Pruning methods allow the tree to grow deep than prune off branches that fail to generalize.

The pruning decision is based on the performance of the tree. The performance is measured using a set of pre-classified data called the test set. The algorithm prunes back to the sub tree that minimizes the error on the test set.



## Exercises for No.1 Chapter 3 (Algorithms)

**Q1** What is the term used to show a algorithm that searches the elements sequentially from the head to the foot of a table?

- a. Linear
- b. Binary
- c. Hash
- d. Heap

**Q2** Values in the array containing n elements, are sequentially compared with data X which is data to be searched. If data X matches a certain value in the table, "exist" is shown. Data X is stored in the position designated as index n+1.

Subscript	1	2	3	...	i	...	n	n+1
Value	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	...	a <sub>i</sub>	...	a <sub>n</sub>	X

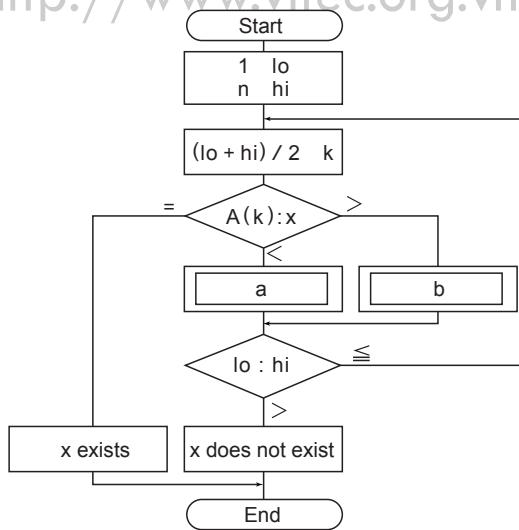
In the linear search algorithm shown below, what condition should be entered in the space  ?

- Step 1 i is set to index i.  
 Step 2 If , the routine jumps to step 5.  
 Step 3 1 is added to index i.  
 Step 4 The routine jumps to step 2.  
 Step 5 If index i is n or less, "exist" is shown.  
 Step 6 End

- a. i ≥ n
- b. i ≠ n
- c. i < n
- d. X = a<sub>i</sub>
- e. X ≠ a<sub>i</sub>

**Q3** There is array A which contains n data item sorted in the ascending order. The following flowchart shows the routine for retrieving data x from array A using the binary search method. Choose a correct combination of operations and enter them in the spaces a and b. Decimal numbers of a value obtained by division should be truncated.

<http://www.vitec.org.vn>



	a	b
a	$k + 1 \rightarrow hi$	$k - 1 \rightarrow lo$
b	$k - 1 \rightarrow hi$	$K + 1 \rightarrow lo$
c	$k + 1 \rightarrow lo$	$k - 1 \rightarrow hi$
d	$k - 1 \rightarrow lo$	$K + 1 \rightarrow hi$

**Q4** There is a table that has 2,000 different elements sorted in the ascending order of the keys. Using a key input from outside, this table is searched using the binary search method, and the elements that match the key are retrieved. What is the maximum number of comparison times needed before all matching elements are found? It is assumed that the table contains the matching key.

- a. 10      b. 11      c. 12      d. 13

**Q5** Which of the following comments made on search methods is wrong?

- a. To use the binary search method, data must be sorted.
- b. To search 100 data using the binary search method, the maximum number of comparison times that is needed to find the target data is 7.
- c. If the linear search method is used, the number of comparison times does not necessarily decrease even if the data is already sorted.
- d. If the number of data is 10 or smaller, the average number of comparison times that the linear search method requires, is smaller than that of the binary search method.
- e. If the number of data is increased from 100 to 1,000, the number of comparison times increases to 10 times as large as when the linear search method is used. Using the binary search method, the number increases twice or less.

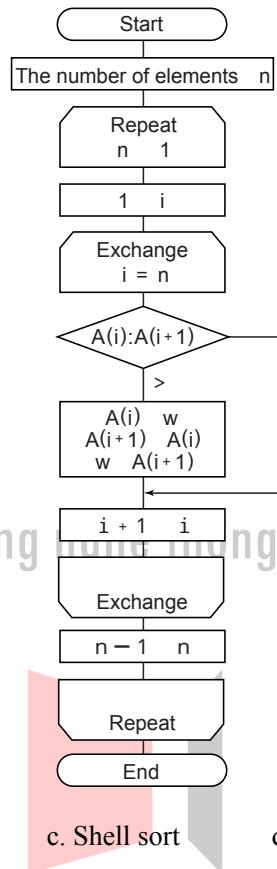
**Q6** Concerning data sort and merge, choose the proper combinations of words and enter them in the space .

Sorting data in the order of small- to large-value queues is referred as  a  b. If a target data sequence is in an auxiliary storage, this operation is called  c.

Integrating two or more files  b in certain order into one file is called  d.

	a	b	c	d
A	descending	sort	external sorting	merge
B	ascending	merge	external merge	sort
C	descending	merge	internal merge	sort
D	ascending	sort	external sorting	merge
E	ascending	merge	internal merge	sort

**Q7** What sort of algorithm does the following flowchart show?



- a. Quick sort      b. Shaker sort      c. Shell sort      d. Insertion sort      e. Bubble sort

**Q8** It took 1.0 seconds for a certain computer to sort 1,000 data using the bubble sort method. How long will it take to sort 100,000 data of the same type? The time that it takes for a computer to bubble sort data is proportional to a square of the number of data that is defined as  $n$ .

- a. 1      b. 10      c. 100      d. 1,000      e. 10,000

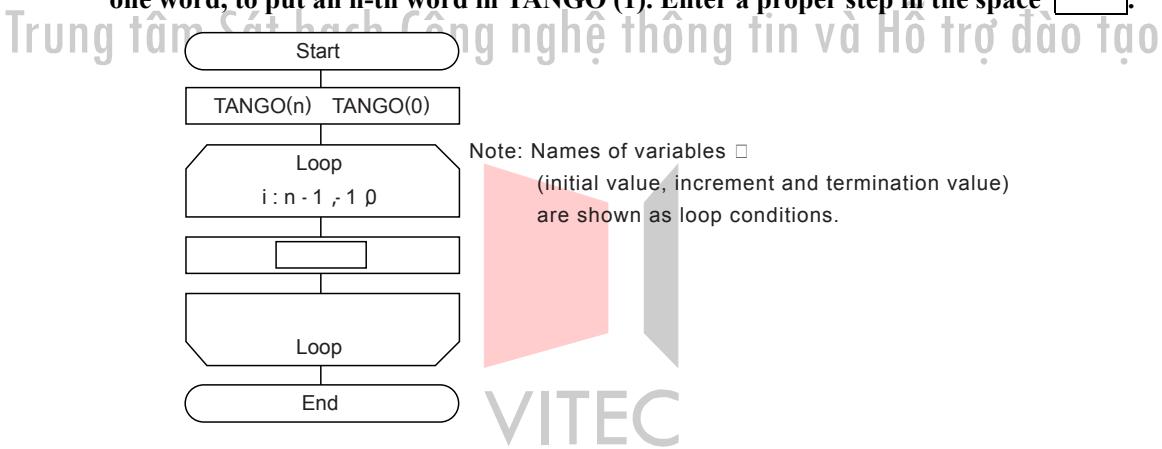
**Q9** Concerning data sorting methods, which of descriptions given below is correct?

- a. Quick sort is a method of sorting data in sub-sequences consisting of data items fetched at an interval and sorting smaller sub-sequences consisting of data items fetched at a smaller interval.
- b. Shell sort is a method of sorting data by comparing a pair of adjacent elements and exchanging them if the second element is larger than the first.
- c. Bubble sort is a method of sorting data by setting a median reference value, allocating elements with values larger than the reference value in one section and those with values smaller than the reference value in the other section and repeating this routine in each individual section.
- d. Heap sort is a method of sorting data by representing an unsorted area as a subtree, fetching the maximum or minimum value from the unsorted area, moving the maximum or minimum value to a sorted area and repeating this routine to narrow down unsorted areas.

**Q10 Which is an appropriate description of quick sort?**

- a. Compare and exchange are executed for two data away from one another at a certain distance. This distance is gradually and continuously narrowed to sort all data.
- b. The first minimum value is found in data. The second minimum value is found in data in which the first minimum value is not included. This routine is repeatedly executed.
- c. Data is divided into one group of data smaller than a reference value and the other group of data larger than a reference value. In each group, a new reference value is then selected and data is likewise divided into two groups based on the reference value. This routine is repeatedly executed.
- d. Adjacent data are repeatedly compared and exchanged to allow smaller data to move to the end of a data array.

**Q11 There is array TANGO whose index number starts with 0. n words are contained in TANGO (1) through TANGO (n). A flowchart below shows the steps for reorganizing the word table by shifting words in TANGO (1) through TANGO (n-1) backward, by one word, to put an n-th word in TANGO (1). Enter a proper step in the space [ ].**



- a. TANGO (i) → TANGO (i+1)
- b. TANGO (i) → TANGO (n-i)
- c. TANGO (i+1) → TANGO (n-i)
- d. TANGO (n-i) → TANGO (i)

**Q12 From the descriptions made on Newton's method, which is known as an algorithm for obtaining an approximate solution of the equation  $f(x) = 0$ , choose the most appropriate one.**

- a. Although a function,  $f(x)$ , cannot be differentiated, an approximate solution can be obtained.
- b. As seen from the geometrical viewpoint, the method is to obtain an approximate solution by using a tangential line of  $y = f(x)$ .
- c. Two different initial values must be provided.
- d. Whatever initial values are provided, an approximate value can always be obtained.

## Hardware

### Chapter Objectives

Among computer related technology, the progress of hardware technology is particularly remarkable.

In this chapter, the learning objective is to understand the mechanism and functions of each of the five main units that are basic to computer hardware. To:

- 1 Understand the roles and functions of the computer's five main units
- 2 Understand basic operations and the registers used to read, decode and execute the instructions and data stored in the main storage unit by the processor (processing unit), which is composed of the control unit and the arithmetic unit,
- 3 Understand the basic approach and configuration circuits of the arithmetic unit that performs arithmetic operations and logical operations
- 4 Understand the mechanism and functions of the main storage unit and the auxiliary storage devices used to store data, and know their types and characteristics,
- 5 Understand the types and mechanisms of the input/output units as well as the input/output control system and the input/output interface,
- 6 Understand computer types and characteristics

## **Introduction**

The function of the hardware composing a computer can be divided broadly into the following five categories:

- Input
- Storage
- Operation
- Control
- Output

The following are the units that implement the above mentioned functions:

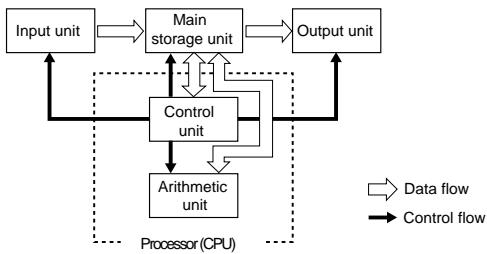
- Input unit: This unit inputs the data and programs for computer processing. It is equivalent to the human eyes and ears.
- Storage unit: This unit stores the input data and programs. It is equivalent to the memory section of the human brain.
- Arithmetic unit: This unit conducts calculation and decision on the stored data according to the instructions of the program. It is equivalent to the thinking section of the human brain.
- Control unit: This unit controls the input unit, storage unit, arithmetic unit and the output unit. It is equivalent to the human central nervous system.
- Output unit: This unit outputs the results of computer processing in a format that can be understood by humans. It is equivalent to the human hands and mouth.

These five units are called the "computer five main units" (Figure 4-1-1)

Since the control unit and the arithmetic unit are handled as one unit, they are called the processor (processing unit) or central processing unit (CPU). The general term "peripheral devices" is used to refer to the input unit, the output unit and the auxiliary storage devices that are outside the processor and exchange data with the main storage unit. Likewise, the storage units are divided into the main storage and auxiliary storage device, depending on their functions.

**Figure 4-1-1**

Computer five main units



## 4.1 Information element

### 4.1.1 Integrated circuit

In today's computers, an integrated circuit (IC) that integrates semi-conductors to a high level is used. According to the integration level, ICs are classified as in Figure 4-1-2.

**Figure 4-1-2**

IC classification according to their integration level

IC	Integration level
SSI ( Small Scale Integration )	$10^1\text{-}10^2$
MSI ( Medium Scale Integration )	$10^2\text{-}10^3$
LSI ( Large Scale Integration )	$10^3\text{-}10^4$
VLSI ( Very Large Scale Integration )	$10^5\text{-}$

Note: The integration level indicates the range of the number of gates (number of transistors) contained in 1 IC.

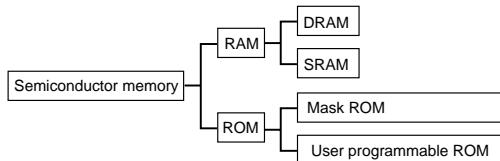
Likewise, according to their structure, ICs can be classified as follows:

### 4.1.2 Semiconductor memory

Logic elements are used in logical operations while storage elements are used in data and instruction storage. Here the storage elements will be explained. (The logical circuit will be explained in detail in the next section, 4.2.) The storage element is called semiconductor memory (or IC memory) and is broadly divided into RAM and ROM.

**Figure 4-1-3**

RAM and ROM



#### (1) RAM (Random Access Memory)

A RAM is semiconductor memory in which data writing and reading is possible. When the computer is turned off, the stored data is lost. This property is called volatility. Since most main storage units are composed of RAMs, the processor can be made to read and write information

from the main storage unit at random by specifying the address.

RAMs are classified into DRAMs and SRAMs.

### 1) DRAM (Dynamic RAM)

A DRAM represents bits, and stores information depending on whether the part called capacitor is being charged (status "1") or is not being charged (status "0"). Since the circuits are simple and small, RAMs of large capacity can be created at low cost. However, since the charge stored in the capacitor is lost after a lapse of time, the memory needs to be rewritten (recharged) at regular intervals. This operation is called refreshing. Once, DRAMs were used in the main storage unit, but currently they are also used in storage units, etc, contained in the input/output units of printers and other devices.

### 2) SDRAM (Synchronous DRAM)

Due to the progress of IC technology, and the consequential substantial improvement of the performance of processors, the operating speed of the DRAMs that composed the storage unit could not keep up with the operating speed of the processors. For that reason, an external clock signal that indicates the processor operation timing is now set in the DRAM and through synchronization with this signal, complicated address specifications are reduced and simplified, enabling the development of DRAMs that operate at high speeds. These types of DRAMs are called synchronous DRAMs (SDRAM).

### 3) SRAM (Static RAM)

SRAMs are created with a circuit called the flip-flop. The flip-flop settles the output according to the previous input and the current input, and can preserve the status "1" and "0" inside the circuit. Since data is not lost unless the computer is turned off, memory refreshing is not necessary. However, since SRAM circuits are complicated, the memory capacity is smaller than that of DRAMs and the cost is higher. However, since its processing speed is high, it is used in devices such as the registers contained in main storage units and processors.

## (2) ROM (Read Only Memory)

The ROM is semiconductor memory for read use only. Since programs and data are stored in the ROM from the beginning, the stored information is not lost even if the computer is turned off. This property is called non volatility.

ROMs are classified into mask ROMs and user programmable ROMs.

### 1) Mask ROM

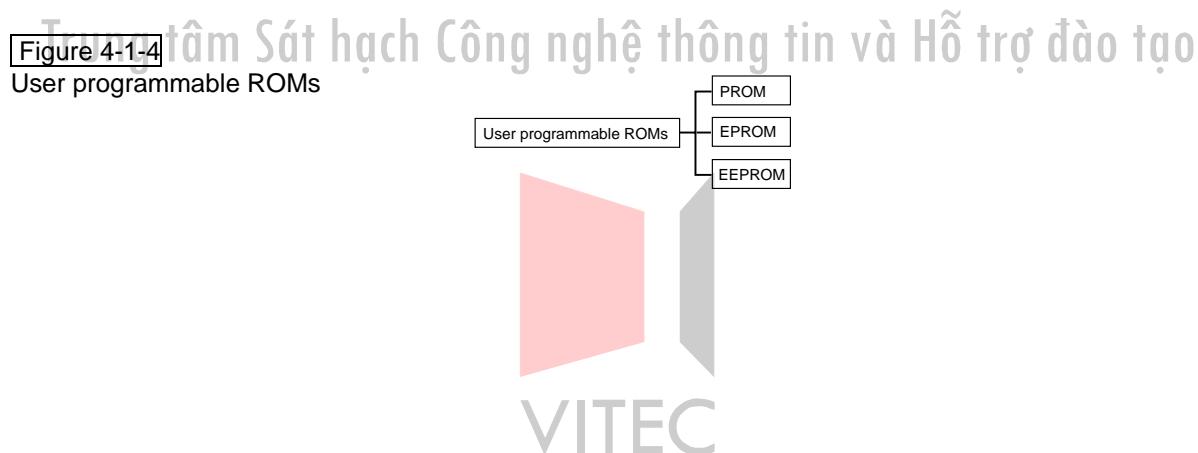
Since program and data are already written in the Mask ROM before it is shipped by the manufacturer, the user cannot add any programs or data. Mask ROMs are used in the memories of game cassettes and IPL (Initial Program Loader) a program used to start the computer, etc.

## 2) User programmable ROM

The user programmable ROM is a type of ROM, but since at the time it is shipped by the manufacturer it has nothing stored in it, the user can write data into it once. The following types of programmable ROM exist (Figure 4-1-4)

- PROM (Programmable ROM): Once data has been written, it cannot be erased.
- EPROM (Erasable PROM): It can be erased with ultraviolet light and rewritten.
- EEPROM (Electricity Erasable PROM): It can be erased through application of electrical voltage and rewritten.

EEPROM is used in a storage medium called flash memory, which is used in the registration of image data of digital cameras, etc. Likewise, it is also used in the storage section of IC cards, etc.



<http://www.vitec.org.vn>

## 4.2 Processor Architecture

---

### 4.2.1 Processor structure and operation principles

#### (1) Processor structure

Among the five main units that composed a computer, the control unit and the arithmetic unit are handled as one unit and are called processor (processing unit). The processor is also called central processing unit (CPU) and as the backbone of the computer, it plays the important roles indicated below.

##### 1) Control unit

The control unit is the unit that controls all the operations of the computer. It retrieves, decodes and executes, one by one, in order, the instructions stored in the main storage unit.

The following are the main functions of the control unit.

- Retrieval of the instructions stored in the main storage unit
- Decoding of the retrieval instructions using the instruction decoder
- According to the decoding, transmission of the specifications required for the execution of the instructions to each unit

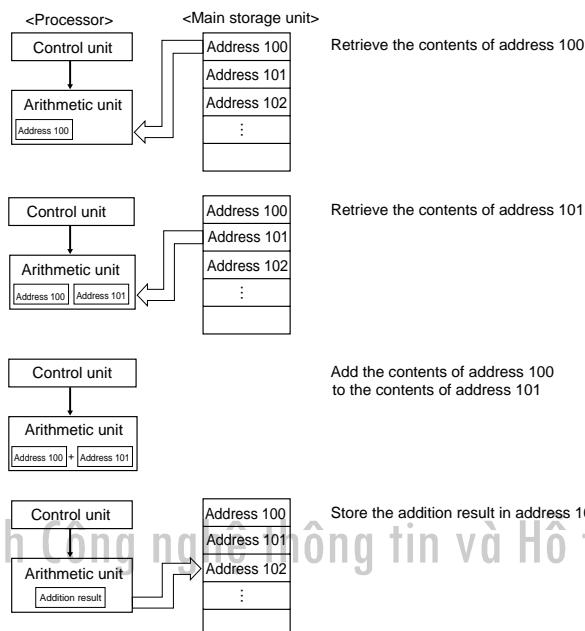
Through the repetition of these operations, the control unit controls each unit, and implements the functions of each of the units as a computer system. The system by which instructions are executed in this way, one by one, is called the sequential control system. The design of this sequential control system and the stored program system (Please refer to main storage unit in 4.3.1) was based on the concepts of John von Neumann, Computers of this kind are called Neumann computers.

<http://www.vitec.org.vn>

In Figure 4-2-1, as an example to explain the mechanism of the sequential control system, the process of the "addition of the contents of address 100 and the contents of address 101 and the storage of the result in address 102" is represented.

**Figure 4-2-1**

Mechanism of the sequential control system



## 2) Arithmetic unit

The official name of the arithmetic unit is arithmetic and logic unit (ALU). This unit performs arithmetic operations, logical operations, comparison, branch, and other processes on the data assigned to be subject to processing. The main instructions are shown in Figure 4-2-2.

**Figure 4-2-2**

Functions of the arithmetic unit

Basic operations	Basic instructions
Arithmetic operations	Addition, subtraction, multiplication and division
Logical operations	Logical sum (OR), logical product (AND), negation (NOT)
Comparison	Comparison instruction (size comparison)
Branch	Branch instruction (change of the sequence of instruction execution according to the conditions)

Depending on the representation method of data assigned to be subject to operations, arithmetic and logic unit has functions performing fixed point operations, floating point operation and decimal operation.

## (2) Processor operation principles

### 1) Instruction readout and decoding

The data and programs retrieved from the main storage unit are transferred to the processor through the data bus. Then the data subject to processing is temporarily stored in the "general-purpose register," while a part of the program indicating the procedure of the process is transferred to the "instruction register."

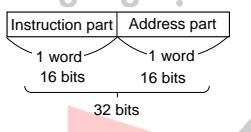
## a. Instruction and instruction format

### Instruction

The program is a set of instructions that indicates "do..." to the computer. Since inside the computer, data is represented in the binary system, instructions are also represented in the binary system. The instructions represented in the binary system are called machine language instructions. Regardless of the program language in which a program is written, at the end it is converted into language that can be understood by the computer, machine language, in order to be decoded and executed. Machine language instructions and instruction formats differ depending on the computer, but, in general terms, they are composed of the following parts.

- **Instruction part:** Indicates instructions and operations
- **Address part:** Specifies the address and register of the main storage unit subject to processing

Figure 4-2-3  
Example of the instruction structure



### Instruction format

In practice, instruction formats differ depending on the computer. There are instructions that have several address parts and according to the structure of the address part, there are four address formats, from zero-address format to the three address format.

#### Zero-address format

The zero-address format performs operations using a dedicated register called a stack pointer. Currently, zero-address format computers are not used. The stack pointer is the register that stores the address to be returned to (return address) after execution completion.

Figure 4-2-4 Zero-address format

Address part

#### Single-address format

The single-address format performs operations between the contents of the main storage unit specified in the address and the accumulator data (Figure 4-2-5). The accumulator stores operation values and operation results. There are cases where general-purpose registers are also used as accumulators.

Figure 4-2-5 Single-address format

Instruction part | Address part

## Two-address format

The two-address format specifies two addresses and uses the address data specified on the main storage unit.

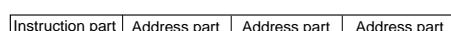
Figure 4-2-6 Two-address format



## Three-address format

The three-address format specifies two addresses to be used for the operation and the address where the operation result is to be stored

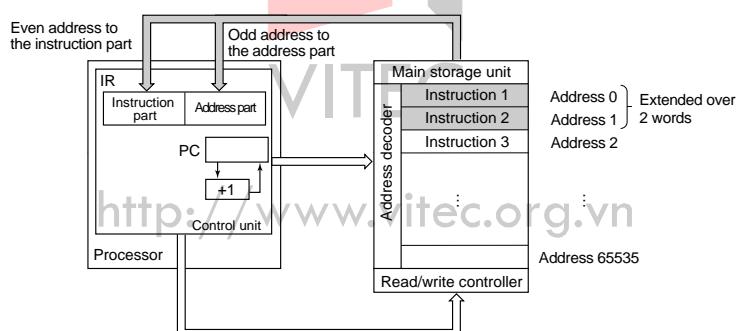
Figure 4-2-7  
Three-address format



### b. Instruction readout

The instruction received from the main storage unit is stored in the instruction register (IR). The length of one word of the main storage unit is 16 bits. It is supposed that one instruction has 32s in the computers used. One instruction is stored in two words. Therefore, the content of the address of the main storage unit accessed is sent to the processor twice. In practice, it is determined beforehand that in one instruction the instruction part is stored in an even address while the address part is stored in an odd address.

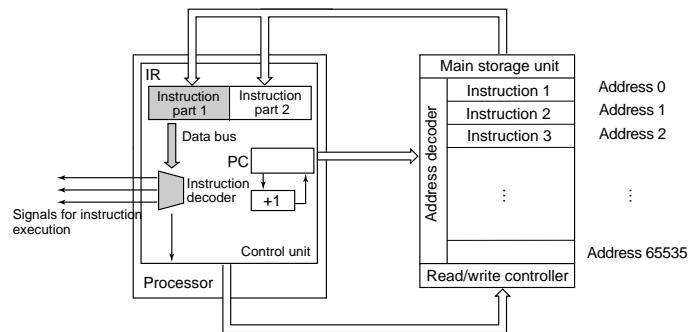
Figure 4-2-8 Instruction loading



### c Instruction decoding

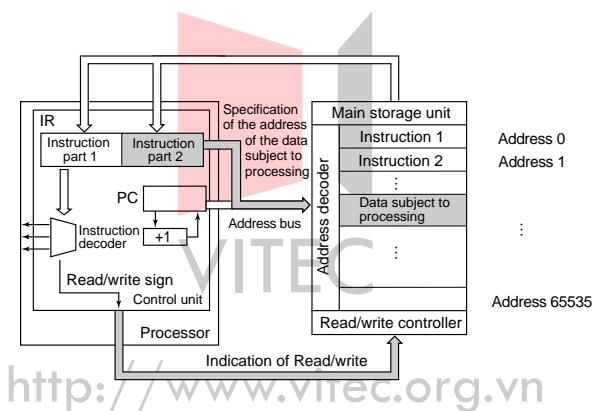
The content of the instruction part of the instruction register is transferred to a device called the decoder. The decoder decodes the type of work indicated by the instruction and sends the signals for the execution of the operation to each unit.

**Figure 4-2-9**  
Instruction decoder



On the other hand, the content of the address part of the instruction register is transferred to the address bus. The content of the address of the main storage unit specified by the address bus corresponds to the data subject to processing. According to the instruction of the instruction part, a read/write signal is sent from the control unit to the data, subject to processing on the main storage unit.

**Figure 4-2-10**  
Address part and  
data subject to  
processing



## 2) Instruction execution

Once the instruction content and the address of the data subject to processing are obtained, the instruction is executed. Using the example of assembler language in which there is a one to one correspondence with the machine language, the instruction execution control and each type of register will be explained below.

### a. Storing retrieved data

If, as a result of decoding the instruction part and the address part using the instruction decoder, the instruction is found to say "Retrieve and transfer to the processor the contents of address 100 of the main storage unit," a place to store the retrieved contents will be needed. Therefore, a general purpose is set in the arithmetic unit of the processor in order to store the retrieved data.

In this example, it is assumed that there are five registers, and, for convenience, the numbers 0 to 4 will be assigned to them. Then, using the initial s of each of the general-purpose registers, they will be represented as GR0, GR1, GR2, GR3 and GR4.

**Figure 4-2-11**  
General-purpose register

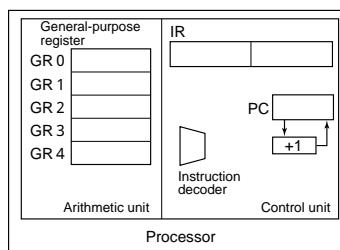
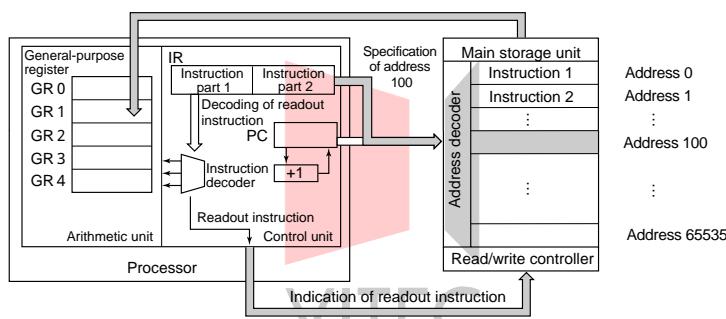


Figure 4-2-12 shows the mechanism by which the contents of address 100 of the main storage unit passes through the data bus to be stored in general-purpose register GR1.

**Figure 4-2-12** Storage in the general-purpose registers



### b. Instruction execution

If, as a result of decoding the instruction part and the address part of the instruction register, the instruction is found to say "Add the contents of address 100 of the main storage unit to GR1 contents and store them in GR1," the retrieved contents of address 100 have to be added to GR1 contents. The unit that performs this kind of addition and subtraction of numeric values is the ALU (Arithmetic Logic Unit)

The ALU has the following arithmetic mechanism.

- |  |   |
|--|---|
| Fixed point operation mechanism to perform operations<br>of integer data<br><br>Floating point operation mechanism to perform operations<br>of floating point data<br><br>Decimal operation mechanism to perform operations<br>of binary coded decimals in packet format | } for scientific and engineering calculations<br><br>} for commercial data processing |
|--|---|

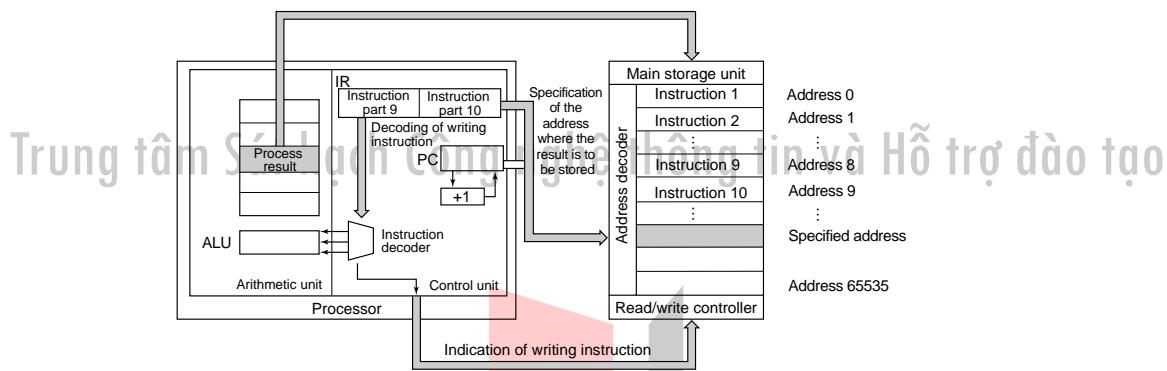
It should be noted that besides arithmetic operations such as addition and subtraction, the operation mechanism of the ALU performs logical operations such as logical products, logical

sums and shifts.

### c Processing subsequent to the instruction execution

Based on the instructions and data retrieved from the main storage unit, the result of the process performed using the operation mechanism and the register contained in the processor is transferred to the main storage unit through the data bus. Then the address where the result is to be stored is specified by the program instruction.

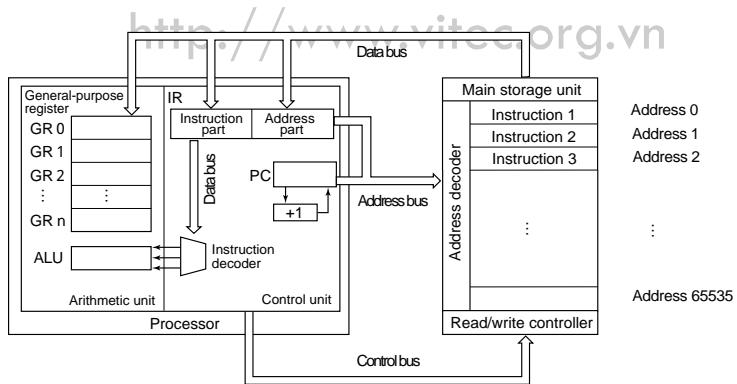
Figure 4-2-13 Storage of the process result



### d. Flow of the instruction from its decoding to its execution and hardware structure

Figure 4-2-14 shows the hardware structure from the instruction readout to its decoding and execution.

Figure 4-2-14 Hardware structure



### e. Various registers

Up to this point, the roles played by the instruction register, the general purpose register, etc, have been explained, but, besides the registers, the following registers exist:

- . Program counter
- . Accumulator
- . Index register
- . Base address register
- . Program Status Word (PSW)
- . Flag register
- . Complement register

### • Program Counter (PC)

In Figure 4-2-15, considering the procedure in which instruction "A" stored in address 101 of the main storage unit is loaded to the processor, the following can be observed:

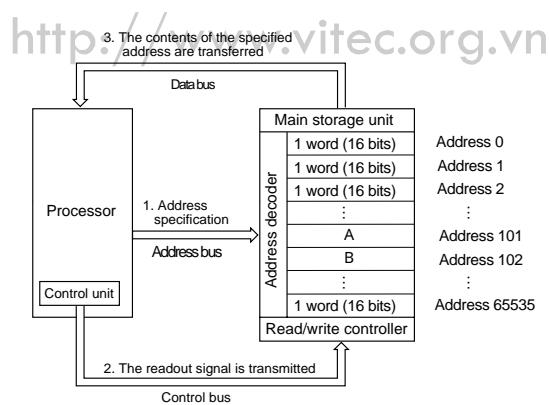
1. The processor specifies address 101 with the address bus
2. The control unit send the readout signal to the main storage unit
3. The main storage unit transfers the contents of address 101 to the processor using the data bus.

## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

At the beginning, the processor specified address 101, but under whose instructions? There is actually a storage unit that exclusively performs this kind of instruction inside the processor. It is called the program counter and is composed of 16 bits. The program counter is also called the instruction address register, instruction counter or sequential control counter.

The devices, as in the program counter, are set inside the processor and temporarily store data are known generally as registers. Among the registers, there are specialized registers whose application is set in advance, as in the program counter's, and general purpose registers whose application is freely decided by the program.

**Figure 4-2-15**  
Address reading



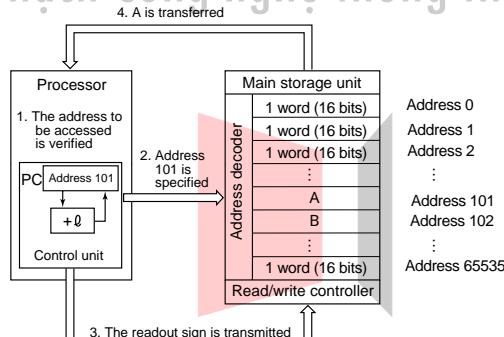
Computer hardware is set in such a way that when the computer is turned on, the content of the program counter is immediately read and the address of the main storage unit to be accessed is

verified. Likewise, every time this program counter is referred to, the content stored is automatically "+l (instruction word length)"

Taking into account the role of the program counter, and if we consider the procedure in which the processor loads instruction "A" stored in address 101 of the main storage unit, the following can be observed:

1. The content stored in the program counter is referred to and the address of the main storage unit to be accessed is verified. After it is referred to, "l" is automatically added to the content of the program counter.
2. The processor specifies address 101 with the address bus.
3. The control unit sends the readout signal to the main storage unit.
4. The main storage unit transfers the contents of address 101 A to the processor using the data bus.

**Figure 4-2-16**  
Role of the  
program counter

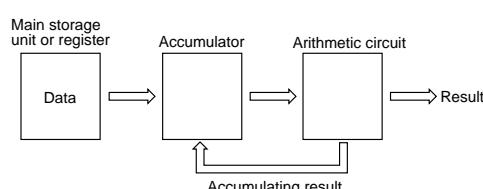


### ● Accumulator

The accumulator is the register used exclusively to store operation results and operation values. Since it stores accumulating results, it is also called the accumulating device. There are cases when the general purpose register is used as a substitute for the accumulator.

When the accumulator is used, it is called accumulator mode, when the general purpose register is used, it is called general purpose register mode.

**Figure 4-2-17**  
Accumulator

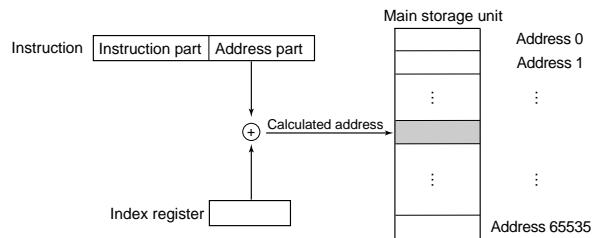


### ● Index register

When an address in the main storage unit is specified, the act of changing the address of the address part of the instruction is called address modification. The register used to perform

this change is the index register.

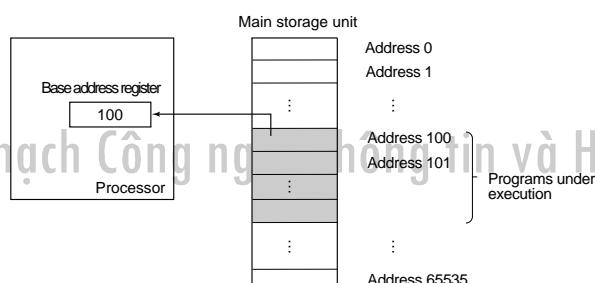
**Figure 4-2-18**  
Index register



### ● Base register

The base register is the register that stores the program top address.

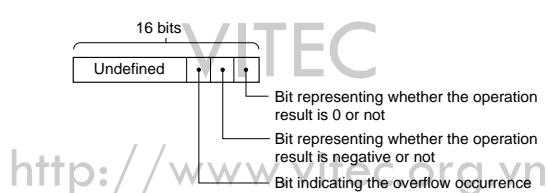
**Figure 4-2-19**  
Base address register



### ● Flag register

The flag register stores information related to the operation result (if it is positive or negative or 0), to the existence of carry, overflow, etc.

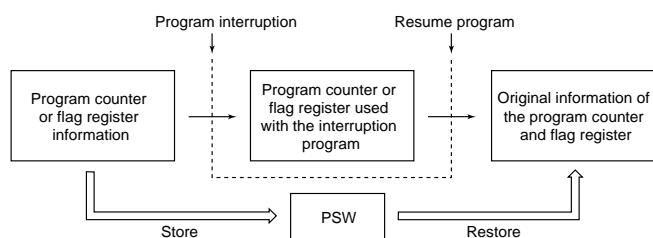
**Figure 4-2-20**  
Example of a flag register



### ● PSW (Program Status Word)

The program counter, flag register and other information are registered in the PSW. In cases where an event that interrupts the program (interruption) in the processor occurs, the program execution can be resumed using the PSW information.

**Figure 4-2-21**  
PSW



### ● Complement register

The complement register generates integer complements in order to perform operations in the addition circuit.

**Figure 4-2-22**  
Complement register

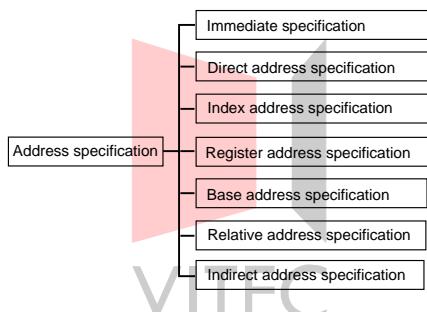


### (3) Address specification mode

The address part of the instruction specifies the main storage unit address and the register subject to processing. This specification method is called the address specification method.

At the instruction execution, the value of the address part of the instruction is not used as the address subject to processing as it appears; the actual address is specified after performing calculations between the specified register and addresses. The act of obtaining the address through calculation is called "address modification" and the actual address obtained is called the "effective address". The type of address specification modes are listed in Figure 4-2-23.

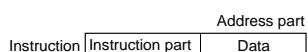
**Figure 4-2-23**  
Types of address specification modes



#### ① Immediate specification

In the immediate specification, the data itself is contained in the address part. Since it is not necessary to access the main storage unit address, it can be immediately executed.

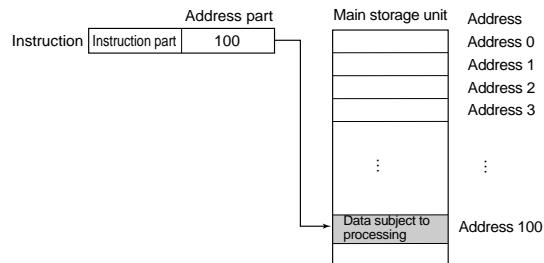
**Figure 4-2-24**  
Immediate specification



#### ② Direct address specification

In the direct address specification, the address of the data subject to processing is contained in the address part.

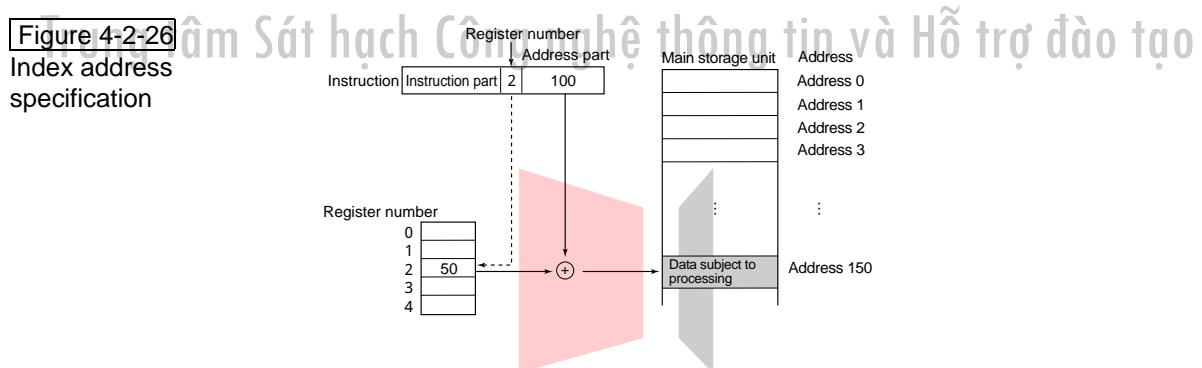
**Figure 4-2-25**  
Direct address specification



### ③ Index address specification

In the index address specification, the address part is divided into the section that specifies the number of the index register and the constant section, and the effective address is the result of the following addition:

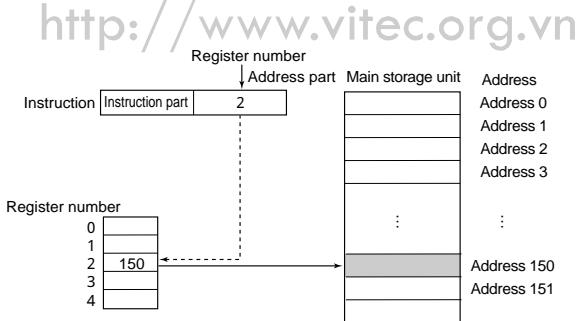
$$(\text{Content of the register content specified with the register number}) + (\text{Address constant})$$



### ④ Register address specification

In the register address specification, the register number is stored in the address part and the address is stored in the register of that number.

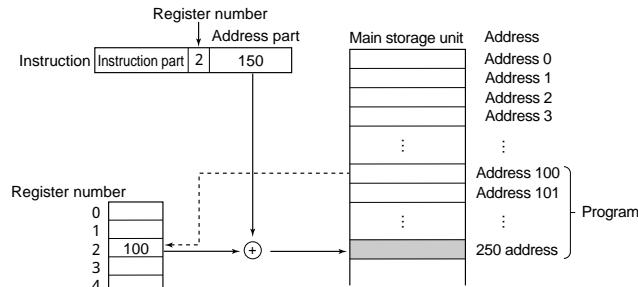
**Figure 4-2-27**  
Register address specification



## ⑤Base address specification

In the base address specification, the program starting address is stored in the base register. the result of the addition of the address contained in this base register and the address constant becomes the effective address.

**Figure 4-2-28** Base address specification

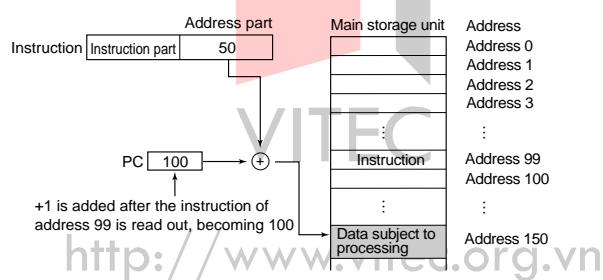


In this example, the program starting address is stored in register number 2, which is used as a base register.

## ⑥Relative address specification

In the relative address specification, the result of the addition of the address of the instruction being executed at the present time (value of the program counter) and the address of the address part become the effective address.

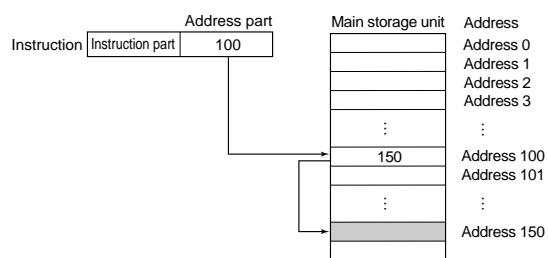
**Figure 4-2-29**  
Relative address  
specification



## ⑦Indirect address specification

In the indirect address specification, the address of the data subject to processing is contained in the address specified in the address part (Figure 4-2-30). There are cases where indirect address specification is performed on two or three levels.

**Figure 4-2-30**  
Indirect address  
specification



#### **(4) Instruction set**

When the computer executes a job requested by the user, the hardware performs the process using several instructions needed from a group of instructions built into that computer. The set of instructions is defined as the interface of the software and the hardware of a specific computer, and depending on the computer, the types and number of instructions differ. This group of instructions (the total of the instructions) is called the instruction set. As a result, computer software packages with identical instruction sets are basically compatible.

#### **(5) Execution control of the instruction**

The execution of a program consists of the repetition of the readout of the instruction from the main storage unit, and the decoding and execution of that instruction by the control unit. If we arrange the operations performed by the processor during program execution, we can divide them as follows:

- Instruction readout
- Instruction execution

##### **① Instruction cycle and execution cycle**

The series of operation consisting of the readout and the instruction stored in the main storage unit, its storage in the instruction register contained in the processor, and the instruction decoding using in instruction decoder, is called the instruction cycle. Likewise, the series of operations consisting of the reading out of the data subject to processing from the main storage unit, and writing and executing the instruction, is called the execution cycle.

##### **a. Instruction cycle**

The instruction cycle consists of the following two consecutive operations:

- According to the value of the program counter, the instruction to be executed is read out from the address of the main storage unit, where it was stored, to be stored in the instruction register.
- The instruction part of the instruction register is decoded using the instruction decoder, and the address part of the data subject to processing is calculated based on the address part of the instruction register.

The instruction cycle is also called 1 cycle, and it is also called F (fetch) cycle.

##### **b. Execution cycle**

The execution cycle consists of the following two consecutive operations:

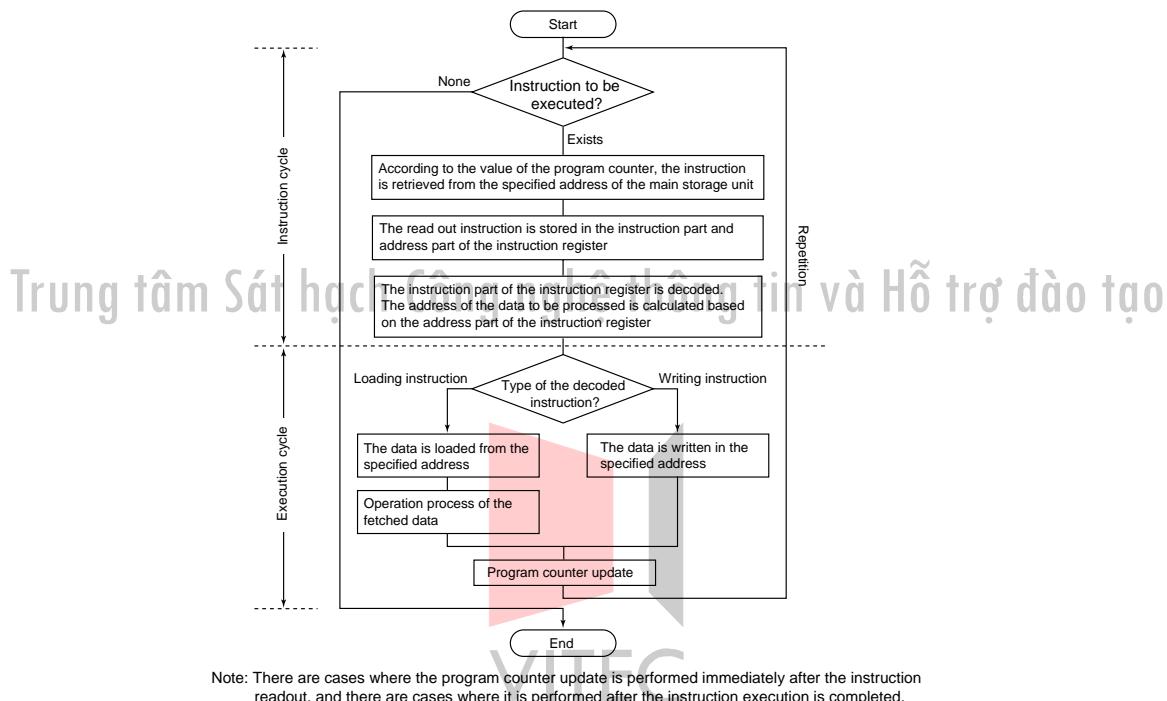
- If the decoded instruction is readout, the data subject to processing is read out from the specified address of the main storage unit; if it is a writing instruction, data is written in the specified address of the main storage unit.
- In the case of readout instructions, the read out data is used to perform the operation process.

Taking the initial of execution, the execution cycle is also called E cycle.

### c. Instruction cycle and execution cycle operations

Figure 4-2-31 shows the flow figure of the operations of the instruction cycle and execution cycle.

**Figure 4-2-31** Flow figure of the operations of the instruction cycle and execution cycle



The processor reads out, in order, the program instructions stored in the main storage unit and executes the program by repeating the instruction cycle and execution cycle.

## (6) Hardwired-logic control and microprogramming control

The instructions set in the computer are executed using logical circuits composed of several logic elements. In other words, logical circuits are the result of hardwiring among the logic elements; for that reason they are called hardwired logic control system or writing logic control system. Since the different instructions are implemented by the hardware, they have the advantage that the operation speed is fast. Opposite to this system, there is one that executes the instructions with the firmware.

As computer performance improves, instructions with more complicated functions become feasible. The higher the function level of the instructions, the more complicated the control procedure becomes. Therefore, instead of executing the instructions with hardware of complicated wiring, an execution method using easily modifiable firmware was designed. This method is called microprogramming control system. Compared to the hardwired logic control

system, the operation speed of the microprogramming control system is slow, but the hardware is simpler and corrections (debugging) are easily performed. The microprogram, which is a string of patterns that determine whether the logic gate is to be on or off, is stored in a special memory called control storage inside the control unit. Instructions are followed by reading out the microprogram sequentially.

#### 4.2.2 Speed performance enhancement in Processor

There are two types of architectures regarding instruction execution. There are

- sequential architecture
- speculative architecture

Another name for pre-fetched control is pipeline. RISC is an example of a processor that uses pipeline processing.

##### (1) Pipeline processing

Sequential architecture is the process of reading, decoding and execution of the instruction till it is finished before another instruction is fetched.

Program execution steps are

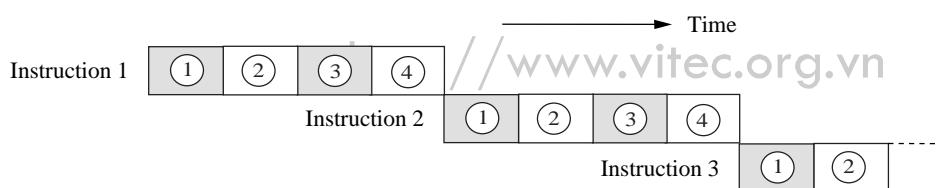
Instruction cycle ①: The instruction is read from the main storage unit

Instruction cycle ②: The instruction is decoded and the address calculated

Execution cycle ③: The reading and writing of the main storage unit is done

Execution cycle ④: The actions specified by the instruction is executed

Figure 4-2-32 Serial control processing



Instruction cycle ① and execution cycle ③ accesses the main storage unit. Instruction cycle ② and execution ④ is executed within the processor.

The processor is idle when the instruction cycle ① and execution cycle ③ are executing. Conversely, when instruction cycle ② and execution ④ are executing, no access to the main storage unit is done.

The next instruction is read in and queued during this idle time. This increases processing efficiency and this method is known as speculative control method. Pipeline processing utilizes this speculative control method and the next set of instructions are read in and processed in

queued to be processing.

Figure 4-2-33 Pipeline processing

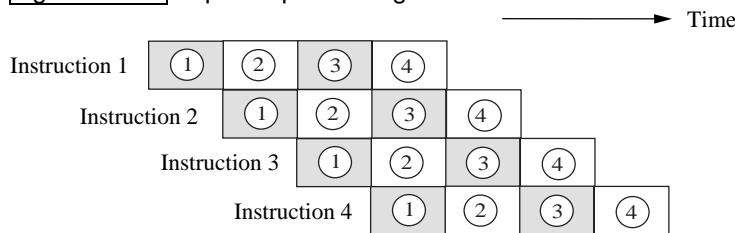


Figure 4-2-33 shows the execution of instructions 1 to 4.

1. Instruction one's instruction cycle ① is executed
2. When instruction one's instruction cycle ② is executing, instruction two's instruction cycle ① is executed
3. When instruction one's execution cycle ③ is executing, instruction two's instruction cycle ② is executed and instruction three's instruction cycle ① is executed
4. When instruction one's execution cycle ④ is executing, instruction two's execution cycle ③ is executed and instruction three's instruction cycle ② is executed and instruction four's instruction cycle ① is executed

In the pipeline processing architecture, when one instruction is being processed, the next instruction is simultaneously read and processing is continued.

This leads to an increased processing speed.

However, the size of the instruction and the execution time should be fixed as the cycles are concurrently executing. The following conditions will lead to degradation in the execution efficiency.

- Fork instructions are issued from within executing instruction. The order of execution of the instructions is changed.
- Program is interrupted and a different program is executed. This phenomenon is known as switching over.

## (2) RISC and CISC

The length of one instruction and the execution time for each instruction should be fixed in pipeline processing. A computer designed with a set of simple instructions with each instruction having a fixed length and execution time is known as the RISC (Reduced Instruction Set Computer) computer.

The opposite of a RISC machine is a CISC (Complex Instruction Set Computer) computer. The computer is designed with sets of complex instructions

**Figure 4-2-34** Characteristics of RISC and CISC

RISC	CISC
<ul style="list-style-type: none"> <li>• Make up of few simple instructions</li> <li>• Instructions are executed by the hardware</li> <li>• The size of the instruction and the execution time of each instruction is about the same</li> </ul>	<ul style="list-style-type: none"> <li>• Complex, high level type instructions</li> <li>• Instructions are executed by the micro-program</li> <li>• There is variation in the instruction size and length of execution</li> </ul>

RISC is not optimized or the number of instruction steps is many, Efficient programs are dependent on the optimization functions found in the compiler.

Recently, many workstations are created with the RISC architecture

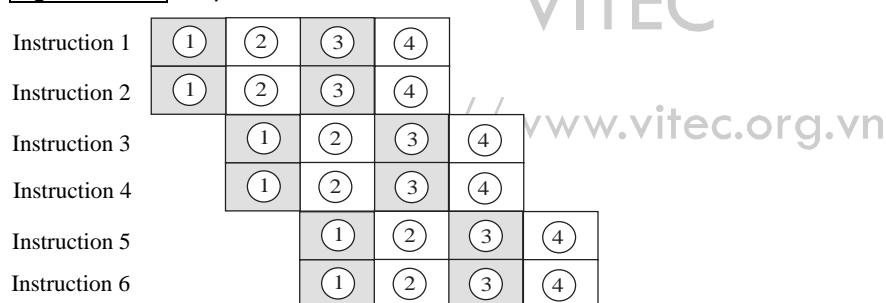
### (3) Parallel method

#### ① Super scalar architecture

Under the best conditions, the normal pipeline architecture is limited to one instruction per cycle. The super scalar architecture allows for multiple instructions to be executed in 1 cycle. Multiple parallel processors in the CPU allow the execution of these multiple instructions in 1 cycle.

Instructions that can be executed in parallel have to be separated from the general instructions. The architecture is known as super scalar if on average, if multiple instructions are running in 1 cycle.

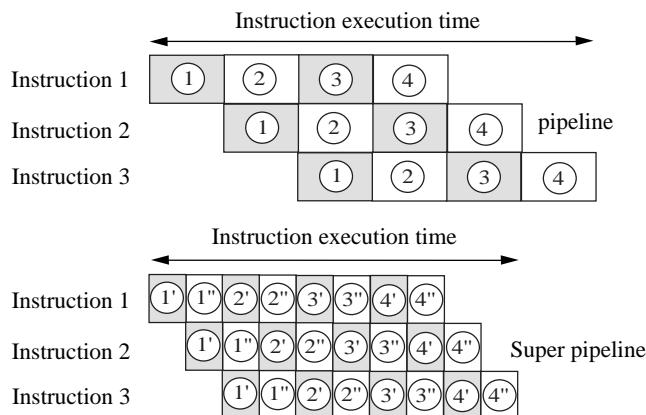
**Figure 4-2-35** Super scalar architecture



#### ② Super pipeline architecture

A super pipeline results when more stages are added to the normal pipeline where each stage does less work and leads to a higher speed.

**Figure 4-2-36** Super pipeline architecture



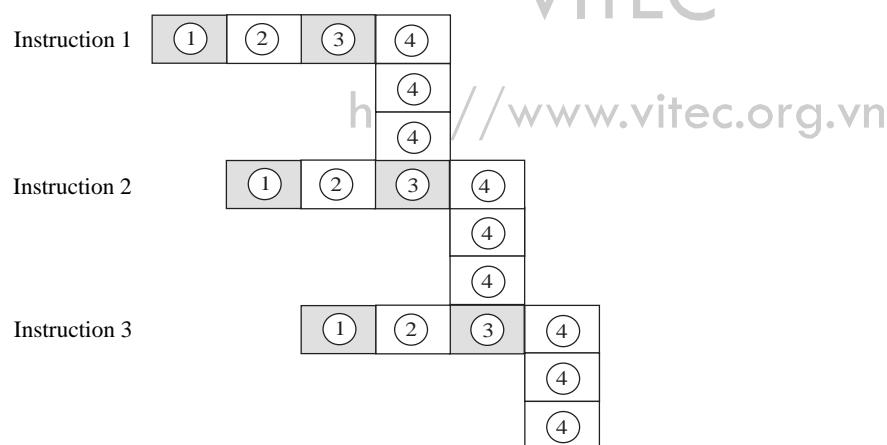
In this architecture, the next instruction starts to be executed before the end of the current instruction. The processing volume is higher than the normal pipeline which means the instructions have a high dependency.

The compiler's sequencing of instructions and the design of a easily processed instruction set is important to ensure a smooth flow of the pipeline

### ③VLIW (Very Long Instruction Word) method

The action of coding multiple tasks in one instruction instead of having short instruction cycles in pipeline processing is known as the VLIW method. The length of one instruction becomes long due to the multiple tasks that are defined within it.

**Figure 4-2-37** VLIW method



### 4.2.3 Operation mechanism

The procedure of the execution of logic circuits studied in Chapter 1 inside the computer will be explained using real logical circuits. In the arithmetic unit, there are numeric operation circuits that handle numeric values and logical circuits that perform logical operations. The following three operations are basic in logical operations:

- Logical product operation (AND operation)
- Logical sum operation (OR operation)
- Negation operation (NOT operation)

Through the combination of circuits that perform these three operations, a wide range of logical circuits implemented

Figure 4-2-38  
Logical operations

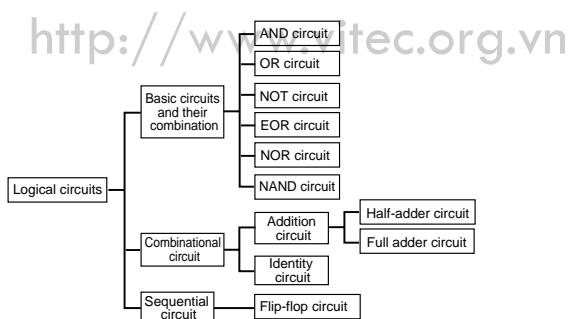
Logical operations	Operation symbols
Logical product (AND)	or $\wedge$
Logical sum (OR)	or $\vee$
Negation (NOT)	or $\neg$
Exclusive logical sum (EOR)	— or $\oplus$
Negative logical sum (NOR)	
Negative logical product (NAND)	

Likewise, according to the combination of logical circuits, circuits can be classified as follows:

- Combinational circuits: Addition circuits, identity circuits, etc that establish the output according to current input.
- Sequential circuits: Flip-flop circuits, etc, that establish the output according to current and past inputs.

The organization of these logical circuits is shown in Figure 4-2-39.

Figure 4-2-39  
Logical circuits



#### (1) Base logical circuits

The three operations "AND", "OR" and "NOT" are base logical operations. The circuits that perform these three operations, AND circuits, OR circuits and NOT circuits will be explained below.

## ①AND circuit

AND circuits are the circuits that perform AND operations (logical product operations), as the one shown in Figure 4-2-40. In these circuits, if both A and B input values are not "1," "1" is not input. The table shown in Figure 4-2-40 is called the "truth table," and in this case, for the input "1" (true) or "0" (false) of A and B, the operation results "1" or "0" are indicated. Likewise, Venn diagram, which clearly represents the operation result, is also displayed.

Besides "A AND B," "AND" operations are represented as " $A \cdot B$ " or as " $A \wedge B$ "

**Figure 4-2-40**  
Truth table and  
Venn diagram of  
AND operations

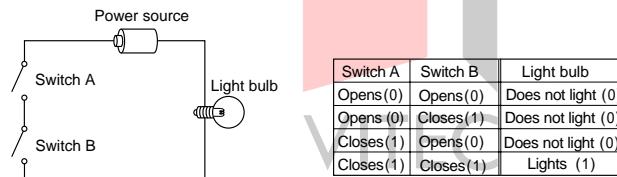
A B (or $A \cdot B$ )		
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

Venn diagram

Figure 4-2-41 represents a comparison between the AND circuit that performs AND operations and a switch and a light bulb. Here, by establishing a correspondence between the switch "opens" and "1," and between "closes" and "0," as well as between the light bulb "lights" and "1" and "does not light" and "0" a truth table can be created.

**Figure 4-2-41** AND circuit of a switch and a light bulb



The electric circuit is often represented using the US army MIL symbol (US MILitary standard, MIL-STD). Figure 4-2-42 shows the AND circuit represented with the MIL symbol.

**Figure 4-2-42** AND symbol



## ②OR circuit

OR circuits are the circuits that perform OR operations (logical sum operations), as the one shown in Figure 4-2-43. If either A or B input value is "1," "1" is input.

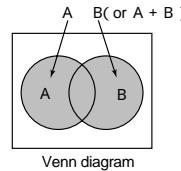
Besides "A OR B," OR operations are represented as " $A + B$ " or as " $A \vee B$ "

**Figure 4-2-43**

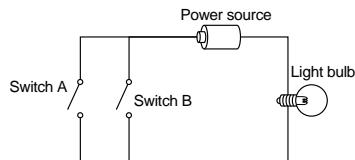
Truth table and Venn diagram of OR operations

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Truth table



**Figure 4-2-44** OR circuit of a switch and a light bulb



Switch A	Switch B	Light bulb
Opens (0)	Opens (0)	Does not light (0)
Opens (0)	Closes (1)	Lights (1)
Closes (1)	Opens (0)	Lights (1)
Closes (1)	Closes (1)	Lights (1)

**Figure 4-2-45** OR symbol

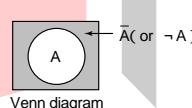
③ NOT circuit

NOT circuits are the circuits that perform NOT operations (negation operations), as the one shown in Figure 4-2-46. The opposite of input value is output. The negation of "1" is "0" and the negation of "0" is "1". Besides "NOT A", "NOT operations are represented as "A'" or as " $\neg A$ ".

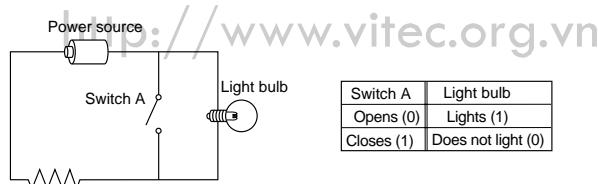
**Figure 4-2-46**  
Truth table and Venn diagram of NOT operations

A	NOT A
0	1
1	0

Truth table

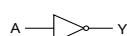


**Figure 4-2-47** NOT circuit of a switch and a light bulb



Switch A	Light bulb
Opens (0)	Lights (1)
Closes (1)	Does not light (0)

**Figure 4-2-48** NOT symbol



## (2) Combination of the basic circuits

Through the combination of the basic circuits, EOR circuits, NOR circuits and NAND circuits can be composed.

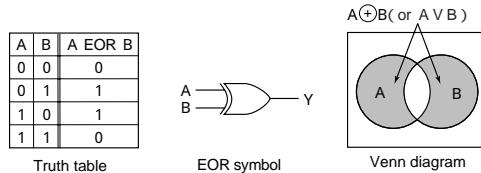
### ①EOR circuit (Exclusive logical sum operation circuit)

Figure 4-2-49 shows the truth table, MIL symbol and Venn diagram of EOR operations.

Besides, "A EOR B," EOR operations are represented as " $A \oplus B$ " or as " $A \vee B$ ."

It should be noted that " $A \oplus B$ " is an operation that means the same as " $A \cdot \bar{B} + A \cdot B$ ."

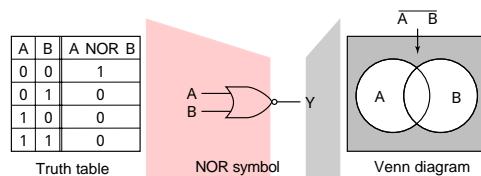
**Figure 4-2-49** Truth table, EOR symbol and Venn diagram of EOR operations



### ②NOR circuit (Negative logical sum operation circuit)

NOR operations are the negation of OR operations. NOR circuits result in the combination of a NOT circuit in the output side of an OR circuit.

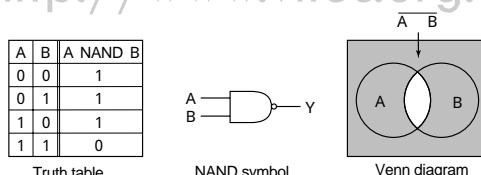
**Figure 4-2-50** Truth table, NOR symbol and Venn diagram of NOR operations



### ③NAND circuit (Negative logical product operation circuit)

NAND operations are the negation of AND operations. NAND circuits result in the combination of a NOT circuit in the output side of an AND circuit.

**Figure 4-2-51** Truth table, NAND symbol and Venn diagram of NAND operations



### ④Addition circuit

Through the combination of several basic circuits and combinational circuits, circuits that perform 1-digit binary additions called addition circuits can be created. There are two types of addition circuits, half adder circuits and full adder circuits.

### a. Half adder circuit (HA)

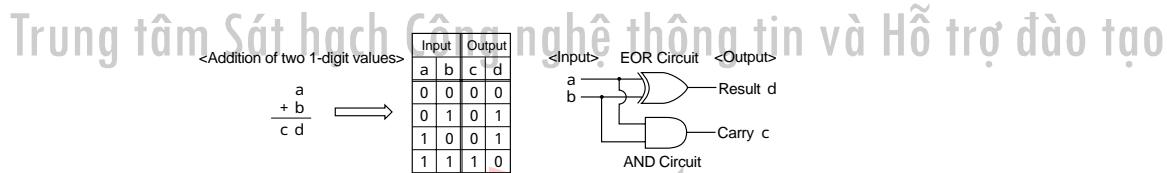
The addition of the 1 digit binaries A and B, A + B can be performed in the following four ways.

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}
 \quad
 \begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}
 \quad
 \begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}
 \quad
 \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

↑  
Carry

The circuit that performs these binary operations is created by the combination of one AND circuit and one EOR circuit. This circuit is called the half-adder circuit. (Figure 4-2-52)

**Figure 4-2-52** Truth table of the half-adder circuit and half-adder circuit



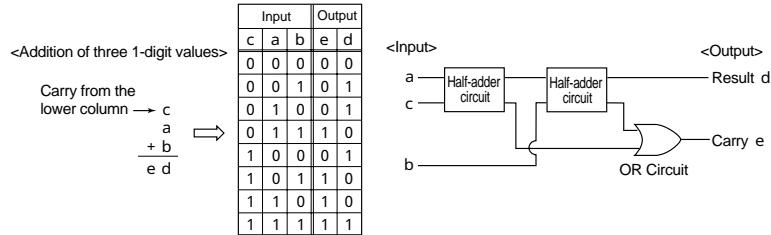
### b. Full adder circuit (FA)

The addition of the 1 digit binaries A, B and C, A + B + C can be performed in the following eight ways:

$$\begin{array}{r} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ + 0 & + 1 & + 0 & + 1 & + 0 & + 1 & + 0 & + 1 \\ \hline 0 & 1 & 1 & 10 & 1 & 10 & 10 & 11 \end{array}$$

The circuit that performs these binary operations is created by the combination of two half adder circuits and one OR circuit. This circuit is called the full adder circuit.

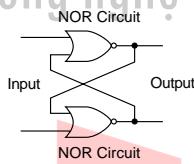
**Figure 4-2-53** Truth table of the full adder circuit and full adder circuit



### (3) Sequential circuit

The sequential circuit is a circuit in which the output is established according to the current input and status preserved (past input). The sequential circuits, whose status changes with time, are composed of a flip-flop circuit and are used in registers, etc.

**Figure 4-2-54**  
Flip-flop circuit



## 4.2.4 Multi-processor

Multi-processor systems are introduced to improve the performance and reliability of the system. Multiple processors in parallel with each processor having a dedicated function. When failure occurs, the processor will do a switch and the remaining processors will distribute the load among themselves.

### (1) Symmetric Multi-processor

Symmetric multi-processors systems are systems where the memory is shared among all the processors executing the same OS. Competition for the use of memory among the processors since the memory is common to all. This means a large number of processors cannot be connected.

Message passing distributed memory multi processor systems are systems where each processor has its own private block of memory. A high speed input output port is used to transfer the data between the different blocks.

### (2) Array processor

High speed scientific computing is done with array processors which use pipeline processing.

Large scale or dedicated mathematical processors

The sub units' acts are in a queue passing the result to the next unit after it has finished its part. This is known as vector processing. Most supercomputers utilized this method of high speed computing.

### (3) Parallel

Multiple processors cooperate with multiple tasks being performed to execute one job.

#### SISD (Single Instruction Single Data Stream)

One instruction stream operating on a single data element and is not parallel

#### SIMD (Single Instruction Multiple Data Stream)

Each instruction may operate on more than one data element and is synchronous.

#### Parallel SIMD

The same instruction is executed by all processors operating on different sets of data.

#### MIMD (Multiple Instruction Multiple Data Stream)

Each processor has its own instruction stream acts on its own data stream independent of the other processors

### 4.2.5 Processor performance

The performance of the processor, which can be considered as the central nervous system of the units that compose the computer system, is measured using the number of instructions that can be executed in a unit of time as an index. These indexes are indicated below.

#### (1) MIPS

MIPS is an acronym of Million Instructions Per Second, and indicates in million units the number of instructions that can be executed in one second. In other words, a 1 MIPS processor is a processor that can execute one million instructions per second. Basically, the larger the number of instructions that can be executed, the higher the value. The term MIPS is mainly used to indicate the performance of processors of high end mainframe computers. However, it is meaningless to use this index to compare processors of different types of machines that execute different instruction contents.

#### (2) Clock

In order to set the pace in which the micro-instructions, which are basic operations, are executed, the processor has a clock inside. A quartz crystal oscillator that pulses in regular intervals when electrical current passes through is used in this clock. The time taken for this oscillator to pulse once (one cycle) is called clock, the basic operations of the processor are performed according to this clock. The number of clocks varies according to the instruction.

The clock reciprocal number is called clock frequency. Clock frequency is used as an index to measure the performance of a personal computer.

**[Example]** Performance of a processor with a clock frequency of 500 MHz

$500 \text{ MHz} = 500 \times 10^6 \text{ Hz} = 500,000,000 \text{ Hz}$  (times/second);    500 hundred million pulses per second

$$\frac{1}{0.5 \times 10^9} = 2 \times 10^{-9} = 2 \text{ nano (seconds/times);}$$

1 pulse for every 2 nanoseconds

### (3) CPI (Cycles Per Instruction)

A CPI is the number of clocks required to execute instruction. This index indirectly indicates the execution time of one instruction

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

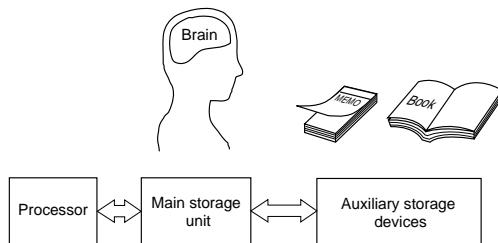
## 4.3 Memory architecture

### 4.3.1 Memory types

The storage function, which is the most important characteristic of the computer, is made possible by the storage units. According to the application, the storage units are divided into main storage unit and auxiliary storage devices.

Figure 4-3-1

Main storage unit and auxiliary storage devices



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

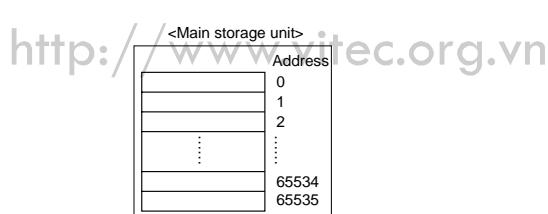
#### (1) Main storage unit

The main storage unit, which is directly connected to the processor by a signal line called bus, is a device that stores the programs and data to be executed by the processor.

The place where data is stored is generally called memory and it has a series of consecutive addresses. By specifying one of these addresses, information is retrieved from or stored in the main storage unit. The act of exchanging data with the main storage unit is performed in the above mentioned cases is called access. In this way, the system that stores in advance the program to be executed in the main storage unit is called the stored program system (or internally programmed system) and it is a basic approach in current computers.

Figure 4-3-2

Main storage unit address concept



The main storage unit is composed of semiconductor elements called RAMs (Random Access Memory). A RAM's property is such that when the computer is turned off, the contents stored are lost. Therefore, if there is data to be saved after the process is finished, it is stored in the auxiliary storage devices.

## (2) Auxiliary storage devices

The auxiliary storage devices are devices that play the supporting role of making up for the shortage of storage capacity of the main storage unit. They are also called external storage units. The auxiliary storage devices are not included among the five main units of the computer, but they are indispensable for current computers. Large volumes of data and programs are stored/saved in the auxiliary storage devices and when the data or program required to perform a process is not found in the main storage unit, it is transferred (copied) from the auxiliary storage devices to the main storage unit in order to perform the process.

Likewise, the data to be saved after the process is completed is transferred to the auxiliary storage devices and saved there. Since the auxiliary storage devices have the property that, even when the computer is turned off, the contents are not lost, they can be used as input/output units of large volumes of data.

The following are the main auxiliary storage devices.

- Magnetic tape unit
- Magnetic disk unit
- Floppy disk unit (Flexible disk unit)
- Optical disk unit
- Magneto-optical disk unit

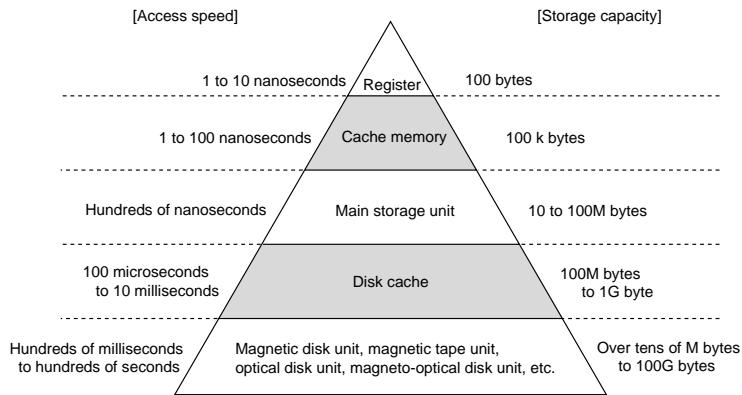
### 4.3.2 Memory capacity and performance

#### (1) Memory hierarchical structure

The computer is composed of the register inside the processor, the main storage unit , the auxiliary storage devices, etc. The storage capacity and the processing speed of each of these devices differ. A is shown in Figure 4-3-3, the access speed is as follows:

(High speed) Register inside the processor > main storage unit > Auxiliary storage devices (Low speed.) This access speed difference is absorbed by a device called the buffer.

**Figure 4-3-3** Memory hierarchical structure



## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### (2) Access time

The access time and cycle time indicate the operation speed of the storage units.

The access time is the time elapsed from when the processor sends the read/write instruction to the storage unit until the data delivery /acceptance is completed.

For the processor to access the main storage unit data, the following three stages are necessary.

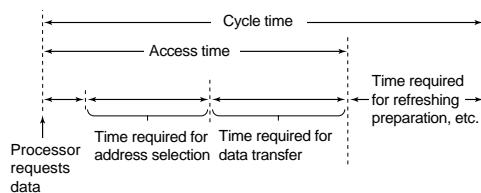
- ①The time during which the processor requests the data readout
  - ②The time during which the processor selects the main storage unit address with the address bus
  - ③The time during which the data of the selected address is transferred through the data bus
- In other words, ①+②+③ represent the time elapsed from when the data access request is sent until the data transfer is completed. The lapse of time is called the access time.

### (3) Cycle time

<http://www.vitec.org.vn>

Among the storage elements of the storage unit, when data is to be stored in the capacitor, there are some whose memory fades with time, as the DRAM. In this case, the refreshing operation that rewrites data at regular intervals becomes necessary. For that reason, after the data transfer is completed, a preparation time in order to receive the next request becomes necessary. The lapse of time that includes the point up to this preparation is called cycle time.

**Figure 4-3-4**  
Access time and  
cycle time



### **4.3.3 Memory configuration**

As was mentioned above, the memory used in the computer can be classified into hierarchies. To provide for the occurrence of malfunctions or failures, these devices are equipped with data error detection and error correction functions. These functions are implemented by several Error Correcting Codes (ECC).

#### **(1) Magnetic disk**

The series of errors caused by a small scratch, etc, on a magnetic disk are called burst errors. The Cyclic Redundancy Check code (CRC code) is adopted in the disk unit to detect these burst errors. Error detection is possible with the CRC code system.

#### **(2) Magnetic tape**

The magnetic tape indicates 1-byte data in the transverse direction of the tape. In order to detect bit errors in this transverse direction, a parity check system, which can detect odd numbers of bit errors by appending vertical parity bits, is adopted. In addition, CRC code is adopted to detect burst errors in a transverse direction

#### **(3) Main memory**

In the main memory, due to the high probability of the occurrence of non consecutive random errors, the Hamming code, which can detect single bit errors and double bit errors, is adopted. It should be noted that, generally speaking, the main memory error detection is performed in general purpose computers but it is not performed in personal computers.

#### **(4) Memory protection system**

Since various kinds of information is handled in a computer, depending on the characteristics of the information, a function that limits the users is necessary. This function is called memory protection and it protects the instructions and data stored in the main memory, auxiliary storage devices and other memories under specific conditions.

When the memory is accessed, operations like the ones mentioned below are performed:

- Read
- Write
- (Instruction) execute

The right to perform these operations is called access right. Data has read/write rights, but instructions do not have a right to write. On the other hand, instructions have a right to execute, but data do not. When an illegal access that violates these access rights occurs, the control is transferred to the OS as a result of the interruption handling routine.

Like wise, as protection mechanism of the main memory implemented by the hardware, the protective boundary register system, in which a dedicated register specifies the accessible

domains, TLB (Translation Look Aside Buffer) system, which applies the memory protection function in a virtual address space, etc, exist.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

## 4.4 Auxiliary storage devices

### 4.4.1 Types and characteristics of auxiliary devices

As was mentioned above, computer storage units are divided as follows:

- Main storage unit
- Auxiliary storage devices

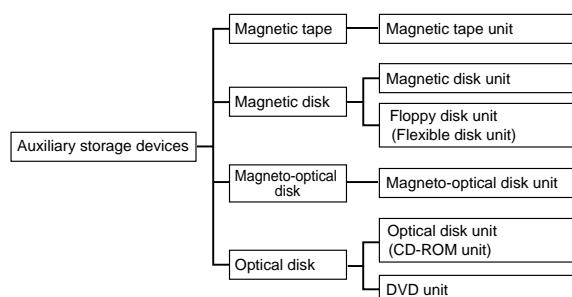
The main storage unit is equivalent to the human brain, while auxiliary storage devices are equivalent to notebooks and texts. Auxiliary storage devices are devices that store and save programs and data while they are not being executed. Likewise, as when one reads a text and writes down the necessary information, or when one writes in a letter the things to be transmitted to another person, these auxiliary storage devices also play the role of input devices and output devices.

There are two types of auxiliary storage device: devices that store data magnetically, as the magnetic tape unit, magnetic disk unit, the floppy disk unit (flexible disk unit), and the magneto-optical disk unit and devices that store data optically as the optical disk unit (Figure 4-4-1). The main storage unit stores programs and data to be used by the processor for instruction execution, but it has a big problem: stored content is lost when the computer is turned off. On the other hand, compared to the main storage unit, the operating speed of auxiliary storage devices is low, but they can store a large volume of data and, even if the computer is turned off, the stored data is retained semi-permanently.

It should be noted that besides these devices, there is also a semiconductor disk unit that is, for example, the flash memory used as an auxiliary storage unit in digital cameras and notebook personal computers. This unit is composed of semiconductors (EEPROM(s)), it does not operate mechanically, and it electrically performs data reading/writing processes at high speed. However, since it cannot store large volumes of data, it is used as the storage unit of small devices with low power requirements.

Here, the operation principles and characteristics of the typical auxiliary storage devices will be explained.

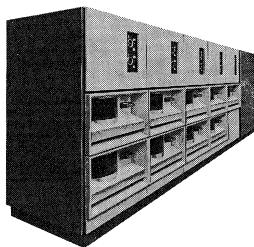
**Figure 4-4-1**  
Diverse auxiliary storage devices



## (1) Magnetic disk unit

Magnetic disk unit is devices that store data using magnetic disks. It is the auxiliary storage device most widely used in today's computer systems. Magnetic disks for personal computers or workstations are also called fixed disks or hard disks but the mechanism is the same.

Figure 4-4-2  
Magnetic disk unit



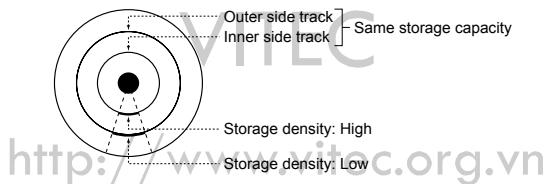
# Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### ① Magnetic disk

#### a. Track

The magnetic disk is a circular shaped magnetic body in which data is recorded along rings called tracks. There are several tracks concentrically set on the magnetic disk. The length of the outer track and that of the inner tracks differ, because of the difference of the storage capacity, the volume of data stored is the same in every track.

Figure 4-4-3  
Data recording side  
of the magnetic disk



The storage density of the disk is based on the average track length and the storage capacity of the magnetic disk is determined by the number of tracks and the storage density of one disk.

#### b. Cylinder

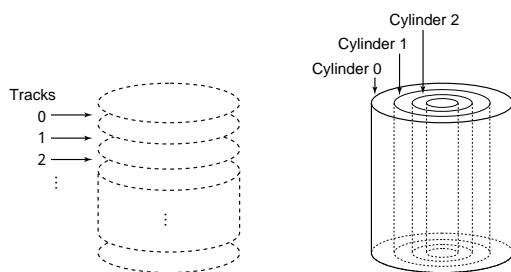
In a magnetic disk unit, which is composed of multiple magnetic disks, the group of tracks with the same radius on each of the disks is set as one data storage area. This storage area is called a cylinder.

When data is stored in a cylinder, if, for example, the data cannot be completely stored on track 0 of cylinder 1, it can be stored on track 1, track 2, etc. of the same cylinder. Therefore, since data access can be performed without moving the access arm (that is, magnetic head), it is

extremely efficient.

To put it in another way, the cylinder is a group of tracks that can be read and written by multiple magnetic heads if the access arm of the magnetic disk unit is fixed.

**Figure 4-4-4**  
Tracks and cylinders



### c. Storage capacity

The storage capacity of the magnetic disk can be determined as follows:

Storage capacity of 1 track x Track number of 1 cylinder x Cylinder number of the magnetic disk

#### Example

Given a magnetic disk with the following specifications, the storage capacity of this magnetic disk is calculated:

[Magnetic disk specifications]

- Cylinder number: 800 cylinders
- Track number /cylinder number : 19 tracks
- Storage capacity/track: 20,000 bytes

The storage capacity per cylinder is as follows:

$$20,000 \text{ bytes/track} \times 19 \text{ tracks/cylinder} = 380,000 \text{ bytes/cylinder} = 380 \text{ kB (kilo bytes)}$$

Since the number of cylinders on this disk is 800, the storage capacity of the magnetic disk is as follows:

$$380 \text{ kB/cylinder} \times 800 \text{ cylinders} = 304,000 \text{ kB} = 304 \text{ MB (Mega bytes)}$$

An example of the calculation of storage capacity when blocking is performed is shown below:

#### Example

Given a magnetic disk with the following specifications, the number of cylinders required when 80 thousand records of 200 bytes each are stored in a sequential access file of 10 records/block per magnetic disk is calculated. It should be noted that the block recording cannot be extended over multiple tracks.

[Magnetic disk specifications]

- Cylinder number: 400 cylinders

- Track number /cylinder number : 19 tracks
- Storage capacity/track: 20,000 bytes
- Inter-block gap (IBG): 120 bytes

1. First, the number of blocks of the whole file is calculated.

Since the number of records is 80,000 and the blocking factor is 10, the number of blocks is determined as follows:

$$80,000 \div 10 \text{ records/block} = 8,000 \text{ blocks}$$

2. The length of 1 block, including the inter-block gap is calculated.

$$200 \text{ bytes/record} \times 10 \text{ records/block} \times 120 \text{ bytes/block} = 2,120 \text{ bytes/block}$$

3. The number of blocks that can be recorded in 1 track is calculated.

$$20,000 \text{ bytes/tracks} \div 2,120 \text{ bytes/block} = 9.43 \dots \text{blocks./track}$$

Since a block cannot be recorded across multiple tracks, the decimals are omitted, and the number of blocks that can be recorded in 1 track becomes 9 blocks/track

4. The number of tracks required for the whole file is calculated.

$$8,000 \text{ blocks} \div 9 \text{ blocks/track} = 888.88\dots \text{tracks}$$

5. The number of cylinders required to record the whole file.

$$889 \text{ tracks} \div 19 \text{ tracks/cylinder} = 46.78\dots \text{cylinders}$$

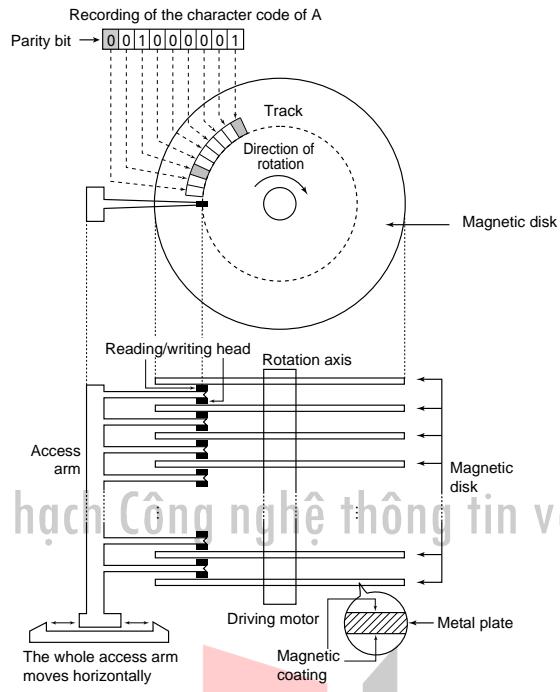
Rounding it up to the next whole number, it becomes 47 cylinders.

## ② Magnetic disk unit structure and operation principles

The magnetic disk unit has multiple magnetic disks, which it rotates at high speed in order to record data along concentric tracks. On each recording side, an access arm with a magnetic head moves forward and backward to reach the track position where data is to be read or recorded.

Compared to the sequential access of the magnetic tape unit, in which access can only be performed in order from the beginning, in the magnetic disk unit, besides sequential access, direct access to the desired recording position can also be performed. Auxiliary storage devices in which this direct access can be performed are called direct access storage devices (DASD).

**Figure 4-4-5** Structure of the magnetic disk unit



#### a. Variable type and sector type

By recording method, magnetic disk unit is classified into "Variable type" and "Sector type".

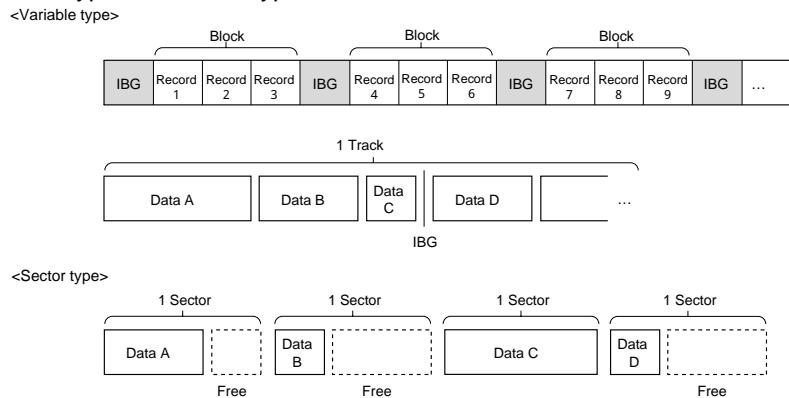
- **Variable type**

In the variable type unit, data reading and writing is performed on a block basis, as in the magnetic tape. A block is a group of data called record and there is an IBG between blocks. In the unit, there is no gap between data or IBG). Reading and writing of any number of bytes can be started from any track position

- **Sector type**

In the sector type, one track is divided into approximately 20 small units called sectors. Data reading and writing is performed on a sector basis. The reading/ writing position is specified with the sector number of the selected track

**Figure 4-4-6** Variable type and sector type



Generally, the variable type is used in magnetic disks, and the sector type is used in floppy disks and hard disks

Generally, the variable type is used in magnetic disks, and the sector type is used in floppy disks and hard disks.

#### b. Parity check

When data is recorded on a magnetic disk, data is written on the track bit by bit using the magnetic head. The same method is used to read data. When this process is performed, as in the magnetic tape, in order to detect reading or writing errors, a parity bit (1 bit) is appended to perform the parity check.

#### c. Defragmentation

In personal computer hard disks, data is stored and deleted repeatedly. Since it is improbable that all the data to be stored will have the same size, a small volume of data can be stored after a big volume of data is deleted, or vice versa. As a consequence, there would be free sectors scattered about and a drop in access efficiency. This status is called defragmentation; in order to solve it, a function called defragmentation is implemented in the OS.

#### ③ Magnetic disk unit performance

The performance of the magnetic disk unit is measured according to access time and storage capacity. Since the storage capacity was already explained in ①, here, the access time significance and the calculation method will be explained.

##### a. Access time

Access is the generic term for the act of reading specific data from the magnetic disk and writing it on a specific cylinder or track. Access time is calculated through the addition of the following:

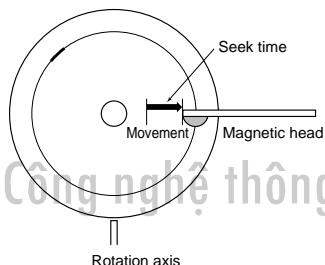
- Seek time
- Search time
- Data transfer time

#### ● Seek time

In order to access the target data, the magnetic head has to be moved to the position of the track where the target data is stored. The time it takes to move the magnetic head is called seek time.

**Figure 4-4-7**

Seek time



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

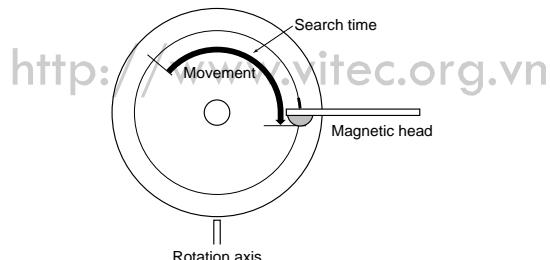
Since seek time differs depending on the distance between the position of the target track and the current position of the magnetic head, an average value is used as the actual seek time. This value is called average seek time.

#### ● Search time

The search time is the elapsed time until the target data reaches the magnetic head position. (Figure 4-4-8)

**Figure 4-4-8**

Search time



As with average seek time, depending on the data position, there are cases where the search time is 0, as well as cases where a wait of 1 revolution is. Therefore, 1/2 revolution of the magnetic disk is used as the search time. This value is called average search time.

## ● Data transfer time

The time elapsed between when the magnetic head data access starts and when the transfer is completed is called data transfer time.

Therefore, the time elapsed between when the magnetic disk unit starts access and when the data transfer is completed, that is, the access time, is calculated as follows:

Seek time + Search time + Data transfer time

Strictly speaking, as in the above mentioned formula, the time elapsed between when the access request occurs and the magnetic disk starts operating is the access time.

$$\text{Access time of the magnetic disk unit} = \text{Average seek time} + \text{Average search time} + \text{Data transfer time}$$

### Example

Given a magnetic disk unit with the following specifications, the access time of this magnetic disk when a record of 9,000 bytes is processed is calculated.

[Magnetic disk unit specifications]

- Capacity per track                    15,000 bytes
- Magnetic disk rotation speed: 3,000 revolutions/minute
- Average seek time:                20 milliseconds

1. First, the average search time is calculated.

Since the rotation speed of the magnetic disk is 3,000 revolutions/minute, through the following operation,

$3,000 \text{ revolutions/minute} \div 60 \text{ seconds/minute} = 50 \text{ revolutions/second}$ ,  
it is determined that the magnetic disk makes 50 revolutions per second. Therefore, the time required to make 1 revolution is as follows:

$$1 \text{ revolution} \div 50 \text{ revolutions/second} = 0.02 \text{ seconds/revolution} = 20 \text{ milliseconds}$$

Since the average search time is the time required to make 1/2 revolution, it is as follows:  
 $20 \text{ milliseconds} \div 2 = 10 \text{ milliseconds}$

2. Since in 1 revolution, the information contained in 1 track passes through the magnetic head, considering that the disk makes 50 revolutions per second, the data transfer speed is as follows:

$$\text{Data transfer speed} = 50 \text{ tracks/second} \times 15,000 \text{ bytes/track} = 750 \times 10^3 \text{ bytes/second}$$

Based on this data transfer speed, the time to transfer 9,000 bytes of data can be calculated as follows:

$$(9 \times 10^3 \text{ bytes}) \div (750 \times 10^3 \text{ bytes/second}) = 0.012 \text{ seconds} = 12 \text{ milliseconds}$$

3. Therefore, the access time is as follows:

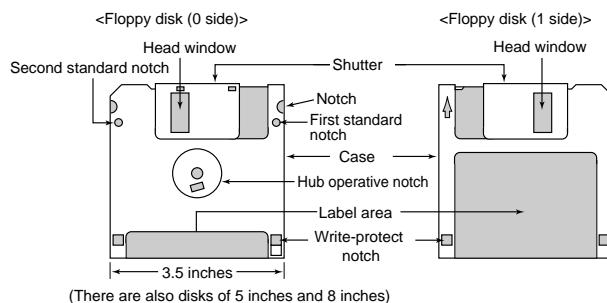
$$\begin{aligned} & \text{Average seek time} + \text{Average search time} + \text{Data transfer time} \\ &= 20 \text{ milliseconds} + 10 \text{ milliseconds} + 12 \text{ milliseconds} = 42 \text{ milliseconds} \end{aligned}$$

## (2) Floppy disk unit

The floppy disk unit is also called a flexible disk unit. In floppy disk units data random access is possible, and, since the floppy disk itself, which is storage medium, is low-priced and easy to carry about, its use has widely spread. As an auxiliary storage device of personal computers, it is the most ordinarily used device.

**Figure 4-4-9**

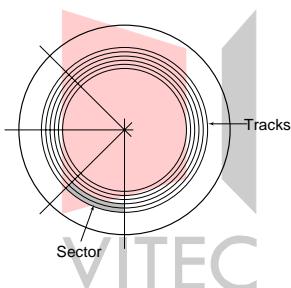
Floppy disk  
(Flexible disk)



**Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo**  
The recording method of the floppy disk is the sector method, and as it is shown in Figure 2-4-10; the track is divided into sectors, and the data is recorded on a sector basis.

**Figure 4-4-10**

Data recording side  
of the floppy disk



### ① Floppy disk

#### a. Types

<http://www.vitec.org.vn>

Among floppy disks, there are magnetic disks that measure 8 inches, 5 inches and 3.5 inches, but the most common disks today are 3.5 inch disks, while 8 and 5 inch disks are almost never used. There are also the following 2 types of 3.5 inch floppy disks, depending on the storage density.

- 3.5 inch 2 HD (double sided High Density)  
Storage capacity: 1.2 to 1.4 megabytes (MB)
- 3.5 inch 2 DD (double sided Double Density)  
Storage capacity: 640 to 720 kilobytes (kB)

**Figure 4-4-11**

Example of the specifications of a floppy disk (2HD)

	1.4MB	1.2MB
Sides available for use	2	2
Track number/side	80	77
Sector number/track	18	8
Storage capacity (B)/sector	512	1,024

Likewise, there is a floppy disk whose storage capacity is 120 MB (UHD) and a disk called Zip whose storage capacity is 100 MB. Both of them are compatible with the 3.5 inch disk (2DD/2HD), but they have not come into wide use.

### b. Storage capacity

The calculation of the access time of floppy disk units is the same as that for magnetic disk units. Therefore, here, the storage capacity of the sector method will be explained.

As was shown in Figure 4-4-11, among floppy disks, the sides available for use, the number of tracks per side, the number of sectors per track, etc. differs.

The storage capacity of a floppy disk is calculated using the following values:

Storage capacity per sector x Number of sectors per track x Number of tracks per side x Number of sides (One side or both sides)

#### Example

Given a floppy disk with the following specifications, the storage capacity is calculated.  
[Specification of a floppy disk]

- Sides available for use: 2 sides
- Track number/side: 80 tracks
- Sector number/track: 9 sectors
- Storage capacity/sector: 1,024 bytes

<http://www.vitec.org.vn>

The storage capacity of 1 track is as follows:

$$1,024 \text{ bytes/sector} \times 9 \text{ sectors/track} = 9,216 \text{ bytes/track}$$

Therefore, the storage capacity of 1 side is as follows:

$$9,216 \text{ bytes/track} \times 80 \text{ tracks} = 737,280 \text{ bytes} \equiv 737 \text{ kB}$$

And, since the sides available for use of the floppy disk are 2 (sides), the following is the storage capacity:

$$737 \text{ kB} \times 2 = 1,474 \text{ kB}$$

Approximately 1.474 MB

### ② Floppy disk unit structure and operation mechanism

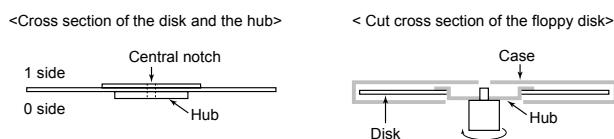
Basically, the floppy disk unit has the same structure as the magnetic disk unit. However, only 1

floppy disk is used and the sector method, in which the tracks are divided into sectors, is the recording method.

The number of tracks and the division of the sectors depend on the operating system used. Therefore, when the user uses a floppy disk, it has to be initialized in the format specified by the operating system. The process is called formatting.

When the floppy disk cartridge is installed in the floppy disk unit, the disk contained in the cartridge rotates. The magnetic head directly traces the magnetic disk surface of the disk, in order to read/write information; the data access time is longer than that of the magnetic disk unit and the magnetic tape unit.

**Figure 4-4-12** Floppy disk structure



### (3) Optical disk (CD, DVD) unit

Besides the magnetic disk unit, the magnetic tape unit and the floppy disk unit, there are various other kinds of auxiliary storage devices. Optical disk units, magneto optical disk units, DVD units, etc. are used to store/save image processing data of extremely large volume or as storage devices of large volume packaged software. These devices can store large volumes of data through a mechanism that reads out information using light reflection.

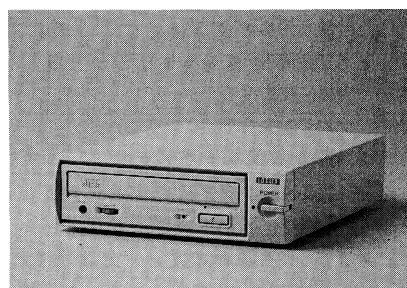
In addition to floppy disk units and hard disk units, as auxiliary storage devices, today's standard personal computer systems are also equipped with CD ROM units. The role played by the CD ROM as a medium to supply software packages to the general marketplace, and as a multimedia storage medium, is extremely important.

#### ① Optical disk

The surface of the optical disk is covered with a hard plastic that makes it resistant to scratches and dust.

Furthermore, since a laser beam is used to read out data, the head does not touch the recording surface directly, so no friction is caused. Among optical disks, CD ROM use in particular is expanding rapidly.

**Figure 4-4-13**  
CD-ROM unit



Among optical disks, there is the music CD (Compact Disc), the CD-G (CD Graphic) for image data, the CD-I (CD Interactive) for interactive applications, the CD-R (CD Recordable), etc. And, as a computer storage medium, the CD-ROM (CD Read Only Memory) is widely used. Furthermore, as an optical disk that supports the multimedia era, the DVD, which has great capacity and high image quality and is capable of storing animated images and audio, exists. Here, the CD ROM and the DVD specifically will be explained.

## ②CD-ROM

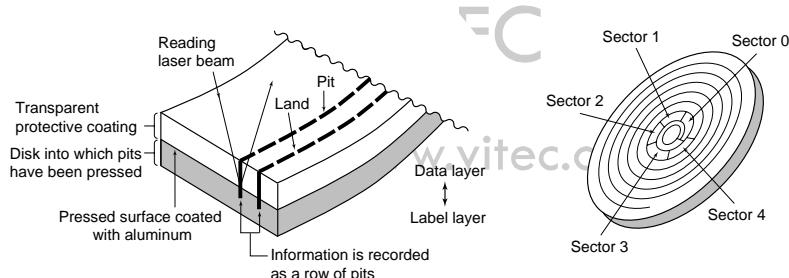
The CD used as a computer storage medium is the CD ROM. The external appearance, diameter, thickness, etc of the CD-ROM is the same as that of a CD (diameter: 12 cm, thickness: 1.2 mm, single disk diameter: 8 cm), but the error correction function, file system, logical format, etc. differ. Since the CD-ROM logical format uses the international industrial standard ISO 9660, it has high compatibility.

However, since the CD-ROM is a read only disk, data can be read but cannot be written.

### a. Structure

The CD ROM, which is a disc-shaped storage medium, does not have a concentric track structure as does the magnetic disk or floppy disk. Tracks of continuous sectors are connected in a spiral as in vinyl records and the data is stored from the inner side to the outer side. Figure 4-4-14 shows a magnification of the data recording surface of the CD ROM.

**Figure 4-4-14** CD-ROM data recording surface



The CD ROM stores "0" and "1" information using the pits and lands of the data recording layer. In order to read out data, a laser beam is applied and the optical head reads out the changes in intensity of the reflected light.

### b. Storage capacity

By creating a master disk with a negative replica of these pits and lands and pressing it against plastic disks, a large quantity of CD ROMs can be replicated at high speed and low cost. 1 CD ROM (12 cm) has a storage capacity of approximately 600MB which makes it an indispensable storage medium to process the enormous volume of information of multimedia

data.

### ③CD ROM unit structure and performance

#### a. Structure

Basically, the structure of the CD ROM unit is the same as that of the magnetic disk unit. The difference is that data is not read out using a magnetic head, but an optical head that detects the laser beams.

#### b. Performance

The CD ROM unit performance is measured according to the head seek time and the data transfer rate.

##### ● Seek time

The CD ROM seek time is extremely slow compared to that of the magnetic disk unit. While the seek time of the magnetic disk unit is measured in tens of milliseconds, it is measured in hundreds of milliseconds in the CD ROM unit. This is due to the use of a heavy lens in the read head and to the CD ROM structure (the data storage format uses a spiral track as in a vinyl record).

##### ● Transfer rate

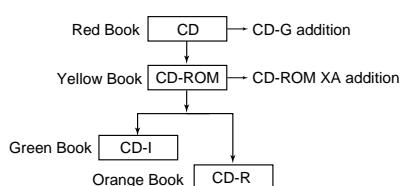
The data transfer rate is expressed in numeric values that represent how much data can be transferred in comparison with audio CDs. The audio CD player can read approximately 150k bytes of data in 1 second, which is an extremely low rate as compared to computer processing speed. Therefore, units with transfer speeds 2 times or 3 times as fast as the transfer rate for audio CDs began to be developed and today the transfer rate has reached levels of 10 times and 20 times as fast.

### ④Optical disk specifications

The optical disk, which was born from the audio CD and is widely used as a computer storage medium, has multiple variations that make the best use of its high storage capacity, portability, mass production through press replication processing, and other advantages. The standard specifications of these optical disks have been established and the basic standard of each of them is called red book, yellow book, etc.

These names were given after the color of the cover of the binder in which the standard specifications were kept.

**Figure 4-4-15**  
Optical disks



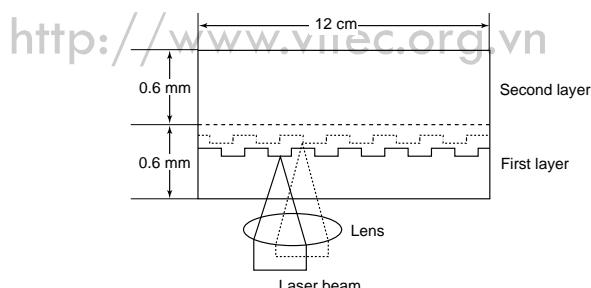
The outline of each standard is indicated below:

- Red Book  
The Red Book is the basic CD standard and describes the physical specifications of the CD. The standard of the CD-G, which made possible the storage of CD (Computer Graphics) in the audio CD, etc., has also been added.
- Yellow Book  
The Yellow Book is the basic CD standard and describes the physical format of the CD-ROM. As an extension to the Yellow Book, the CD-ROM XA , which made possible extended audio playback and multiple graphics recording, is also specified and its specification becomes the bridge between the CD-ROM and the CD-I.
- Green Book  
The Green Book describes the specifications of the CD-I (CD-Interactive), which is capable of storing audio, images, CGs, characters, programs, data etc.
- Orange Book  
The Orange Book defines the physical structure of the CD-R (CD-Recordable) , a writable CD  
Among the CD-Rs, the CD-WO (Write Once), a recordable type in which once information is written, the content cannot be rewritten, and the CD-MO (CD-Magneto Optical) a re-writable type which can be rewritten exist. this CD-R is used in photo CDs (camera film images recorded in CD)

## ⑤DVD

The DVD \*(Digital Versatile Disk or Digital Video Disk) is an optical disk capable of storing approximately 2 hours of animated images and audio data. The external appearance of DVD disks is the same as that of CD-ROMs, 12 cm of diameter and 1.2 mm of thickness. However, while the recording side of the CD-ROM consists of one side (1 layer), the DVD has a maximum of 2 layers, and data can be stored on both sides.

**Figure 4-4-16**  
DVD structure



At present, DVD-ROMs are commercialized and can be played on DVD players. Likewise, the use of DVD-ROM units in personal computers is expanding.

DVD-ROM storage capacity is as follows:

- Single layer single sided recording: 4.7 Gbytes
- Dual layer single sided recording: 8.5 Gbytes
- Single layer dual sided recording: 9.4 Gbytes
- Dual layer single sided recording: 17 Gbytes

The DVD uses a compression method of animated images called MPEG2, which enables playback of extremely clear images. Due to its large capacity and high image quality, the DVD is attracting attention as a computer auxiliary storage medium/device. The standards of the read-only DVD-ROM, the recordable DVD-R and the re-writable DVD-RAM have been established.

#### **(4) Magneto Optical disk (MO) unit**

The magneto optical disk unit has almost the same structure as the magnetic disk unit. While data can only read from the CD-ROM unit, data can be read and written onto the magneto optical disk unit.

The main characteristic of the magneto optical disk is that the data reading and recording methods differ.

Data recording is performed by applying the laser beam to the magnetized recording side to heat it then record high density information using the magnetic head. Data reading is performed focusing the laser beam on the magnetic disk and reading out the polarization direction of the reflected light. In other words, data writing is performed using a magnet while data reading is performed using a beam.

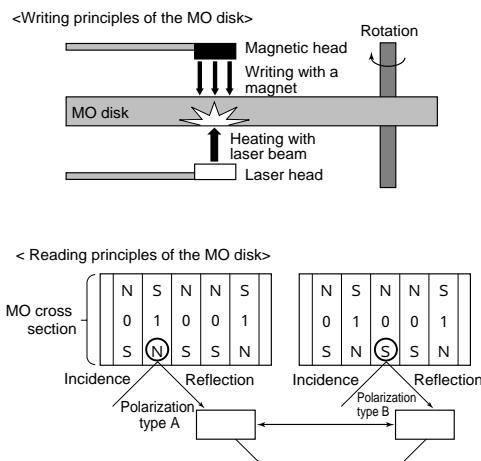
Figure 4-4-17 shows the writing and reading principles of the magneto optical disk unit. In this diagram, when the magnetization of the magneto optical disk from top to bottom is from N to S the value is "0" and when it is from "S" to "N", "1" is the read out value.

The magneto optical disk shipped today as a standard is the 3.5 inch disk, with a storage capacity of 128 MB or 230 MB. As a substitute for the floppy disk, the magneto optical disk has come into wide use as a computer auxiliary storage medium.

## (5) Semiconductor disk unit

Figure 4-4-17

Data reading and writing operation of the magneto optical disk unit



The semiconductor disk unit is a storage unit of high speed and large capacity, which uses flash memories and other devices. In most cases, it is used in high end mainframe computers as a storage unit positioned between the main storage unit and the auxiliary storage devices. It has the advantage that, despite having several G bytes of storage capacity, its access time is 1/100 that of the magnetic disk unit.

### 4.4.2 RAID types and characteristics

Since computers have largely penetrated into our daily life, playing a critical role, high reliability is required together with high performance. Therefore, in order to achieve high reliability, a great variety of technologies are used. The method to duplicate system components to allow continued operation in case of a failure of a certain component is called fault tolerance technology. RAID (Redundant Array of Inexpensive Disk) is one such technology.

RAID is a method that consists of the parallel use of multiple hard disks (SCSI drives, etc) in networks, etc and has a high fault tolerance in the event that a failure occurs in more than one drive. There are 5 levels of RAID, which are used according to the objective.

#### (1) RAID 0

In RAID 0, data is distributed to more than 1 drive, but there is no spare drive.

#### (2) RAID 1

In RAID 1, the same content is recorded on 2 hard disks with the same capacity. One of them automatically continues operating and the other is used as backup. This RAID 1 is called disk mirroring or disk duplexing.

#### (3) RAID 2

In RAID 2, bit interleave data is distributed to and recorded onto multiple drives, and the parity

and error correction information is recorded in an extra drive.

#### **(4) RAID 3**

In RAID 3, bit interleave data is distributed to and recorded onto multiple drives, but only 1 drive is used as the parity drive.

#### **(5) RAID 4**

In RAID 4, data is not recorded by bits, but by sectors, and a separate drive is used as the parity drive for error detection.

#### **(6) RAID 5**

In RAID 5, the data is distributed and recorded by sectors and the parity information is added as separate sectors in the same way as ordinary data.

Among the above mentioned RAIDs, the most used is RAID 5, which does not need a parity drive. A dedicated disk for the error correcting code (parity information) is required in RAID 2 and RAID 4.

However, even if the RAID technology is adopted, since it is only a countermeasure in the event that failure occurs in the disk itself, data backup at regular intervals is indispensable.



<http://www.vitec.org.vn>

## 4.5 Input/output architecture and devices

Since there are many mechanical operations in the input devices and the output devices, a wide gap between the operation speed of these devices and that of the processors which perform electronic operations only, is generated. If, ignoring this operation speed gap, the processor and the input or output device are connected, the operation speed of the whole computer system will become slow. And as a consequence, the computer characteristic of high speed processing becomes ineffective.

In order to solve this problem, input/output control and interruption are performed.

### 4.5.1 Input/output control method

When data is exchanged between the processor or the main storage unit and the auxiliary storage devices, input devices, output devices, etc the following control methods are provided:

- Direct control method
- DMA method
- Input/output channel control method

#### (1) Bus

The bus is a bunch of signal lines that connects units. In computers with a 16 bit word length, a bunch of 16 signal lines constitutes a bus.

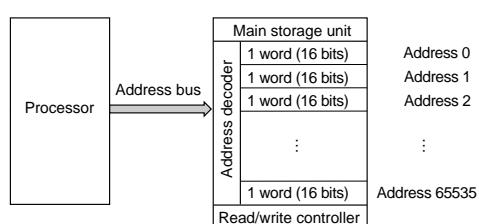
Access and information exchange are performed by a processor and a main storage unit using the following buses:

- Address bus
- Control bus
- Data bus

##### ① Address bus

The address bus connects the main storage unit and the processor. This bus is used for the specification of the main storage unit address by the processor.

Figure 4-5-1  
Address bus

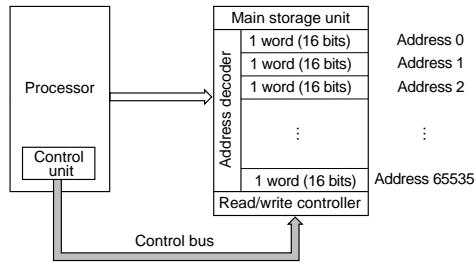


## ② Control bus

The control bus connects the main control unit and the main storage unit. This bus is used for the transmission of the instruction signal to the main storage unit from the control unit.

Figure 4-5-2

Control bus

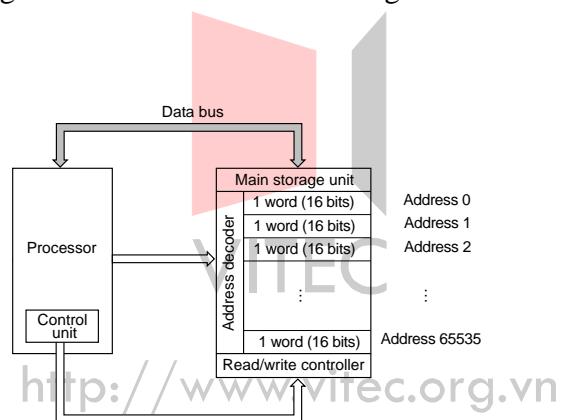


## ③ Data bus

The data bus connects the main storage unit and the processor and is used to exchange data. Only this bus is used to exchange data between the main storage unit and the processor in both directions.

Figure 4-5-3

Data bus

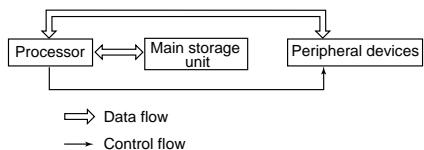


### (2) Direct control method

The direct control method is the method by which the processor directly controls the input/output operations of the peripheral devices, and the data exchange is performed through the processor. The structure of this method is simple, but it has a big drawback, which is that the processor cannot proceed to perform the next operation until the input/output operation is completed. For that reason, since processor efficiency is low, this method is not widely used.

**Figure 4-5-4**

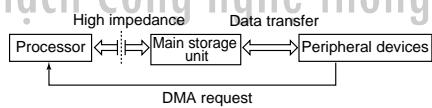
**Direct control  
method**



### **(3) DMA (Direct Memory Access) method**

When a request signal is issued from an input device or any peripheral device such as an input device or output device, the connection between the processor and the main memory unit and between the processor and the peripheral device are set to the high impedance status and data transfer is performed between the main memory unit and the peripheral device. During this input/output process, the processor performs other processes, and it is not involved in the input/output process at all. This control method is called DMA method and is widely used as the input/output control system of personal computers.

**Figure 4-5-5**  
DMA method



#### 4.5.2 Input/Output interfaces

An interface is an agreement for the connection of multiple devices and for the operation of these devices by humans. Among these, the interface related to the data input/output is called input/output interface.

The input/output control method explained above in Section 4.5.1 would not function correctly if the input/output interface were not established.

According to the transfer method, the input/output interface is divided as follows:

- Serial interface
  - Parallel interface

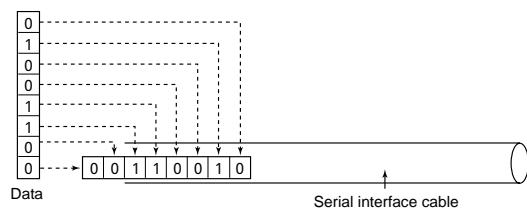
### (1) Serial interface

The interface that supports serial data exchange between the computer and the input device/output device is called the serial interface. The serial interface is the transfer method conducted by lining up data of 8 bit or 16 bit processing units in one row and transferring one bit at a time (Figure 4-5-6). The data transfer rate is lower than that of parallel interface, but it has the advantage that only one transmission channel is required. Since during a serial transfer there is no signal delay, long distance transfers can also be performed.

The following serial interfaces are widely used:

- RS-232C (Recommended Standard 232 C)
- USB (Universal Serial Bus)

**Figure 4-5-6**  
Serial transfer



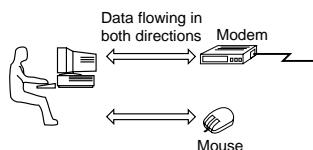
### ①RS-232C

The RS-232C is an interface that connects the computer, and the modem that converts digital signals into analog signals, or vice versa (Figure 4-5-7). This interface was standardized by EIA (Electronic Industries Association) and the physical connector format, pin number, pin role, etc are strictly established.

#### <Characteristic>

- Transfer units: the start bit that indicates the start of 1 byte data: 1 bit, the top bit that indicates the end: 1 bit, the parity bit used for error detection: 1 bit. A total of 11 bits.
- Data exchange can be performed bi directionally
- The mainstream in transfer rates are 28.8 kbps and 33.6 kbps.
- Besides the modem, it is widely used to connect image scanners , mouse and other peripheral devices and personal computers
- Data flow is bi directional

**Figure 4-5-7**  
RS-232C interface  
connection example

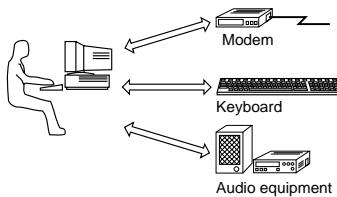


### ②USB

The USB is a new interface standard by which, besides the keyboard, modem and other peripheral devices, audio signals, image data and other input/output device data can be processed indistinctly with one connector (Figure 4-5-8). According to the USB specification, when a peripheral device is supported by the USB is connected, the personal computer automatically is configured. The connection of cables has also been simplified and its use as an input/output

interface for multimedia systems is attracting attention.

**Figure 4-5-8**  
USB interface  
connection example



③IEEE 1394

IEEE 1394 is a serial interface used to send animated image data in real time. Since real time transfer is supported, animated images can be smoothly represented. Therefore, IEEE 1394 is used as a multimedia interface supporting connections such as those between digital video cameras and personal computers.

The maximum data transfer rate is 400 Mbps and a connection of a maximum of 63 nodes can be performed.

④IrDA (Infrared Data Association)

The IrDA is an interface for wireless (infrared) data transmission. It has the advantage that, since connection cables are not used, layout modifications inside the office can be easily performed. There are several IrDA versions, and the transmission speed ranges between 2.4 kbps and 4.0 Mbps. These versions are equipped in PDA (Personal Digital Assistant) and notebook type personal computers.

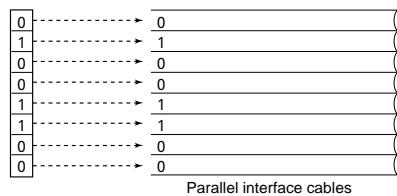
## (2) Parallel Interface

In parallel Interface, instead of transferring data in sequence, 1 bit at a time, as in serial interfaces, data is transferred in parallel using 8 or 16 cables (Figure 4-5-9). Compared to serial interface, the data transfer rate in parallel interfaces is high. However, since multiple transmission channels are required, the transmission channel maintenance cost becomes high.

The following parallel interfaces are widely used:

- Centronics interface
- SCSI (Small Computer Systems Interface)
- GPIB (General Purpose Interface Bus)

**Figure 4-5-9**  
Parallel transfer



## ① Centronics interface

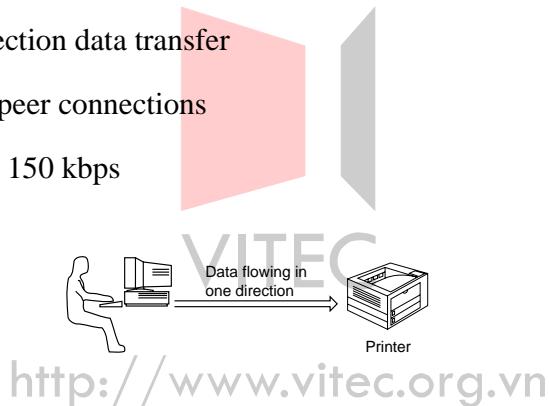
The Centronics interface is a printer interface developed by the U.S. company, Centronics Data Computer (Figure 4-5-10).

It is not an interface formally standardized by an international industrial standard organization, but, since it has been adopted by a very large number of manufacturers as the interface to connect printers and personal computers, in practice, the Centronics interface has become the "*de facto* standard printer interface".

### **<Characteristic>**

- 8 bit parallel transfer is possible
- Limited to one direction data transfer
- Limited to peer to peer connections
- The transfer rate is 150 kbps

**Figure 4-5-10**  
Centronics interface connection example



## ② SCSI

The SCSI was approved as a personal computer standard interface by ANSI (American National Standards Institute).

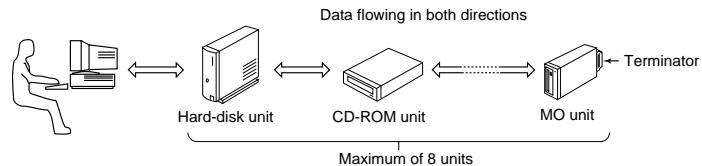
### **<Characteristic>**

- 8-bit parallel transfers can be performed bi-directionally
- The transfer rate ranges between 1.5 and 4 Mbps but the transfer rate of SCSI-2, an extensive enhancement of SCSI, is 20 Mbps.
- Up to 8 auxiliary storage devices, such as the hard disk unit and the CD-ROM unit, can be connected one after another. This is called daisy chain. An ID (a number to

distinguish the devices) is assigned to the connected devices and a resistor, called a terminator, that indicates the termination, is attached. (Figure 2-5-11)

- The pin number is 50 (or 25)
- The data flow is bi directional

Figure 4-5-11 SCSI connection example (daisy chain)



### ③GPIB

The GPIB was originally approved as a standard interface to connect microcomputers and measurement instruments, but it is currently an interface with a wide range of uses that connects the microcomputer and its peripheral devices.

This interface was standardized as IEEE-488 by the U.S. Institute of Electrical and Electronics Engineers (IEEE).

#### <Characteristic>

- 8-bit parallel transfers are possible
- It is composed of 24 signal lines
- The transmission distance is within 20m
- The data transfer rate ranges between 1 kbps and 1 Mbps
- Connection of up to 15 devices is possible

Figure 4-5-12 synthesizes the above mentioned input/output interfaces.

**Figure 4-5-12**  
Types of input/output interfaces

	Name	Transfer rate	Connected devices	Content	Industrial standard
Serial transfer	RS-232C	• 28.8kbps • 33.6kbps • 56kbps	• Modem • Printer • Mouse • Plotter, etc.	• Connector format and role • Pin number and role	EIA standard
	USB	• 12Mbps	• Keyboard • Modem • Speaker, etc.	• Multimedia support • Audio, images, etc. can be processed with one connector • Cable connection simplification	
	IEEE1394	• 400Mbps	• Digital video camera	• Multimedia support • Real-time function • Possibility to connect up to 63 nodes	IEEE standard
	IrDA	• 2.4kbps ~ 4.0Mbps	• Hard disk • Printer • Modem • Mouse	• Infrared data transmission	IrDA standard
Parallel transfer (8 bits)	Centronics interface	• 150kbps	• Printer • Plotter • Digitizer, etc.	• Wide use as printer standard interface	De facto standard developed by Centronics Data Computer Corp.
	SCSI	• 1.5 ~ 4Mbps	• Auxiliary storage devices	• Possibility to connect up to 8 devices in a daisy chain	ANSI standard
	GPIB	• 1kbps ~ 1Mbps	• Measurement instruments • Peripheral devices	• Developed by the U.S. company, Hewlett-Packard • Possibility to connect up to 15 devices	IEEE-488

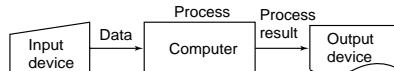
### 4.5.3 Types and characteristics of input/output devices

While the data we handle is made up of characters, numeric values (decimals), symbols, etc only binary digits can be handled by the computer. Therefore, the data and information subject to process must be converted into a format that can be processed by the computer before being transferred to the processor.

The devices equipped with this function are generally known as input devices.

Likewise, if we see the binary results processed by the computer, it will not be easy to understand their meaning. Therefore, the content processed using binary digits have to be converted into a format that can be understood by humans. The devices that perform this kind of function are generally known as output devices.

**Figure 4-5-13**  
Roles of input devices and output devices



The input device is a device exclusively used to transfer information to the computer, and the output device is a device that represents the result of the computer process in a format that can be understood by us. But there is also a device equipped with both functions. It is called the input/output device.

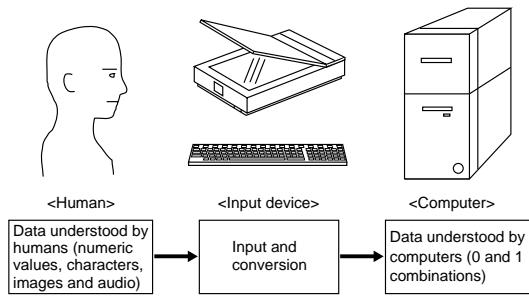
#### (1) Input devices and output devices

The device that enters data and programs into the computer is called the input device, and the device that represents/outputs computer data and programs is called the output device.

## ① Input device

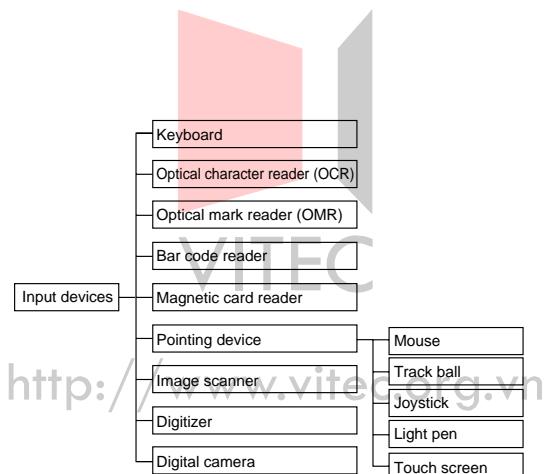
An input device is a device that converts data that can be understood by humans such as numeric values, characters, images and audio, into a data format (0 and 1 combination) that can be understood by computers, and loads it into the computer main storage unit.

**Figure 4-5-14**  
Input device's role



The early computers were limited to processing characters and numeric values, but with the progress of information technology today, computers can also process image and audio data. The different types of input devices that can process all these kinds of information are shown in Figure 4-5-15.

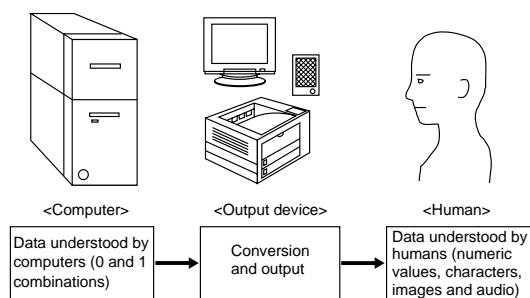
**Figure 4-5-15**  
Various input devices



## ② Output device

Output device is the general term for the devices that convert the data processed in the computer (which processes all the data using 0s and 1s to produce results which are combinations of 0s and 1s into data that can be understood by humans such as numeric values, characters, images, still images, animated images and audio , and output it.

**Figure 4-5-16**  
Output device's role



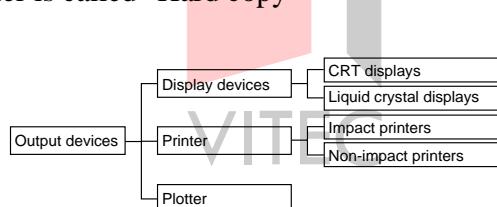
As in the input devices, there are different kinds of output modes for the output devices (Figure 4-5-17). For example, if the output is divided into "Display" and "Printing," it can be classified into the following two types:

- Display devices: The output is displayed on a television screen
- Printer: The output is printed on the surface of a piece of paper

Furthermore,

- Output into a display device is called "Soft copy"
- Output into a printer is called "Hard copy"

**Figure 4-5-17**  
Various output devices



<http://www.vitec.org.vn>

## (2) Keyboard

The keyboard, which is the input device we find most familiar, inputs the code corresponding to the key of the character or symbol we press in the processing unit. A keyboard layout is specified by a JIS (Japanese Industrial Standard). However, in order to improve the efficiency of Japanese input, some word processor manufacturers have developed unique keyboard layouts of their own.

Figure 4-5-18

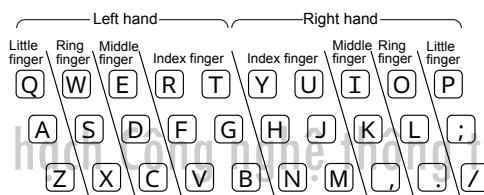
Keyboard



Figure 4-5-19 shows the correct finger position to be assumed to press the keys. Once, one gets used to it, it becomes easy to press the correct keys without looking at the keyboard. The typing method is called touch typing.

Figure 4-5-19

Touch typing



### (3) Optical character reader (OCR)

The optical character reader is a device that, based on the intensity of the reflected light, reads out characters and symbols and inputs them. (Figure 4-5-20).

Figure 4-5-20

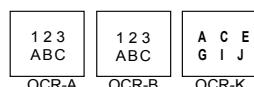
Optical character reader (OCR)



At the beginning, the optical character reader could only read easily identifiable special printed characters and JIS OCR fonts (Figure 4-5-21). However, at present, the character pattern recognition has improved and even handwritten characters can be recognized.

Figure 4-5-21

OCR fonts



#### (4) Optical mark reader (OMR)

The optical mark reader is the device that reads and inputs data according to the marks made on marksheets.

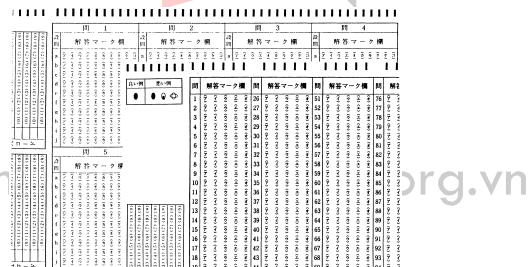
Alphanumeric characters are printed on marksheets; the part to be input is marked with a pencil, etc. The difference with the optical character reader is that this device does not directly recognize the patterns of characters or numeric values, but reads them out based on the marked position instead. The information reading principle is the same as that of the optical character reader. Based on the reflected light, this device judges whether or not marks exist and inputs the character or numeric value corresponding to the marked position. Therefore, there are cases where reading errors occur or reading cannot be performed -- when marksheets are dirty or folded.

**Figure 4-5-22**  
Optical mark reader (OMR)



Since the mark reading position is set through the program, the pattern of the marksheets can be freely designed. In practice, marksheets are used in different fields, and are also used as answer sheets in the Information Processing Engineer Examination.

**Figure 4-5-23**  
Marksheet example



#### (5) Bar code reader

The bar code reader is the device that reads and inputs the bar code attached to diverse products. The following types of bar code readers exist:

- Pen type
- Touch type
- Laser type

When the pen type device is used, the bar code has to be traced with LED (light emitting diode). With the touch type device, the LED only has to be focused on the bar code (Figure 4-5-24).

Likewise, since it is not necessarily to directly touch the bar code with the laser type device, it is widely used in convenience stores and super markets.

Figure 4-5-24  
Bar code reader



Generally, in account processing using bar codes, not only product identification and accounting are performed, but also, based on the information input, stock control and order control are performed.

## (6) Magnetic card reader

The magnetic card reader is a device that reads and inputs the information needed from a magnetic card.

There are different types of readers depending on the magnetic card to be read. Automatic train ticket gate machines are also magnetic card readers.

The magnetic card is a paper or plastic card which has a magnetic strip on the surface to store information such as numeric values and characters.

At present, as it is said, we live in a "card society" and magnetic card use has widely expanded as a paying method that replaces cash in our daily life. Among the most familiar cards, we have the following:

- Phone cards
- Cash cards
- Credit cards
- Tickets for automatic ticket gates

VITEC  
<http://www.vitec.org.vn>

Magnetic cards have become indispensable in our daily life. There is also the IC card, which has increased the storage capacity of magnetic cards and incorporated information processing functions. It is more expensive than regular magnetic cards, but it is superior in security and functional aspects.

## (7) Pointing device

"Pointing device" is the generic term for the devices that input positional information on the screen of a display device.

With the expansion of computer use, different kinds of pointing devices have appeared. Among the main pointing devices, the following can be mentioned.

- Mouse
- Track ball
- Joystick
- Light pen
- Touch screen

Without the troublesome operations on keyboards, etc, using any of these devices, anybody can easily input data while watching the screen of the display unit.

### ①Mouse

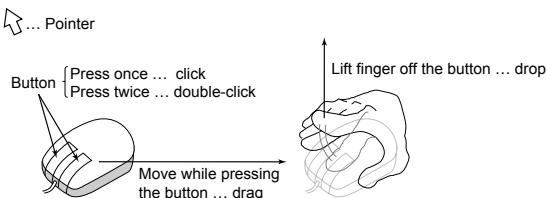
Along with the keyboard, the mouse is the most used input device. It was named mouse due to the similarity of its external appearance to a mouse.

**Figure 4-5-25**  
Mouse



The mouse has a mechanism by which, when it is moved, the ball on the underside rolls, and the screen pointer moves according to the rolled distance and direction. When the pointer has been moved to the aimed position, by pressing the button of the mouse, the positional information is entered. This operation is called clicking. Likewise, as it is shown in Figure 4-5-26, besides clicking, there are other button operations such as double clicking, which mean processing the button two times and dragging, which is the specification of an area by moving the mouse while holding the button down, etc.

**Figure 4-5-26**  
Mouse operations  
and pointer



The biggest characteristic of the mouse is that, unlike the keyboard, it does not input data directly to the computer; instead, by pointing at the icons and windows indicating the operation locations which appear on the screen, it indicates and inputs operations to the computer. In other words,

the mouse supports the GUI (graphical user interface) environment.

#### ② Track ball

The principle is the same as that of the mouse, but, since the track ball is moved directly with the fingers, it does not require the moving space needed by the mouse. For that reason, it is often equipped in lap top and notebook personal computers.

Figure 4-5-27

Track ball



#### ③ Joystick

With the joystick, the stick is moved back and forth as well as to the right and left, and the pointer moves according to the direction and the angle in which the stick is moved. It can perform the same operations as the mouse, but, since indications have to be performed with buttons, etc. besides the stick, it is not so easy to handle as the mouse, which can be manipulated with one hand. For that reason, joysticks are widely used for game software manipulation.

Figure 4-5-28

Joystick



#### ④ Light pen

The light pen is a device that inputs the coordinate information by pointing and tracing on the screen of the display device directly.

Since the optical sensor at the point of the light pen detects the position of the information on the screen and inputs it, the responsiveness is high.

As the uses of the light pen, data entry on palm top personal computers entry, and the entry of handwritten characters into a word process can be mentioned.

**Figure 4-5-29**

Light pen



⑤ Touch screen

The touch screen, which is also called a touch panel, takes advantage of the static electricity that passes through the human body. By directly touching with the finger the screen of the display device, the positional information is entered. In this mechanism, a transparent panel is attached to the screen surface and the sensor on the panel senses changes in the voltage and detect the touched position.

Due to the dimensions of the area touched with the finger, detailed manipulations and instructions can not be performed, but since it can be easily manipulated by anybody, it is widely used in the automatic teller machines (ATMs) of banks, automatic ticket vending machines at train stations, reception/information at hospitals, etc.

**Figure 2-5-30**

Touch screen



(8) Image scanner

The image scanner is a device that reads and inputs figure and picture data from a sheet of paper with the same principle as a fax. The mechanism consists of decomposing a figure or image into a dot image, composed of small dots, and focusing light on it of order to load the intensity of the light reflected as electronic codes into the computer.

The device which moves the reading mechanism over a fixed piece of paper is usually called the image sensor.

**Figure 4-5-31**  
Image scanner



(9) Digitizer

The digitizer is a device that, by tracing a plane panel figure with a pen or cursor, detects the coordinate's position and, based on this consecutive coordinate information, inputs the figure. This device is used by the CAD (Computer Aided Design) application, in which the input of figures of high precision is required. Small sized devices are sometimes called tablets for making a distinction.

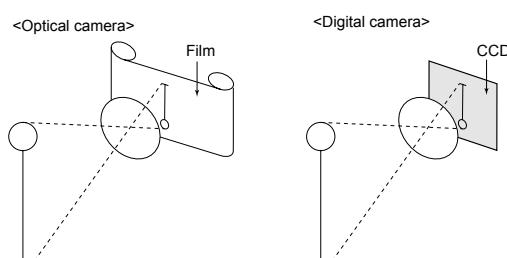
**Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo**  
**Figure 4-5-32**  
Digitizer



(10) Digital camera

The digital camera is a camera that can input the picture taken to the computer as data. While optical cameras record images through the chemical change of the sensitized material of the film surface, digital cameras, using the image sensor of a semiconductor element called CCD (Charge Coupled Device), convert optical pictures into digital data and record this data as image files. A semiconductor memory called flash memory is widely used as the storage medium.

**Figure 4-5-33**  
Optical camera and  
digital camera  
mechanism



## (11) Display device

In ordinary computer systems, operations are conducted verifying on the spot the input data and process results on the screen. The display device is one of the devices which are indispensable for human use of computers. Displays are roughly divided into the following two types:

- Character displays: Capable of displaying characters only
- Graphics display: Capable of displaying characters and graphics

Likewise, according to the colors that can be displayed, displays are divided as follows:

- Black and white displays
- Color displays

At present, color graphics displays are the standard. Furthermore, according to the structure of the display screen, display devices can be classified as follows:

- CRT displays
- Liquid crystal displays

Here CRT displays and liquid crystal displays will be explained.

### ①CRT display

The display device that has the same structure as the television, and uses the cathode ray tube , is called CRT (cathode ray tube) display.

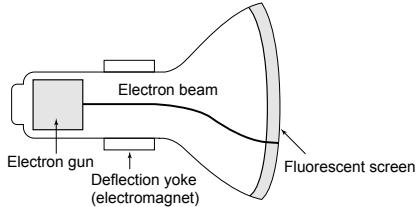
Figure 4-5-34  
CRT display



#### a. Mechanism

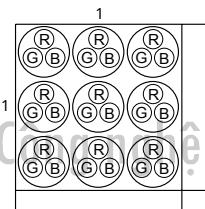
As it is shown in Figure 4-5-35, in the CRT display, when struck by the electron beam, the fluorescent screen emits light, which is displayed on the screen.

**Figure 4-5-35**  
CRT structure



Color display is generally used in CRT displays. Color images are represented by striking the dots that contain the "three primary colors of light," R,G, and B (Red, green, blue) with the electron beam.

**Figure 4-5-36**  
Color screen diagram  
magnification



Besides the above mentioned, the screen display is equipped with an important function called screen saver. In the CRT display, when the same screen is displayed for a long period of time, the image of that screen is burnt into the fluorescent tube. In order to prevent this, an animated image is displayed on the screen. This software is called screen saver.

#### b. Resolution

The screen size is represented by the diagonal length of the screen. According to this, there are screens of 15 inches, 17 inches, 21 inches, etc.

The screen resolution is represented by the value of the number of dots which can be represented in 1 screen (width x height), and resolution of 640 x 480, 800 x 600, 1024 x 768, 1280 x 1024, etc. Today, as a result of the expansion of multimedia, 1280 x 1024 resolutions has become a *de facto* standard due to its capacity to process high image quality. Likewise, the CRT display called multi scan monitor, in which the resolution can be switched according to necessity, is expanding.

#### ② Liquid crystal display

The liquid crystal display is a display device widely used in the display screen of calculators, etc. (Figure 4-5-37). The liquid crystal material used has the property of aligning in one direction when voltage is applied, changing from non transparent to transparent. The liquid crystal display takes advantage of this property, and by controlling whether or not light passes through it with the voltage applied, the appropriate display is produced.

Currently, the majority of liquid crystal displays are also color displays that use R, G, B color filters.

Unlike CRT displays that require a specific depth, liquid crystal displays are thin, and moreover,

have low power requirements. Due to these reasons, they are widely used in lap top and notebook personal computers.

**Figure 4-5-37**  
Liquid crystal display



The following two types of liquid crystal display exist:

a. Passive matrix type

The system in which multiple liquid crystal pixels are controlled by one semiconductor is called the passive matrix type. This system is adopted by the STN (super twisted nematic) liquid crystal display.

Currently, the DSTN (Dual scan STN) liquid crystal display, which divides the liquid crystal panel into upper and lower sides enabling double scanning, is expanding.

b. Active matrix type

The system in which one liquid crystal pixel is controlled by one semiconductor is called the active matrix type. This system is adopted by the TFT (Thin Film Transistor) liquid crystal display. The TFT liquid crystal display uses a transistor as a switch to apply voltage. The screen contrast, response speed, viewing angle, etc are dramatically superior to the STN liquid crystal display. However, due to the complex structure, the production cost is high.

## (12) Printer

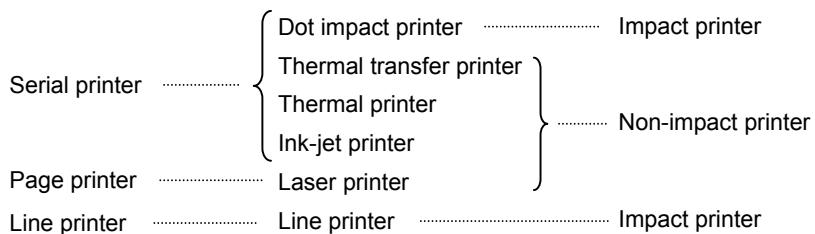
The printer is the oldest computer output device and, today, it is, such more widely used. There are many types of printers. According to the printing methods, printers are classified as follows:

- Impact printers: Print by mechanically hitting preformed characters against an ink ribbon.
- Non impact printers: Print using heat, ink, laser, etc.

Likewise, according to the printing unit, printers are classified as follows:

- Serial printers: Print 1 character at a time, as typewriters
- Line printers: Print 1 line at a time
- Page printers: Print 1 page at a time

Here, among the different types of printers, the printers indicated below will be explained:



### ① Dot impact printer

The dot impact printer is the printer that prints by striking a head against an ink ribbon, which, in turn, hits the paper. Alphanumeric characters are sets of dots, and since printing is performed by striking a head containing multiple small pins, dot impact printers are noisy. This is a drawback; however, they are convenient for performing multiple printing at once using carbon copies. Compared to thermal transfer prints, etc., the quality of the printed characters is not so good. The density of the dots composing a character (width x height) determines the print resolution. The number of dots forming one character varies from 9 x 7, 16 x 16, 24 x 24. The higher these values, the better the resolution of the printed characters. Likewise, the characters that can be printed vary. Only alphanumeric characters can be printed with a 9 x 7 resolution, while kanji can also be printed with 16 x 16 and 24 x 24 resolutions.

**Figure 4-5-38**  
Dot impact printer



### ② Thermal transfer printer

The thermal transfer printer is a printer in which the print head heats and melts the ink of the ink ribbon to print the dot composed character on normal paper. Since this printer uses a non impact method, the noise level is lower than that of dot impact printers.

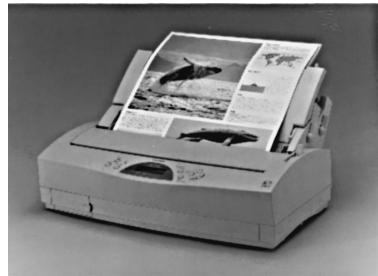
On the other hand, the thermal printer uses thermal paper as printer paper. Since thermal paper fades with time, it is not suitable of long term conservation. Furthermore, the fact that the running cost becomes high due to the high price of thermal paper is a drawback.

### ③ Ink jet printer

The ink jet printer is a printer in which, according to the form of the dot composed character, tiny ink nozzles in the print head squirt ink onto the paper. Color ink of 3 colors "Cyan: C, Magenta:

M and Yellow: Y) or 4 colors" "3 colors + Black: K" are used in color printing. Today, the use of color ink jet printers as personal computer printers has expanded.

**Figure 4-5-39**  
Ink-jet printer



④Laser printers

The laser printer is a page printer that, using toner (powder ink), creates the printing image of one page on the photoreceptive drum and transfers it to the paper through the application of laser beams. The printing principle is the same as that of copy machines, and the character size, space between the lines, etc can be selected freely. Figures, images, etc can also be printed and print quality as well as printing speed are high. For that reason, it is the mainstream printer for business use.

**Figure 4-5-40**  
Laser printer



## 4.6 Computer types

### 4.6.1 Computer types

Computers used in a great variety of field will be explained below.

#### (1) Personal computer

As the name implies, personal computers are computers that were developed for personal use, commonly called PCs for short. Based on their external appearance, different types of personal computers have multiplied. These personal computers can be classified as follows:

- Desk top type, which can be placed on a desk (Figure 4-6-1)
- Lap-top type, which can be placed on one's lap
- Notebook type, the size of A4 or B5 paper, thin and light (Figure 4-6-2)  
Likewise, expansion of the palm top type (Figure 4-6-3) ultra small-sized personal computers which can be held in one's palm is starting.  
On the other hand, according to the place where they are set up, and the main use purpose, computers can be classified as follows:
  - Home: Used as word processors and to play games at home and hobby computers.
  - Enterprise: Word processor, spreadsheet software and database software are used for business.  
Used in software development
  - School and enterprise: used in CAI (Computer Aided Instruction) application for education

Likewise, considering the sue mode, up to now, stand-alone system was used in most computers, but recently, the network system in which personal computers are connected by communication lines is becoming the mainstream.

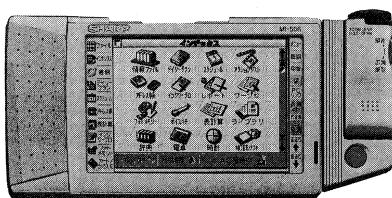
Figure 4-6-1  
Desk-top type



Figure 4-6-2  
Notebook type



Figure 4-6-3  
Palm-top type



## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo (2) Workstation

Due to the expansion of use of personal computers, enterprises are adopting the system of one computer per user. However, personal computers lack the capacity to perform technological calculation processing, software development, etc. In order to solve this problem, computers called engineering workstations

(EWS) were created. Compared to personal computers, workstations are capable of performing high quality image processing, etc with high speed (Figure 4-6-4).

The main applications of these workstations are listed below:

- Research and development fields
- High speed processing of complex scientific and engineering calculations
- Product design/manufacturing fields  
Used in CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), etc application
- Software development field  
Use of CASE tools (Computer Aided Software Engineering tool), etc
- Communication network field  
Used as client machines or server machines in distributed processing systems

**Figure 4-6-4**

Workstation



(3) General purpose computer

The general purpose computer is a computer which, literally, can be used for multiple purposes, capable of performing both office work as well as scientific and engineering calculations. Since it is the main frame of a great number of computers used in the enterprise, it is also called mainframe. (Figure 4-6-5)

The computers that conduct enterprise core business system processing, an automatic ticketing processing, bank services, etc are all general purpose computer.

Since most general purpose computers are large sized, due to the high heat generation, it is necessary to install them in an air-conditioned room called computer room.

**Figure 4-6-5**  
General-purpose  
computers



(4) Super computer

There is no precise definition of supercomputers. Commonly, computers capable of performing enormous and complex calculations at extremely high speed are called supercomputers. In other words, it can be said that supercomputers are computers whose design attaches importance to high speed calculation.

Supercomputers are computers that compile computer high speed technology using forefront semiconductor element technology as well as vector processes that perform floating point operations and vector operations, etc. However, their use purposes are limited. Among the main purposes, the following can be mentioned.

- Weather forecast
- Simulation of nuclear power generation
- Orbit calculation of artificial satellites

As a manufactured product, the U.S. Cray Inc.'s Cray supercomputer is famous. In Japan, NEC's SX and Fujitsu's FACOM VP are produced.

**Figure 4-6-6**  
Supercomputer



Trung tâm Sát hạch công nghệ thông tin và Hỗ trợ đào tạo

#### (5) Microcomputer

Microcomputers are small sized computers into which a processor is built. The computers that are imbedded into machines, especially household appliances such as washing machines, air conditioners and AV appliances, in order to control the machine operation are called microcomputers. These microcomputers are electronic parts with bare integrated circuits. According to the purpose, information on the temperature and number of revolutions can be entered using a sensor. Since their function is to repeat the same operation, control data is recorded in ROMs (Read Only Memory). Likewise, since the output devices are motors or electric switches, they are also called actuators.

**Figure 4-6-7**  
Microcomputer

<http://www.vitec.org.vn>



#### (6) Process control computer

Process control computer are computes that control the different types of machines in steel mills, automobile plants, petroleum refineries, etc. Chemical plants, etc. are entirely automated with process control computers. When the supervising computer detects an abnormality, it

immediately controls each machine and adjusts the production process.

In addition to this, centralized control and automation have been achieved through the use of process control computers in power system control, general building security systems, highway traffic control, etc.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

## **4.7 Wireless standards and Mobile computing**

---

Wireless LAN is an alternative to the traditional cable based LANs. It sends information among the devices but without the constraint of physical cabling. This means wireless node can be moved easily.

Wireless is also a good choice if the physical structure of the building makes it difficult to run wire within it.

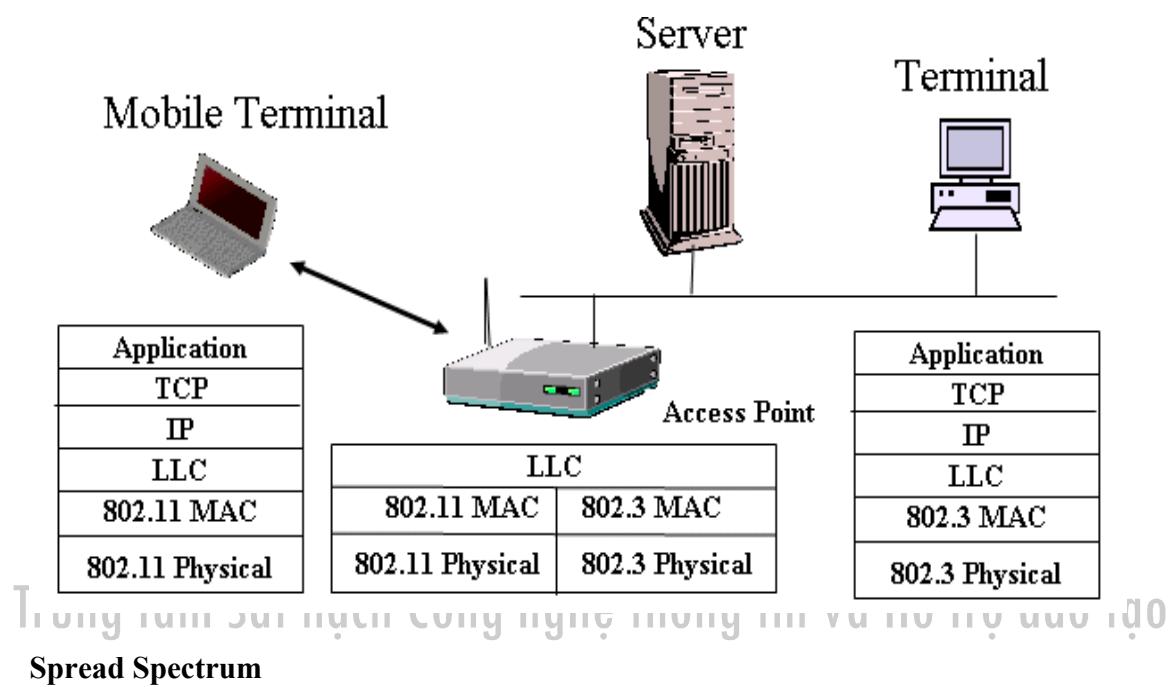
Portable computers benefit from the use of wireless LANs allowing them to be truly portable without being tied by a physical wire to a network. Wireless networks can be combined with cabled LANs.

The common standards used are Bluetooth and IEEE 802.11 wireless Ethernet. Both utilize the unregulated 2.4 GHz radio band

### **4.7.1 Transmission methods**

The following transmission methods are used in wireless LANs  
<http://www.vitec.org.vn>

- i) spread spectrum
- ii) narrowband microwave, and
- iii) infrared.



### Spread Spectrum

This is the most commonly used transmission method for wireless LAN. The signal is spread over a range of frequencies comprising Industry, Scientific and Medical (ISM) bands. The frequency range is 902 MHz to 928 MHz and 2.4 GHz to 2.484 GHz. A frequency hopping spread spectrum is used.

The signal is broadcasts over a seemingly random series of frequencies. The receiver receives the message in synchronization with the sender. This means the message can only be received if the frequencies are known. The frequency hopping spread spectrum is limited to the 2.4-GHz band based on standards set by the IEEE 802.11 committee. Another type of spread spectrum communication is called direct sequence spread spectrum. Redundant data bits called "chips" are added to the sequence to spread their transmission.

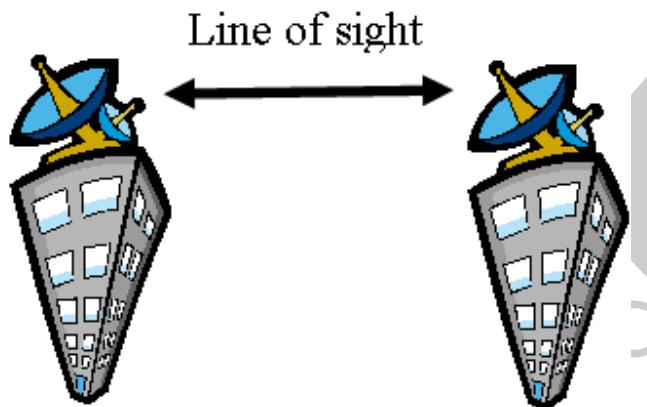
A correlator is used by the receiver to remove the chips and reduce the signal to its original length.

The practical raw data throughput is up to 2 Mbps in the 902-MHz band and 8Mbps in the 2.4-GHz band. Frequency hopping radios currently use less power than direct sequence radios and generally cost less.

### **Narrowband Microwave**

Microwave technology is used to interconnect LANs between buildings. Microwave dishes are used at both ends of the link. A line of sight is required for the signal

Trung tâm Cát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

### **Infrared**

Infrared signals are used to transmit the data. The technology is similar to those found in remote controls for televisions and VCRs. A point-to-point configuration is possible. The main advantage is a high bandwidth is supported. However, they can be easily obstructed, since light cannot pass through solid objects.

The following compares the different kinds of Wireless LAN Transmission Techniques.

	<b>Spread Spectrum</b>	<b>Narrowband Microwave</b>	<b>Infrared</b>
Frequency	902MHz to 928 MHz ; 2.4 GHz to 2.4385 GHz	18.825 GHz to 19.205 GHz	$3 \times 10^{14}$ Hz
Transmission power	Less than 1 W	25 mW	Not applicable
Use between buildings	Possible with antenna	No	Possible
Line of sight	No	No	Yes
Maximum coverage	30 to 240 m up to 15 km	12 to 39 m up to 1.5 km	9 to 24 m

#### 4.7.2 Bluetooth

L. M. Ericsson invented Bluetooth in 1994. This technology is used for short range

indoor wireless networks. It utilizes a Frequency Hopping - Code Division Multiple

Access/Time Division Duplexing method.

Multiple devices can form overlapping networks called piconets in an ad-hoc fashion.

Bluetooth is a radio link and therefore does not require line-of-sight (LOS)

connections. An 'Ad-hoc' network is a network that can be set up anywhere at anytime.

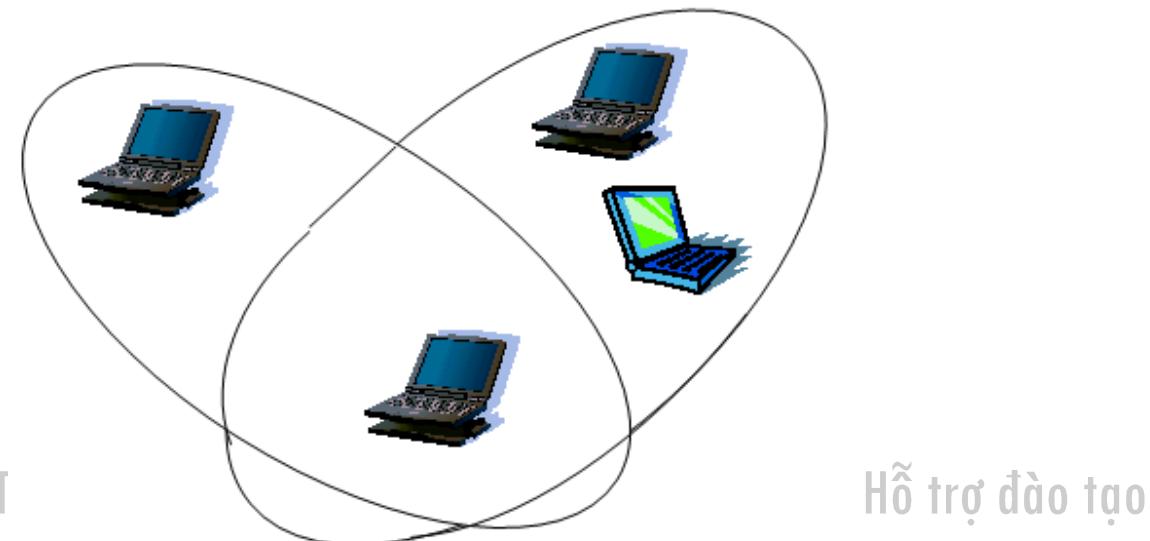
Piconets can overlap to form a scatternet.

<http://www.vitec.org.vn>

Piconet



## Scatternet



Hỗ trợ đào tạo

### 4.7.3 IEEE 802.11

IEEE 802.11 is intended for use inside buildings such as offices, hospitals, banks, shops.

It supports an asynchronous and time-bounded delivery service. It can be used within a extended areas via a system such as Ethernet.

	IEEE 802.11 Direct Sequence Spread Spectrum (DSSS)	IEEE 802.11a	IEEE 802.11b
Application	Wireless Ethernet (LAN)	Wireless ATM	Wireless Ethernet (LAN)
Frequency Range	2.4 GHz	5 GHz	2.4 GHz
Data Rate	1-2Mbps	20-25 Mbps	5.5 Mbps, 11 Mbps

#### 4.7.4 Network Architecture

A wireless station, usually the PC or a Laptop with a wireless network interface card (NIC), and an Access Point (AP) is used in the 802.11 network architecture. The Access Point acts like a bridge between the wireless stations and wired networks.

The two operation modes in IEEE 802.11b are

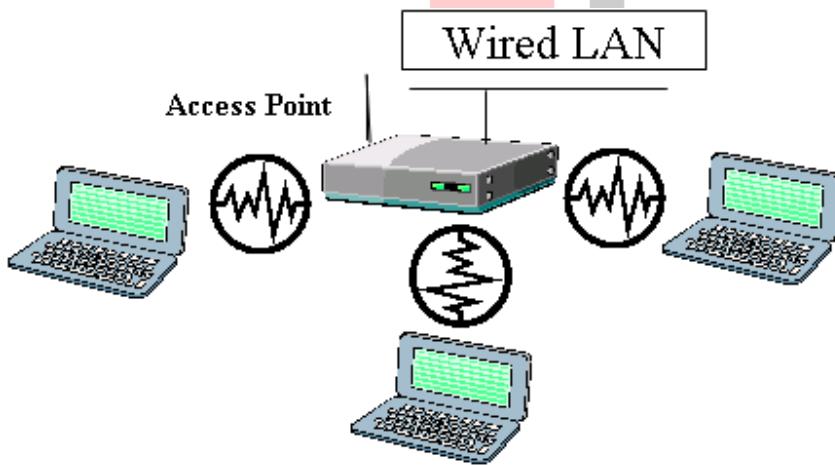
- 1) Infrastructure Mode
- 2) Ad Hoc Mode.

##### **Infrastructure Mode**

A minimum of one access point must be present.

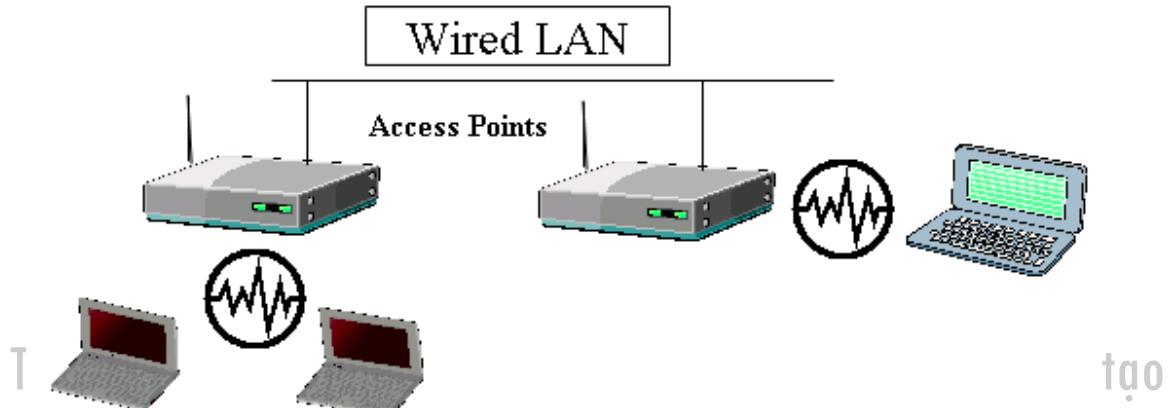
##### **Basic Service Set (BSS)**

The wireless stations communicate with each other through the access point. They do not directly communicate with each other. The frames are relayed by the Access Point.



## Extended Service Set (ESS)

The access points conveys the messages with each other to carry the message from one BSS to another.

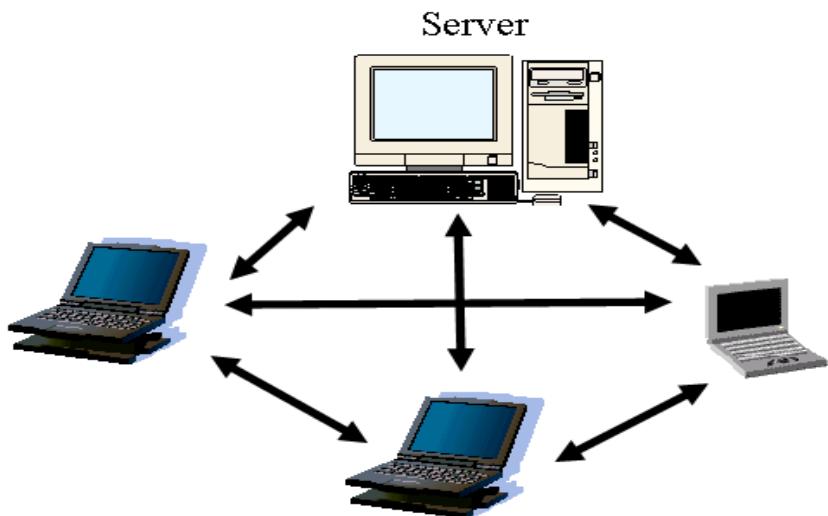


## Ad Hoc Mode

### Peer to Peer or Independent Basic Service Set (IBSS)

The wireless stations communicate directly with each other. They are no access points.

This means the communication is limited by the range of the stations



#### **4.7.5 Security**

Security in 802.11 is provided by a process called WEP (Wired Equivalent Privacy).

This prevents an intruder from:

- Reaching the resources of the network by using the same wireless equipment.
- Listening to the network

Prevent (warn) the access to the resources of the network

This is obtained by using a mechanism of authentication where a station is obliged to prove its knowledge of a key, what is similar to the safety(security) on cable networks, in the sense(direction) where the intruder should enter places (by using a physical key) to connect his post to the cable network.

#### **Eavesdropping**

Eavesdropping is blocked by the use of the algorithm WEP which is a generator of random numbers initialized by a shared secret key.

The generator of random numbers re-releases a sequence of keys of random bits, equal in length to the widest possible packet, which, combined with entering or outgoing packets, produces the transmitted packet.

The algorithm WEP is a simple algorithm based on RSA's algorithm RC4, which has the following properties:

- 1) Reasonably hard: the attack by brute force of this algorithm is difficult by the fact that every frame is sent with a vector of initialization which re-launches the generator of random numbers.
- 2) Auto synchronization: the algorithm re-synchronizes for every message.

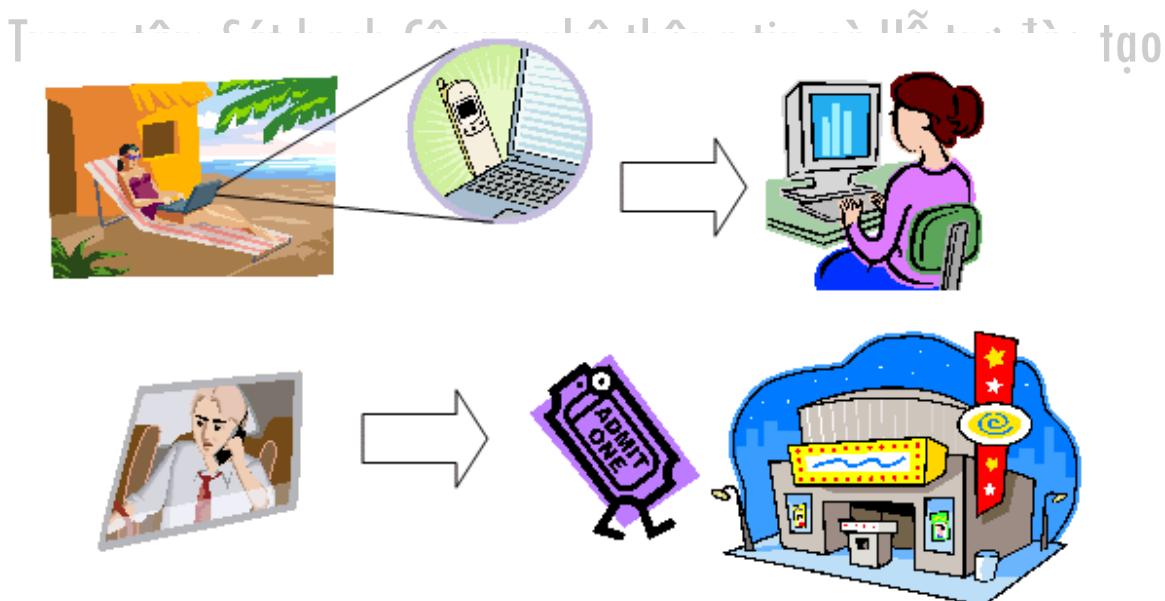
This is necessary to work in non connected mode, where packets can be lost, as in any local area network.

### Data rate

Bluetooth has a maximum capacity of 1 Mbps. This is compared to the, now standard, 802.11b which runs up to 11 Mbps.

### 4.7.6 Mobile computing

#### Characteristics of mobile computing



#### 1) User mobility

Communication can take place “anytime, anywhere, with anyone.”

#### 2) Device portability

Devices can be connected anytime, anywhere to the network.

Wireless	mobile	Examples
x	x	Stationary computer
x	o	Notebook in a hotel
o	x	Wireless LANs in historic buildings
o	o	Personal Digital Assistant (PDA)

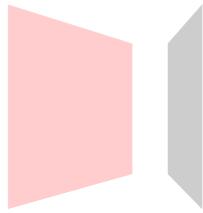
This creates the need for mobile communication to integrate wireless networks and existing fixed networks to wide area networks.

#### **4.7.7 Mobile computing infrastructure**

The following mobile computing devices are used

1) Cellular phone

Cell phones are increasing, become smaller and have more functionality. They come with a display and include SMS, internet and fax capabilities. Some recent models even support a camera function.



2) Attachable keyboard

Plug in keyboards can be used to allow for easier interaction.

3) PDA (personal digital assistants)

PDAs come with access to the internet.



#### **4.7.8 Type of software**

The main software required to run these devices are

##### **(1) Microweb browser**

These are used to view online content created by using the WML (Wireless Markup Language). These are WAP (Wireless Application Protocol) based devices. A different standard called i-mode is adopted in Japan using the CHTML (Compact HTML). The new WAP 2.0 standard has adopted XHTML as the standard. XHTML merges HTML with XML leading to a standard that allows content to be created for both the mobile and desktop computing.

##### **XHTML**

XHTML is based on XML standards. This means a document must follow "well formed" rules, that is, XML syntax. The main rules are

- 1) XML is case sensitive. In XHTML this means every HTML tag must be written in lower case. Example use <table> not <TABLE>.
- 2) When generating HTML dynamically make sure to use lower case.
- 3) Non-empty tags must be properly nested. This means tags do not cross over each other.

In the invalid example below notice that the form and table tags are improperly nested, that is, they cross over one another.

```
<form action="/scripts/my.asp" >
<table border>
<tr><td>Contents
</form>
</td></tr>
</table>
```

should be corrected to

```
<form action="/scripts/my.asp" >  
<table border>  
<tr><td>Contents</td></tr>  
</table>  
</form>
```

4) Attribute values must be within quotes. So `<form action=`my.asp`>` is not legal but

`<form action="`my.asp`">` is legal.

5) All tags must be closed. For tags which don't normally have a closing element, end the tag within itself.

For example, `<br>` by itself is not legal. Rather, use `<br/>`. These tags may also end

like `</br>`, but `<br/>` syntax seems to work better with current browsers.

6) No attribute may appear more than once in the same tag.

7) The first tag in a XHTML document must be `<!DOCTYPE>`. This tag informs the reader which definition to use in describing the XHTML document. XHTML uses DTD modules to translate tags. In selecting among the three DTDs follow these rules.

When writing pure XHTML use the [strict DTD](#):

<http://www.vitec.org.vn>  
`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-strict.dtd">`

When writing for the most HTML compatibility use the [transitional DTD](#) :

`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd">`

When using frames use the [frameset DTD](#):

`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "DTD/xhtml1-frameset.dtd">`

8) The second tag in a XHTML document must be <html> and the xmlns attribute is mandatory.

9) The <title> tag is mandatory in a XHTML document.

10) Form tags must have an action attribute. For example, <form action="my.asp"></form>

11) Style tags such as <font> and <center> have been removed. Use style sheets for formatting.

12) Every <img> tag must have an alt attribute.

13) Every <style> tag must have a type attribute.

14) Scripting elements pose a problem for XHTML compatibility. The XML parser will parse the script as a XML document unless you enclose your script in a CDATA block. Therefore, a JavaScript element would now look like:

```
<script type="text/javascript">
<![CDATA[ alert("hello"); ]]>
</script>
```

VITEC

## (2) Mobile Operating System

These are operating systems like Windows CE, Palm OS etc. There are specialized OS that resides on the mobile device.

## (3) Wireless Middleware

These allow multiple wireless networks to be linked to the application servers.

#### 4.7.9 Mobile based protocols

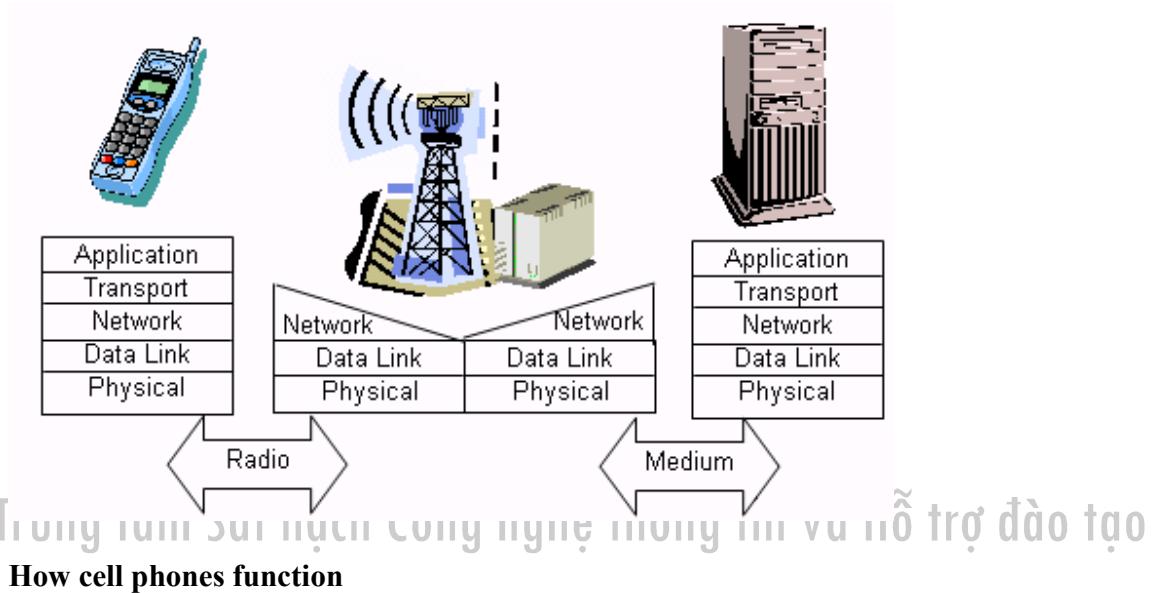
1G wireless	AMPS (Advanced Mobile Phone Service)	Analog voice service No data service
2G wireless	CDMA (Code Division Multiple Access) TDMA (Time Division Multiple Access) GSM (Global System for Mobile Communications) PDC (Personal digital cellular)	Digital voice service 9.6K to 14.4K bit/sec. CDMA, TDMA and PDC offer one-way data transmissions only Enhanced calling features like caller ID No always-on data connection
3G wireless	WCDMA (Wideband Code Division Multiple Access) CDMA2000 Based on the Interim Standard95 CDMA standard TDSCDMA (Time division synchronous code division multiple access)	Superior voice quality Up to 2M bit/sec. always-on data Broadband data services like video and multimedia Enhanced roaming

**3G**

VITEC

Third generation (3G) services combine high speed mobile access with Internet Protocol (IP)-based services Japan's NTT DoCoMo launched the world's first official 3G system based on WCDMA on 1st October 2001.

## Layers used in mobile communication

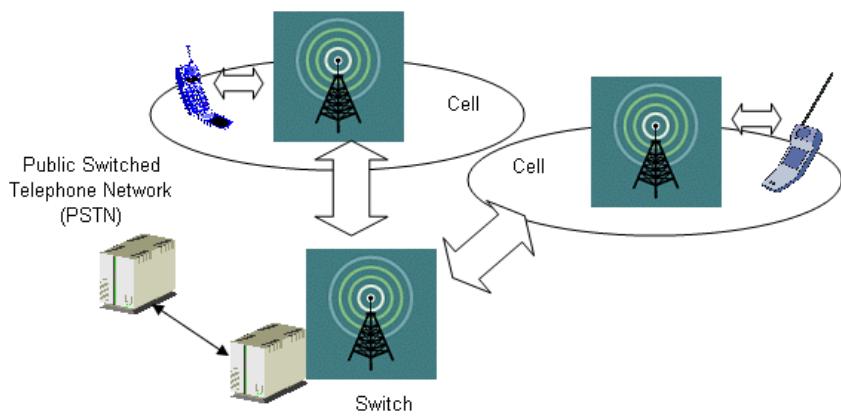


Trung tâm CNTT Công nghệ thông tin và hỗ trợ đào tạo

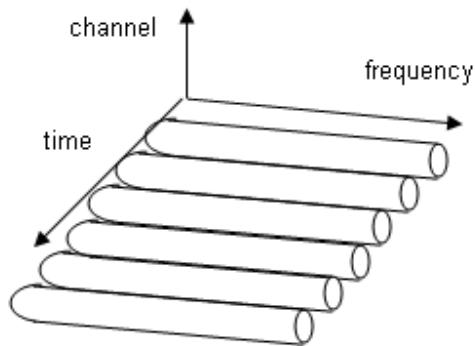
### How cell phones function

The cell phones utilize high frequency radio waves in the range of 806 to 890 MHz or 1,850 to 1,990 MHz to communicate with cell towers. A request is made to the tower to connect to a counterpart caller.

If sufficient resources are available at the cell tower, the switch patches the call through a channel to the PSTN. This channel is held until the call is terminated.



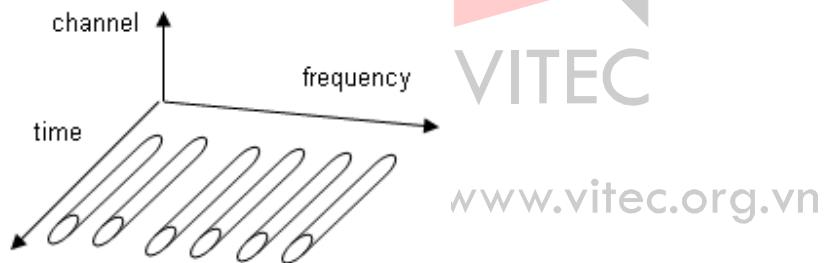
## Time Division Multiplex



This acts like a time sharing mechanism where a channel gets the whole spectrum for a certain amount of time. There is only one carrier in the medium at any time. The throughput is high even for many users however precise synchronization is necessary.

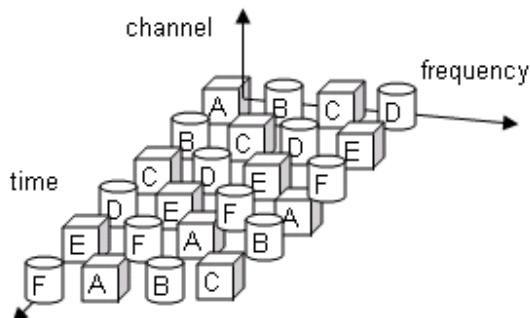
This is used in the Ethernet.

## Frequency Division Multiplex



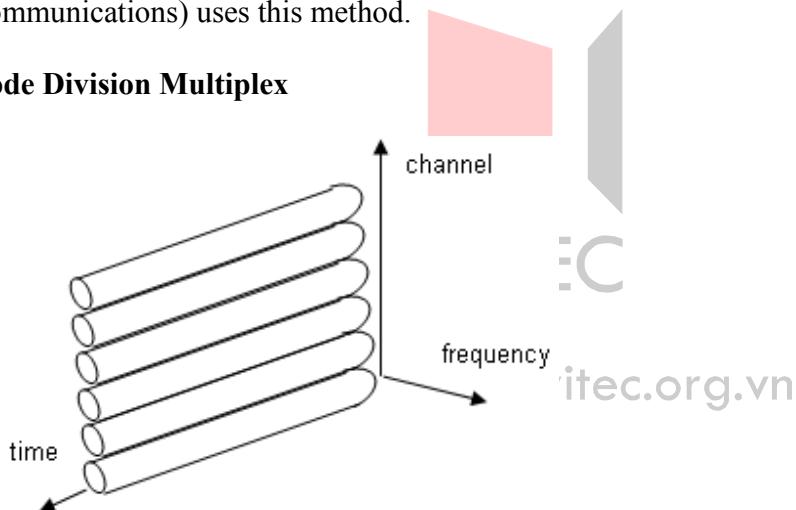
The whole spectrum is divided into smaller frequency bands. A channel gets a certain band of the spectrum for the whole time. This means no dynamic coordination is necessary. It also works for analog signals. It becomes a waste of bandwidth if traffic is not distributed evenly. It is inflexible. This is used in radio broadcast.

## Time and Frequency Division Multiplex



This combines both methods. A channel is allocated a certain frequency band for some time. There is protection against frequency selective interference and tapping. It is adaptive however precise coordination required. GSM (Global System for Mobile Communications) uses this method.

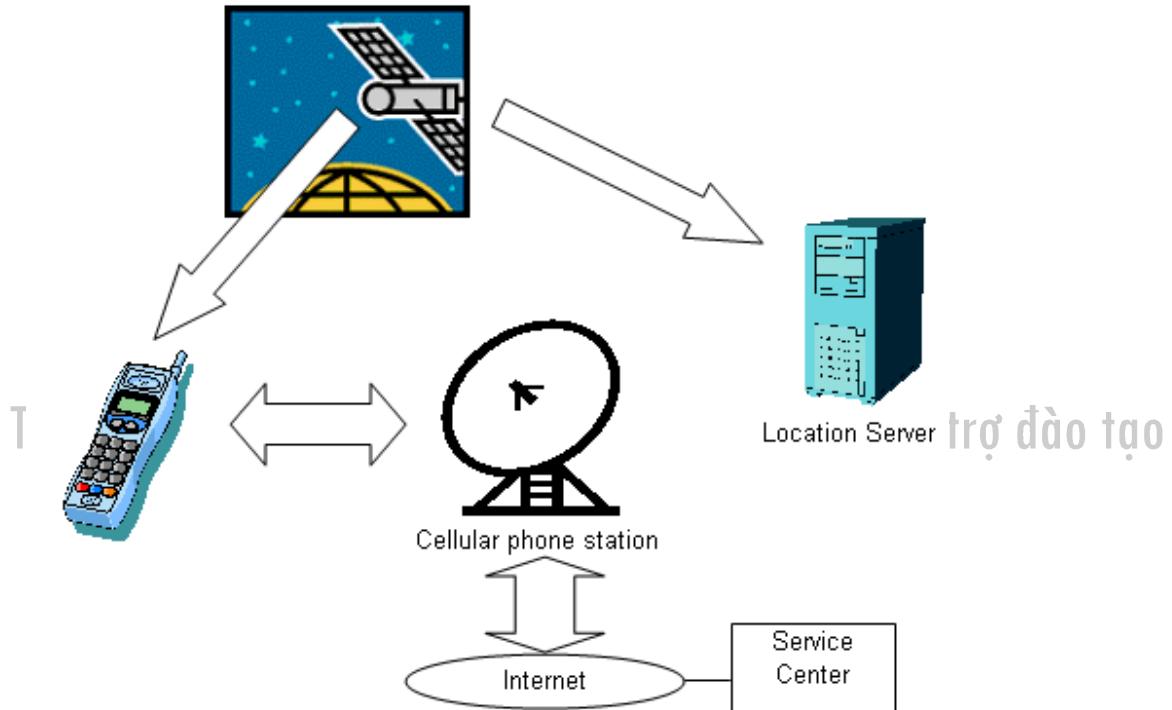
## Code Division Multiplex



Each channel is assigned a unique code. The same spectrum is used by all channels at the same time. This results in bandwidth efficiency. There is no need for coordination or synchronization. It is hard to tap and almost impossible to jam. It has a lower user data rates and a more complex signal regeneration. UMTS (Universal Mobile Telephone System) is using this method.

## Location based services

GPS (Global Positioning System)

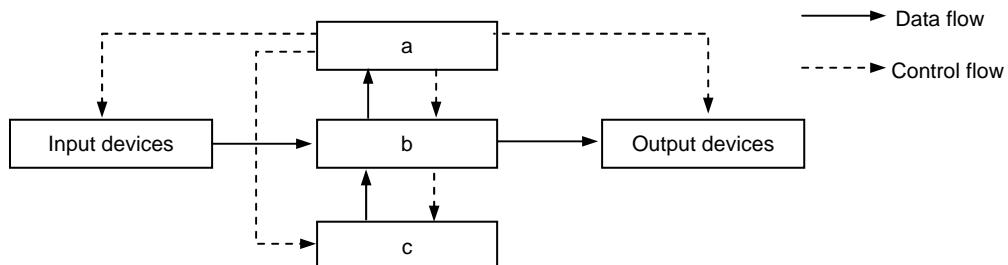


This utilizes satellites to allow users to determine their position. Navigation services can be provided through the use of GPS.

<http://www.vitec.org.vn>

## Exercises for No.1 Chapter4 (Hardware)

**Q1** What is the combination of words that should fill in the blanks of the diagram representing the computer basic configuration?



	a	b	c
A	Arithmetic unit	Main storage unit	Main storage unit
B	Main storage unit	Control unit	Arithmetic unit
C	Control unit	Arithmetic unit	Main storage unit
D	Control unit	Main storage unit	Arithmetic unit

**Q2** What is the appropriate explanation of a DRAM?

- A. DRAM represents 1 bit depending on whether the capacitor is charged or not. It is commonly used as a main storage unit.
- B. Data is written at the time it is manufactured. It is used as a microprogramming storage memory.
- C. Data can be written using a special device and erased with ultraviolet light.
- D. It is composed of flip flops. The speed is high but the manufacturing cost is high as well. It is used in cache memories, etc.

**Q3** Regarding the index modification of the machine language instruction, which of the following would be the effective address?

Address in which the instruction is stored: 1000

Value of the instruction language: 100

Value of the index register: 10

- A. 10
- B. 100
- C. 110
- D. 1100
- E. 1110

**Q4** Given the following circuit, when the input values are A=1, B=0, C=1, what is the appropriate output value for P, Q and R? Here represents an AND gate, represents an OR gate and represents a NOT gate.



	P	Q	R
A	0	1	0
B	0	1	1
C	1	0	1
D	1	1	0

**Q5** When the sum of 1-bit values A and B is represented in 2-bit values, which of the following corresponds to the combination of the logical expression of the higher bit D and the lower bit S? Here, "·" represents the logical product (AND), "+," the logical sum (OR) and  $\bar{A}$ , the negation (NOT) of A.

A	B	Sum of A and B	
		C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

	C	S
A	$A \cdot B$	$(A \cdot \bar{B}) + (\bar{A} \cdot B)$
B	$A \cdot B$	$(A + \bar{B}) + (\bar{A} + B)$
C	$A + B$	$(A \cdot \bar{B}) + (\bar{A} \cdot B)$
D	$A + B$	$(A + \bar{B}) + (\bar{A} + B)$

**Q6** In a computer with 3 types of instruction sets, which of the following corresponds to the MIPS value when their respective execution speed and frequency rate are as follows?

Instruction set	Execution speed (microseconds)	Frequency rate
A	0.1	40%
B	0.2	30%
C	0.5	30%

- A. 0.25      B. 0.8      C. 1.25      D. 4

**Q7** A given program executes instructions a, b, c and d in the following order:



The CPI required to execute each instruction is indicated in the following table. If 1 clock cycle of the CPU is 10 nanoseconds, how many nanoseconds will the CPU execution of this instruction string require?

Instruction	CPI
a	6
b	2
c	4
d	8

- A. 20      B. 32      C. 200      D. 320

**Q8** Which of the following is the method of the ordinary computer basic architecture that loads programs and data together in a computer storage device and sequentially reads and executes them?

- |                                  |                           |
|----------------------------------|---------------------------|
| A. Address method                | B. Virtual storage method |
| C. Direct program control method | D. Stored program method  |

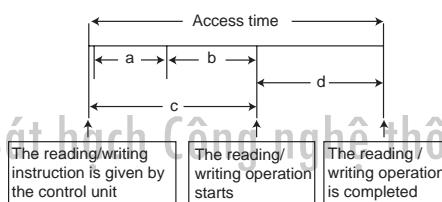
**Q9** Given the following magnetic disk unit specifications and conditions of the data subject to storage, how many tracks does the necessary area have when the blocking factor is 20? Here, the area is assigned by track and the file organization is sequential.

Magnetic disk unit specifications	
Storage capacity per track	25,200 Bytes
Inter-block gap	500 Bytes

Conditions of the data subject to storage	
Record length	200 Bytes
Number of records	10,000 Records

- A. 80      B. 83      C. 89      D. 100

**Q10** The following diagram represents the access time of the magnetic disk unit. Which is the a, b, c and d correct combination?



a	b	c	d
A Seek time	Search time	Latency	Data transfer time
B Seek time	Latency	Search time	Data transfer time
C Latency	Seek time	Data transfer time	Search time
D Latency	Data transfer time	Seek time	Search time

**Q11** Given the magnetic disk unit with the following performance, what is the average access time in milliseconds required to read the 2,000-byte-length-block data recorded in this magnetic disk?

Magnetic disk unit performance	
Storage capacity per track (bytes)	20,000
Revolution speed (revolution/minute)	3,000
Average seek time (milliseconds)	20

- A. 30      B. 31      C. 32      D. 42

**Q12** Regarding the optical disk characteristics, which of the following descriptions is correct?

- A. CD-ROMs have large storage capacity, but since high-level technology is required for their manufacture, compared to magnetic disks, the cost is higher for the same amount of information.
- B. In the magneto optical disk, which is one of the rewritable storage media, data is recorded by changing the medium magnetization direction.
- C. In the recordable optical disk, in which data is recorded by making microscopic holes in the medium, data can be rewritten as many times as required.
- D. Since the access mechanism of magneto optical disks is very similar to that of magnetic disks, the average access time is also of the same level.
- E. Since magneto optical disks are susceptible to heat, light and dust, compared to magnetic disks, the magneto optical disk's durability is lower.

**Q13** Which of the following is the most appropriate explanation of mirroring, which is one of the methods used to improve the magnetic disk unit reliability?

- A. By giving a mirror-like finish to the disk surface the resistance at the time the disk rotates is reduced.
- B. The data block and the parity block are stripped and stored across multiple disks.
- C. Besides the disks that record the data, another disk for parity recording is used.
- D. Identical data is recorded simultaneously in separate disks.

**Q14** Which is a feasible combination of interfaces for connecting the peripheral devices indicated below? Here ATA/ATAPI-4 represents the interface that is normally called IDE.

	Hard disk, CD-ROM	Modem	Keyboard
A	ATA / ATAPI-4	GPIB	SCSI
B	GPIB	SCSI	RS-232C
C	SCSI	RS-232C	USB
D	USB	IrDA	ATA / ATAPI-4

**Q15** If an image, whose height and width in pixels is 480 dots and 640 dots, respectively, is represented in 256 types of colors: approximately how many kilo bytes would be required in order to save this data in a storage device? It should be noted that no compression process is performed.

- A. 170
- B. 310
- C. 480
- D. 9,840
- E. 78,650

**Q16** Which of the following printers uses a heating element to melt the ink of the ink ribbon and is capable of printing on normal paper?

- A. Ink-jet printer
- B. Dot impact printer
- C. Thermal printer
- D. Thermal transfer printer
- E. Laser printer

### Chapter Objectives

This chapter focuses on multimedia systems that occupy important positions in the current computer systems.

- 1 Understanding the multimedia application fields
- 2 Understanding the concept of multimedia system design
- 3 Understanding available tools for multimedia system development.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



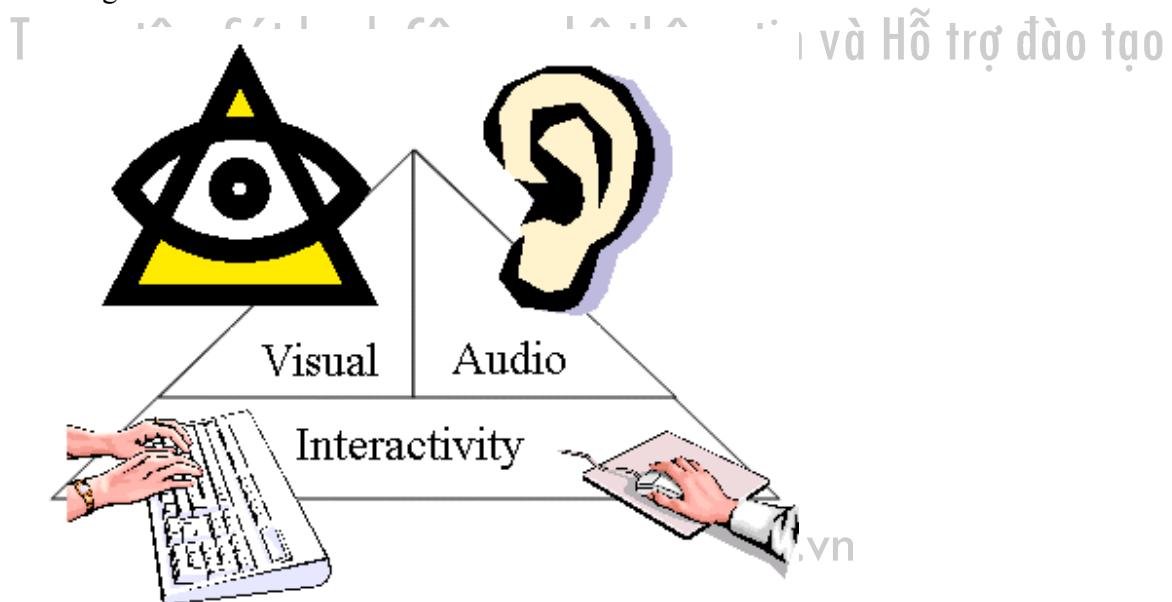
<http://www.vitec.org.vn>

## 5.1 Multimedia System Design

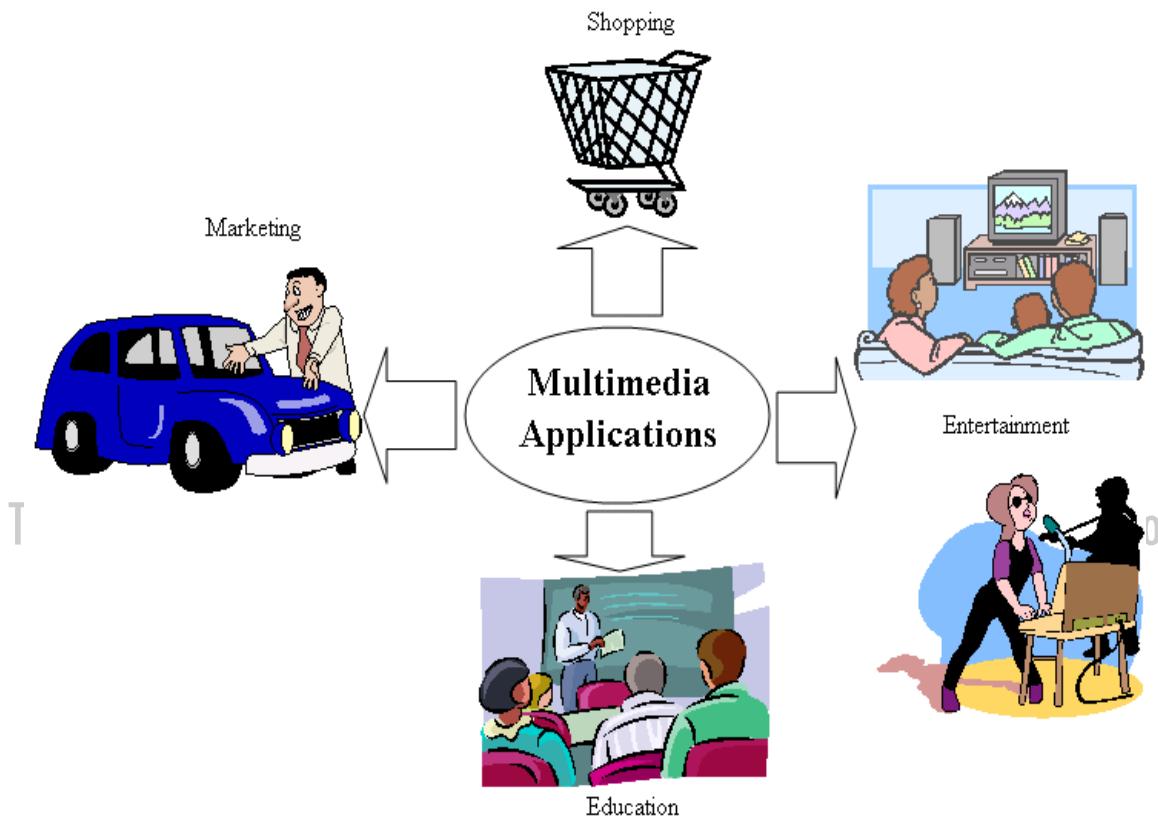
Multimedia comprises a combination of

- 1) Audio
- 2) Animation
- 3) Video
- 4) Static pictures
- 5) Text

Multimedia uses a combination of the visual, audio and tactile senses to deliver the message across.



### 5.1.1 Types of multimedia applications



#### Education

The benefits of using multimedia as a training tool are

- "Just in time" training can be implemented in multimedia.
- Courses can be taken any time and anywhere. Moreover, it allows the user to learn at their own pace.
- Interactive multimedia makes a good simulation tool.
- User interactions can be recorded, analyzed and reported.

#### Entertainment

Movies, Net gaming and concerts can be streamed as multimedia contents. They are many examples of the use of multimedia effects in the movie industry. Character Graphics is also becoming more prominent with whole clips comprising of computer generated characters.

#### Marketing

They are an increase use of multimedia in advertising. Computer characters have sometimes replaced the real world character in the advertisement.

### **Shopping**

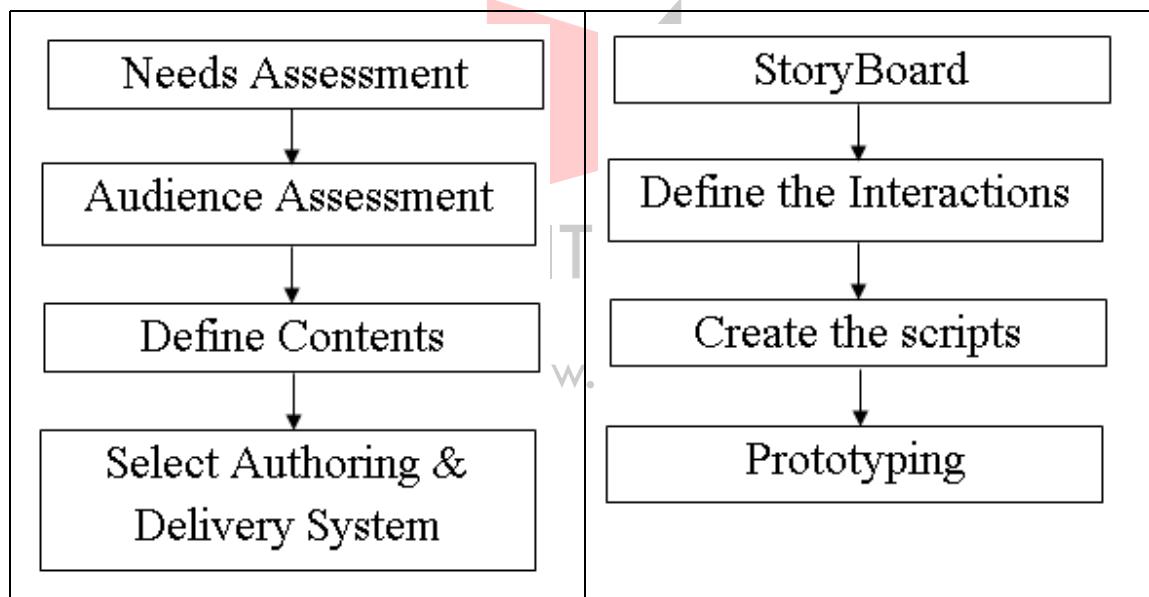
Multimedia allows the users to manipulate the products in 3 dimensional spaces. This is unlike a physical shop where it is possible to feel and touch the product. The use of multimedia allows the display of the products as a 3 dimensional figure that can be manipulated by the user.

In addition, the properties of the figure like color and pattern can be modified interactively.

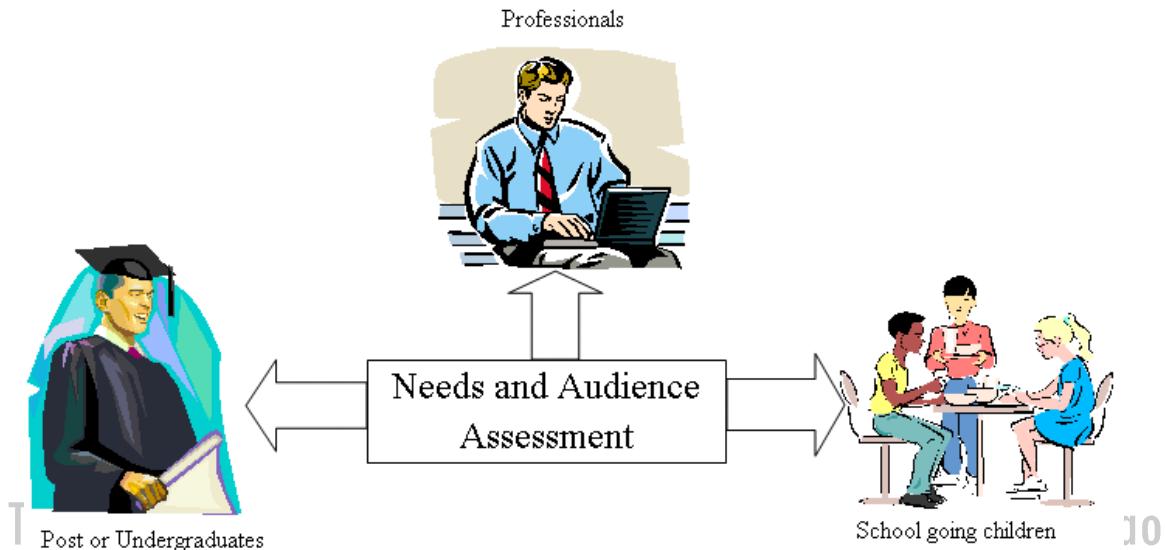
## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### 5.1.2 Design Steps

The steps for the design of a multimedia system is shown below



### 5.1.3 Needs Assessment



The objectives of the multimedia production have to be clearly spelt out. As shown in the example, above the needs of the audience varies from one group to the next.

Content for school going children need to be interesting enough to capture their attention and convert the learning experience into a fun filled experience.

Professionals will look for knowledge based content that allow skills to be learnt in as a short a time as possible.

VITEC

Undergraduates focus on content that allows them to do their assignments easily and to practice their skills through mock examinations and exercises provided by the contents. Organizations will focus on improving their learning curve. The knowledge requirement to perform the various processes in the work flow is collected and analyzed. These knowledge requirements become the objectives.

### 5.1.4 Audience assessment

- You must take technical limitations in to consideration as well as audience composition. Think about what machine the person

will be working on and what their connection to the Internet may be like (modem vs. direct link).

- What is their educational or occupational background? Will they understand technical jargon or be confused by it? Do they want to see pretty pictures or just get to the heart of the "data"?
- Once you have established with whom you wish to communicate you need to think about the best way to get your ideas across.  
This is where layout and design come to play.

You have to take the following criteria into consideration

- 1) Familiarity with the use of interactive media
- 2) Easy help
- 3) Ease of navigation

### 5.1.5 Define contents

The nature of the contents is very important to the site. Clarification with the user to the objectives and the knowledge requirement to achieve the objectives are essential to allow the creation of the right content.

Operation based content can be recorded and played back.

<http://www.vitec.org.vn>



rợ đào tạo

The following are defined when creating contents

- 1) Objectives
- 2) Use the medium that best conveys the information
- 3) Use the multimedia as a supportive and not just for decoration
- 4) Synchronize the multimedia with the text or picture
- 5) Make the interface interactive

Match the type of testing to the information covered.

### **5.1.6 Select Authoring and delivery System**

The following products can be developed

- Multimedia presentations
- CD-ROM titles
- Kiosks
- Computer-based training
- Artistic visualizations

- Web pages

The type of tool can be classified as

1) Video authoring

These are used to edit video. There include tools like Premier or AfterEffects that allow

2) Multimedia authoring

These include tools like Macromedia Director or Flash

3) Landscape generators

These include tools like Bryce, Vue

4) Character modeling

These cover tools like Poser.

5) Audio editing

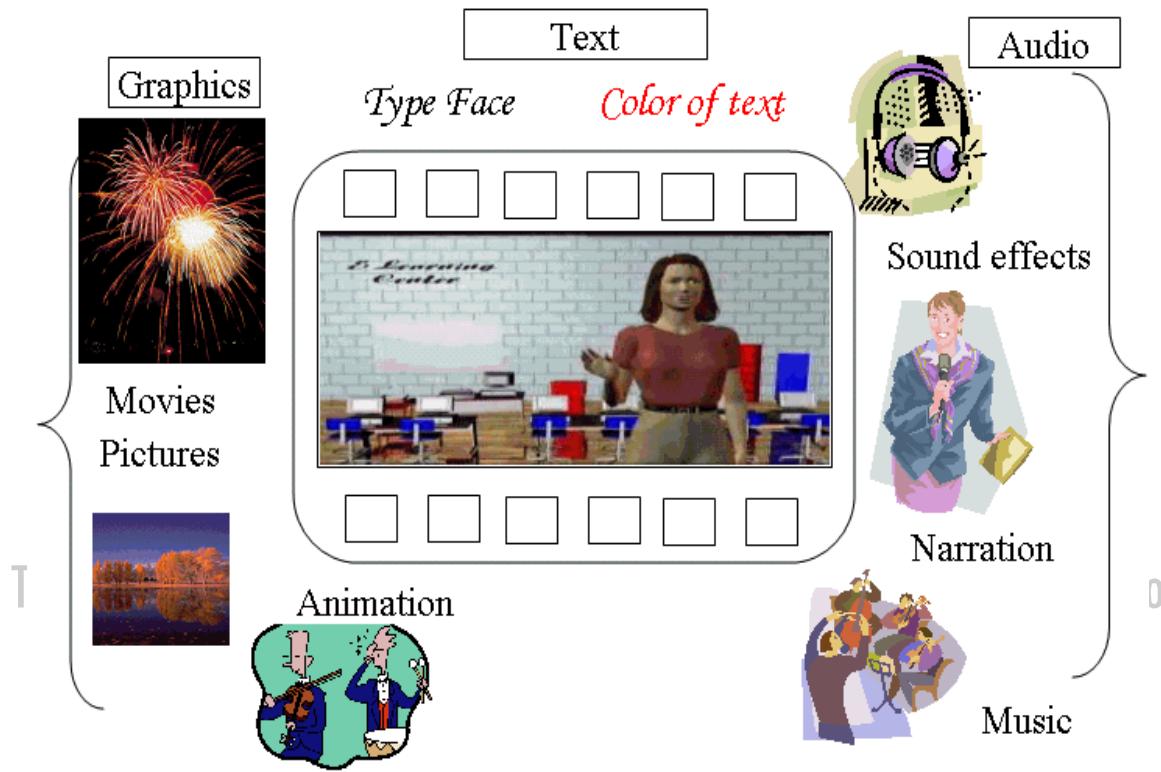
There include tools like SoundForge.

6) Studio

7) PowerPoint

8) Image editing software like Photoshop

Authoring base	Product
Presentation	PowerPoint
Timeline	Premier, Flash, Director



### Director

Macromedia *Director* uses a *theatre* metaphor to describe its components and its creations are called *movies*.

The main screen is called the *stage*, the stage is always open and is behind other windows.

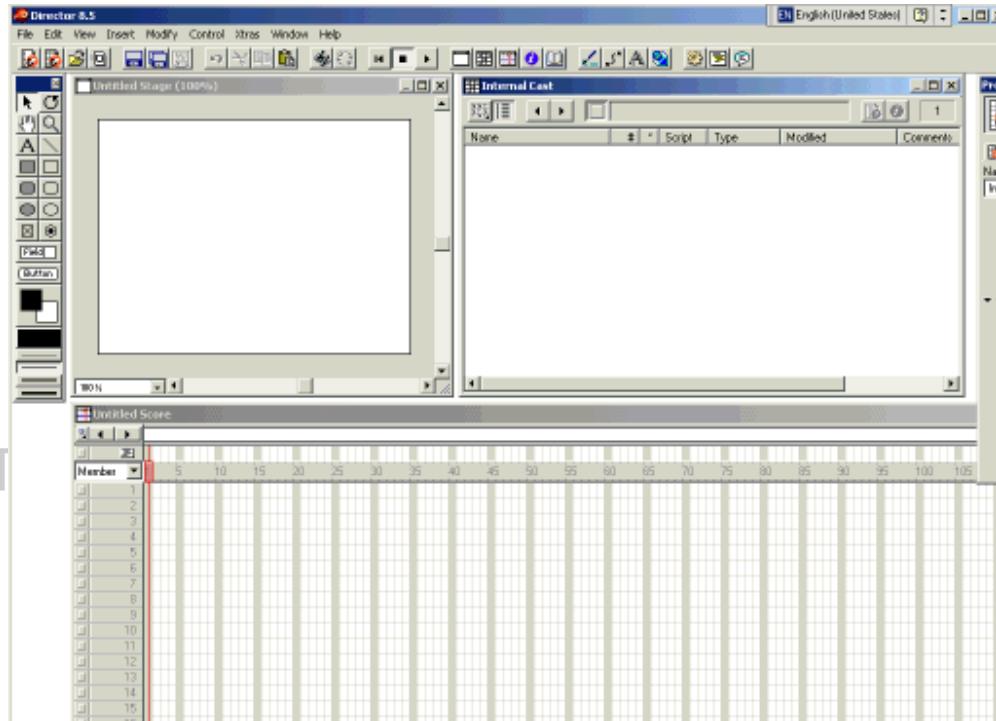
These include:

<http://www.vitec.org.vn>

- the *cast* window which displays the cast members, that is the files, art, sound, movies, etc that make up your production.
- The *score* window that keeps up with what cast member is on the stage in each frame of a movie and controls the timing, transitions, sound timing and palettes.
- The *script* window is where you type in instructions to create interactive links.

*Director* also includes a toolbar made up of short cut icons, a control panel similar to the controls on a VCR, and a tool palette. Because time-based authoring programs work by displaying a series of frames, they are especially good for creating animations. It is able to import movies, audio and pictures

Director also has support for the 3D format. This feature is useful if the content requires to convey spatial orientation.

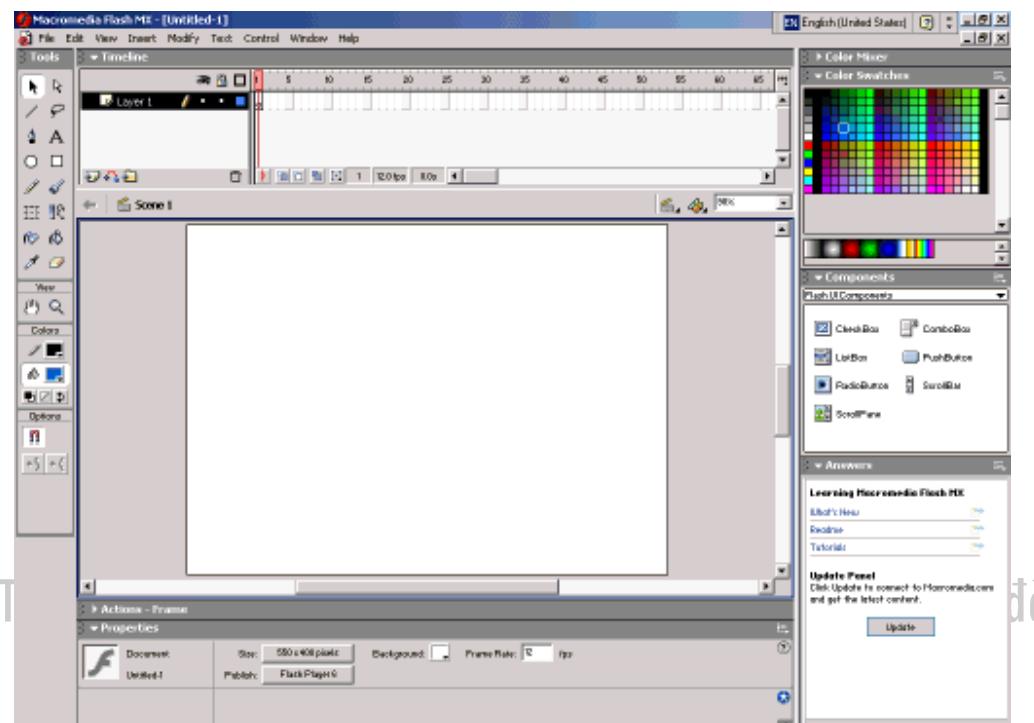


đào tạo

### Flash

Flash is similar to director. A library contains all the resources. It is intended for a 2 D type development.

<http://www.vitec.org.vn>



đào tạo

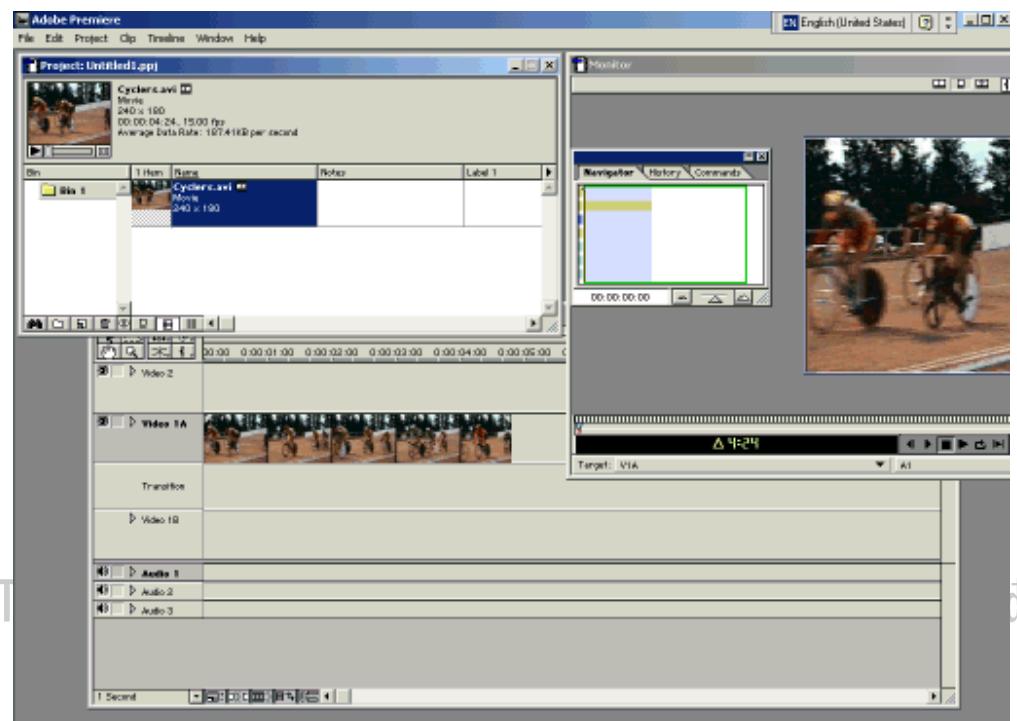
## Adobe Premier and AfterEffects

These are video editors. They support the AVI, MOV and MPEG. They also allow the capture and output of the medium into tape.

A suitable video capture board must be used. Movies are good to showing operations.

Keep in mind that the size and length of the movie may make it difficult to distribute.

Unlike Flash or Director, scripting is not supported. It is used more as a post production tool to add the opening, closing credits. It is also used to edit the movies and add titles or effects on the movies.



đào tạo

### Poser

This creates characters and allows movement to be added to the character. It outputs static images or video.

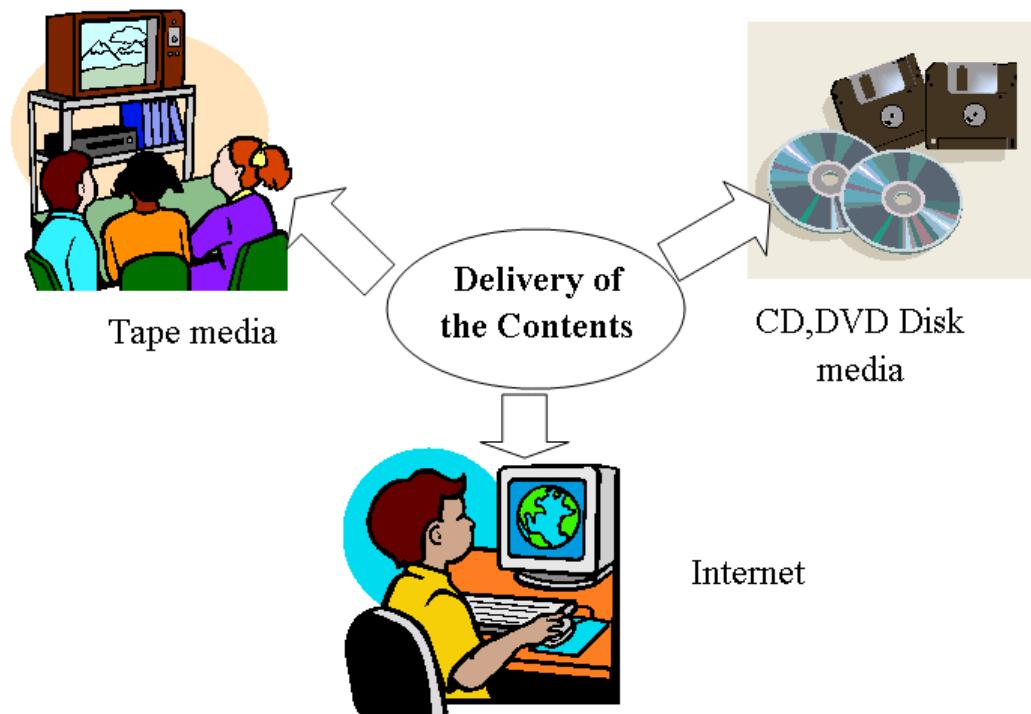


<http://www.vitec.org.vn>



đào tạo

Depending on the communication infra-structure, the content may be distributed in various forms.



### 5.1.7 Storyboard

	
Lead in background music start.wav ( 5 minutes) Figure starts to speak	Figure starts to wave and narrate
Narrative: Good morning, Ladies and gentlemen ....	Narrative: I hope that ....

The relationship between individual screens and actions represented as a sequence of images is known as a Storyboard.

It comprises a set of images showing the menus, dialogue boxes and windows. The navigation structure and functionality available in the system is shown by the story board. It is shown to the potential users, to allow the visualization of the composition.

The scope of the interface can be seen clearly in the storyboard. The use of storyboards allows the design to be understood easily.

Each panel in the story board also shows what resources are in the frame. e.g. the narration, background music or movie to be shown

A preview is possible using a storyboard. It can be treated as a visual script for camera angles, lighting, sound effects, prop arrangement, and the narrative development and continuity.

The procedure for creating a storyboard is

1. Give consideration to the scenarios of use which the storyboard will reflect. A storyboard may represent several activities such as entering, saving or printing information. Alternatively a separate storyboard may be created to represent each distinct theme.
2. Construct the storyboard as a sequence of screen representations using separate images to reflect changes in system appearance. Thus the storyboard indicates the availability and purpose of dialogue windows, menu items, toolbars and icons.
3. The elements of a storyboard can be annotated with explanatory captions to aid audience understanding and evaluation.
4. The completed storyboard can be shown to design teams as well as intended users to solicit evaluative feedback. Several storyboards can be created and shown to an audience in order to explore different design options.
5. It may be useful to video or audio record the feedback sessions for later review or to show to other colleagues.

### 5.1.8 Define the interactions

#### Building blocks

The model upon which the building blocks of a multimedia application are based is referred to as an *activities model*. An activity can be defined as a series of related actions, happening one after another. Activities invariably commence at a single starting point, but they can have various outcomes.

The model for an application is designed by clustering groups of activities together using associative connections called links.

Broadly this model can consist of two types of activities: *presenting* activities and *controlling* activities. Multimedia applications of enormous detail and complexity can be designed from webs of these activities.

#### Presenting activity

The most basic activity is to display a single, non-interactive block of text, image or video sequence. A more complex activity could involve the display of a collection of individual images and video segments. A further refinement might involve the use of

audio to cue the start of a video and then present a series of images each cued to certain sections of the audio track.

### **Controlling activity**

The most basic variation of a one-after-the-other approach is the use of a selection structure based on menus. From an opening menu screen, the user chooses between a number of selections. On selecting the item a single activity is usually performed, after which the user is returned to the main menu. The menu activity in this respect is an example of a controlling activity.

Each item has a single starting point but has multiple end points. The nature of the input made by the user determines which of the multiple end points will be chosen.

Using this simple activity model, it is possible to identify various types of multimedia application structures: linear presentation, data driven engine, hierarchical menu, information retrieval and *hypermedia*.

### **Linear presentations**

A *linear presentation* has one event after another but, for such a presentation to be of benefit to the user, there needs to be some element of control, with the presentation alternating between presenting and controlling activities. The controlling activities fix the duration of the presentation.

For example, in the event of a live paper presentation at a conference, the controlling activities would be programmed to accept cues from the speaker during their presentation. The cues could either be keystrokes (any key) or the use of a mouse device to move between the different screens. This allows the presentation to be synchronized with the messages being expressed by the speaker.

A more sophisticated speaker-support presentation might allow some other forms of information manipulation at each control point, such as the ability to go backwards or

forwards through a presentation, or the option to jump out of the presentation into another application.

### **Data-driven engine (DDE)**

One strategy to make linear presentations easier to develop, and simpler to control, is to use a data-driven engine approach. This involves using an engine, which interprets instructions from a file that defines what is to occur at each stage of the presentation. The file (which is usually a plain text — ASCII — file) has the items to be shown arranged in the most appropriate order; thereafter each presenting activity is included to show whatever is described. This is followed by a single controlling activity that awaits user commands about what should happen next. An example of this could involve the presentation of a series of slides in which the sequence list would merely be a list of the titles of the slides to be shown.

The composition of a data-driven engine (DDE) remains the same irrespective of how many items are shown: the controlling file just gets extended. Should the designer wish to rearrange the sequence of items all that needs to be done is to edit the items in the file. By programming using an authoring or coding system it is possible for the presentation activity to read and load different script files and, depending on the level of sophistication of the coding, it may also be able to interpret more advanced instructions from the file. Moreover, different presentations can be created simply by making new links in the file.

### **Hierarchical menus**

A menu is executed by using a presenting activity to reveal the menu, and then a controlling activity so that the user's selection is activated. This might take the user to another presenting activity or it might branch off to another level of the menu. When a menu selection is completed, the user is usually returned to the menu. Likewise, when the

user exits a sub-menu, they return to the main menu. The designer needs to ensure that the user goes back to the main menu before closing the application.

Such a menu structure can quickly become difficult to use, especially if some of the activities (and menu items) appear in several menus. Using a data-engine approach may help to simplify the whole process by associating with each menu a script file that outlines the menu items and the titles of the presentations that will be executed when chosen. Code for a menu engine can be generated to display the menu from this file, which then sends the name of the selected item to a single-presentation engine program. This simplifies the menu structure down to two engine programs

- presentation and menu — and the detailed application structure is actually stored in a
- group of text files, which will be easy to modify and manage without conventional programming skills.

### **5.1.9 Create the scripts**

The actual scripts showing the details on each scene is created. The storyboard can be enhanced with details of the script



#### **Prototyping**

#### **Visual clarity**

<http://www.vitec.org.vn>

We seek clarity, order, and trustworthiness in information sources, whether they are traditional paper documents or Web pages. The spatial organization of graphics and text on the Web page can engage the user with graphic impact, direct the user's attention, prioritize information, and make the user's interactions with your Web site more enjoyable and more efficient

#### **Consistency**

Establish a layout grid and a style for handling your text and graphics, then stick with it to build a consistent rhythm and unity across all the pages of your site. Repetition is not

boring; it gives your site a consistent graphic identity that reinforces a distinct sense of "place", and that makes your site more memorable.

A consistent approach to layout and navigation allows readers to quickly adapt to your design, and to confidently predict the location of information and navigation controls across the pages of your site. If you choose a graphic theme, use it throughout your site.

*Consistency* in the use of all visual elements, tools, commands, operations and organizational style is the foundation of good interface design.

Knowing what type of information will be covered and when and where it will appear builds *predictability* into a software application.

**Good mappings**  
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

There should be clear relationships between the user's actions and the machine's

responses, the controls and their effects and the state of the system and the interface.

Graphic user interfaces were designed to give people direct control over their personal computers. Users now expect a level of design sophistication from all graphic interfaces, including Web pages. The goal is to provide for the needs of all of your potential users, adapting Web technology to their expectations, and never requiring the reader to simply conform to an interface that puts unnecessary obstacles in their paths.

<http://www.vitec.org.vn>

### 5.1.10 Feedback

Feedback may take the form of:

- written responses
- visual responses
- auditory responses

Ideally, the user should receive immediate full and continual feedback concerning the results of all self-initiated and program-driven actions. This promotes a sense of progress. Feedback from the user interface should be immediate and unambiguous, in the form of written, visual or auditory signals indicating that the computer has received input from

the user and is acting upon that stimulus. For instance, clicking on a screen button may cause the button colors to reverse or a button 'click' sound may be heard.

### **An easily-grasped metaphor or concept**

Use metaphors involving concrete, familiar ideas and make the metaphor plain so that users have a set of expectations to apply to the computer environment. For example,

desktop  
folders  
trash can  
menus

The metaphor can be a powerful verbal and semantic tool for conveying both superficial and conceptual similarities between familiar and novel situations. Learned associations

should mould the metaphor because these familiar experiences help users to grasp complex ideas.

An alternative to the metaphor is the development of a unifying *concept*. A concept might be a derivative of a story line, a particular theme, or a stylized look.

### **Efficient navigation schema**

Users should be able to find out where they are quickly and how to get back to where they were.

The main interface problem with most Web sites is the lack of a sense of where you are within the local organization of information. Users should always be able to return to your home page, and to other major navigation points in your local site.

Navigation tools include

- directional devices
- maps
- headings
- visual effects
- timelines
- on-line help
- user guidance facilities

- sound cues

### **System transparency**

It is important that the user can 'look through' your interface and concentrate on what's important - the information your document presents. To this end, the interface should be *transparent*, concealing the complexity of the system so that the user can focus on the tasks, rather than on the tool, the data rather than the "data container".

The user should always feel in direct control of the computer interface and should never feel that the computer has "automatically" taken actions that could arbitrarily change the user's preferences, destroy data, or force the user to waste time.

### **Data organization**

Psychologically, a user needs to get closure on one stage of the interactive process in

order to move onto the next. This is accomplished by grouping information into readily digestible units. Visually stratifying various layers of information reduces noise and enriches the content of the display.

Good screen design creates visual logic, an optimal balance between visual sensation and graphic or text information. A careful, systematic approach to screen design can simplify navigation, reduce errors, and make it much easier for users to take full advantage of the information and features of your Web site.

- Develop content before design
  - design should be determined by the information you want to communicate
- Have an objective
  - to educate
  - to inform
  - to entertain
- Establish a visual hierarchy
  - emphasize important elements
  - organize content logically and predictably

## **Interactivity**

Users want to feel they are in charge of the computer's activities. They want their physical actions to have physical results and they want buttons, icons, menus etc. to provide feedback.

An active participatory interface allows the users to control the environment by directly manipulating materials. Interactivity is challenging and inspires the user to respond more creatively. Consistency in screen design is important because of the interactive nature of the computer interface, where visual design elements appear and disappear in response to the user's actions.

User's actions should always be reversible.

## **Evaluation**

User evaluation is essential during the design process. User evaluation, combined with feedback from content specialists, keeps the developer on track.



<http://www.vitec.org.vn>

---

## Answers for No.1 Chapter 1 (Basic Theories of Information)

### Answer list

Answers

Q 1: C	Q 2: B	Q 3: D	Q 4: B	Q 5: C
Q 6: D	Q 7: D	Q 8: D	Q 9: C	Q 10: B
Q 11: C	Q 12: C	Q 13: A	Q 14: C	Q 15: B
Q 16: B	Q 17: B	Q 18: C	Q 19: D	

### Answers and Descriptions

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
Q1

Answer

C.  $k < M < G < T$

Description

K (kilo)	$10^3$
M (Mega)	$10^6$
G (Giga)	$10^9$
T (Tera)	$10^{12}$



Q2

<http://www.vitec.org.vn>

Answer

B. 1 microsecond / 1,000,000

Description

M (mili)	$10^3$
$\mu$ (micro)	$10^6$
N (nano)	$10^9$
P (pico)	$10^{12}$

1 picosecond

= 1 second  $\times 10^{-12}$

= 1 microsecond  $\times 10^{-6}$

## **Q3**

### **Answer**

D. 10111011

### **Description**

Addition of binary numbers

$$0+0=0, 0+1=1, 1+0=1, 1+1=10$$

01010101

+01100110

10111011 ← Answer

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

## **Q4**

### **Answer**

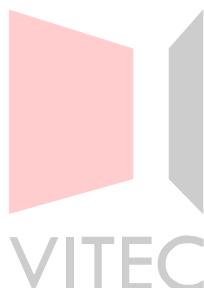
B. BE

### **Description**

Convert hexadecimal to binary

$$(DD)_{16} = (11011101)_2$$

$$(1F)_{16} = (00011111)_2$$



<http://www.vitec.org.vn>  
Do subtraction as follows.

11011101

-00011111

10111110 = (BE)<sub>16</sub>    Answer : B

Note: other possible solutions

Compute as it is in hexadecimal

or

Convert to decimal, then do subtraction

## Q5

### Answer

C. 3.5

### Description

Convert the numbers from binary format to decimal format

$$1.1011 = 1 \times 2^0 + 1 \times 1/2 + 1 \times 1/2 \times 2^2 + 1 \times 1/2 \times 2^2 \times 2 = 1 + 0.5 + 0.125 + 0.0625 = 1.6875$$

$$1.1101 = 1 \times 2^0 + 1 \times 1/2 + 1 \times 1/2 \times 2 + 1 \times 1/2 \times 2^2 \times 2 = 1 + 0.5 + 0.25 + 0.0625 = 1.8125$$

Do addition as follows.

1.6875

+1.8125

3.500

## Q6

### Answer

D. 0.5

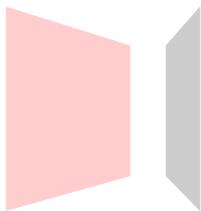
### Description

Relationships between binary fractions and decimal fractions are as follows.

$$(0.1)_2 = 1 \times 2^{-1} = (0.5)_{10}$$

$$(0.01)_2 = 0 \times 2^{-1} + 1 \times 2^{-2} = (0.25)_{10} \dots$$

$$(0.001)_2 = 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = (0.125)_{10} \dots$$



Therefore, a binary fraction can be without errors if it is represented as the sum of one or more of the above.

Among listed options,  $(0.5)_{10}$  is the answer.

## Q7

### Answer

D.  $\frac{73}{512}$

### Description

$$(0.248)_{16}$$

$$= 2 \times 16^{-1} + 4 \times 16^{-2} + 8 \times 16^{-3}$$

$$\begin{aligned}
 &= (2 \times 16 \times 16 + 4 \times 16 + 8) / 16 \times 16 \times 16 \\
 &= (512 + 64 + 8) / (16 \times 16 \times 16) = 73 \times 8 / 16 \times 16 \times 2 \times 8 \\
 &= 73 / 512
 \end{aligned}$$

## Q8

### Answer

D. 0100 0011 0010 1100

### Description

In packed decimal,

Each digit is represented by 4 bit BCD code

“+” or “-“ sign is represented by the last 4bit

$(1100)_2$  for “+”, and  $(1101)_2$  for “-“

Therefore, a decimal number +432 is represented as follows. → The answer is D.

0100	0011	0010	1100
4	3	2	+

- A. 0000 0001 1011 0000
- B. 0000 0001 1011 1100
- C. 0001 1011 0000 1100
- D. 0100 0011 0010 1100
- E. 0100 0011 0010 1101

<http://www.vitec.org.vn>

## Q9

### Answer

C.  $-2^{n-1}$  to  $2^{n-1} - 1$

### Description

In fixed decimal format representing a negative number using 2's complement, an n-bit binary number can represent a number ranging from  $-2^{n-1}$  to  $2^{n-1} - 1$ .

For example, in case of 8 bits, from 1000 0000 (= -128 in decimal) to 0111 1111 (= 127 in decimal).

## Q10

### Answer

B. Subtractions can be processed as additions.

### Description

Since the addition of a number and its 2's complement make zero as shown in the following example, 2's complement of a number can represent a negative number.

#### [Example]

$5 = 00000101$ , 2's complement of 5 =  $11111011$ , and those two make zero.

00000101

+11111011

100000000 (leftmost 1 is discarded because it is outside of 8 bits)

And then Introduction of negative numbers makes it possible to process subtractions (e.g.

$10 - 7$ ) as additions (e.g.  $10 + (-7)$ ).

## Q11

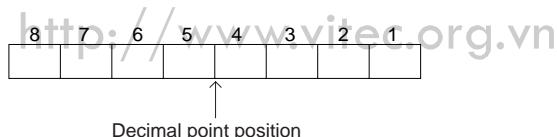
### Answer

C. 10100110

### Description



8-bit fixed decimal point format (decimal point between the 4<sup>th</sup> and 5<sup>th</sup> bits) of -5.625



1) Decimal part

$$5 = (0101)_2$$

2) Fractional part

$$0.625 \times 2 = \underline{1}.25 \quad (0.1)_2$$

$$0.25 \times 2 = \underline{0}.5 \quad (0.10)_2$$

$$0.5 \times 2 = \underline{1} \quad (0.101)_2$$

By doubling the number and taking the decimal part of the result,

$$0.625 = (0.101)_2$$

Therefore

$$5.625 = (0101.1010)_2$$

This is represented as 01011010 in the specified format.

2's complement of the above is 10100110

## Q12

### Answer

C		
0	1111	11000000000
1 bit	4bits	11bits

### Description

- 1) Represent 0.375 in binary format

$$0.375 \times 2 = 0.75 \quad (0.0)_2$$

$$0.75 \times 2 = 1.5 \quad (0.01)_2$$

$$0.5 \times 2 = 1 \quad (0.011)_2$$

Therefore

$$0.375 = (0.011)_2$$

- 2) Represent  $(0.011)_2$  in floating point

Mantissa is  $(0.11)_2$

Exponent is -1

- 3) Represent exponent in 4-bit binary

-1 in 2's complement is  $(1111)_2$

- 4) Summary of the above

Mantissa sign	<a href="http://www.vitec.org.vn">http://www.vitec.org.vn</a>
Exponent part	1111
Absolute value of the mantissa part	11000000000

## Q13

### Answer

A. 2AF3

### Description

ABCD = 1010 1011 1100 1101

By shifting this 3 bits to right,

101010111001101 original

001010101110011 shifted

Therefore,

$$0010\ 1010\ 1111\ 0011 = 2AF3$$

## Q14

### Answer

C. 4

### Description

Shifting one bit to left doubles a binary number.

For example

00001001 original (=9 in decimal)

00010010 1-bit shifted to left (=18 in decimal)

$$m \times 19 = (m \text{ after a-bit shifted to left}) + m \times 2 + m$$

Therefore, (m after a-bit shifted to left) should make  $m \times 16 = m \times 2^4$ , doing 1-bit shifting to left 4 times.



## Q15

### Answer

B. -13

### Description

+100 in binary is 01100100

0	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---

the 2's complement representation of the decimal number -100 is obtained as follows.

1) Switch all the 0 and 1 bits of the original bit string

1	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---

2) Add 1 to the above

1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

Therefore, 2's complement of 100 is 10011100

Shifting 3 bits to the right, using arithmetic shift (i.e. the sign bit is NOT sifted, and the bit shifted is lost and the sign bit value is put into the vacated bit positions)

The result is 11110011

1	1	1	1	0	0	1	1
---	---	---	---	---	---	---	---

1) Subtract 1 from the above.

1	1	1	1	0	0	1	0
---	---	---	---	---	---	---	---

2) Switch all the 0 and 1 bits of the above

0	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---

$$(00001101)_2 = (13)_{10}$$

Therefore, the result is -13. The answer is B.

## Q16

### Answer

- B. Due to the limited number of digits in the number representation, it is an error generated as a result of rounding off, rounding up, or omitting portions smaller than the least significant digit.

### Description

A describes “overflow”  
(A. It is an error generated when an operation result exceeds the maximum value that can be processed by the computer.)

B describes “rounding error” → answer

C describes “cancellation”.

- (C. It is an error generated due to the loss of the most significant values in the subtraction operation of numeric values, whose absolute values are almost equal.)

D describes “Loss of Information”

- (D. It is an error generated due to the loss of the least significant values of the mantissa portion of the numeric value, with the lower exponent value in the subtraction operation of floating point numbers.)

## Q17

### Answer

<http://www.vitec.org.vn>

- B. 6

### Description

A to Z 26 characters

0 to 9 10 characters

To represent above 36 different characters using n bits, the following should be true.

$$2^n \geq 36$$

Since  $2^5 = 32 < 36 < 64 = 2^6$ , the answer is 6 → B

- A. 5                    B. 6                    C. 7                    D. 8

## Q18

### Answer

C. (NOT x) AND y

### Description

In this question, the expression that is equivalent to the operation given in the following truth table is to be found.

Truth table			
x	y	x	y
True	True	False	
True	False	False	
False	True	True	
False	False	False	

Logical operations are done as follows.

### AND operation

x	y	x AND y
True	True	True
True	False	False
False	True	False
False	False	False

### OR operation

x	y	x OR y
True	True	True
True	False	True
False	True	True
False	False	False

### NOT operation

x	NOT x
True	False
False	True

Because of the following result, the answer is c.

x	y	NOT x	(NOT x) AND y
True	True	False	False
True	False	False	False
False	True	True	True
False	False	True	False

## Q19

### Answer

D.  $(\bar{A} \cdot \bar{B}) + \bar{C}$

Trung Tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### Description

In this question, the expression that is equivalent to  $(\bar{A} \cdot \bar{B}) + \bar{C}$  is to be identified.

Apply the De Morgan's law to solve this question.

Let  $X=A+B$  in the given expression. Then

$$(\overline{A+B}) \cdot C = \overline{X \cdot C} = \overline{X} + \overline{C} = (\overline{A+B}) + \overline{C} = (\bar{A} \cdot \bar{B}) + \bar{C}$$

Therefore, the answer is D.

VITEC

<http://www.vitec.org.vn>

---

## Answers for No.1 Chapter 2 (Data Structures)

### Answer list

Answers	_____	_____	_____	_____	_____	_____	_____	_____	_____
Q 1:	C	Q 2:	B	Q 3:	B	Q 4:	D	Q 5:	C
Q 6:	C	Q 7:	D	Q 8:	B	Q 9:	A	Q 10:	C
Q 11:	B	Q 12:	C						

### Answers and Descriptions

#### Q1

**Answer**  
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
c. 190

##### Description

The position of a[5,6] in the array is as follows.

a[1,1]	a[1,2]		...					...		a[1,10]
a[2,1]	a[2,2]		...					...		a[2,10]
a[3,1]	a[3,2]		...					...		a[3,10]
a[4,1]	a[4,2]		...					...		a[4,10]
a[5,1]	a[5,2]	a[5,3]	a[5,4]	a[5,5]	a[5,6]					

That is, it comes in the 46th.

The 2nd one (a[1,2])'s address is  $100 + 2^1 = 102$

and the third one(a[1,3])'s address is  $100 + 2^2 = 104$ .

Therefore, the 46th one's address is

$$100 + 2^4 \times 6 = 190$$

#### Q2

##### Answer

- b. The pointer for Shizuoka to be set to 70 and that for Atami to be set to 150

##### Description

This unidirectional list can be visualized as follows.

10 Tokyo-->50 ShinYokohama-->90 Atami-->70 Hamamatsu-->30 Nagoya

After inserting Shizuoka between Atami and Hamamatsu, it will be

10 Tokyo-->50 ShinYokohama-->90 Atami-->150 Shizuoka-->70 Hamamatsu-->30 Nagoya

Therefore, Shizuoka's pointer should be 70. Atami's pointer should be 150.

Pointer for the head	Address	Data	Pointer
	10	Tokyo	50
	30	Nagoya	0
	50	Shin Yokohama	90
	70	Hamamatsu	30
	90	Atami	150
	150	Shizuoka	70

### Q3

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo  
b. Queue

#### Description

Queue is called the first-in first-out (FIFO) system. → Answer

Stack is termed the last-in first-out (LIFO) system.

Binary tree is a kind of tree structure in which order of insertion and order of access are independent (neither FIFO or LIFO)

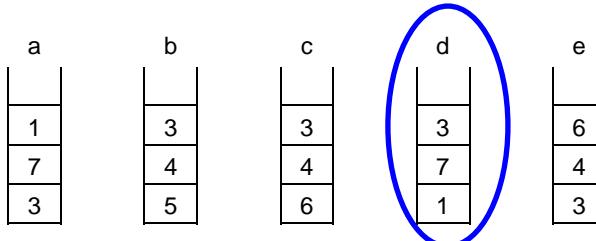
Heap is a kind of binary tree.

VITEC

### Q4

<http://www.vitec.org.vn>

#### Answer



#### Description

Stack manages data in FILO(First In Last Out) basis.

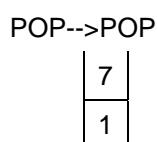
PUSH 1-->PUSH 5-->POP-->PUSH 7-->PUSH 6-->PUSH 4-->POP-->POP-->PUSH 3

PUSH 1-->PUSH 5





PUSH 7-->PUSH 6-->PUSH 4



PUSH 3



## Q5

### Answer

c. 4

### Description

<http://www.vitec.org.vn>

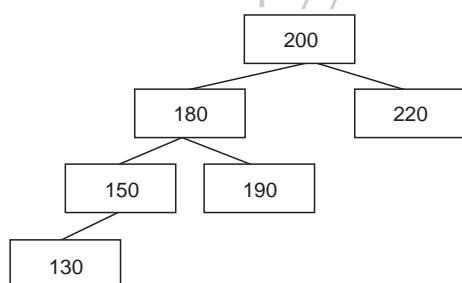


Figure 1 Binary tree

Subscript	Value	Pointer 1	Pointer 2
1	200	3	2
2	220	0	0
3	180	5	a
4	190	0	0
5	150	6	0
6	130	0	0

Figure 2  
Array representation of a binary tree

In the above Figure 1, for each node represented by the subscript number, the Pointer1 indicates the index value of its left child node, the Pointer2 indicates the index value of its right child node. "a" means the index value of the 180's right node. What comes there is 190 and its index value is 4.

## Q6

### Answer

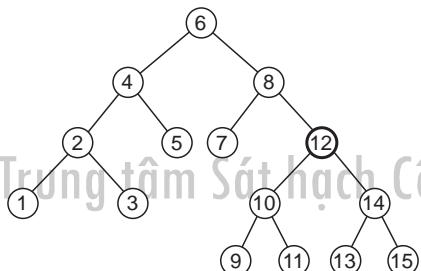
c. 13

### Description

In a binary tree, each node must satisfy the following.

max key value of its left nodes < its key value < min. key value of its right nodes  
therefore,

the ex-12 node should have some value greater than 10 and less than 14.



## Q7

### Answer

d. hicdbjkgea

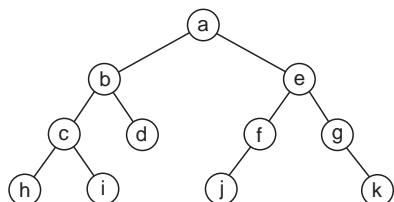
### Description

Search is performed using the post-order method, that is,

With a leaf at the bottom left as a starting point, the right side of each node is tracked in a sequential way.

<http://www.vitec.org.vn>

Therefore h,i and c are shown first.

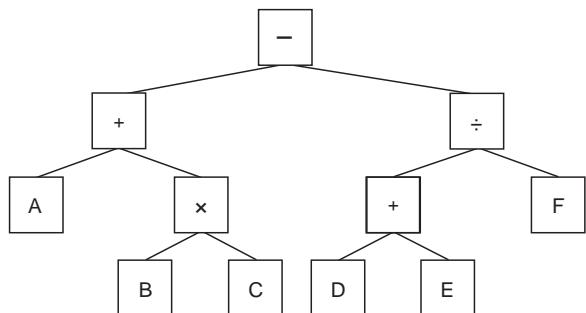


## Q8

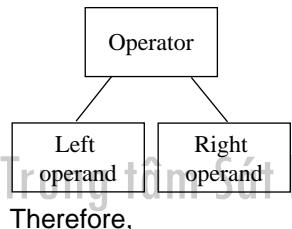
### Answer

b. A + B × C - (D + E) ÷ F

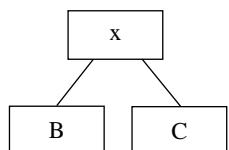
### Description



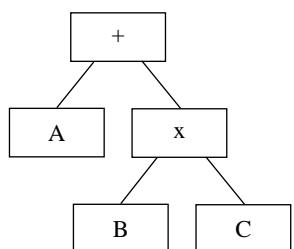
### Notation



Therefore,



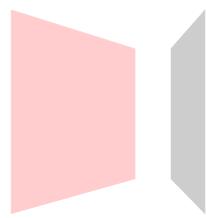
represents  $B \times C$ ,



represents  $A + B \times C$

By interpreting the rest in a similar manner, the answer is

$$A + B \times C - (D + E) / F$$



<http://www.vitec.org.vn>

### Q9

#### Answer

- Split and merge nodes to allow the hierarchical depth to become the same.

### Description

a is correct.

b refers to hash

- (b. Identify the position where data is to be stored by using a certain function and key value.)  
 c describes sequential access files  
 (c. Only sequential access to head data and subsequent data is possible.)  
 d represents partitioned organization files  
 (d. There are a directory and a member (or members). A member is a sequentially organized file.)

## Q10

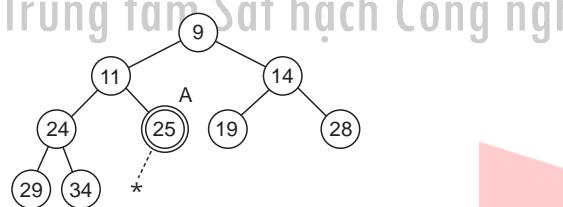
### Answer

c. 11

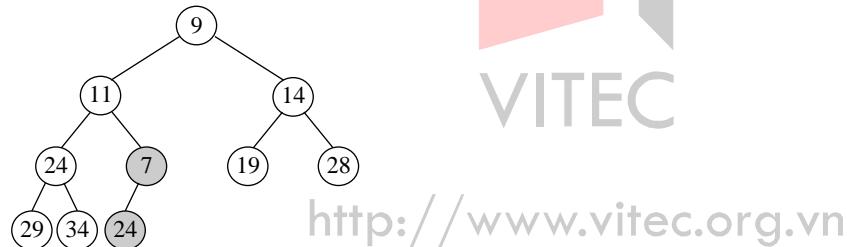
### Description

In this question, the element in the position A after 7 is inserted at the “\*” position is to be found.

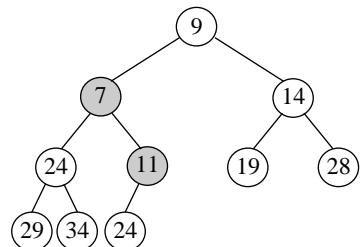
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



1) Swap 7 and 25 because  $7 < 25$

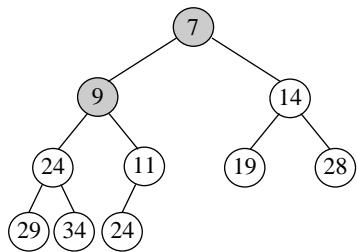


2) Swap 7 and 11 because  $7 < 11$



3) Swap 7 and 9 because  $7 < 9$ .

This satisfies the requirement and completes swapping.



After the above swapping, the element in the position A is 11. Therefore, the answer is c.

## Q11

### Answer

b. 2

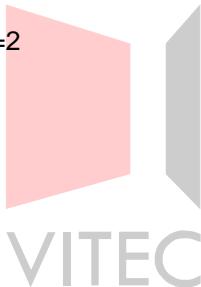
**Description**

Hash function

$$\text{mod}(a_1 + a_2 + a_3 + a_4 + a_5, 13)$$

$$\text{mod}(54321, 13) = \text{mod}(5+4+3+2+1, 13) = 2$$

Therefore index for 54321 is 2.



## Q12

### Answer

c. 0.7

**Description**

<http://www.vitec.org.vn>

A hash table with 10 elements

Therefore the probability that synonym does not occur for 5 values is

$$9/10 \times 8/10 \times 7/10 \times 6/10 = 3024/10000$$

The probability that collision occur is

$$1 - 0.3024 = 0.6976$$

---

## Answers for No.1 Chapter3 (Algorithms)

### Answer list

Answers	Q 1: A	Q 2: D	Q 3: C	Q 4: B	Q 5: D
	Q 6: D	Q 7: E	Q 8: E	Q 9: D	Q 10: C
	Q 11: A	Q 12: B			

### Answers and Descriptions

#### Q1 Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

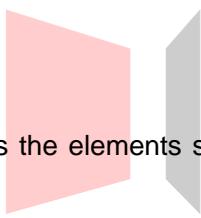
##### Answer

- a. Linear

##### Description

- a. Linear

The linear search algorithm searches the elements sequentially from the beginning to the end of a table → answer



VITEC

##### <http://www.vitec.org.vn>

A hash is a way of using an array-type data structure. Using a hash, you can directly access specific data using a key without accessing recorded data one by one.

- d. Heap

A heap is a kind of binary search tree. It is a perfect binary tree that has a certain size relationship between a parent node and a child node. A heap is different from a binary search tree in that a heap does not have a certain size relationship between brothers.

## Q2

### Answer

d.  $X = a_i$

### Description

Subscript	1	2	3	...	i	...	n	n+1
Value	$a_1$	$a_2$	$a_3$	...	$a_i$	...	$a_n$	X

The step2 through step4 form a loop.

The space  represents the loop condition.

The condition for exit is either "the value is found" or "at-end is reached".

Since "whether at end or not" is examined in the step 5, the space should be "the value is found", i.e. X matches the current element value. → the answer is d

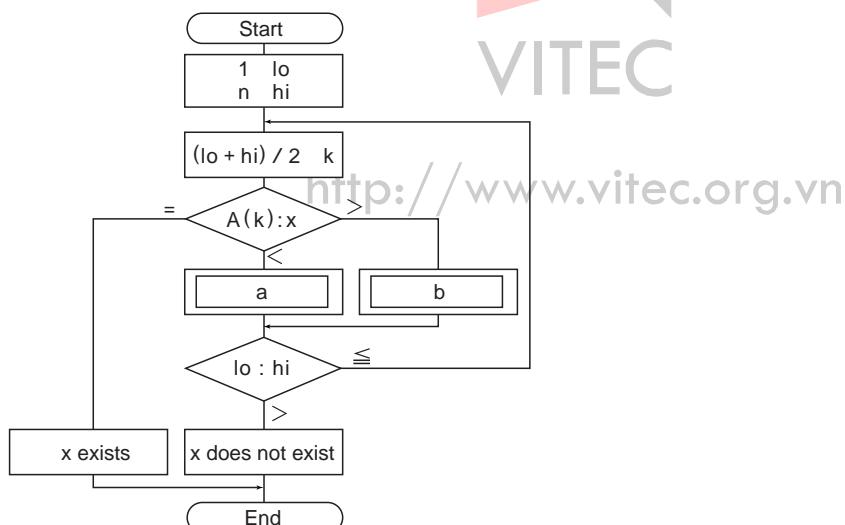
Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

## Q3

### Answer

	a	b
c	$k + 1 \rightarrow lo$	$k - 1 \rightarrow hi$

### Description



A binary search method is the method of narrowing down target data while successively dividing a search area into two parts.

1) space a

If  $A(k) < X$ , further divide the higher half area into two parts, therefore " $k+1 \rightarrow lo$ ".

2) space b

If  $A(k) > X$ , further divide the lower half area into two parts, therefore  $k-1 \rightarrow hi$

Therefore, the answer is c.

## Q4

### Answer

b. 11

### Description

In binary search, the maximum and average number of times a comparison is made: If the number of elements is N,

-Average number of times is  $\lceil \log_2 N \rceil$

-Maximum number of times is  $\lceil \log_2 N \rceil + 1$

( $\lceil \cdot \rceil$  is a Gaussian symbol and decimal numbers of the value shown in this symbol are truncated.)

Average number of times "k" is

$$1 \leq 2000/2^k < 2$$

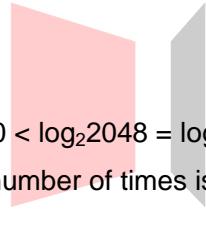
$$2^k \leq 2000/2^k < 2^{k+1}$$

Here,

$$\log_2 1024 = \log_2(2)^{10} = 10 \leq \log_2 2000 < \log_2 2048 = \log_2(2)^{11} = 11$$

Therefore k = 10 and the Maximum number of times is k+1 = 11

→ the answer is b.



VITEC

## Q5

<http://www.vitec.org.vn>

### Answer

- d. If the number of data is 10 or smaller, the average number of comparison times that the linear search method requires, is smaller than that of the binary search method.

### Description

In binary search, the maximum and number of times a comparison is made: If the number of elements is N,

-Average number of times is  $\lceil \log_2 N \rceil$

-Maximum number of times is  $\lceil \log_2 N \rceil + 1$

In linear search, the maximum number of times a comparison is made: If the number of elements is N,

-Average number of times is  $N/2$

-Maximum number of times is  $N$

- a. To use the binary search method, data must be sorted.  
 a is correct.
- b. To search 100 data using the binary search method, the maximum number of comparison times that is needed to find the target data is 7.

if  $N=100$ , Average number of times for binary search is

$$\log_2 100 < \log_2 128 = \log_2(2)^7$$

Therefore Average is 6 and Maximum is 7 → b is also correct.

- c. If the linear search method is used, the number of comparison times does not necessarily decrease even if the data is already sorted.  
 C is also correct.
- d. If the number of data is 10 or smaller, the average number of comparison times that the linear search method requires, is smaller than that of the binary search method.

if  $N=10$ ,  
 Average number of times for linear search is  $10/2 = 5$

Average number of times for binary search is  $\log_2 10 < \log_2(2)^4 = 4$

Therefore d is incorrect → Answer

- e. If the number of data is increased from 100 to 1,000, the number of comparison times increases to 10 times as large as when the linear search method is used. Using the binary search method, the number increases twice or less.

Average number of times for binary search for  $N=100$  and  $N=1000$  is as follows. Therefore e. is also correct.

If  $N=100$ ,  $\log_2 100 < \log_2 128 = \log_2(2)^7$

If  $N=1000$ ,  $\log_2 1000 < \log_2 1024 = \log_2(2)^{10}$

## Q6

### Answer

	a	b	c	d
D	ascending	sort	external sorting	merge

### Description

1) a, b

“Arranging data in the order of small- to large-value queues” means “ascending sort”.

2) c

If a target data sequence is in an auxiliary storage, this operation is called “external sorting”.

c.f. Internal sorting means sorting data in a main memory unit. External sorting means sorting data stored on a magnetic disk and other auxiliary storage unit.

3) d

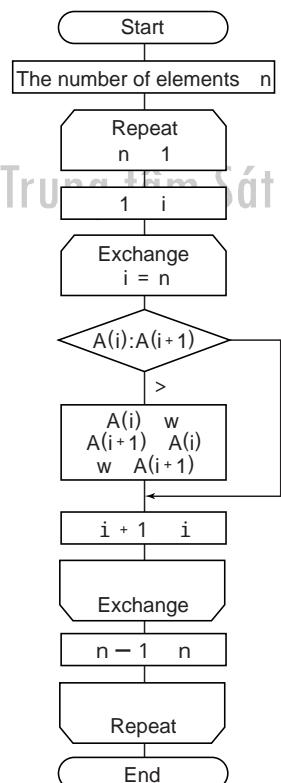
Integrating two or more files sorted in certain order into one file is called “merging”

## Q7

### Answer

e. Bubble sort

### Description



Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

In the above flowchart,  $A(i)$  and  $A(i+1)$  are compared, and if  $A(i) > A(i+1)$ , they are swapped. i.e. a pair of elements next to each other in the sequence is compared and if sequence is wrong, swap occurs.

This describes the bubble sort. → the answer is e.

## Q8

### Answer

e. 10,000

### Description

In the bubble sort mechanism, a pair of elements next to each other in the sequence is compared and if sequence is wrong, swap occurs.

Its computational complexity is  $n^2$  (proportional to a square of n, n: number of data)

If number of data becomes 100 times bigger than earlier (previously 1000 and now 100,000), time for computation will be  $(100)^2 = 10,000$  times longer than earlier.

→ the answer is e. 10,000.

## Q9

### Answer

- d. Heap sort is a method of sorting data by representing an unsorted area as a subtree, fetching the maximum or minimum value from the unsorted area, moving the maximum or minimum value to a sorted area and repeating this routine to narrow down unsorted areas.

### Description

- a. Quick sort is a method of sorting data in sub-sequences consisting of data items fetched at an interval and sorting smaller sub-sequences consisting of data items fetched at a smaller interval.

This describes “Shell sort method”, in which two items of data located away from each other at a certain interval are picked out of a data sequence.

- b. Shell sort is a method of sorting data by comparing a pair of adjacent elements and exchanging them if the second element is larger than the first.

This describes “Bubble sort method” where comparison of a pair of adjacent elements takes place.

- c. Bubble sort is a method of sorting data by setting a median reference value, allocating elements with values larger than the reference value in one section and those with values smaller than the reference value in the other section and repeating this routine in each individual section.

This describes “Quick sort method”.

- d. Heap sort is a method of sorting data by representing an unsorted area as a subtree, fetching the maximum or minimum value from the unsorted area, moving the maximum or minimum value to a sorted area and repeating this routine to narrow down unsorted areas.

This is a correct description of the Heap sort method. → answer

## Q10

### Answer

- c. Data is divided into one group of data smaller than a reference value and the other group of data larger than a reference value. In each group, a new reference value is then selected

and data is likewise divided into two groups based on the reference value. This routine is repeatedly executed.

### Description

In this question, an appropriate description of quick sort is to be found.

The quick sort method was designed by Hoare. It is presently the fastest sort method using the recursive call method.

A reference value (pivot or pivotal value) is chosen from the data to be sorted. A median value or an intermediate value of three elements (right, center and left elements) is usually used. Then Data items smaller than the pivot are moved to the left of the pivot while data items larger than the pivot are moved to the right of it. All the data items are thus divided into two sections (division). A pivot is chosen from each section of data items so that the section is further divided into two sections.

This dividing operation is repeatedly performed until only one element remains.

→ c. is an appropriate description of the quick sort method → answer

- a. Compare and exchange are executed for two data away from one another at a certain distance. This distance is gradually and continuously narrowed to sort all data.
- a. describes the shell sort method.
- b. The first minimum value is found in data. The second minimum value is found in data in which the first minimum value is not included. This routine is repeatedly executed.
- b. describes the basic selection method.
- d. Adjacent data are repeatedly compared and exchanged to allow smaller data to move to the end of a data array.
- d. describes the bubble sort method.

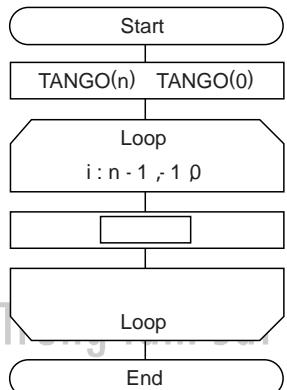
<http://www.vitec.org.vn>

## Q11

### Answer

- TANGO (i) → TANGO (i+1)

### Description



Note: Names of variables  
(initial value, increment and termination value)  
are shown as loop conditions.

Hàng chương Công nghệ thông tin và Hỗ trợ đào tạo

This algorithm consists of

- 1) TANGO(n) → TANGO(0)
- 2) A loop, index "i" value changing from n-1 to 0 decrementing by 1

Take a look at an example of applying the above.

Assume that n=5 and the array named "TANGO" is initially as follows. i.e. the words "FE", "SW", "JITEC", "JIPDEC" and "METI" are in stored in TANGO(1), TANGO(2), TANGO(3), TANGO(4) and TANGO(5) respectively.

<http://www.vitec.org.vn>

	FE	SW	JITEC	JIPDEC	METI
TANGO(0)	TANGO(1)	TANGO(2)	TANGO(3)	TANGO(4)	TANGO(5)

After applying this algorithm, the word in TANGO(n) is supposed to be stored in TANGO(1), then the remaining words are supposed to be shifted to right.

METI	METI	FE	SW	JITEC	JIPDEC
TANGO(0)	TANGO(1)	TANGO(2)	TANGO(3)	TANGO(4)	TANGO(5)

After doing "TANGO(5)→TANGO(0)", the following should be done in the loop (whose index "i" value changing from n-1 to 0 decrementing by 1)

TANGO(4) → TANGO(5)

TANGO(3) → TANGO(4)

:

TANGO(0) → TANGO(1)

This can be described using index "I" as TANGO(i) → TANGO(i+1).

Therefore the answer is a.

- b. TANGO (i) → TANGO (n-i)
- c. TANGO (i+1) → TANGO (n-i)
- d. TANGO (n-i) → TANGO (i)

b,c,d are incorrect.

## Q12

### Answer

b. As seen from the geometrical viewpoint, the method is to obtain an approximate solution by using a tangential line of  $y = f(x)$ .

### Description

Algorithm of Newton's method is described as follows. As shown below, It obtains a solution by using a tangential line  $y=f(x)$ . Therefore the answer is b.

#### Step 1

A tangential line  $y = f(x)$  at point  $p_1$  of the coordinate  $x_1, y_1$  is drawn and point  $x_2$  where the tangential line intersects x-axis is obtained.

#### Step 2

A tangential line  $y = f(x)$  at point  $p_2$  of the coordinate  $x_2, y_2$  is likewise drawn and point  $x_1$  where the tangential line intersects x-axis is obtained. As this step is repeatedly executed the tangential line moves closer to a solution.

#### Step 3

The difference between adjacent approximate values obtained in step 2 is compared with a predetermined convergence judgment value precision. Steps 1 and 2 are repeatedly executed until this difference becomes smaller than the predetermined convergence judgment value.

The equation for a tangential line at point  $p_1$  is  $y-f(x_1) = f'(x_1)(x-x_1)$ , point  $x_2$  where a tangential line intersects x-axis can be obtained using the following equation:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (i = 0, 1, 2, \dots)$$

a. Although a function,  $f(x)$ , cannot be differentiated, an approximate solution can be obtained.  
This is incorrect because this method uses differential as explained above.

c. Two different initial values must be provided.  
This is also incorrect because one solution can be obtained from one initial value.

d. Whatever initial values are provided, an approximate value can always be obtained.  
In case  $f'(x_n) = 0$ , convergence will never take place, thus no approximate value can be obtained.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

---

## Answers for No.1 Chapter4 (Hardware)

### Answer list

#### Answers

Q 1: D	Q 2: A	Q 3: C	Q 4: A	Q 5: A
Q 6: D	Q 7: D	Q 8: D	Q 9: D	Q 10: A
Q 11: C	Q 12: B	Q 13: D	Q 14: C	Q 15: B
Q 16: D				

### Answers and Descriptions

#### Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo Q1

#### Answer

	a	b	c
D	Control unit	Main storage unit	Arithmetic unit

#### Description

b is main storage unit because data from the input device goes to this unit.

a is control unit because it performs controls over other units.

c is arithmetic unit because it exchanges data between b, i.e. main storage unit.

### Q2

<http://www.vitec.org.vn>

#### Answer

- A. DRAM represents 1 bit depending on whether the capacitor is charged or not. It is commonly used as a main storage unit.

#### Description

A. Answer

B. This describes ROM.

- (B. Data is written at the time it is manufactured. It is used as a microprogramming storage memory.)

C. This explains EPROM (Erasable and Programmable ROM).

- (C. Data can be written using a special device and erased with ultraviolet light.)

D. This is for SRAM.

- (D. It is composed of flip flops. The speed is high but the manufacturing cost is high as well. It is used in cache memories, etc.)

## Q3

### Answer

C. 110

### Description

Since operand address is 100 and the index register value is 10,  
effective address =  $100+10=110 \rightarrow$  answer is C

## Q4

### Answer

	P	Q	R
A	0	1	0

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

### Description

In the circuit diagram, process values from left to right direction

#### 1) AND gate

A and B are input to the AND gate

$A=1, B=0$  therefore  $P=0$

#### 2) OR gate

P and C are input to the OR gate

$P=0, C=1$  therefore  $Q=1$

#### 3) NOT gate

Q is input to the NOT gate

$Q=1$  therefore  $R=0$

from the above, the combination of P,Q,R is 0,1,0

$\rightarrow$  answer is A



VITEC

<http://www.vitec.org.vn>

## Q5

### Answer

	C	S
A	$A \cdot B$	$(A \cdot \bar{B}) + (\bar{A} \cdot B)$

**Description**

A	B	Sum of A and B	
		C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

1) C becomes "1" only when both A and B are "1".

Therefore C represents logical product of A and B.

$$\text{i.e. } C = A \cdot B$$

2) S becomes "1" only when A and B are different. It becomes "0" if A and B are the same.

Therefore S represents exclusive logical sum of A and B.

$$\text{i.e. } S = (A \cdot \bar{B}) + (\bar{A} \cdot B)$$

## Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo Q6

**Answer**

D. 4

**Description**

MIPS value represents the number of instructions per second in unit of million ( $10^6$ )

1) Compute the average instruction execution time

Average instruction execution time

$$= (0.1 \times 0.4) + (0.2 \times 0.4) + (0.5 \times 0.3)$$

$$= 0.04 + 0.08 + 0.15 = 0.27 \text{ (microseconds)}$$

2) Translate the above (average execution time for one instruction) into number of instructions executable per second

Number of instructions executable per second

$$= 1 / (0.27 \times 10^{-6})$$

$$= 1 / 0.27 \times 10^6$$

$$4 \times 10^6$$

-->answer is D

## **Q7**

### **Answer**

D. 320

### **Description**

Add CPI of each instruction in the instruction sequence

sequence of execution is a->a->b->a->c->d

CPI in total is  $6+6+2+6+4+8=32$ CPI

Since 1 clock cycle time is 10 nano seconds,

Total execution time is  $32 \times 10 = 320$  (nano seconds)

## **T8** Ng tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

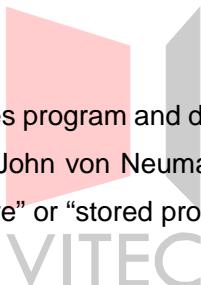
### **Answer**

D. Stored program method

### **Description**

In this question, the method that stores program and data together in memory is to be found. The above-mentioned method is by John von Neumann, at Institute of Advanced Studies, thus called “von Neumann architecture” or “stored program method.”

→ The answer is D.



## **Q9**

<http://www.vitec.org.vn>

### **Answer**

D. 100

### **Description**

Hard disk capacity calculation

Compute in the following sequence, because the unit of storing data in a hard disk is block.

1) Block length

=record length X block factor + inter-block gap length =  $200 \times 20 + 500 = 4500$ (bytes)

2) Number of blocks per track

= track length / block length =  $25200 / 4500 = 5.6$       5

(Convert to integer because a block cannot span over multiple tracks)

3) Total number of blocks

= total number of records / block factor =  $10000/20 = 500$

#### 4) Necessary number of tracks

= total number of blocks / number of blocks per track =  $500/5=100$

## Q10

### Answer

	a	b	c	d
A	Seek time	Search time	Latency	Data transfer time

### Description

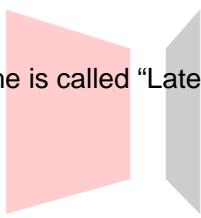
Definition of hard disk access time

Magnetic disk access is performed as follows

- 1) Magnetic head moves to the destination track ("Seek time")
- 2) Platters are rotated until the destination data comes to the head position ("Search time")
- 3) Data transfer is started. (The duration between transfer start and transfer completion is called "Data transfer time")

The sum of Seek time and Search time is called "Latency"

Therefore, the answer is A.



## Q11

### Answer

C. 32

<http://www.vitec.org.vn>

### Description

Average access time

= average seek time + average search time + data transfer time

Average seek time is known to be 20(milliseconds), other two need to be calculated

- 1) Average rotation waiting time

= rotation speed / 2

Rotation speed is

3,000(rotations) per minute

Required time per rotation is

required time per rotation

$$\begin{aligned}
 &= 1 / \text{rotation speed} \\
 &= 60 \times 10^3 / 3,000 \text{ (milliseconds)} \\
 &= 20 \text{ (milliseconds)}
 \end{aligned}$$

Therefore

$$\text{average rotation waiting time} = 20 \div 2 = 10 \text{ (milliseconds)}$$

2) Data transfer time = data length / data transfer speed

1 track data is transferred by a single rotation 20 (milliseconds),

Data transfer speed

$$\begin{aligned}
 &= \text{capacity per track} / \text{time for 1 disk rotation} \\
 &= 20,000(\text{bytes}) / 20(\text{milliseconds}) \text{ (in same units)} \\
 &= 1,000 \text{ (bytes/milliseconds)}
 \end{aligned}$$

Therefore  
 $\text{data transfer time} = 2,000 \div 1,000 = 2 \text{ (milliseconds)}$

3) Average access time

$$= 10 + 20 + 2 = 32 \text{ (milliseconds)}$$

--> answer is C



## Q12

### Answer

- B. In the magneto optical disk, which is one of the rewritable storage media, data is recorded by changing the medium magnetization direction.

### Description

In this question, a correct description of optical disk characteristics is to be found.

The main characteristic of the magneto optical disk is that the data reading and recording methods differ. To be more specific, data writing is performed using a magnet while data reading is performed using a beam. → B is correct.

## Q13

### Answer

- D. Identical data is recorded simultaneously in separate disks.

### Description

A incorrect description

- (A. By giving a mirror-like finish to the disk surface the resistance at the time the disk rotates is reduced.)

B RAID5 distributes data blocks and parity blocks into multiple disks

- (B. The data block and the parity block are stripped and stored across multiple disks.)

C RAID3 uses parity disk

- (C. Besides the disks that record the data, another disk for parity recording is used.)

D correct (describes RAID1)

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

## Q14

### Answer

	Hard disk, CD-ROM	Modem	Keyboard
C	SCSI	RS-232C	USB

### Description

Possible selections for listed devices from the given options

Modem : RS-232C or USB

Keyboard : USB

Hard disk, CD-ROM : ATA/ATAPI-4 or SCSI or USB

Therefore, USB should be for keyboard, RS-232C for modem, and SCSI for Hard disk, CD-ROM. → The answer is C

## Q15

### Answer

- B. 310

### Description

An image with 480 x 640 dots and in 256 colors

256 colors require 8 bits or 1 byte.

Therefore, the image file size is

$$480 \times 640 \times 1 = 327,200 \text{ (bytes)} = \text{approx. } 310 \text{ (KB)}$$

## **Q16**

### **Answer**

- D. Thermal transfer printer

### **Description**

A thermal transfer printer uses a heating element to melt the ink of the ink ribbon and is capable of printing on normal paper.

- B. Thermal printer

This uses a heating element but cannot print on normal paper.

- C. Dot impact printer

This does not “melt” the ink of the ink ribbon.

- A. Ink-jet printer

- E. Laser printer

The above 2 does not use an ink ribbon.



<http://www.vitec.org.vn>

# Index

[1]		cell 2-8, 2-10 chain method 2-19 Character modeling 5-8 character string compression 3-35 character string processing 3-1 character string search 3-32 character type 2-3, 2-7 child 2-13 code 1-17, 1-18 collation algorithm 3-62 complement 1-21, 1-22 computational complexity 3-16 Controlling activity 5-16	FIFO 2-12 figure drawing 3-44 file processing 3-1, 3-37 file updating 3-40 first-in first-out 2-12 fixed point 1-24, 1-25 Flash 5-10 floating point 1-25, 1-26 functional programming 3-3, 3-102
[3]			
3G	4-96		
[A]			
abstract data type	2-7		
Ad Hoc Mode	4-87, 4-88		
AfterEffects	5-11		
approximation algorithm	3-66, 3-67		
arithmetic shift	1-31, 1-32		
array type	2-4, 2-8		
ascending order	3-14, 3-24		
ASCII code	1-37		
Audience assessment	5-5		
Audio editing	5-8		
auxiliary units	1-8		
[B]			
balanced tree	2-16, 2-17		
basic data structure	2-1, 2-2		
basic data type	2-1, 2-3		
basic exchange method	3-14, 3-15		
basic insertion method	3-18, 3-19		
basic selection method	3-17, 3-18		
Basic Service Set (BSS)	4-87		
BCD code	1-17		
bias	1-29		
bi-directional list	2-10		
binary numbers	1-2, 1-3		
binary search method	3-9, 3-11		
binary search tree	2-17		
binary tree	2-16		
bisection method	3-53, 3-54		
bit	1-3, 1-4,		
Bluetooth	4-82, 4-85, 4-90		
borrow	1-10, 1-11		
Boyer-Moore method	3-33		
branch	2-14		
breadth-first search method	3-50		
B-tree	2-17		
bubble sort	3-14, 3-20		
byte	1-4, 1-5		
[C]			
cancellation	1-35		
carry	1-3, 1-6		
[D]			
data structure	2-1, 2-2		
data type	2-2, 2-3		
Data-driven engine (DDE)	5-17		
De Morgan's law	1-43		
decimal arithmetic system	1-18		
Define	5-6		
Define the interactions	5-15		
depth-first search method	3-49		
dequeue	2-12		
descending order	3-11, 3-12		
design steps of a multimedia system	5-4		
Dijkstra search method	3-52		
directed graph	3-49		
Director	5-9		
divide-and-conquer method	3-24		
double precision	1-34		
dynamic programming method	3-117		
[E]			
EBCDIC code	1-18, 1-37		
eight-queen question	3-29, 3-30		
encapsulation	2-7		
enqueue	2-12		
enumeration type	2-3		
error	1-28, 1-33		
Excess 64	1-26		
exclusive OR	1-41, 1-42		
exhaustive search method	3-9, 3-11		
exponent	1-7, 1-8		
Extended Service Set (ESS)	4-88		
external sorting	3-28		
[F]			
Feedback	5-19		
[G]			
gap	3-22		
garbage	2-9		
GPS (Global Positioning System)	4-100		
graph	3-48, 3-49		
greedy algorithm method	3-117		
group control	3-37, 3-39		
[H]			
hash	2-18, 2-19		
heap	2-17		
hexadecimal system	1-5, 1-6		
Hierarchical menus	5-17		
home record	2-19		
[I]			
IEEE 802.11	4-82, 4-86, 4-87		
IEEE 802.11b	4-86, 4-87		
index	2-4		
infinite binary fraction	1-13, 1-14		
information hiding	2-7		
Infrared	4-84, 4-85		
Infrastructure Mode	4-87		
integer type	2-3, 2-7		
International Organization for Standardization	1-37		
ISO	1-37, 1-38		
ISO code	1-37		
[J]			
JIS	1-18, 1-37, 1-38		
JIS 8-bit code	1-37		
JIS code	1-18, 1-37, 1-38		
[K]			
knapsack problem	3-66, 3-67, 3-68		

[L]			
Landscape generators	5-8	perfect binary tree	2-14, 2-17
last-in first-out	2-11	pivot	3-24, 3-25
leaf	2-16	pointer type	2-4
LIFO	2-11	pop	2-11
linear list	2-10	POP	2-11
Linear presentation	5-16	Poser	5-12
linear search	3-6, 3-11, 3-115	precision	5-16, 5-28
linear search method	3-11	Premier	5-11
list structure	2-8, 2-9, 2-13	primarity test problem	3-69
logic programming	3-4, 3-101	probability algorithm	3-65, 3-69
logical operation	1-1, 1-40	probability algorithm with bounded errors	3-69
logical product	1-42	problem-oriented data structure	2-8
logical shift	1-32	procedural programming	3-3
logical sum	1-42, 1-43	proposition	1-1, 1-40, 1-41
logical symbol	1-41, 1-42	proposition logic	1-40
logical type	2-3	push	2-12
loss of information	1-36	PUSH	2-12
[M]		[Q]	
merge sort	3-28, 3-29	queue	2-12
Microbrowser	4-92	quick sort	3-24, 3-25
Mobile based protocols	4-95		
Mobile Operating System	4-95		
Monte Carlo method	3-71		
Multimedia	5-2		
Multimedia authoring	5-8		
multiway tree	2-13		
[N]			
Narrowband Microwave	4-84, 4-85		
N-ary tree	2-13		
Needs Assessment	5-5		
negation	1-41, 1-42, 1-43		
Newton's method	3-54, 3-55		
node	2-14, 2-16		
normalization	1-28, 1-30		
NULL	2-10		
numerical integration	3-56		
[O]			
one-dimensional array	2-5, 2-12		
overflow	1-24, 1-35		
[P]			
packed decimal format	1-18, 1-19		
parallel transfer	4-60		
parent	2-13, 2-14		
partial type	2-3		
[S]			
scatternet	4-85		
Selecting Authoring and delivery System	5-7		
sentinel search method	3-10, 3-11		
sequential method	2-19		
Shaker sort	3-20, 3-21		
Shell sort	3-22, 3-23		
shift	1-28, 1-30, 1-31		
shift JIS code	1-38		
shift operation	1-30, 1-31, 1-32		
shortest path problem	3-49		
simple type	2-3		
Simpson's method	3-56, 3-59		
[T]			
table	2-4		
table search	3-9		
Third generation (3G)			
services	4-96		
three-dimensional array	2-6		
trapezoidal rule	3-56, 3-57		
tree structure	2-14, 2-15		
truth table	1-40, 1-42		
two-dimensional array	2-4, 2-5		
Types of multimedia applications	5-3		
[U]			
underflow	1-35		
undirected graph	3-49		
uni-directional list	2-10		
unpacked decimal format	1-18		
[V]			
validity	3-115, 3-116		
Video authoring	5-8		
[W]			
WCDMA	4-96		
weighted graph	3-49		
WEP (Wired Equivalent Privacy)	4-89		
Wireless LAN	4-82, 4-85, 4-91		
word	1-5, 1-24, 1-25		
[X]			
XHTML	4-93, 4-94, 4-95		
[Z]			
zoned decimal format	1-18		